From Decision Trees to LLaMA: Evaluating Model Performance in News Political Bias Detection

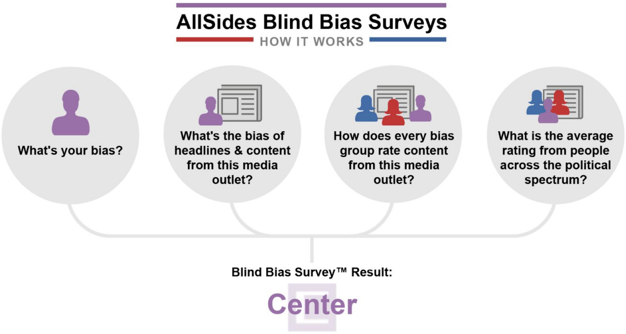
Group 10

Benjamin Lee Jun Cheng A0286163Y
Lai Foong Ming A0268245X

Project Motivation

Problem statement:

- Human-based news rating systems are not scalable
- Breaking news evaluated too late to be useful
- Push notifications and RSS feeds outpace manual review
- Rating sites have limited coverage of sources



Our approach:

- Develop ML-based political bias classifier

Goal:

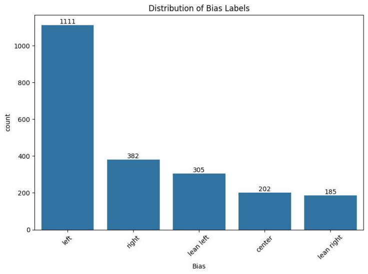
- Enable instant, on-demand bias assessment to enhance media literary

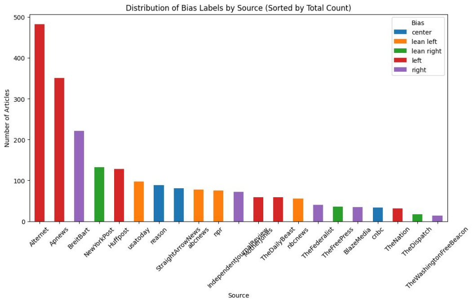
EDA

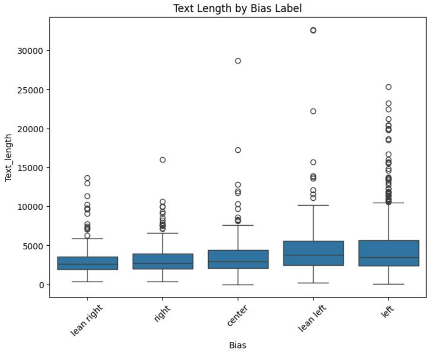
Dataset:

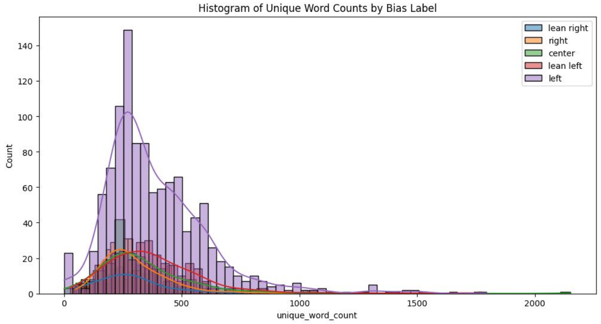
- From Kaggle
- n= 2185
- 5 target labels for bias:
  - Right, Lean Right, Centre, Lean Left, Left
- News articles from 21 unique sites
- Heavily imbalance towards left
- Source == Bias

- Analysis of text length and unique vocabulary shows that both are unlikely to be a good indicator with similar medians and IQR, as it will fail to discriminate strongly between adjacent classes



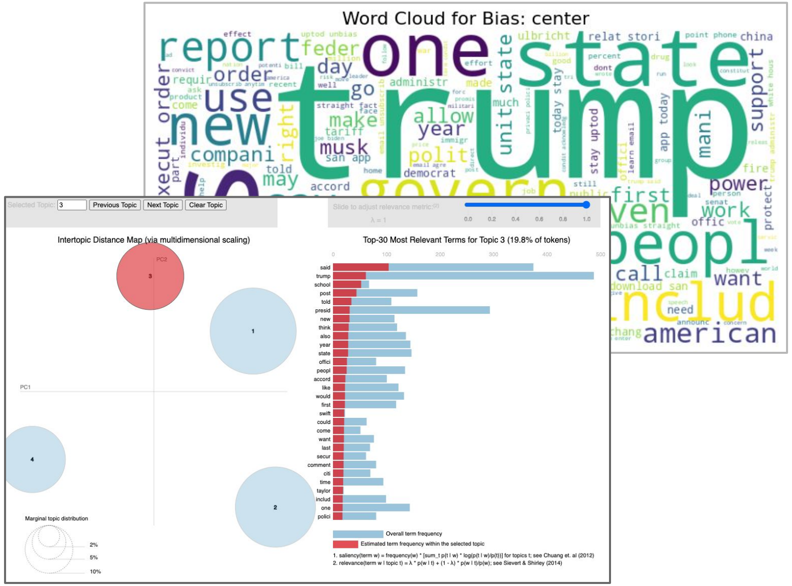






EDA

- Word clouds and LDA charts for every label shows that all bias address Trump as the top content
- Hence again we do not expect vocabulary to be a strong indicator



