RES MILITARIS
REVUE EUROPEENNE D ETUDES EUROPEAN JOURNAL OF MILITARY STUDIES
Social Science Journal

A New Method for Graph-Based Representation of Text in Natural Language Processing

Ishrat Ahesan Gawandi
Assistant Professor
Department of Mathematics
Sonopant Dandekar Arts, V. S. Apte Commerce and M. H. Mehta Science College,
Palghar, Maharashtra, India
Husnawaz Contractor
Assistant Professor
Department of Mathematics
Sonopant Dandekar Arts, V. S. Apte Commerce and M. H. Mehta Science College,
Palghar, Maharashtra, India

1.Abstract
Natural language processing is a field within machine literacy that's still in its early stages. The need for easy communication with operating systems, new operations for artificial intelligence, and increased access to textual data sets all have an impact on how important natural language processing is to the development of artificial intelligence. The traditional textual representation ways like Bag- of- Words have failings, as they do not consider word dependences and semantics. To enable a more suggestive textual representation, we suggest a new system grounded on graph representations that considers both original environment and global relations between words. The exploration aims to show the operation of graph representations in textbook bracket and probe their implicit operations in natural language processing. Our suggested system builds a point vector by using common sets in graphs that represent documents, which is new. Tests corroborate that it can increase bracket effectiveness. The delicacy, perfection, recall, and F1- score of our new textbook representation approach that was used to read book orders grounded on content analysis was further than 90. Switching from conventional styles to a graph- grounded approach in natural language processing and textbook analysis may have a significant impact and produce new openings.

2.Introduction
I fully agree that having an effective textbook representation is pivotal for natural language processing operations similar as textbook bracket. Although conventional models like Bag- of-Words and T- IDF are extensively used, they frequently ignore word connections and semantics, leading to significant information loss. In recent times, the emergence of graph representations has handed a new way to model textbook semantics and achieve further suggestive textual representations. By using graphs, we can support both global and original word dependences, which has led to significant advancements in textbook analysis and processing. still, it's important to rightly interpret and display textbook before rooting any

knowledge from it. Graph ways view words as bumps and produce edges between them via the participated- word fashion. I am interested in exploring how graph representations can be used for textbook analysis in a real- world book content bracket challenge. With the adding vacuity of both electronic and print books, automatic book bracket could be useful for digital depositories, bookstores, and libraries.Classifying a book into the right order is pivotal to make it easier for people to detect it. Bookstores and librarians can complete their tasks much more snappily when they know which shelf to place a book on, and compendiums can find their asked kidney much more snappily. Proper bracket of books can also profit authors as it would be easier for them to detect workshop that are competitive with their own. These reasons led us to test our methodology on a library of books.

Our exploration aims to explore a new system of textbook representation grounded on a new graph representation and demonstrate its use in textbook categorization. Our approach examines the practicality of the textbook representation fashion bandied before, which is also applied to train bracket machine literacy models. Our system is innovative in that it identifies frequent sets in graphs that represent documents and builds point vectors around them. We aim to estimate the bracket criteria (delicacy, perfection, recall, and F1- score) grounded on the textbook representation used by assaying the matrices attained from our proposed textbook representation approach. assaying categorization measures is done to see if the issues are similar to those of further conventional ways. We prognosticate that the suggested strategy, which makes use of graph representations, will lead to new developments in the field of natural language processing and ameliorate textbook categorization effectiveness.
To add up, the primary benefactions we've made and participated in this work are as follows

• We propose an innovative approach grounded on graph representations, taking into account both the original environment and global connections between words — graphs are a structural reflection of the textbook, which can give new perceptivity and information that aren't taken into account in traditional styles of textbook representation;
• We gain and process real long textbooks, similar as entire books to expand the possibilities of textbook analysis and representation, which were preliminarily limited to short fractions;
• We estimate the efficacity of textbook bracket by comparing the efficacity of textbook bracket grounded on a graph representation with the issues of traditional styles. We use bracket measures similar as delicacy, perfection, recall, and F1- score to assess the new system's effectiveness compared to being approaches.
• We identify and use common sets of words in graphs representing documents, which is a new perspective in textbook analysis. sets are collections of words that do together in environment and can have semantic meaning.
Our platoon has made significant benefactions to the field of wisdom by developing a methodology for textbook representation using graphs. Specifically, we've created a unique approach for applying sets in natural language processing, which is effective when compared to conventional ways. We aim to ameliorate textbook bracket effectiveness and produce new openings by incorporating graph representations into natural language processing.

