Baseline Methods

We compare our approaches with 4 state-of-the-art methods from the literature. These were selected based on performance, taking into account the difficulty level of the data they were tested on. Here we briefly summarize some of their key points and provide the details on how we implemented them:

Barberá (Barberá 2015; Barberá et al. 2015) This item response model assigns scores to users based on who they follow. We use the Tweetscores[7] implementation, which compares a user against "elite" (politicians and media) users with pre-trained scores.

We classify any user with score greater (resp. less) than 0 as Republican (resp. Democrat), which both matches intuition and gives the best performance empirically. In particular, we show here the results of different classification thresholds for Barberá's model. This shows a single run in the setting of Table 7 (all users available), with threshold increments of 0.05 from -2 to 1.

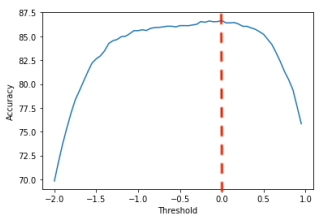

Figure 3: Tuning the threshold for classification with Barberá's approach. Accuracy is maximized at exactly 0.

Preoţiuc-Pietro (Preoţiuc-Pietro et al. 2017) This is a bag-of-words style approach with a custom, specialized vocabulary of 352 politics-related words. For each user we concatenate all their tweets into one document and calculate the counts of these words. Following the original paper, we then use this feature vector as input to a logistic regression that classifies each user. We implement the logistic regression through scikit-learn (Pedregosa et al. 2011) with default hyperparameters.

POLITICS (Liu et al. 2022) This model is based on RoBERTa-base, but adds political domain adaptation through training on news articles with ideology labels. We pass the final-layer embedding of the [CLS] token through a single fully-connected one-layer classification head. With this architecture, we fine-tune the model with the same setup that we used for RoBERTa-base. We also report performance of an untuned version where we obtain user embeddings from averaging the pretraining-only tweet chunk [CLS] embeddings, and classifying users in the same way that the we classify the user embeddings from GCN.

TIMME (Xiao et al. 2020) This approach is based on a variation of a GCN, but the architecture is adapted to learn from multiple data types simultaneously and end-to-end. This contrasts with our GCN approach which trains an independent GCN per data type, and combines them in a separate final stage. Therefore, this design hopes to learn more complex interactions between the different data types, at the cost of being much more computationally intensive and potentially more difficult to train. Unfortunately, even when using high-end AI-specialized hardware and the original authors' code,[8] we found the computational burden is severe. We elaborate in the following note.

A Note on TIMME model

We attempted to implement TIMME using 100GB RAM and an RTX8000 GPU (which has 48GB VRAM), using the code released by the authors. However, we found both versions TIMME and TIMME-Hierarchical frequently produced out of memory errors with our scale of data. This is likely because, unlike all our proposed graph-based approaches, TIMME requires loading multiple graphs in memory simultaneously for end-to-end training. We also note that our data has over 4 times more interactions than the largest dataset used in the TIMME paper, and over 100 times more than the smallest. In addition, in some tests where we restricted the data and got it to run, it was over 100 times slower per epoch than our GCN approach (which is itself 100 times slower than our label propagation approach), and did not give good performance. Therefore, although this is arguably the most sophisticated approach and might give strong performance in some settings like the original paper, the current version is challenging to apply to larger datasets.

Distribution of Unretrievable US Public Users

We examine the distribution of the missing users in Table 11. We see that a strong majority are Republicans. We hypothesize that many of these users left around the January 6^{th} capitol attack, either voluntarily or during the subsequent wave of suspensions.

Additional Information on Canadian Data

Collection We sampled 1% of real-time tweets containing at least one of the following keywords:

'trudeau', 'freeland', 'o'toole', 'bernier', 'blanchet', 'jagmeet singh', 'annamie', 'debate commission', 'reconciliation', 'elxn44', 'cdnvotes', 'canvotes', 'canelection', 'cdnelection', 'cdnpoli', 'canadianpolitics', 'canada', 'forwardforeveryone', 'readyforbetter', 'securethefuture', 'NDP2021', 'votendp', 'orangewave2021', 'teamjagmeet', 'UpRiSingh', 'singhupswing', 'singhsurge', 'VotePPC', 'PPC', 'peoplesparty', 'bernierorbust', 'mcga', 'saveCanada', 'takebackcanada', 'maxwillspeak', 'LetMaxSpeak', 'FirstDebate', 'frenchdebate', 'GovernmentJournalists', 'JustinJournos', 'everychildmatters', 'votesplitting', 'ruralcanada', 'debatdeschefs', 'électioncanadienne', 'polican', 'bloc', 'jevotebloc'

