PROJECT 2C:
Topic Modeling on GitHub README Files Using Classical, Embedding-based and Hyperbolic Models

1st Huynh Thai Linh, 2nd Truong Minh Khoa, 3rd Ho Gia Thanh
and Le Nhat Tung
HUTECH University, Vietnam
{huynh.thai.linh2004, truongminhkhoa16, hogiathanh572}@gmail.com and lenhattung@hutech.edu.vn

Abstract

This study investigates topic modeling on a large corpus of GitHub README files, comparing five approaches: LDA, NMF, BERTopic, Top2Vec, and the proposed GTFomer model. We evaluate performance using standard metrics including coherence, perplexity, Jensen–Shannon divergence, and topic cohesion. Experimental results show that NMF and BERTopic achieve high coherence for moderate topic numbers, whereas Top2Vec suffers from topic fragmentation due to noisy text. GTFomer, leveraging hierarchical graph neural networks in hyperbolic space, captures multi-level topic structures beyond the capacity of traditional models. These findings highlight the trade-offs between flat and hierarchical topic modeling, providing guidance for selecting models based on semantic granularity and interpretability in software documentation corpora.

Keywords

Topic Modeling, GitHub README Analysis, Natural Language Processing (NLP), BERTopic; LDA, NMF, Top2Vec, GTFomer

I. Introduction

In the era of the open-source software (OSS) [1] explosion, source code hosting platforms such as GitHub have evolved into global collaborative ecosystems containing vast amounts of technical knowledge and intellectual assets. Understanding the structure, content, and developmental trends of millions of repositories is crucial for both academia and industry. The **README.md** file, serving as the introductory, instructional, and representational document of each project [2], provides an invaluable source of unstructured textual data for uncovering hidden information and insights [3].

However, the exponential growth of repositories, combined with the inconsistency and sparsity of user-assigned topics (tags), presents a significant challenge for navigation, discovery, and content categorization [4], [5]. Traditional keyword-based search methods often fail to capture the semantic essence or conceptual topics of a project. To address this issue, Topic Modeling has emerged as a powerful unsupervised machine learning approach that automatically discovers latent thematic structures within large text corpora [6], [7].

This study tackles the aforementioned challenge by systematically applying, comparing, and evaluating state-of-the-art topic modeling techniques on a large-scale dataset of *GitHub README* files. We examine a spectrum of models—ranging from the classical probabilistic approach of Latent Dirichlet Allocation **(LDA)** [6] and the matrix factorization-based Non-negative Matrix Factorization **(NMF)** [8], to modern embedding-based techniques such as **Top2Vec** [9] and **BERTopic** [7], and a novel architecture, **GTFomer** [10], [11], designed to capture hierarchical dependencies and complex spatial relationships among topics.

The foundation of this analysis is a meticulously constructed large-scale dataset comprising **57,368 unique README.md** files, collected via the **GitHub GraphQL API**[1] across more than *50 diverse* technology topics. To ensure comprehensiveness and minimize bias stemming from popularity metrics, we designed a multi-directional crawling strategy: for each topic, we retrieved repositories evenly distributed across four criteria—*(1) most starred, (2) most forked, (3) most recently updated*, and *(4) best match*—with randomized pagination. After rigorous deduplication to ensure repository independence.

The main objectives of this research are threefold: (1) To quantitatively evaluate and compare the performance of *traditional* and *modern* topic modeling methods in the context of software technical documentation. (2) To introduce and assess the effectiveness of *GTFomer* in modeling hierarchically structured topics. (3) To provide empirical insights into the thematic landscape of open-source software projects on GitHub, laying the groundwork for applications such as repository recommendation systems, technology trend analysis, and documentation automation.

he remainder of this paper is organized as follows: Section 2 reviews related work. Section 3 describes the research methodology. Section 4 outlines the topic modeling approaches—*LDA, NMF, Top2Vec, BERTopic, and GTFomer*. Section 5 presents the results and discussion, and Section 6 concludes with key findings and future directions.

[1]GraphQL API: A query-based interface provided by GitHub for accessing repository metadata and content.

II. Related Work

Our study lies at the intersection of two rapidly evolving research areas: *Topic Modeling* and *Mining Software Repositories (MSR)*. This section provides an overview of foundational studies in both domains and identifies the research gaps that this work aims to address.

A. The Evolution of Topic Modeling Methods

Topic modeling methods have undergone a significant evolution—from classical probabilistic and statistical approaches based on the *bag-of-words* representation to modern models leveraging *semantic embeddings*.

Traditional topic models such as *Latent Dirichlet Allocation (LDA)* [6] and *Non-negative Matrix Factorization (NMF)* [8] are the most prominent and widely used approaches. LDA is a *generative probabilistic model* that assumes each document is a mixture of topics, and each topic is a probability distribution over words. NMF, on the other hand, is a *linear algebraic technique* that decomposes the document–term frequency matrix into two lower-dimensional matrices representing topic–word and document–topic associations. Despite their effectiveness, these models exhibit inherent limitations: they disregard word semantics and order, require extensive text preprocessing (e.g., stop-word removal and stemming), and demand manual specification of the number of topics (k), typically determined through heuristic metrics such as topic coherence.

Modern (semantic) topic models have emerged with the rise of word embedding models (e.g., Word2Vec, GloVe) and Transformer-based language models (e.g., BERT), giving birth to a new generation of topic modeling techniques. Among them, *Top2Vec* [9] and *BERTopic* [7] are two notable representatives, both operating directly within the semantic vector space.

- *Top2Vec* simultaneously embeds documents and words into a shared high-dimensional vector space. It employs UMAP [12] for dimensionality reduction and HDBSCAN [13] for clustering dense document regions to identify topics. A key advantage of Top2Vec is its ability to automatically determine the number of topics and generate semantically coherent topic representations.
- *BERTopic* follows a similar yet more flexible pipeline: it first generates document embeddings using BERT (or its variants), then applies UMAP for dimensionality reduction and HDBSCAN for clustering. Finally, it uses a variant of TF-IDF—class-based TF-IDF (c-TF-IDF)—to extract representative keywords for each identified topic.

While these models outperform traditional ones in capturing semantic information, they typically produce a flat list of topics, neglecting the hierarchical relationships naturally present in many domains—particularly in software engineering.

Hierarchical and Graph-based topic models were developed in response to the observation that topics often exhibit hierarchical structures *(e.g., Data Science → Machine Learning → Deep Learning)*. Recently, representing data in non-Euclidean geometric spaces, especially hyperbolic space, has shown great promise for modeling tree-like structures with minimal distortion [14], [15]. Models integrating Graph Neural Networks (GNNs) within hyperbolic space—such as GTFormer [10] (adapted and applied in this study)—enable the simultaneous capture of hierarchical and semantic relationships among topics, a dimension largely overlooked by most existing topic modeling approaches.

B. Text Mining in Software Repositories

The field of **Mining Software Repositories (MSR)** focuses on analyzing the rich data generated throughout the software development lifecycle to gain deeper insights into software ecosystems and processes. Textual artifacts such as *README files, commit messages*, and *issue discussions* provide invaluable sources of information for such analyses.

Numerous prior studies have employed topic modeling—predominantly *LDA*—to analyze GitHub repositories. Researchers have used topics to classify repositories [16], identify documentation barriers [17], and explore technological trends [18]. However, these studies often suffer from two major limitations: a heavy reliance on LDA, which fails to capture the deep semantic nature of technical text; and the use of datasets collected based on single-criterion sampling (e.g., repositories with the most stars), which introduces bias toward popular or mature projects.

