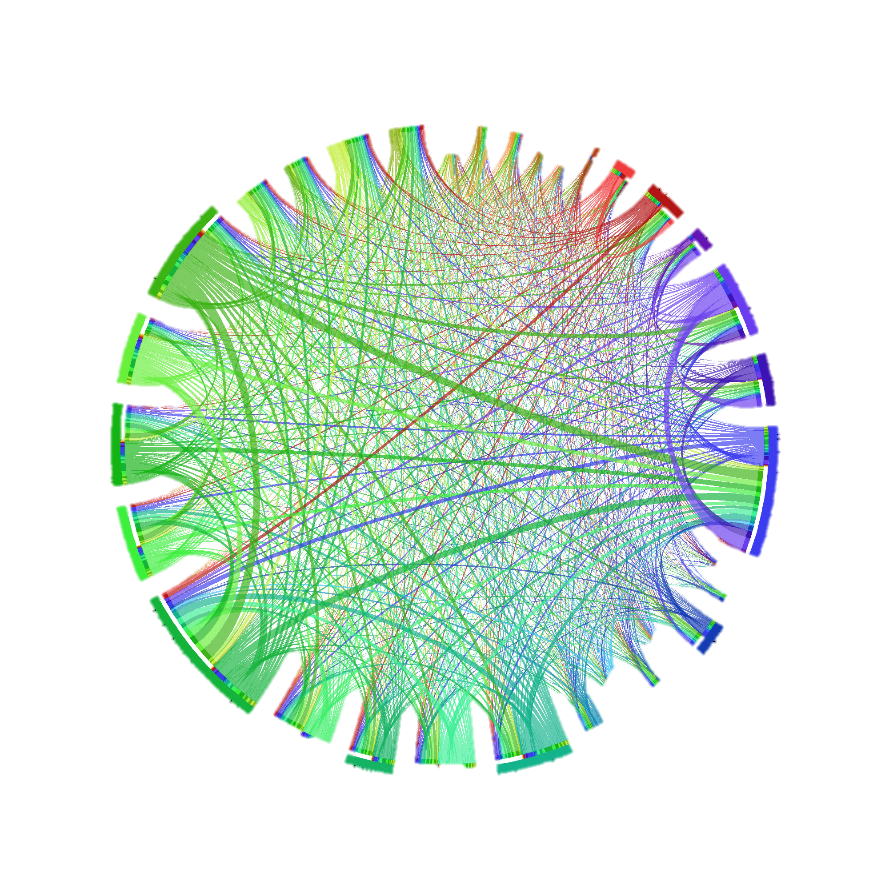
A blue and white sign with white text

Description automatically generated with low confidence

Software Engineering Department

Braude College

**Citation Prediction using GAN**



Final Project in Software Engineering – Course 61998

Biran Fridman 314877580

Ilya Lev 316154285

Supervisors:

Dr. Renata Avros

Prof. Zeev Volkovich

Table of Contents

[Abstract 4](#_Toc137207143)

[1. Introduction 4](#_Toc137207144)

[2. Background and Related Work 5](#_Toc137207145)

[2.1 Link Prediction 5](#_Toc137207146)

[2.2 Complex network Graph 5](#_Toc137207147)

[2.2.1 Proximity Order 5](#_Toc137207148)

[2.3 Generative countermeasure network (GAN) 6](#_Toc137207149)

[2.4 Node Similarity indexes 7](#_Toc137207150)

[2.4.1 Adamic-Adar (AA): 7](#_Toc137207151)

[2.4.2 Local-Path (LP): 7](#_Toc137207152)

[2.4.3 Katz index (KZ): 7](#_Toc137207153)

[2.5 Hierarchical Network Graph 8](#_Toc137207154)

[2.6 Hierarchical Network algorithm 8](#_Toc137207155)

[2.6.1 Edge Folding 8](#_Toc137207156)

[2.6.2 Vertex Merging 9](#_Toc137207157)

[2.7 Network graph layering algorithm – Netlay 9](#_Toc137207158)

[2.8 Node Embedding 10](#_Toc137207159)

[2.8.1 Random walk 10](#_Toc137207160)

[2.8.2 Node2Vec Algorithm 11](#_Toc137207161)

[2.9 EmbedGAN 12](#_Toc137207162)

[2.9.1 Discriminator optimization 12](#_Toc137207163)

[2.9.2 Builder Sampling Strategy 12](#_Toc137207164)

[2.9.3 EmbedGAN Algorithm 13](#_Toc137207165)

[2.10 GAHNRL Algorithm 15](#_Toc137207166)

[3. Expected Achievements 16](#_Toc137207167)

[3.1 Algorithm Development 16](#_Toc137207168)