Labeling From the main dataset, we sampled users with the following keywords in their profile:

CPC: 'erin o'toole', 'andrew scheer', 'conservative', 'conservative party', 'cpc', 'cpc2021', 'cpc2019', 'conservative party of canada'

GPC: 'annamie paul', 'green party', 'gpc', 'gpc2019', 'gpc2021', 'green party of canada'

LPC: 'justin trudeau', 'liberal', 'liberal party', 'lpc', 'lpc2021', 'lpc2021', 'lpc2019', 'liberal party of canada'

NDP: 'jagmeeet singh', 'new democrat', 'new democrats', 'new democratic party', 'ndp', 'ndp2021', 'ndp2019'

PPC: 'maxime bernier', 'people's party', 'ppc', 'ppc2019', 'ppc2021', 'people's party of canada'

[7] https://github.com/pablobarbera/twitter_ideology

[8] https://github.com/PatriciaXiao/TIMME

Table 10: Appendix: Survey of methods to predict ideology (with accuracy below 65% or no mention of the accuracy)

Methods	Accuracy	Media	Activity	Relation	Content	Type	Size	Code	Difficulty
Liu et al. (2022)	<50%				✓	Public	1,079	✓	Hard
Pastor-Galindo et al. (2020)	NA				✓	Public	20,364	✓	Easy
Sinno et al. (2022)	55%	✓			✓	Political news	175		Easy
Yang, Hui, and Menczer (2022)	NA	✓			✓	Public	NA	✓	Easy
Chen (2015)	NA			✓	✓	Both	NA		Hard
Bright (2016)	NA		✓		✓	Both	NA		Medium
Gaisbauer et al. (2021)	NA		✓		✓	Both	NA		Medium
Garimella and Weber (2017)	NA	✓	✓	✓	✓	Both	NA		Medium
Kamienski et al. (2022)	NA		✓	✓	✓	Public	NA		Easy

Table 11: Distribution of users whose data could not be retrieved retroactively. The majority are Republican, possibly users who left around January 2021, i.e., the wave of suspensions that included Donald Trump.

	Republican	Democrat
Suspended	1,824	439
Deleted	1,857	543
Private	186	267

Two political science graduate students then manually verified or corrected the labels. This led to the counts of labeled users shown in Table 12

Table 12: Counts of manually-labeled Canadian users.

Party	Count
CPC	2039
LPC	1808
NDP	704
PPC	451
GPC	299

Supplementary Implementation Information

For each individual interaction type, we train only on the top 50% of most active users for that relation according to raw counts. This filter is not applied to test data.

GCN We train for 1000 epochs with the Adam optimizer (Kingma and Ba 2014) with PyTorch default parameters (learning rate = 1e-3). When we use a random forest to combine different data types, we use the scikit-learn (Pedregosa et al. 2011) implementation with default hyperparameter settings. As in all experiments, we train it using users with profile classifier labels, and report test results on a (fully separate) set of users using manual labels.

Label Propagation There are two parameters: the number of iterations of the propagation process, and the rate α at which new label information replaces the old one. We first tested it on projected graphs, where we found performance decreases consistently and monotonically with more iterations irrespective of α value (specifically, tested iterations in [1,2,3,4,5,6,8,10,15,20,25], with fully tested alpha values [0.1,0.5,0.9] and partially tested [0.03,0.97]). When using a single iteration, this method corresponds to taking the majority vote of the neighbors' labels, and changing α has no effect.

We later found that performance is better on the direct graph, specifically with two iterations and $\alpha = 0.5$. With 1 iteration on this graph, the performance is poor, while with more iterations performance remains constant (accounting for margin of error) or decreases. The direct graph takes much longer to run, so due to time/computation constraints we were not able to test more values of α in this setting. Therefore, except where stated otherwise, we report results from the best performing version, i.e., two iterations on direct graph with $\alpha = 0.5$.

RoBERTa In order to provide additional context for the models, We concatenated each user's tweets in time order into chunks. Since the positional encoding scheme of these language models limits the length of input to a fixed number of tokens, we start a new chunk each time the previous chunk goes above the length limit of the model. Any tweets that don't fit into a single chunk are truncated and placed into a chunk of their own. As a result, no tweet are split between chunks. Note that we also use this same preprocessing for the POLITICS (Liu et al. 2022) baseline model described previously, which has a similar architecture.

