Reducing Polarization in Social Media

A Thesis

submitted to the designated
by the General Assembly
of the Department of Computer Science and Engineering
Examination Committee

by

Leonidas Boutsikaris

in partial fulfillment of the requirements for the degree of

COMPUTER SCIENCE AND ENGINEERING

University of Ioannina
March 2021

Examining Committee:

- Panayiotis Tsaparas, Associate Professor, Department of Computer Science and Engineering, University of Ioannina (Supervisor)
- Evaggelia Pitoura , Associate Professor, Department of Computer Science and Engineering, University of Ioannina
- Nikos Mamoulis, Associate Professor, Department of Computer Science and Engineering, University of Ioannina

Table of Contents

List of Algorithms

Abstract

Leonidas Boutsikaris, Diploma in Computer Science, Department of Computer Science and Engineering, University of Ioannina, Greece, March 2021.
Reducing Polarization in Social Media.
Advisor: Panayiotis Tsaparas, Assistant Professor.

We know for a fact that opinions are formed through social interactions. Online communities offer public access to social disputes on controversial matters that allows the study and moderation of them. The majority of studies in social networks are based in the Friedkin-Johnsen model.

Users of online communities are receiving biased information that amplify their own viewpoints. This creates a fragmented community and users interact only with individuals that hold the same opinions. In this thesis we use the polarization index to measure the polarization of a social graph.

We try to reduce the polarization by connecting individuals. We propose new social connections between different and extreme opinions by following the intuition of the Friedkin-Johnsen model.

Finally, probabilities are adopted to our heuristic algorithms that now make selections based on how probable is the acceptance of recommendation for a new social connection.

Περιληψη

Λεωνίδας Μπουτσικάρης, Δίπλωμα στην Επιστήμη των Υπολογιστών, Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Ελλάδα, Μάρτιος 2021.
Μείωση της Πόλωσης στα Κοινωνικά Δίκτυα.
Επιβλέπων: Παναγίωτης Τσαπάρας, Επίκουρος καθηγητής.

Γνωρίζουμε ως γεγονός το ότι οι γνώμες διαμορφώνονται μέσω των κοινωνικών συναναστροφών. Οι διαδικτυακές κοινότητες προσφέρουν δημόσια την πρόσβαση σε συζητήσεις για αμφιλεγόμενα ζητήματα που επιτρέπουν την μελέτη αλλά και τον έλεγχο τους. Η πλειοψηφία των ερευνών για τα κοινωνικά δίκτυα βασίζονται στο μοντέλο του Friedkin-Johnsen.

Οι χρήστες των διαδικτυακών κοινοτήτων λαμβάνουν μεροληπτικές πληροφορίες που ενισχύουν την δική τους οπτική. Αυτό δημιουργεί μία κατακερματισμένη κοινότητα και οι χρήστες αλληλεπιδρούν μόνο με άτομα που έχουν τις ίδιες γνώμες με αυτούς. Σε αυτήν την διπλωματική εργασία, θα χρησιμοποιήσουμε τον δείκτη πόλωσης για να μετρήσουμε το πόσο πολωμένο είναι ένα κοινωνικό γράφημα.

Προσπαθούμε να μειώσουμε την πόλωση με το να συνδέσουμε τα άτομα μεταξύ τους. Προτείνουμε νέες κοινωνικές συνδέσεις μεταξύ ατόμων που έχουν διαφορετικές και ακραίες γνώμες ακολουθώντας τον τρόπο που λειτουργεί το μοντέλο του Friedkin-Johnsen.

Τελικά, ενσωματώνουμε τις πιθανότητες στους ευριστικούς μας αλγορίθμους, που τώρα κάνουν επιλογές με βάση το πόσο πιθανό είναι να γίνει αποδεκτή μία πρόταση για μια νέα κοινωνική σύνδεση.

Chapter 1

Introduction and the Theory Behind Polarization

1.1 Introduction

Real world events such as Brexit and the 2016 U.S. presidential elections give us a clear hint about the polarization our society is witnessing.

Social media polarization has a strong effect on politics, opinion formation and how people interact with each other in a society. Users of social media are now receiving biased information that amplify their own viewpoints.

Enclosed in their filter bubble, they will ignore everyone else and only acknowledge opinions that fit their own reality. In combination with fake news a malicious entity could use social media as a tool to polarize certain groups of people for their own interest.

Our goal is to decrease the polarization by proposing new social connections. These additions are computed using heuristic algorithms.

In a real world setting, new social connections are not always accepted. For example we would not accept friend requests from people we barely know. This is relevant with link prediction. Link prediction is the problem of predicting the existence of a link between two entities in a network in the future. For example the ”People you may know” section on Facebook.

Link prediction algorithms are based on how similar two different nodes are, what features they have in common, how are they connected to the rest of the network or how many other nodes are connected to a single node. Link prediction is also used in recommendation systems and information retrieval.

By using the acceptance probabilities of a link prediction model we can define the expected decrease of the polarization. For computing these probabilities we will use graph embeddings.

1.2 Motivation

Polarization describes the division of people into two contrasting groups or sets of opinions or beliefs. The term is used in various domains such as politics and social studies. In social media settings, users tend to join communities of like-minded individuals.

In these settings the opinions of the users are amplified and reinforced by the continuous communication and recycling of the same view. These communities are referred to as echo chambers.

Inside an echo chamber users can easily find information that reinforces their existing opinion without being exposed to opposing views. Echo chambers can be created where information is exchanged, whether it’s online or in real life.

