Adam Hare
Bernardo Pérez Orozco
Machine Learning
1 March 2017

Silent Majority:
Predicting Political Leanings from Facebook Likes
Using Collaborative Filtering & Neural Networks

1 Introduction and Motivation

The term “silent majority” has been part of the political vernacular since Richard Nixon used it in 1969 to refer to a large group of voting Americans who generally refrain from publically expressing political views [1]. More recently, such a term could be applied to the results of the Brexit vote and the 2016 presidential election in the United States. The majority of political analysts failed to correctly predict either outcome despite complex models and extensive polling.

Similarly, applications and extensions that use Facebook page likes to draw conclusions about the political nature of the network, such as PolitEcho [2], rely on handpicked pages determined by the creators to indicate a political bias. There was no data on users who had not liked a page from this list so it is unlikely those from the “silent majority” would be included. How to determine the political leanings of these users can be thought of as a traditional classification problem, perhaps lending itself well to analysis by machine learning.

The decision to use Facebook page likes as the dataset was guided by the website’s widespread use as a political forum and breadth in terms of information. It was also inspired by the fact that Facebook does a similar analysis on its users to generate targeted ads [3]. The goal was to use a machine learning method to associate groupings of ostensibly apolitical page likes to a political leaning. To gather the data, a Google Chrome extension was built and run, using a single user’s Facebook profile and available friends information. The data was then passed through a collaborative filter and then given to a neural net to train. The neural net was run with both the unfiltered and filtered data. The section 2 will elaborate on these methods.

2 State of the Art and Background

2.1 Collaborative Filtering

Collaborative filtering is a very popular machine learning technique for predicting user preferences given sparse data. It is widely used by major corporations when creating suggestions for their users. Perhaps most famously, various methods of collaborative filtering were combined to win the Netflix Grand Prize in 2009 [4] and are presumably still used to make recommendations on Netflix today.

Collaborative filtering is specifically used to fill in “missing” data. In the case of analysis of Facebook likes, the fact that a user does not like a page cannot be taken as expressing positive or negative preference. Similar to the observations about movies in Hu, Koren, and Volinsky [5], it is possible that the user has not seen the page, is unfamiliar with the topic the page is promoting, or even enjoys what the page is promoting but does not want to make the information public. Even a page like on Facebook may not indicate approval; one could imagine liking the page of their representative to follow local news even if they did not vote for the politician in question. For the purposes of this experiment, we will assume that these instances are rare and balanced out by other page likes. Without further user input, these cases would be almost impossible to handle.

2.1.1 Explicit and Implicit Feedback

This problem reveals the fundamental difference between what Hu, Koren, and Volinsky [5] term explicit and implicit feedback. Explicit feedback requires a user to indicate positively or negatively how they feel about a particular factor, for instance, Netflix star ratings. Naturally, this allows for the most accurate predictions or analyses of user preferences, since each preference is known exactly and explicitly. However, often this data is difficult to gather. For the simple “like” interface that Facebook uses, an additional user survey about liked pages would be required to get

explicit feedback. Instead, the data in this case can be thought of as implicit feedback. Implicit feedback is gathered from observations about user habits as opposed to directly asking the user. While it is generally much easier to acquire than explicit feedback, as mentioned previously, it is unable to express negative feedback. Additionally, implicit feedback is by nature noisy because it is only an observation of behavior. Page likes may not be always indicative of preference and it is possible that a page was accidentally liked or liked so long ago that it is inconsistent with the user's current views. Lastly, predictions on explicit feedback can be difficult to evaluate. The data may be very sparse and it is difficult to tell if a user has even heard of or formed an opinion on the item being predicted about. So, while implicit feedback has many drawbacks compared to explicit feedback, it is often the only data available.

