תמונה שמכילה גופן, לוגו, סמל, גרפיקה

התיאור נוצר באופן אוטומטי

Software Engineering Department Braude College

Capstone Project Phase A

24-2-R-14

**Anomaly Detection in Texts Using Graph-Based Methods: Leveraging Graph Fourier Transform and Laplacian**

Supervisor:

Prof. Zeev Volkovich

Submitters:

Alexander Kovtonyuk– [alexkobt3@gmail.com](mailto:alexkobt3@gmail.com)

Github Repository:  
<https://github.com/Theoddalex/Anomaly-Detection-in-Texts-Using-Graph-Based-Methods.git>

**Table of Content**

.**1**[**General Description** 3](#_Toc165634805)

**.2**[**Solution Description** 4](#_Toc165634806)

**2.1** [**Code Algorithm** 4](#_Toc165634807)

**2.2** [**Implementation and Iteration** 5](#_Toc165634808)

**2.3** [**Research and Development Process** 5](#_Toc165634814)

**2.3.1** [**Early Approach** 6](#_Toc165634815)

**2.4**  [**Final Implementation** 8](#_Toc165634816)

**2.5** [**Testing and Evaluation** 8](#_Toc165634817)

**2.6** [**Dataset selection and Preprocessing 8**](#_Toc165634818)

[**2.7 Parameter Sensitivity and Graph Analysis** 9](#_Toc165634819)

[**2.8 Reflectin and Iterartion 9**](#_Toc165634820)

[**2.9 Tools Used** 9](#_Toc165634821)

**.3** [**Challenges and Solutions** 10](#_Toc165634822)

**.4**[**Results and Conclusions** 11](#_Toc165634823)

**4.1**[**Hyperparameters** 11](#_Toc165634824)

[**4.2 Pride and prejudice Dataset results** 13](#_Toc165634825)

[**4.3 The Greek New Testament Dataset Results** 14](#_Toc165634826)

[**4.4 And Quiet flows Don Dataset results** 14](#_Toc165634827)

[**4.5 Conclusions** 15](#_Toc165634827)

[**4.6 Lessons Learned** 16](#_Toc165634827)

[**.5 Project Benchmarks** 17](#_Toc165634828)

[**5.1 Achievements** 18](#_Toc165634829)

[**5.2 Testing** 18](#_Toc165634830)

[**6. User's Guide Opereation Instrctuions** 19](#_Toc165634831)

[**7. Bibliography** 20](#_Toc165634832)

## 

## **1. General Description**

**Project Goal**: This project explores anomaly detection in textual data using a novel hybrid approach that leverages both semantic embeddings and structural text adjacency in a graph-based framework. By integrating BERT embeddings for semantic understanding, employing the Graph Fourier Transform, and applying spectral graph analysis for outlier detection, we capture anomalies characterized by either abrupt context shifts or inconsistent structural patterns. Our pipeline includes preprocessing large text corpora into manageable chunks, constructing a hybrid adjacency graph, computing the normalized Laplacian, and identifying high-frequency components—these serve as indicators of anomalies. Empirical evaluations show that the method successfully detects both synthetic and real-world anomalies with high precision and scalability.

**Project Implementation:** The implementation employs a hybrid view of the text, where each segment or “chunk” is transformed into a rich representation that reflects its contextual meaning. These representations are then compared to each other, highlighting how similar or dissimilar different parts of the text might be. By treating these chunks and their relationships as a network or graph, the approach makes it possible to measure how well each segment aligns with its neighbors and with the overall structure of the document. Detecting anomalies, then, becomes an exercise in spotting points in this network whose connections suggest they are out of place. Although this method involves a few under-the-hood steps such as splitting the text into chunks and encoding them using language models at a high level, the process is centered on analyzing how segments connect or deviate from one another. These methods can be extended or adapted to larger datasets, more specialized documents, or real-time feeds, depending on the needs of the user.

