TECHNICAL UNIVERSITY OF CRETE, GREECE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Using Recommendations to Reduce Opinion Polarization in Social Networks

Chochlakis Ioannis

Thesis Committee

Professor Thrasyvoulos Spyropoulos (ECE), Supervisor
Professor Georgios Chalkiadakis (ECE)
Professor Michail G. Lagoudakis (ECE)

CHANIA, FEBRUARY 2025

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ, ΕΛΛΑΔΑ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Χρήση Συστάσεων για τη Μείωση της Πόλωσης Απόψεων στα Κοινωνικά Δίκτυα

ΧΟΧΛΑΚΗΣ ΙΩΑΝΝΗΣ

Εξεταστική Επιτροπή

Καθηγητής Θρασύβουλος Σπυρόπουλος (ΗΜΜΥ), Επιβλέπων
Καθηγητής Γεώργιος Χαλκιαδάκης (ΗΜΜΥ)
Καθηγητής Μιχαήλ Γ. Λαγουδάκης (ΗΜΜΥ)

ΧΑΝΙΑ, ΦΕΒΡΟΥΑΡΙΟΣ 2025

ABSTRACT

In recent years, social media platforms have become a battleground for sociopolitical discourse. Once a herald of global connection, social networks are now a key factor in amplifying political and ideological divides around the globe. Although the internet offers an abundance of information, from diverse perspectives, individuals often gravitate towards and connect with others who share similar opinions. This "natural" segregation of social networks is made worse by malicious actors, such as bots and misinformation agents, exploiting users' psychological tendencies (confirmation bias) in order to further their own goals. The personalization of recommenders also played a key role in furthering this divide among online communities. Individuals are funneled into insulated groups or "echo chambers" where preexisting beliefs get reinforced and diverse viewpoints get limited exposure, with the goal of maximizing engagement on the platform and thus profit. The consequences of this algorithmic bias extend beyond the social network space and pose a threat to our society as a whole, dividing us slowly and steadily. In this thesis, we focus on the role of the recommender inside of the social network and attempt to create a recommendation system that values not only user engagement, but also the reduction of polarization and the disruption of echo chambers inside the network. First, we provide a detailed explanation of how we create a polarized social network space, modeled after the social platform X, where each user inside the network has an opinion along with a set of users they follow. Next, we introduce our model of asynchronous recommendation-driven opinion evolution, explaining how suggested content can influence users' opinions in our network. Additionally, we explain how in this model, a group of users may be left completely segregated, unable to be influenced by other users. Next, we define our proposed metric for measuring polarization that will be used in order to evaluate our proposed system's success. We then explain how we will be adapting this problem to a Deep Reinforcement Learning framework in order to properly train and create agents that will act as recommenders, without any knowledge of the model parameters. This means our implementation can be used with other opinion evolution models and polarization metrics, as long as the necessary Reinforcement Learning modeling parameters are properly defined. We then define our state space, action space, the reward function and the terminal states for our proposed Reinforcement Learning Model. Finally, we conduct a set of experiments, on small polarized network communities, to evaluate if our proposed solution can indeed learn to reduce polarization. Our results confirm the effectiveness of our approach.

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια, οι πλατφόρμες κοινωνικής δικτύωσης έχουν γίνει πεδίο μάχης για την κοινωνικοπολιτικές αντιπαραθέσεις. Από την αρχική τους υποσχόμενη παγκόσμια διασύνδεση, τα μέσα κοινωνικής δικτύωσης έχουν αλλάξει ρόλο και στην τωρινή εποχή ενισχύουν τις πολιτικές και ιδεολογικές διαφορές. Αν και το Διαδίκτυο προσφέρει πληθώρα πληροφοριών από διαφορετικές οπτικές γωνίες, τα άτομα συχνά έλκονται και συνδέονται με άλλους που μοιράζονται παρόμοιες απόψεις. Αυτός ο «φυσικός» διαχωρισμός των κοινωνικών δικτύων επιδεινώνεται από κακόβουλους παράγοντες, όπως bots και κακόβουλοι χρήστες, που εκμεταλλεύονται τις ψυχολογικές τάσεις των χρηστών (confirmation bias) για να προωθήσουν τους δικούς τους στόχους. Η εξατομίκευση των αλγορίθμων συστάσεων έπαιξε επίσης βασικό ρόλο στην προώθηση αυτού του χάσματος μεταξύ των διαδικτυακών κοινοτήτων. Οι χρήστες διοχετεύονται σε μονωμένες ομάδες ή αλλιώς «echo chambers», όπου οι προϋπάρχουσες πεποιθήσεις ενισχύονται και οι διαφορετικές απόψεις έχουν περιορισμένη έκθεση, με στόχο τη μεγιστοποίηση της δέσμευσης στην πλατφόρμα και συνεπώς του κέρδους. Οι συνέπειες αυτής της αλγοριθμικής προκατάληψης εκτείνονται πέρα από τον χώρο των κοινωνικών δικτύων και αποτελούν απειλή για την κοινωνία μας στο σύνολό της, διαχωρίζοντας μας αργά και σταθερά. Στην παρούσα διπλωματική εργασία, εστιάζουμε στον ρόλο των αλγορίθμων συστάσεων που βρίσκονται στο εσωτερικό του κοινωνικού δικτύου και προσπαθούμε να δημιουργήσουμε ένα σύστημα συστάσεων που εκτιμά όχι μόνο τη δέσμευση των χρηστών αλλά και τη μείωση της πόλωσης και την εξάλειψη των «echo chamber» μέσα στο δίκτυο. Αρχικά, παρέχουμε μια λεπτομερή εξήγηση του τρόπου με τον οποίο δημιουργούμε ένα πολωμένο κοινωνικό δίκτυο, διαμορφωμένο σύμφωνα με την κοινωνική πλατφόρμα X, όπου κάθε χρήστης μέσα στο δίκτυο έχει μια γνώμη μαζί με ένα σύνολο χρηστών που ακολουθεί. Στη συνέχεια, παρουσιάζουμε το μοντέλο μας για την ασύγχρονη εξέλιξη γνώμεων βάσει συστάσεων, εξηγώντας πώς το προτεινόμενο περιεχόμενο μπορεί να επηρεάσει τις απόψεις των χρηστών στο δίκτυό μας. Επιπλέον, εξηγούμε πώς σε αυτό το μοντέλο μια ομάδα χρηστών μπορεί να μείνει εντελώς διαχωρισμένη, χωρίς να μπορεί να επηρεαστεί από άλλους χρήστες. Ύστερα, ορίζουμε την μετρική για τη μέτρηση της πόλωσης που θα χρησιμοποιηθεί στην αξιολόγηση του προτεινόμενου συστήματος. Στη συνέχεια εξηγούμε πώς θα προσαρμόσουμε αυτό το πρόβλημα σε ένα πλαίσιο Βαθιάς Ενισχυτικής Μάθησης (Deep Reinforcement Learning) προκειμένου να εκπαιδεύσουμε σωστά και να δημιουργήσουμε πράκτορες που θα λειτουργούν ως πάροχοι συστάσεων, χωρίς καμία γνώση των παραμέτρων του μοντέλου. Αυτό σημαίνει ότι η υλοποίησή μας μπορεί να χρησιμοποιηθεί με άλλα μον-

τέλα εξέλιξης απόψεων και μετρήσεις πόλωσης, εφόσον καθοριστούν σωστά οι απαραίτητες παράμετροι μοντελοποίησης Βαθιάς Ενισχυτικής Μάθησης. Στη συνέχεια, ορίζουμε τον χώρο κατάστασής, τον χώρο δράσης, τη συνάρτηση ανταμοιβής και τις τερματικές καταστάσεις για το προτεινόμενο Μοντέλο Ενισχυτικής Μάθησης. Τέλος, διεξάγουμε ένα σύνολο πειραμάτων, σε μικρές πολωμένες κοινότητες, για να αξιολογήσουμε εάν η προτεινόμενη λύση μας μπορεί πράγματι να μειώσει την πόλωση. Τα αποτελέσματά μας επιβεβαιώνουν την αποτελεσματικότητα της προσέγγισής μας.

Acknowledgements

First and foremost, I would like to thank my supervisor, Professor Thrasyvoulos Spyropoulos, for his guidance and trust throughout this thesis. Second, I wish to thank the members of the committee, Professor Michail G. Lagoudakis and Professor Georgios Chalkiadakis. Finally, I wish to thank all of my friends, family and my girlfriend for their unconditional support, not only throughout my studies, but also in every part of my life.

List of Contents

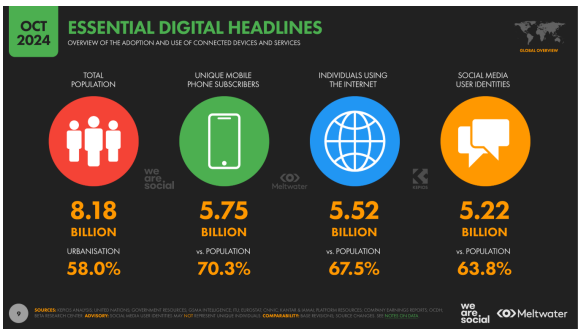
List of Figures

Chapter 1

Introduction

1.1 Motivation

As of October 2024 social media platforms have taken over the world, amassing over 5.22 billion users across all platforms [2]. The rise of social media/social networks has led to unprecedented changes in the scale and speed with which people share information. Due to the efficiency that social networks have provided us, the ways that people share / read news have changed substantially, with social media feeds becoming key tools for accessing high volumes of news, opinions, and public information. People, since the early 2010's, have started to use social networks as a new method of receiving information about the things that they are most interested in, those that are very popular in society and to just relax and pass time. A survey conducted by Pew Research Center [3] shows that a little more than half of all U.S American adults sometimes receive news from social media and 18 percent prefer to receive their news in that way. The share of Americans who prefer social media has increased by 6 percentage points since 2023.



It has been argued that the increasing use of social media for information consumption as well as discussion on various topics, has also been leading to increased polarization. However, not all relevant research finds a connection between the increased use of social media and polarization[4]. Instead, there are numerous factors that contribute to this phenomenon, both personal and algorithmic.

1.1.1 Community Creation

Firstly, social networks, both real and online, tend to be structured into communities of like-minded people. This is because individuals experience positive feelings when presented with information which confirms that their beliefs or decisions are correct. This behavior, known as *cognitive dissonance*, leads to users reducing their exposure to diverse viewpoints. Additionally, *homophily*, which is defined as the tendency of individuals to associate and bond with others who are similar to themselves, leads to users connecting and interacting mostly with other individuals that share similar opinions, thus perpetuating the creation of communities with only like-minded individuals.

1.1.2 Selective exposure, the function and role of the recommender.

Confirmation bias[5], which is the tendency to favor information in a way that confirms or supports one's prior beliefs or values and a similar phenomenon called *selective exposure*[6], affect the type of content that users inside of a social network interact with. People are less likely to read an article that supports an opposing view or interact with other users in the network that have an opposite viewpoint. These phenomena ultimately lead to biased news consumption, increasing polarization between different communities and can be exploited by malicious users (bots, misinformation agents) [1] in an attempt to further their own goals.

Recommenders on the other hand, aim to make sure the user is as engaged as possible, in order to maximize the time each individual spends inside the social network(thus maximizing profits through advertisements inside the platform). In order to keep users engaged, social media recommenders (algorithms) must make recommendations pertaining to each individual user's preferences. Based mainly on prior interactions with posts (likes on X(Twitter)/Facebook, shares/retweets, replies) and connections inside the network, social media platforms can predict with what content a

user is more likely to engage with and thus what content they should present to each individual user.