On social media almost anyone can quickly find like-minded people and countless news sources. This has made echo chambers far more numerous and easy to fall into.

Echo chambers online are referred to as filter bubbles. Filter bubbles are created by algorithms that keep track of the online behaviour of a user such as search histories, shopping activity and many more.

Social media will then use those algorithms to show content that is similar to what the user is already aligned with. This can lead users to adopt a more extreme version of their opinions.

1.3 Thesis Outline

We begin by exploring the Friedkin and Johnsen Model. This model uses the terms of internal and external opinion. By repeated averaging and combining these two values we can get the opinion vector of the graph. This is a vector that contains information about the opinions of the whole network. Then, by defining and computing the polarization index we can get an image of the social graph.

We then proceed and define our two problems. First, want to find the best k edges that will lead to the greatest reduction of the polarization index. The second problem incorporates the acceptance probabilities. We also observe that the addition of new edges between opposing opinions will not necessary decrease the polarization index and prove it with a counter-example.

Our heuristics are based on the intuition that our model has the biggest decrease when we connect different and extreme opinions and classify them in two categories.

In these two categories the heuristics do or do not recompute the graph structure after the addition of an edge. This is derived from the fact that when adding an edge to the network the structure of the graph changes.

The heuristics that consider network changes are the $Greedy$ the $FTGreedy$ and the $ExpressedOpinion$. These three are then modified into a batch version that does not consider network changes.

We continue by using Graph Embeddings and the $Node2Vec$ algorithm to compute acceptance probabilities. We use these probabilities in a modified version of our heuristics to compute the expected decrease of the polarization index.

Our heuristics are applied in 6 datasets of various topics and compared with each other. The Greedy heuristics cannot run on graphs that contain a lot of nodes due to time limitations.

1.4 Roadmap

At chapter 2 we look into the related work around polarization and decreasing polarization. We see how polarization is measured, the relation of polarization and random walks and how polarization can be combined with disagreement and conflict.

At chapter 3 we define the Friedkin and Johnsen model and the polarization index. We also define our 2 problems, the $k - Addition$ problem and the $k - Addition - Expected$. Then, we proceed to explore the monotonicity of the polarization index.

At chapter 4 we define our heuristics and at chapter 5 we continue by including acceptance probabilities in them.

At chapter 6 we take a look at our datasets and obtain the results of the polarization decrease for each heuristic in each dataset.

Chapter 2

Related Work

We will now present some work that is related to the work in this thesis.

2.1 Opinion models

How people form their opinions has long been the subject of research in the field of social sciences. Models of opinion formation and dynamics are being used by computer scientists to explore and quantify polarization, conflict and disagreement on social networks.

Opinion models study these quantities and how they change by manipulating the opinions and by changing the network structure of a set of nodes of the social graph.

Many of these models are based on the influence that goes with social interaction and the Friedking-Johnsen model is a very popular one.

2.2 Measuring the Polarization of a Network

In this paper the polarization index is defined. The direct link between the Friedkin-Johnsen model and random walks is also explored.

Two problems are introduced, the $ModerateInternal$ and the $ModerateExpressed$. When moderating opinions a small set of nodes T_s is being set to zero, in each problem, as their names suggests, internal or external opinions are set to zero. Two algorithms are proposed for the $ModerateInternalproblem$.

A greedy algorithm that finds the set T_s of nodes iteratively according to the biggest decrease it causes and the Binary Orthogonal Matching Pursuit (BOMP) algorithm.

For the $ModerateExpressed$ problem the same greedy algorithm is used. [1]

2.3 Polarization and Disagreement

Another way of looking at polarization is by combining it with disagreement. The main problem of minimising polarization and disagreement lies in the opinions of each user and how targeted ads and recommendations influence their opinions.

Considering the disagreement in combination with polarization a network can choose how to respond in different situations. Their recommendation system could choose between keeping the disagreement low or exposing users to radically different opinions.

There are situations that this optimisation can reduce the overall polarization-disagreement in the network by recommending edges in different parts of the network than the ones that agree with the human confirmation bias.

In this paper the disagreement of an edge is defined as the squared difference between the opinions of this edge. The total disagreement is defined as the sum of the disagreements and polarization is measured as a deviation from the average with the standard definition of variance.

The polarization-disagreement index is defined as follows $I_{G,s} = P_{G,s} + D_{G,s}$ which is the sum of the total disagreement and polarization. The objective is to minimise this index. This objective is addressed as an optimization problem. [2]

2.4 Quantifying and Minimizing Risk of Conflict in Social Networks

This paper addresses the main problem in the Friedkin-Johnsen model metrics. The external opinion of a user, which by definition is hard to measure, combined with the internal opinion which is impossible to be measured.

Another problem occurs in the editing of the social graph. When the social graph is edited its is done in a way that minimises the conflict of a certain social issue. This can lead to an increased conflict of one or more social issues inside the network.

Chen, Lijffijt and De Bie still use the Friedkin-Johnsen model to evaluate the network conflict but the quantifications depend only on the network topology in a way that the conflict can be reduced over all issues.

Worst-case(WCR) conflict risk and average-case conflict risk(ACR) are defined to represent two separate problems, how the network can be minimised in the worst case or in the average case scenario by altering the social graph.

These problems consider the measures of internal conflict, external conflict, and controversy. Internal conflict (ic) measures the difference of the internal and the expressed opinion of a user. $ic = \sum_i (z_i - s_i)^2$.