2.1.2 Model-based and Memory-based Models

Regardless of the type of feedback, there are two general classes of collaborative filtering: model-based and memory-based [6, 7, 8]. Model-based collaborative filtering uses a dataset to build a model that is then run on new data to make predictions. An example of a model-based collaborative filter would be one that uses a naïve Bayes classifier, assuming that preferences expressed within a class are independent. This seems an unrealistic assumption for the data at hand. For instance, if a person likes a page about the college they attend, it is much more likely that they will like another page related to that college than someone who attends a different college. More complex models can account for this, but they still are not as well suited as memory-based algorithms for predicting on the entire dataset.

Memory-based algorithms, like the one used in this experiment, use the entire dataset to make predictions about the dataset itself. They often use correlation measures (such as Pearson correlation coefficient or cosine similarity) and inverse user frequency to determine weights between pages that appear together [6]. The main conceit of inverse user frequency is that strong relationships between infrequently appearing pages are more indicative than relationships with frequently occurring pages. This balance is usually accomplished by adding a coefficient related to how often a given page is liked within the dataset. Breese, Heckerman, and Kadie [6] show that collaborative filtering with these and other techniques (denoted CR+ in the paper) performs comparably or better to other memory-based or model-based algorithms.

2.1.3 Neighborhood and Matrix Decomposition Models

There is another distinction between types of collaborative filtering methods: neighborhood models and matrix decomposition models. The most complex collaborative filtering schemes involve combinations of both forms [4], but more commonly only neighborhood models are used [8]. Neighborhood models can be based either on users or pages, although those based on pages are used most today. As part of a memory-based algorithm, neighborhood models use correlation to gauge similarity between pages and predict the rating of a given page as the weighted sum of a fixed number of the nearest pages. This is simply a K-Nearest Neighbors (KNN) algorithm. This method also allows for a view of how similar pages are by simply exploring the similarity weights after the algorithm has concluded.

Although less commonly used than neighborhood-based filtering, matrix decomposition or matrix-factorization models generally "perform best in terms of prediction accuracy" [8]. Matrix decomposition methods break the original matrix expressing preference (in this case whether or not the page is liked by a given user) into two matrices. Using alternating ridge regression, these matrices are optimized so that their product closely resembles the original matrix. This causes previously zero-entries to take on a value, in this case the prediction for how likely the page is to be preferred by the user. See section 3.1 for more on the specific implementation used in this project.

2.2 Neural Networks

Neural networks have been at the forefront of classification for several years, popular due to their self-adaptive capabilities that require little prior knowledge about the data and nonlinearity [10]. These characteristics make neural networks incredibly flexible and suitable to a wide range of applications. Most neural networks use feed-forward processing and back propagation to train the network. Back propagation is generally associated with the sigmoid function due to its non-linearity

and convenient derivative. However, in [11], it is put forward that using artificial neurons in place of back propagation and an exponential function in place of the sigmoid can offer significant performance improvements when back propagation is a computationally heavy part of the model. Due to the small nature of this project, such adaptations were not necessary and the traditional model was used.

Another feature of the traditional model of a neural network is the number of layers. For most applications (excepting deep learning), neural networks contain only an input layer, a hidden layer, and an output layer. More layers are not generally necessary because "networks with sigmoidal non-linearities and two layers of weights can approximate any decision boundary to arbitrary accuracy" [12 pg. 130]. However, the number of hidden units, treated here as a hyper-parameter (see section 3.2.1) can have enable or prevent a neural net from behaving as a universal approximator [13]. Specifically, in [14], the authors show that two hidden layers can hinder classification performance compared to a single hidden layer. More complex approaches may benefit from additional layers, but the single hidden layer method is sufficient for the purposes of this project.