[3.2 Success Criteria 16](#_Toc137207169)

[3.3 Unique features and Research Challenges 16](#_Toc137207170)

[4. Engineering Process 17](#_Toc137207171)

[4.1 Problem Identification 17](#_Toc137207172)

[4.2 Literature review 17](#_Toc137207173)

[4.3 Data Collection 17](#_Toc137207174)

[4.3.1 Data Pre-Processing 18](#_Toc137207175)

[4.3.2 Graph data representation 18](#_Toc137207176)

[4.3.3 GAN model training and evaluation 18](#_Toc137207177)

[4.4 Data collection and model training flow chart 19](#_Toc137207178)

[4.5 Algorithm pseudo code 21](#_Toc137207179)

[5. Evaluation / Verification Plan 22](#_Toc137207180)

[5.1 Testing Approach 22](#_Toc137207181)

[5.2 Evaluation 22](#_Toc137207182)

[5.3 Testing / Evaluation Process 22](#_Toc137207183)

[6. References 24](#_Toc137207184)

# Abstract

This project addresses the issue of including irrelevant citations in scholarly articles, which undermines research integrity and slows down access to reliable information and in some cases may even have a negative effect on the writers. Existing approaches to tackle this problem, such as manual screening and expert reviews do exist, but these methods suffer from subjectivity and inadequate accuracy. By utilizing Generative Adversarial Networks (GANs) and gaining insights from "A Link Prediction Algorithm Based on GAN" [[1](#Refernces)].

This project introduces a different approach. It involves creating a citation graph and utilizing the GAN-based algorithm to predict missing citations, enabling the assessment of their relevance. Successful predictions indicate probable relevance, while unsuccessful predictions suggest likely irrelevance.

# Introduction

In today's digital age, scholarly articles play a critical role in spreading knowledge and advancing research. However, one persistent issue that researchers face is the inclusion of irrelevant citations within their articles. Irrelevant citations refer to references that do not directly contribute to the content or purpose of the article, often added for various reasons such as promoting other articles, accommodating personal relationships, or meeting external pressures.

The problem of citing irrelevant articles poses significant challenges for both authors and readers. For authors, the inclusion of irrelevant citations can undermine the integrity of their research and dilute the focus of their work. It becomes crucial for researchers to identify and exclude such citations to maintain the quality and relevance of their articles. On the other hand, readers heavily rely on citations to navigate the vast landscape of scientific literature. Irrelevant citations not only impede the readers' ability to access reliable information but also lead to time wasted in pursuing irrelevant references.

Motivation for this project arises from the need to address the persisting issue of citing irrelevant articles. While this problem has been acknowledged in the research community, finding an effective and automated solution has remained elusive. Various approaches have been explored to combat this problem, including manual screening of citations, expert review processes, and algorithmic techniques. However, these existing methods suffer from limitations such as subjectivity, time-consuming nature, and insufficient accuracy.

In the realm of link prediction, a promising technique has emerged in the form of Generative Adversarial Networks (GANs). GANs have shown remarkable success in modeling complex relationships within data, making them a suitable candidate for tackling the problem of identifying irrelevant citations. Leveraging the insights from the article "A Link Prediction Algorithm Based on GAN" [[1](#Refernces)].

The proposed solution involves adapting the GAN-based link prediction algorithm to construct a citation graph connecting articles based on their citations. By systematically removing each A blue and white sign with white text

Description automatically generated with low confidencecitation from an article and predicting the missing citation using the algorithm, the relevance of the citation to the article can be assessed. Successful prediction of the removed citation indicates a high likelihood of relevance while unsuccessful predictions suggest a high likelihood of irrelevance, and thus highlighting citations added for reasons other than scholarly merit.

The key individuals that will benefit from this solution are researchers, authors, academic institutions, publishers, and readers.

In this research project, we aim to help academic researchers tackle the problem of citing irrelevant articles. Our goal is to improve the quality and relevance of scientific literature and make it easier for researchers to find valuable information. By using a special algorithm based on GANs, we can analyze the citations in articles and identify those that may not be relevant. This will save researchers time and effort by streamlining the citation process and ensuring that they only include citations that truly contribute to their work.

# 2. Background and Related Work

## 2.1 Link Prediction

Link prediction is a technique used in network analysis to predict missing links or future connections between nodes in a network. It’s based on the assumption that the probability of a link between two nodes depends on the similarity of their attributes or their connectivity patterns with other nodes in the network.

In other words, link prediction aims to identify pairs of nodes that are likely to be connected in the future or that have a high probability of being connected but are currently not connected in the network.