This document is structured as follows Section 1 provides an preface to the content of our study, while Section 2 reviews applicable workshop on textbook categorization, NLP ways, and graph representation of textbook. In Section 3, we bandy the necessary theoretical background. Our approach for classifying books using a graph representation is presented in

Section 4, along with the visualization tools we've developed to explore the relations among different rudiments in book textbooks. Our trials and findings are presented in Section 5, and in Section 6, we offer some general compliances about our work and suggest avenues for unborn study.

Associated Works

Natural language processing(NLP) is a complex field of artificial intelligence that utilizes computational styles to understand and induce language content. NLP can dissect any form of language, whether spoken or written, for mortal communication purposes. Despite being a fairly new discipline, practical operations of NLP have been the subject of multitudinous exploration papers, including textbook summarization.

In one study, B. Sharifi et al. presented algorithms for summarizing Twitter microblog entries. The authors used a graph to process groups of brief messages, producing short summaries that are comparable to those created by humans. M.A. Mosa et al. also utilized graphs to summarize brief texts, with their use of graph coloring proving successful. The authors suggested a method for summarizing comments that incorporated Jensen-Shannon divergence and ant colony optimization. When tested against traditional document summary algorithms, their system performed exceptionally well on a set of Facebook posts and accompanying comments.

In contrast, R.Y. Rumagit et al. conducted a study to determine the most effective technique for summarizing text by comparing the graph-based approach and the phrase-weighting method. The results showed that the term-weighting method outperformed the graph-based approach. Additionally, S.A. Crossley et al. developed an automatic abstract scoring model that accurately categorized summaries as poor or high quality with over 80% accuracy. Meanwhile, B. Liang et al. demonstrated that a model using a graph convolutional network could surpass state-of-the-art techniques on various public datasets. Furthermore, studies have compared graph and vector techniques to text abstraction and found that graph-based systems perform better. For instance, S.M. Ali et al. utilized the KeyGraph technique to identify keywords and entities of tweets based on the vertices of their graph.

A keyword-based summary was effectively generated, followed by sentiment analysis and a graph representation. The authors assigned weight to the graph's edges, revealing a resemblance in the graph. A superior abstract quality was found in the suggested method compared to other text summary strategies. The graph representation also proved successful in text classification, creating homogeneous text graphs with word nodes that allowed for conclusions to be drawn about new documents. The authors demonstrated the superiority of their method using experimental findings on five benchmarks. BERT models supported automatic subject indexing for digital library collections and successfully assigned books to the appropriate category. T. Betts et al. extracted structured information from book texts to study information extraction approaches in a text categorization challenge. Although text classification issues are not new, they remain pertinent given the volume of publications on the subject. Our representation is also thought to have potential applications in machine learning techniques that work with vectors, such as text clustering, filtering, or summarizing, all while maintaining the document structure.

3. Historical Context

The advancement of information technology has made it possible to store enormous volumes of texts digitally and to employ cutting-edge methods for natural language processing. Millions of individuals worldwide now have the chance to utilize the wealth of literary materials without needing to physically access the books thanks to the digitization process of books. Texts in electronic format and NLP techniques facilitate easy sharing, searching, and processing of the information they contain.

Furthermore, the digitalization of books through Project Gutenberg has presented significant difficulties in handling and interpreting vast volumes of text. As a result, text graph representation and natural language processing are becoming more and more crucial for efficient text analysis and information extraction. As a result, we have included here the primary theoretical facets—natural language processing, text representation in graphs, and book classification—that are pertinent to the Gutenberg project.

3.1. Gutenberg Project

Discover a wealth of free eBooks at the Gutenberg Project library, available online at https://www.gutenberg.org/ (accessed on May 12, 2023). Established by Michael Hart in 1971, Project Gutenberg was the pioneering organization to make eBooks freely accessible. Initially holding around 300 publications in 1987, the library has now grown to include over 50,000 titles. The Project's key aim was to digitize books and publications whose copyrights had lapsed or never existed. It strives to encourage everyone with an interest to write and distribute eBooks.

3.2. The Interpretation of Natural Language

Natural language processing (NLP) is a field that merges the expertise of linguistics, computer science, and mathematics to understand language structure and fundamentals. NLP is a versatile tool that can be applied to both spoken and written language in all human languages. It can be used for automatic text or speech translation, spam and message filtering, grammar and word checking, mood analysis, and document summarization. Some NLP programs can even suggest responses to messages based on their content. NLP is a crucial component of Project Gutenberg's ability to analyze and process vast collections of text, including books, articles, and other materials.