C. Dataset Construction and Research Gap

One of the central challenges in MSR research lies in constructing a dataset that is both comprehensive and representative. Previous data collection efforts have tended to focus on "popular" repositories, overlooking emerging, academic, or niche projects.

To address this issue, our dataset was constructed using a multi-faceted crawling strategy. We collected **57,368 unique README** files (after deduplication) from *GitHub* via the *GraphQL API*, spanning over 50 diverse technology topics. This collection strategy was designed to ensure diversity and representativeness across repositories.

Research Gap: While many studies have applied individual topic modeling techniques, there remains a lack of a comprehensive benchmarking study comparing multiple generations of topic models on the same large-scale, well-structured dataset. Specifically, to the best of our knowledge, no prior work has directly compared the performance of *traditional models (LDA, NMF), modern semantic models (BERTopic, Top2Vec)*, and advanced *hierarchical models (GTFormer)* on a unified corpus of GitHub README files. This study aims to bridge that gap.

III. Methodology

This study proposes a comprehensive framework for extracting and modeling topics from GitHub README files. Our methodology consists of four main stages: (1) Data collection and dataset construction, (2) Domain-specific text preprocessing, (3) Application and comparative evaluation of topic models, and (4) Implementation of an advanced hierarchical model (GTFormer). The overall workflow is illustrated in Figure 1.

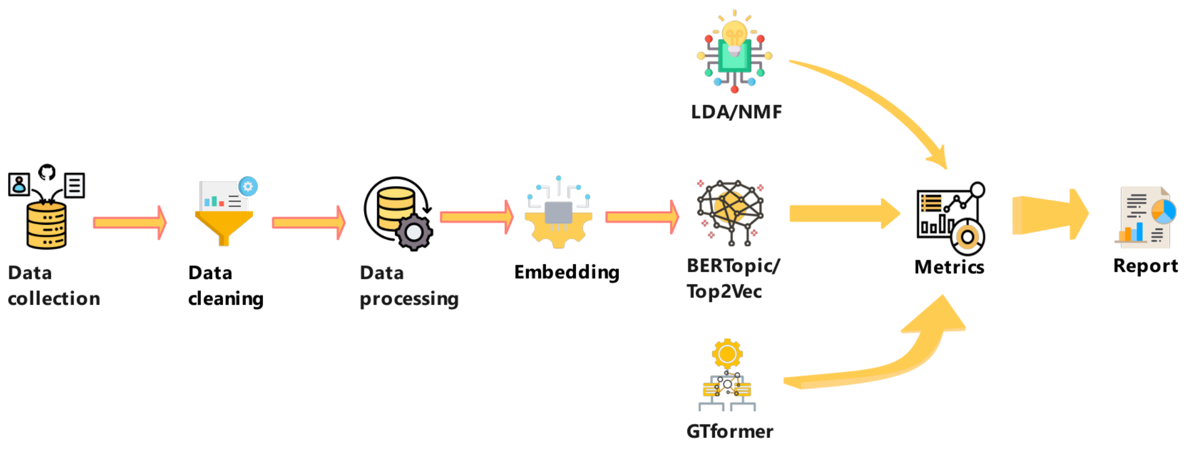

Figure 1: Flowchart for Data Preprocessing and Comparative Analysis of Topic Models.

A. Data Collection

In this study, **"Topic Modeling on GitHub README Files Using Classical, Embedding-based, and Hyperbolic Models"**, we constructed a comprehensive dataset by collecting data from GitHub using its GraphQL API. The data collection process, conducted from *October 3 to October 5, 2025*, was executed in multiple stages to ensure the dataset's breadth and diversity.

- The process began with the selection of **50 distinct topics** on GitHub. For each topic, we systematically crawled repository data by sampling from four different ranking categories: *most starred, most forked, recently updated*, and *best match.* We aimed to gather 500 repositories from each category, totaling an initial crawl of 2,000 repositories per topic, with randomized page ordering to minimize sampling bias.
- Next, we performed deduplication to remove redundant repositories, retaining only unique repositories for each topic. For topics with fewer 1000 repositories, additional samples were collected using random page sampling or star-range filtering to ensure balanced topic representation.
- For each repository, essential metadata such as repository ID, name, description, topics, and programming languages were extracted and stored in CSV format. The corresponding README.md files were then separately crawled and saved in JSONL format.

After data collection, we conducted a strict deduplication process based on the repository ID to ensure that each repository appeared only once. Finally, the resulting dataset consists of **57,368 README.md files** collected across *50 topics* were further reorganized into a taxonomy of *10* core information technology domains. *(e.g., "Web Development," "AI/Data," and "Software").* These domains presented in GitHub's Octoverse report. [19], [20]

B. Data Preprocessing

After compiling the dataset of **57,368 README.md files**. Given the semi-structured nature of README files, a specialized preprocessing pipeline was essential. A critical distinction was made: embedding-based models *(BERTopic, Top2Vec, GTFormer)* operate on minimally processed text to preserve semantic context, while classical models *(LDA, NMF)* require syntactically cleaned, tokenized text.

1. Pipeline for Classical Models (LDA, NMF)

For the TF-IDF vectorization model, we implemented a comprehensive preprocessing pipeline designed to reduce dimensionality and standardize the vocabulary. This process included the following steps:

- **Cleaning and Normalization:** The text was first converted to lowercase. Regular expressions were employed to systematically remove noise, including URLs, code blocks, Markdown syntax, and non-alphanumeric characters.
- **Tokenization:** The cleaned text was segmented into individual words (tokens) using the NLTK library's tokenizer.

- **Stop Word Removal:** High-frequency, low-information words were eliminated. This was achieved using a standard English stop word list, which was augmented with a custom list of common technical and project-related terms (e.g., 'install', 'usage', 'github', 'license') that were deemed non-discriminatory for clustering purposes.
- **Lemmatization:** Finally, tokens were converted to their base dictionary form using the WordNetLemmatizer from NLTK. This step ensures that different inflections of a word are treated as a single item (e.g., "running" and "ran" both become "run").

2. Pipeline for Embedding Models (BERTopic, Top2Vec, GTFormer)

For these models, only the raw README text was used, with the exception of the LSA scaffolding in GTFormer which used a TF-IDF matrix derived from this raw text. This approach allows the underlying transformer models **(all-MiniLM-L6-v2)** to leverage the full semantic and syntactic structure of the document.

Table I: Description of the collected datasets

File	Column	Description	Type	Non-Null Count
`github_repos.csv`	repo_id	Unique repository identifier	object	57,368
	name	Repository name	object	57,367
	full_name	Full repository path (user/name)	object	57,368
	description	Textual description of the repository	object	56,274
	topics	List of associated GitHub topics	object	57,368
	language	Main programming language	object	52,730
	stars_count	Number of stars	int64	57,368
	forks_count	Number of forks	int64	57,368
	created_at	Repository creation date	object	57,368
	updated_at	Last updated date	object	57,368
	url	Repository URL	object	57,368
`readme_data.json`	repo_id	Repository identifier	object	57,368
	full_name	Repository full name	object	57,368
	readme	Extracted README content	object	57,368
	timestamp	Retrieval timestamp	datetime64[ns]	57,368

C. Topic Modeling Frameworks

We implemented and compared five models, encompassing classical approaches (LDA, NMF), modern embedding-based methods (BERTopic, Top2Vec), and a novel hierarchical Graph Neural Network (GNN) architecture (GTFormer).