**User Audience:** While the core technical foundation will likely appeal to data scientists and machine learning researchers interested in newer approaches to text analysis, the project’s emphasis on revealing hidden or misleading content also makes it valuable for anyone involved in content moderation, information security, or analytics in sensitive fields like finance and healthcare. Engineers or product teams aiming to integrate anomaly detection into their workflows might use this approach to enrich existing dashboards or alerting systems, and educators can harness the project’s conceptual framework to illustrate modern text-processing techniques in a classroom setting.

## **2. Solution Description**

**2.1 Code Algorithm**

**Data Preprocessing**

* **Sentence Tokenization**: Split the input text into individual sentences using NLTK, discarding those shorter than a certain length threshold (e.g., 30 characters) to reduce noise and spurious anomalies.
* **Chunking**: Group the remaining sentences into fixed-size chunks (e.g., five sentences per chunk). This balances contextual richness against computational overhead.

**Embedding Generation**

* **BERT Embeddings**: For each chunk, generate a dense vector representation using a pre-trained BERT model (e.g., bert-base-multilingual-cased). This step captures deeper semantic nuances than traditional bag-of-words or keyword-based methods.

**Graph Construction**

* **k-NN Semantic Adjacency**:
  + Compute a cosine similarity matrix across all chunk embeddings.
  + For each chunk (node), find its top-k most similar neighbors according to cosine similarity.
  + Form edges for these node pairs, assigning higher weights to pairs with stronger similarity.

**Node Signal Computation**

1. **Global Distance**: Calculate the Euclidean distance between each chunk’s embedding and the global mean embedding of all chunks.
2. **Local Mismatch**: For each node, measure the average Euclidean distance to its neighbors within the graph.
3. **Combined Signal**: Weight these two metrics (e.g., α⋅Global Distance+β⋅Local Mismatch) to create a single score reflecting how anomalous each chunk is, both globally and relative to its local neighborhood.

**Spectral Analysis**

1. **Normalized Laplacian**: Construct the matrix *Lnorm=D-A* ​, where A is the adjacency matrix and D is the degree matrix.
2. **Eigen-decomposition**: Decompose *Lnorm* to retrieve its eigenvalues and eigenvectors; the larger eigenvalues often correspond to higher-frequency variations in the graph.
3. **Outlier Detection**: Project each node’s combined signal onto the high-frequency components. Nodes with the largest high-frequency amplitudes are flagged as anomalies.

**2.2. Implementation & Iteration**

During development, we experimented with different K values to tune the complexity of the k-NN graph. We also adjusted the weighting factors (α,β) to determine the right mix of global vs. local emphasis when computing node signals. Small-scale tests helped identify a balanced configuration before applying the pipeline to larger, more complex texts.

**Evaluation**

1. **Synthetic Anomalies**: We inserted artificially crafted chunks such as foreign-language text or random gibberish and anomalies that are in the same language as the dataset but vary differently from the text itself into otherwise coherent corpora to test if our pipeline flagged these segments as outliers.
2. **Real-World Text**: We applied the method to literary texts (e.g., MOBY-DICK , Pride and Prejudice and more) and observed that very short or cryptic sentences were often misidentified. Filtering sentences below 30 characters greatly reduced false positives.

**Documentation & Finalization**

We finalized the pipeline by packaging all code modules data preprocessing, BERT embedding generation, graph construction, and spectral analysis into a reproducible workflow. Comprehensive documentation, including inline comments and usage instructions, was added to aid future maintenance. Additionally, visual analyses (e.g., histograms of node signals) were prepared to illustrate how anomalies stand out in the high-frequency components of the Laplacian decomposition.

**2.3 Research/Development Process**

**Problem Identification**: We recognized the need for a reliable way to detect anomalies in textual data, particularly in multilingual corpora. Many existing methods rely on surface-level features or rudimentary statistics, which often fail to capture deeper contextual clues. Therefore, we aimed to design an approach that could reveal subtle irregularities by exploring the semantic space of text segments.

**Literature Review**: Our initial research surveyed both classic anomaly detection methods like simple statistical outlier detection and more modern graph-based approaches leveraging deep language models. We also considered techniques that combined text-order adjacency with semantic adjacency to preserve both structure and content. However, we found that focusing on semantic similarity alone could yield good results for detecting abrupt semantic shifts.