3 Methods Used

3.1 Collaborative Filtering

The collaborative filtering method used is a modification of the code presented in [15], a worksheet from the previous version of this course. The ideas used in the worksheet are from [5]. To summarize, a preference variable (p_{ui}) is assigned to each user and page pair. This is the value of the matrix at the pair: 1 if the page has been liked, 0 if it has not. A weight for each pair is also assigned. In this case each weight (c_{ui}) is the absolute value of the corresponding p_{ui}. This choice was made because there is no other information about a user's preference beyond whether or not the page is liked. Then, an f x n matrix of factors (Y) is created, with each column corresponding to the features of a page, where f is a hyper-parameter and n is the number of pages. An m x f matrix (X) has rows that correspond to each user, with f again being the number of features and m the number of users. The product of these two matrices will be the m x n output matrix of the collaborative filtering, and should approximate the original matrix. Alternating ridge regression approximates the matrices X and Y. The factors for each user are calculated by minimizing the following loss function:

$$L(\boldsymbol{x}_u) = \sum_{i=1}^{n} c_{ui}(p_{ui} - \boldsymbol{x}_u \boldsymbol{y}_i)^2 + \lambda \left|\left|\boldsymbol{x}_u\right|\right|_2^2$$

where $\boldsymbol{x_u}$ is a row vector of X, $\boldsymbol{y_i}$ is a column vector of Y, and λ is the regularization coefficient [15].

3.1.1 Fitting Hyper-parameters

The method described in section 3.1 has two hyper-parameters: the number of factors (f) and the regularization coefficient (λ). The optimal value for both of these must be determined empirically. To determine which values were optimal, a measure of scoring a collaborative filter had to be developed. For this project, two factors were considered: the number of confirmed preferences less than one in the new matrix and the number of users who have a minimum preference of over 0.5. For the first part, every value in the new matrix was compared to its corresponding entry in the original. If the original had a zero or the original had a one and the new a value greater than one, the entry did not affect the score. If the original had a one and the new a value less than one, the difference between the old and new values was added to the score. This is a variation on Least Absolute Deviation and attempts to measure how well the new matrix maintains the known preferences of the original. For this reason, preferences above one were not penalized. The second part tries to maintain a spread of the data and make sure that the algorithm does not simply assign users with many page likes a high probability of liking *every* page. The algorithm was run until the entries of the matrix appeared to converge, for a minimum of 100 iterations and a maximum of 5000 iterations. Convergence was determined by comparing the sum of the entries of both X and Y to the previous iteration. If the difference was less than a certain

threshold for five iterations, the algorithm was deemed to have converged. See CFFunctions.py for the exact implementation. The number of factors tested ranged from 2 to 256 by powers of 2 and 10^{-5} to 10^{5} by powers of 10 for the regularization coefficient. The following results were generated from running CFHyperParameters.py and reformatting the data to a spreadsheet. See CFHyperData for the raw data.

Hyper-parameters for Collaborative Filtering

	Regularization Coefficient										
Number of Factors	10^-5	10^-4	10^-3	10^-2	10^-1	1	10	10^2	10^3	10^4	10^5
Error:											
2	0.165608646	47.71952241	11.89624847	8.218915258	87.21714964	696.5717112	5485.090113	13068	13068	13068	13068
4	0.141144939	0.398086751	1.283671511	7.802457697	74.83828449	683.8579727	5485.090135	13068	13068	13068	13068
8	0.199567995	0.229106553	0.776841836	7.081367025	69.32423103	683.8517321	5485.090163	13068	13068	13068	13068
16	0.358562887	0.677756481	1.323318363	6.825574417	69.01892977	683.8406528	5485.089906	13068	13068	13068	13068
32	0.00456282	0.050232896	0.650988428	6.722402467	68.96445406	683.8318817	5485.089895	13068	13068	13068	13068
64	0.001155013	0.021107455	0.621922871	6.751853362	68.97400436	683.8274969	5485.089876	13068	13068	13068	13068
128	0.000737448	0.020719448	0.520252215	6.751977417	68.97355167	683.823852	5485.090026	13068	13068	13068	13068
256	0.001356355	0.018696552	0.60331337	6.752583324	68.97865864	683.8223542	5485.08971	13068	13068	13068	13068
Number of Minimums Above 0.5:											
2	121	0	0	203	0	147	0	0	0	0	0
4	0	0	200	157	169	140	0	0	0	0	0
8	0	0	176	183	186	140	0	0	0	0	0
16	0	0	0	180	186	140	0	0	0	0	0
32	0	0	112	174	186	140	0	0	0	0	0
64	0	0	122	178	186	140	0	0	0	0	0
128	0	0	7	178	186	140	0	0	0	0	0
256	0	0	97	183	186	140	0	0	0	0	0