As we previously mentioned, individuals are most likely to form connections with other like-minded people and prefer to interact with content that reinforces prior beliefs. Recommendation algorithms, capitalizing on the confirmation bias of the users, will quickly learn what each user agrees with and recommend content that is consistent with that opinion, possibly further polarizing the user rather than diversifying recommendations (which could be beneficial for the individual and society, but not necessarily for the social network engagement or profit). This is called the "filter bubble" phenomenon. At the same time, in many social media, the recommender algorithm will mostly (or exclusively) suggest posts among connected nodes/friends [7]. If the user is indeed part of a community with mostly other people of similar opinions, this further exacerbates the problem leading to echo chambers often observed in social media [8,9,27].

The combination of the above phenomena, as well as a number of other sociological and psychological factors in play, have arguably led to an increased polarization in society on various controversial as well as highly important topics.

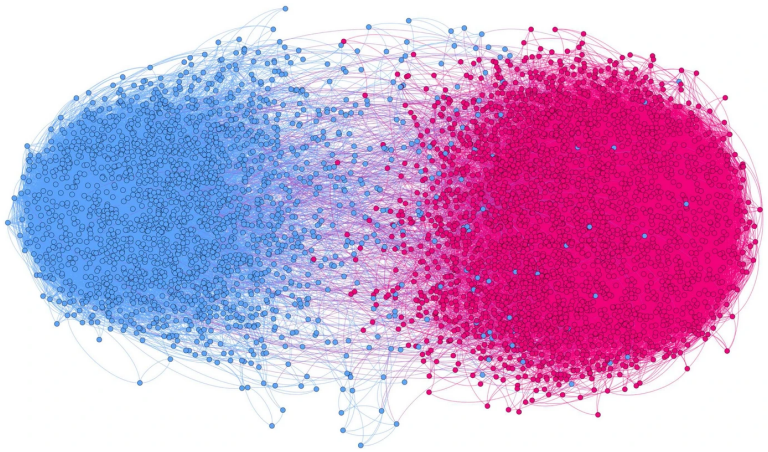


Figure 1.1: Example of a polarized and segregated network on Twitter. The network visualizes retweets of political hashtags from the 2010 US midterm elections. The nodes represent Twitter users, colors represent political preference: red for conservatives and blue for progressives.(Image taken from [10])

1.2 Countermeasures

Given the many potential detrimental effects of this polarization, a number of researchers have attempted to combat this phenomenon, by coping with different aspects of the problem.

1.2.1 Tackling the psychological/sociological aspects of the problem

A number of studies have argued that, in addition to the existence of homophily and communities in most social networks and the potentially aggravating impact of recommenders, the user itself is to blame for the problem due to their inherent preference to choose content that confirms their opinion rather than the opposite. As a result, many studies, especially coming from the field of (computational) sociology, psychology, political science, etc. have attempted to better understand the root causes of opinion formation and influence, incorporating these into various models and experiments. We will see some examples shortly.

Our focus is more on the technological aspects of the problem, and what could be done there to alleviate the problem. In this aspect, there are two things one could do to try to do:

1.2.2 Inter-connectivity

One approach to improve a social network is by altering its structure in order to weaken the strong community ties, through the introduction of cross-links. Real networks, like Facebook, routinely recommend new friends to existing users, in an attempt to create new connections between users. For instance, Facebook's algorithm analyzes user interactions, mutual friends, and shared interests to suggest potential new friends. This recommendation system could be leveraged by prioritizing the creation of connections between users from different clusters/communities, thus promoting diversity and combating the formation of echo chambers. By increasing the number of crossing links between clusters the platform can create a less segregated social space for its users.

1.2.3 Content Recommendations

The second way is to modify the algorithm that is responsible for showcasing posts on every user's feed/wall. More specifically let us, again, focus our attention on Facebook and how it decides which posts to recommend.

The media shown and consumed by individuals on Facebook's front page is not only influenced by what our friends share but also on how the *News Feed* algorithm sorts these posts [7]. The ranking order depends on many factors, like click-through rate, interactions and friendships. Content ranked higher, is shown at the top of the page and thus is more likely to be seen and interacted with. We can see how, with regard to our previous discussion on how individuals are more likely to interact and connect with like minded individuals, this algorithm can exacerbate the problem of community segregation.

A simple modification to this algorithm would involve inserting within relevant posts near the top of the feed, some posts which may not align with the user's beliefs and would otherwise be "hidden" due to the ranking system. This could be done by either selecting random posts from those ranked low on the ranking system or by selecting posts from communities with which the user has minimal connections/interactions with. This ensures that every user, still views the relevant content at the top of their respective feeds, while still being shown diverse content from around the social network.

The above example is quite simplistic and does not take into account users' reactions to irrelevant posts, such as less time spent on the platform due to dissatisfaction or in the case of users seeing directly opposing political views how this can lead to users becoming more polarized [26]. The most likely scenario is that users will ignore irrelevant posts altogether. One must also take into account what other users are currently doing and how each recommendation affects possible future recommendations.

This thesis focuses on the latter aspect of the recommendation problem, how recommendations made now affect possible future (and maybe better) recommendations, attempting to tackle it using the toolshed of Reinforcement Learning.

1.3 Contributions

Our contributions in this thesis are as follows. We create a simulated version of a polarized social network, where we apply a modified "asynchronous" version of the DeGroot opinion evolution model. This is where the problem of the recommender arises since in our model, opinions are updated based on the recommendations made by the recommendation system.

We attempt to fit this model in a Reinforcement Learning framework to examine if and how well a DQN based agent may assume the role of a recommender and learn to reduce polarization inside the network, whilst

also learning the opinion evolution dynamics and trade-offs that our system comes with. Our experimental results, although limited to small scenarios, showcase the potential of our approach and how recommenders that examine the state of the network before making recommendations, can help combat the problem of polarization.

Chapter 2

Background and Related Work

2.1 Background

In this thesis we are interested in creating a recommender with vision, one that can evaluate the current state of the network and select the recommendation that will prove most helpful in the long-term. The field of reinforcement learning has evolved significantly over the years, giving rise to various methods that model decision-making processes.

One of the early concepts in this domain is the Markov decision process (MDP), which provides a mathematical framework for modeling decision-making in situations where outcomes are partly random and partly under the control of a decision-maker (agent).

2.1.1 Markov Decision Process

A Markov Decision Process essentially is a 4-tuple including: a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, transition probabilities between states $P_a(\sigma, \sigma')$, and reward functions $R_a(\sigma, \sigma')$. The goal is to find a policy, that maximizes the cumulative reward over time.

Most algorithms used to calculate optimal policies for MDPs, with finite states σ and actions a, require storage for two arrays. One of the arrays contains the real values, labeled as V, while the other will contain the actions, labeled as π. When the algorithm comes to an end, π will contain the steps that make up the solution/optimal policy and $V(\sigma)$ will contain the expected discounted sum of the rewards to be earned when that policy is followed,

starting from state σ. More specifically the algorithm is made up of two distinct steps:

1. A value update : $V(\sigma) = \sum_{\sigma'} P_{\pi(\sigma)}(\sigma, \sigma')(R_{\pi(\sigma)}(\sigma, \sigma') + \gamma V(\sigma'))$
2. A policy update : $\pi(\sigma) = \arg\max_a \{P_a(\sigma, \sigma')(R_a(\sigma, \sigma') + \gamma V(\sigma'))\}$

These steps are repeated for every state until there are no further changes. In both steps, the update is calculated using prior estimations of these values with the order of the updates depending on what specific algorithm is used. The only requirement, in order to find the correct solution, is that no states are excluded from these calculations.

There exist two notable variants of the main MDP algorithm: value iteration and policy iteration. Value iteration does not calculate $V(\sigma)$ and $\pi(\sigma)$ separately rather the calculation of $\pi(\sigma)$ happens within $V(\sigma)$. Meanwhile in the case of policy iteration both steps are required. Policy iteration itself is divided into two parts, policy evaluation and policy improvement. In policy evaluation we update the values for all states for a fixed policy π until they converge (step 1). After the values converge, we move to the policy improvement step where the policy is updated (step 2). Evaluation and improvement are repeated until the policy π converges.

In dynamic programming, the framework provided by MDP's was used in order to find optimal policies. However, these methods require full knowledge of the environment's dynamics, which may not be possible in real world scenarios. This necessitated the development of methods that do not rely on prior knowledge of a specific environment model. These methods are called model-free reinforcement learning methods, learning from direct interaction with the environment, receiving feedback signals in the form of rewards. More specifically in Reinforcement Learning problems, the state of the environment is observed by an agent in discrete time slots. Based on this observation, the agent takes an action in each slot and the environment returns a feedback signal, called the reward, to indicate how good this action was.

2.1.2 Q-Learning

A significant class of model-free learning methods is Q-learning. Q represents the function that the algorithm calculates, which determines the expected reward associated with an action taken in a specific state. A "tabular" Q-learning (QL) agent maintains a table of size $|\mathcal{S}| \times |\mathcal{A}|$, for every possible state $\sigma \in \mathcal{S}$ and action $a \in \mathcal{A}$, where each table entry captures an estimate of the corresponding Q-value.

Actions Selection: Actions are chosen, by the agent, based on this $Q(\sigma, a)$ table according to the well known ϵ-greedy algorithm: With probability $1 - \epsilon$ at a given state the agent picks the configuration with the maximum predicted Q value, and with probability ϵ it picks an action randomly. The first option "exploits" the knowledge that the agent has collected in order to maximize the reward it receives, while the latter makes sure that the agent is constantly "exploring" new paths, which may lead to higher rewards.

Action Improvement: Based on the reward it observes at that round it applies a stochastic approximation step that aims to improve the estimate $Q(\sigma, a)$ for that specific state and action only.

The main advantage of tabular QL is that it converges to an optimal policy, when given "infinite" training time and a partly random policy, and thus finding the actions that maximize the long term rewards for each possible state.

However, due to the combinatorial nature of states and actions, serious scalability bottlenecks arise in all of the three key features of QL, the decision function, action improvement and action selection.

- In regards to the decision function, the size of the Q-table quickly "explodes", leading to slow convergence and high memory requirements.
- Only the value of the visited state-action pair is updated at each time step. This means that in order for action improvement to take place multiple visits are required over every state-action pair for convergence.
- The expensive maximization operations during action selection, over all possible configurations at every training step lead to slow convergence.

Algorithm 1 Q-Learning Main algorithmic steps

Step 1 (Agent): when at state σ take an action a with ϵ-greedy:

$$a = \begin{cases} \text{random } a \in \mathcal{A}, & \text{with probability } \epsilon \\ \underset{a \in \mathcal{A}}{\text{argmax}} Q(\sigma, a), & \text{with probability } 1 - \epsilon \end{cases} \tag{2.1}$$

Step 2 (Env): returns the next state σ' and reward r.
Step 3 (Agent): update the corresponding Q entry:

$$Q(\sigma, a) \leftarrow (1 - \eta) Q(\sigma, a) + \eta (r + \gamma \max_{a' \in \mathcal{A}} Q(\sigma', a')) \tag{2.2}$$

where η is the learning rate and γ is the discount factor.
Repeat steps 1-3 till termination criterion.

2.1.3 Approximate QL : Deep Q Network (DQN)

The shortcomings associated with Q-Learning, in regards to both state and action space explosion, can be circumvented if the Q-table is replaced with a function. This function labeled $Q_\theta(\sigma, a)$ is parameterized by a set of variables (features) θ, which attempt to approximate the optimal Q-table efficiently. Using Deep Neural Networks, to conduct this approximation, as they are great at approximating non-linear functions, gave rise to a new class of RL called Deep Reinforcement Learning.

DNNs are characterized by a set of hidden layers, between the input and the output layers. In the case of Deep RL, the DNN receives as input the state of the system and has in total $|\mathcal{A}|$ outputs that attempt to predict the long term reward of every possible configuration we could choose. This change allows us to now be able to use continuous values as part of the input, while also reducing the number of parameters we have to learn (when compared to the Q-table entries).

Action Selection: is performed same as before, using the ϵ-greedy algorithm with $Q_\theta(\sigma, a)$.

