DS4400: Machine Learning and Data Mining I

Spring 2021

Project Report

Project Title: Categorizing Unscripted Conversations

TA: Saurabh Parkar

Team Members: Matthew Hosking

Problem Description

For our project we will be delving into the world of unscripted conversation categorization. We are looking to create a model that can accurately categorize different snippets of a conversation into a selection of categories: conversations, interviews, meetings, panels, press conferences, question and answer sessions, seminar discussions, service encounters, working group discussions, and workshop discussions. The categories are based on the categories found in the Vienna Oxford International Corpus of English. This work could be a potential foundation for further exploration into conversational agents that could properly converse in certain genres of dialogue. Agents that can fit into certain kinds of conversational genres will be far easier to work with as teammates in those settings. An agent that speaks appropriately to a press conference is not necessarily applicable to a working group discussion. Our work is intended to help differentiate more specifically between these different categories in order to provide a basis for novel conversational agents.

We are looking to use the Vienna Oxford International Corpus of English (VOICE) which contains over a million words and many individual sessions of conversations. The transcriptions and recordings are already categorized and labelled into categories so we will only have to extract features and train the model. The feature dimensionality is more complex since we are looking at training a language model. We will examine word groupings, specific vocabulary, and overall word importance. Based on recommendations in the Natural Language Processing space, we will examine both a bag of words feature set and a TF-IDF word importance feature set.

References

(1) VOICE. 2013. The Vienna-Oxford International Corpus of English (version 2.0 XML). Director: Barbara Seidlhofer; Researchers: Angelika Breiteneder, Theresa Klimpfinger, Stefan Majewski, Ruth Osimk-Teasdale, Marie-Luise Pitzl, Michael Radeka.

Dataset and Exploratory Data Analysis

The Vienna-Oxford International Corpus of English is a collection of recorded and transcribed unscripted conversations across 49 different first language backgrounds. All conversations take place in English and were recorded over six years between July 2001 and November 2007. The conversations were transcribed so as to be as truthful to the recordings as possible and readable by computers. As a result the dataset features numerous vocal artifacts and tics such as “er,” “um,” and “mhm.” There are two versions of the transcriptions provided in the dataset: XML and txt. The XML transcriptions feature transcribed non-vocal events such as pauses alongside indications of intonation, emphasis, and lengthening. The txt transcriptions only contain the spoken elements of the conversations. For this project, we used the txt transcriptions to focus on highlighting the unique vocabulary and formality of individual categories of conversation and interaction.

The 151 transcriptions are divided into ten categories of speech event types: conversations, interviews, meetings, panels, press conferences, question and answer sessions, seminar discussions, service encounters, working group discussions, and workshop discussions with a distribution that can be seen in table 1.

Category	Transcriptions	Words
con (conversation)	36	158075
int (interview)	16	36362
mtg (meeting)	20	273458
pan (panel)	10	92719
prc (press conference)	5	17588
qas (question-answer session)	10	27541
sed (seminar discussion)	6	63625
sve (service encounter)	11	14894
wgd (working group discussion)	19	181055
wsd (workshop discussion)	18	157870

Table 1: Speech Event Categorical Breakdown

In order to train and test on snippets of these transcribed speech events, we need to deconstruct each transcription into similarly sized collections. We separated sections every forty lines so that we maintain a rich set of information in each section. In figure 1 can be seen the entry breakdown for each category of speech event.

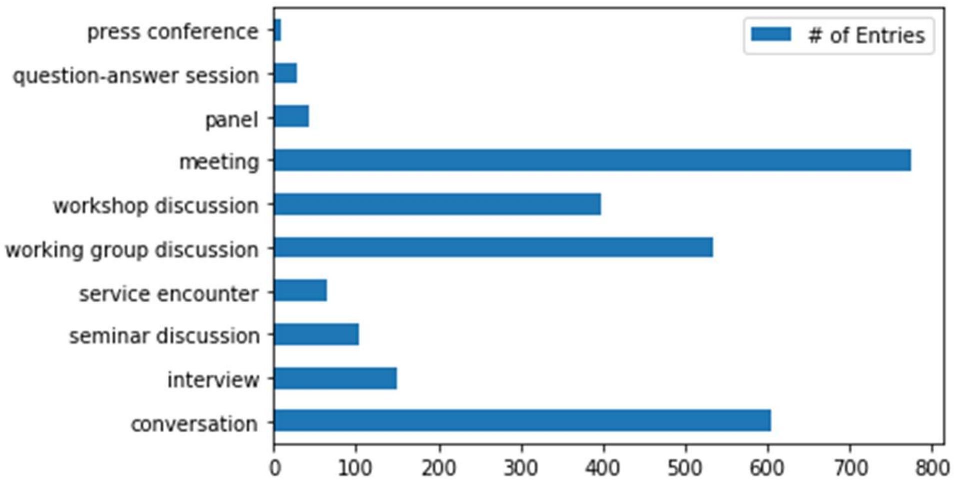


Fig. 1: Categorical Distribution Across Transcriptions

As can be seen, there remains a significant underrepresentation of press conferences, question-answer sessions, and panels. There is an overrepresentation of meetings, conversations, and discussions. Ideally, the vocabulary and relevant features are unique enough to each category and classification will be accurate.

Approach and Methodology

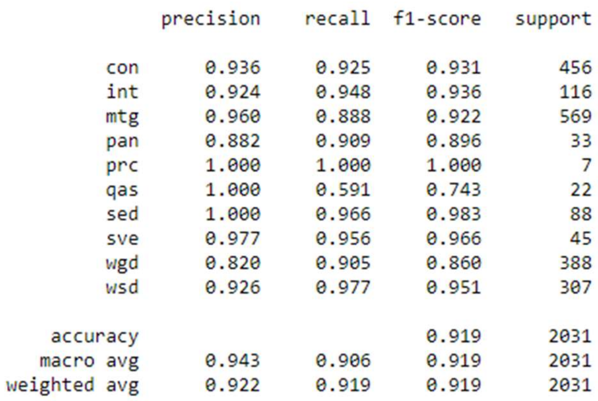
For feature selection, we examined two commonly used NLP approaches: bag of words and TF-IDF (term frequency-inverse document frequency) but settled on TF-IDF as our central feature selection methodology. We tokenized the dataset and cleaned it by removing stop words and replacing incomplete words (tokens like “approchi-“ or “-uable”) that signified interruptions with a universal token. From those tokens we calculated the most common tokens, compiled word counts for each token, and evaluated the TF-IDF score for each token.

With TF-IDF scores for each token in the dataset, we began training and evaluating different machine learning models. The models we looked at were Multinomial Naïve Bayes, Logistic Regression, Support Vector Machines (Linear and Sigmoid), Random Forest Classification, and AdaBoost.