Based on these results, it appears that the combination of 128 factors and a regularization coefficient of 10^{-5} performs the best with these metrics. One could of course argue for other methods of evaluating the performance of the collaborative filter, which may yield different results, but the values of 128 and 10^{-5} were used for the rest of this project. Note that the regularization coefficient here is very low, meaning that this evaluation method may be prone to overfitting. Furthermore, 13,068 is the number of observations in the dataset, meaning that for large values of the regularization coefficient, the resulting matrix must be the zero matrix.

3.2 Neural Network

The implementation of a neural network for this experiment is the canonical general-purpose neural network. That is, it has one hidden layer that uses a sigmoid activation function and an output layer that uses the softmax function. Both layers are fully connected and the network trains using feed forward processing and back propagation. The neural network was created using the library Keras with the TensorFlow backend. The network was trained using the binary cross entropy loss function and the Adam optimizer. The Adam optimizer is a version of stochastic gradient descent [16]. The binary cross entropy function was most efficiently computes the error for the two class system relevant for this project. In addition, the training makes use of a built-in L2 regularization.

3.2.1 Fitting Hyper-parameters

As with collaborative filtering, the neural network has two hyper-parameters: the number of nodes in the hidden layer (f) and the regularization coefficient. To learn these parameters, a cross-validation method was used. First, 15% of the data was set aside as the testing data. Then, for each pair of parameters, the data was shuffled and 10% of the training data was set aside for validation. Two neural networks were trained, one using the original data and one using the filtered data. They were then tested on the validation data. The accuracy, in terms of a percentage of validation data labeled correctly, was saved. This process was repeated 10 times for each pair of hyper-parameters and the average of the 10 trials was reported. The number of hidden nodes range from 2 to 1024 as powers of 2 and the regularization parameter varies from 10^{-5} to 10^{5} by powers of 10. The results included below come from running NNHyperParameters.py and placing the data in a table. The raw data is available in NNHyperData.

Unfiltered Data

	Reg Parameter										
Output Dim	10^-5	10^-4	10^-3	10^-2	10^-1	1	10	10^2	10^3	10^4	10^5
2	55.29%	58.82%	51.76%	62.94%	54.12%	50.00%	60.59%	52.35%	57.65%	54.12%	51.76%
4	56.47%	55.29%	55.88%	56.47%	53.53%	54.71%	54.12%	52.35%	48.24%	52.35%	57.06%
8	59.41%	58.82%	58.82%	60.00%	58.82%	48.82%	53.53%	42.94%	55.88%	51.76%	54.12%
16	52.94%	62.35%	54.71%	55.29%	57.06%	59.41%	54.71%	52.94%	42.94%	48.24%	49.41%
32	54.12%	52.94%	56.47%	60.00%	57.06%	62.94%	52.94%	52.35%	45.29%	48.24%	53.53%
64	59.41%	51.76%	61.76%	56.47%	61.76%	62.35%	45.29%	59.41%	47.65%	47.65%	57.06%
128	65.88%	54.71%	58.24%	56.47%	52.35%	57.06%	60.59%	41.18%	48.82%	46.47%	56.47%
256	58.24%	62.94%	61.18%	57.65%	57.06%	58.82%	48.82%	57.65%	42.94%	47.06%	49.41%
512	67.65%	62.35%	64.12%	59.41%	51.76%	68.24%	55.29%	48.82%	50.59%	51.76%	45.29%
1024	65.88%	71.18%	74.12%	59.41%	49.41%	60.00%	60.00%	48.24%	48.82%	54.71%	52.35%