**2.3.1. Early Approaches:**

**Text-Order Syntactic Graph Construction**

In the early stages of the project, we validated the pipeline using a small dataset of 30 sentences, ensuring the correctness of core components such as graph construction, Laplacian matrix calculation, and visualizations. This phase allowed us to test our implementation before scaling up to larger datasets.

For graph construction, we initially adopted a k-nearest neighbors (k-NN) approach, connecting nodes based on syntactic similarity by retaining the sequential flow of text. We modified the connection rules so that each node was linked to its k-nearest neighbors symmetrically. For instance, with *k=3*, a node connected to its *k−1* and *k+1* neighbors, for *k=5*, it was linked to *k−2* and *k+2*, forming a broader network. We experimented with *k={3,5,7,9,15,21,30, …}* to analyze the impact of connectivity density on anomaly detection.

However, results across all *k* values were underwhelming. Despite creating structured graphs with varying levels of connectivity, the method failed to reliably detect the inserted anomalies in this small dataset. This suggested that our initial syntactic-based approach did not sufficiently capture meaningful irregularities, prompting us to explore alternative methods for improving anomaly detection performance.

A diagram of a network

Description automatically generated

Figure 1: Text-Order Syntactic Graph Construction, Built with *k=7* and cosine similarity calculated weights.

**Semantically focused Graph Construction POC:**

To capture more nuanced relationships, we adopted a purely semantic approach based on BERT embeddings. With the newer approach, every sentence in the text is converted into a high-dimensional vector using a multilingual BERT model, and a k-nearest neighbors (k-NN) graph is constructed using cosine similarity. This means that each node in the graph is directly connected only to the sentences it most closely resembles in semantic space. By focusing on the actual meaning and content of text sentences, rather than on their proximity in the document, we can more reliably isolate anomalies that deviate from the overall theme or language style. For instance, foreign-language inserts or even just text with drastically different topics will stand out in a semantic graph because their BERT embeddings differ substantially from those of their neighbors.

This shift toward semantic adjacency yielded significantly better results in our tests, improving the detection of anomalies that might otherwise go unnoticed if we relied solely on sentence order. By analyzing the normalized Laplacian of this k-NN graph and looking at high-frequency components in the node signal, we more consistently flagged inserted anomalies especially when they were in different languages or conveyed content unrelated to the main text. This approach not only simplified tuning efforts (by focusing on one type of adjacency rather than balancing multiple) but also provided a more robust framework for capturing the wide range of ways textual anomalies can manifest.  
  
A network of dots and lines

Description automatically generated

Figure 2 Semantically focused Graph Construction, Built with *k=5* and cosine similarity calculated weights anomalies found represented in red.

**2.4 Final Implementation**

After creating a proof of concept (PoC) using the semantic graph construction, we moved forward by initially treating each sentence as a single node in the graph. This approach worked for small texts but proved both computationally intensive and less reliable when applied to lengthier works. For example, large novels or academic manuscripts contain thousands of sentences, making sentence-level embedding generation and graph building prohibitively slow. Moreover, the analysis at a per-sentence granularity often yielded mediocre detection results, as single-sentence embeddings could be too fragmented or noisy to capture meaningful context.

To address these challenges, we decided to chunk the text into groups of four or five sentences, treating each chunk as a single node. Bundling sentences together provided a richer context for the BERT embeddings, capturing a fuller semantic snapshot per chunk. This not only reduced the total number of nodes leading to more manageable computational requirements but also improved our ability to detect genuine anomalies. Clusters of sentences that formed a coherent thematic or linguistic unit were more likely to stand out when drastically different text was inserted.

In testing, we started with well-known literature such as *Alice’s Adventures in Wonderland* to see how the pipeline behaved before injecting any artificially constructed anomalies. We discovered that very short sentences like exclamations, single-word interjections, or partial quotes were frequently flagged, even though they did not necessarily represent meaningful anomalies. As a result, we introduced a filtering step to discard any sentences under 30 characters. This thresholding reduced spurious detections in the final anomaly list and helped ensure that each chunk had enough linguistic substance for the BERT embeddings to capture.