Preprocessing:

- Drop Source due to being a 1:1 predictor of bias
- Remove source names from article text to remove bias indicator

- For TD-IDF in decision tree and FFNN, apply stemming, remove numbers and punctuation
- For LSTM, BERT and Llama, use full words (retain numbers and punctuation)

- Split into test and train

Scrapper

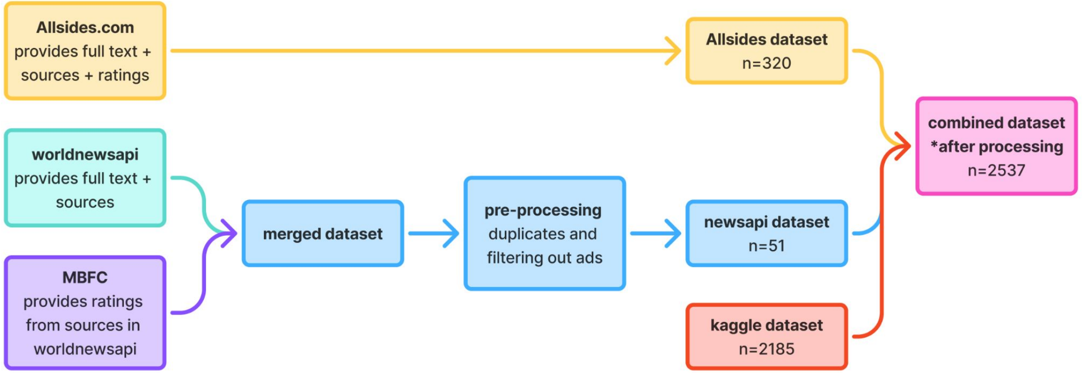
To deal with imbalance:

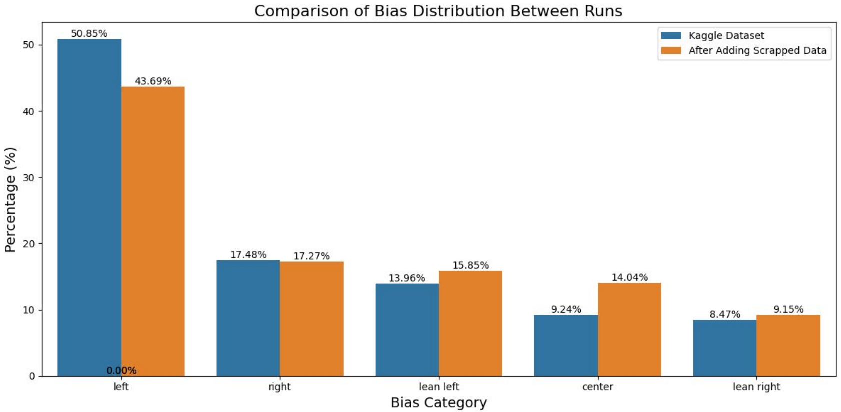
- Develop a scrapper in parallel to augment data for classes with lower representations
- Data needed:
  full text, source and bias label