Action Improvement: Based on the reward r the environment returns, the agent then calculates a value called Temporal Difference (TD) error δ. δ is essentially a comparison between a prediction of the long term reward of the chosen action a and an "improved estimate" based on the reward r and the predicted remaining rewards:

$$\delta = Q_\theta(\sigma, a) - (r + \gamma \max_{a' \in \mathcal{A}} Q_\theta(\sigma', a')) \tag{2.3}$$

The latter quantity of this equation is referred to as the TD-target and is used as a proxy of the true value in approximate RL. The agent then applies

a gradient step on the parameters θ of the DNN (sometimes referred to as backpropagation), attempting to slightly "correct" them toward reducing δ and improving its Q value prediction.

$$\theta = \theta - \eta \nabla_\theta \delta^2 \tag{2.4}$$

Although this change resolves, in theory, the issue of the state space, such methods may prove unstable and fail to converge [19].

For our implementation, instead of using the well known stochastic gradient descent algorithm to update our parameters θ, we will be utilizing the ADAM (Adaptive Moment Estimation) algorithm [20].

Algorithm 2 ADAM Algorithm. $f(\theta)$=objective function, η=learning rate, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\mu_0 \leftarrow 0$, $u_0 \leftarrow 0$, $\epsilon = 10^{-8}$

for t=1 to ... do
- $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
- $\mu_t \leftarrow \beta_1 \mu_{t-1} + (1 - \beta_1) g_t$
- $u_t \leftarrow \beta_2 u_{t-1} + (1 - \beta_2) g_t^2$
- $\widehat{\mu}_t \leftarrow \mu_t / (1 - \beta_1^t)$
- $\widehat{u}_t \leftarrow u_t / (1 - \beta_2^t)$
- $\theta_t \leftarrow \theta_{t-1} - \eta \widehat{\mu}_t / (\sqrt{\widehat{u}_t} + \epsilon)$

DQN mechanisms: There are two main ideas, each targeting a specific shortcoming of the vanilla DNN-based approximation.

Experience Replay Buffer: Each transition (σ, a, σ', r) experienced by the agent, is stored in the replay buffer and can be used in a next timestep to update the DNN parameters. Hence, in Eq 2.3, we don't use anymore the current experience but rather select from the replay buffer.

Mini-batch updates: Instead of using just one sample from the memory buffer, multiple experiences are sampled, at each step. This is a standard idea in stochastic gradient descent methods to reduce variance [21].

Target Network: Instead of using the current DNN $Q_\theta(\sigma)$ for the calculation of the TD-target in Eq 2.3, we use an older version of the network $Q'_\theta(\sigma)$, that is only periodically updated. This is to remove correlations between $Q_\theta(\sigma, a)$ and the TD-target. We will be showcasing how this addition affects the performance of our agents in a later section.

The DQN implementation comes with two main advantages, (i) it can handle both arbitrarily large discrete or continuous-valued state spaces and (ii) the use of approximation methods allows for each experience (σ, a, σ', r) to improve not only the corresponding Q-table value for the pair (σ, a) but,

by affecting the weights/parameters of the approximation it can affect the predicted Q value of many other (σ, a) that share similar features. Combined with the extra mechanisms of the replay buffer, the mini-batch updates and the target network, DQNs are able to handle large problems effectively.

However DQN does not come without its own set of drawbacks. First, the size of the network's output is directly correlated to the size of the action space $\mathcal{A}$, meaning that with a combinatorial $\mathcal{A}$, the fanout of the network (and thus the number of parameters) quickly increases in large scale scenarios. There is also the max operation which is performed twice in each round once at the action selection stage, leading to higher memory requirements, and once for the calculation of the TD error δ, increasing the number of flops per round leading to delays. The second obstacle DQN faces is that uniformly sampling samples from the replay buffer is often suboptimal.

Algorithm 3 DQN Main Algorithm. The important differences from QL (Alg 1.) are highlighted in red

Step 1 (Agent): when at state σ take an action a with ϵ-greedy:

$$a = \begin{cases} \text{random } a \in \mathcal{A}, & \text{with probability } \epsilon \\ \underset{a \in \mathcal{A}}{\text{argmax}} Q_\theta(\sigma, a), & \text{with probability } 1 - \epsilon \end{cases} \tag{2.5}$$

Step 2 (Env): returns the next state σ' and reward r.
Step 3 (Agent): store transition (σ, a, σ', r) in the replay buffer $\mathcal{B}$
Step 4 (Agent): copy the policy network parameters θ to the target network θ' (only every T timesteps)
Step 5 (Agent): pick N samples from replay buffer and calculate δ_i for each sample i:

$$\delta_i = Q_\theta(\sigma_i, a_i) - (r_i + \gamma \max_{a'_i \in \mathcal{A}} Q_{\theta'}(\sigma'_i, a'_i)) \tag{2.6}$$

Then update the parameters θ based on the ADAM algorithm with objective function $\mathbb{E}_{i \sim U(\mathcal{B})}[\delta_i^2]$, where $i \sim U(\mathcal{B})$ symbolizes that experiences have been sampled from the replay buffer. Repeat steps 1 to 5 till termination criterion.

2.2 Related Work

This section is divided into two parts. In the first part we will be focusing at work that models social influence dynamics, specifically how individuals' opinions inside a social network evolve and how this evolution often leads into network segregation. In the second part, we will be examining work aimed at mitigating polarization inside social media platforms.

2.2.1 Opinion evolution.

M. Moussaïd et al. [15] conducted two controlled experiments confirming how individuals tend to exhibit significant bias towards their own opinions and thus seek out information that confirms their beliefs. Additionally their findings that individuals that hold vastly different beliefs exert minimal influence on each other, is consistent with the idea of *bounded confidence*(minimal influence between individuals when opinions are distant), an idea we will adopt for our model. When scaling up their model from individual to collective, they found that confidence plays a key role in community formation, where low confidence individuals will follow a user with high confidence sharing the same opinion as them. But the existence of individuals with neutral opinions will result in a crowd that is overall less susceptible to high confidence *opinion attractors*.

Sahasara et al. [10] create a model of a social network where each user has an opinion in the interval [-1,+1] on a specific topic and is randomly connected with other users in the network. The opinions' of users are influenced based on a wall of recent posts made by friends they follow. They also include a *bounded confidence* variable ϵ along with the ability for each user to remove friends and create new connections inside the network. Their results show how the formation of echo chambers may be an inevitable consequence of the social processes that social media platforms facilitate, even if the original opinions are not polarized and with minimal unfollowing between users.

W. S. Rossi et al. [28] propose a mathematical model of a user and a personalized recommendation system providing recommendations. When taking into account the confirmation bias of users, they found that personalized recommendations typically lead to a feedback loop where both opinions and recommendations get more extreme.

2.2.2 De-polarization

Yue Wu et al. [13], propose an opinion evolution model where users only update their opinions when the social influence by their neighbors surpasses their own self-belief. Their depolarization strategies aim to mitigate polarization by targeting different groups of users based on their confidence level. Users with low confidence can adopt a more neutral opinion with the addition of cross-links. Medium level confidence individuals can be influenced by local neutral users being inserted in the network. Finally, users with the highest amount of self-belief can only be influenced by fully connected users (mimicking mass media) with neutral opinions, that increase status pressure.

Matakos et al. [12] adopt the Friedkin-Johnsen model [24] of opinion evolution. Their objective is to identify a set of individuals (nodes) within the network such that if these individuals adopt a more neutral position (internal opinion set to 0), polarization is minimized. However, their approach does not address the method by which these users will be moved towards a more neutral opinion, which is the primary focus of our thesis.

Using a similar opinion formation model, Musco et al. [14] introduce the concept of a polarization-disagreement index. They propose a method of reducing polarization by dynamically adjusting the weights of connections within the network, while still displaying relevant content to users to minimize disagreement.

Garimella et al. [16] attempt to reduce polarization by making link recommendations that have a high probability to succeed, aiming to nudge users closer to neutrality.

Similar to our work, Vendeville et al. [29] working with real data from Twitter, attempt to create a recommender that actively showcases more diverse content to users. They find that even a small increase in the diversity of content displayed in users' feeds helps mitigate the echo chamber problem in the network space. They demonstrate, how recommenders can actively contribute to alleviating polarization without jeopardizing engagement.

The closest to our work lies that of Cinus et al. [30]. In their work they use a bounded confidence model of opinion evolution (similar to our own model) and a recommender that makes user to user recommendations. They showcase how, an opinion-based recommender making partially random recommendations will always increase the echo chamber effect inside the network. In the end, they evaluate different mitigation policies, finding that a recommendation method focusing on opinion diversity is most effective in combating the echo chamber effect.

Chapter 3

Problem Setup

3.1 Creating a Polarized Social Network.

In our experiments, we will be simulating a Twitter social space where users post content that represents their opinions. The recommender can show these posts to other users in the network.

We will represent the social network as a directed graph $G = \{V, E\}$ where V is the set of users in the network and E the edges between users, which represent the follows.

We assume there are a total of $K = |V|$ users connected into an initial network G_0. There are a number of different ways to create this initial network. As in much related work [16], we will focus on scenarios with two echo chambers *already* existing in the network, on the topic we are interested in. This is a common case for example observed in real networks like Facebook and Twitter (currently X) [9]. We create an initial G_0 social network with echo chamber structure as follows:

- All users are split into two echo chambers, with each echo chamber having $K/2$ users, and an initial opinion s_i is then generated for each user i. This opinion is drawn from a Normal Distribution $N_1(s_{avg}, \sigma)$, if the user was assigned to the first echo chamber, or from the Normal Distribution $N_2(-s_{avg}, \sigma)$ if he was assigned to the second. Because we want $s_i \in [-1, +1]$ for every i and any values generated where $s_i \notin [-1, +1]$, which is possible although unlikely, are then clipped and set to either -1 or $+1$.

- The existence of links between nodes will depend on their opinion difference. More specifically, we draw a link between two users i and

j with probability:

$$p = e^{-|s_i - s_j| \cdot \lambda} \tag{3.1}$$

Parameter λ captures how strong the impact of opinion difference is on friendship, with larger λ values implying stronger echo chambers, but also sparser networks. Since the two echo chambers are located around $\pm s_{avg}$ we can assume the average distance between two opposing nodes will be $\mathbb{E}[d] = 2s_{avg}$, where $d = |s_j - s_i|$ and i, j are users that belong in different echo chambers.

Example Graphs:

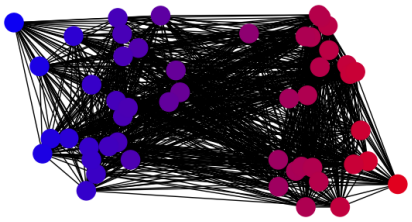
To get a feel of the type of graphs this model creates, and the impact of the various parameters, let's consider a simple example. Let $s_{avg} = 0.5$ and $\sigma = 0.1$. Now let's calculate P_1, the probability that a specific node from one echo chamber has at least one connection with any node from the opposite echo chamber, for $\lambda = 1$ and $\lambda = 5$.

$$P_1 = 1 - P_0$$

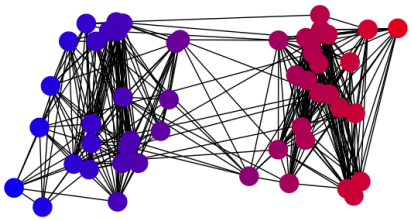
P_0 is the probability that the node has no connections with the opposite echo chamber. Given that the creation of each link is independent and follows a Bernoulli trial (success with probability (p), failure with probability (1-p)), we have:

$$P_0 = (1 - p)^{K/2}$$

When choosing λ we must also take into account for the size of the network, K. Larger networks will demand a larger λ so that P_0 stays relatively close to 1. In some of the testing scenarios we will be addressing in this experiment K will be equal to 50, so we will be using that number in our calculations. For $\lambda = 1$, we find $p = 0.36$, leading to $P_0 = 0.00001$. Therefore, an average node will connect with at least one node from the opposite echo chamber with probability $P_1 = 0.99999 \approx 1$. Conversely, for $\lambda = 5$, $p = 0.006$, yielding $P_0 = 0.84$ and $P_1 = 0.16$. Choosing $\lambda = 1$ will not create a polarized environment of 2 clusters as it results in all nodes from one echo chamber having at least one connection with the opposite echo chamber. However, with $\lambda = 5$, there is a clear separation between the two clusters, as illustrated in Fig 3.1.