(1) Latent Dirichlet Allocation (LDA):

Latent Dirichlet Allocation (LDA) [21] is a generative probabilistic model that assumes each document is a mixture of latent topics, and each topic is a distribution over words. Formally, the model can be defined as follows:

$$\theta_d \sim \text{Dirichlet}(\alpha), \quad \phi_k \sim \text{Dirichlet}(\beta) \tag{1}$$

$$z_{dn} \sim \text{Multinomial}(\theta_d), \quad w_{dn} \sim \text{Multinomial}(\phi_{z_{dn}}) \tag{2}$$

where θ_d denotes the topic distribution for document d, ϕ_k the word distribution for topic k, and z_{dn} the topic assignment for the n-th word in document d.

The model was trained with varying topic numbers $k \in [5, 60]$ using the `LdaMulticore` implementation [22], and evaluated based on coherence [23], perplexity [24], and inter-topic divergence metrics.

a) Perplexity:: Perplexity is a standard metric to evaluate the generalization ability of probabilistic topic models and is defined as:

$$\text{Perplexity} = \exp\left(-\frac{1}{N}\sum_{d=1}^{N}\log p(w_d)\right) \tag{3}$$

as introduced in [24].

b) Jensen–Shannon Divergence:: The semantic separation between topic-word distributions is measured using the Jensen–Shannon (JS) divergence [25]:

$$JS(T_i, T_j) = \frac{1}{2}D_{KL}(T_i \| M) + \frac{1}{2}D_{KL}(T_j \| M), \quad M = \frac{T_i + T_j}{2} \tag{4}$$

where D_{KL} is the Kullback–Leibler divergence.

c) Jaccard Similarity:: To quantify topic overlap, Jaccard similarity [26] is used:

$$J(T_i, T_j) = \frac{|T_i \cap T_j|}{|T_i \cup T_j|} \tag{5}$$

d) Topic Superiority Score:: To combine multiple quality measures into a single interpretable metric, we propose the *Topic Superiority Score*, defined as:

$$S_{\text{topic}} = \frac{\text{Perplexity} \times \text{IoC}}{JS \times Jaccard} \tag{6}$$

e) Index of Coincidence (IoC):: The Index of Coincidence (IoC), originally used in information theory [27], measures intra-topic concentration as:

$$IoC = \sum_i p_i^2 \tag{7}$$

A higher IoC indicates greater lexical cohesion within a topic.

(2) Non-negative Matrix Factorization (NMF)

Non-negative Matrix Factorization (NMF) [28] is a linear-algebraic topic modeling method that decomposes a term-weight matrix (typically TF–IDF) into two non-negative factors. Given a document-term matrix $V \in \mathbb{R}^{m\times n}$, NMF seeks two non-negative matrices W and H such that:

$$V \approx WH, \quad W \in \mathbb{R}^{m\times k}, \quad H \in \mathbb{R}^{k\times n} \tag{8}$$

where W represents the document–topic associations and H captures the topic–word distributions.

The factorization is obtained by minimizing the reconstruction error under the Frobenius norm:

$$\min_{W,\, H\geq 0} \|V - WH\|_F^2 \tag{9}$$

This optimization can be solved via multiplicative update rules [29] or alternating least squares. The number of topics k was selected through coherence maximization, using the C_v measure [23] to ensure interpretability and semantic consistency across topics.

(3) Top2Vec

Top2Vec [30] integrates semantic embeddings with unsupervised clustering to jointly learn document and topic representations in a continuous vector space. Given document embeddings E_D, topics are obtained by density-based clustering (`HDBSCAN`) in the embedding space, and each topic vector $\mathbf{t}_k$ is defined as the centroid of its document cluster:

$$\mathbf{t}_k = \frac{1}{|\mathcal{C}_k|} \sum_{\mathbf{e}_d \in \mathcal{C}_k} \mathbf{e}_d \tag{10}$$

The top-n words per topic are those maximizing cosine similarity with $\mathbf{t}_k$:

$$\text{sim}(\mathbf{t}_k, \mathbf{w}_i) = \frac{\mathbf{t}_k \cdot \mathbf{w}_i}{\|\mathbf{t}_k\|\,\|\mathbf{w}_i\|} \tag{11}$$

4) BERTopic

BERTopic [31] leverages transformer-based embeddings (here, `all-MiniLM-L6-v2`) and class-based TF–IDF (c-TF–IDF) to extract interpretable topic–word distributions. The c-TF–IDF weight of a word w in topic t is defined as:

$$\text{c-TF–IDF}_{t,w} = \frac{f_{t,w}}{\sum_{w'} f_{t,w'}} \times \log\left(\frac{N}{n_w}\right) \tag{12}$$

where $f_{t,w}$ is the frequency of word w in topic t, n_w is the number of topics containing w, and N is the total number of topics. Topic representations are refined iteratively through UMAP dimensionality reduction [32] and HDBSCAN clustering [33].

D. GTFomer (Graph–Transformer Topic Model)

To capture hierarchical and relational dependencies among topics, we introduce GTFomer, a novel hybrid architecture integrating Graph Neural Networks (GNNs) and Transformer encoders within a hyperbolic space.

The proposed architecture consists of three main stages:

•LSA Topic Tree Construction

We first apply Latent Semantic Analysis (LSA)—implemented via TruncatedSVD on the TF-IDF document-term matrix—to obtain low-dimensional document representations in the latent semantic space. A hierarchical clustering algorithm with Ward linkage is then applied to these LSA vectors to construct a dendrogram, which captures the hierarchical relationships among documents. This tree structure serves as the foundation for modeling topic hierarchies.

•Hyperbolic Graph Encoding

Tree-like structures are inherently difficult to represent in Euclidean space. [34] The hyperbolic space—specifically the Poincaré ball model—offers a more suitable geometric framework, as its volume grows exponentially with radius, mirroring the branching nature of hierarchical trees. [35] We represent the LSA-derived topic tree as a graph and encode it using a Hyperbolic Graph Convolutional Network (HGCN), composed of HyperbolicGCNConv and HGNN layers. [36] Each HGCN layer performs hyperbolic graph convolution by mapping neighboring nodes into the tangent space, aggregating their representations, and then projecting the results back into the hyperbolic space.

•Hierarchy Prediction

The final hyperbolic node embeddings from the HGNN are fed into the GTFomer model. Before entering the Transformer layers, these representations are mapped back from the Poincaré ball to the tangent space at the origin using the logarithmic map: [37]

$$\text{logmap}_0^c(y) = \frac{1}{\sqrt{c}} \operatorname{artanh}(\sqrt{c}\,\|y\|)\,\frac{y}{\|y\|} \tag{13}$$

The resulting Euclidean tangent-space vectors are then processed through two output heads:

- **Level Predictor:** estimates the hierarchical depth (level) of each node in the tree.
- **Path Predictor:** predicts the probability of the path from the root to each node.

The model is trained by optimizing the CrossEntropyLoss function for the level prediction task on leaf nodes, corresponding to individual documents.

To identify the optimal number of topics K (or equivalent hyperparameters such as `min_topic_size` in BERTopic), we employed model-specific evaluation strategies tailored to each approach.

E. Determining the Optimal Number of Topics

For NMF, BERTopic, and Top2Vec, we optimized model performance using the Topic Coherence Score—specifically the C_v metric. The C_v coherence measure evaluates the semantic similarity among the top-ranked words within a topic by leveraging a sliding window approach combined with Normalized Pointwise Mutual Information (NPMI). A higher C_v indicates greater semantic consistency among topic words.