- Built with selenium

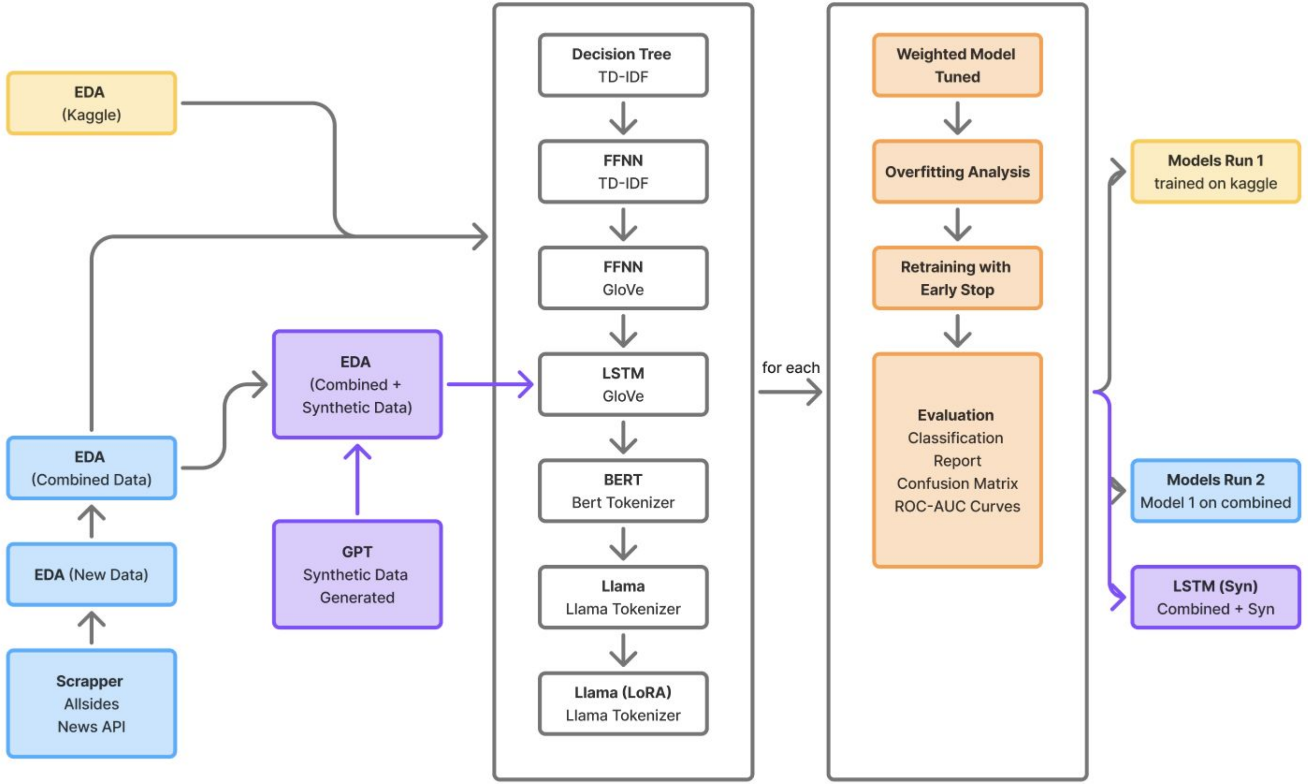
- For allsides, we wrote a script to scrape news from the allsides.com website, focusing on general news each day, sampling roughly equal from a broader category (left, center, and right) to balance our dataset

- For news api, only worldnewsapi gave the required data though it was limited and required augmentation