Filtered Data

	Reg Parameter										
Output Dim	10^-5	10^-4	10^-3	10^-2	10^-1	1	10	10^2	10^3	10^4	10^5
2	56.47%	57.65%	60.00%	61.18%	58.82%	55.29%	54.12%	55.88%	58.82%	48.82%	48.24%
4	55.88%	55.29%	55.29%	56.47%	60.00%	64.71%	53.53%	52.35%	54.12%	48.24%	54.71%
8	59.41%	60.00%	58.82%	60.00%	60.59%	57.65%	52.35%	58.82%	55.29%	50.59%	51.76%
16	52.94%	62.35%	54.12%	55.29%	57.06%	57.65%	56.47%	58.24%	56.47%	47.65%	51.76%
32	57.06%	52.35%	51.18%	60.00%	58.24%	61.18%	48.82%	54.71%	54.12%	44.12%	55.29%
64	65.29%	55.88%	68.24%	63.53%	61.76%	62.35%	52.35%	47.06%	48.24%	49.41%	42.35%
128	67.06%	64.12%	65.88%	63.53%	52.94%	57.06%	60.00%	50.00%	51.76%	46.47%	48.24%
256	64.12%	65.88%	67.06%	61.76%	58.24%	58.82%	56.47%	44.12%	60.59%	44.71%	45.88%
512	61.18%	55.29%	65.29%	62.94%	64.71%	68.24%	52.94%	55.29%	47.65%	50.59%	48.82%
1024	48.82%	57.65%	61.76%	53.53%	64.71%	60.00%	61.18%	50.00%	45.88%	58.24%	50.59%

As we can see from these charts, the unfiltered data performs the best with a regularization parameter of 10^{-3} and 1024 hidden nodes, while the filtered data does best with a regularization parameter of 10^{-3} and 64 hidden nodes. Note that another pair does equally well for filtered data: regularization parameter equal to 1 and 512 hidden nodes. The decision was made to go with the first instance because it aligns more closely with the unfiltered parameters, which we would expect to be similar.

4 Experiments and Data

As explained previously, the data was gathered from Facebook using a Google Chrome extension and then saved to csv files. The data was then filtered and formatted to prepare it for collaborative filtering and further analysis. For a full discussion of data gathering, preparation, and formatting, see Appendix A.

After fitting the hyper-parameters as described in sections 3.1.1 and 3.2.1, the experimentation itself was straightforward. The unfiltered data was read from the csv files and prepared as described in 4.1.2. The filtered data was read from a file that had been created by applying collaborative filtering using the chosen hyper-parameters. Both the filtered and unfiltered data were shuffled with 15% set aside for testing. The rest was sent to train the neural network using the established hyper-parameters. The trained neural network was then tested on the set aside data and the confusion matrices in section 5 were created.

5 Results

To generate these tables, the models were each run 300 times, with a random set of testing and training data each time. The size of each set did not change. The overall accuracy and data for the confusion matrices were stored each time and at the end the mean and standard deviation were taken. The results are below. For ease of reading, accuracies have been converted to percentages. For the raw data in table form and for standard deviations, see Appendix B. For the raw results see the file Results1.

5.1 Without Collaborative Filtering

Original Means

	Actually Conservative	Actually Liberal
Predicted Conservative	14.37%	19.35%
Predicted Liberal	28.14%	38.14%

Overall Accuracy: 52.51%

Note that despite the fact that 43% of the data was labeled conservative, the algorithm predicted nearly two-thirds of the test set to be liberal. Additionally, close to one-fifth of the users labeled conservative were actually liberal.

5.2 With Collaborative Filtering

Filtered Means

	Actually Conservative	Actually Liberal
Predicted Conservative	10.27%	13.92%
Predicted Liberal	32.24%	43.57%

Overall Accuracy: 53.84%

The trends observed in the original data can also be observed here. The algorithm predicted over three-fourths of the data to be liberal and seemed to make its meager gain over the original data simply by predicting liberal more frequently. Neither of these methods performs much better than random. It is worth noting that the collaborative filtering data had lower standard deviations for each category, but a similar standard deviation with respect to overall accuracy.