**2.5 Testing and Evaluation**  
To assess the effectiveness of our anomaly detection framework, we conducted extensive testing and evaluation across multiple datasets and parameter settings. Our validation process involved systematically inserting anomalies and analyzing their detection performance under varying conditions.

**2.6. Dataset Selection and Preprocessing**

We tested our approach using publicly available datasets, including books from Project Gutenberg, such as *Moby-Dick, Pride and Prejudice*, *The Greek New Testament* and *Quiet Flows The Don (Classical Russian literature)*. These texts were preprocessed to remove non-textual content, split into sentences, and grouped into fixed-size chunks for graph construction. To simulate real-world anomalies, we inserted predefined anomalous text segments in multiple languages and formats, including Hebrew, Greek, Russian, and gibberish-like sequences.

**2.7 Parameter Sensitivity and k-NN Graph Analysis**

We evaluated the impact of different *k* values (*e.g., k={3,5,7,9,15,21,30}*) and eventually a fully connected k-NN graph, observing how connectivity influenced anomaly detection. Lower *k* values provided localized structures, often highlighting anomalies but failing to amplify their significance. Higher *k* values increased graph density and improved contextual relationships, leading to better anomaly differentiation. The optimal *k* value was determined based on a balance between sensitivity to anomalies and overall graph coherence.

**2.8 Reflection and Iteration**

Throughout the development of this project, we encountered several challenges that required iterative refinement of our approach. Initially, our k-NN graph construction relied on simple syntactic ordering, which failed to capture the deeper semantic relationships necessary for effective anomaly detection. This led us to experiment with alternative graph structures, adjusting connectivity rules to balance local and global relationships. Additionally, the choice of k in the k-NN graph played a critical role in detection performance, requiring extensive testing to find an optimal setting.

Our initial evaluations highlighted issues with false positives, as many normal variations in text were mistakenly flagged as anomalies. This prompted us to refine our feature representation using more robust embedding techniques and experiment with different frequency thresholding methods in the Graph Fourier Transform (GFT). By iterating on these aspects, we improved anomaly detection accuracy, particularly in distinguishing meaningful anomalies from natural linguistic variations.

Overall, the iterative nature of our testing and refinement process underscored the importance of adapting both graph-based and spectral analysis techniques to the characteristics of the dataset. These reflections shaped our final methodology and provided insights for further improvements in future work.

**2.9 Tools Used**

* **Python 3.x**: Primary programming language.
* **Hugging Face Transformers**: For BERT model inference (bert-base-multilingual or similar).
* **NetworkX or PyTorch Geometric** (optional): Building and manipulating graphs.
* **NumPy, pandas, and SciPy**: Data manipulation, statistical analysis.
* **Matplotlib, Seaborn, or Plotly**: Data visualization, histograms, distribution plots.
* **NLTK, Beatifulsoup, re**: For text preprocessing and loading.

**3. Challenges and Solutions**

**High False Positive Rate**

* **Challenge:** Early tests resulted in a high number of false positives, where normal linguistic variations were incorrectly identified as anomalies.
* **Solution:** We refined the embedding process using BERT-based representations and adjusted the frequency thresholding in the Graph Fourier Transform (GFT) to improve precision.

**Selecting the Optimal k in k-NN Graph Construction**

* **Challenge:** Small k values highlighted anomalies but lacked strong differentiation, while larger k values captured too many contextual relationships, reducing anomaly visibility.
* **Solution:** Through extensive testing, we identified an optimal k range that balanced local and global relationships, improving anomaly detection without excessive complexity.

**Graph Connectivity and Structural Limitations**

* **Challenge:** The initial graph construction relied solely on syntactic ordering, which failed to capture meaningful semantic relationships.
* **Solution:** We transitioned to a semantic-based k-NN approach, allowing nodes to connect based on embedding similarities rather than strict text order.

**Computational Efficiency with Large Datasets**

* **Challenge:** Increasing the dataset size significantly impacted computational efficiency, particularly in eigen decomposition for spectral analysis.
* **Solution:** We optimized matrix operations and implemented dimensionality reduction techniques to make the process more scalable without sacrificing accuracy.