(a) $\lambda = 1$



(b) $\lambda = 5$

Figure 3.1: Example of 2 Networks with 2 clusters/echo chambers and 50 users in total.

3.2 Recommendation-driven Opinion Evolution

Having described how to create an initial graph, we will now describe how the nodes' opinions can evolve over time, as a function of recommendation made to different users. In most social networks and more specifically on Twitter, users can make posts that represent their opinions. These posts are then displayed on the feed of their followers in two different ways, chronologically or through recommendation system that selects which posts to display (mostly based on engagement metrics and previous likes and dislikes).

In this thesis we are interested in the latter approach, where posts are chosen by a recommender to be shown to the users. Inevitably, what our friends share and care about is expected to affect our opinion(s) on different topics as well. Such interactions and respective influences are rather complex and currently the subject of various sociological and other studies. First, it would be beneficial to take a look at two other pivotal continuous opinion dynamics models, the DeGroot model [23] and the Friedkin-Johnsen (FJ) model [24].

3.2.1 DeGroot

DeGroot's model consists of n users, each with an initial opinion $x_i(0) \in \mathbb{R}$. Each user updates their opinion, from timestep t to $t+1$, by taking a weighted average of their opinion and those of their neighbors:

$$x_i(t+1) = \sum_{j=1}^{n} w_{ij} x_j(t) \tag{3.2}$$

with

$$\sum_{j=1}^{n} w_{ij} = 1$$

for all i.

A key parameter to this model is the weight matrix W which contains the weights of all the connections in the graph. The weights w_{ij} typically reflect trust or influence within the network. Over time, if certain conditions are met (aperiodic and strongly connected weight matrix W [25]), through repeated iterations, opinions tend to converge reaching a consensus.

3.2.2 Friedkin-Johnsen

Friedkin and Johnsen extended the DeGroot model, to include both the concept of innate beliefs and predispositions within its users. More specifically in this model, each node (user) maintains an *internal* opinion s_i that remains fixed over time. However, individuals can still change their *expressed* opinion z_i, at every new timeslot $t+1$. This change is based on a combination of their own *internal* opinion and their neighbors' *external* opinions at time t, as shown in Eq. 3.3.

$$z_i(t+1) = (1-\lambda_i)s_i + \lambda_i \sum_{j \in N(i)} w_{ij} z_j(t) \tag{3.3}$$

where N(i) denotes the neighbors of user i.

Similarly to the DeGroot model, the FJ model also utilizes a weight matrix W, which represents the influence that users exert over one another. The parameter λ_i corresponds to the *susceptibility* of each user i to external influences, affecting how much an individual's opinion is swayed by their neighbors. Nodes with $\lambda_i = 0$ are stubborn nodes that never change their opinion. It is also possible for external opinions to achieve consensus with this model under certain conditions [25].

3.2.3 Our model

For our model we will make the following generic assumptions:

- We focus on a specific subject only (that tends to polarize social media users), and consider posts on this topic only.
- We assume that posts by each user represent their (average) opinion on the topic (some randomness can easily be included in the model).
- Unlike the previous two models, only one user will update their opinion at each time step, hence ours is an asychronous evolution model. This is more realistic, and also more tractable in terms of action space, as we shall see in the next chapter.
- If a post by a friend, whose opinion differs, is recommended to another user, it is expected that their opinion will be "nudged" by some amount towards their friend's.
- How much the opinion will be "nudged" might also depend on how much "trust" the user has in their friend (and thus their related posts).

A number of different modeling approaches could be taken to capture (some of) the above intuitive aspects in our model.

Opinion evolution: First, we assume that under certain conditions when a user, for example user i, is recommended a post from a neighbor, let j, which represents j's opinion, i's opinion will be nudged closer to j's by some amount. This is consistent with the previous two models we discussed and can be expressed with a generic formula like:

$$s_i(t+1) = s_i(t) + w_{ij}(t)(s_j(t) - s_i(t)) \tag{3.4}$$

One could interpret the above formula as gradient step towards j's opinion with (relative) step size $w_{ij}(t)$ with the term $w_{ij}(t)$ again symbolizing the influence user j has on user i at time t. This can be seen as a 2-user variant of the DeGroot model.

Selective Bias: There are a number of different ways one could take to capture the intuitive aspects of influence in our model. E.g (a) as long as user i, follows user j, they trust him equally to all other users they currently follow. This trust, w_{ij}, could be constant, interpreted as a probability of being influenced at all by j's opinion, or a *relative amount of opinion difference reconciliation*. Or more realistically, (b) we could assume that "not all

friends are equal" and friends with opinion further away have less influence, in relative terms, in which case this weight is *also* some non-increasing function of opinion difference. To capture the essence of unequal and dynamic influence, we define $w_{ij}(t)$ specifically as:

$$w_{ij}(t) = 1 - \frac{|s_j(t) - s_i(t)|}{2} \tag{3.5}$$

This formula reflects that the influence decreases as the opinion difference between i and j increases. When the opinions are identical $s_i(t) = s_j(t)$, the weight $w_{ij}(t)$ reaches the maximum of 1 indicating the highest influence. Conversely, when the opinions are entirely opposite, the weight diminishes to its minimum value 0, signifying no influence. For example, let's say at time-step t we have two users i and j with $s_i(t) = 0.6$ and $s_j(t) = 0.3$ and we recommend user j to user i. With $w_{ij}(t) = 1 - \frac{0.3}{2} = 0.85$ user i's opinion at the next time-step, $t + 1$, will be :

$$s_i(t+1) = 0.6 + 0.85(0.3 - 0.6) = 0.6 - 0.85 \cdot 0.3 = 0.345$$

Bounded Confidence: We also introduce another modeling assumption, a variable called $D_{max} \in [0, 2]$, which acts as a limiting factor. Users whose opinion differences exceed the threshold D_{max} will not be able to influence each other. This is not necessary for our model, but it often captures cases where posts by online friends with contrasting opinions, tend to be ignored. More specifically, if $|s_j(t) - s_i(t)| > D_{max}$, user i will not be able influence user j and vice versa, although user i and user j can still follow each other. By "can't influence" what we mean is that recommending user j to user i **will not** change the opinion of user i, even if user i follows user j.

This addition to our model, overall grounds our model to realism, as users/people are not so easily influenced by directly opposing views but can be nudged slowly to a more neutral position by viewing opinions relatively close to their own.

If we substitute Eq 3.5, in Eq 3.4 we get:

$$s_i(t+1) = s_i(t) + (1 - \frac{|s_j(t) - s_i(t)|}{2})(s_j(t) - s_i(t)) \tag{3.6}$$

To get a feel of what this formula implies, let's look at this formula as a function of $d = s_j - s_i \in [-D_{max}, +D_{max}]$, the opinion difference between the two users, before interacting:

$$s_i(t+1) = \begin{cases} -\frac{d^2}{2} + d + s_i(t), & d > 0 \\ \frac{d^2}{2} + d + s_i(t), & d < 0 \\ s_i(t), & d = 0 \end{cases} \tag{3.7}$$

Trade-offs: Now let's look how the choice of our weight function effects the opinion shift and the trade offs our recommender will encounter. For the examples below, and for the rest of our thesis, we will set $D_{max} = 0.5$ as we believe this represents a realistic tolerance of opinion difference that users have inside a social network.

For $d = 0.2$ we get :

$$\begin{aligned} s_i(t+1) &= -\frac{0.2^2}{2} + 0.2 + s_i(t) \\ &= -0.02 + 0.2 + s_i(t) \\ &= 0.18 + s_i(t) \end{aligned}$$

whereas for $d = 0.4$:

$$\begin{aligned} s_i(t+1) &= -\frac{0.4^2}{2} + 0.4 + s_i(t) \\ &= -0.08 + 0.4 + s_i(t) \\ &= 0.32 + s_i(t) \end{aligned}$$

It seems as though, choosing a user to recommend that is near the threshold gives us a larger opinion shift. This weight function can therefore be considered as a greedy weight function, incentivizing us to make recommendations as close to the threshold as possible.

However, by not examining the state of the graph before making a recommendation and merely focusing at our end goal, we could encounter a scenario where a set of users, let $\mathbb{Q}$, have been "left behind". This means that there exists no other user $j \in V \setminus \mathbb{Q}$ that satisfies $|s_j - s_i| \leq D_{max}, i \in \mathbb{Q}$. If such a gap between users' opinions is created, users belonging to $\mathbb{Q}$ cannot be influenced by the other users outside of $\mathbb{Q}$. In that case, the most neutral opinion users in $\mathbb{Q}$ can adopt is:

$$s_{min} = \min_{j \in \mathbb{Q}} |s_j(t)|$$

So, the trade-off with this implementation of the weight function lies between maximizing the reduction of polarization in the network by making recommendations close to the threshold and the risk of leaving users "stuck" in their own echo chambers.

There are of course countless other weight functions which we could look to implement but can't due to the scope of our work. We will only be examining an additional weight function which introduces an interesting

new trade-off for the recommender when making recommendations within the threshold.

$$w_{ij} = \begin{cases} 4|d|^2 - 4|d| + 1, & d \in [-0.5, 0.5] \\ 0 & else \end{cases} \tag{3.8}$$

Let us again, do some quick math to get a better understanding of how this new weight function affects the opinion shift and what kind of new trade off it introduces.

For $d = 0.4$ we get:

$$\begin{aligned} s_i(t+1) &= s_i(t) + (4 * (0.4)^2 - 4 * 0.4 + 1) * (0.4) \\ &= s_i(t) + (0.64 - 1.6 + 1) * 0.4 \\ &= s_i(t) + 0.016 \end{aligned}$$

whereas for $d = 0.2$:

$$\begin{aligned} s_i(t+1) &= s_i(t) + (4 * (0.2)^2 - 4 * 0.2 + 1) * (0.2) \\ &= s_i(t) + (0.16 - 0.8 + 1) * 0.2 \\ &= s_i(t) + 0.072 \end{aligned}$$

and for $d = 0.1$:

$$\begin{aligned} s_i(t+1) &= s_i(t) + (4 * (0.1)^2 - 4 * 0.1 + 1) * (0.1) \\ &= s_i(t) + (0.04 - 0.4 + 1) * 0.1 \\ &= s_i(t) + 0.064 \end{aligned}$$

With this weight function, there seems to be a tradeoff between two opposing forces, (i) a post with high opinion difference than mine will also "blunt" my opinion more, *if I end up trusting it*, (ii) but the trust I have in users with differing opinions is reduced. Hence, the recommended post should neither be too close ($d \to 0$) nor too far (d right below the threshold), in order to bring user i opinion as close to 0 as possible in that step. Additionally, our recommender still has to be careful not to "leave" any users behind inside their own echo chamber.

In our model we will be implementing only the weight function from Eq 3.5, but our Deep RL recommender should be able to learn and balance the trade offs introduced from implementing the weight function from Eq 3.8 as well.

If the above model is known by the recommender, then it could easily "solve" this model and choose a post that most impacts the targeted user's

opinion. However, in our framework, we'll assume that this model is unknown to the recommender, and we'll use reinforcement learning algorithms (DQN) that can learn to optimize a sequence of recommendations while only implicitly learning the model details needed, while doing so. To this end, we will use later as a metric the concept of "successful/unsuccessful" recommendations, as a proxy of our algorithms learning the model's impact. Specifically, we will assume that a recommendation is *unsuccessful* if:

- **Not a friend:** The recommender gives us a post by someone we don't know. In our model, we assume such recommendations are ignored (and thus make no progress towards depolarizing the network).
- **Friend beyond threshold:** In this case, the post recommended is by a friend, but the opinion difference is over the threshold D_{max}, again leading to us ignoring that post.