Text pre-processing is an essential part of any NLP system, as the letters, words, and sentences identified at this stage serve as the foundation for later processing stages. Since text data often contains specific forms like dates, numbers, and common word formats, it is important to handle them correctly to ensure complete process support. Tokenization is a pre-processing technique that involves breaking down the text analysis into tokens, such as sentences, phrases, and word fragments. Additionally, during the actual text data pre-processing, a normalization procedure can be carried out on each word in the text. Common normalizing techniques in NLP include lemmatization and stemming.

Stemming involves identifying the central part of a word that remains constant despite changes in case, number, or person-based inflection. The objective of stemming is to remove the

inflection ending while preserving the unchanging portion, which often happens to be a fragment of a recognized term already in the dictionary.

Lemmatization is often confused with stemming since the two methods may produce the same outcomes. However, lemmatization involves extracting a word's basic form (known as the lemma), which remains unaffected by any circumstances, persons, or forms. The lemmatizer requires dictionary usage to function effectively, enabling it to link multiple forms of the same word.

3.3. Representation of Text

Effective text representation is crucial in the world of natural language, as it allows computers to process and analyze language with ease. Depending on the analysis's context and goal, text representation can take on various forms.

One fundamental text representation technique is stepwise representation, which involves segmenting text into discrete units, such as words or letters, and processing them sequentially. This technique is commonly used in lexical analysis, where words are the basic units, and the analysis aims to identify and assign appropriate labels to each word, such as a part of speech.

In Natural Language Processing (NLP), statistical techniques such as word frequency counting are often used for text analysis. These techniques provide the necessary variables for string processing by representing the document as a set of term vectors, with term weights serving as the vector's values. Three-word weight measures can be used for this representation: TF, T-IDF, and a binary measure. These text representations have been effectively utilized in machine learning techniques to categorize text and unlock its true potential.

The word embedding model, which is another widely used NLP technique, involves representing words or text entities in a vector space to capture the syntactic and semantic relationships between words. This makes it possible for machine learning models to comprehend word meaning and context more effectively and efficiently. These methods are predicated on the idea that words typically have comparable meanings when used in similar settings. To achieve this, word embedding models like Word2Vec, GloVe, and FastText are trained to identify words with comparable meanings or usage patterns and assign corresponding vectors to them. Accordingly, words with comparable syntactic or semantic characteristics have comparable vector representations, and their relationships are reflected in the distances between their embeddings. Words like "king" and "queen," for instance, need to be closer in embedding space and have similar vector representations than words like "dog" or "cat." On the other hand, we can obtain the vectors of other words by applying vector arithmetic, which involves vector addition or subtraction. For instance, we may estimate the vector representation of the word "queen" by subtracting the vector representation of "male" from "king" and adding "female."

Word embedding models rely on large text corpora to function, but they struggle with uncommon terms that are infrequent in the training data. These models also face difficulty distinguishing between different meanings of ambiguous words, as multiple meanings are often incorporated into a single embedding. Additionally, since word embedding models only consider the words themselves and not the context in which they appear, two identical words

in different contexts may be assigned the same embedding, resulting in a loss of semantic information.

However, some mind-based models employ word embedding as a common input representation. By utilizing attention mechanisms to focus on relevant segments of the input sequence during training, these models learn to comprehend the semantics and contextual links between words. This leads to improved prediction accuracy and the generation of useful output.

3.4. Representation of Graphs

In the terrain of Project Gutenberg, the examination of handbooks can be enhanced through the operation of a graph representation. This type of representation involves the use of syntactic or semantic connections between words, rulings, or documents as edges, and words, rulings, or documents themselves as vertices. By exercising this graph structure, one can engage in semantic analysis of the text, identify patterns or thematic correlations, and use advanced graph analysis tools to gain perceptivity from the vast amount of textual data collected by Project Gutenberg. Graph proposition is predicated on the fundamental generality of a graph, which is a fine structure that represents relations between objects. A graph consists of a set of vertices, denoted as V, that can be connected by edges, represented by E so that each edge starts and ends at a vertex. However, the graph is considered complete, If there is an edge connecting every brace of vertices. also, specific labels can be applied to the vertices to produce a labeled graph. The bumps and edges of a graph can represent various language- related particulars and associations, depending on how natural language processing ways are employed.

The whole scrap of the G graph that is not included in any major full subgraph of G is known as the C crowd (i.e., C is a maximum full subgraph of G). The Bron- Kerbosch Algorithm was first forth in 1973. It was erected to descry every minimum crowd in an undirected graph. All subsets of vertices having two attributes are listed. firstly, an edge connects every brace of vertices in each of the below groups. Second, it's impossible to add new vertices to any of the listed subsets and still have full connectedness.

