NODE EMBEDDINGS AND CLASSIFICATION FOR TURKISH SCIENTIFIC LITERATURE: A GRAPH-BASED APPROACH

Elif Özyürek **Hakan Çoban**

ABSTRACT

The growth of scientific publications poses challenges in organizing and understanding domain-specific research, especially when language translation and terminology consistency are involved. This study introduces a graph-based framework tailored to Turkish scientific literature, with a focus on capturing semantic relationships and conceptual coherence across translated texts. By constructing a homogeneous graph of papers and applying Node2Vec, SentenceTransformer, BERTopic and Graph Neural Networks (GNNs), we aimed to generate meaningful node embeddings that facilitate effective node classification and clustering. This work intends to support the discovery, comparison, and categorization of research in linguistically diverse academic ecosystems.

1 INTRODUCTION

The rapid publication of thousands of studies has made it increasingly difficult to track new research, particularly studies translated from English to Turkish, especially when it comes to conceptual terminology. This challenge is particularly prominent in specialized fields, where the accurate translation of technical terms is essential.

Graph-based representations are useful for modeling relationships in unstructured data. In the context of academic literature, papers can be represented as nodes, with edges defined by shared authors, overlapping keywords, or semantic similarity. By utilizing graph learning methods like Node2Vec for generating node embeddings and Graph Neural Networks (GNNs) for classification, hidden patterns within the graph structure can be discovered and effectively used for various applications.

However, constructing meaningful graphs over noisy or semi-structured textual datasets poses several challenges. The effectiveness of models like Node2Vec depends heavily on the graph quality. Additionally, unsupervised topic modeling techniques such as BERTopic and semantic embedding models like Sentence Transformers offer alternative ways to enrich graph construction or generate embeddings for classification and retrieval.

In this study, we propose a framework for constructing, analyzing, and comparing multiple embedding and classification strategies over Turkish scientific literature. The structure of the paper is as follows: Section 2 reviews related work, Section 3 introduces the framework and node classification tasks, Section 4 outlines the experimental setup and discusses results, and Section 5 and 6 concludes with suggestions for future research.

2 RELATED WORK

Several key datasets and frameworks support this area of research:

- **ArXiv Dataset:** The ArXiv dataset [3] is a freely available dataset that contains millions of pre-print articles from the last three decades. The subjects of the articles in the dataset range from physics to computer science. Having a natural graph structure makes the dataset an eligible candidate for machine learning models.
- **Topic-Based Specialization:** BERTopic [7] is a modern modeling technique that uses transformer based embeddings and density-based clusterings to discover coherent topics

in large text-based datasets. Unlike traditional models which rely on bag-of-words representations, BERTopic uses contextual embeddings generated by pre-trained models to capture deeper semantic relations.

3 Methodology

3.1 Dataset Generation

3.1.1 Graph Creation for Turkish Index Journals

We have constructed a graph that models relationships between papers, authors, and topics in Turkish indexed journals from Journal of Polytechnic [6]. We have used three different approaches for modeling our dataset as a graph. The common step for each of the models was to look for the Jaccard [5] similarity score for the keywords and/or the number of common authors of the papers. If the similarity score or the common author count exceeded the threshold value, which were 0.3 and 1 respectively, those nodes were connected. The first model utilizes Node2Vec[8] to generate embeddings, then performs similarity search and node classification. We refer this simple version as Node2Vec, which is discussed in section 3.2.1. In the second model, we use SentenceTransformer [16] to generate embeddings from the abstract and title attributes of each node, further discussed in section 3.2.2. Lastly, for the third model, we use BERTopic 3.2.3 to add a 'topic' attribute for each node.

3.1.2 Graph Visualization

We use NetworkX [9], matplotlib [1] and pyvis [15] to visualize graph structures, node connectivity, clusters, and topic distribution across the dataset. Initially, each model is visualized with the functionalities of Networkx to show how each node is distributed 1. For models that utilize the K-Means [10] algorithm, we also provide related elbow method and silhouette score 7 plots. For the models that use K-Means or BERTopic, each assigned cluster or topic is also plotted for more information 8. To deepen our analysis on the generated graph to have more insight into the assignment of groups or topics, we have used pyvis to generate interactive graphs, where we can look for every node and have the option of further comparison 15.

3.2 Models

3.2.1 Node2Vec Model

Node2Vec is an algorithm for learning embeddings of the nodes in a graph. These embeddings can capture the graph structure so that nodes with similar roles or connections can be situated closer in the embedding space. The algorithm is inspired by Word2Vec [14] which learns embeddings for words based on their surrounding context in sentences, so it generates random walks on the graph, using the edges as if they were words.

Node2Vec simulates multiple biased random walks from each node. To control these random walks, the algorithm introduces two parameters named "p" and "q", which are return and in-out parameters respectively. These parameters determine the likelihood of the algorithm exploring new nodes or revisiting previous ones, effectively controlling the breadth-first (BFS) or depth-first (DFS) nature of the random walk.

Once biased random walks are generated from the graph, each walk is treated analogously to a sentence in natural language. Node2Vec applies the Skip-Gram model, originally proposed in Word2Vec, to learn embeddings by maximizing the likelihood of observing a node's neighborhood within a certain window size in the walk.