These measures are not independent. Reducing one of them results in the increase of another. This leads to the conservation law of conflict.

There are two methods of minimising the conflict of the network for each of the ACR and WCR problems. One is a gradient method that considers deleting and adding edges simultaneously and the other is a descent method that suggests deleting or adding a single edge. [3]

2.5 Reducing Controversy by Connecting Opposing Views

Garimella et al. rely on a measure of controversy that is shown to work reliably in multiple domains in contrast with other measures that focus on a single topic. The controversy measure consists of the following steps:

1. Given a topic t they create an endorsement graph $G = (V, E)$. This graph represents users who have generated content relevant to t. For example hashtags of a user.
2. The nodes of this graph a re partitioned in two disjoint sets X and Y. The partition is obtained using a graph-partition algorithm.
3. The last step, is computing the controversy measure through a random-walk, thus creating the controversy score RWC. This score is defined as the difference of the probability that a random walk starting on one side of the partition will stay on the same side and the probability that the random walk will cross to the other side. A personalised PageRank is used where the restart probabilities are set to a random vertex of each side.

Garimella et al. states that real graphs often have a star-like structure. Small number of highly popular vertices have a lot of incoming edges. These nodes can be seen as thought leaders and their followers. It is shown that connecting the high degree vertices minimises the RWC score.

Probabilities are also incorporated in the sense that a new edge addition may be not accepted by the user. [4]

Chapter 3

Premilinaries and Problem Definition

3.1 The Friedkin and Johnsen Model

The Friedkin-Johnsen model is a very popular extension of the DeGroot's model. This model uses information about the opinion of the user assuming there is an internal and external opinion. The internal opinions cannot change and is the specific opinion of an individual for a certain matter. On the other hand the expressed opinion is influenced by social interactions.

The expressed opinion of a user is computed as a weighted average of the external opinions of the neighbourhood of the user, for example, the opinions of the users friend list or the accounts the user follows.

The opinions of the users are stored in a vector. This vector is a metric for the whole social graph and can give us insights about its current situation.

The vector values range from [-1,1]. Values closer to the range limits indicate bigger polarization. Polarized graphs create groups of nodes that are strongly connected with each other.

Let $G = (V, E)$ be a connected undirected graph representing a network. Let z be the vector of expressed opinions for the whole network. Each value of the vector represents a node and can be computed with the opinion-formation model of Friedkin and Johnsen as follows.

$$z_i = \frac{w_{ii} * si + \sum_{j \epsilon N(i)} w_{ij} * z_j}{w_{ii} + \sum_{j \epsilon N(i)} w_{ij}} \tag{3.1}$$

Where s_i denotes the internal and z_i the expressed opinion of a user. The internal opinion of a user corresponds to the views that inherently holds for a controversial topic while the expressed refers to the views that the user shares on a social network with his friends.

The length of the opinion vector $||z||^2$ measures the polarization of the network. To make the polarization independent of its network we can normalize it by dividing it with the length of the vector z.

An equivalent way of obtaining the vector z from a graph is the following: if L is the laplacian matrix of a graph $G = (V, E)$, and I is the identity matrix, then $z = (L + I)^{-1}S$ [5].

3.2 Measuring the polarization

We use the definition of the polarization index by Matakos et al. The polarization is measured by its distance from a neutral opinion.

We can quantify this with the length of the vector of the second norm L_2^2 [1].

$$\pi(z) = ||z||_2^2 \tag{3.2}$$

This value can be independent of the network if we normalize it by dividing with the size of the graph.

3.3 Problem Definition

Let $G = (V, E)$ be a connected undirected graph representing a network. Let z be the vector of expressed opinions for the whole network and $\pi(z) = ||z||_2^2$ the polarization index of this social graph.

Problem 1 [k-Addition]. Let $C \subseteq V \times V$ a set of edges that are not in the graph. We want to find a subset of $S \subseteq C$ of k edges whose addition to a graph G leads to the greatest reduction of $\pi(z)$.

3.3.1 Including probabilities into the problem

Problem 1 is trying to find edges that will minimize the polarization index. We must not take for granted that these edges will be accepted. For example a social media user could reject a new follow/friend request. This leads us to consider additions with the expectation of being accepted.

The expected polarization can be defined as follows.

$$E[\pi(z)] = P(u, v) * Val \tag{3.3}$$

Where $P(u, v)$ is the probabilty of u and v forming an edge and Val is a value specific to our heuristic algorithms. For example Val can be the polarization decrease after the edge addition of u and v or the absolute distance of their opinions.

Problem 2 [K-Addition-Expected]. Given a graph $G = (V, E)$ and an integer k, we want to find a set of k edges $E' \subseteq V \times V\ E$ that when added to G creates a new graph $G' = (V, E \cap E')$ so that the expected polarization score $E[\pi(z)]$ is minimized.

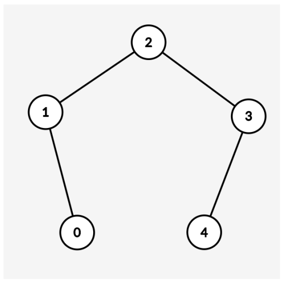
3.4 Monotonicity of the Problem

We observe that $\pi(z)$ is not monotone with respect to the edge additions. This means that adding an edge will not necessarily decrease the polarization index.