For LDA, due to its inherent stochastic nature, we designed a composite metric called the *Topic Superiority Score* [38], implemented in our framework as `Topic_superiority_score`. This metric jointly accounts for topic cohesion and topic separation, balancing the internal consistency of topics with their distinctiveness across the corpus:

$$\text{Score} = \frac{\text{Perplexity} \times \text{Avg\_IoC}}{\text{Avg\_JS\_Divergence} \times \text{Avg\_Jaccard}} \tag{14}$$

where:

- **Perplexity** measures how well the model predicts unseen documents (model generalization).
- **Index of Coincidence (IoC)** quantifies the peakedness of the word distribution within a topic, reflecting its internal cohesion.
- **Jensen–Shannon (JS) Divergence** measures the average dissimilarity among topic-word distributions, indicating inter-topic separation.
- **Jaccard Index** assesses the average lexical overlap between topics, providing another measure of separation.

IV. Baselines / Comparison Methods

To rigorously evaluate the effectiveness of GTFomer for topic discovery on software-engineering documentation, we benchmarked its performance against four widely-used topic modeling approaches: Latent Dirichlet Allocation (LDA), Non-negative Matrix Factorization (NMF), Top2Vec, and BERTopic. These models serve as strong baselines, representing probabilistic, matrix-factorization, embedding-based, and transformer-based paradigms, respectively.

All baseline models were trained on the preprocessed corpus and tuned across a consistent range of topic granularity to enable fair comparison. For coherence-based models *(LDA, NMF)*, the number of topics was varied from *5 to 60* in increments of 5. Embedding-based models *(Top2Vec, BERTopic)* were evaluated using transformer-based sentence embeddings to reflect modern semantic-aware topic discovery.

1) Baseline Models:

a) LDA (Latent Dirichlet Allocation): LDA is a probabilistic generative model that assumes documents are mixtures of latent topics and topics are distributions over words. LDA serves as a classical baseline for comparison. We optimize the number of topics using multiple metrics, including coherence, divergence-based topic separation, and topic redundancy measures.

b) NMF (Non-negative Matrix Factorization): NMF decomposes the document–term matrix into non-negative latent components, which are interpreted as topics. As a matrix factorization approach, NMF is often more stable than LDA on short-text corpora such as README content. Topic coherence is used as the primary selection criterion for determining the optimal number of topics.

c) Top2Vec: Top2Vec represents a neural embedding-based approach in which documents and words are mapped into a joint semantic space using transformer sentence embeddings. Topic clusters are subsequently discovered through hierarchical clustering in the embedding space. This model provides an important comparison point for transformer-based topic quality and semantic separation.

d) BERTopic: BERTopic advances embedding-based topic modeling by combining transformer embeddings, dimensionality reduction, and class-based TF–IDF (c-TF–IDF) to generate dense and semantically cohesive topic representations. We evaluate BERTopic across multiple minimum topic sizes to obtain the highest semantic coherence baseline.

2) Proposed Model: GTFomer: GTFomer is included as the target model for comparison, leveraging graph-structured representations of README semantics and transformer-based topic induction. Unlike the baselines, GTFomer explicitly incorporates relational structures across repositories and topic hierarchies, making it well-suited for metadata-enriched technical documentation such as GitHub projects.

3) Evaluation Protocol: All models were evaluated using a consistent data split (80/20 train–test). Quantitative evaluation focuses on:

- Topic semantic quality (coherence-based),
- Topic separation and redundancy,
- Topic diversity and interpretability,
- Stability across different topic granularities.

V. Experimental Results

This section presents the experimental evaluations conducted on the **GitHub README** corpus using five topic modeling approaches: *LDA, NMF, BERTopic, Top2Vec*, and the proposed *GTFomer model*. All models were trained on the preprocessed training split, while model selection and performance comparison were based on widely adopted topic quality metrics, including *Coherence (C_V), Perplexity, Jensen–Shannon Divergence (JSD), Jaccard Similarity,* and Index of *Coincidence (IoC).* Where appropriate, visualizations and result tables are referenced and denoted for later inclusion.

A. Experimental Setup

The study utilizes a dataset comprising 57,383 GitHub repositories, collected via the GitHub API. The dataset includes README files along with their associated metadata. It is partitioned into two subsets: a training set (80%) and a test set (20%), using a random stratified split with a fixed random state of 42.

Table II: Data after 80/20 Split

Dataset	**Number of Samples**
Training Set (80%)	45,894
Test Set (20%)	11,474

An important aspect of our experimental setup is the application of distinct preprocessing strategies tailored to the architectural characteristics of each model:

- For LDA and NMF (Classical Topic Models): The textual data *(cleanedText)* undergoes a rigorous preprocessing pipeline consisting of: removal of code blocks, Markdown syntax, URLs, and special characters; *lowercasing; tokenization; stopword removal (including both standard English stopwords* and a custom stopword list specific to software development, such as “install”, “usage”, “github”); followed by *lemmatization using WordNetLemmatizer* to reduce tokens to their base forms.
- For BERTopic, Top2Vec, and GTFormer (Transformer-based Models): We retain the *README text* in its near-original form (readme) to preserve contextual semantics that can be effectively leveraged by embedding models (e.g., *all-MiniLM-L6-v2).*

B. Hyperparameter Tuning and Model Configuration

The hyperparameter search was conducted on the training set for each model independently.

LDA (Latent Dirichlet Allocation):

We employ the `LdaMulticore` implementation from the gensim library. The input corpus is vectorized using CountVectorizer with `min_df = 10` and `max_features = 5000`. The number of topics k is tuned within the range [5,60] with increments of 5. Other parameters are kept fixed: `passes = 5`, `alpha = 'symmetric'` and `random_state = 42`. The optimal model is selected based on the lowest *Topic Superiority Score.*

NMF (Non-negative Matrix Factorization):

We employ the `sklearn` implementation of Non-negative Matrix Factorization (NMF), with input features encoded using `TfidfVectorizer` (`max_df` = 0.9, `min_df` = 2). The number of topics k is tuned within the range $[5, 60]$. The initialization method is set to `nndsvd` to improve numerical stability, with `random_state` = 42. The optimal model is selected based on the highest C_v coherence score.

BERTopic:

This model uses `all-MiniLM-L6-v2` from `SentenceTransformers` to generate document embeddings. Instead of predefining the number of topics, we tune the `min_topic_size` parameter in the range [5,60] with a step size of 5. The optimal model is selected based on the highest C_v Coherence score.

Top2Vec:

Top2Vec also uses `all-MiniLM-L6-v2` as the embedding model. We set `speed = "deep-learn"` to obtain high-quality embeddings and topic clustering. Since Top2Vec automatically detects the number of topics, no tuning of k is required.

GTFormer (Graph-Transformer Former):

This is the most complex model in the study, consisting of three key stages:

- **Tree Construction:** A hierarchical structure is first constructed by applying Latent Semantic Analysis (LSA) using `TruncatedSVD` with $n\_components = 50$ on the TF–IDF matrix, followed by hierarchical clustering using the Ward linkage method.
- **Hierarchical Embedding:** The resulting topic tree is then embedded into a hyperbolic space (Poincaré ball) using a `PoincaréModel` with `size` $= 2$, trained for 50 epochs.
- **GTFormer Architecture:** The model consists of a Hyperbolic Graph Neural Network (HGNN) module and a prediction module. Key hyperparameters include: `in_features` $= 50$ (from LSA), `hgnn_hidden` $= 1024$, `hgnn_out_dim` $= 512$, and curvature $c = 0.005$.
- **Training:** GTFormer is trained for *4000 epochs* using the Adam optimizer with a learning rate of 0.01, and `CrossEntropyLoss` for the level prediction task.