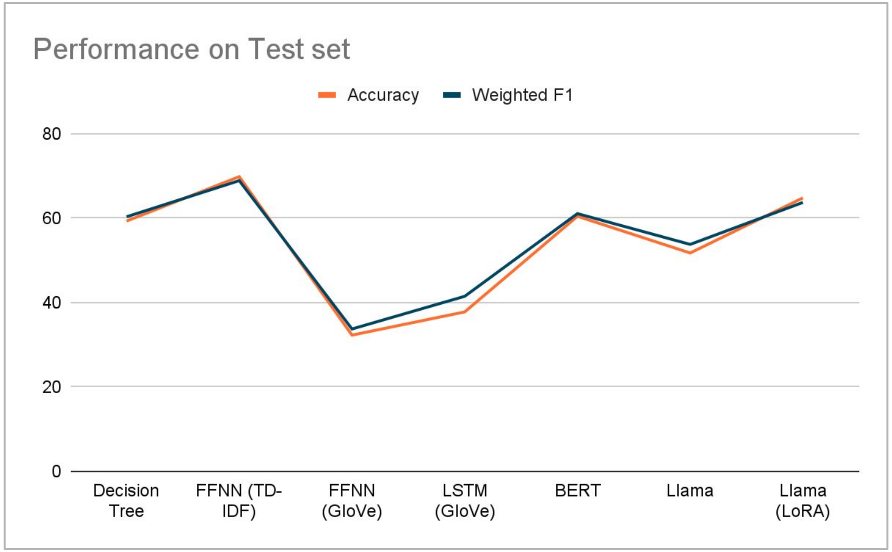




Project Flow



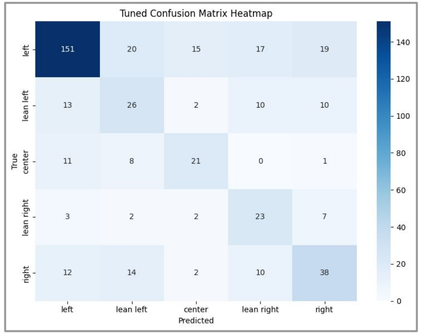
Model 1 - Headline Metrics Performance

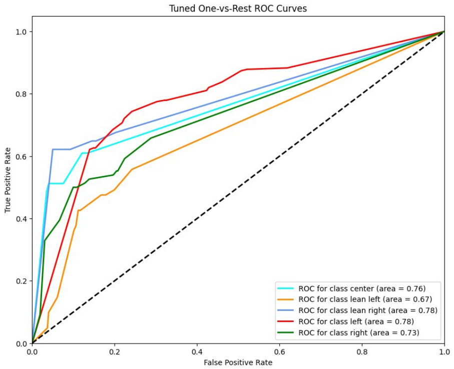


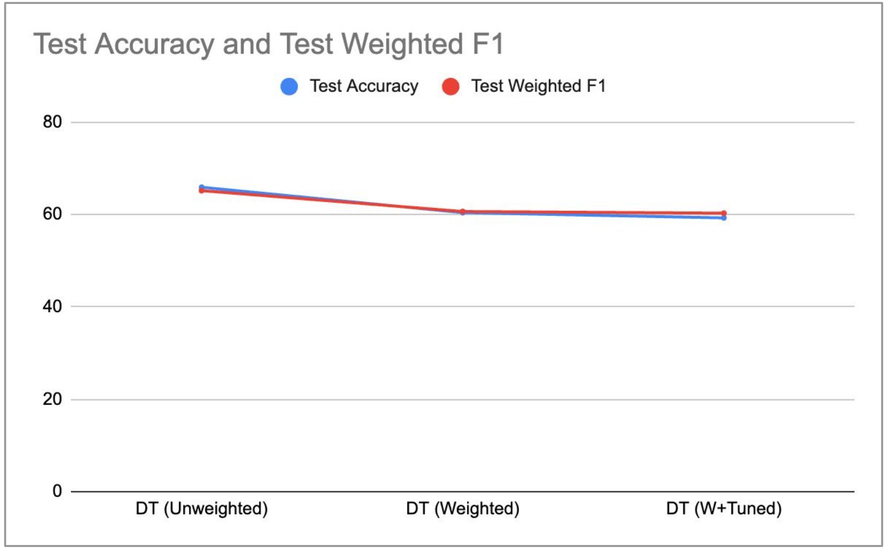
- Decision Tree's performance placed a benchmark of at least 60% for all subsequent models
- FFNN (TD-IDF) has the highest performance overall, suggesting that traditional methods with bag of words representations retain strong baseline especially when paired with dense architecture on relatively shallow tasks
- FFNN (GloVe) performed worst as expected due to purposeful mismatch in design
- LSTM's significant underperformance is surprising. We explore this further.
- BERT performed well out of the box and performance suggests that optimizing the model for the task might be worth testing.
- TinyLlama performed worse than BERT despite the higher complexity, but the loss metric signals fine tuning is needed
- TinyLlama (LoRA) performed well out of the box, but required significant time and memory to train

Decision Tree

- Three variants:
  - Unweighted
  - Weighted
  - Tuned
- Headline metrics fell across three treatments
- Examining confusion matrix and one vs rest ROC curves show that class separability improves
- Hence all future models adopt weighting and tuning





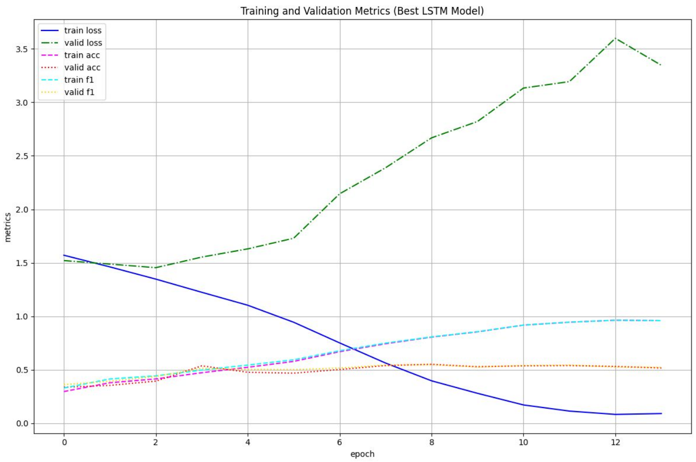


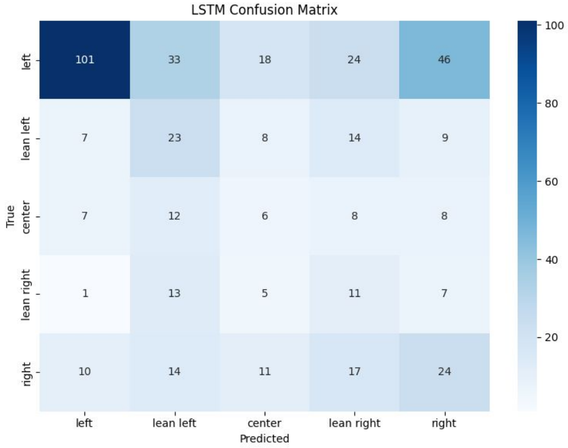
LSTM

- Smaller range in hyperparameters gridsearch due to compute constraints
- Early stop at epoch 6
- Model shows confusion between opposite ends of the spectrum with samples of true left and true right being distributed across all classes indicating that a more balanced data set could be advantageous

```
# FF TD-IDF Grid search parameters
param_grid = {
    'module__hidden_dim': [128, 256, 512],
    'module__dropout': [0.1, 0.3, 0.5],
    'lr': [0.0005, 0.001, 0.01],
    'batch_size': [16, 32, 64],
}
```