Given a walk $W = (v_1, v_2, \ldots, v_l)$, the Skip-Gram model seeks to maximize the following objective:

$$\max_{\Phi} \sum_{v \in V} \sum_{u \in \mathcal{N}_w(v)} \log \Pr(u \mid \Phi(v)) \tag{1}$$

Here:

- $\Phi : V \to \mathbb{R}^d$ is the embedding function mapping nodes to d-dimensional vectors.
- $\mathcal{N}_w(v)$ is the set of context nodes within window size w around node v in the generated walks.
- $\Pr(u \mid \Phi(v))$ is modeled using the softmax function:

$$\Pr(u \mid \Phi(v)) = \frac{\exp(\Phi(u)^\top \Phi(v))}{\sum_{n \in V} \exp(\Phi(n)^\top \Phi(v))} \tag{2}$$

Due to the computational cost of the full softmax, Node2Vec employs negative sampling as an approximation, which optimizes a simplified objective:

$$\log \sigma(\Phi(u)^\top \Phi(v)) + \sum_{i=1}^{k} \mathbb{E}_{u_i \sim P_n(u)} [\log \sigma(-\Phi(u_i)^\top \Phi(v))] \tag{3}$$

Where:

- $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function,
- $P_n(u)$ is a noise distribution over nodes,
- k is the number of negative samples per positive pair.

3.2.2 Sentence Transformer Model

SentenceTransformer is a Python framework built on top of pretrained Transformer models such as BERT [4], RoBERTa [11], or DistilBERT [17]. Its primary goal is to generate semantically meaningful sentence embeddings that capture the contextual meaning of entire sentences or documents, rather than just individual words.

Unlike vanilla BERT models, which are optimized for token-level classification tasks and produce contextualized word embeddings, SentenceTransformer modifies the architecture to support sentence-level embedding generation. It applies a pooling strategy (such as mean pooling) over the output token embeddings to obtain a fixed size vector representation for a given sentence.

Formally, given an input sentence $s = (w_1, w_2, \dots, w_n)$, the underlying Transformer model produces token embeddings $(\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_n)$. A pooling operation is then applied to produce a sentence embedding $\mathbf{s} \in \mathbb{R}^d$:

$$\mathbf{s} = \text{Pooling}(\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_n) \tag{4}$$

The resulting vector $\mathbf{s}$ captures the semantic content of the sentence and can be used for a wide variety of downstream tasks, including semantic similarity, clustering, retrieval, and classification.

Since our main objective was to utilize Node2Vec algorithm to its full potential, we have proposed a second model to increase the structural depth of the graph. By using SentenceTransformer, we have obtained two embeddings: title and abstract. With the help of these embeddings, we iterated over the graph an additional time. For each node, we looked at embedding similarities with other nodes and add a new link between these nodes if one of the similarity scores exceeds our threshold, which was 0.8 and 0.7 respectively.

Previously before the SentenceTransformer, our graph had 4073 edges. By making new connections with the help of title and abstract embeddings, we have increased this number to 4843. In graph's final version, the number of connected nodes were 1441, whereas the number of isolated nodes were 427.

3.2.3 BERTopic Model

BERTopic is a topic modeling technique that leverages transformer based embeddings to generate interpretable and semantically meaningful topics from textual data. Unlike classical topic modeling approaches such as Latent Dirichlet Allocation (LDA) [2], which rely on word co-occurrence frequencies, BERTopic incorporates contextual language understanding using modern language models such as BERT.

The topic modeling pipeline of BERTopic consists of three main components:

1. **Document Embedding:** Each document d_i is transformed into a dense vector $\mathbf{e}_i \in \mathbb{R}^d$ using a SentenceTransformer model. These embeddings capture the semantic meaning of the documents.
2. **Dimensionality Reduction:** Since transformer based embeddings are typically high-dimensional, BERTopic applies a dimensionality reduction technique such as UMAP (Uniform Manifold Approximation and Projection) [13] to project the embeddings to a lower-dimensional space:
$$\mathbf{z}_i = \text{UMAP}(\mathbf{e}_i), \quad \mathbf{z}_i \in \mathbb{R}^{d'}, \quad d' \ll d \tag{5}$$
3. **Clustering and Topic Representation:** The reduced embeddings $\mathbf{z}_i$ are then clustered using a density-based clustering algorithm, commonly HDBSCAN [12]. Each cluster corresponds to a potential topic. For each cluster, representative words are extracted based on term frequency–inverse document frequency scores over the documents in that cluster.

For our final model, this time we did not rely on the Node2Vec embeddings. For a deeper analysis, we combined title and abstract attributes of the nodes and by utilizing the functionality of BERTopic to it's fullest, we created a topic model. We have assigned topic number to each node as an attribute and removed any node that could not be determined by the topic model's capability. Our final graph after truncating it, shrunk to 957 connected nodes and 469 isolated nodes.

3.3 Similarity Search

After obtaining low-dimensional node embeddings via Node2Vec or BERTopic, we can perform similarity-based retrieval tasks such as identifying related papers or authors. Each node in the graph is embedded into a vector space. To find nodes that are semantically or structurally similar to a given query node, we compute the cosine similarity between its embedding and all other node embeddings.