C. Performance of Individual Topic Modeling Approaches

Latent Dirichlet Allocation (LDA)

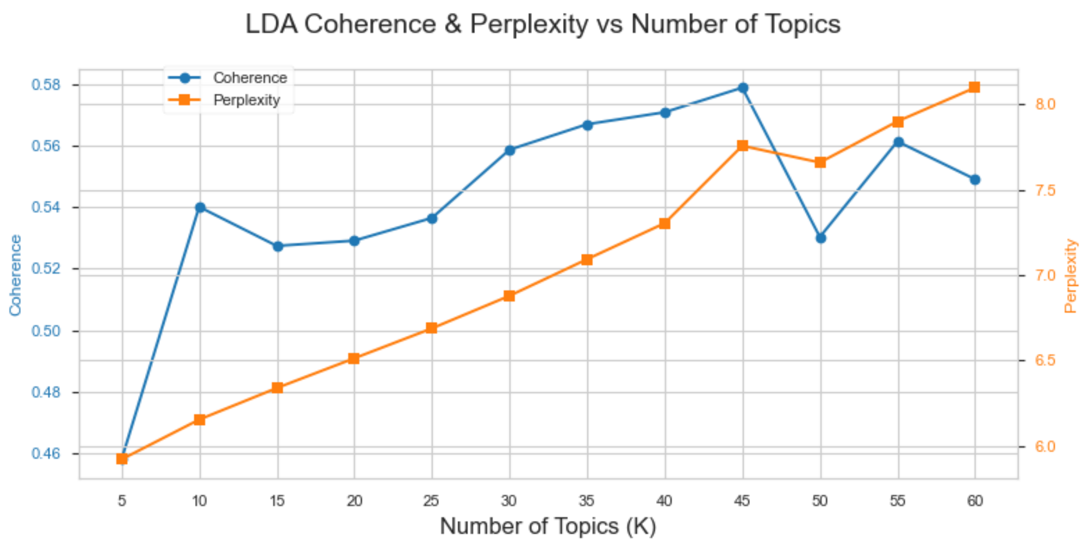

Figure 2: LDA Coherence & Perplexity

As the number of topics (K) increased from 5 to 45, *the Coherence score* exhibited a consistent upward trend, peaking at 0.579 when K=45. This indicates that the extracted topics became more semantically meaningful and interpretable as the model was allowed to form a finer-grained topic space. However, beyond K=45, a slight decline in coherence was observed, suggesting a saturation point at which additional topics no longer contribute to semantic clarity and may instead introduce redundancy.

In contrast, *Perplexity* increased steadily with larger K values, implying reduced generalization capability. This pattern aligns with common LDA behavior: while increasing K improves semantic clarity, it often leads to overfitting and a less compact representation of the corpus. Overall, the range K=35–45 appears to offer an optimal balance between interpretability and generalization.

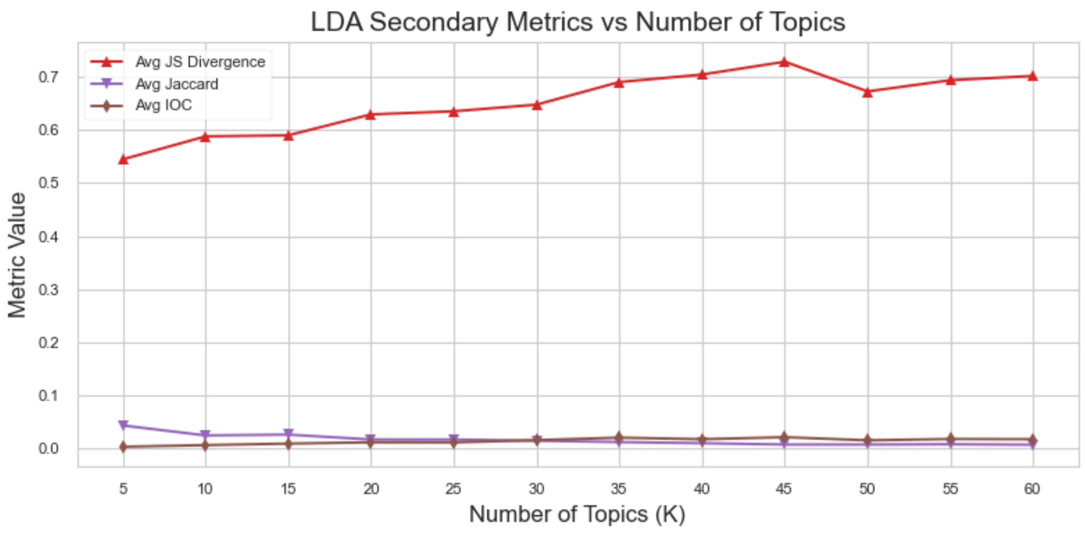

Figure 3: Secondary Metrics

Topic distinctiveness and internal cohesion were examined through three complementary metrics. The *Avg_Jaccard score* decreased markedly as K increased, indicating a reduction in vocabulary overlap and therefore more diverse topic formation. At the same time, *Avg_IOC* demonstrated a rising trend, meaning that words within each topic became more semantically cohesive. Furthermore, *Avg_JS_Divergence* increased with K, reflecting enhanced separation among topic distributions.

Collectively, these trends show that increasing the number of topics leads to more differentiated, internally coherent, and well-separated topics. Nevertheless, this improvement must be weighed against interpretability, which may decline when topics become overly granular.

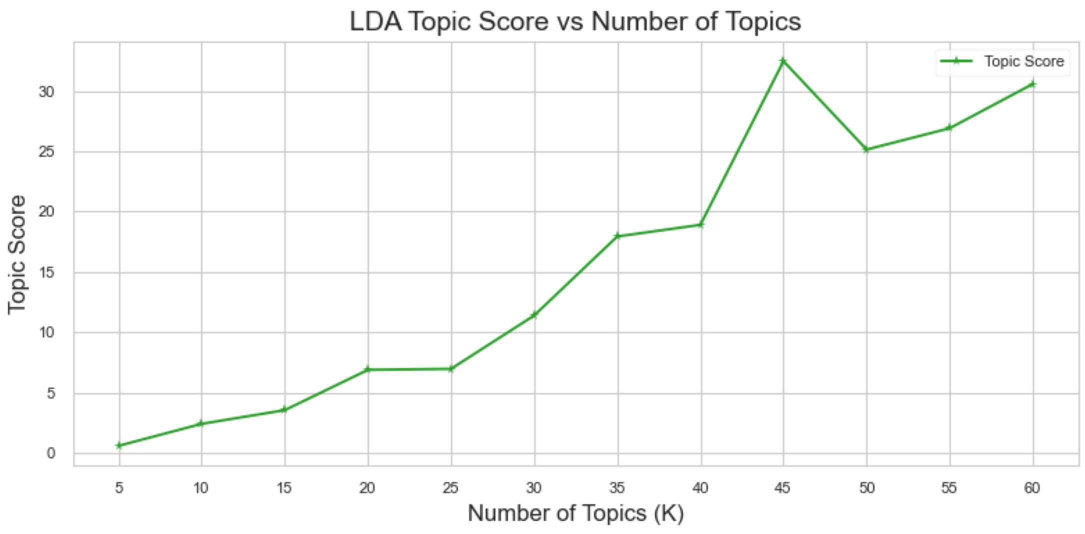

Figure 4: LDA Topic Score

The *Topic Score* increased substantially with K and reached its maximum value at K=45, after which it oscillated without clear improvement. This metric, which jointly accounts for topical cohesion and separation, suggests that topic quality improves significantly with a higher number of topics up to a threshold. The sharp decline at K=50 further reinforces that exceeding the optimal range leads to a degradation in topic quality due to topic fragmentation or redundancy.