These models are pretrained without a sequence classification task. Consequently, despite RoBERTa models having a [CLS] token like BERT and many similar language models, the representation of this token is not pretrained and thus unsuitable for direct use in downstream tasks. In some experiments we tested the model without the finetuning needed to properly learn this token. Therefore, to improve comparability, in all experiments with RoBERTa instead of [CLS] we use the mean of all individual word embeddings from the final output layer.

To further improve the prediction accuracy, we fine-tuned these models with a tweet chunk classification task. First, we label each chunk in the train set according to the profile classifier label of the corresponding user. We add a fully-connected dense layer (the "classification head") to each model. The input size of this dense layer is equal to the size of the hidden dimensions of each transformer model, while the output size of this layer is equal to the number of label classes (2 in our case.). We then finetuned each model end-to-end for 1 epoch using the Optax (Babuschkin et al. 2020) implementation of the adamw optimizer (Loshchilov and Hutter 2017) with default weight decay strength 1e-4. We set the learning rate to 1.592e-5 for RoBERTa-base, 2.673e-5 for POLITICS, and 4.269e-6 for RoBERTa-large, based on gridsearch hyperparameter tuning on a validation set of users labeled by our profile classifier. When evaluated on the profile classifier labels, this setup leads to $91.5\% \pm 0.2\%$ test accuracy on tweet chunks for RoBERTa-base, $91.6\% \pm 0.3\%$

Table 13: Comparing semisupervised (SS) vs. unsupervised (US) GCN. Semisupervised performs better.

	Retweet	Mention	Quote	Hashtag	Friend	Follow
GCN SS	96.7 ± 0.7	83.7 ± 1.8	90.3 ± 1.5	88.7 ± 1.8	96.8 ± 0.6	93.3 ± 1.1
GCN SS+RF	96.2 ± 1.2	85.6 ± 1.7	89.4 ± 1.1	88.1 ± 1.8	96.5 ± 0.7	92.5 ± 1.2
GCN US+RF	96.1 ± 0.6	83.1 ± 2.2	87.6 ± 1.2	87.1 ± 1.3	94.0 ± 1.1	91.9 ± 0.9
GAT SS	96.9 ± 0.4	88.1 ± 1.3	93.5 ± 1.0	91.4 ± 1.3	96.7 ± 0.4	95.4 ± 0.8
GAT SS+RF	91.9 ± 1.6	73.3 ± 3.5	81.9 ± 1.6	78.2 ± 3.4	88.3 ± 1.1	86.3 ± 1.2
Label Prop.	97.2 ± 0.5	91.0 ± 0.9	95.7 ± 0.8	92.6 ± 0.7	96.5 ± 0.6	96.1 ± 0.8

for POLITICS and $91.9\% \pm 0.3\%$ for RoBERTa-large. Note that these accuracies for tweet chunk classification are only on weak labels, and are correlated with but not directly comparable to user classification reported in all other experiments.

To get a prediction for each user, instead of a prediction on an individual tweet chunk, we first predict a label for each of a user's chunks. Then we take the majority vote.

Additional Experiments

Effect of Supervision

We compare our GCN model, which does both unsupervised link prediction and supervised party prediction with training data labeled by the profile classifier, with training embeddings through a fully unsupervised GCN doing link prediction alone. We also evaluate the difference between using the supervision in the graph model itself versus sending the embeddings to a random forest (RF) for the final prediction.

Results are shown in Table 13. First, we see that although the margin is not huge, the semisupervised GCN version performs consistently better. This indicates the supervision helps to produce more informative embeddings from the GCN.

Second, we see that the RF, although it enables one to combine different interaction types together as in the experiments above, hurts performance slightly with GCN. With GAT, it is significantly detrimental, possibly due to architectural differences. However, even without the RF, both GCN and GAT are outperformed by label propagation in almost every case.[9]

Effect of Projection

In this experiment, we examine the impact on performance of using projected (indirect) graphs vs. the original (direct) ones. We show results in Table 14.

We see that GCN performance degrades when not projecting the graph. Adding another layer to the GCN, which in principle might help it use information from two-hop neighbors in a similar way to projection, turns out to be further detrimental. On the other hand, for GAT and label propagation, the direct graph generally performs better. The exception is the follow relation, which in all cases is better with the projected graph, which might reflect underlying user behavior differences between the interaction types. For label propagation in particular, the best version we found is two iterations on the direct graph. The best projected graph version is one iteration, but it is clearly worse overall, while one iteration on the direct graph is worse still.