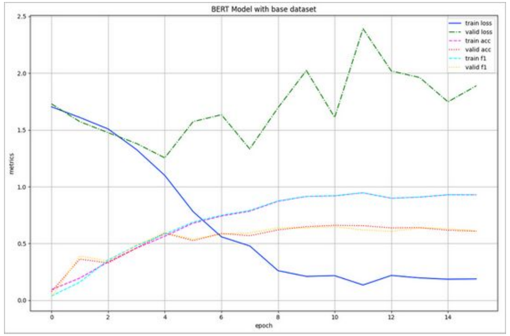
```
# LSTM Grid Search parameters
# as following exactly as in FFNN would give us 972 folds
# and require too much compute,
# we lower the params to those chosen by the 2 FFNN model
param_grid = {
    'module__hidden_dim':    [256, 512],
    'module__num_layers':    [1, 2],
    'module__bidirectional': [False, True],
    'module__dropout':       [0.5], #both FFNN modles chose 0.5
    'lr':                    [0.0005, 0.001],
    'batch_size':            [16] #both FFNN modles chose 16
}
```





BERT

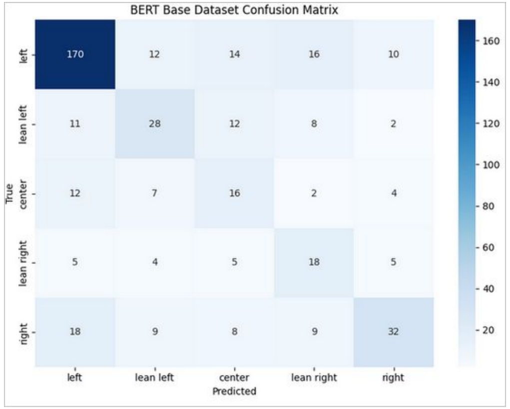
- The base model performed fairly well considering that we only used the pretrained model with an additional linear layer for classification.
- The time complexity while high, was still within reason for a consumer grade GPU (RTX 3060) at 19s per epoch. Performance was also good for the time complexity, since we did not do any optimizations on the model.
- There were some signs of overfitting when comparing the training and validation loss.



Training Base datset net

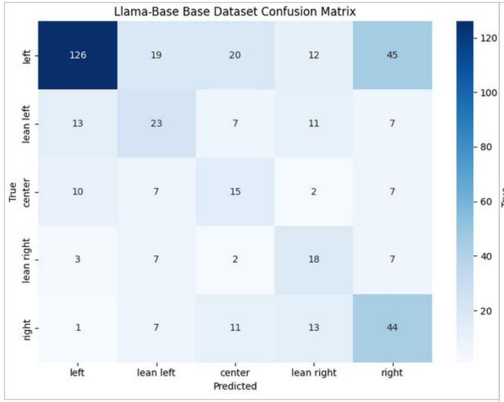
epoch	train_acc	train_f1	train_loss	valid_acc	valid_f1	valid_loss	lr	dur
1	0.0937	0.0381	1.7041	0.0771	0.0261	1.7296	0.0000	19.0082
2	0.1953	0.1594	1.6102	0.3629	0.3897	1.5732	0.0000	18.9863
3	0.3376	0.3550	1.5089	0.3286	0.3456	1.4763	0.0000	19.0022
4	0.4599	0.4813	1.3280	0.4600	0.4854	1.3810	0.0000	19.0065
5	0.5644	0.5812	1.0997	0.5914	0.5956	1.2545	0.0000	19.0403
6	0.6795	0.6879	0.7810	0.5257	0.5380	1.5733	0.0001	19.0338
7	0.7425	0.7486	0.5588	0.5886	0.5810	1.6346	0.0001	19.0562
8	0.7854	0.7908	0.4788	0.5686	0.5939	1.3328	0.0001	19.0712
9	0.8727	0.8747	0.2606	0.6200	0.6365	1.6963	0.0001	19.0734
10	0.9142	0.9151	0.2105	0.6486	0.6386	2.0239	0.0001	19.0650
11	0.9199	0.9211	0.2167	0.6600	0.6478	1.6106	0.0001	19.0802
12	0.9464	0.9467	0.1347	0.6571	0.6192	2.3915	0.0001	19.0703
13	0.8984	0.8992	0.2187	0.6371	0.6075	2.0192	0.0001	19.0914
14	0.9084	0.9092	0.1970	0.6400	0.6359	1.9621	0.0002	19.0935
15	0.9292	0.9302	0.1855	0.6171	0.6314	1.7478	0.0002	19.0711

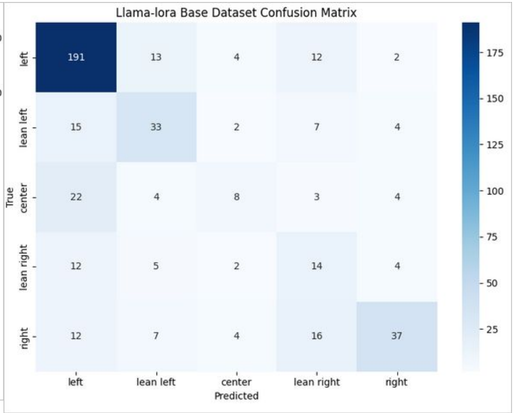
Stopping since valid_f1 has not improved in the last 5 epochs.
==========



Llama (Base & LoRA)

- Base TinyLlama model poor performance relative to the LoRA optimized version of TinyLlama, suggesting that the model needed more fine tuning before having task-specific strong performance.
- The LoRA implementation did convert faster (13 vs 22 epochs) during training, which seems to outweigh the additional complexity added from the optimization.
- Both the Base & LoRA optimized version showed signs of overfit, but did generalize well relative to the other models.