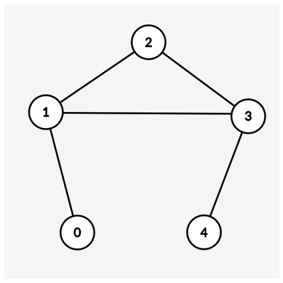
Lemma 3.1. The polarization index does not necessarily decrease after an edge addition between opposing views.

We will show this with a counter example. In the network 3.1 nodes 0, 2 and 3 have a value of $s_i = -1$, and nodes 2 and 4 have a value of $s_i = +1$. For both examples we assume that $w_{ii} = w_{ij} = w_{ji} = 1$ and n the number of nodes.

We will now compute the polarization index of the original graph



(a)



(b)

Figure 3.1: Edge addition between opposed opinions.

$$(L+I)^{-1}s = z = \begin{pmatrix} \frac{-27}{55} \\ \frac{1}{55} \\ \frac{-5}{11} \\ \frac{-21}{55} \\ \frac{17}{55} \end{pmatrix}, \qquad \pi(z) = 0.13785123966 \tag{3.4}$$

We will now compute the polarization index after the addition of the edge $1 \to 3$.

$$(L+I)^{-1}s = z = \begin{pmatrix} \frac{-53}{99} \\ \frac{-7}{99} \\ \frac{-5}{11} \\ \frac{-29}{99} \\ \frac{35}{99} \end{pmatrix}, \qquad \pi(z) = 0.14180185695 \tag{3.5}$$

We can see an increase of the polarization index after adding this particular edge. This example was discovered after brute-forcing different graph topologies with different combinations of opinion values.

Chapter 4

Heuristics

4.1 Introduction

In this section we consider a greedy algorithm and some heuristics for minimising $\pi(z)$. All the heuristics use the intuition that connecting the most extreme opinions of each community draw both of them into neutrality.

We can classify these algorithms based on how they perceive the network when exploring new edge additions. This is derived from the fact that when adding an edge to the network the structure of the graph is changes.

This leads to a different Z vector. The heuristics can choose to recompute the Z vector or not.

4.2 Heuristics that recompute the opinion vector

We begin with a Greedy algorithm. Greedy algorithms work in stages and during each stage a choice is made which is locally optimal. The Greedy algorithm computes the decrease in $\pi(z)$ and selects the edge with the largest decrease every time.

After finding the best edge, the *Greedy* algorithm adds this edge to the graph. This result in a change of the network structure. Then a recomputation of the z vector is happening and the procedure is repeated.

To reduce running time, in our implementation, we consider only addition of different opinions. This way the heuristics do not have to traverse all the existing edge combinations

Algorithm 4.1 Greedy

INPUT: Graph $G(V, E)$; k number of edges to add;

OUTPUT: A set S of k edges to be added to G that minimize the polarization index $\pi(z)$

```
1: for $i = 1 : k$ do
2:     for each edge in $|V| \times |V| \setminus E$ do
3:         Compute the decrease of $\pi(z)$ if edge is added to $G$;
4:     end for
5:     Select the edge with the largest decrease and add it to $G$.
6: end for
7: Return the set of edges that were selected.
```

A second heuristic we consider is the $FirstTopGreedy$. Let X be the set of nodes that their expressed opinions ϵ [-1,0) sorted by increasing order and Y the set of nodes that their expressed opinions ϵ (0,1] sorted by decreasing order.

This heuristic is taking the first K nodes of X and Y, resulting in $K \times K$ nodes.

A greedy search is performed in this smaller space. This allows the $FirstTopGreedy$ to reduce the amount of time spend searching for the best edge to add.

Algorithm 4.2 FirstTopGreedy

INPUT: Graph $G(V, E)$; k number of edges to add;
X, the set of nodes that their expressed opinions ϵ [-1,0) sorted by increasing order
Y, set of nodes that their expressed opinions ϵ (0,1] sorted by decreasing order

OUTPUT: A set S of k edges to be added to G that minimize the polarization index $\pi(z)$

```
1: $A \leftarrow$ first $k$ items of $X$
2: $B \leftarrow$ first $k$ items of $Y$
3: for $i = 1 : k$ do
4:     $Decrease \leftarrow EmptyList$;
5:     for each edge in $|A| \times |B| \setminus E$ do
6:         Compute the decrease of $\pi(z)$ if edge is added to the graph;
7:         Append the decrease on the $Decrease$ list;
8:     end for
9:     Select the edge with the largest decrease from the $Decrease$ list.
10:    Add this edge to the graph.
11: end for
12: Return the set of edges that were selected.
```

Last we consider two heuristics that choose edges based on the value of the expressed opinion of their nodes. Let u and v two nodes that we want to connect and Z_u and V_u their expressed opinions respectively. The Distance of their opinions can be defined as $D = |Z_u - V_u|$.

This heuristic computes the distance between every edge candidate and then chooses to add the edge with the maximum distance.

Algorithm 4.3 ExpressedOpinion

INPUT: Graph $G(V, E)$; k number of edges to add

OUTPUT: A set S of k edges to be added to G that minimize the polarization index $\pi(z)$

```
1: for $i = 1 : k$ do
2:     $EdgesToAdd \leftarrow EmptyList$;
3:     Compute the $z$ values.
4:     for each edge in $|V| \times |V| \setminus E$ do
5:         Append to $EddgesToAdd$ the value $D = |Z_u - V_u|$.
6:     end for
7:     $Sorted \leftarrow sort(EdgesToAdd)$ by decreasing order;
8:     Add the first edge of $EdgesToAdd$ to the graph
9: end for
10: Return the set of edges that were selected.
```

4.3 Heuristics that do not recompute the opinion vector

Bellow we explore some heuristics that do not consider the network changes and thus can run more easily in larger datasets. Computing the $\pi(z)$ is an expensive operation due to the computation of the inverse matrix.