6 Conclusions

The results shown in section 5 seem to show that both prediction methods only perform slightly better than random. It may indicate a problem when choosing hyper-parameters: perhaps 10 iterations of cross-validation were insufficient and susceptible to outliers. A shuffle that happened to perform particularly well or poorly could have skewed the results for the entire hyper-parameter. Unfortunately, time constraints would not allow for further testing.

Ultimately, the method likely performed poorly due to lack of data. With more users and preferences, a better network would have been trained and the classification probably would have been more accurate. Neural networks require large amounts of data to train properly and that data was simply not available in this case. The results may also have been helped by choosing a higher

or lower threshold for minimum number of likes of a page in the dataset. A lower threshold may have allowed for a better analysis of rare relationships or it could simply have contributed more noise. A higher threshold could have eliminated some noise, but may have lost distinguishing groups of pages.

7 Future Work

7.1 Improving Existing Methods

The best way to improve the current model is a much larger dataset. The one used in this project contained only around 200 users and 2200 unique pages. The small dataset did have a few advantages: it is relatively evenly split between political factions (57% liberal), it is small enough to do computations on relatively quickly, and it is small enough to hand-label each user. However, it is far too small and limited in scope to make predictions about a user that is outside of this network. It is worth noting that perhaps such models would perform better when limited to analyzing on a certain demographic as more general data may be too noisy to perform accurately in specific cases. Regardless, this model would benefit greatly from more data.

Another important expansion of the current model would be to label the data automatically. This might be easy in some cases, such as individuals who have their political preference explicitly in their profile, but almost impossible in others. However, it would immeasurably improve the scalability of the project. Finding a way to access Facebook's political label for targeted ads would be perfect for this automated labeling. A similar expansion would be to use post likes or comments in addition to pages to generate more current data. This would come with its own problems, such as figuring out if a comment is positive or negative in relation to the parent post. Ideally one could even gather explicit feedback by asking users direct questions, but this would be difficult to implement on a large scale. A more efficient way to prune pages would also be useful. Perhaps by looking at the total number of likes on a page or the date of the page's most recent activity one could eliminate irrelevant and inactive pages from the analysis. This might run the risk of losing valuable information, as a campaign page from the 2008 election may no longer be active but would still reveal a political leaning. It would speed up the computations and reduce noise if done properly. Even testing the different thresholds in this dataset could improve performance.

Lastly, more specific political labels could be used. It is impossible to summarize the nuance of hundreds of political views in a binary label. By allowing for third parties, a more fitting political picture might be gathered. However, this would require a much larger dataset and an intimate knowledge of what each party stood for. It might also lose some generality, as the broad labels of "liberal" and "conservative" can be carried across nations and regions whereas political parties in different states may have different goals and policies.

7.2 Using Alternative Methods

There are a few methods other than the collaborative filtering and neural network that might be well suited to this problem. Perhaps most promising is unsupervised learning. Clustering would reveal interesting relationships between the data, likely beyond simple politics. It would also make the nuance of additional parties easier to handle. It is possible that unsupervised learning methods would tend to cluster based on factors other than political belief, such as location or school affiliation for this dataset. Seeding the data, as described in [9], and using semi-supervised learning could solve this problem. By using for seeds profiles that have many representative page likes for a political affiliation, the clustering could be steered towards political leaning. Comparing mutual friends to clustered data might be a starting point for validation as well.

Using a different collaborative filtering model, such as one of those described in [4], might provide better results, although [5] seems to suggest that the method used is ideal for the dataset. Comparing the current method with a nearest neighbors method may be worthwhile anyway. Lastly, instead of using a neural net, one could use a Naïve Bayes Classifier (NBC). As mentioned in 2.1.2 of this paper, it is unlikely that pages are uncorrelated within a class. Regardless, it is possible that a NBC would behave well in practice despite its simplifying assumption, especially on this small dataset, and so it might be worth comparing to a neural network in this case.

References:

(1) "Nixon Declares 'Silent Majority' Backs His Speech." *New York Times*, 5 November 1969, p. 1 col. 1. Web. http://query.nytimes.com/mem/archive-free/pdf?res=9F00EEDD1E3AEF3BBC4D53DFB7678382679EDE. Accessed 2 March 2017

(2) He, Sunny, Zachary Liu, Vivian Mo, and Jonathan Zong. *PolitEcho*. politecho.org. Accessed 2 March 2017.