Considering all metrics jointly, K=35–45 emerges as the most suitable range for this dataset of GitHub README documents, providing a strong balance between semantic interpretability, diversity, cohesion, and distributional separation of topics. Larger topic sizes beyond this range yield diminishing returns and may hinder practical interpretability.

Non-negative Matrix Factorization (NMF)

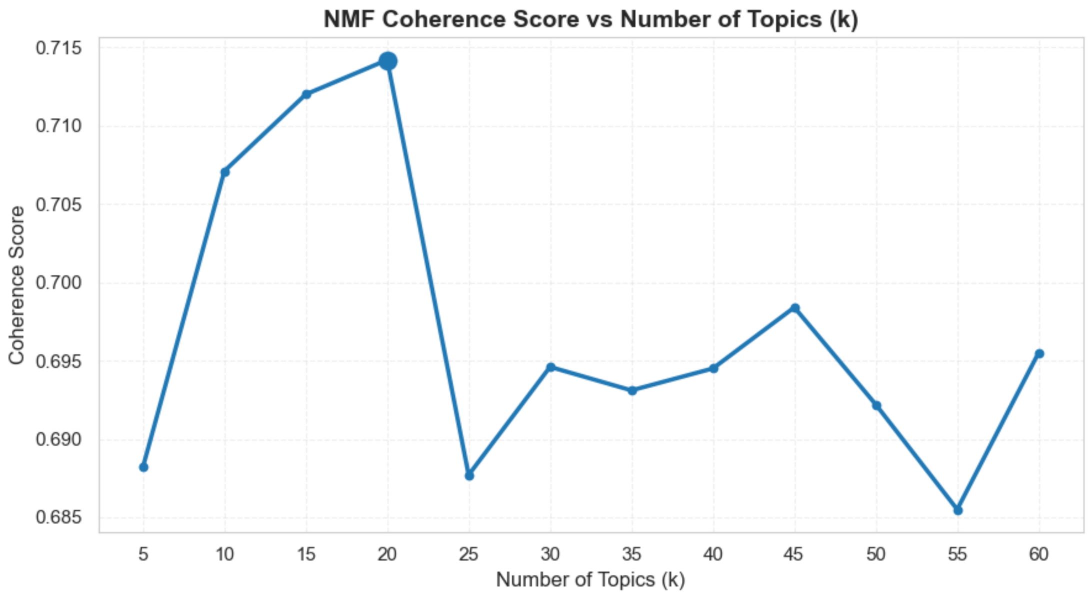

Figure 5: NMF Coherence Score

As k increases from 5 to 20, the coherence score gradually improves from 0.6882 to 0.7142, suggesting that a moderate increase in the number of topics enables the model to better disentangle the thematic structure of GitHub README documents.

The highest coherence is observed at $k = 20$ (0.7142), followed closely by $k = 15$ (0.7120) and $k = 10$ (0.7071). This indicates that NMF produces the most meaningful and semantically coherent topic representations when the number of topics lies within the range of 15 to 20. Beyond $k = 20$, the coherence score begins to fluctuate and generally declines, stabilizing around ~ 0.69 for values $k > 25$. This reflects topic fragmentation, where increasing the number of topics results in overly fine-grained and less consistent themes, thereby reducing semantic clarity.

Overall, the findings suggest that NMF performs most effectively on GitHub README data when configured with a moderate number of topics. The range of $k = 15$–20 provides an optimal balance between thematic coverage and interpretability, avoiding both under-segmentation and over-fragmentation of topics.

BERTopic

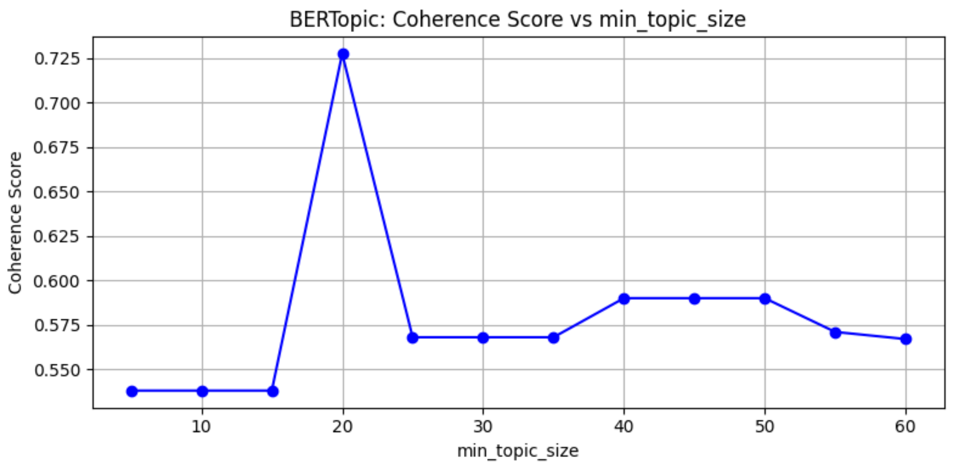

Figure 6: BERTopic Coherence Score"

To determine the optimal model configuration, we performed a rigorous tuning process for BERTopic's *min_topic_size* hyperparameter. This parameter controls the minimum cluster size detected by the underlying HDBSCAN algorithm, directly influencing the number and granularity of the generated topics. We tested a range of values for *min_topic_size* from [5, 60] and evaluated each model using the Topic Coherence score, where a higher score indicates more meaningful and semantically coherent topics.

The results, visualized in Figure 6, revealed a distinct performance peak. When *min_topic_size* was set at low values [5, 10, 15], the coherence score remained relatively low at 0.538. This suggests that the resulting topics were likely too granular or noisy, failing to capture meaningful broader thematic structures. However, at *min_topic_size = 20*, the coherence score experienced a sharp and significant increase to 0.72753, the highest observed score across all tested values.

Immediately following this optimal point, increasing the *min_topic_size* to 25 caused a drastic drop in coherence to 0.568. This trend persisted for larger values [25-60], which fluctuated in the 0.56-0.59 range. This abrupt decline indicates that *min_topic_size* values greater than 20 likely forced the model to merge semantically distinct topics, leading to over-generalized themes and reduced internal coherence.

Based on this empirical analysis, we selected *min_topic_size = 20* as the optimal hyperparameter value for our final BERTopic model. This value provides the best trade-off, maximizing the semantic coherence of the topics while avoiding both the topic fragmentation seen at lower values and the over-generalization observed at higher values.

Top2Vec

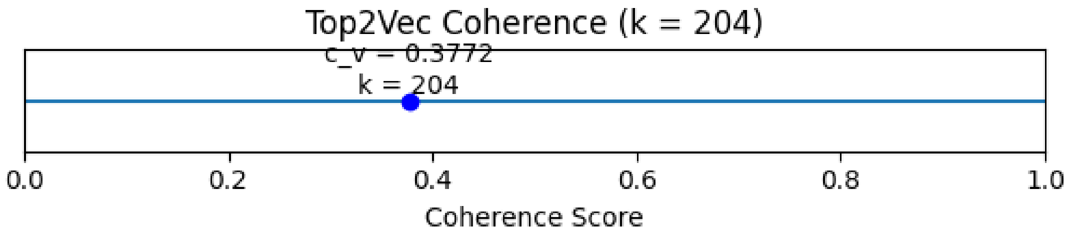

Figure 7: Top2vec Coherence Score

The experimental results from the Top2Vec model indicated an automatically detected topic count of 204 ($k = 204$). Unlike probabilistic models such as LDA (Latent Dirichlet Allocation), which require a predefined number of topics, Top2Vec utilizes the HDBSCAN clustering algorithm to identify "natural" topic clusters within the vector space. While this high topic count could reflect the high granularity and diversity of projects on GitHub, it more likely signals significant *topic fragmentation*. This phenomenon often occurs when noise or boilerplate text—such as installation guides, license information, or contribution guidelines—is misinterpreted by the model as distinct topics rather than as noise spanning across documents.