Multinomial Naïve Bayes

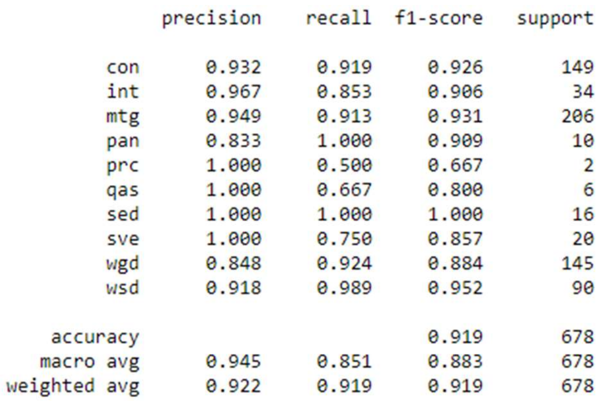
For Multinomial Naïve Bayes we used the word counts as our feature set since it was most applicable to the Bayesian model. This model served as a baseline for the overall classification task as it is a relatively simple model without much room for tuning. Results and confusion matrices can be found below.

Training Data

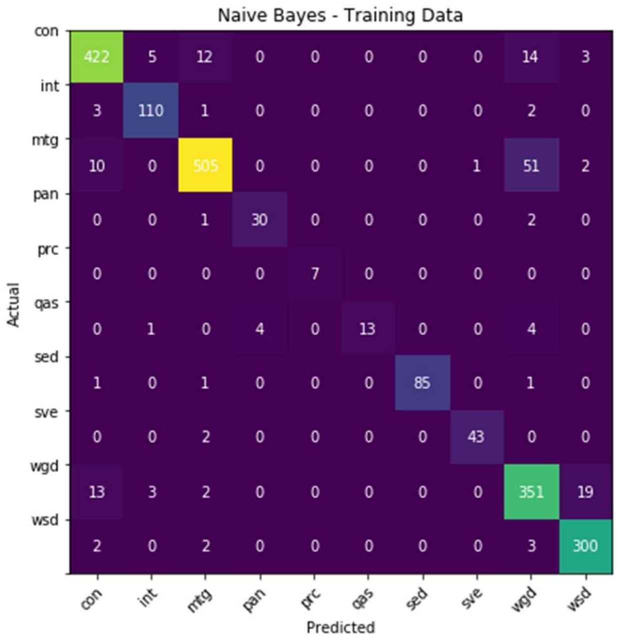


```
              precision    recall  f1-score   support

         con      0.936     0.925     0.931       456
         int      0.924     0.948     0.936       116
         mtg      0.960     0.888     0.922       569
         pan      0.882     0.909     0.896        33
         prc      1.000     1.000     1.000         7
         qas      1.000     0.591     0.743        22
         sed      1.000     0.966     0.983        88
         sve      0.977     0.956     0.966        45
         wgd      0.820     0.905     0.860       388
         wsd      0.926     0.977     0.951       307

    accuracy                          0.919      2031
   macro avg      0.943     0.906     0.919      2031
weighted avg      0.922     0.919     0.919      2031
```

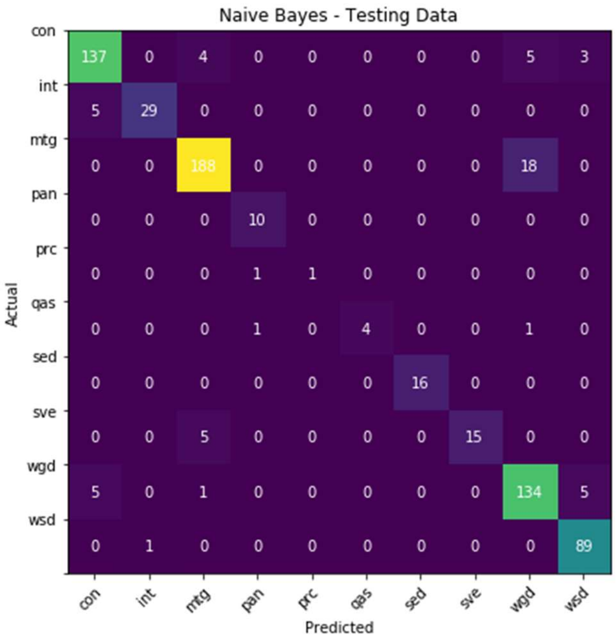
Testing Data



```
              precision    recall  f1-score   support

         con      0.932     0.919     0.926       149
         int      0.967     0.853     0.906        34
         mtg      0.949     0.913     0.931       206
         pan      0.833     1.000     0.909        10
         prc      1.000     0.500     0.667         2
         qas      1.000     0.667     0.800         6
         sed      1.000     1.000     1.000        16
         sve      1.000     0.750     0.857        20
         wgd      0.848     0.924     0.884       145
         wsd      0.918     0.989     0.952        90

    accuracy                          0.919       678
   macro avg      0.945     0.851     0.883       678
weighted avg      0.922     0.919     0.919       678
```



	con	int	mtg	pan	prc	qas	sed	sve	wgd	wsd
con	422	5	12	0	0	0	0	0	14	3
int	3	110	1	0	0	0	0	0	2	0
mtg	10	0	505	0	0	0	0	1	51	2
pan	0	0	1	30	0	0	0	0	2	0
prc	0	0	0	0	7	0	0	0	0	0
qas	0	1	0	4	0	13	0	0	4	0
sed	1	0	1	0	0	0	85	0	1	0
sve	0	0	2	0	0	0	0	43	0	0
wgd	13	3	2	0	0	0	0	0	351	19
wsd	2	0	2	0	0	0	0	0	3	300



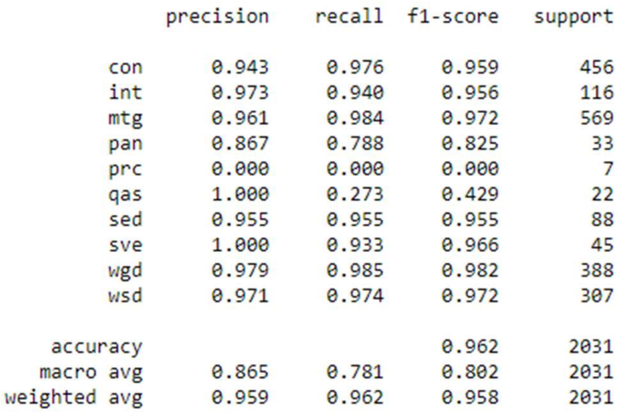
	con	int	mtg	pan	prc	qas	sed	sve	wgd	wsd
con	137	0	4	0	0	0	0	0	5	3
int	5	29	0	0	0	0	0	0	0	0
mtg	0	0	188	0	0	0	0	0	18	0
pan	0	0	0	10	0	0	0	0	0	0
prc	0	0	0	1	1	0	0	0	0	0
qas	0	0	0	1	0	4	0	0	1	0
sed	0	0	0	0	0	0	16	0	0	0
sve	0	0	5	0	0	0	0	15	0	0
wgd	5	0	1	0	0	0	0	0	134	5
wsd	0	1	0	0	0	0	0	0	0	89

As can be seen, the Naïve Bayes model struggles with the underrepresented categories but overall manages a respectable overall accuracy of 0.919 across both training and testing data. There is evidence of overfitting in the underrepresented categories since the metrics between training and testing for press conferences and question-answer sessions greatly vary.