Given a query node with embedding $\mathbf{q} \in \mathbb{R}^d$, and a set of node embeddings $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$, the cosine similarity between $\mathbf{q}$ and a node embedding $\mathbf{v}_i$ is defined as:

$$\text{sim}(\mathbf{q}, \mathbf{v}_i) = \frac{\mathbf{q} \cdot \mathbf{v}_i}{\|\mathbf{q}\| \|\mathbf{v}_i\|} \tag{6}$$

Where:

- $\mathbf{q} \cdot \mathbf{v}_i$ is the dot product between the two vectors.
- $\|\mathbf{q}\|$ and $\|\mathbf{v}_i\|$ denote the Euclidean norms.

The similarity scores are computed for all nodes in the graph and ranked in descending order. The top-k nodes with the highest cosine similarity values are selected as the most similar entities to the query node.

Practically, we iterate over the learned Node2Vec embedding matrix and compute similarity scores using the cosine similarity metric. We also retrieve the associated metadata—such as title and abstract of the top-ranked nodes for further inspection and evaluation.

3.4 Node Classification

3.4.1 Graph Neural Network (GNN) Application

For node classification, we applied multiple GNN architectures: Graph Convolutional Network (GCN), Graph Attention Network (GAT), GraphSAGE, and Graph Isomorphism Network (GIN).

3.4.2 GNN Methodology

For the embeddings obtained by the BERTopic model, the topic cluster labels were used as ground truth. For the embeddings from Node2Vec and SentenceTransformer approaches, K-Means cluster labels were used as ground truth. Each model was trained on node embeddings derived from the early stages of our different approaches and evaluated for their effectiveness in learning meaningful node representations.

Each of the architectures used in our study has distinct operational mechanisms, making them suitable for capturing various structural and semantic patterns in the data. All models were trained using the Adam optimizer with Cross-Entropy loss, and the final hidden representations were passed through a Softmax layer for multi-class node classification. We experimented with different values for dropout and learning rate, and the best performance across models was achieved using a dropout rate of 0.5 and a learning rate of 0.01.

1. **Graph Convolutional Network (GCN):** GCN is a foundational GNN architecture that performs neighborhood aggregation by averaging the features of a node's neighbors, normalized by degree. In our implementation, we used two GCNConv layers, separated by ReLU activations and dropout for regularization.

2. **Graph Attention Network (GAT):** GAT introduces an attention mechanism to GNNs, allowing nodes to weigh the importance of their neighbors during feature aggregation. Our implementation has two GATConv layers, first layer having 8-head multi-head attention and second layer having a single-head output. Similar to the GCN, we used ReLU activation and dropout between the GATConv layers.

3. **GraphSAGE:** GraphSAGE (SAmple and aggreGatE) generalizes the GCN architecture by allowing flexible aggregation functions such as LSTM, mean, and max pooling. In our model, we used two SAGEConv layers with ReLU activation and dropout. GraphSage's ability to learn from both the node's own features and the aggregated information from neighbors makes it particularly effective for heterogeneous graphs with diverse node attributes. Although our graph is homogeneous, the diversity in node attributes makes GraphSAGE a suitable architecture, which consistently yielded strong performance across our experiments.

4. **Graph Isomorphism Network (GIN):** GIN is an architecture designed to have as much power as the Weisfeiler-Lehman graph isomorphism test. This architecture uses Multi Layer Perceptrons (MLPs) in its GINConv layers and it is strong in capturing subtle structural differences in node-level predictions. In our implementation, we used two GINConv layers with a ReLU activation between them.

4 Evaluation

4.1 Evaluation Metrics

As outlined in the previous sections, our study follows three different approaches to construct a graph based paper similarity and classification pipeline. To evaluate the performances of the selected Graph Neural Network (GNN) architectures, we utilize standard classification metrics such as precision, recall, F1 score, and test accuracy. These metrics allow us to assess the performance of each model, based on the model's ability to correctly classify relevant nodes in the test set.

To determine the optimal number of clusters for unsupervised algorithms, we utilize elbow method and silhouette scores, which measure the cohesion of nodes within clusters. For the SentenceTransformer model, the optimal number of clusters is 20, with a silhouette score of 0.2583. For the Node2Vec model, the optimal number of clusters is 23, with a silhouette score of 0.4009.

And finally, for similarity-based tasks, we use cosine similarity scores to identify the most similar nodes in the embedding space.

4.2 Results

4.2.1 Node Similarity Evaluation

In our study, we performed a similarity search based on node embeddings, where the top-5 most similar papers were retrieved in response to a query paper. As part of the evaluation, we compared the cosine similarity scores of the returned nodes and also manually inspected the results for semantic relevance.

Tables 1-3 4.2.1 present the top-5 similarity search results for a given paper, with each table corresponding to a different embedding approach. While all embedding methods retrieved the same top paper, the similarity scores varied across methods. This variation is due to the different geometric properties of the embedding spaces generated by each model. Since each embedding method captures semantic and structural information differently, the resulting similarity scores reflect those distinctions.