This assessment is further corroborated by the topic coherence score (c_v), which registered at 0.3772. This is a low-to-moderate result, indicating that the top keywords within many of the detected topics lack strong semantic relatedness. The combination of a very high topic count ($k = 204$) and low coherence ($c_v = 0.3772$) strongly suggests that the quality of the topics was negatively impacted by the noisy nature of the README file data. It is likely that boilerplate terms (e.g., `install`, `run`, `license`) "diluted" the semantic purity of meaningful topics (e.g., `pandas`, `numpy`, `dataframe`), thereby reducing overall topic coherence.

Table III: Comparison of Topic Models on Coherence Score

ID	Model	Type	Num_k	Coherence
0	LDA	Classic	5	0.63064
1	NMF	Classic	20	0.71422
2	BERTopic	Modern	20	0.72753
3	Top2Vec	Modern	204	0.37716

The comparative results demonstrate that modern topic modeling approaches generally outperform classical models in terms of coherence. BERTopic achieved the highest coherence score (0.72753) with 20 topics, indicating its strong capability to generate semantically rich and clearly distinguishable topics. NMF also performed well (0.71422), surpassing LDA (0.63064), suggesting a more effective separation of latent themes. In contrast, Top2Vec yielded the lowest coherence score (0.37716) despite producing a very large number of topics (204), reflecting high noise and weak thematic consistency.

D. GTFomer (Proposed Model)

In this study, GTFomer is implemented as a hierarchical topic modeling framework operating in hyperbolic space, integrating two core components: (i) a Hyperbolic Graph Neural Network (HGNN) for learning hierarchical topic graph representations, and (ii) a HypDRNN module for modeling semantic propagation across the topic tree. The objective of this experiment is to evaluate the capability of GTFomer in learning multi-level topic hierarchies and producing topic probability distributions for documents.

1) Preprocessing and Topic Tree Construction: To prepare the input for GTFomer, we first construct a hierarchical topic tree from the textual dataset.

- **Feature Extraction:** We employ
- **TfidfVectorizer** (with parameters `max_df=0.8`, `min_df=5`, and `stop_words='english'`) to transform the README documents into a TF-IDF matrix.
- Dimensionality Reduction and Initial Feature Construction: Truncated Singular Value Decomposition (TruncatedSVD) is applied to reduce the dimensionality to $d = 20$ (LSA). The resulting LSA vectors are used as the initial features for the leaf nodes of the tree.
- **Tree Structure Construction:** We adopt hierarchical agglomerative clustering with Ward's linkage on the LSA vectors, which minimizes the variance of the clusters being merged. This process yields a full binary tree. The features of each internal node are computed by taking the arithmetic mean of its two child nodes, forming a graph G.

Tree Construction Summary: The hierarchical tree constructed from this process consists of:

- Total number of leaf nodes (documents): 44,995
- Total number of nodes (leaf + internal): 89,989

All 89,989 nodes, together with the 89,988 edges connecting them, constitute the input graph for the HGNN model. The graph data is initialized as $x \in \mathbb{R}^{89989 \times 20}$.

2) Model Architecture and Experimental Configuration: We implement a customized GTFomer model comprising two main components:

HGNN Encoder (Hyperbolic Graph Neural Network): A GCN-based architecture operating in hyperbolic space (Poincaré ball) with curvature $c = 0.005$. The encoder consists of two `HyperbolicGCNConv` layers with the following dimensions:

$$in\_features = 20 \rightarrow hgnn\_hidden = 1024 \rightarrow hgnn\_out\_dim = 512$$

Prediction Heads: The hyperbolic embeddings Z_h obtained from the HGNN are projected into the tangent space at the origin $T_0\mathcal{P}_c^d$ via the logarithmic map $_0(Z_h)$. The resulting tangent-space (Euclidean) representations are then fed into two separate MLP heads:

- a *level_predictor* for predicting the level/depth of a node,
- a *path_predictor* for predicting the hierarchical path.

Table IV: GTFomer Hyperparameters Configuration

Parameter	Value	Description
`in_features`	20	Initial LSA feature dimension
`hgnn_hidden`	1024	Hidden dimension of the first HGNN layer
`hgnn_out_dim`	512	Output dimension of the HGNN encoder
Curvature (c)	0.005	Curvature of the Poincaré hyperbolic space
Optimizer	Adam	Optimization algorithm
Learning Rate	0.01	Step size for updating parameters
Epochs	4000	Number of training iterations
Gradient Clipping	1.0	Threshold to stabilize gradient updates
Total Parameters	721,716	Total trainable parameters

3) Evaluation Metrics: Perplexity and Structural Accuracy: A key consideration lies in the evaluation methodology. Traditional topic models such as LDA are typically assessed using *Perplexity*, which measures the model's ability to predict unseen words based on the learned topic-word distributions.

However, in the case of GTFomer, the primary objective is not to model word distributions, but rather to learn and preserve the *hierarchical structure*. Therefore, we evaluate the model using a proxy task: **Level Prediction**. Each of the 89,989 nodes in the tree is assigned a label corresponding to its depth, and the model is trained to classify this depth (a multi-class classification problem).

The node dataset is split into training, validation, and test sets using an 80/10/10 ratio:

- **Training set:** 71,991 nodes
- **Validation set:** 8,998 nodes
- **Test set:** 9,000 nodes

The model is trained using `CrossEntropyLoss` on this level prediction task.

4) Results Analysis: After 4000 training epochs, the model achieved the following performance:

- **Best Validation Perplexity:** 0.3255 (32.55%)

Interpretation: The perplexity value of 0.3255 indicates how well the model predicts hierarchical structures. While perplexity is typically lower in generative tasks, this value must be viewed within the context of the task:

- **Task Complexity:** This is a challenging multi-class classification task, where the model must predict the correct depth level from multiple hierarchical levels. For instance, if the tree has 20 levels, a random guess would yield a perplexity score corresponding to about 5% accuracy. Achieving a perplexity of 0.3255 suggests the model is significantly better than random guessing, capturing valuable information from the structure of the hierarchical tree.
- **Effectiveness of Hyperbolic Space:** The low perplexity demonstrates the effectiveness of embedding the hierarchical graph in hyperbolic Poincaré space. The HGNN encoder utilizes message passing in this curved space, enabling child nodes to incorporate information from their parent nodes. This contrasts with traditional Euclidean embeddings, which struggle to capture such hierarchical dependencies.
- **Limitations:** Despite the promising perplexity score, 0.3255 still indicates that the model struggles to fully capture the complexity of the hierarchy. This suggests that relying solely on LSA features, combined with the tree structure, might not be sufficient. Incorporating richer semantic features (such as BERT-based embeddings) or refining the loss function (e.g., by combining path loss with level loss) could reduce perplexity further and improve the model's performance.

In summary, GTFomer demonstrates the potential of using hyperbolic geometry to organize topics. It provides an insight into the hierarchical structure of GitHub repositories that goes beyond the capabilities of traditional flat topic models.