We compute the opinion vector only once and we sort the edges according to the decrease. Then we select the top-K edges.

At first we can see a variation of the *Greedy* algorithm. Its implementation is similar to the *Greedy* but without recomputing the $\pi(z)$ at each round.

Algorithm 4.4 GreedyBatch

INPUT: Graph $G(V, E)$; k number of edges to add;

OUTPUT: A set S of k edges to be added to G that minimize the polarization index $\pi(z)$

```
1: $EdgesToAdd \leftarrow EmptyList$;
2: for each edge in $|V| \times |V| \backslash E$ do
3:     Compute the decrease of $\pi(z)$ if edge is added to $G$ and add it to $edgesToAdd$
4: end for
5: $Sorted \leftarrow sort(EdgesToAdd)$ by decreasing order;
6: Select the $k$ edges with the largest decrease and add it to $G$.
7: Return the set of edges that were selected.
```

We continue by using a variation of the $FirstTopGreedy$.The $FirstTopGreedyBatch$ heuristic.

$FirsTopGreedy$ works exactly like $GreedyBatch$ with the difference that the opinion vector is not recomputed. Let X be the set of nodes that their expressed opinions ϵ [-1,0) sorted by increasing order and Y the set of nodes that their expressed opinions ϵ (0,1] sorted by decreasing order.

This heuristic is taking the first K nodes of X and Y, resulting in $K \times K$ nodes.

Algorithm 4.5 FirstTopGreedyBatch

INPUT: Graph $G(V, E)$; k number of edges to add;
X, the set of nodes that their expressed opinions ϵ [-1,0) sorted by increasing order
Y, set of nodes that their expressed opinions ϵ (0,1] sorted by decreasing order

OUTPUT: A set S of k edges to be added to G that minimize the polarization index $\pi(z)$

```
1: $A, B \leftarrow$ first $k$ items of $X$ , $Y$
2: $Decrease \leftarrow EmptyList$;
3: for each edge in $|A| \times |B| \setminus E$ do
4:     Compute the decrease of $\pi(z)$ if edge is added to the graph;
5:     Append the decrease on the $Decrease$ list;
6: end for
7: Add the first $k$ edges from the $Decrease$ list and return them.
```

Finally we consider the batch version of the *ExpressedOpinion* algorithm.
The *ExpressedOpinionBatch*, like the previous heuristics, does not recompute the opinion vector.

Algorithm 4.6 ExpressedOpinionBatch

INPUTS: Graph $G(V, E)$; k number of edges to add

OUTPUT: A set S of k edges to be added to G that minimize the polarization

```
1: $EdgesToAdd \leftarrow EmptyList$;
2: for  each edge in $|V| \times |V| \backslash E$ do
3:     Append to $EddgesToAdd$ the value $D = |Z_u - V_u|$.
4: end for
5: $Sorted \leftarrow sort(EdgesToAdd)$ by decreasing order;
6: Add the first $k$ edges of $EdgesToAdd$ to the graph and return them.
```

Chapter 5

Including Probabilities in the heuristics

5.1 Graph Embeddings

A graph embedding is the transformation of the properties of the graphs to a vector or a set of vectors. The embedding will capture the topology of the graph and will consider the relationship between nodes. The embedding will be used to make predictions on the graph.

Machine learning on graphs is limited while vector spaces have a much bigger toolset available. In essence embeddings are compressed representations in a vector that has a smaller dimension.

5.1.1 Word2Vec

At first we have to define Word2Vec. Suppose we have a sentence of words. For a certain task a simple neural network with a single hidden layer is created.

The trained neural network is not actually used for the task that we trained it on.

The goal is to learn the weights of the hidden layer. These weights represent the ”word vectors”.

After giving the neural network a word in the middle of a sentence, it is trained to look for nearby words and pick a random one. The network is going to give the probability for every word in our vocabulary of being inside a window size we set.

The output probabilities are going to relate to how likely it is to find each vocabulary word near our input word. The neural network is trained by feeding it word pairs found in training examples.

The hidden layer of this model is operating as a lookup table. The output of the hidden layer is just the “word vector” for the input word.The word vector will then get fed to the output layer. The output layer is a softmax regression classifier. Each output neuron will produce an output between 0 and 1, and the sum of all these output values will add up to 1. If two different words have very similar context then our model needs to output very similar results for these two words.

5.1.2 DeepWalk

After defining Word2Vec we can use its logic in graphs. DeepWal k uses random walks to produce embeddings. The random walk starts in a selected node and then moves to a random neighbour from a current node with certain number of steps. The method consists of three steps.

- Sampling: A graph is sampled with random walks. Authors show that it is sufficient to perform from 32 to 64 random walks from each node.
- Training skip-gram: Random walks are comparable to sentences in word2vec approach. The skip-gram network accepts a node from the random walk a vector as an input and maximizes the probability for predicting neighbour nodes.
- Computing embeddings: Embedding is the output of a hidden layer of the network. The DeepWalk computes embedding for each node in the graph.