Table 1: Node Similarity Results for Paper ID 1327987 in Node2Vec Model

Ranking	Most Similar Paper ID	Similarity Score
1	810896	0.9997
2	1327964	0.9996
3	386970	0.6870
4	672077	0.6854
5	810768	0.6730

Table 2: Node Similarity Results for Paper ID 1327987 in SentenceTransformer Model

Ranking	Most Similar Paper ID	Similarity Score
1	810896	0.7047
2	1327964	0.6276
3	367170	0.4977
4	700947	0.4877
5	367930	0.4869

Table 3: Node Similarity Results for Paper ID 1327987 in BERTopic Model

Ranking	Most Similar Paper ID	Similarity Score
1	810896	0.6347
2	1327964	0.6089
3	435581	0.5209
4	477311	0.4777
5	443219	0.4573

4.2.2 Node Classification Evaluation

Tables 4-6 4.2.2 depict the node classification performance of different GNN architectures across the three embedding approaches. After evaluating the metrics, we observe that **GraphSAGE with SentenceTransformer embeddings** achieves the highest F1-score of 0.9842 and test accuracy of 0.9875. This outcome demonstrates a strong balance between semantic expressiveness and structural awareness. This approach introduces a hybrid embedding, using Sentence Transformers to embed the semantic information contained in the titles and abstracts of the papers while preserving the topological features.

Node2Vec embeddings also perform well, with GCN reaching an F1-score of 0.9752 and the highest test accuracy of 0.9875. However, Node2Vec only focuses on the graph topology and lacks semantic understanding. Although high scores are reached, this approach might be missing the contextual relationships within the nodes.

In contrast, BERTopic based models yield significantly lower performance with F1-scores dropping as low as 0.4765 with GIN, and test accuracies fluctuating between 0.57–0.79. While BERTopic captures high-level topic semantics, it likely loses relational patterns, making it less suitable for GNNs that rely on discriminative node features.

Table 4: Node Classification Results for Node2Vec Model

GNN Model	Precision	Recall	F1-Score	Test Acc
GCN	**0.9825**	**0.9730**	**0.9752**	**0.9875**
GAT	0.9799	0.9714	0.9728	0.9857
GraphSAGE	0.9817	0.9691	0.9730	0.9857
GIN	0.9928	0.9492	0.9655	0.9768

Table 5: Node Classification Results for SentenceTransformer Model

GNN Model	Precision	Recall	F1-Score	Test Acc
GCN	0.9876	0.9794	0.9828	**0.9875**
GAT	0.9870	0.9807	0.9832	**0.9875**
GraphSAGE	**0.9886**	**0.9812**	**0.9842**	**0.9875**
GIN	0.9744	0.9675	0.9704	0.9804

Table 6: Node Classification Results for BERTopic Model

GNN Model	Precision	Recall	F1-Score	Test Acc
GCN	0.6573	0.5925	0.5932	0.6963
GAT	0.6348	0.5381	0.5454	0.6495
GraphSAGE	**0.7531**	**0.6943**	**0.7056**	**0.7967**
GIN	0.5615	0.4757	0.4765	0.5701

5 Conclusion

This paper presents a framework for node embedding and its applications in node classification tasks. It was observed that in a dataset considered relatively sufficient in size, connecting nodes based on keyword and author overlap failed to capture deeper relationships between nodes. Consequently, the Node2Vec embeddings generated from such a graph structure were found to be semantically insufficient. The connections created in this manner not only deviated from the relationships that should exist in reality, but also resulted in unreliable similarity scores when evaluated through the embeddings (Table 1 in 4.2.1).

To address this limitation, the addition of the SentenceTransformer method yielded more meaningful results according to the analysis. However, similar to the issues encountered with the BERTopic method, the dataset contains noise in the form of spelling errors, punctuation mistakes, and inconsistent abbreviations. These factors make it difficult to definitively conclude that the lower similarity scores accurately reflect semantic similarity (table 2 in 4.2.1).

On the other hand, for GNN models, using clusters obtained from simple KMeans applications of Node2Vec or SentenceTransformer embeddings as pseudo ground truth introduces a form of data leakage, making the resulting classification scores unreliable (table 4 and 5 in 4.2.2). However, since the dataset lacks true ground truth labels, the reliability of results from more complex clustering methods like BERTopic also remains open to debate (table 6 in 4.2.2).

6 Future work

- **Improved Graph Construction:** Future work can explore more semantically rich edge creation strategies, such as leveraging citation networks, co-occurrence of specific domain terms.
- **Dataset Cleaning and Normalization:** Enhancing data quality by correcting spelling errors, unifying abbreviations, and improving sentence structure could significantly improve the quality of embeddings and the accuracy of clustering and classification tasks.
- **Human-Annotated Ground Truth:** Constructing or incorporating a manually labeled subset of the dataset would allow for more reliable evaluation of node classification and clustering performance, especially for benchmarking GNNs and unsupervised methods.
- **Evaluation of Domain-Specific Models:** Utilizing or fine-tuning transformer models pretrained on scientific or Turkish corpora may improve embedding quality and downstream task performance.
- **Dataset Expansion:** Extend the classification tasks to broader and more diverse datasets to evaluate generalizability.
- **Link Prediction:** Explore the potential of using node embeddings for link prediction tasks to uncover hidden or missing relationships.
- **RAG with Knowledge Graphs:** Investigate the integration of Retrieval-Augmented Generation (RAG) frameworks with knowledge graphs to support explainable and context-aware document retrieval or generation.