Logistic Regression

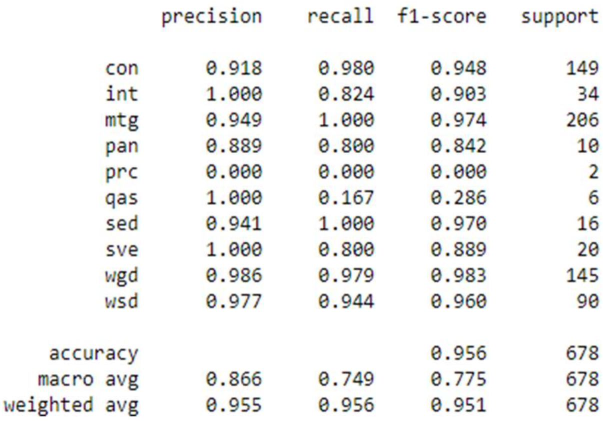
For Logistic Regression, we used the TF-IDF feature set on a standard SKLearn logistic regression algorithm. The model provided excellent first results and further tuning proved unnecessary. Results and confusion matrices can be seen below.

Training Data

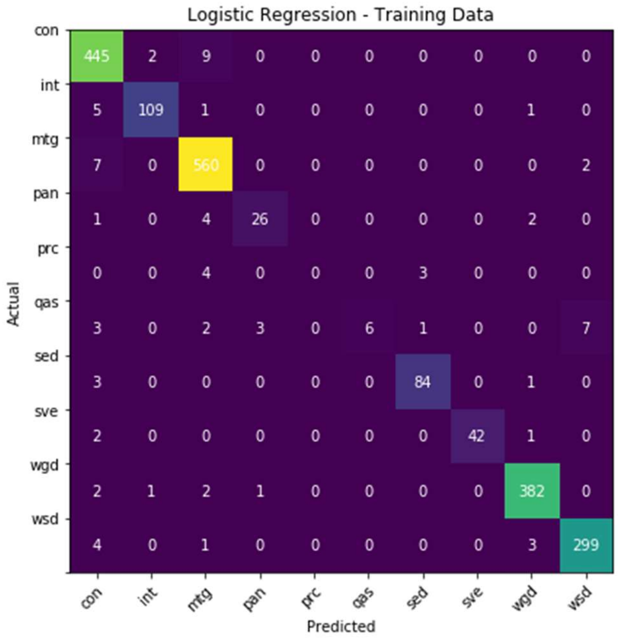


```
              precision    recall  f1-score   support

         con      0.943     0.976     0.959       456
         int      0.973     0.940     0.956       116
         mtg      0.961     0.984     0.972       569
         pan      0.867     0.788     0.825        33
         prc      0.000     0.000     0.000         7
         qas      1.000     0.273     0.429        22
         sed      0.955     0.955     0.955        88
         sve      1.000     0.933     0.966        45
         wgd      0.979     0.985     0.982       388
         wsd      0.971     0.974     0.972       307

    accuracy                          0.962      2031
   macro avg      0.865     0.781     0.802      2031
weighted avg      0.959     0.962     0.958      2031
```

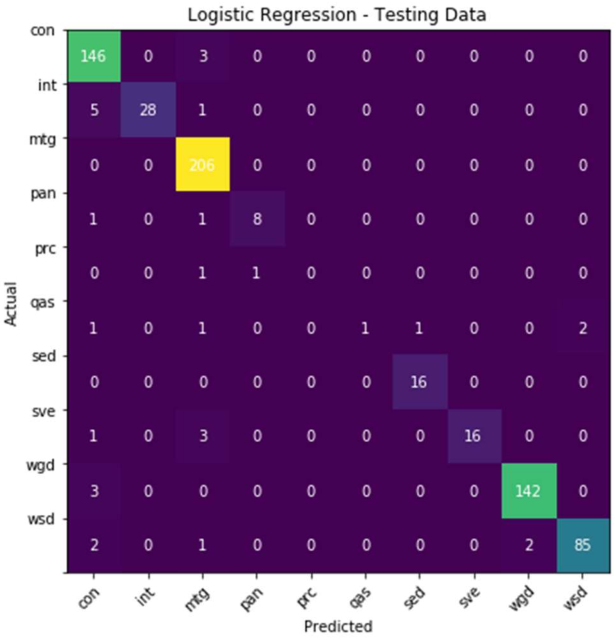
Testing Data



```
              precision    recall  f1-score   support

         con      0.918     0.980     0.948       149
         int      1.000     0.824     0.903        34
         mtg      0.949     1.000     0.974       206
         pan      0.889     0.800     0.842        10
         prc      0.000     0.000     0.000         2
         qas      1.000     0.167     0.286         6
         sed      0.941     1.000     0.970        16
         sve      1.000     0.800     0.889        20
         wgd      0.986     0.979     0.983       145
         wsd      0.977     0.944     0.960        90

    accuracy                          0.956       678
   macro avg      0.866     0.749     0.775       678
weighted avg      0.955     0.956     0.951       678
```



	con	int	mtg	pan	prc	qas	sed	sve	wgd	wsd
con	445	2	9	0	0	0	0	0	0	0
int	5	109	1	0	0	0	0	0	1	0
mtg	7	0	560	0	0	0	0	0	0	2
pan	1	0	4	26	0	0	0	0	2	0
prc	0	0	4	0	0	0	3	0	0	0
qas	3	0	2	3	0	6	1	0	0	7
sed	3	0	0	0	0	0	84	0	1	0
sve	2	0	0	0	0	0	0	42	1	0
wgd	2	1	2	1	0	0	0	0	382	0
wsd	4	0	1	0	0	0	0	0	3	299