**Difficulty Detecting Subtle Anomalies**

* **Challenge:** Certain anomalies blended into the dataset due to their proximity in semantic space, making them difficult to distinguish.
* **Solution:** We adjusted the high-frequency filtering parameters in the spectral domain, refining the anomaly detection process to focus on meaningful deviations.

**4. Results and Conclusions**

Overall, our experiments demonstrated that the graph-based spectral anomaly detection pipeline does indeed succeed in identifying artificial anomalies, especially those that exhibit stark differences from the surrounding text. When we inserted foreign-language paragraphs, code snippets, or random gibberish into otherwise coherent documents, these segments consistently appeared among the top high-frequency amplitude nodes in the spectral decomposition. This outcome affirms the pipeline’s capacity to recognize semantic or stylistic breaks, lending credence to the core approach of embedding-based node signals and high-frequency detection.

However, we found that moderate or subtle anomalies such as slight topic shifts, stylistic changes, or smaller snippets of foreign text were not always flagged accurately. In some cases, these borderline changes blended in with naturally occurring variations in the text. Consequently, we observed a higher rate of false positives where legitimate yet stylistically unique passages (e.g., particularly short sentences, quotes, or abrupt dialogue shifts) were sometimes classified as outliers. This highlights one of the method’s current limitations: to reliably detect anomalies, they need to stand out quite starkly from the rest of the corpus.

**4.1 Hyper Parameters**

**BERT Model**: bert-base-multilingual-cased

**Batch Size** (for embedding generation): 8

**K Values**: k∈ {3,5,7,9,15,21,30}

**Node Signal Weights**: α, β varied in {(0.3,0.7),(0.7,0.3),(0.5,0.5)}

**Fraction-Based High-Frequency Cutoff**: *0.7×λmax*

**Top 10% Amplitude**: We consider the top 10% of high-frequency amplitude nodes as anomalies.

**Text Embedding Dimension:** 768 (for BERT embeddings)

**Laplacian Type:** Symmetric normalized

**4.2. "Pride and Prejudice" English dataset experiments results**

We tested our anomaly detection approach on the *Pride and Prejudice* dataset using different k values in the k-NN graph construction. We focused on three specific values k=3, k=15, and k=30 analyzing how varying neighborhood sizes influenced anomaly detection performance. Histograms were used to visualize the distribution of high-frequency amplitudes, allowing us to interpret how effectively anomalies were detected at each k value.

With **k=3**, the method performed poorly, failing to identify most of the injected anomalies. The small neighborhood size restricted the graph's connectivity, making the spectral analysis ineffective in distinguishing anomalous text from regular text. The limited contextual relationships led to weak frequency signals, preventing anomalies from being highlighted in the Fourier space.

A graph of different colored dots

AI-generated content may be incorrect.

Figure 3: Graph Histogram of the amplitude for each node for project Gutenberg's "Pride and Prejudice" English dataset built with k=3, alpha=0.5, beta=0.5.

Increasing to **k=15** produced significantly better results, successfully identifying most anomalies, with only one remaining undetected. At this k value, the graph structure captured a more meaningful semantic context around each node, allowing the frequency analysis to better distinguish anomalies from normal patterns. The histogram reflected a clearer separation between anomalous and non-anomalous nodes, demonstrating improved detection capability.

A graph of a number of dots

AI-generated content may be incorrect.

Figure 4: Graph Histogram of the amplitude for each node for project Gutenberg's "Pride and Prejudice" English dataset built with k=15, alpha=0.5, beta=0.5.

With **k=30**, the results were even more robust. Nearly all anomalies were detected in the top 10%, and the overall separation between normal and anomalous text became more pronounced. Higher k values beyond 30 did not introduce significant improvements, indicating a saturation point where additional connectivity no longer contributed to better detection. The reason for this is that as k increases, the graph becomes denser, and anomalies retain their distinct frequency patterns, but additional connections do not substantially alter their spectral characteristics.

In conclusion, the experiment demonstrated that choosing an optimal k value is crucial. Too low a k value limits the ability to capture meaningful relationships, while too high a k value leads to diminishing returns. In this dataset, **k=15 to k=30** provided the best balance between contextual awareness and anomaly separation, reinforcing the importance of tuning k based on dataset characteristics.