References

[1] Paul Barrett et al. *matplotlib – A Portable Python Plotting Package*. Dec. 2005.

[2] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation". In: *J. Mach. Learn. Res.* 3.null (Mar. 2003), pp. 993–1022. ISSN: 1532-4435.

[3] Colin B. Clement et al. "On the Use of ArXiv as a Dataset". In: (2019). arXiv: 1905.00075 [cs.IR]. URL: https://arxiv.org/abs/1905.00075.

[4] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[5] Luciano da F. Costa. *Further Generalizations of the Jaccard Index*. 2021. arXiv: 2110.09619 [cs.LG]. URL: https://arxiv.org/abs/2110.09619.

[6] Gazi Üniversitesi. *Politeknik Dergisi [Journal of Polytechnic]*. https://dergipark.org.tr/en/pub/politeknik. Published quarterly by Gazi University, Ankara, Turkey. Multilingual journal indexed in Web of Science, Crossref, and others. 1998.

[7] Maarten Grootendorst. *BERTopic: Neural topic modeling with a class-based TF-IDF procedure*. 2022. arXiv: 2203.05794 [cs.CL]. URL: https://arxiv.org/abs/2203.05794.

[8] Aditya Grover and Jure Leskovec. *node2vec: Scalable Feature Learning for Networks*. 2016. arXiv: 1607.00653 [cs.SI]. URL: https://arxiv.org/abs/1607.00653.

[9] Aric Hagberg, Pieter Swart, and Daniel Chult. *Exploring Network Structure, Dynamics, and Function Using NetworkX*. June 2008. DOI: 10.25080/TCWV9851.

[10] Abiodun M. Ikotun et al. "K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data". In: *Information Sciences* 622 (2023), pp. 178–210. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2022.11.139. URL: https://www.sciencedirect.com/science/article/pii/S0020025522014633.

[11] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL]. URL: https://arxiv.org/abs/1907.11692.

[12] Claudia Malzer and Marcus Baum. *A Hybrid Approach To Hierarchical Density-based Cluster Selection*. Sept. 2020. DOI: 10.1109/mfi49285.2020.9235263. URL: http://dx.doi.org/10.1109/MFI49285.2020.9235263.

[13] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2020. arXiv: `1802.03426 [stat.ML]`. URL: `https://arxiv.org/abs/1802.03426`.

[14] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: `1301.3781 [cs.CL]`. URL: `https://arxiv.org/abs/1301.3781`.

[15] Giancarlo Perrone, Jose Unpingco, and Haw-minn Lu. *Network visualizations with Pyvis and VisJS*. 2020. arXiv: `2006.04951 [cs.SI]`. URL: `https://arxiv.org/abs/2006.04951`.

[16] Nils Reimers and Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 2019. arXiv: `1908.10084 [cs.CL]`. URL: `https://arxiv.org/abs/1908.10084`.

[17] Victor Sanh et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020. arXiv: `1910.01108 [cs.CL]`. URL: `https://arxiv.org/abs/1910.01108`.

A APPENDIX

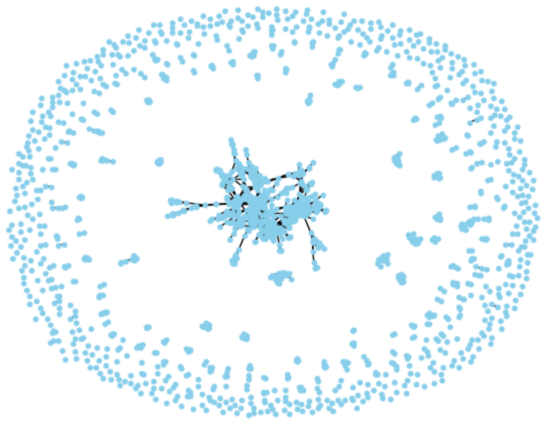

Figure 1: A plot of the graph with connected nodes according to keyword similarity or author overlap

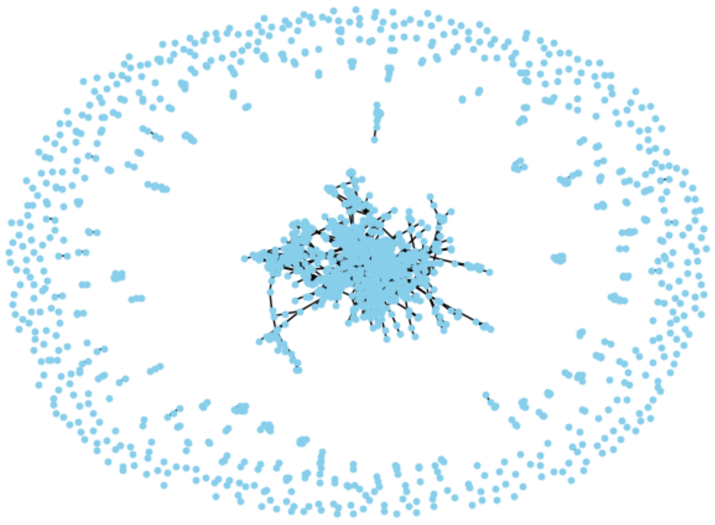

Figure 2: A plot representing the graph that has been constructed with the help of SentenceTransformer embeddings before running the Node2Vec algorithm