All other recommendations are considered successful, whether they have a positive or a negative effect.

3.3 Measuring Polarization

There are plenty of ways to measure polarization in related literature, some of them actually leading to qualitatively equivalent behaviors(see recent analytical study [22] that discusses all these metrics in detail). For our study we choose the L1 metric (referred to as P4 in [22]) which we divide by the number of users in the network.

$$P = \frac{||\mathbf{S}||_1}{|V|}. \tag{3.9}$$

Intuitively this metric informs us of how far away from complete neutrality all opinions are as it can also be expressed as the absolute average opinion of all users in the graph. We do not take into account the connections in the graph as those depend on the original opinions and opinion distances between users. So by bringing all opinions closer to neutrality, in this case 0, we are also encouraging users to make more connections with users with whom they would otherwise not be connected with, because their opinions were too distant.

Chapter 4

Depolarizing Recommendations based on A Deep Reinforcement Learning Algorithm

4.1 Reinforcement Learning Model

In this section, we define the state space, action space, reward function, terminal states and transition model of our problem.

State Space $\mathcal{S}$. The state of the system at time slot t is a vector of size $2K$ (where K is the number of users), consisting of the opinion vector $\mathbf{S}_t$ and the M randomly chosen users represented in a one-hot vector, u_t. More specifically, if we have $M = 1$, ex. user 3, then the one-hot vector will be $u_t = (0, 0, 1, 0, ..., 0)$. If $M = 2$ and we have two users, ex. users 3 and 5, then $u_t = (0, 0, 1, 0, 1, 0, ..., 0)$. The M users are picked at random from all available users, at every time slot t.

Definition 1 (State). $\sigma_t = (\mathbf{S}_t, u_t)$

Since the opinions of users in this graph are continuous values $\in [-1, +1]$ it implies an infinite state space, and thus cannot be handled by standard ("tabular") methods. We will handle such scenarios with approximate RL methods only.

The main reason we did not include the connections of the M online users in our state is that, in stationary environments (no new follows or unfollows), the agent will eventually learn the existing connections over time. Providing that information in a 0-1 vector is not very helpful. In the best-case scenario,

the learning process might be accelerated, but this is unlikely, because the agent still has to decipher what this extra information means and how it affects the solution. In the worst case, it might slow down that process.

Action Space $\mathcal{A}$. Given the current state of the system σ_t the agent's action a_t is recommending one user $\in V$ to each of the M randomly chosen users.

Definition 2 (Action). $a_t = (a_1, a_2, \ldots, a_M), a_m \in V$

This implies that both the state space and the action space of this problem quickly explode in bigger scenarios. Our state space scales linearly with the number of users in the network, but our action space grows exponentially with the increase of M, the number of users that are chosen and see recommendations at time step t.

For example, if we have a network with $K = 50$ users and $M = 1$ recommendation per timeslot, our action space size is $|A| = 50^1 = 50$, since there are 50 possible recommendations we can make in each round ($a_1 \in V$). If we decide to increase M by 1, we will have to make two recommendations per round, because two random users will now be active in the network and we need one recommendation for each user. Thus, the action space size becomes $|A| = 50^2 = 2500$, since there are now 2500 distinct combinations of (a_1, a_2) recommendation tuples. Meanwhile, if we double the amount of users in the network instead of increasing M, we get $|A| = 100^1 = 100$.

This is unlike recent video game problem setups [17], where Deep Neural Networks can manage large state spaces, such as video game images, but where action spaces are relatively small. More recent successes in the field of board games [18] require handling both large state and action spaces, focusing on "planning" problems, where the environment and rules are fully known. This is in contrast to our model where environment variables are unknown (next M users, opinion evolution model).

Transition model $T(\sigma_{t+1}|\sigma_t, a)$. At every time step t the next state we might end up in is partly deterministic and influenced by our actions (recommendations), since the opinion vector $\mathbf{S}_{t+1}$ (first half of σ_{t+1}) is only influenced by our recommendations made in the current time-slot. Contrary, the one-hot vector containing the M users is random (latter half of the state vector), since the M users are uniformly chosen at random at every time-slot t.

For example, if $K = 3$ and $M = 1$ and the user that is chosen is user 2 we will have $\sigma_t = (s_1(t), s_2(t), s_3(t), 0, 1, 0)$. Choosing a recommendation a_t will only impact the opinion of user 2, so the new opinion vector and first

half of the new state will be

$$\mathbf{S}_{t+1} = (s_1(t+1), s_2(t+1), s_3(t+1))$$

where all $s_i(t+1) = s_i(t)$ for $i \neq 2$, while $s_2(t+1)$ will be calculated using Eq. 3.6 if the recommendation was successful, otherwise $s_2(t+1) = s_2(t)$. This part of the transition, which is the one that we control with our recommendations is deterministic. Then, independent of whether the recommendation indeed changed the opinion of user 2, with equal probability our next state might be :

$$\sigma_{t+1} = \begin{cases} (s_1(t), s_2(t+1), s_3(t), 1, 0, 0) \\ (s_1(t), s_2(t+1), s_3(t), 0, 1, 0) \\ (s_1(t), s_2(t+1), s_3(t), 0, 0, 1) \end{cases}$$

It would be interesting to also consider the choice of next user as part of our algorithm; we defer this to future work.

Reward Function r. Reward function design is a critical task that significantly influences the effectiveness of the learning process. A reward function can incorporate various factors, such as immediate task completion, long-term strategy, efficiency, to guide the agent towards optimal behavior [31]. In our case we choose a reward function focusing on immediate rewards, guiding our agent towards reducing polarization at every time-step t.

Given the current state of the system σ_t, the respective agent action taken a_t and the next state σ_{t+1} we define the reward when making a successful recommendation as:

$$r = ||\mathbf{S}(t)||_1 - ||\mathbf{S}(t+1)||_1 \tag{4.1}$$

The above equation can be simplified to:

$$r = \sum_{i \in M} (|s_i(t)| - |s_i(t+1)|) \tag{4.2}$$

since the only opinions that change are those of the M random users.

This reward structure only rewards or penalizes successful recommendations, while unsuccessful ones that do not shift opinions receive a reward of zero (neutral moves). We will be applying a penalty of 0.05 whenever the agent makes an unsuccessful recommendation, in order to incentivize the agent to make the least amount of unsuccessful recommendations possible. In the end, our reward function, for $M = 1$, can be defined as:

Definition 3 (Reward).

$$r = \begin{cases} |s_i(t)| - |s_i(t+1)|, & \text{successful} \\ -0.05, & \text{otherwise} \end{cases} \tag{4.3}$$

Terminal States σ_τ. When the environment reaches certain predetermined states, the algorithm stops. These steps can signify either a victory or a loss. In our experiments, if at the end of any time step t there is a gap between two user groups larger than D_{max} (trade-offs), the environment is considered "stuck" and the episode ends (**loss**). Conversely, if at the end of an episode the polarization index P has reached a value below $\frac{D_{max}}{2}$, the episode, again, ends (**victory**). The inclusion of this terminal state, as opposed to conditions, like all values reaching zero or all opinions converging to a singular viewpoint, was necessary to ensure that our algorithm completes within a reasonable timeframe.

The reason we considered the last scenario a victory is: if $P \leq \frac{D_{max}}{2}$ then most opinions will have values $\in [-\frac{D_{max}}{2}, +\frac{D_{max}}{2}]$. This means that on average the opinion distance between two users from different echo chambers will be equal to D_{max}. We feel as though this is a good stopping point for the algorithm, where all users are mostly neutral in their opinions, but they can now also interact with users who where previously on the "opposing" echo chamber.

We also introduced an upper limit to the rounds an agent can play, in order to avoid scenarios where the algorithm cannot reach any of the two other conditions, in a reasonable amount of rounds, in any particular episode while training.

4.2 Reinforcement Learning Agents

The goal of any Reinforcement Learning problem like the above is to create an agent that gradually learns to take better actions over time, in the end maximizing the rewards per round. This learning is achieved by interacting with the environment (in our case the network) of the problem. Below we provide a brief description of these interactions, between our DQN agent and the environment, using some standard RL definitions:

-Environment: M users, are chosen randomly at every round t.

-Agent: Based on the current state of the system $\sigma_t = (\mathbf{S}_t, u_t)$ (consisting of the current opinion vector and one-hot user vector), it must choose a user to recommend to each of the M users $a_t = (a_1, a_2, \ldots, a_M)$. This action is taken based on a value function $Q_\theta(\sigma, a)$ that the agent maintains, which estimates the expected (discounted) reward starting from state σ and taking action a (for all σ, a).

-Environment: The recommendations are shown to the users and Eq 3.6 is applied. This moves the state of the problem to the new state σ_{t+1} and the reward r_t is calculated (a function of the old opinion vector $\mathbf{S}_t$ and the new opinion vector $\mathbf{S}_{t+1}$), according to the details of the previous section.

-Agent: Given the tuple $(\sigma_t, a_t, \sigma_{t+1}, r_{t+1})$ ("experience"), the agent first attempts to "improve" its decision function $Q_\theta(\sigma, a)$, based on the recent and past experiences and then proposes new recommendations for the next time window.

-Environment: Again applies the recommendations to the users and returns the respective rewards, the agent will improve its value function and proceed with a new recommendation proposal, and so on and so forth, until we reach a terminal state σ_τ.

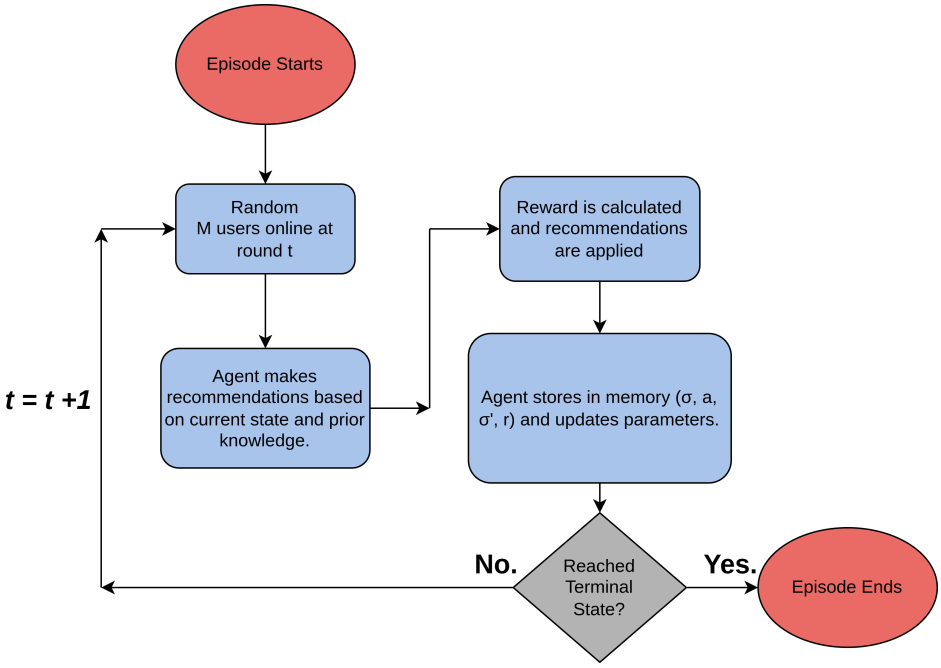


Figure 4.1: Flowchart illustrating the sequence of steps in a training episode.

4.3 DNN architecture

We choose simple and relatively small DNN architectures as Q-function approximators. Our DQN agents will utilize 1 hidden linear layer with 64 neurons.