A graph of a graph with numbers and dots

AI-generated content may be incorrect.

Figure 5: Graph Histogram of the amplitude for each node for project Gutenberg's "Pride and Prejudice" English dataset built with k=30, alpha=0.5, beta=0.5.

**4.3. "The Greek New Testament" dataset experiments results**

We conducted the same anomaly detection experiment on the Greek New Testament dataset, this time using k=30 for the k-NN graph construction. The results were highly effective, with anomaly nodes displaying the highest amplitudes in the frequency domain. This suggests that the anomalies were distinctly separated from the normal text, making them easy to detect.

One possible explanation for these strong results is the nature of the dataset. Compared to *Pride and Prejudice*, the Greek New Testament is a smaller corpus, meaning that anomalies introduced into the text were more isolated. Additionally, the anomalies were fundamentally different from the original text both in language structure and content resulting in a stark contrast in the graph’s spectral representation. This contrast translated into a high-frequency response, which was clearly reflected in the amplitude histograms.

Overall, the k=30 setting worked exceptionally well, reinforcing the idea that anomaly detectability depends not only on k but also on dataset characteristics. In this case, a smaller dataset and larger linguistic differences between anomalies and the source text amplified their spectral distinction, making them easier to identify.

A graph of a graph showing different types of data

AI-generated content may be incorrect.

Figure 6: Graph Histogram of the amplitude for each node for "The Greek New Testament" dataset built with k=30, alpha=0.5, beta=0.5.

**4.4. "*And Quiet Flows the Don*" dataset experiments results**

For this experiment, we applied our anomaly detection pipeline to the Russian dataset *And Quiet Flows the Don*. Given the large size of the dataset, we set k=90 in the k-NN graph construction. A higher k value was necessary to ensure a dense connectivity structure, allowing the graph to effectively capture relationships within this vast and complex text.

Despite the well-connected graph, the results were not as strong as in previous experiments. While we did manage to detect some anomalies, their amplitudes were not as distinctly high as in smaller datasets like the Greek New Testament. One major reason for this is the richness and complexity of the text the novel is filled with varied linguistic styles, historical references, and intricate sentence structures. This makes it harder for spectral analysis to isolate anomalies, as the natural variations in writing style create a lot of noise in the frequency domain.

Another challenge was the scale of the dataset. With such a large corpus, even well-placed anomalies were somewhat diluted within the massive textual graph, making their spectral signature less pronounced. However, the method was still able to identify several anomalous nodes, demonstrating that even in large and complex datasets, graph-based spectral techniques can highlight unusual patterns, though with reduced precision.

A graph of a number of objects

AI-generated content may be incorrect.

Figure 7: Graph Histogram of the amplitude for each node for " And Quiet Flows the Don" dataset built with k=90, alpha=0.5, beta=0.5.

**4.5 Conclusions**

This study demonstrated the effectiveness of Graph Fourier Transform (GFT) for anomaly detection in text datasets, leveraging spectral analysis to identify linguistic irregularities. Through experiments on multiple datasets, we observed that k-NN graph construction and frequency analysis play crucial roles in detecting anomalies, with the choice of k significantly impacting performance. Lower k values often failed to distinguish anomalies, while higher k values improved detection by capturing broader contextual relationships.

Our findings suggest that dataset size and linguistic complexity greatly influence the success of the method. In smaller datasets, such as the Greek New Testament, anomalies were easily detectable due to their stark contrast with the original text. In contrast, larger and more complex datasets, such as *And Quiet Flows the Don*, posed challenges, as the rich linguistic diversity and vast corpus diluted the spectral signal of anomalies. However, even in these cases, the model successfully identified irregular patterns, proving its robustness across different textual structures.

While the results were promising, there is room for improvement in the pipeline. Enhancing the embeddings by using context-aware models like GPT-based embeddings or domain-specific transformers could significantly improve representation quality. Additionally, graph construction could be optimized by exploring dynamic k values instead of fixed ones, ensuring the graph better captures text dependencies. Other improvements include experimenting with different similarity metrics for edge formation (e.g., Jaccard similarity, dependency-based structures) and hybrid models that integrate both syntactic and semantic features. These enhancements could further refine anomaly detection, making it more effective across diverse textual datasets.