Base

Training base dataset net

epoch	train_acc	train_f1	train_loss	valid_acc	valid_f1	valid_loss	lr	dur
1	0.3677	0.3528	1.6418	0.3829	0.3556	1.6483	0.0000	71.5293
2	0.3598	0.3505	1.6380	0.3771	0.3547	1.6405	0.0000	72.1421
3	0.3484	0.3483	1.6265	0.3657	0.3568	1.6281	0.0000	72.2789
4	0.3476	0.3544	1.6099	0.3629	0.3599	1.6135	0.0000	72.2924
5	0.3519	0.3625	1.5896	0.3686	0.3679	1.5972	0.0000	72.1382
6	0.3627	0.3719	1.5660	0.3686	0.3675	1.5793	0.0000	72.1533
7	0.3863	0.3958	1.5393	0.3629	0.3612	1.5599	0.0000	72.1418
8	0.4106	0.4202	1.5103	0.3943	0.3886	1.5398	0.0000	72.1504
9	0.4320	0.4436	1.4800	0.3943	0.3947	1.5195	0.0000	72.1785
10	0.4521	0.4644	1.4490	0.4057	0.4070	1.4996	0.0000	72.2189
11	0.4685	0.4819	1.4181	0.4286	0.4318	1.4805	0.0000	72.2495
12	0.4936	0.5070	1.3878	0.4257	0.4320	1.4625	0.0000	72.1295
13	0.5007	0.5160	1.3584	0.4286	0.4395	1.4458	0.0000	72.1398
14	0.5079	0.5239	1.3302	0.4486	0.4607	1.4303	0.0000	72.1801
15	0.5200	0.5358	1.3032	0.4514	0.4641	1.4162	0.0000	72.1769
16	0.5272	0.5424	1.2774	0.4600	0.4743	1.4032	0.0000	72.1556
17	0.5379	0.5535	1.2527	0.4629	0.4779	1.3914	0.0000	72.2230
18	0.5443	0.5601	1.2292	0.4657	0.4813	1.3806	0.0000	72.2713
19	0.5608	0.5763	1.2066	0.4686	0.4833	1.3706	0.0000	72.2910
20	0.5701	0.5852	1.1849	0.4714	0.4867	1.3614	0.0000	72.2706
21	0.5808	0.5963	1.1639	0.4743	0.4895	1.3529	0.0000	72.2166
22	0.5937	0.6079	1.1436	0.4743	0.4899	1.3448	0.0000	72.3235
23	0.6044	0.6178	1.1239	0.4743	0.4897	1.3373	0.0000	72.2904
24	0.6094	0.6228	1.1048	0.4686	0.4837	1.3302	0.0000	72.3470
25	0.6209	0.6341	1.0861	0.4686	0.4847	1.3234	0.0000	72.3300
26	0.6288	0.6415	1.0679	0.4714	0.4873	1.3169	0.0000	72.1906

Stopping since valid_f1 has not improved in the last 5 epochs.