Graph- predicated approaches concentrate on how to use the rates of text handbooks by representing them as graphs. Connections between various textual rudiments, analogous as realities, rulings, and documents, can be recorded using a graph. Studies on the marriage of graph representation and machine knowledge have demonstrated the superiority of semi-supervised graph- predicated knowledge over semi- supervised Bag- of- Words knowledge.

The graph- predicated representation of words can store and use more complex semantic information, including synonyms, antonyms, hyperonyms, hyponyms, and other connections. This type of representation is more effective than traditional word representations in directly conveying the meaning of words in terrain. By reflecting connections between words along various confines, graphs can serve as important tools for comprehending the terrain and syntax of a judgment. In addition, a graph- predicated representation is more complete at handling word ambiguity. Understanding the connections between words in the graph and their terrain helps one discern between a word's multiple meanings and choose the applicable bone in a given situation. In uncertain situations, a graph- predicated representation can give lower strictness and be more salutary.

Compare:

Text type is a pivotal aspect of natural language processing, and it plays a significant part in grading books within the Project Gutenberg frame. Machine knowledge heavily relies on type as well. principally, the type model is responsible for assigning objects from the set to their corresponding classes. various classical type models can effectively handle textual data. One of these models is the Bracket and Retrogression Trees (wagon) algorithm, which is used in constructing decision trees. The algorithm divides the bumps of a decision tree into sub-nodes predicated on the threshold value of an particularity. Using the swish particularity and threshold value, the root knot is resolve into two subsets as the training set. The same principle applies to the posterior subsets until either the last pure subset is reached or the maximum number of leaves allowed in the tree is reached.

Breiman (1996) also proposed the Bagging algorithm. An array of different classifiers is called a bag. There are three main way in this algorithm Bootstrapping (creating a variety of samples by etesting the training dataset and concluding data points at arbitrary and with relief); similar training (bootstrapping samples are trained independently and in resemblant with each other using weak or base trainers); and Aggregation (ultimately, there is a maturity vote).

Breiman put forth the arbitrary timber refinement in 2001. multitudinous independent decision trees work together to form a arbitrary timber. The class that receives the most votes becomes the model's prophecy, and each tree in the arbitrary timber produces a class prophecy.

The AdaBoost classifier is a volition to the algorithms that have been mooted. Its thing is to produce a single strong classifier by combining several weak classifiers. While a single classifier might not be suitable to correctly read an object's class, a important model can be constructed by grouping numerous weak classifiers, each of which gradually gains knowledge from the particulars that the others have erroneously classified.

Support Vector Machine is a fresh algorithm (SVM). SVM divides samples into multiple classes by erecting hyperplanes in multidimensional space to carry out type tasks. The number of features determines the hyperplane's dimension. The hyperplane is principally a line if there are just two input features. The hyperplane transforms into a two- dimensional airplane if there are three input features. However, it gets hard to imagine, if there are further than three features.

New developments in the field of text type have been observed recently. piecemeal from exercising sophisticated models like Mills, two truly encouraging styles calculate on cooperation and the architectures of Convolutional Neural Networks (CNN) and crossbred Long Short- Term Memory (LSTM). These number adding up the vaticinations from numerous models trained on identical input. Every model in the assembly can have a unique architecture and hyperparameter setup. The issues of each model's prophecy are combined to produce the final categorization. By exercising a variety of model prognostications, this system seeks to enhance overall type performance.

Using deep knowledge algorithms to identify irony and propagate generalizations is one contemporary system. The suggested model had five turns, a maximum delicacy of 0.92, and an average delicacy of 0.89. It was predicated on a data addition (DA) caste and a convolutional

neural network (CNN). Classical machine learning techniques have also been applied to solve the irony detection problem. The Bag-of-Words sequence was created by the authors of the research using a text vectorization layer, and it was subsequently fed to three distinct text classifiers (naive Bayes, convolutional neural network, and decision tree).

I recently read that Support Vector Machine (SVM) was chosen as the final classifier, and it achieved an impressive binary accuracy of 0.9474 using the suggested strategy on the best validation division. Also, Natural Language Processing (NLP) algorithms are playing a significant role in detecting fake news and sarcasm. However, identifying sarcasm in text can be challenging for these algorithms as it requires deciphering the hidden meaning and intention behind the utterance. To detect sarcasm, NLP models use techniques like sentiment analysis, contradiction detection, and context analysis. Additionally, these models can evaluate the veracity of the source, examine the sentence structure, look for inconsistencies, and evaluate content about other messages to determine the accuracy of information. The outcomes of these studies show how creative methods based on CNN, Hybrid LSTM, or Transformers networks can be quite beneficial. However, because of their stability, interpretability, and reduced computational and resource needs, traditional machine-learning techniques are still useful in the context of text classification.