## 2.2 Complex network Graph

A network graph consisting of a Vertex set and an Edge set , where each vertex represents a data object, and each edge represents a relationship between the data objects. For a given vertex , represents the vertex directly connected to , the direct neighbor of .

### 2.2.1 Proximity Order

* **First-order proximity:** A type of proximity between two vertices in a network, which represents the local similarity between them. If there is an edge between two vertices u and v, then they have first-order proximity [[4](#Refernces)].
* **Second-order proximity:** A type of proximity between a pair of vertices in a network, which represents the similarity between their neighborhood network structures. If two vertices u and v have the same neighbor node, then they have second-order proximity [[4](#Refernces)].
* **Third-order proximity (high-order proximity):** A type of proximity between pairs of vertices in a network, which represents the similarity between global network structures. Taking third-order proximity as an example, vertices u and v have third-order proximity if they are connected to two vertices with second-order proximity, respectively.

## 2.3 Generative countermeasure network (GAN)

A diagram of a sample

Description automatically generated with low confidenceGAN is a machine learning model framework that consists of two neural networks, namely the **Generator** and the **Discriminator**. GANs are designed to generate realistic data samples that resemble a training dataset such that the discriminator could not determine between synthetic data and real data.

**Figure 1.** Example of GAN network structure [[7](#Refernces)].

The **Generator** is the part of the network responsible for generating synthesized data based on the real data. At each iteration, the generated synthesized data is passed to the discriminator, and based on the feedback from the discriminator, the generator makes correction to generate data as similar to the real data as possible such that the discriminator cannot distinguish between real data and synthesized data.

The **Discriminator** is the second part of the network, trained to distinguish between real data samples from the training set and the synthetic samples produced by the generator. It aims to correctly classify whether a given input is real or generated. The discriminator is trained using a combination of real and generated data samples, adjusting its parameters to improve its ability to differentiate between the two, while passing the results to the generator.

During the training process, the generator and discriminator networks are pitted against each other in a game-like manner. The generator aims to generate more realistic samples to fool the discriminator, while the discriminator aims to become better at distinguishing between real and generated data. The networks are trained iteratively, with the generator trying to minimize the discriminator's ability to differentiate, and the discriminator trying to maximize its discriminative power.

The training process of a GAN involves a back-and-forth competition between the generator and discriminator networks. As the training progresses, the generator becomes better at generating realistic samples, while the discriminator becomes more skilled at identifying generated data.

This iterative process continues until a balance is reached, ideally resulting in a generator that can produce high-quality synthetic samples that are indistinguishable from real data.

In the contexts of link prediction, a GAN is a type of neural network that can be used to predict missing links in a network.

The goal of a GAN model is to generate new links in a network that are likely to be present and are currently missing.

## 2.4 Node Similarity indexes

Indexes that Are used as a measure to determine or evaluate the similarity between nodes in a network or graph [[6](#Refernces)].

### 2.4.1 Adamic-Adar (AA):

“The number of neighbors in the complex network is called the degree of the node. The AA index gives a weight to each common neighbor of two nodes according to the degree information of the common neighbors of two nodes”. AA indicators are defined as:

where is defined as the neighbors of node , and is the degree of node .

### 2.4.2 Local-Path (LP):

The local path as the name might suggest considers the common neighbors with path lengths of 2 and 3 between two nodes to show and get an assessment on how similar two nodes are. The LP index is defined as:

* α: Adjustable parameter to control the proportion of third order paths.
* : An adjacency matrix.
* : Represents the number of paths of length n between node .

### 2.4.3 Katz index (KZ):

Katz index is based on the LP index and considers all the common neighbors and the path length between them, such that shorter path lengths are granted heavier weight then longer paths and is defined as:

And β is a “Weight attenuation factor, and the value of β is less than the reciprocal of the maximum eigenvalue of the adjacency matrix”.

## 2.5 Hierarchical Network Graph

The hierarchical processing of a network graph refers to the approach of analyzing and understanding the graph's structure and information in a hierarchical or multi-level manner. It involves breaking down the graph into different levels or layers and performing computations and analyses at each level to gain insights into the network's organization and properties.

In a hierarchical processing framework, the network graph is represented as a collection of nodes and edges, where nodes represent entities, and edges represent relationships or connections between entities. The goal is to understand the graph's characteristics and connectivity patterns by considering different levels of abstraction.

At each level of the hierarchy, various techniques and algorithms can be employed to extract meaningful information. This hierarchical analysis helps to reveal the complex organization and relationships within the network [[1](#Refernces)].

## 2.6 Hierarchical Network algorithm

The Hierarchical network algorithm includes two key Parts: Edge Folding and Vertex Merging.