(3) Merrill, Jeremy B. "Liberal, Moderate or Conservative? See How Facebook Labels You." *New York Times*, 24 August 2016, p. A13. Web. https://www.nytimes.com/2016/08/24/us/politics/facebook-ads-politics.html?_r=1. Accessed 2 March 2017.

(4) Töscher, Andreas, Michael Jahrer, and Robert M. Bell. *The BigChaos Solution to the Netflix Grand Prize*. Web. http://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf. Accessed 2 March 2017.

(5) Hu, Yifan, Yehuda Koren, and Chris Volinsky. *Collaborative Filtering for Implicit Feedback Datasets*. *IEEE International Conference on Data Mining*, p. 263-272, 2008. Web. http://yifanhu.net/PUB/cf.pdf. Accessed 2 March 2017.

(6) Breese, John S., David Heckerman, and Carl Kadie. *Empirical Analysis of Predictive Algorithms for Collaborative Filtering*. *UAI '98 Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence,* p. 43-52. Web. http://dl.acm.org/citation.cfm?id=2074100. Accessed 2 March 2017.

(7) Sarwar, Badrul, George Karypis, Joseph Konstan, and John Riedl. *Item-Based Collaborative Filtering Recommendation Algorithms. WWW '01 Proceedings of the 10th International Conference on the World Wide Web*, p. 285-295. Web. http://dl.acm.org/citation.cfm?id=372071. Accessed 2 March 2017.

(8) Lee, Joonseok, Mingxuan Sun, and Guy Lebanon. *A Comparative Study of Collaborative Filtering Algorithms*. *The Ohio State University*, 14 May 2012. Web. http://www.stat.osu.edu/~dmsl/Lee_2012.pdf. Accessed 28 February 2017.

(9) Baso, Saguto, Arindam Banerjee, and R. Mooney. *Semi-supervised Clustering by Seeding*. *19th International Conference on Machine Learning (ICML 2002)*. Web. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.7.9416. Accessed 28 February 2017.

(10) Zhang, Guoqiang Peter. *Neural Networks for Classification: A Survey*. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 30, issue 4, Nov 2000. Web. http://ieeexplore.ieee.org/abstract/document/897072/. Accessed 3 March 2017.

(11) Specht, Donald F. *Probabilistic Neural Networks*. *Neural Networks,* vol. 3, issue 1, 1990, p. 109-118. Web. http://www.sciencedirect.com/science/article/pii/089360809090049Q. Accessed 3 March 2017.

(12) Bishop, Christopher M. *Neural Networks for Pattern Recognition*. Oxford; Clarendon Press: Oxford University Press, 1996. Print.

(13) Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. *Multilayer Feed-forward Networks are Universal Approximators. Neural Networks*, vol. 2, p 359-366, 1989. Web. http://www.sciencedirect.com/science/article/pii/0893608089900208. Accessed 3 March 2017.

(14) Villiers, Jacques de and Etienne Bernard. *Backpropogation Neural Nets with One and Two Hidden Layers*. *IEEE Transactions on Neural Networks*, vol. 4, no. 1, January 1992. Web. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=182704. Accessed 3 March 2017.

(15) Fritas, Nando de. "Exercise Sheet 2." *Machine Learning*, course at Oxford University 2015. 5 February 2015. Web. https://www.cs.ox.ac.uk/teaching/materials14-15/ml/class2.pdf. Accessed 3 March 2017.

(16) Kingma, Diederk P and Jimmy Lei Ba. *Adam: A Method for Stochastic Optimization*. 3rd *International Conference for Learning Representations*, San Diego, 2015. Web. https://arxiv.org/pdf/1412.6980.pdf. Accessed 5 March 2017.