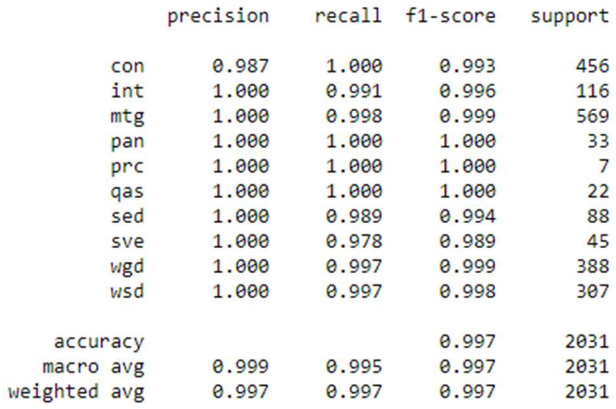
	con	int	mtg	pan	prc	qas	sed	sve	wgd	wsd
con	146	0	3	0	0	0	0	0	0	0
int	5	28	1	0	0	0	0	0	0	0
mtg	0	0	206	0	0	0	0	0	0	0
pan	1	0	1	8	0	0	0	0	0	0
prc	0	0	1	1	0	0	0	0	0	0
qas	1	0	1	0	0	1	1	0	0	2
sed	0	0	0	0	0	0	16	0	0	0
sve	1	0	3	0	0	0	0	16	0	0
wgd	3	0	0	0	0	0	0	0	142	0
wsd	2	0	1	0	0	0	0	0	2	85

While the overall accuracy scores outperformed the Multinomial Naïve Bayes model, there is a clear flaw in the Logistic Regression model. It completely fails to categorize the press conference entries correctly and as a result the macro average metrics suffer. It does manage to provide similar metrics across testing and training data so it’s clear that overfitting is not the issue in this case. Obviously in a high risk situation of categorization it would not be advisable to use this model as there is a significant blind spot.

Linear SVM

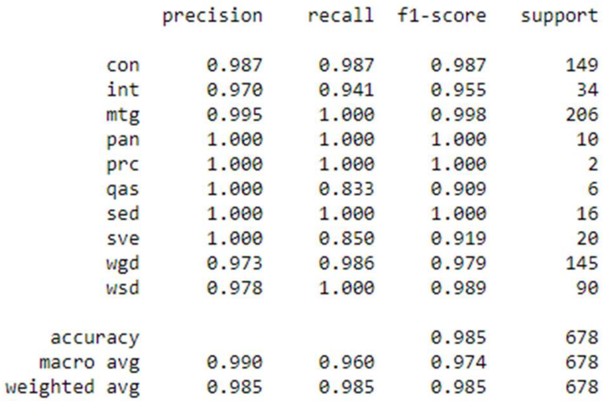
For the linear SVM model, we used the TF-IDF feature set and the default SKLearn settings. These results proved to be incredibly impressive and no further tuning was required. Results and confusion matrices can be found below.

Training Data

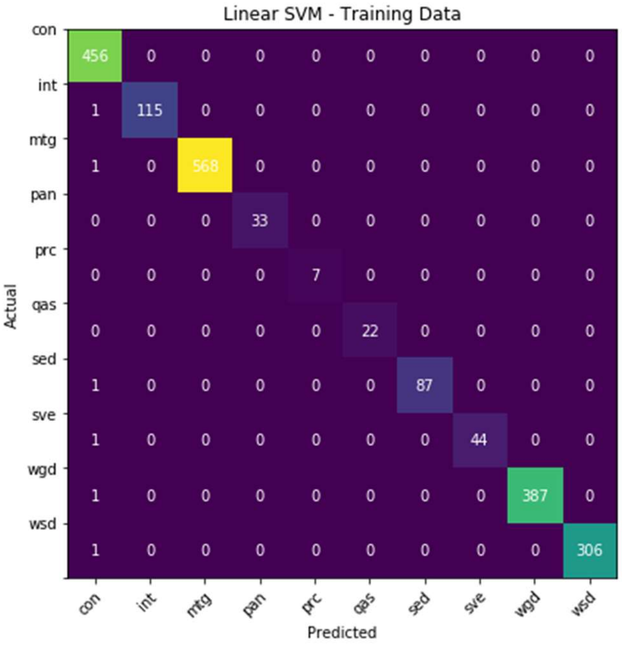


```
              precision    recall  f1-score   support

         con      0.987     1.000     0.993       456
         int      1.000     0.991     0.996       116
         mtg      1.000     0.998     0.999       569
         pan      1.000     1.000     1.000        33
         prc      1.000     1.000     1.000         7
         qas      1.000     1.000     1.000        22
         sed      1.000     0.989     0.994        88
         sve      1.000     0.978     0.989        45
         wgd      1.000     0.997     0.999       388
         wsd      1.000     0.997     0.998       307

    accuracy                          0.997      2031
   macro avg      0.999     0.995     0.997      2031
weighted avg      0.997     0.997     0.997      2031
```

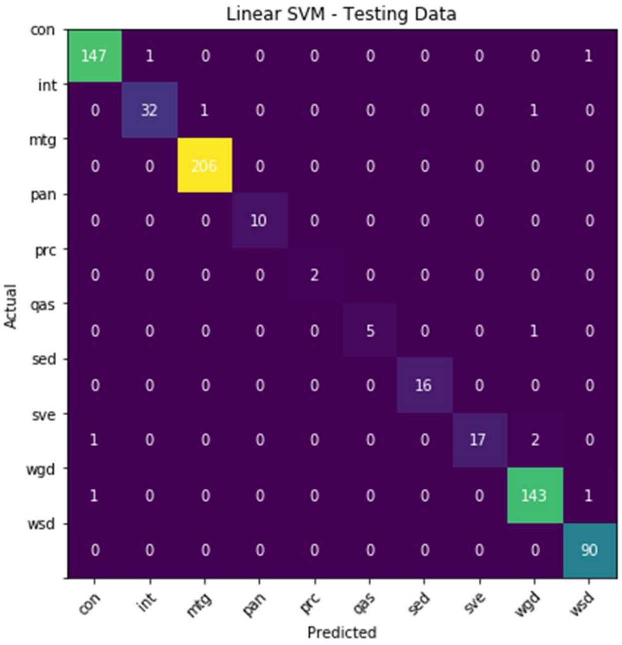
Testing Data



```
              precision    recall  f1-score   support

         con      0.987     0.987     0.987       149
         int      0.970     0.941     0.955        34
         mtg      0.995     1.000     0.998       206
         pan      1.000     1.000     1.000        10
         prc      1.000     1.000     1.000         2
         qas      1.000     0.833     0.909         6
         sed      1.000     1.000     1.000        16
         sve      1.000     0.850     0.919        20
         wgd      0.973     0.986     0.979       145
         wsd      0.978     1.000     0.989        90

    accuracy                          0.985       678
   macro avg      0.990     0.960     0.974       678
weighted avg      0.985     0.985     0.985       678
```


Linear SVM - Training Data

Actual \ Predicted	con	int	mtg	pan	prc	qas	sed	sve	wgd	wsd
con	456	0	0	0	0	0	0	0	0	0
int	1	115	0	0	0	0	0	0	0	0
mtg	1	0	568	0	0	0	0	0	0	0
pan	0	0	0	33	0	0	0	0	0	0
prc	0	0	0	0	7	0	0	0	0	0
qas	0	0	0	0	0	22	0	0	0	0
sed	1	0	0	0	0	0	87	0	0	0
sve	1	0	0	0	0	0	0	44	0	0
wgd	1	0	0	0	0	0	0	0	387	0
wsd	1	0	0	0	0	0	0	0	0	306