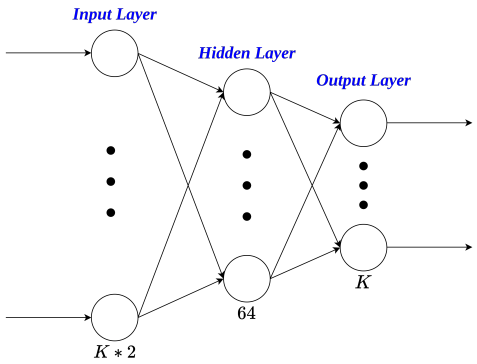


Figure 4.2: The neural network receives the state of the system as input and outputs the Q-value estimates.

Hyperparameters:

Hyperparameter	Value
η(learning rate)	10^{-4}
ϵ	0.1
batch size	64
Replay buffer $\mathcal{B}$ size	10^5

The final parameter we need to set is the discount factor γ. In general, γ relates to the recommendation reward and how far ahead the agent is willing to look in order to assess the immediate reward. For example, a value of 0.9 allows the agent to look roughly 10 timesteps ahead. For $\gamma = 0.99$ the agent will look ~ 100 timesteps ahead. Conversely, if the value of γ is set to 0 it will result in an essentially "greedy" agent, choosing the action that maximizes the reward it receives in the current state, not taking into account future states it may end up in.

Another possible effect our choice of γ may have relates to the overall rounds it takes to reach a winning state. The higher the γ the more rounds our agent might go through in order to win, which in our case is achieving $P \leq \frac{D_{max}}{2}$. We will observe the impact that different γ have on our agents

in the next chapter.

In our implementation of the DQN agents, we will be updating the Target Network parameters θ' after every $T = 2000$ time steps. This periodic, rather than instantaneous, update of our system's parameters should prevent rapid oscillations during learning and what is often referred to as "catastrophic forgetting", where the agent, after it has learned how to effectively play the game, will "forget" which moves are good and start performing poorly again.

Chapter 5

Simulation Results

5.1 Setup

So far, we have described our problem setup, introduced the basic algorithmic architecture we will use to try to solve the depolarization problem, and also discussed briefly the computational challenges we expect this algorithm will face. In this section, we will use a range of simulation scenarios to better understand two key questions: (i) When and how well our algorithmic framework can tackle such depolarization tasks, and (ii) how its behavior and/or success relates to various scenarios parameters.

For our first set of experiments, we used environments created using the methods described in subsection 3.1, with $N_1(-0.5, 0.1)$ and $N_2(0.5, 0.1)$, $K = 50$ users, $M = 1$ recommendation per round and $D_{max} = 0.5$.

5.2 Performance comparison between different γ Agents.

In this section we will be focusing on how our choice of γ affects our solution. First we initialized 10 different networks, and for each network we implemented three separate agents, each one set with a different γ value (0, 0.9 and 0.99). After training each agent, we recorded their performance for 1000 episodes.

Let us first look at, possibly the most important statistic, the average win percentage for all 3 different γ, when we include a Target Network and train our agents for 5000 episodes.

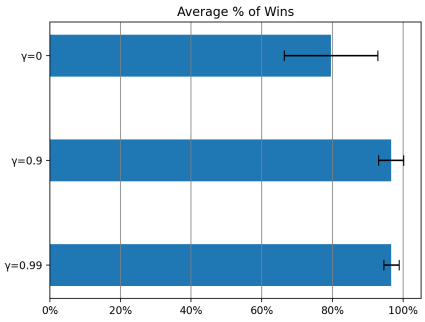


Figure 5.1: Win Comparison: We can observe that $\gamma = 0$ performs worse out of all the γ implementations.

In Fig 5.1, there are two key points that we need to focus on:

1. In order to best solve our problem, we require a recommender with "vision"($\gamma > 0$); one that looks at the environment as a whole with regard to future states and rewards, not one that maximizes the reward at every time step.

2. The "greedy" agent still performs good enough (near 80 percent win rate) but the high standard deviation, compared to the other two implementations (although it comes from a relatively small sample size of networks), highlights that on a case-by-case basis an agent with $\gamma = 0$ can be unreliable.

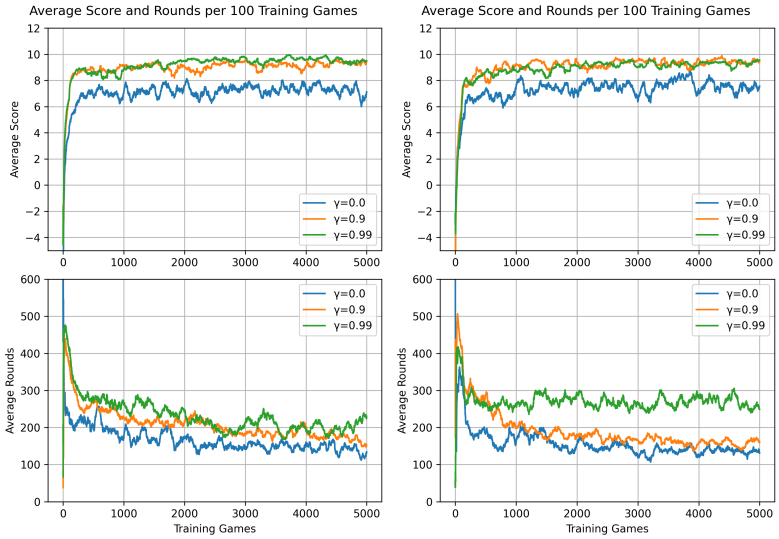
The latter point can also be seen when looking at the raw data from all the runs in Table 5.1.

Table 5.1: Performance stats of $\gamma = 0$ implementation

Env	Wins	Losses	Limit
1	785	211	4
2	900	77	23
3	855	142	3
4	800	197	3
5	548	448	4
6	1000	0	0
7	787	210	3
8	902	96	2
9	629	366	5
10	757	240	3

The limit refers to the upper limit of rounds that the Agent has in every episode, which we have set to a value of 1000. If that limit is reached then the episode ends.

We can also look at the training graphs from the two worst case scenarios (Environments 5 and 9 in Fig 5.2) to validate that this is not a convergence issue, where the "greedy" agent has not yet converged.



(a) Environment 5 *(b) Environment 9*

Figure 5.2: The $\gamma = 0$ implementation is more unstable(more oscillation around the convergence point), but still it converges.

As we can see, the "greedy" agent does indeed converge but at a lower point than the other two agents and with more oscillation around its convergence point. We can also observe, both in Fig 5.2 and Fig 5.3 that this implementation overall achieves victory in the fewest rounds (128.6), as expected, followed closely by the agent with $\gamma = 0.9$ (129).

The standard deviation of these values is relatively large in all three cases. That is due to the fact that at each time-step t, the user who will receive the recommendation is chosen at random, out of 50 total users. Thus, as the game goes on and users are brought closer to the middle, if some users have still been left out, the agents will have a small chance each round to "get" these users and lower their polarization, leading to victory. This randomness is also the cause for the extra "instability" observed in the

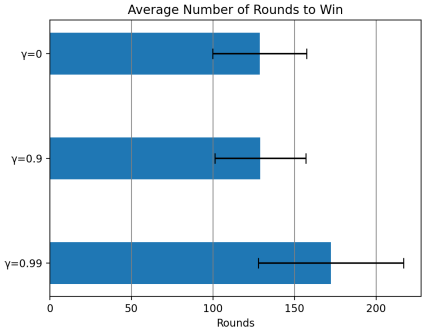
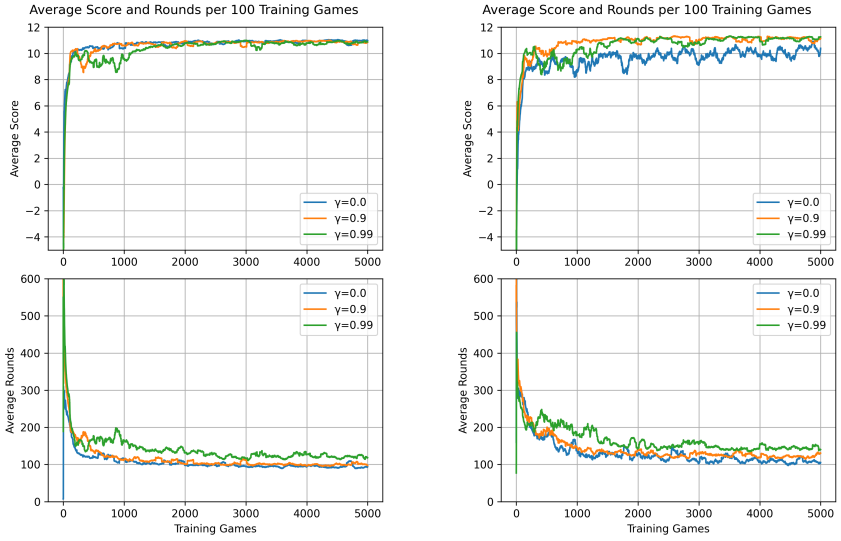


Figure 5.3: Average Rounds to win Comparison: The $\gamma = 0.99$ Agent has the largest error with 44.5 with the $\gamma = 0$ and $\gamma = 0.9$ Agents following with 28.83 and 27.81 respectively.

two "*Average Rounds*" graphs in Fig 5.2 compared to the "*Average Score*" and its lower oscillation around the convergence point.

Finally, we will look at some scenarios where the $\gamma = 0$ agent's performance is on par with the other two implementations.



(a) Environment 6 *(b) Environment 8*

Figure 5.4: On both Environment 6 and Environment 8 the "greedy" Agent converges closer to the other two implementations and with less oscillation in the first case.

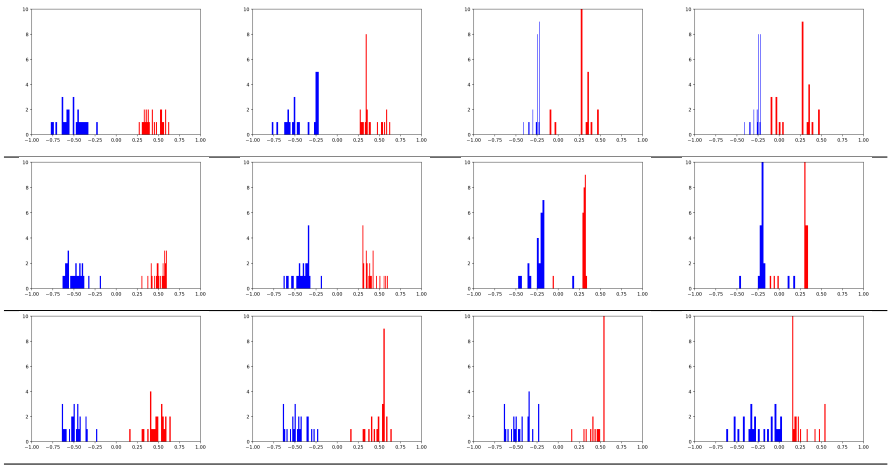


Table 5.2: Three examples showcasing how the original distribution of opinions is moved towards neutrality through recommendations ($\gamma = 0.99$). Users starting from the positive echo chamber are red, blue otherwise. From left to right, First column: Starting opinion distribution, Last Column: Winning State the agent achieved. X axis = value of opinion, Y axis = # of users.

In conclusion, based on their performance, both $\gamma > 0$ implementations can be used reliably to solve the problem, with the $\gamma = 0.9$ agent group being the fastest of the two. Due to its unstable behavior, it is clear that, while some scenarios can be solved with a $\gamma = 0$ agent, the problem solution requires vision and thus a more sophisticated and less greedy approach.