**4.6. Lessons Learned**

**Evaluation of Current Practices**  
Throughout this project, our reliance on BERT embeddings and graph-based spectral analysis proved both feasible and adaptable to multiple languages. Nonetheless, we found that the method’s performance can vary significantly depending on the dataset’s thematic scope and inherent diversity. Our experiences with moderate or subtle anomalies revealed that small shifts in style or topic could easily go undetected if they did not diverge enough from the baseline embedding space. This underscores the importance of carefully selecting (and sometimes curating) a dataset that matches the intended anomaly criteria, as well as the need to fine-tune our thresholding strategies for each domain.

**Identification of Strengths**  
Our primary advantage lies in the **flexible and modular** nature of the pipeline. By chunking sentences and leveraging pre-trained BERT models, we achieved a system that can handle multilingual inputs and still detect markedly out-of-place segments, especially those with obvious linguistic or topical dissimilarities. The k-NN graph approach further allowed us to explore different connectivity offering a valuable parameter for balancing granularity with computational expense. Moreover, the spectral decomposition phase gave us insights into global versus local disruptions in the text, making it easier to characterize anomalies in both a macro- and micro-level sense.

**Areas for Improvement**  
Despite these strengths, we identified several directions for refinement. First, false positives remain a challenge in high-diversity corpora, indicating that purely embedding-based adjacency may conflate naturally varied sections with genuine anomalies. Next, subtle anomalies particularly near-boundary or stylistic shifts could be missed unless we integrate additional features such as text-order adjacency, syntactic cues, or domain-specific embeddings. Finally, a more robust thresholding or multi-stage filtering process could help reduce noisy outlier detection, ensuring that anomaly scores align more closely with human judgments of textual irregularities. By addressing these concerns, we anticipate building a more precise and flexible system capable of handling an even wider array of real-world text anomaly cases.

**5. Project Benchmarks**

**Multilingual Coverage**

**Goal**: Ensure the anomaly detection pipeline handles text in multiple languages, including English, Hebrew, Greek, and Russian.

**Success Metric**: The system should reliably process, embed, and analyze text without errors in any of these languages.

**Scalability to Medium-Sized Corpora**

**Goal**: Efficiently process literature-scale texts (e.g., a few thousand chunks) within reasonable runtime constraints.

**Success Metric**: The pipeline (from embedding to anomaly listing) completes within a set time frame on standard hardware.

**Anomaly Detection Sensitivity**

**Goal**: Identify injected anomalies that deviate linguistically or thematically from the main corpus.

**Success Metric**: Accurately flag at least 80% of the artificially inserted anomalies in testing scenarios, while minimizing false positives.

**5.1 Achievements**

* **Multilingual Success**
  + We demonstrated that the pipeline can process **English, Russian, Hebrew, and Greek** texts. In each case, the BERT-based embeddings and k-NN graph construction handled the differing scripts without requiring separate tokenizers or models.
* **Handling Medium-Sized Texts**
  + We processed datasets like the *Greek New Testament* and *Alice in Wonderland*—ranging from hundreds to a few thousand chunks—within manageable time (typically under an hour on standard hardware). The batching approach for BERT embeddings proved sufficient for these corpus sizes.
* **Detecting Extreme Anomalies**
  + Our method reliably flagged large linguistic disruptions foreign-language inserts, gibberish, or code fragments demonstrating that stark deviations from the corpus norm consistently appeared in the top outlier nodes after spectral analysis.

In sum, while the pipeline excels at uncovering strongly out-of-context chunks in multiple languages, our experimentation reveals potential areas for refining detection sensitivity and reducing false positives particularly for subtler anomalies or highly variable texts. Nonetheless, we met our key benchmarks for multilingual support, scalability, and user documentation, laying a solid foundation for continued improvement and expansion.