The purpose of edge folding and vertex merging is to simplify the hierarchical network graph by collapsing or "folding" edges between nodes and merging vertices that share similar properties or features.

Edge Folding and Vertex Merging is used to reduce the complexity of the hierarchical network graph and to remove redundant or irrelevant information.

This method can help to improve the accuracy and efficiency of the network analysis and machine learning algorithms, by reducing the dimensionality of the data and focusing on the most relevant and informative features [[1](#Refernces)].

### 2.6.1 Edge Folding

Diagram

Description automatically generatedSay vertices and are connected and neither are in a closed loop, merging them will fold the edge and as a result the vertices and will be merged into a single vertex while preserving the first order proximity [[1](#Refernces)].

**Figure 2.** Example of Edge Folding [[1](#Refernces)].

### 2.6.2 Vertex Merging

In cases where there are a lot of vertices with common neighbors (closed loop), there is no option to merge them by edge folding.

Say vertices and had common neighbors they can be merged into using Vertex Merging method while preserving the structure and data of the original graph. [[1](#Refernces)]

A picture containing watch

Description automatically generated

**Figure 3.** Example of Vertex Folding [[1](#Refernces)].

## 2.7 Network graph layering algorithm – Netlay

The following algorithm is performed until the generated sub-network graph is smaller than a specific threshold.

A threshold represents the minimal number of vertices in a “compressed” hierarchical representation of the original graph [[1](#Refernces)].

Text

Description automatically generated

**Figure 4.** Network layering algorithm NetLay pseudo code [[1](#Refernces)].

A picture containing diagram, sketch, circle, drawing

Description automatically generated

**Figure 5.** Example of network graph layering [[1](#Refernces)].

## 2.8 Node Embedding

Node embedding is a way of representing nodes as low – dimensional vector representations. The goal of node embedding is to capture the structural and semantic information of nodes in a way that preserves their relationships and properties within the graph.

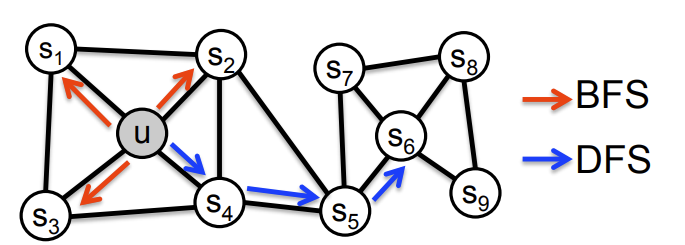
### 2.8.1 Random walk

Random walk is a mathematical model used to describe a random process that involves moving through a network or graph, where each step is taken at random based on a set of probabilities. In a random walk, an entity starts at a particular node in a network and takes a step to a neighboring node at random, with each neighboring node having a certain probability of being chosen as the next step. The process then continues, with the entity taking successive steps at random until it reaches a stopping condition.

Random walks are not completely random, the way to control the walk is by giving weight for the edges in the graph.

The weight is determined by the function where is the previously visited node, is the potential next node and is the distance (shortest path) from note to the following node .

The and parameters are used to control the probabilities of moving towards neighboring nodes during a random walk. Specifically, determines the likelihood of returning to the previous node in the walk, while determines the likelihood of exploring a new, unvisited node. When is low and is high, the random walk favors exploring nodes that are far away from the starting node, resulting in a "depth-first search" behavior. Conversely, when is high and is low, the walk favors exploring nodes that are close to the starting node, resulting in a "breadth-first search" behavior. By tuning these parameters, the algorithm can capture different types of structural information in the network [[5](#Refernces)].



**Figure 6.** Example of BFS and DFS Algorithms [[8](#Refernces)].

A picture containing circle, diagram, line, font

Description automatically generated

**Figure 7.** Settings of deviation coefficient in random walk [[5](#Refernces)].

### 2.8.2 Node2Vec Algorithm

Node2Vec is an algorithm used to create vector representations of nodes in a graph for machine learning applications. It works by using random walks through the graph starting at a target node to learn low-dimensional representations of each node. The algorithm treats the random walks like sentences in a corpus, where each node is treated as an individual word and a random walk is treated as a sentence. By feeding these "sentences" into a skip-gram or continuous bag of words model, the paths found by random walks can be treated as sentences and traditional data-mining techniques can be used. Node2vec is an improvement on prior work that was based on rigid notions of network neighborhoods and argues that the added flexibility in exploring neighborhoods is the key to learning richer representations of nodes in graphs. It is considered one of the best graph classifiers [[2](#Refernces)].

## 2.9 EmbedGAN

The EmbedGAN algorithm is a link prediction algorithm based on generative adversarial networks (GANs). It uses a generative adversarial model to obtain the low-dimensional vector form of the vertices recursively and backwardly in each layer of the network graph. The low-dimensional vector form of all the vertices in the original network graph is then used for link prediction.