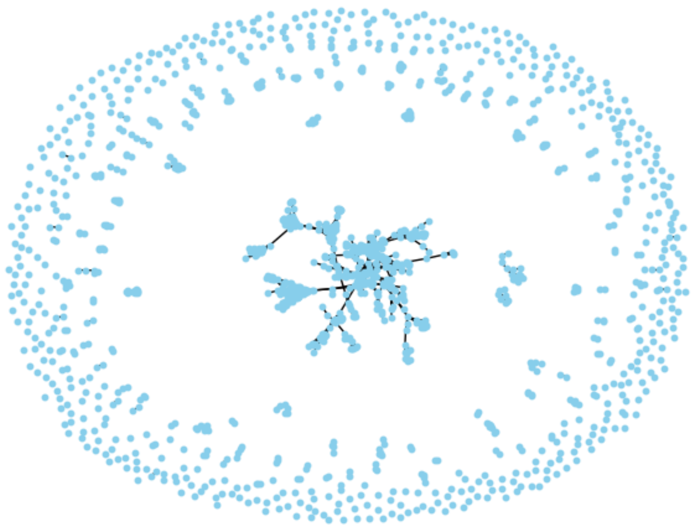

Figure 3: A plot representing the graph that has been constructed with using only the BERTopic embeddings

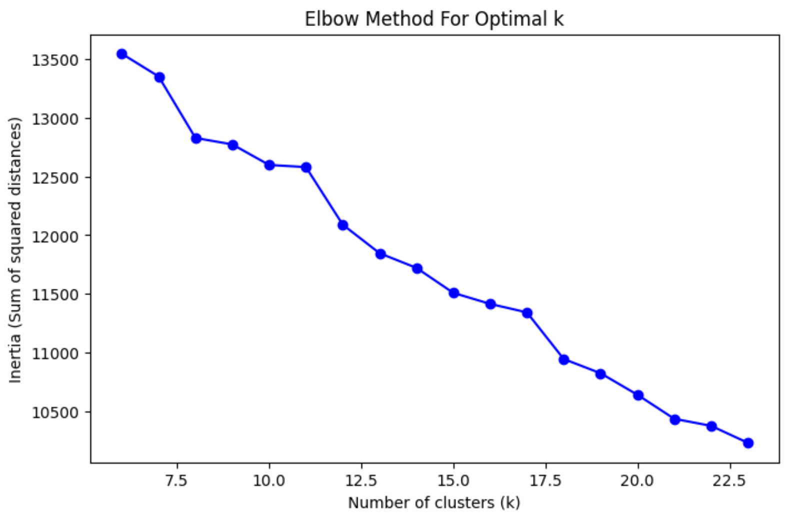

Figure 4: Elbow method plot for determining the optimal k value for k-means algorithm in node2vec model

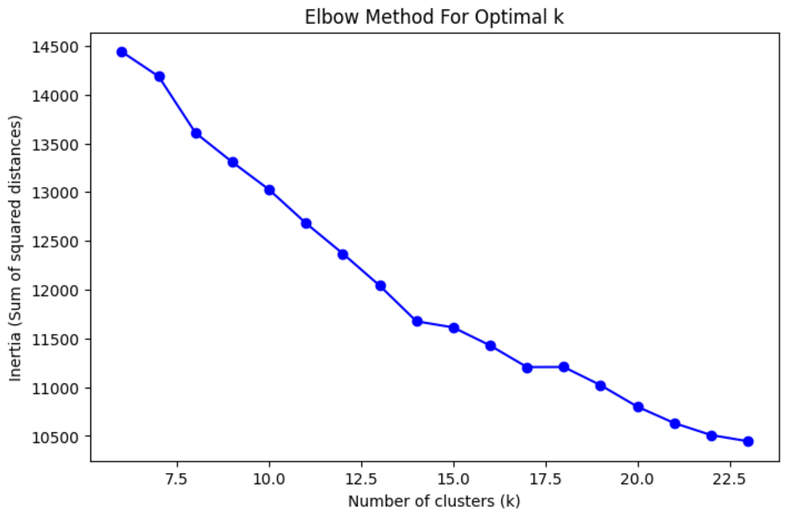

Figure 5: Elbow method plot for determining the optimal k value for k-means algorithm in sentence transformer model

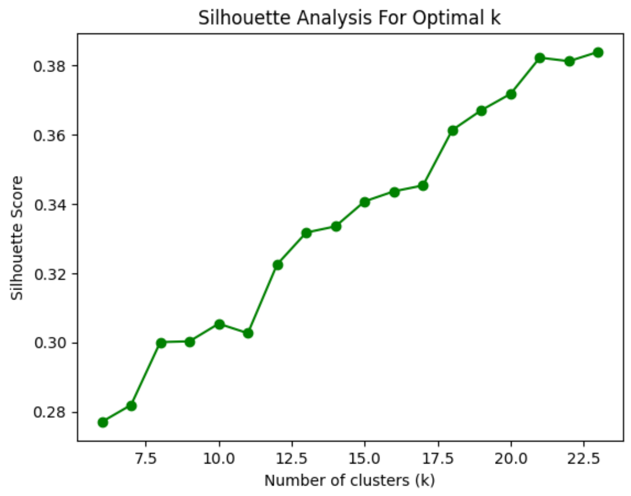

Figure 6: Silhouette score plot for calculating the optimal k value for k-means algorithm in Node2Vec model