Appendix A: The Data

A.1 Gathering the Data

The data used for this project was gathered from Facebook using a Google Chrome extension. Facebook's Graph API did not offer access to a complete friends list or a list of page likes of a user's friends. Instead, the custom Google Chrome extension opens the necessary pages of Facebook and scrapes the relevant data from the htmls. The user must be signed into Facebook for this extension to run properly. This is a slow process but only needs to be run once. The extension saves a stack consisting of a friend name, a friend href, a page name, and a page href and exports the entire stack to one or more csv files. It also provides periodic updates to inform the user of its progress and when it has finished. To run the extension, simply open the folder "Extension" in the Developer Mode in Google Chrome Extensions. Click on the American flag and the program will run automatically.

The extension gathers data from the categories of books, movies, television shows, music, and other. The "other" category includes any political pages. Note that some friends may not be represented in the data, either because they do not have any page likes or because they have deleted their account. The categories of books, movies, and television shows may all be missing one entry per user due to a change in Facebook's style for those pages. It was not considered worthwhile to adapt the code to include this small amount of data. Additionally, commas from page names were removed because they interfered with reading the csv files to which they were exported. Again, this was not considered a significant alteration.

A.2 Preparing the Data

Before analyzing the data with collaborative filtering, a few more steps were taken to clean up and format the data. First, pages with the same page name but different page hrefs were combined. This decision was made from the describe function in pandas, which revealed the most common page name had more appearances than the most common page href. This is likely because sometimes people like an unofficial page unintentionally. For the purposes of this project, the official and unofficial pages indicate the same preference and so they were combined. Users who had liked different pages with the same name but different hrefs were reduced to liking only one page with the common name under the same principal. Lastly, a decision was made to discard pages with few appearances in the data. To determine the best threshold, the following chart was created using the pandas describe function:

Data by Page Likes in Common

Minimum Number of Occurrences	Number of Unique Friends	Number of Unique Pages	Total Number of Pages
1	219	16,543	29,216
2	215	4,023	16,696
3	211	2,209	13,068
4	210	1,454	10,803
5	207	1,030	9,107
6	206	746	7,687
7	203	562	6,583
8	203	436	5,701
9	203	337	4,909
10	203	269	4,297
11	203	223	3,837

As indicated by the highlighted row, the choice was made to eliminate pages with fewer than three appearances in the data. This number was chosen as a compromise between keeping a good deal of unnecessary and distracting data and eliminating potentially useful pages with few likes. This threshold would almost certainly have been chosen differently had the dataset been larger and less sparse. Arguments could be made for other thresholds in this dataset, but for the purposes of this project, the indicated numbers are used throughout.

A.3 Labeling the Data

For this dataset, each profile was hand labeled using a one-hot encoding method where the first class is conservative and the second is liberal. To determine the label, each profile was looked at individually. If the user indicated their political party explicitly, that was used. Otherwise, a combination of the pages liked, posts by the user, and first-hand knowledge of the user was used to assign a label. This process was somewhat subjective and would be nearly impossible to scale. For a few suggestions on how to improve this, see section 7.1.

Appendix B: Raw Results

B.1 Original Data

Original Mean

	Actually Conservative	Actually Liberal
Predicted Conservative	0.143687707641196	0.193521594684385
Predicted Liberal	0.281353820598006	0.381436877076411

Overall Accuracy: 0.525124584717607

Original Standard Deviations

	Actually Conservative	Actually Liberal
Predicted Conservative	0.105033816647761	0.140504677209418
Predicted Liberal	0.116440158202458	0.148042639438468

Overall Accuracy: 0.089399964567636

B.2 Filtered Data

Filtered Mean

	Actually Conservative	Actually Liberal
Predicted Conservative	0.102678571428571	0.139223421926910
Predicted Liberal	0.322362956810631	0.435735049833887

Overall Accuracy: 0.538413621262458

Filtered Standard Deviations

	Actually Conservative	Actually Liberal
Predicted Conservative	0.0664117570617753	0.0788565113143011
Predicted Liberal	0.0852686351933445	0.1052449019878520

Overall Accuracy: 0.0876730373352650