Linear SVM - Testing Data

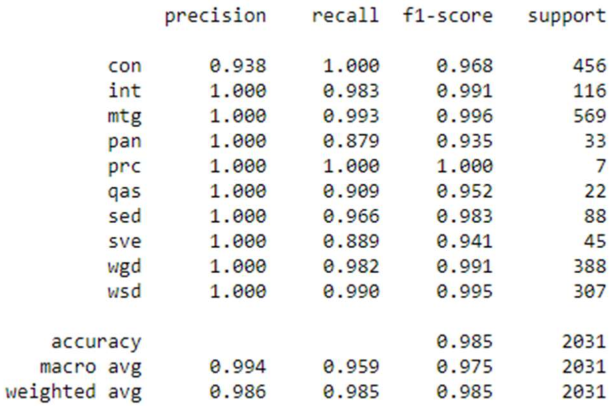
Actual \ Predicted	con	int	mtg	pan	prc	qas	sed	sve	wgd	wsd
con	147	1	0	0	0	0	0	0	0	1
int	0	32	1	0	0	0	0	0	1	0
mtg	0	0	206	0	0	0	0	0	0	0
pan	0	0	0	10	0	0	0	0	0	0
prc	0	0	0	0	2	0	0	0	0	0
qas	0	0	0	0	0	5	0	0	1	0
sed	0	0	0	0	0	0	16	0	0	0
sve	1	0	0	0	0	0	0	17	2	0
wgd	1	0	0	0	0	0	0	0	143	1
wsd	0	0	0	0	0	0	0	0	0	90

As can be seen, the accuracy metrics across all categories outperform all prior models and the underepresented categories are categorized consistently. There is some miscategorization that leans toward the overrepresented categories of conversations and the two types of discussions. A difference here, however, is that the meeting category is just about perfect in comparison to the prior two models.

Sigmoid SVM

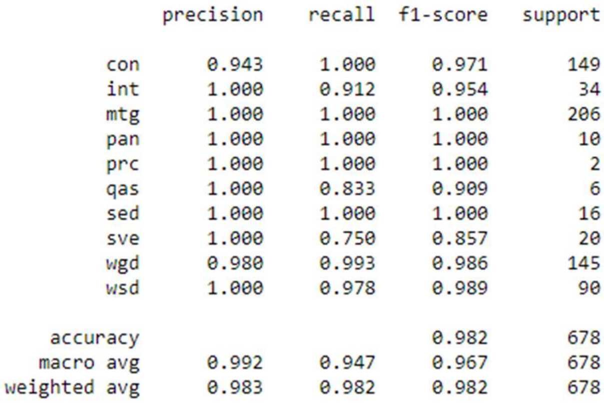
The sigmoid SVM was chosen as the kernel representative after some preliminary tests of each kernel SVM provided in the SKLearn SVM implementation. Results and confusion matrices for the sigmoid model can be found below.

Training Data

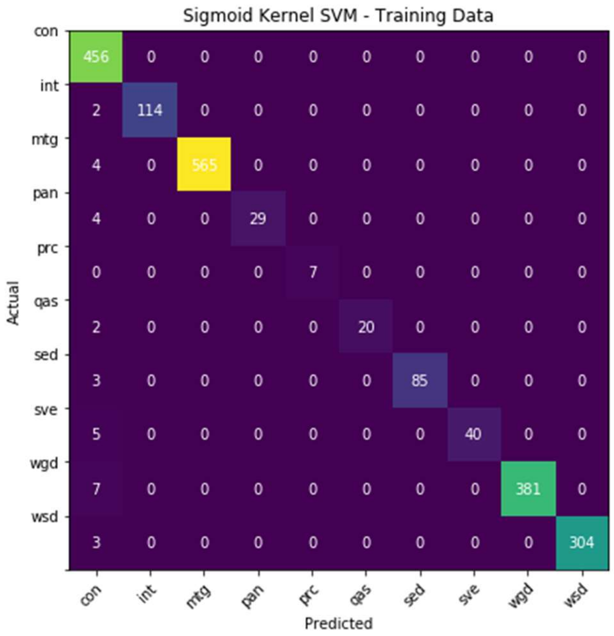


```
              precision    recall  f1-score   support

         con      0.938     1.000     0.968       456
         int      1.000     0.983     0.991       116
         mtg      1.000     0.993     0.996       569
         pan      1.000     0.879     0.935        33
         prc      1.000     1.000     1.000         7
         qas      1.000     0.909     0.952        22
         sed      1.000     0.966     0.983        88
         sve      1.000     0.889     0.941        45
         wgd      1.000     0.982     0.991       388
         wsd      1.000     0.990     0.995       307

    accuracy                          0.985      2031
   macro avg      0.994     0.959     0.975      2031
weighted avg      0.986     0.985     0.985      2031
```

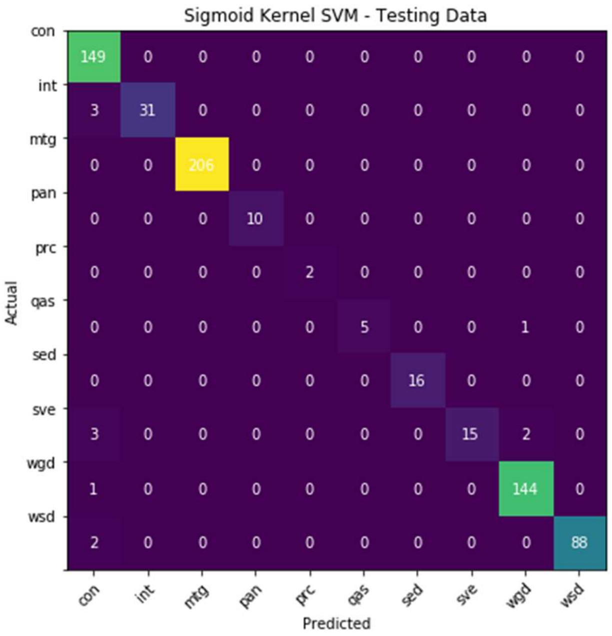
Testing Data



```
              precision    recall  f1-score   support

         con      0.943     1.000     0.971       149
         int      1.000     0.912     0.954        34
         mtg      1.000     1.000     1.000       206
         pan      1.000     1.000     1.000        10
         prc      1.000     1.000     1.000         2
         qas      1.000     0.833     0.909         6
         sed      1.000     1.000     1.000        16
         sve      1.000     0.750     0.857        20
         wgd      0.980     0.993     0.986       145
         wsd      1.000     0.978     0.989        90

    accuracy                          0.982       678
   macro avg      0.992     0.947     0.967       678
weighted avg      0.983     0.982     0.982       678
```



	con	int	mtg	pan	prc	qas	sed	sve	wgd	wsd
con	456	0	0	0	0	0	0	0	0	0
int	2	114	0	0	0	0	0	0	0	0
mtg	4	0	565	0	0	0	0	0	0	0
pan	4	0	0	29	0	0	0	0	0	0
prc	0	0	0	0	7	0	0	0	0	0
qas	2	0	0	0	0	20	0	0	0	0
sed	3	0	0	0	0	0	85	0	0	0
sve	5	0	0	0	0	0	0	40	0	0
wgd	7	0	0	0	0	0	0	0	381	0
wsd	3	0	0	0	0	0	0	0	0	304