DeepWalk method performs the walks randomly and that means that embeddings do not preserve the local neighbourhood. Node2vec approach fixes that [6].

5.1.3 Node2Vec

Node2vec is a modification of DeepWalk with a small difference in the implementation of random walks. There are two parameters introduced, P and Q.

Parameter Q defines how probable is that the random walk will explore the undiscovered part of the graph, while parameter P defines how probable is that the random walk will return to the previous node and retain a locality[7].

5.2 Methodology

Our objective is to predict whether there would be a link between 2 unconnected nodes. At first we will find the pairs of nodes that don't have a link between them. The next step is to label these pairs. This is needed for preparing a training dataset. The edges that are present in the graph will be labeled as 1 (positive samples) and the unconnected node pairs as 0 (negative samples).

After the labelling we will use the node2vec algorithm to extract node features from the graph. For computing the features of an edge we can add up the features of the nodes of that pair. These features will be trained with a logistic regression model.

After the model is trained we will obtain a dictionary containing the probability of an edge being accepted. The expected decrease can now be defined.

$$E[\pi(z)] = P(u, v) * Val \tag{5.1}$$

Where $P(u, v)$ is the probabilty of u and v forming an edge and Val is a value that can be either the polarization decrease, the absolute distance of the expressed opinions of nodes u and v or the multiplication of their values.

5.3 Expected Heuristics

The expected heuristics use the acceptance probabilities from the $Node2vec$ algorithm.

All heuristics are edited to include this probability when computing the polarization decrease, thus computing the expected polarization decrease. Bellow we can see the edited heuristics.

Algorithm 5.1 pGreedy

INPUT: Graph $G(V, E)$; k number of edges to add,
Dictionary D that holds acceptance probabilities for every edge;

OUTPUT: A set S of k edges to be added to G that minimize the polarization index $\pi(z)$

```
1: for $i = 1 : k$ do
2:     for each edge in $|V| \times |V| \backslash E$ do
3:         Compute the decrease of $\pi(z) * D[(u, v)]$ if edge is added to $G$;
4:     end for
5:     Select the edge with the largest decrease and add it to $G$.
6: end for
7: Return the set of edges that were selected.
```

Algorithm 5.2 pFirstTopGreedy

INPUT: Graph $G(V, E)$; k number of edges to add,
Dictionary D that holds acceptance probabilities for every edge; X, the set of nodes that their expressed opinions ϵ [-1,0) sorted by increasing order
Y, set of nodes that their expressed opinions ϵ (0,1] sorted by decreasing order

OUTPUT: A set S of k edges to be added to G that minimize the polarization index $\pi(z)$

```
1:  $A \leftarrow$ first $k$ items of $X$
2:  $B \leftarrow$ first $k$ items of $Y$
3:  for $i = 1 : k$ do
4:      $Decrease \leftarrow EmptyList$;
5:      for each edge in $|A| \times |B| \backslash E$ do
6:          Compute the decrease of $\pi(z) * D[(u, v)]$ if edge is added to the graph;
7:          Append the decrease on the $Decrease$ list;
8:      end for
9:      Select the edge with the largest decrease from the $Decrease$ list.
10:     Add this edge to the graph.
11: end for
12: Return the set of edges that were selected.
```

Algorithm 5.3 pExpressedOpinion

INPUT: Graph $G(V, E)$; k number of edges to add,
Dictionary D that holds acceptance probabilities for every edge;

OUTPUT: A set S of k edges to be added to G that minimize the polarization index $\pi(z)$

```
1: for $i = 1 : k$ do
2:    $EdgesToAdd \leftarrow EmptyList$;
3:    Compute the $z$ values.
4:    for each edge in $|V| \times |V| \backslash E$ do
5:       Append to $EddgesToAdd$ the value $|Z_u - V_u| * D[(u, v)]$.
6:    end for
7:    $Sorted \leftarrow sort(EdgesToAdd)$ by decreasing order;
8:    Add the first edge of $EdgesToAdd$ to the graph
9: end for
10: Return the set of edges that were selected.
```

Algorithm 5.4 pGreedyBatch

INPUT: Graph $G(V, E)$; k number of edges to add,
Dictionary D that holds acceptance probabilities for every edge;

OUTPUT: A set S of k edges to be added to G that minimize the polarization index $\pi(z)$

```
1: $EdgesToAdd \leftarrow EmptyList$;
2: for each edge in $|V| \times |V| \backslash E$ do
3:    Compute the decrease of $\pi(z) * D[(u, v)]$ if edge is added to $G$ and add it to
      $edgesToAdd$
4: end for
5: $Sorted \leftarrow sort(EdgesToAdd)$ by decreasing order;
6: Select the $k$ edges with the largest decrease and add it to $G$.
7: Return the set of edges that were selected.
```

Algorithm 5.5 pFirstTopGreedyBatch

INPUT: Graph $G(V, E)$; k number of edges to add,
Dictionary D that holds acceptance probabilities for every edge;
X, the set of nodes that their expressed opinions ϵ [-1,0) sorted by increasing order
Y, set of nodes that their expressed opinions ϵ (0,1] sorted by decreasing order
OUTPUT: A set S of k edges to be added to G that minimize the polarization index $\pi(z)$

```
1: $A, B \leftarrow$ first $k$ items of $X$ , $Y$
2: $Decrease \leftarrow EmptyList$;
3: for  each edge in $|A| \times |B| \backslash E$ do
4:    Compute the decrease of $\pi(z) * D[(u, v)]$ if edge is added to the graph;
5:    Append the decrease on the $Decrease$ list;
6: end for
7: Add the first $k$ edges from the $Decrease$ list and return them.
```