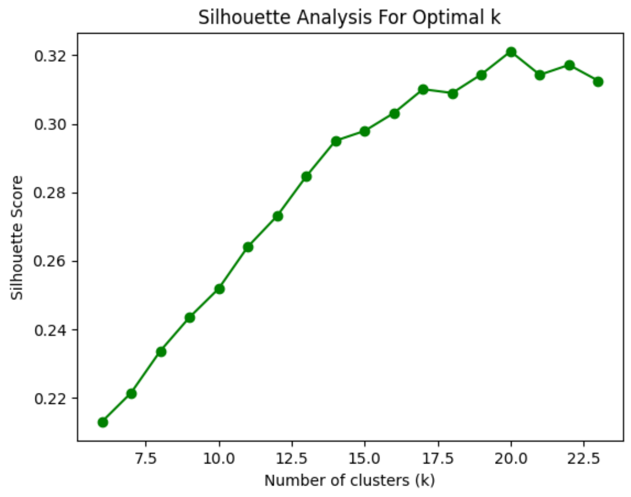

Figure 7: Silhouette score plot for calculating the optimal k value for k-means algorithm in SentenceTransformer model

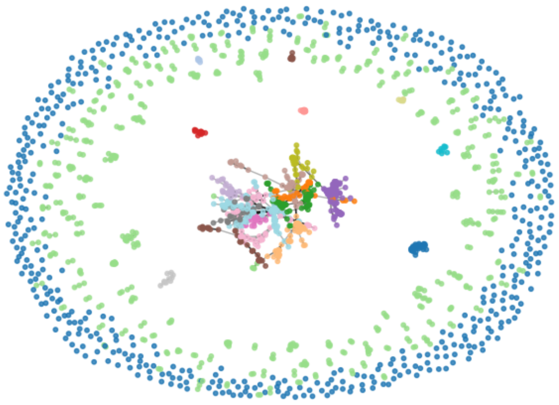

Figure 8: A graph visualization with nodes colored by their assigned clusters, as computed by the Node2Vec

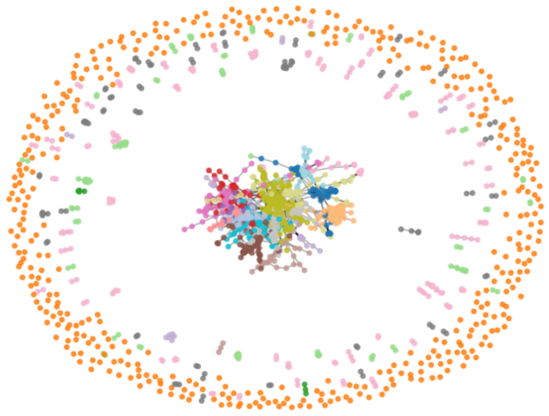

Figure 9: A graph visualization with nodes colored by their assigned clusters, as computed by the SentenceTransformer

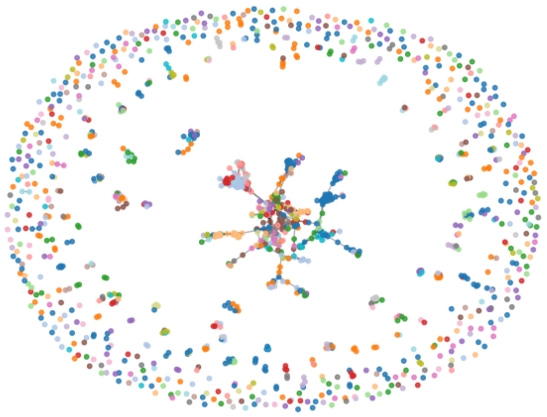

Figure 10: A graph visualization with nodes colored by their assigned topics, as computed by the BERTopic