**5.2 Testing**

To ensure the robustness and correctness of our anomaly detection pipeline, we conducted a series of tests that target key components of the system. Our primary goals were to **verify accuracy** in chunk and anomaly handling, **validate logic** in adjacency and spectral computations, and **confirm that the BERT-based embedding process** integrates correctly with the rest of the pipeline. The following categories outline the tests we performed:

1. **Data Preprocessing Tests**
   * **Chunk Creation**: We verified that the text was correctly split into chunks of the desired size (e.g., five sentences each), ensuring that short sentences below a chosen threshold (e.g., 30 characters) were filtered out properly.
   * **Anomaly Insertion**: Using small sample texts, we inserted multiple anomaly chunks and confirmed that the final node IDs matched our expectations. This involved checking that exactly one node was created per anomaly chunk and that all anomalies were discoverable in the final graph structure.
2. **Embedding Generation Tests**
   * **BERT Shape & Device Checks**: We tested whether the output of our generate\_bert\_embeddings function had the correct dimensions—one embedding vector per chunk, typically 768 dimensions for a base BERT model—and that it was consistently generated on the intended device (CPU or GPU).
   * **Robustness**: We also ran small-scale tests to confirm that truncation, padding, and batching were handled gracefully, with no discrepancies when splitting inputs into batches.
3. **Graph Construction Tests**
   * **k-NN Adjacency**: We created a set of synthetic embeddings (e.g., random vectors) to ensure that the k-nearest neighbors graph was constructed as expected, with each node having edges to its top-k similar neighbors according to cosine similarity.
   * **Edge Cases**: We tested graphs with very few nodes (e.g., fewer than k), verifying the pipeline handled these scenarios without errors and with a sensible number of edges.
4. **Spectral Analysis Tests**
   * **Laplacian Dimensions**: We confirmed that the normalized Laplacian was the correct size, matching the number of nodes in the final graph.
   * **Eigen-decomposition**: Using our test embeddings, we checked that the eigenvalue computation completed without errors and that the returned eigenvalues and eigenvectors were correctly indexed.
   * **High-Frequency Signal**: We performed smoke tests to ensure that the node signal projection onto high-frequency eigenvectors produced a vector of the correct length (i.e., one amplitude value per node).
5. **Integration & Workflow Tests**
   * **End-to-End Runs**: Finally, we performed a minimal end-to-end test, where a small text corpus plus anomalies was run through the entire pipeline. We validated whether the inserted anomalies appeared in the list of top-amplitude outliers, thus confirming that each major stage preprocessing, embedding, graph construction, and spectral analysis worked cohesively.

**6. User’s Guide Operating Instructions**

You can run the project on Google Colab notebook by clicking the link below: <https://colab.research.google.com/drive/1CypZbOudgdoR9gvJagvyBUOscCeXNsJX?usp=sharing>

There's a detailed explanation for every part of the pipeline including instructions for the user on how to run the pipeline.

**Maintenance Tips**

**Keep Dependencies Updated**: Regularly update the dependencies listed in google collab notebook.

**Consistent Coding Style**: Follow a consistent coding style throughout the project.

**Regular Testing**: Run unit tests regularly to catch any bugs or inconsistencies in the pipeline

**Dependency Issues**

If you encounter issues with dependencies, ensure that all required packages are installed and compatible with your Python version.

**Performance Issues**

Make sure when running the google colab project notebook the runtime type is set on any GPU as instructed in the notebook for best performance.

**7. Bibliography**

1. Verdoja, Francesco, and Marco Grangetto. "Graph Laplacian for image anomaly detection." Machine Vision and Applications 31.1 (2020): 11. arXiv:1802.09843

2. Comparative Analysis of Anomaly Detection Algorithms in Text Data (https://aclanthology.org/2023.ranlp-1.131) (Xu et al., RANLP 2023)

3. Akoglu, L., Tong, H. & Koutra, D. Graph based anomaly detection and description: a survey. Data Min Knowl Disc 29, 626–688 (2015). <https://doi.org/10.1007/s10618-014-0365-y>

4. Sawada, Ryosuke & Sakumoto, Yusuke & Takano, Chisa & Aida, Masaki. (2016). Experimental study on relationship between indices of network structure and spectral distribution of graphs.