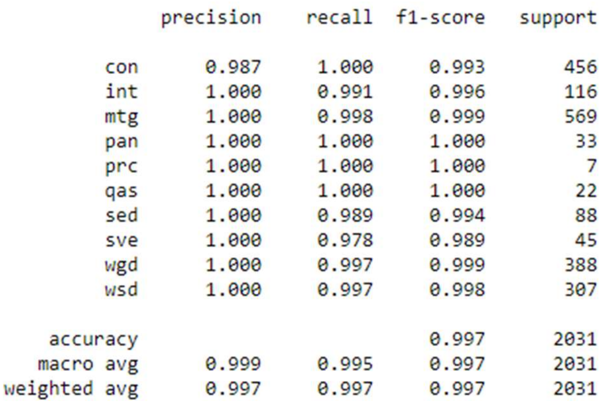
	con	int	mtg	pan	prc	qas	sed	sve	wgd	wsd
con	149	0	0	0	0	0	0	0	0	0
int	3	31	0	0	0	0	0	0	0	0
mtg	0	0	206	0	0	0	0	0	0	0
pan	0	0	0	10	0	0	0	0	0	0
prc	0	0	0	0	2	0	0	0	0	0
qas	0	0	0	0	0	5	0	0	1	0
sed	0	0	0	0	0	0	16	0	0	0
sve	3	0	0	0	0	0	0	15	2	0
wgd	1	0	0	0	0	0	0	0	144	0
wsd	2	0	0	0	0	0	0	0	0	88

As can be seen, there is a similar level of success to the linear SVM results. The precision is better but recall and accuracy suffer in comparison to the linear results. However, the model does outperform both the Logistic Regression and Naïve Bayes results with a consistent categorization across all categories.

Random Forest

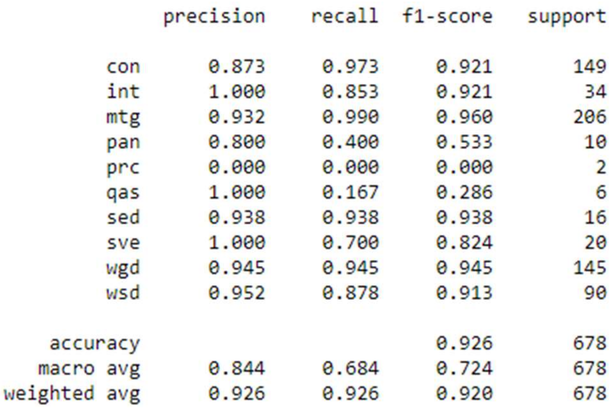
For the Random Forest classifier, the number of estimators was set to 100 and the feature set used was the TF-IDF values. Results did not change as the estimator number was changed so 100 was settled upon. The results and confusion matrices can be seen below.

Training Data

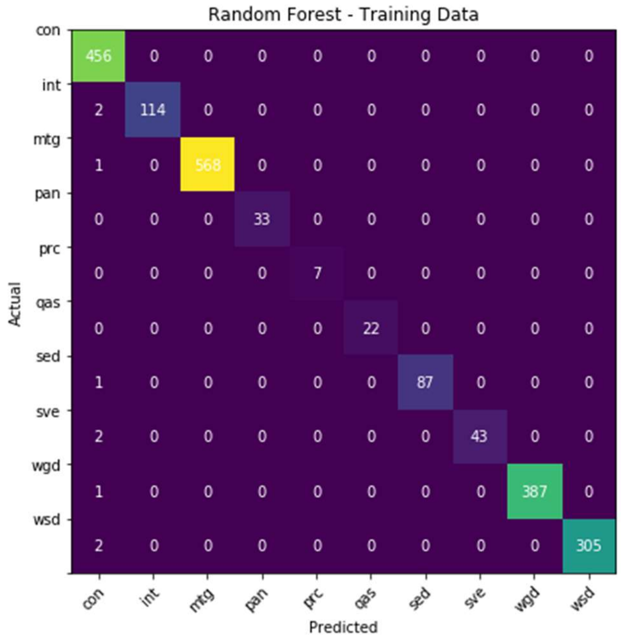


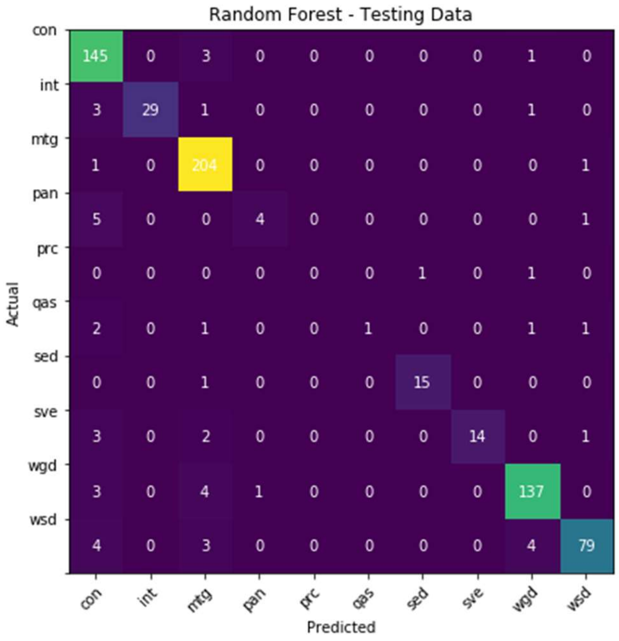
```
              precision    recall  f1-score   support

         con      0.987     1.000     0.993       456
         int      1.000     0.991     0.996       116
         mtg      1.000     0.998     0.999       569
         pan      1.000     1.000     1.000        33
         prc      1.000     1.000     1.000         7
         qas      1.000     1.000     1.000        22
         sed      1.000     0.989     0.994        88
         sve      1.000     0.978     0.989        45
         wgd      1.000     0.997     0.999       388
         wsd      1.000     0.997     0.998       307

    accuracy                          0.997      2031
   macro avg      0.999     0.995     0.997      2031
weighted avg      0.997     0.997     0.997      2031
```