Overall, along with the earlier results on runtime, the choice of direct vs. projected graph can have a significant impact. It is not as commonly tested as simple hyperparameters, and may be worth examining in more contexts.

[9]Consequently, in other experiments, unless otherwise noted, we report all GCN results using the semisupervised version with RF, so the results are comparable when combining data types. While for GAT, which we seldom use for combining data types, we report the version without RF.

Table 14: Comparing projected (Pro.) vs. direct (Dir.) graphs. Projection improves GCN performance. But it hurts label propagation (LP) performance, at least if the number of iterations is tuned for it.

	Retweet	Mention	Quote	Hashtag	Friend	Follow
Pro. GCN-1L	96.2 ± 1.2	85.6 ± 1.7	89.4 ± 1.1	88.1 ± 1.8	96.5 ± 0.7	92.5 ± 1.2
Dir. GCN-1L	92.1 ± 0.8	77.1 ± 2.3	83.1 ± 1.7	79.1 ± 1.7	88.9 ± 1.8	75.1 ± 2.8
Dir. GCN-2L	87.4 ± 1.5	71.2 ± 2.3	76.3 ± 2.6	76.3 ± 1.3	82.4 ± 2.8	69.3 ± 2.7
Pro. GAT	96.6 ± 0.6	66.9 ± 1.3	92.2 ± 1.3	90.2 ± 1.6	94.5 ± 0.9	96.1 ± 0.9
Dir. GAT	96.9 ± 0.4	88.1 ± 1.3	93.5 ± 1.0	91.4 ± 1.3	96.7 ± 0.4	95.4 ± 0.8
Pro. LP-1I	96.6 ± 0.6	66.5 ± 1.9	83.4 ± 0.9	86.3 ± 1.1	70.1 ± 1.6	96.6 ± 0.8
Dir. LP-1I	75.4 ± 1.9	68.1 ± 1.8	68.7 ± 1.9	66.3 ± 1.9	88.0 ± 1.4	87.3 ± 1.5
Dir. LP-2I	97.2 ± 0.5	91.0 ± 0.9	95.7 ± 0.8	92.6 ± 0.7	96.5 ± 0.6	96.1 ± 0.8

Users without all interaction types experiment: additional notes

Average users retrievable is calculated by first taking the per user counts of the necessary data type in our dataset. From there and the numbers in Table 6, we calculate how many API requests are needed to retrieve that user's data, and finally how many users can be retrieved in every 15 minutes. Retrieval from different API endpoints can be run in parallel, so when an approach uses a combination of data types, we report on the one that takes the longest to retrieve.

The standard deviation reported for our approaches here is the result of re-running the model itself 10 times, as in other experiments, but here because we examine all possible test users the test set does not change within these 10 runs. Existing state-of-the-art models were run once. The standard deviation for users retrievable is not from multiple runs; rather, it is the standard deviation of our sample in its estimation of the average users retrievable.

When evaluating combinations of interaction types we use the GCN embedding where available; otherwise we treat the embedding as all zeros. So for example, if we are considering Retweet plus Quote (RT+QT in Table 7), then if both types of activity are available we use both. While if only one is available we concatenate that one with a vector of zeroes for the missing one, which tells the final classification model that quote is not available.

Politicians experiment: additional notes

In this experiment we use the unsupervised GCN version of our model. This lets us test how training the final classification on US Politicians alone will translate to performance on US Public, with the GCN part of our model (and thus the embeddings) held constant.

When testing on US Public using US Politicians, we use all of them, while when testing on US Politicians, we do a 75-25 random split. In all cases the setting corresponds to

Table 4, i.e. users with all interaction types, again to keep the test set consistent between the different approaches.

Computational Resources and Libraries

Most experiments were done using RTX8000 GPUs. A number of text-based experiments were also run on v3-8 and v4-8 TPU VMs. Graph-based models were implemented using DGL (Wang et al. 2019), and language models using HuggingFace (Wolf et al. 2019) in JAX (Bradbury et al. 2018). To run all models of the main comparison experiment (Table 4), the roughly estimated time using 10 RTX8000 is one to two weeks.

Additional Literature Summary

In Table 10 we provide information on papers with lower or not reported performance to predict ideology, below the threshold for inclusion in Table 1. In contrast with Table 1, this survey includes only papers for which the accuracy was less than 65% or unknown.