The algorithm consists of two main components: the generator and the discriminator. The generator is responsible for generating low-dimensional vector representations of the vertices in the network graph. The discriminator is responsible for distinguishing between real and fake low-dimensional vector representations.

### 2.9.1 Discriminator optimization

Discriminator is defined as the sigmoid function of the inner product represented by the low-dimensional vector of two input vertices using the following equation [[1](#Refernces)]:

where and are low-dimensional vector representations corresponding to vertices v and .

### 2.9.2 Builder Sampling Strategy

**”**The generator uses a biased random walk with step size l to sample the negative samples. If the source node is , the current node is , and the previous hop node is , then the next hop node needs to be determined. Define as the set of direct neighbors of vertex (that is all vertices directly connected with in the graph), and the transition probability between and its neighbor is defined as” [[1](#Refernces)]:

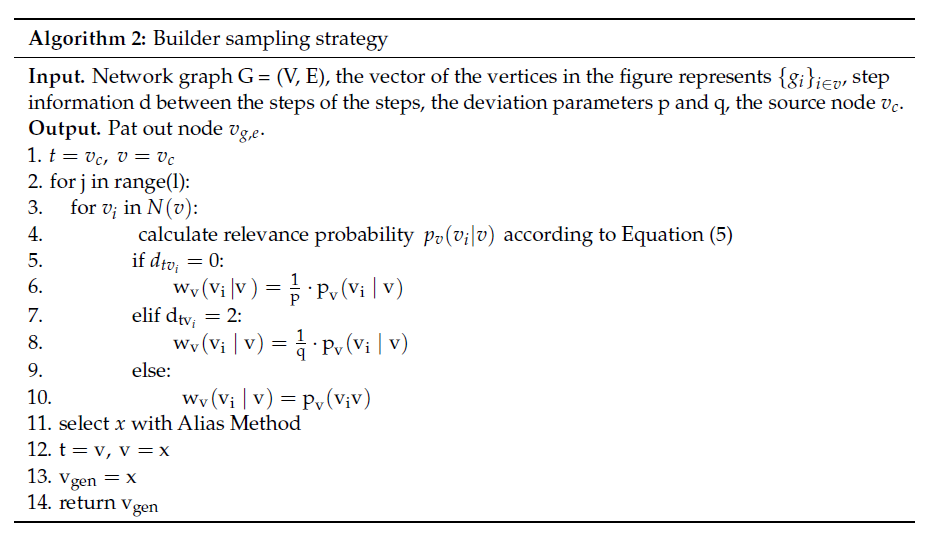
where and are k-dimensional vector representations of vertices and relative to the generator.

At each step, the transition probability between the current node and its neighbor nodes is calculated according to the formula. Then the transition probability is weighted by the deviation coefficient of random walk, and the non-standard transition probability

is obtained.

According to the non-standard transition probability, the next hop node of the random walk is extracted using the alias method [[3](#Refernces)]: is set as shown in [[2.9](#_2.9_EmbedGAN)].

When the non-standard metastatic probability is obtained, one node is selected as the next hop node x using the alias method. When the step of the swing reaches the length of the set, the current node v is extracted as a negative sample vertex”.



**Figure 8.** Builder sampling strategy algorithm pseudo code [[1](#Refernces)].

“Path from source node to extracted vertex is

where and , then connectivity is defined as” [[1](#Refernces)]:

### 2.9.3 EmbedGAN Algorithm

EmbedGAN is an algorithm that utilizes a framework consisting of a generator and a discriminator. The generator creates negative samples for each source node and extracts positive samples, which are then used by the discriminator for training. The discriminator assigns labels to these samples and updates its vector representation to minimize errors. The generator also updates its vector representation based on the feedback from the discriminator. This process continues until the discriminator cannot differentiate between positive and negative samples, resulting in the final low-dimensional vector representation of the graph's vertices.

A picture containing text, diagram, screenshot, line

Description automatically generated

**Figure 9.** EmbedGAN algorithm framework [[1](#Refernces)].

A screenshot of a graph

Description automatically generated with low confidence

**Figure 10.** EmbedGAN algorithm pseudo code [[1](#Refernces)].

## 2.10 GAHNRL Algorithm

* Using the network graph layering algorithm explained in 2.7 to create a subnetwork of n graphs.
* Process graph using Node2Vec Algorithm explained in 2.8.2 to generate the initial vector representation of the vertices of the graph .
* The initial vector representations of each layer in the subnetwork graph are recursively fed into the generative countermeasure network for training, starting from graph .
* The algorithm employs the EmbedGAN algorithm explained in 2.9 to learn low-dimensional vector representations of vertices in . These learned representations serve as the initial vector representations for the upper layer subnetwork graph, .