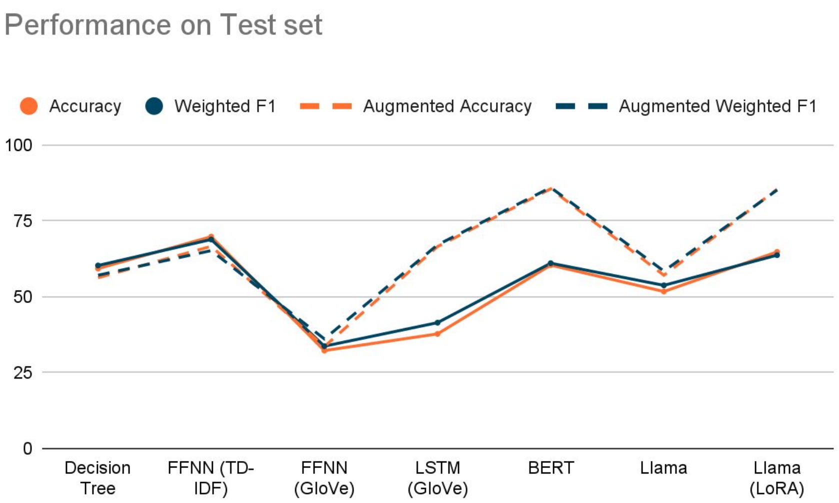
LoRA

Training base dataset net

epoch	train_acc	train_f1	train_loss	valid_acc	valid_f1	valid_loss	lr	dur
1	0.1431	0.1048	1.6409	0.1514	0.1176	1.6564	0.0000	139.8936
2	0.1567	0.1273	1.6287	0.1829	0.1556	1.6327	0.0000	140.3941
3	0.2089	0.2193	1.5947	0.2714	0.2890	1.5989	0.0000	140.4814
4	0.3212	0.3505	1.5578	0.3829	0.4021	1.5733	0.0000	140.4849
5	0.4099	0.4259	1.5223	0.4371	0.4470	1.5413	0.0000	140.5388
6	0.4700	0.4825	1.4685	0.4714	0.4820	1.4772	0.0000	140.5368
7	0.5072	0.5241	1.3769	0.4857	0.5029	1.3770	0.0000	140.5697
8	0.5465	0.5662	1.2815	0.5314	0.5496	1.2979	0.0000	140.5695
9	0.5994	0.6164	1.1573	0.5714	0.5945	1.1969	0.0000	140.6051
10	0.6545	0.6710	0.9913	0.5829	0.6047	1.1131	0.0000	140.6277
11	0.7117	0.7233	0.8217	0.6143	0.6339	1.0818	0.0000	140.6706
12	0.7775	0.7858	0.6452	0.6343	0.6446	1.1395	0.0000	140.6365
13	0.8398	0.8446	0.4828	0.6629	0.6719	1.2262	0.0000	140.6059
14	0.8841	0.8866	0.3404	0.6571	0.6583	1.3588	0.0000	140.6183
15	0.9242	0.9254	0.2327	0.6486	0.6525	1.5068	0.0000	140.6417
16	0.9134	0.9144	0.2278	0.6400	0.6347	1.6261	0.0000	140.6260
17	0.9349	0.9356	0.1822	0.6343	0.6450	1.4799	0.0000	140.6416

Stopping since valid_f1 has not improved in the last 5 epochs.

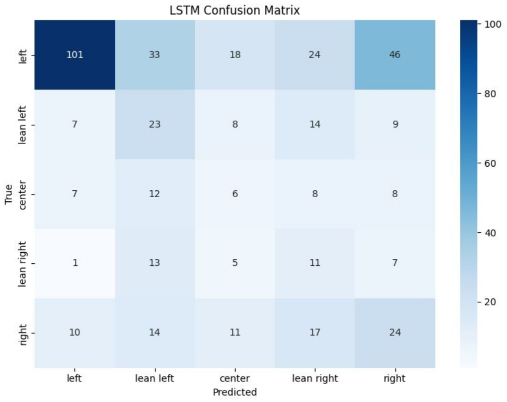
Model 2 - Retraining Headline Metrics Performance



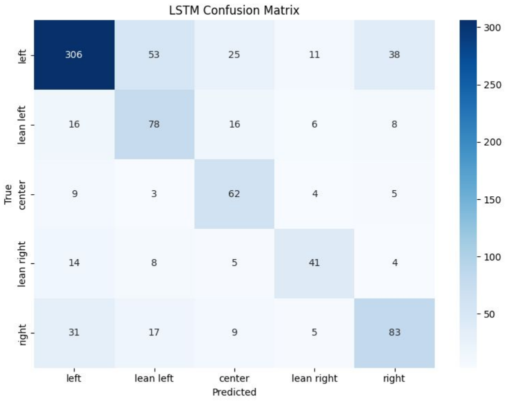
- Utilising the same model parameters found in the original runs, the models were re-run with the augmented datasets
- Slight falls for decision tree and FFNN (TD-IDF) were not expected, which indicated signs of overfitting previously
- Dismal performance for FFNN (GloVe) not improving was also expected.
- While we expected performance to increase across the board, of note are the dramatic increases in performance for LSTM and BERT
- While BERT & Llama (LoRA) increased significantly, it was surprising that Llama did not. This could have been due to Llama not being able to perform well with only an additional layer for classification.

LSTM

- Increase in sample size by 16% resulted in jump in performance metrics
- Accuracy: 37.76 to 66.51
- Weighted F1: 41.46 to 67.08
- Modest increase in data resolved ideological confusion and diagonal performance increased
- Suggests that balanced data and sequential information are highly relevant for this classification task
- With the expanded dataset, LSTM could leverage the additional contextual information, while the FFNN (GloVe) did not capture these patterns from averaged embeddings.



Before



After

Bert & Llama

- Performance for BERT & TinyLlama (LoRA) improved significantly after using the augmented dataset.
- TinyLlama (Base) only increased slightly, suggesting that fine tuning is required to adapt the model to classification (or other specific) tasks
- Duration increased for all 3 models, around 1.5 times with the augmented dataset. Despite that, BERT was still running much quicker than TinyLlama, and yielded equivalent or better results.