Algorithm 5.6 ExpressedOpinionBatch

INPUTS: Graph $G(V, E)$; k number of edges to add,
Dictionary D that holds acceptance probabilities for every edge;

OUTPUT: A set S of k edges to be added to G that minimize the polarization

```
1: $EdgesToAdd \leftarrow EmptyList$;
2: for  each edge in $|V| \times |V| \backslash E$ do
3:    Append to $EddgesToAdd$ the value $|Z_u - V_u| * D[(u, v)]$.
4: end for
5: $Sorted \leftarrow sort(EdgesToAdd)$ by decreasing order;
6: Add the first $k$ edges of $EdgesToAdd$ to the graph and return them.
```

Chapter 6

Experiments

6.1 Datasets

In this section we consider datasets that are separated in two opposing communities. The information about the opinions of each member of this community is known. Thus, we can assign internal opinions -1 and 1 to the nodes depending on their community membership[1]. We consider the following.

1. The Karate dataset, that represents the friendships between the members of a karate club at a US university. This network is split in two equal size polarized communities around two rival karate instructors.

2. The Books dataset, that is a network of US politics books. These books were published near the 2004 presidential election and sold by Amazon. These Books are classified as ”Liberal”, ”Conservative”, or ”Neutral”. There are in total 43 liberal books, 49 conservative and 13 neutral.

3. The Blogs dataset, a network of hyperlinks between online blogs on US politics. Blogs are classified as either Liberal or Conservative.

4. The Elections dataset, this dataset is the network between the Twitter followers of Hillary Clinton and Donald Trump collected in the period 15/12/2016-15/01/2017 – around the time of the 2016 presidential elections. Members of this network are assigned an internal opinion of 1 or -1 based on which one of the two candidates they follow. We took a subsampled portion that has be done by Matakos, et al [1].

5. The beefban dataset, a hashtag that Twitter users used in March 2015 to signal that their posts referred to a decision by the Indian government about the consumption of beef meat in India.

6. The GermanWings dataset, a hashtag that Twitter users used after the crash of GermanWings Flight 9525.

6.2 Dataset statistics

Table 6.1: Stats

Name	# of Nodes	# of Edges	Avg. Degree	$\pi(z)$
Karate	34	78	4.5882	0.33964
books	105	441	8.4	0.43429
beefban	799	6026	15.0839	0.30326
polblogs	1490	16718	22.4403	0.30983
GermanWings	2111	7329	6.9436	0.44479
ClintonTrump	2832	18551	13.1010	0.07582

6.3 A Visualisation of Edge Additions

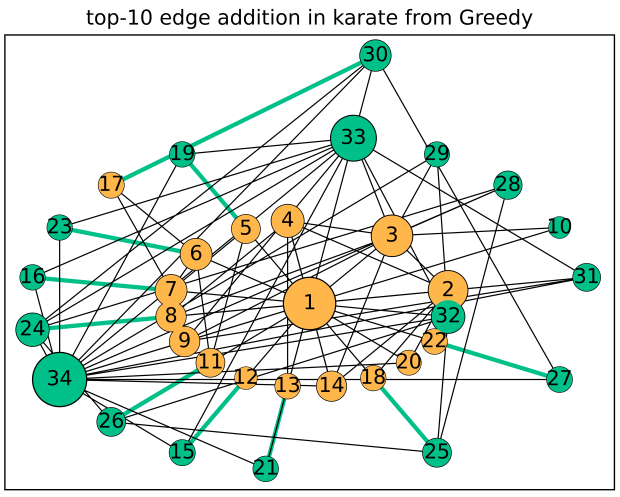


Figure 6.1: the top-10 edges proposed by the greedy algorithm

6.4 Experiments

All experiments were made with an 2,7 GHz Dual-Core Intel Core i5 on the PyCharm IDE. We can only experiment with the *Karate* and the *Books* dataset on all the heuristics. The *Greedy* algorithm cannot run on the rest of the datasets because they contain thousands of nodes. The *Greedy* algorithm needs to consider changes in the network structure so it is impossible to compute the polarization so many times.

The same applies for the *GreedyBatch* algorithm. Although the *GreedyBatch* algorithm can run on the *beefban* dataset that contains 799 nodes it would take a lot of time even for the *polblogs* dataset that has 1490 nodes.

The *FirstTopGreedy* and the *ExpressedOpinion* can run in all datasets and us with a fairly good decrease in polarization compared to the batch algorithms. This decrease can be greater if we consider edge additions that are proportional to the size of the dataset. Greater number of edges would make *FirstTopGreedy* nonrunnable. The *ExpressedOpinion* can run in our larger datasets with big number of additions.