5.3 Target Network Performance Improvements

In this section will be examining how the inclusion of the Target Network (TN) impacts our agents' performance for all three γ and if it's a necessary component for our proposed solution. We will be using the previously generated networks(environments) and training the agents, this time without a TN. In the case of $\gamma = 0$ we expect to notice no significant performance differences since, by definition, the inclusion (or not) of the TN does not affect its calculations (Eq 2.6).

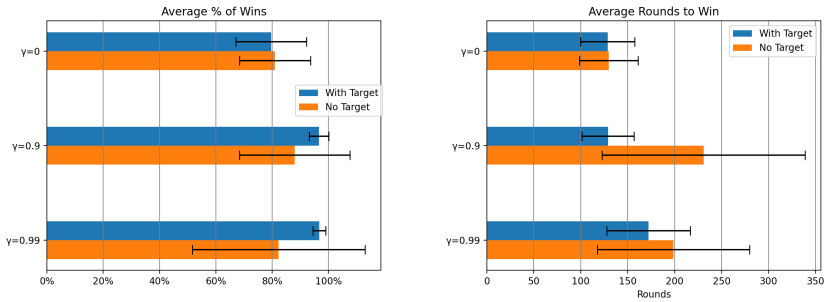


Figure 5.5: The $\gamma = 0$ agents' performance remains unchanged, while the other implementations see a significant drop in performance.

For $\gamma = 0.9$, we observe a performance decline of approximately 9 percent, from 96.6 to 88 percent. Similarly, for $\gamma = 0.99$ the win percentage decreased from 96.7 to 82.3 percent. Alongside this absolute performance drop, there was also an increase in the error for both implementations where $\gamma > 0$.

5.3.1 $\gamma = 0.9$

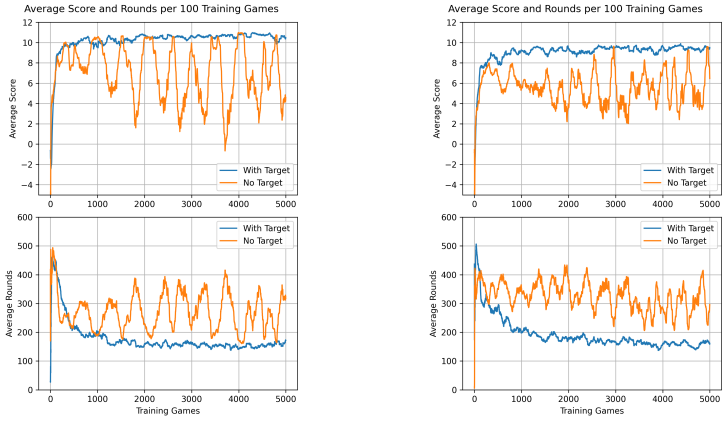
Table 5.3: $\gamma = 0.9$, no TN

Wins	Losses	Limit
965	34	1
995	0	5
866	37	97
999	0	1
999	1	0
1000	0	0
663	15	322
1000	0	0
374	0	626
943	57	0

Table 5.4: $\gamma = 0.9$, with TN

Wins	Losses	Limit
985	15	0
982	16	2
964	28	8
996	0	4
922	68	10
999	0	1
964	34	2
997	2	1
883	109	8
969	21	10

Taking a closer look at the data for the $\gamma = 0.9$ implementation, in tables 5.2 and 5.3 we can see why and how this large disparity was caused between the versions that have a TN and those that do not. The most significant performance drops occurred in environments 7 and 9, meanwhile in the other eight networks, we observe a small fluctuation in overall wins.



(a) Environment 7 *(b) Environment 9*

Figure 5.6: $\gamma = 0.9$: Unstable learning from the agents with no TN leads to unreliable performance.

When we take a closer look at the graphs in Fig 5.6, we can see that in the first 600 training episodes, the agents have very similar behavior. After that point (600-5000 training episodes), the agents with a TN retain a relatively steady learning curve, where some improvements are still made in regards to both average score and average rounds. In the case of the group without a TN, we can observe a lack of convergence, with large oscillations and many "catastrophic forgetting" phases. We also see how in some points these agents are able to achieve good performance, where the average score and average rounds match those from the opposite group, but these points are few and far between.

Even in Environment 6 (Fig 5.7) where both Agents perform equally good, seemingly perfect(in the case of the no TN implementation), when we look at the learning graphs we observe major differences in their behavior during training.

The No Target agent has "converged" on a lower score despite still winning, indicating that it receives more negative rewards throughout each game. This suggests it is making more unsuccessful recommendations, leading to a higher sum of penalties, even though it manages to win. Examining

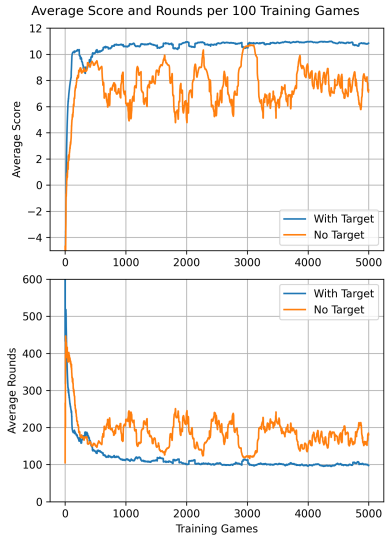


Figure 5.7: $\gamma = 0.9$: Environment 6

the statistics, *33 percent* of the no TN agent's recommendations are unsuccessful, meanwhile the agent with the TN, experiences less failed recommendations, with roughly *10 percent* of its recommendations being unsuccessful.

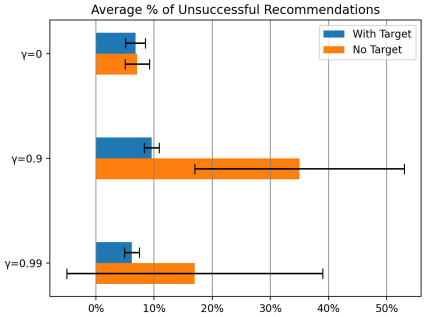
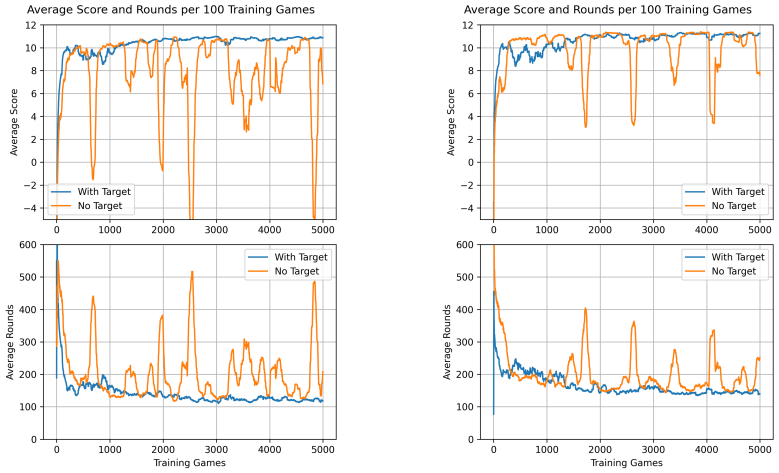


Figure 5.8: Unsuccessful recommendations increased without a TN for $\gamma > 0$.

When we look at the average unsuccessful recommendations percentage, we notice an increase across the board for both $\gamma > 0$. The high error in these graphs also reinforces our findings that these agents are less reliable, than those with a TN.

5.3.2 $\gamma = 0.99$



(a) Environment 6 *(b) Environment 8*

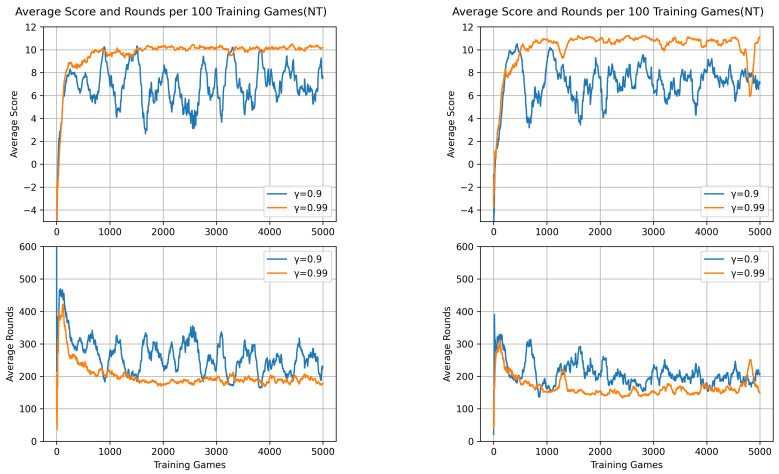
Figure 5.9: $\gamma = 0.99$: Large oscillations affect performance, when no TN is used.

For the largest γ in our experiments, the absence of a TN had less of an impact compared to the $\gamma = 0.9$ case. However, its effects are still evident in two out of the 10 environments, namely Environment 6 and Environment 8. If we remove from our average calculations these 2 environments we obtain almost identical results across all metrics.

	Win%	Loss%	Limit%	UR%	Rounds
No Target	96.25	3.625	0.125	6.3	172.07
With Target	96.71	2.73	0.56	6.2	172.22

Table 5.5: $\gamma = 0.99$ performance comparison, UR = Unsuccessful Recommendations

Overall, these agents performed more closely to their counter-parts, which used a TN during training, leading to similar performance. This is contrary to the behavior of the $\gamma = 0.9$ agents, which were more unstable and although they were able to achieve a high win rate, most of the time, the did so while being slower(requiring more rounds to win) and making more unsuccessful recommendations. Despite their similarities, in 8 out 10 scenarios, these agents are still proving to be less reliable than those with a TN, so we can conclude that a TN is necessary for our problem if we want consistently good performance across all networks.



(a) Environment 1 *(b) Environment 2*

Figure 5.10: Stability differences between the two $\gamma > 0$ implementations during training without a Target Network.

5.4 Effective Results with Early Termination

In the previous sections, all agents were trained for 5000 episodes with each session taking approximately 50-60 minutes to conclude. Based on the training graphs we've seen so far (when using a TN), the agents' performance has reached a good enough level before 1000 training episodes. Beyond that mark, there are some slight improvements, with minor increases and decreases in the average score and average rounds graphs, respectively. Is it possible then, to achieve similar performance with only a fraction of the training time previously used?

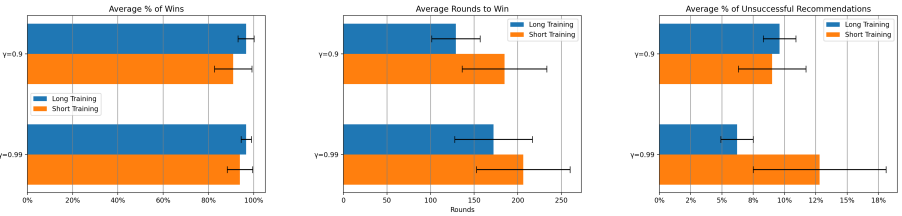


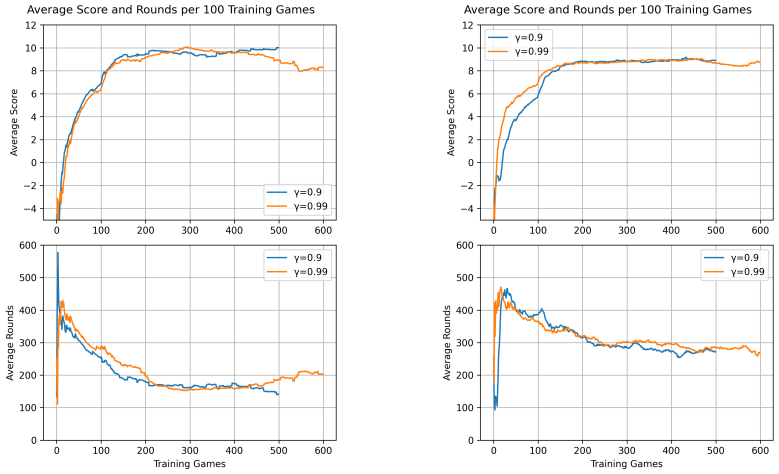
Figure 5.11: Performance comparison between the long and short training groups.