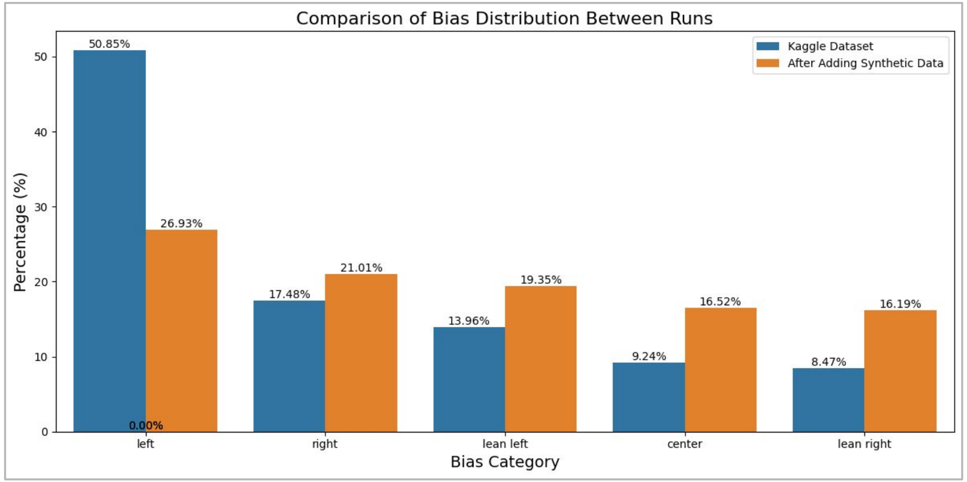
Model	Accuracy	Weighted F1	Accuracy (Aug)	Weighted F1 (Aug)
Decision Tree	59.27	60.29	56.25	57.08
FFNN (TD-IDF)	69.79	68.85	66.60	65.20
FFNN (GloVe)	32.27	33.72	33.59	36.11
LSTM	37.76	41.46	66.51	67.08
BERT	60.41	61.04	85.53	85.97
TinyLlama	51.72	53.75	57.06	58.44
TinyLlama (LoRA)	64.76	63.69	85.41	85.14

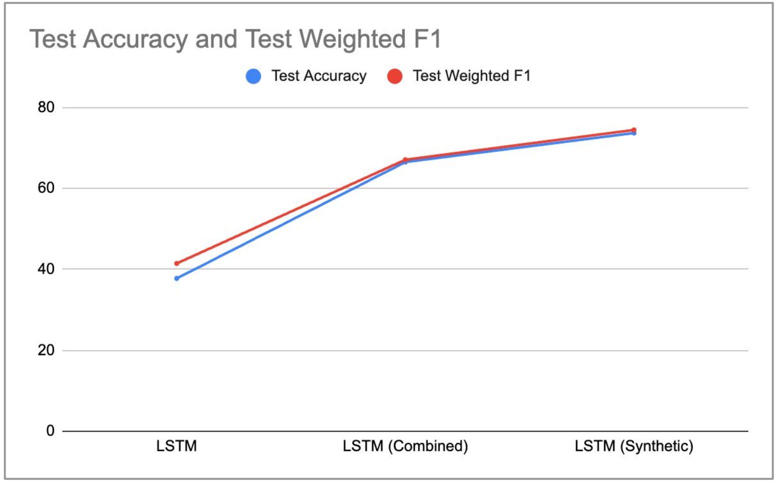
Time per epoch

Model	Time (Base)	Time (Augmented)
BERT	19s	37s
TinyLlama (Base)	72s	141s
TinyLlama (LoRA)	140s	275s

Utilising Synthetic Data

- **Baseline LSTM:** Poor performance
- **LSTM with Combined Data:** Significant performance boost showing that augmenting the dataset with real additional data substantially helps.
- **LSTM with Synthetic Data:** Further marginal gains, suggesting diminishing returns but confirming the value of data augmentation, even when synthetic.





Conclusion

Goal: Enable instant, on-demand bias assessment to enhance media literary

Recommendation: BERT

1. Has the best overall performance with peak accuracy of 85.53 and peak weighted f1 of 85.97
2. Only requires 128 tokens for inferencing, which aligns to what is available in article previews and before paywalls
3. Supports fast inferencing to match a user's scrolling speed when deployed on local devices
4. Lower compute needed compared to Llama or Llama (LoRA) for implementation in production such as on a user's handphone or laptop

References

- AllSides. (n.d.). Media bias rating methods. AllSides. Retrieved May 2, 2025, from https://www.allsides.com/media-bias/media-bias-rating-methods
- World News API. (n.d.). The only News API you'll ever need. https://worldnewsapi.com/
- Media Bias/Fact Check. (n.d.). Search and learn the bias of news media. https://mediabiasfactcheck.com/
- Khandelwal, U., He, H., Qi, P., & Jurafsky, D. (2018). Sharp nearby, fuzzy far away: How neural language models use context. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 284–294). Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1027
- Kawin Ethayarajh. 2019. How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 55–65, Hong Kong, China. Association for Computational Linguistics.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V., 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.