6.4.1 Polarization decrease

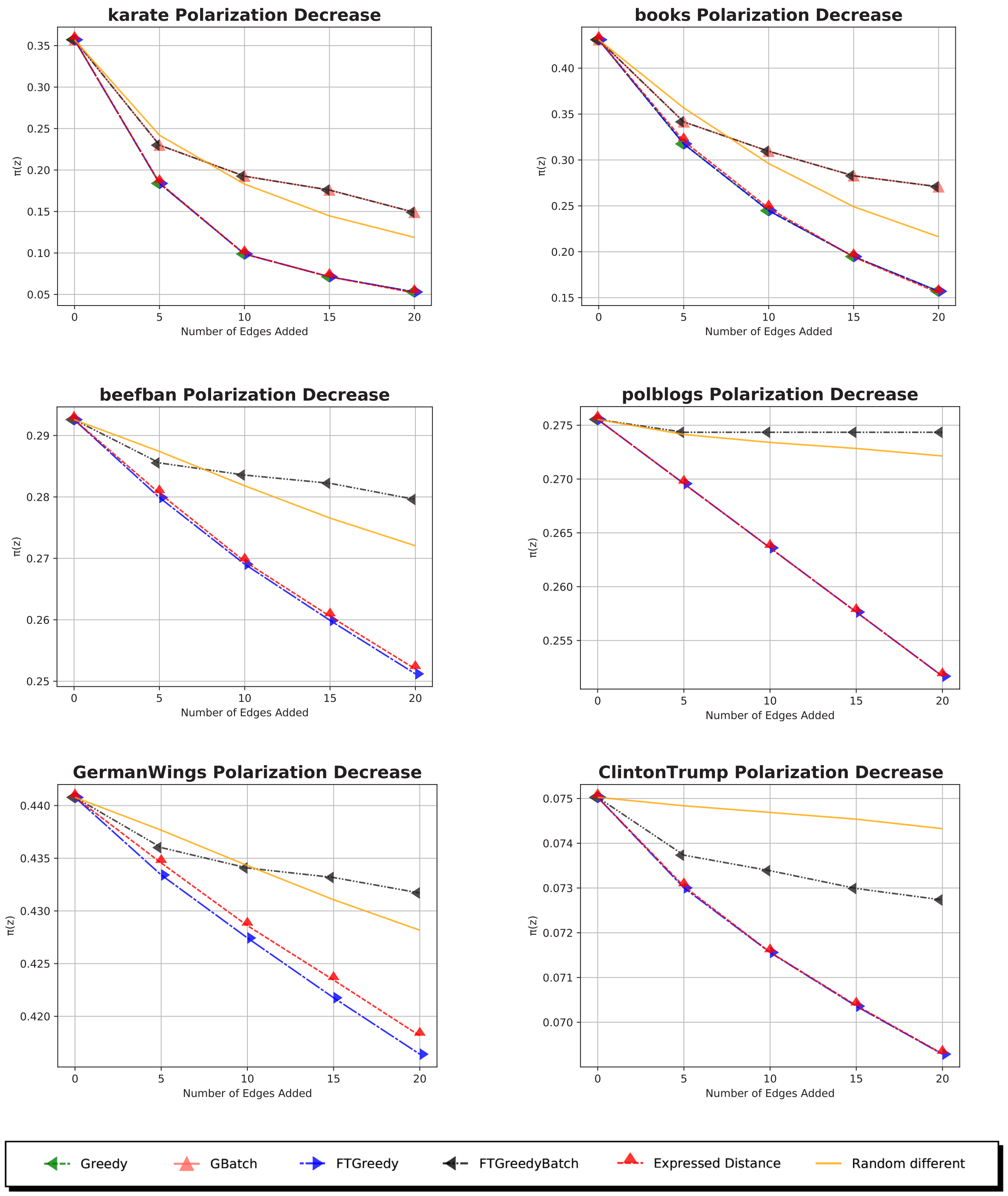


Figure 6.2: Comparison of the heuristics between datasets

We start by applying the heuristics described at 4.1 in these 6 datasets. The algorithms perform as expected. The greedy ones have better results but are expensive in time.We can not run the $Greedy$ algorithms in larger datasets due to time limitations.

- The Expressed Opinion heuristic that is based on the distance of the opinions performs very well and close to $Greedy$ and are cheap on time.
- Batch algorithms perform very poorly, even worse than Random. When adding a new edge the Batch algorithms do not recompute the Z vector. In later edge selections they have a false view of the opinions of the network. For example in the $ExpressedOpinionBatch$ the nodes that have the most extreme opinions of each side may reused even if their value is changed after an addition and are no longer the ones with the most extreme values.

6.4.2 Expected polarization decrease

Bibliography

[1] Matakos, Terzi, and Tsaparas, “Measuring and moderating opinion polarization in social networks,” Data Mining and Knowledge Discovery, vol. 15, 2017.

[2] C. Musco, C. Musco, and Tsourakakis. (2017) Minimizing polarization and disagreement in social networks. [Online]. Available: https://arxiv.org/abs/1712.09948

[3] Chen and D. B. Lijffijt. (2018) Quantifying and minimizing risk of conflict in social networks. [Online]. Available: https://dl.acm.org/doi/proceedings/10.1145/3219819

[4] Garimella, Morales, Gionis, and Mathioudakis. (2018) Reducing controversy by connecting opposing views. [Online]. Available: https://arxiv.org/abs/1611.00172

[5] Bindel, Kleinberg, and Oren. (2015) How bad is forming your own opinion? [Online]. Available: https://arxiv.org/pdf/1203.2973.pdf

[6] S. Perozzi, Al-Rfou. (2014) Deepwalk: Online learning of social representations. [Online]. Available: https://arxiv.org/pdf/1403.6652.pdf

[7] G. Leskovec. (2016) node2vec: Scalable feature learning for networks. [Online]. Available: https://cs.stanford.edu/~jure/pubs/node2vec-kdd16.pdf