The process recursively continues through backtracking learning until reaching the initial network graph, . Ultimately, the algorithm achieves the low-dimensional vector representation, , for all vertices.

* According to the similarity between vertices calculated by , the algorithm predicts whether there is a link between two vertices [[1](#Refernces)].

A picture containing text, screenshot, font, number

Description automatically generated

**Figure 11.** GANHRL algorithm pseudo code [[1](#Refernces)].

# 3. Expected Achievements

In this project, the expected goal is to achieve several outcomes that will contribute to addressing the issue of citing irrelevant articles and improving the quality of scholarly literature. The key achievements/products that will be developed include:

## 3.1 Algorithm Development

Development of such an Algorithm based on the insights from the article:

"A Link Prediction Algorithm Based on GAN" [[1](#Refernces)].

The mentioned algorithm is designed to analyze the citation graph of scholarly articles and predict the relevance of citations. It will provide researchers with a tool to assess the importance of citations before including them in their articles.

## 3.2 Success Criteria

The success of the project will be measured based on the accuracy and efficiency of the developed algorithm in identifying irrelevant citations. The criterion for success includes high prediction accuracy in distinguishing relevant and irrelevant citations, reduced time and effort required for researchers to evaluate citations, and improved overall quality of scholarly literature.

## 3.3 Unique features and Research Challenges

This project involves several unique features and research challenges. Firstly, adapting the GAN-based link prediction algorithm for citation relevance assessment is not commonly in use, therefore it requires addressing challenges such as handling the complexity of the citation graph, developing efficient prediction models, and ensuring scalability to handle large datasets of scholarly articles.

To overcome these challenges, the following methods are used:

* Leveraging real-world datasets.
* Feedback from our supervisors Dr. Renata Avros and Prof. Zeev (Vladimir) Volkovich, will be incorporated to refine the algorithm.
* Extensive research on the topic and utilizing the latest solutions to issues the algorithm might face.

In conclusion by achieving these expected outcomes, this project aims to provide researchers with a valuable tool to combat the problem of citing irrelevant articles. The algorithm aim is to make the citation process simpler, improve the accuracy of citation evaluation, and ultimately enhance the quality and relevance of scientific literature.

# 4. Engineering Process

## 4.1 Problem Identification

The proposed project revolves around the citation issues in articles, in every article we can find countless citations to different sources, but are they all relevant?

are all of them relevant to the paper or could some be irrelevant and could be removed without harming or altering the original paper? To answer this question, we must recognize the importance of removing citations from articles without adversely affecting their content, what will be the benefits of such a system, how can we use the suggested algorithm in the article to achieve this goal?

With the aid of such a system, one can assist in addressing the forthcoming issues related to citations in articles.

* Plagiarism
* Inaccurate or Missing Citations
* Inadequate or Insufficient Citations
* Misleading or Confusing References
* Misrepresentation of information
* Citing unrelated or irrelevant articles to promote specific papers or authors.
* Misallocation of Resources

Our project focuses on the issue - “Citing unrelated or irrelevant articles”.

## 4.2 Literature review

In the literature review phase, our objective is to examine relevant literature on a Link Prediction Algorithms based on GAN, specifically focusing on its application in identifying removable citations from given articles without compromising the integrity of the original paper. This comprehensive review will allow us to gain a deeper understanding of the existing methods and techniques employed in link prediction algorithms based on GAN. By exploring the state-of-the-art in this field, we can identify gaps and limitations in the literature, ultimately guiding our research in developing an approach to determine which citations can be safely removed from articles while preserving the original structure, idea, and integrity of any given article.

## 4.3 Data Collection

The data collection stage for our algorithm involves a systematic approach to gather the necessary data. Firstly, a suitable dataset comprising a diverse range of scholarly articles is identified, serving as the foundation for our analysis. Next, the relevant information from the articles, including citation networks, textual content, and metadata, is extracted. To ensure the data's quality and reliability, rigorous preprocessing techniques, such as data cleaning, standardization, and removal of duplicate or irrelevant entries, are employed. Additionally, established academic databases, digital libraries, and relevant online repositories are explored to supplement the dataset with external sources if necessary. The collected data is appropriately organized and structured for further analysis and model training.

### 4.3.1 Data Pre-Processing

The next step is to get it ready for analysis. Referred to as data preprocessing.

This stage involves cleaning up the data to make sure it's in good shape, Thorough examination of the data is undertaken to rectify any errors or missing information diligently. For example, if there are some numbers or words that don't look right, necessary corrections are made to address such issues.