Testing Data



```
              precision    recall  f1-score   support

         con      0.873     0.973     0.921       149
         int      1.000     0.853     0.921        34
         mtg      0.932     0.990     0.960       206
         pan      0.800     0.400     0.533        10
         prc      0.000     0.000     0.000         2
         qas      1.000     0.167     0.286         6
         sed      0.938     0.938     0.938        16
         sve      1.000     0.700     0.824        20
         wgd      0.945     0.945     0.945       145
         wsd      0.952     0.878     0.913        90

    accuracy                          0.926       678
   macro avg      0.844     0.684     0.724       678
weighted avg      0.926     0.926     0.920       678
```



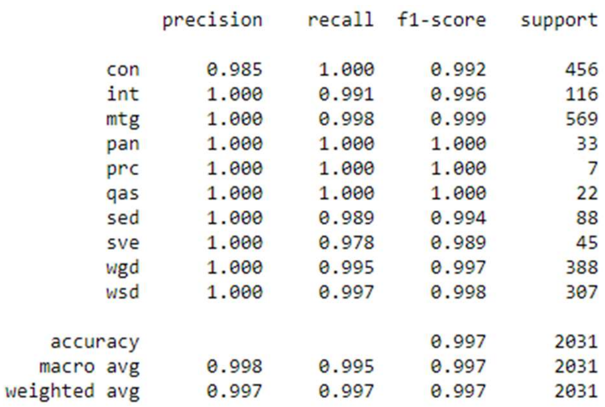


Similar to the first two models, the random forest classifier failed to maintain a high level of recall across all categories and struggled with the underrepresented ones. Accuracy was high but it’s clear that was only due to the high support of the larger categories.

AdaBoost

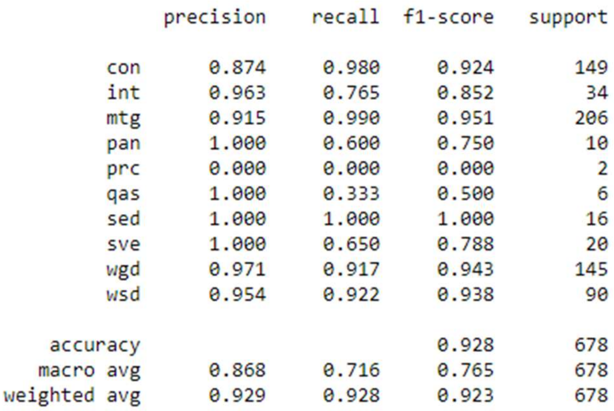
The AdaBoost ensemble learning model was trained with 50 estimators using a base estimator of a decision tree with depth 10. Due to time constraints, further testing of base estimator configurations could not be completed and depth 10 was selected as the best performing model. The results can be seen below along with confusion matrices.

Training Data

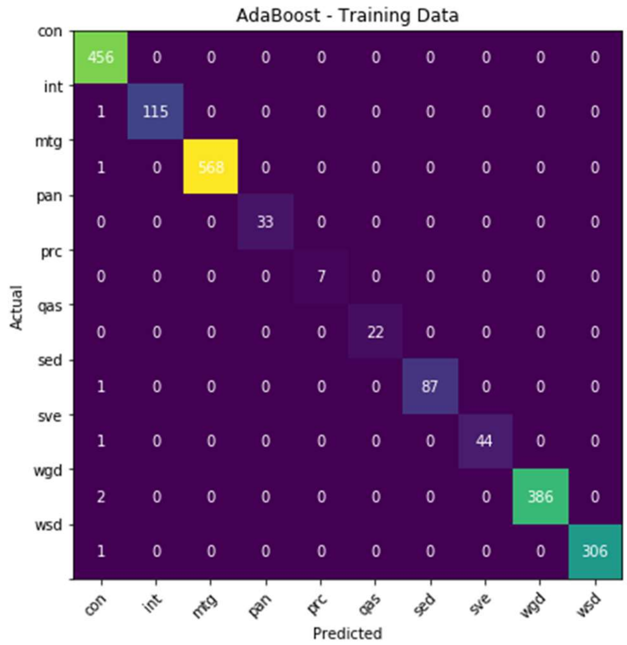


```
              precision    recall  f1-score   support

         con      0.985     1.000     0.992       456
         int      1.000     0.991     0.996       116
         mtg      1.000     0.998     0.999       569
         pan      1.000     1.000     1.000        33
         prc      1.000     1.000     1.000         7
         qas      1.000     1.000     1.000        22
         sed      1.000     0.989     0.994        88
         sve      1.000     0.978     0.989        45
         wgd      1.000     0.995     0.997       388
         wsd      1.000     0.997     0.998       307

    accuracy                          0.997      2031
   macro avg      0.998     0.995     0.997      2031
weighted avg      0.997     0.997     0.997      2031
```

Testing Data

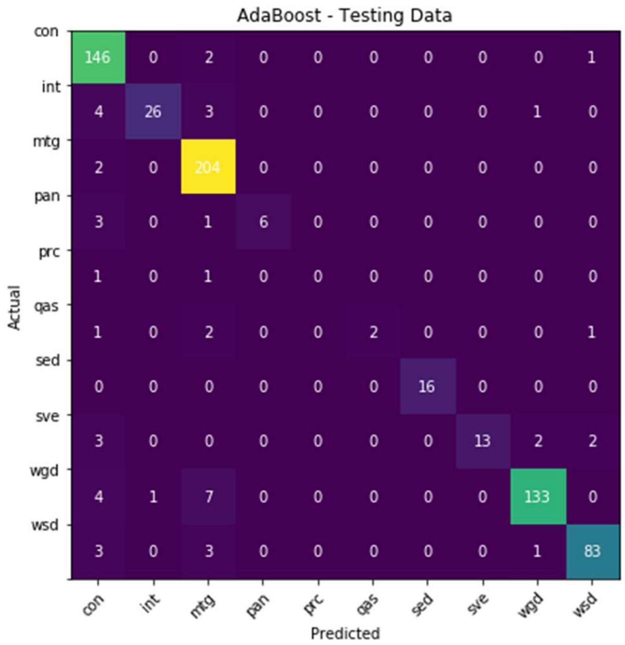


```
              precision    recall  f1-score   support

         con      0.874     0.980     0.924       149
         int      0.963     0.765     0.852        34
         mtg      0.915     0.990     0.951       206
         pan      1.000     0.600     0.750        10
         prc      0.000     0.000     0.000         2
         qas      1.000     0.333     0.500         6
         sed      1.000     1.000     1.000        16
         sve      1.000     0.650     0.788        20
         wgd      0.971     0.917     0.943       145
         wsd      0.954     0.922     0.938        90

    accuracy                          0.928       678
   macro avg      0.868     0.716     0.765       678
weighted avg      0.929     0.928     0.923       678
```