VI. Discussion

Overall, the experimental findings highlight notable differences in how classical, embedding-based, and hierarchical transformer-based approaches capture thematic structures within GitHub README documents.

Classical models (LDA, NMF) demonstrated strong performance when configured with a moderate number of topics, with NMF achieving the highest coherence (0.7142 at k=20). LDA showed improvement in coherence and topic quality as k increased, but exhibited diminishing returns beyond k=45, along with rising perplexity and topic fragmentation. These results suggest that while classical methods remain effective for interpretable topic extraction, they require careful tuning of k to balance granularity and generalization.

Embedding-based models revealed contrasting behavior. BERTopic achieved its strongest coherence at min_topic_size = 20, indicating its ability to form semantically meaningful clusters when noise is controlled. In contrast, Top2Vec automatically generated a large number of topics (204) with low coherence, implying sensitivity to boilerplate content and a tendency toward topic over-segmentation in software-related text.

The proposed GTFomer model provides a different perspective by modeling hierarchical topic structures rather than flat topic sets. By leveraging hyperbolic geometry and graph-based learning, GTFomer is able to encode multi-level semantic relationships that classical and embedding-based models cannot inherently capture. While its training is computationally more demanding, the hierarchical representations it learns offer enhanced interpretability for complex, layered domains such as software repositories.

In summary, the results demonstrate a trade-off between interpretability, coherence, scalability, and hierarchical expressiveness across the five approaches. Classical models provide concise and coherent flat topics, embedding-based models benefit from semantic richness but vary in robustness to noise, and GTFomer introduces a promising direction for hierarchical topic modeling where structural relationships between topics are essential.

VII. Conclusion

In this work, we systematically evaluated five topic modeling approaches on GitHub README data. Classical methods (LDA, NMF) are effective for extracting coherent flat topics when carefully tuned, while embedding-based models (BERTopic, Top2Vec) offer semantic richness but vary in robustness to noise. The proposed GTFomer model extends topic modeling to hierarchical structures, successfully capturing multi-level semantic relationships using hyperbolic graph embeddings. Overall, our results demonstrate that model choice should consider both topic coherence and the desired structural representation, with GTFomer providing a promising direction for hierarchical analysis of complex technical corpora. Future work may explore richer semantic features and human-centered evaluation for topic interpretability.

References

[1] E. Kalliamvakou, D. Damian, K. Blincoe, L. Singer, and D. German, "Open source-style collaborative development practices in commercial projects using github," 05 2015.

[2] G. A. A. Prana, C. Treude, F. Thung, T. Atapattu, and D. Lo, "Categorizing the content of github readme files," 2018. [Online]. Available: https://arxiv.org/abs/1802.06997

[3] M. Gaughan, K. Champion, S. Hwang, and A. Shaw, "The introduction of readme and contributing files in open source software development," 02 2025.

[4] N. Munaiah, S. Kroh, C. Cabrey, and M. Nagappan, "Curating github for engineered software projects," 12 2016.

[5] A. Lima, L. Rossi, and M. Musolesi, "Coding together at scale: Github as a collaborative social network," 2014. [Online]. Available: https://arxiv.org/abs/1407.2535

[6] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," vol. 3, 01 2001, pp. 601–608.

[7] M. Grootendorst, "Bertopic: Neural topic modeling with a class-based tf-idf procedure," 2022. [Online]. Available: https://arxiv.org/abs/2203.05794

[8] S. Cho, "Lee and seung (2000)'s algorithms for non-negative matrix factorization: A supplementary proof guide," 2025. [Online]. Available: https://arxiv.org/abs/2501.11341

[9] D. Angelov, "Top2vec: Distributed representations of topics," 2020. [Online]. Available: https://arxiv.org/abs/2008.09470

[10] D. C. Zhang, M. Yang, X. Wu, J. Zhang, and H. W. Lauw, "Hierarchical graph topic modeling with topic tree-based transformer," 2025. [Online]. Available: https://arxiv.org/abs/2502.11345

[11] F. Lopez, M. Fey, and J. Leskovec. An introduction to graph transformers - kumo. [Online]. Available: https://kumo.ai/research/introduction-to-graph-transformers/

[12] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2020. [Online]. Available: https://arxiv.org/abs/1802.03426

[13] R. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," vol. 7819, 04 2013, pp. 160–172.

[14] S. Shahid, T. Anand, N. Srikanth, S. Bhatia, B. Krishnamurthy, and N. Puri, "Hyhtm: Hyperbolic geometry based hierarchical topic models," 2023. [Online]. Available: https://arxiv.org/abs/2305.09258

[15] Hierarchical graph topic modeling with topic tree-based transformer. [Online]. Available: https://arxiv.org/html/2502.11345v1

[16] M. M. Aslam, M. Farhan, A. Raza, J. Iqbal, and M. Iqbal, "A smart model for categorization of github repositories," *KIET Journal of Computing and Information Sciences*, vol. 6, p. 50, 07 2024.

[17] J. Bjerkan, V. Valderaune, and R. Olsen, "Patient safety through nursing documentation: Barriers identified by healthcare professionals and students," *Frontiers of Computer Science (electronic)*, vol. 3, 06 2021.

[18] J. Shuford and M. M. Islam, "Exploring the latest trends in artificial intelligence technology: A comprehensive review," *Journal of Artificial Intelligence General science (JAIGS) ISSN:3006-4023*, vol. 2, 02 2024.

[19] G. Staff. Octoverse: AI leads python to top language as the number of global developers surges. [Online]. Available: https://github.blog/news-insights/octoverse/octoverse-2024/

[20] Build software better, together. [Online]. Available: https://github.com

[21] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[22] R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 2010, pp. 45–50.

[23] M. Röder, A. Both, and A. Hinneburg, "Exploring the space of topic coherence measures," in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, 2015, pp. 399–408.

[24] H. M. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno, "Evaluation methods for topic models," *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1105–1112, 2009.

[25] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 145–151, 1991.

[26] P. Jaccard, "The distribution of the flora in the alpine zone," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[27] M. G. Kendall, *Advanced Theory of Statistics*. Charles Griffin & Co., 1948.

[28] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[29] ——, "Algorithms for non-negative matrix factorization," *Advances in Neural Information Processing Systems*, vol. 13, pp. 556–562, 2001.

[30] D. Angelov, "Top2vec: Distributed representations of topics," *arXiv preprint arXiv:2008.09470*, 2020.

[31] M. Grootendorst, "Bertopic: Neural topic modeling with a class-based tf-idf procedure," *arXiv preprint arXiv:2203.05794*, 2022.

[32] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[33] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2013, pp. 160–172.

[34] Hyperbolic graph learning: A comprehensive review. [Online]. Available: https://arxiv.org/html/2202.13852v3?utm_source=chatgpt.com

[35] O.-E. Ganea, G. Becigneul, and T. Hofmann, "Hyperbolic entailment cones for learning hierarchical embeddings."

[36] Y. Xu, Z. Zang, B. Hu, Y. Yuan, C. Tan, J. Xia, and S. Z. Li, "Complex hierarchical structures analysis in single-cell data with poincaré deep manifold transformation," vol. 26, no. 1, p. bbae687, _eprint: https://academic.oup.com/bib/article-pdf/26/1/bbae687/61616788/bbae687.pdf. [Online]. Available: https://doi.org/10.1093/bib/bbae687

[37] O.-E. Ganea, G. Bécigneul, and T. Hofmann, "Hyperbolic neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[38] J. Gan and Y. Qi, "Selection of the optimal number of topics for LDA topic model—taking patent policy analysis as an example," vol. 23, no. 10, p. 1301. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC8534395/