To ensure data consistency, efforts are made to harmonize all elements into a unified format. Sometimes, the data can be a bit messy or complicated, to enhance comprehension, certain parts of the data may be simplified or combined as deemed necessary. Our goal is to end up with a neat and organized dataset that we can use for analyzing the data and building the algorithm.

### 4.3.2 Graph data representation

To represent the data in a graph format for the algorithm, a network or graph structure is utilized for our purposes. In this representation, each scholarly article is represented as a node, and the citations between articles are depicted as edges connecting these nodes. This creates a network of interconnected nodes, forming a visual graph representation.

To construct the graph, unique identifiers or labels are assigned to each article, and corresponding nodes are created in the graph to represent them. The citations between articles are modeled as directed edges in the graph, indicating the direction of the citation relationship. For example, if Article A cites Article B, there will be an edge pointing from the node representing Article A to the node representing Article B.

Additionally, additional attributes or properties of the articles can be incorporated as node or edge attributes in the graph. These attributes could include information such as the publication year, authors, keywords, or other relevant metadata associated with each article. By including such attributes, the graph representation is enriched, providing additional context for the link prediction algorithm to effectively utilize.

### 4.3.3 GAN model training and evaluation

The next step is to train and evaluate the model we're building. First, the model is taught using the preprocessed data that was processed earlier. The model is presented with numerous examples from the data, allowing it to learn patterns and make predictions. Subsequently, the model's performance is evaluated through testing. New examples are provided, and the model's ability to accurately predict missing or removable citations is assessed. By comparing the model's predictions with the known correct answers, its effectiveness is measured. If the model's performance is unsatisfactory, adjustments are made, and further iterations are conducted until a satisfactory level of performance is achieved. Our goal is to create a model that can effectively predict which citations can be removed without affecting the original paper's integrity.

Finally, in our current research, the study primarily focuses on articles related to Generative Adversarial Networks (GANs) and link prediction using GAN networks. Additionally, the exploration of citations addresses the issue of citing unrelated or irrelevant articles. To tackle this problem, a project plan is devised with several key steps. Currently, the literature review examines existing methods and techniques in the field of link prediction algorithms based on GANs. Following this, relevant data is collected and preprocessed to ensure its quality and suitability for analysis. Subsequently, a model is trained and evaluated to predict removable citations while preserving the integrity of the original paper. Ongoing efforts involve refining the model to achieve a satisfactory level of performance.

Overall, our research aims to contribute to the development of an improved link prediction algorithm that effectively addresses the problem of citing irrelevant articles.

## 4.4 Data collection and model training flow chart

The following diagram (Fig. 12) represents the process of collecting data and finding a suitable GAN model that fits our specific need for link predictions using GAN. The data is presented as a graph representation of articles or papers, aiming to obtain correct and desirable results from the network. In cases where a suitable model or network is not readily available, modifications are made to a pre-trained network to align it with our requirements. The model is adjusted to ensure its performance meets the expectations, ultimately achieving the goal of "Citation predictions using GAN."

A picture containing text, screenshot, diagram, design

Description automatically generated**Figure 12.** Workflow diagram of Product process.

## 4.5 Algorithm pseudo code

**Input** The network Graph , the step size the deviation parameters and the distance information between vertices

**Output** the low-dimensional vector of the vertices in the network graph ­­ represents .



A picture containing text, diagram, plan, technical drawing

Description automatically generated

**Figure 13.** Framework of the proposed algorithm between 5-12 stages [[1](#Refernces)].

# 5. Evaluation / Verification Plan

## 5.1 Testing Approach

The selected approach for testing our algorithm is the following:

Divide the Dataset into two parts, one part will be used for training while the other part will be used for testing the correctness of the algorithm. As we mentioned in section [[4.3.3](#_4.3.3_GAN_model)] the training is conducted on roughly (70-80) % of the Dataset while testing will be performed on the remaining (20-30) %, while ensuring that both sets have same ratio of both relevant and irrelevant citations.

In addition to the mentioned above and referred to as “Simple Split” another technique to be implemented is “Cross-Validation”, this technique involves dividing the Dataset into multiple subsets or “folds”. The Model is trained and evaluated multiple times, each time using a different fold as the test set and the remaining folds as the training set. This approach provides a more robust evaluation of the model’s performance.