AdaBoost - Training Data

Actual \ Predicted	con	int	mtg	pan	prc	qas	sed	sve	wgd	wsd
con	456	0	0	0	0	0	0	0	0	0
int	1	115	0	0	0	0	0	0	0	0
mtg	1	0	568	0	0	0	0	0	0	0
pan	0	0	0	33	0	0	0	0	0	0
prc	0	0	0	0	7	0	0	0	0	0
qas	0	0	0	0	0	22	0	0	0	0
sed	1	0	0	0	0	0	87	0	0	0
sve	1	0	0	0	0	0	0	44	0	0
wgd	2	0	0	0	0	0	0	0	386	0
wsd	1	0	0	0	0	0	0	0	0	306



AdaBoost - Testing Data

Actual \ Predicted	con	int	mtg	pan	prc	qas	sed	sve	wgd	wsd
con	146	0	2	0	0	0	0	0	0	1
int	4	26	3	0	0	0	0	0	1	0
mtg	2	0	204	0	0	0	0	0	0	0
pan	3	0	1	6	0	0	0	0	0	0
prc	1	0	1	0	0	0	0	0	0	0
qas	1	0	2	0	0	2	0	0	0	1
sed	0	0	0	0	0	0	16	0	0	0
sve	3	0	0	0	0	0	0	13	2	2
wgd	4	1	7	0	0	0	0	0	133	0
wsd	3	0	3	0	0	0	0	0	1	83

Like the other ensemble learning method, the Random Forest model, the AdaBoost model struggled with recall and performed moderately well in accuracy in comparison with the other models used in this categorization problem. The press conferences once again proved too difficult for the model to successfully categorize correctly.

Conclusion

Overall each model managed excellent categorization results, with overall accuracy scores never dipping below 0.919. Of course these scores were supported by the overrepresentation of certain categories and the resulting ease of categorization for those categories, but in a low-stakes categorization situation such as this, these scores are excellent. A grouping of average accuracy scores across each model can be found in the figure 2.

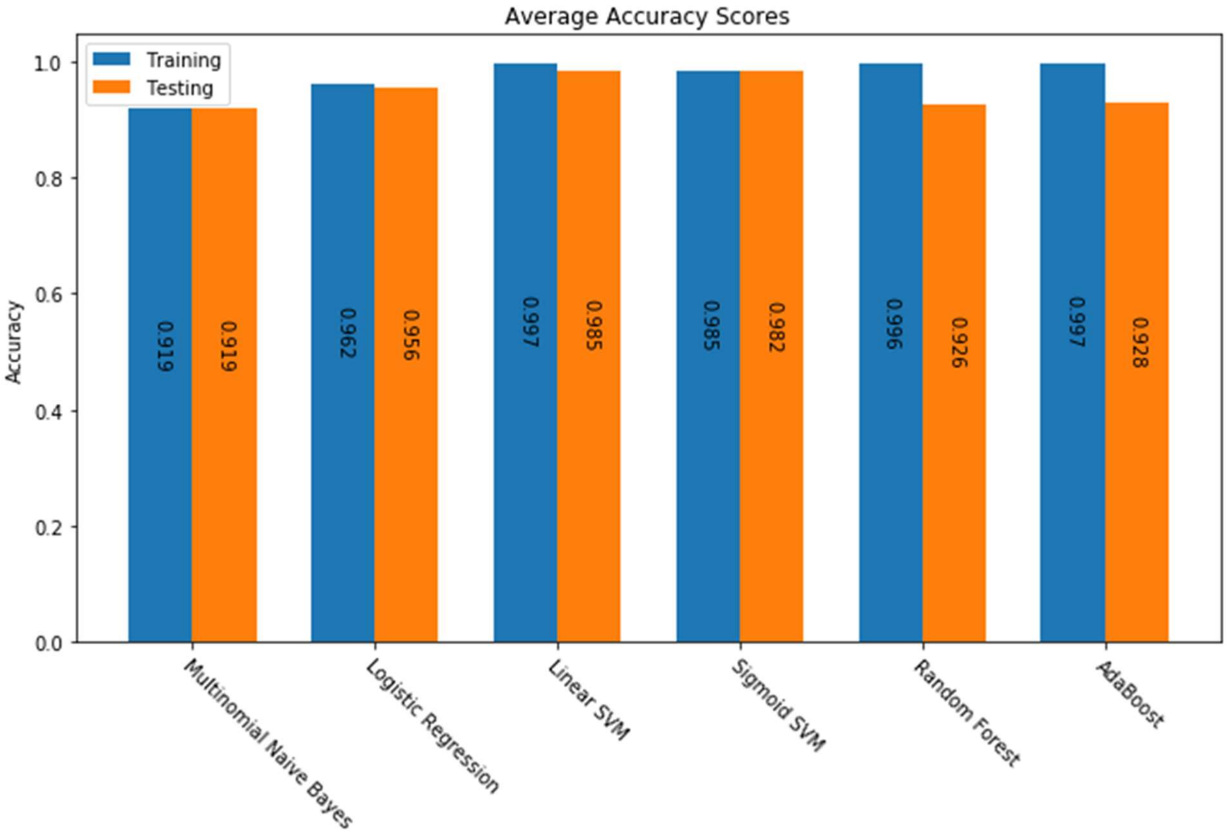


Fig 2: Average Accuracy Scores for Training and Testing Data Across All Models

For this exact problem, the linear SVM model proved most accurate, and its other metrics signal its overall better performance as well. The overrepresentation and underrepresentation of certain categories proved to be an issue and future research into this categorization problem would involve a deeper effort into balancing the representation of each category. Perhaps under sampling could have been used to help mitigate some of these issues. A key takeaway however, despite the flaws with the overall investigation, is that the SVM models perform the best when there is an imbalanced representation of categories. For NLP categorization problems with imbalanced representation, we would recommend using SVM models as a first effort before balancing.

If this project were to be continued, we would want to examine the XML versions of the transcriptions. There is a lot of a rich information in those versions that might be indicative of certain categories of conversation. Perhaps where people emphasize or put pauses in certain words might be useful signals of formality or of setting. Beyond that research, we believe there is a lot of room for investigation of imperfect speaker models, namely generative language models that output text that would fit into live, unscripted conversations. If artificial agents want to enter conversational spaces, it might be useful if they speak like we do, imperfectly and emotionally.

Team member contribution

Matthew Hosking – Entire Project

Code and Presentation Links

Slides link: https://drive.google.com/file/d/1Plrwdy4sSixwKhGpQnW2U3vtQedfrfJP/view?usp=sharing

Code link: https://drive.google.com/drive/folders/1P_-tozCAm-5Ze6upbZLd5qHA91mr9iIW?usp=sharing

All code and accompanying files can be found in the provided google drive folder. The entire codebase is found in the working notebook file.