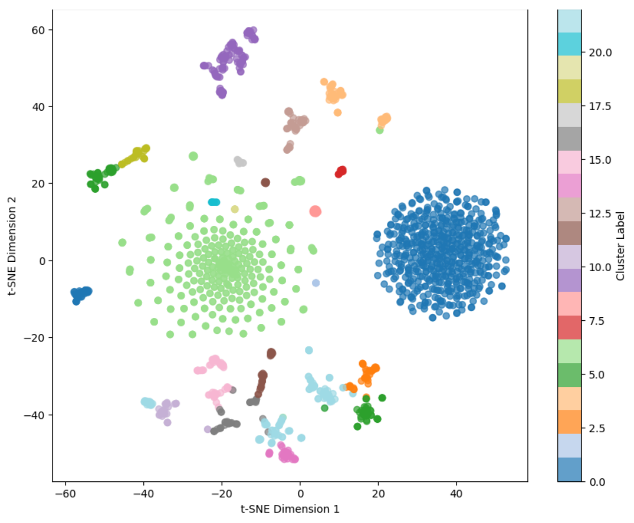

Figure 11: T-SNE visualization of node embeddings by Node2Vec

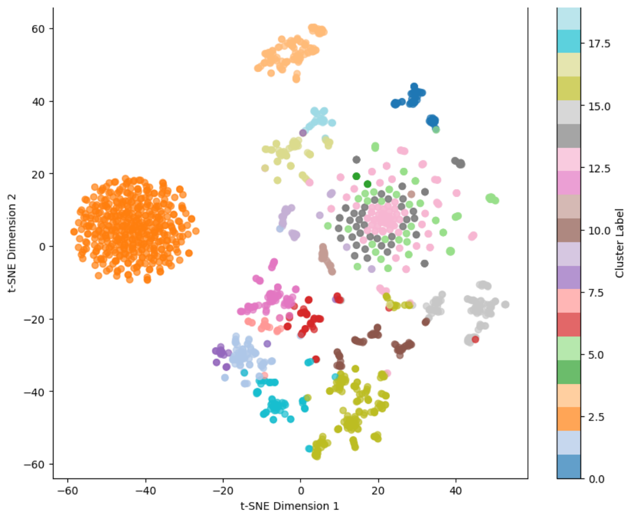

Figure 12: T-SNE visualization of node embeddings by SentenceTransformer

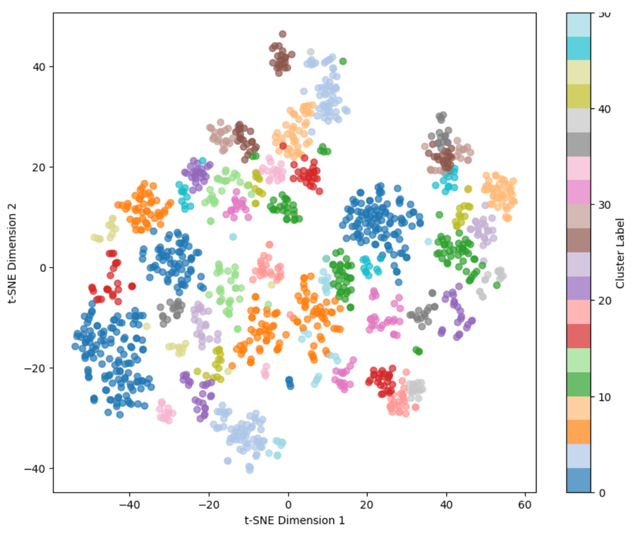

Figure 13: T-SNE visualization of node embeddings by BERTopic

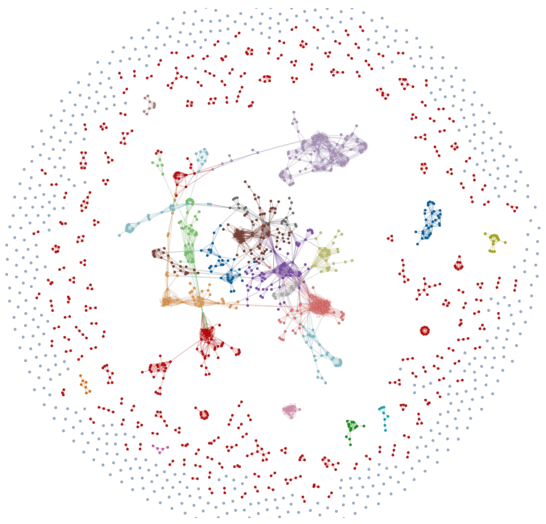

Figure 14: Interactive plot of the graph for the Node2Vec model

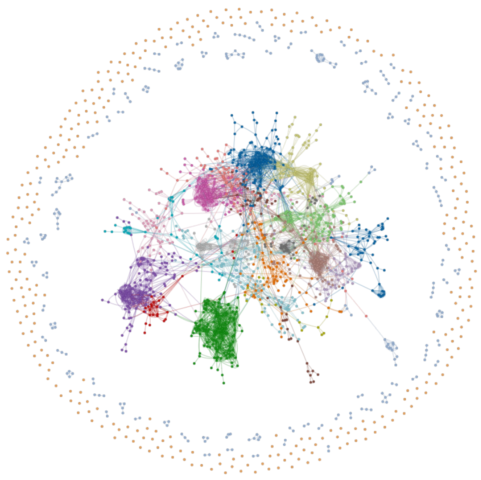
Figure 15: Interactive plot of the graph for the sentence transformer model

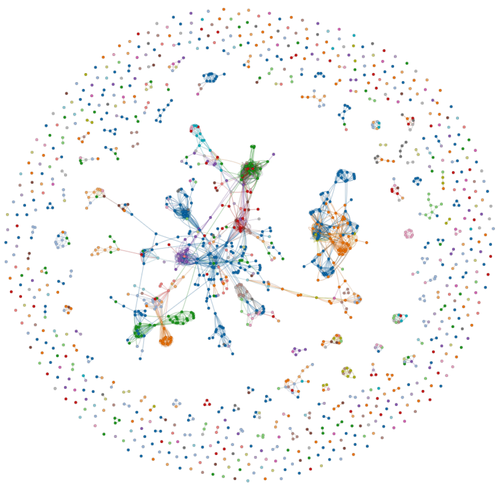
Figure 16: Interactive plot of the graph for the BERTopic model