The Metrics are as follows:

* **Accuracy:** Accuracy measures the proportion of correctly predicted links (both positive and negative) out of all the links in the testing set. It provides an overall measure of how well the model predicts the presence or absence of the links.
* **Precision**: Precision focuses on the proportion of true positive predictions (relevant citations correctly identified) out of all the positive predictions. It measures the precision of the model in the identifying irrelevant citations correctly.
* **Recall (Sensitivity):** Recall calculates the proportion of true positive predictions out of all actual positive instances (relevant citations). It captures the ability of the model to identify relevant citations in the testing set.

## 5.2 Evaluation

The metrics mentioned above will be assessed on a numeric value, for each of them the following will be considered a “success” or a “inefficient.”

* **Accuracy**: any numeric value above 80% will defined as success, and while anything below will be considered inefficient.
* **Precision and Recall:** This metrics are interrelated, meaning that they are both dependent on the tradeoff between them, thus a numeric value between 70-80% is considered a success.

## 5.3 Testing / Evaluation Process

1. **Data Preparation**: Process the dataset by dividing it into a training set and a testing set. The training set will be used in order to train the model, while the testing set will be used to evaluate the model’s performance, while ensuring that the testing set contains a representative sample of both relevant and irrelevant citations.
2. **Model Training**: Train the link prediction model using the training set. This involves feeding the model with the relevant features and labels (presence or absence of a relevant and irrelevant citation) for each instance. The model learns the patterns and relationships between citations to make predictions.
3. **Prediction Generation**: Apply the trained model to the testing set to generate predictions for the presence or absence of a relevant and irrelevant citation. The model will output a probability or a binary prediction for each citation instance in the testing set.
4. **Evaluation Metrics** Calculation: Calculate the evaluation metrics mentioned earlier (accuracy, precision, recall) by comparing the predicted values with the actual values in the testing set. For each metric, the corresponding formulas are used to generate a numeric value in order to evaluate the outcome.
5. **Interpretation and Analysis**: Analyze the evaluation metrics to gain insights into the model's performance. Compare the metrics to assess the accuracy and effectiveness of the link prediction model in identifying irrelevant and relevant citations.
6. **Fine-tuning and Iteration**: Based on the analysis of the evaluation metrics, iterate, and fine-tune the model if necessary. This may involve adjusting model parameters, feature selection, gather additional datasets. Repeat the training and testing process to observe the impact of the modifications on the model's performance.
7. **Reporting and Conclusion**: Summarize the results of the testing process in a comprehensive report. Present the evaluation metrics, discuss the findings, and draw conclusions about the model's ability to predict irrelevant and relevant citations in academic articles.

# 6. References

[1]. Haiyan Jin, Guodong Xu \*, Kangda Cheng, Jinlong Liu and Zhilu Wu ,

- A Link Prediction Algorithm Based on GAN , Harbin Institute of Technology, Harbin 150001 - MDPI electronics Journal. [A Link Prediction Algorithm Based on GAN](https://www.mdpi.com/2079-9292/11/13/2059)

[2]. Grover, A.; Leskovec, J. Node2vec: Scalable feature learning for networks. In Proceedings of the KDD’ 16, San Francisco, CA,

USA, 13–17 August 2016; ACM Press: New York, NY, USA, 2016; pp. 1–10.

[Node2Vec: Scalable Feature Learning for Networks](https://arxiv.org/abs/1607.00653)

[3].Kronmal, R.A.; Peterson, A.V., Jr. On the alias method for generating random variables from a discrete distribution. Am. Stat. **1979**, 33, 214–218.

[4]. Zhang, T.; Li, H.; Hong, W.; Yuan, X.; Wei, X. Deep first formal concept search. Sci. World J. **2014**, 2014, 275679.

[Deep First Formal Concept Search](https://www.hindawi.com/journals/tswj/2014/275679/) , [Deep first formal concept search](https://pubmed.ncbi.nlm.nih.gov/25254232/)

[5]. Wang, H.; Wang, J.; Wang, J.; Zhao, M.; Zhang, W.; Zhang, F.; Li, W.; Xie, X.; Guo, M. GraphGAN: Graph representation learning

with generative adversarial nets. IEEE Trans. Knowl. Data Eng. **2018**, 33, 3090–3103.

[Learning Graph Representation With Generative Adversarial Nets](https://ieeexplore.ieee.org/document/8941296)

[6]. Lü, L.; Zhou, T. Link prediction in complex networks: A survey. Phys. A Stat. Mech. Its Appl. **2010**, 390, 1150–1170. [Link Prediction in Complex Networks: A Survey](https://arxiv.org/abs/1010.0725)

[7]. Machine learning Overview of GAN structure [developers.google.com](https://developers.google.com/machine-learning/gan/gan_structure)

[8]. Overview of deep learning on graph embeddings [Towards data science](https://towardsdatascience.com/overview-of-deep-learning-on-graph-embeddings-4305c10ad4a4)