To assess how these agents perform with reduced training, we halted training for the $\gamma = 0.9$ agents at 500 training episodes and for the $\gamma = 0.99$

agents at just 600 training episodes. This adjustment reduced the training time to between 8 to 10 minutes, which is 20 percent of the time it previously took. In Fig. 5.11, we present a direct comparison between the two different agent groups.

In the short training group, both $\gamma > 0$ implementations maintain an above 90 percent win percentage. For $\gamma = 0.9$ we observed a 6% reduction (from 96.61 to 90.92), while for $\gamma = 0.99$, we saw a reduction of less than 3 percentage points (from 96.71 to 93.9).

The increase in the average rounds to win was expected. Due to the inherent randomness in our environment, more training time is required to find the most rewarding paths to success and to explore all available recommendation options(for all users). For the $\gamma = 0.99$ implementation, we notice that the unsuccessful recommendation percentage nearly doubled when compared to the long training group. In conclusion, when training time is reduced we can achieve comparable but ultimately worse results than the agents with longer training.



(a) Environment 4 *(b) Environment 5*

Figure 5.12: Reduced training length performance graphs.

5.5 Larger Networks

In this section we will increase the size of the networks to $K = 100$ users to observe how our solution scales with the number of users. As we previously discussed in subsection 3.1, when choosing λ we must take into account the size of the network. Now with $K = 100$ we must also increase λ to keep probability P_1 roughly the same as before. With $\lambda = 6$ we get $P_1 = 0.12$, which is close to the $P_1 = 0.16$ we had in the $K = 50$ networks.

Below we see the performance from 10 different scenarios where the agents were trained for 3000 episodes, with a TN.

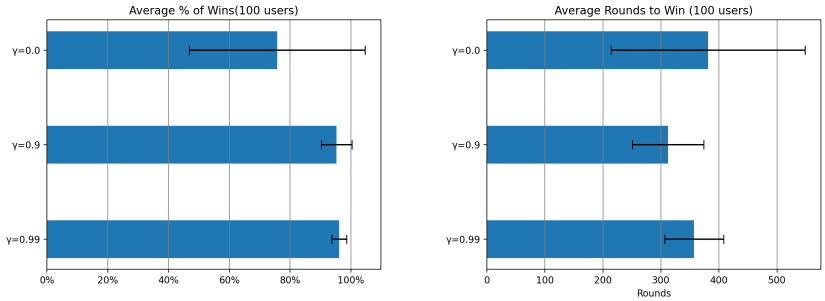
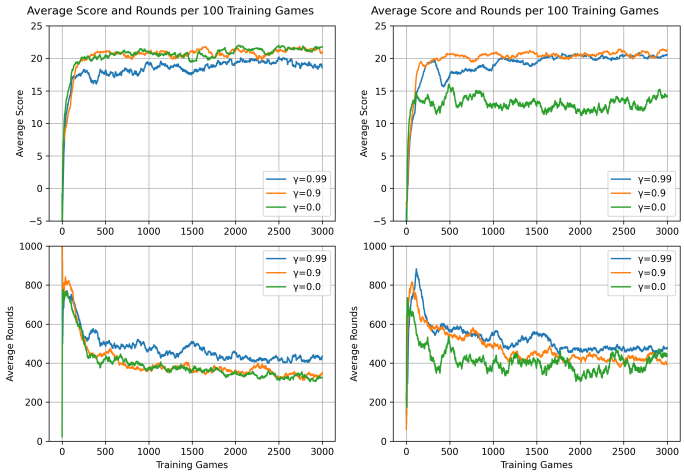


Figure 5.13: K=100 users: Performance comparisons between all γ agents.



(a) Environment 2 *(b) Environment 5*

Figure 5.14: K=100 users: Training comparisons between all γ agents.

Our data suggests similar behavior to the previous smaller environments, with the "greedy" agent performing the worse among all agent implementations.

Chapter 6

Conclusion

6.1 Summary

In this thesis, we attempted to construct a small but realistic model of a polarized social network, where users share their opinions through posts. We then introduced the problem of the recommender system, that recommends posts to users and how these recommendations can help reduce polarization by influencing users' opinions based on our opinion evolution model. Next, we identified how recommendations made without considering the current state of the network might lead to users being "trapped" in their own echo chamber.

We proposed a Deep Reinforcement Learning solution, more specifically DQN, in order to create agents that can assume the role of the recommender and reduce polarization. We tested many different configurations in order to evaluate our proposed system's success, finding that recommenders with "vision" can be more reliable than greedy recommenders, in addressing the polarization problem.

6.2 Future Work

An interesting line of research is to adapt this approach in real life polarized networks (ex. Twitter), with thousands of users. This would require a more realistic definition of the opinion update formula, through psychology or extensive observations of such networks and how they move with the passing of time, of how users are affected by other users' posts in their feed. The proper scaling would also need to be applied to the Agents that will have to de-polarize such networks as now the number of users have increased substantially. This entails a more careful tuning of the neural networks' hyper-paremeters (learning rate, number of layers and neurons, target network update etc.).

Another interesting change would be to increase the amount of users that receive recommendations at every time step t. Increasing the number of online users M, leads to an exponential increase of the size of Action Space $\mathcal{A}$. One way to solve this issue, is to assign to each individual user a separate DQN, effectively having M DQN Agents working independently (iDQN) each one tasked with making 1 (or more) recommendations to each online user. Although this implementation helps with the reduction of the Action Space size, it would require careful consideration as having too many Agents can introduce stability problems during training (each Agent sees the other Agents as part of the environment). Training will also be affected by the model that the Deep RL recommender is trying to learn so maybe the solution lies in more complex implementations of the DQN algorithm (Dueling DQN, Rainbow DQN etc).

References

1. Delaney, Kelsey, "The Plight of Social Media: An Analysis of the Effects Social Media has on Political Discourse" (2021). Honors Theses. 2454. https://digitalworks.union.edu/theses/2454

2. Datareportal. (2024, October). Digital 2024 October Global Statshot Report.

3. https://www.pewresearch.org/journalism/fact-sheet/news-plat- form-fact-sheet/

4. Emily Kubin, Christian von Sikorski, The Role of (Social) Media in Political Polarization: A Systematic Review, Annals of the International Communication Association, Volume 45, Issue 3, September 2021, Pages 188–206, https://doi.org/10.1080/23808985.2021.1976070

5. Nickerson, R. S. (1998). Confirmation bias: A ubiquitous phenomenon in many guises. Review of General Psychology, 2(2), 175–220. https://doi.org/10.1037/1089-2680.2.2.175

6. Jonathan L Freedman and David O Sears. Selective exposure. Advances in experimental social psychology, 2:57–97, 1965.

7. Eytan Bakshy et al., Exposure to ideologically diverse news and opinion on Facebook. Science 348, 1130-1132 (2015).

8. Grömping, M. (2014). 'Echo Chambers': Partisan Facebook Groups during the 2014 Thai Election. Asia Pacific Media Educator, 24(1), 39-59. https://doi.org/10.1177/1326365X14539185

9. Barberá P, Jost JT, Nagler J, Tucker JA, Bonneau R. Tweeting From Left to Right: Is Online Political Communication More Than an Echo Chamber? Psychol Sci. 2015 Oct;26(10):1531-42. https://doi.org/10.1177/0956797615594620. Epub 2015 Aug 21. PMID: 26297377.

10. Sasahara, K., Chen, W., Peng, H. et al. Correction to: Social influence and unfollowing accelerate the emergence of echo chambers. J Comput Soc Sc 5, 1779 (2022). https://doi.org/10.1007/s42001-022-00167-7

11. Bellman., R. E. (2003) [1957]. Dynamic Programming (Dover paperback ed.). Princeton, NJ: Princeton University Press. ISBN 978-0-486-42809-3.

12. Antonis Matakos, Evimaria Terzi, and Panayiotis Tsaparas. 2017. Measuring and moderating opinion polarization in social networks. Data Min. Knowl. Discov. 31, 5 (September 2017), 1480–1505. https://doi.org/10.1007/s10618-017-0527-9

13. Yue Wu, Linjiao Li, Qiannan Yu, Jiaxin Gan, Yi Zhang, Strategies for reducing polarization in social networks, Chaos, Solitons Fractals, Volume 167, 2023, 113095, ISSN 0960-0779, https://doi.org/10.1016/j.chaos.2022.113095. (https://www.sciencedirect.com/science/article/pii/S0960077922012747)

14. Cameron Musco, Christopher Musco, and Charalampos E. Tsourakakis. 2018. Minimizing Polarization and Disagreement in Social Networks. In Proceedings of the 2018 World Wide Web Conference (WWW '18). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 369–378. https://doi.org/10.1145/3178876.3186103

15. PLoS One. 2013 Nov 5;8(11):e78433. doi: 10.1371/journal.pone.0078433

16. Kiran Garimella: Polarization on Social Media (2018) https://aaltodoc.aalto.fi/items/e8f436bb-2dae-4d71-b098-da46e4c64a06

17. V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

18. D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel et al., "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," Science, vol. 362, no. 6419, pp. 1140–1144, 2018.

19. J. Tsitsiklis and B. Van Roy, "Analysis of temporal-diffference learning with function approximation," Advances in neural information processing systems, (NIPS) vol. 9, 1996.

20. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. arXiv:1412.6980v9

21. L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," SIAM review, vol. 60, no. 2, pp. 223–311, 2018

22. E. Biondi, C. Boldrini, A. Passarella, M. Conti "Dynamics of Opinion Polarization" May 2023 arXiv:2206.06134

23. M. H. DeGroot. Reaching a consensus. Journal of the American Statistical Association, 69(345):118–121, 1974.

24. Friedkin NE, Johnsen E (1990) Social influence and opinions. J Math Soc 15(3–4):193–206

25. A. V. Proskurnikov and R. Tempo, "A tutorial on modeling and analysis of dynamic social networks. Part I," Annu. Rev. Control, vol. 43, pp. 65–79, May 2017.

26. The Affective Tipping Point: Do Motivated Reasoners Ever "Get It"? Political Psychology, Vol. 31, No. 4, 2010 doi: 10.1111/j.1467-9221.2010.00772.x

27. Pariser, E.: The Filter Bubble: What the Internet Is Hiding from You. The Penguin Group (2011)

28. Rossi, W. S., Polderman, J. W., Frasca, P. (2022). The closed loop between opinion formation and personalised recommendations. IEEE Transactions on Control of Network Systems, 9(3), 1092-1103. https://doi.org/10.1109/TCNS.2021.3105616

29. Vendeville, A., Giovanidis, A., Papanastasiou, E., Guedj, B. (2023). Opening up Echo Chambers via Optimal Content Recommendation. In: Cherifi, H., Mantegna, R.N., Rocha, L.M., Cherifi, C., Miccichè, S. (eds) Complex Networks and Their Applications XI. COMPLEX NETWORKS 2016 2022. Studies in Computational Intelligence, vol 1077. Springer, Cham. https://doi.org/10.1007/978-3-031-21127-0_7

30. Cinus, F., Minici, M., Monti, C., Bonchi, F. (2022). The Effect of People Recommenders on Echo Chambers and Polarization. Proceedings of the International AAAI Conference on Web and Social Media, 16(1), 90-101. https://doi.org/10.1609/icwsm.v16i1.19275

31. Eschmann, J. (2021). Reward Function Design in Reinforcement Learning. In: Belousov, B., Abdulsamad, H., Klink, P., Parisi, S., Peters, J. (eds) Reinforcement Learning Algorithms: Analysis and Applications. Studies in Computational Intelligence, vol 883. Springer, Cham. https://doi.org/10.1007/978-3-030-41188-6_3