4. Research Methodology

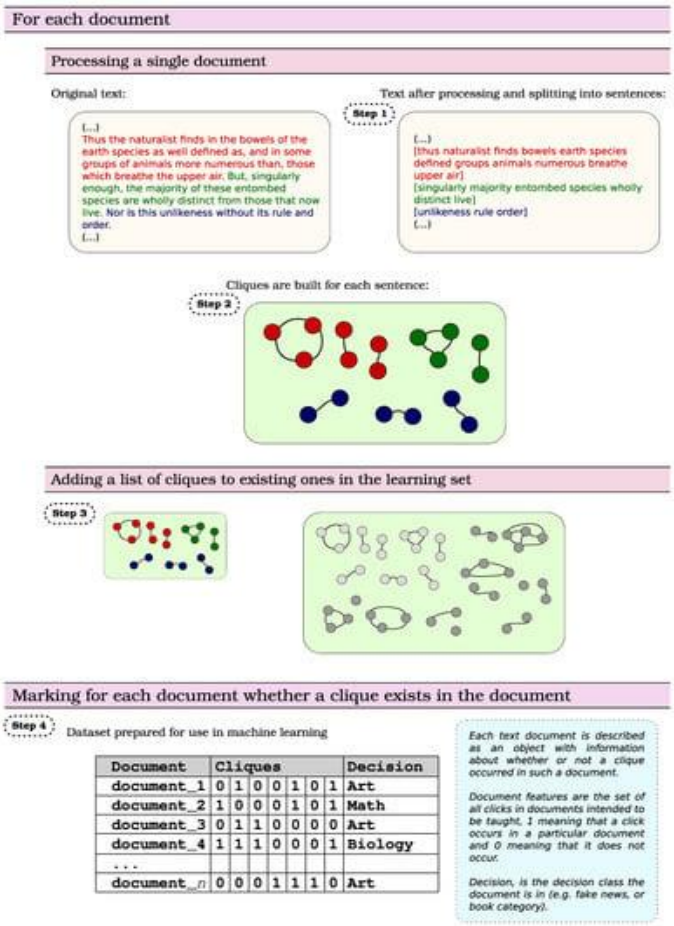
Our research was initiated to explore new ways of representing text that account for the complex interplay between words. Natural language processing heavily relies on how text is depicted, so we wanted to investigate an alternative method that uses graph structures to capture both local context and broader word relationships within documents. By establishing a more suitable framework for text representation, we can improve text classification, text clustering, and summarization processes. Our innovative approach lies in identifying common cliques within document graphs as a foundation for constructing feature vectors. This approach, which considers word relationships, holds promise for yielding more nuanced textual representations and advancing semantic analysis.

To evaluate the efficacy of our graph-based text representation against conventional methods, we conducted experiments that aimed to match books to appropriate categories to streamline reader access, providing a potential solution for libraries to enhance book classification accuracy.

Our research methodology involved exploring a fresh perspective on text representation, utilizing an innovative graph-based approach that incorporates word-based cliques. We focused on assessing the utility of the proposed text representation and leveraging it to train machine-learning models for classification purposes. Our primary objective was to demonstrate the practical application of this method in text classification, particularly in the context of categorizing books based on content. A key aspect of our approach involved identifying common cliques within document graphs and generating feature vectors based on them. We evaluated the effectiveness of this text representation method within the context of book classification and compared it with traditional representations. Our methodology revolves around text processing and its representation.

Figure 1 illustrates the algorithmic flow of our proposed approach. The initial step involved pre-processing document content, which included removing stop words and diacritical marks, and applying word normalization techniques such as lemmatization. We also eliminated punctuation marks outside of periods as they served as demarcations for sentence boundaries, which were essential for preserving word order within each sentence. However, the occurrence of periods posed challenges as they disrupted the formation of edges between words in the graph, resulting in inconsistencies.

In Figure 1, we present an algorithmic diagram for our suggested approach. The initial stage of our method (Step 1 in Figure 1) involved pre-processing the document content. To accomplish this, we removed stop words and diacritical signs and used lemmatization techniques to normalize the words. In addition, we eliminated punctuation marks outside of the dots to prepare for the next phase. In Step 2, we represented the text as a labeled graph. The graph's vertices were the words that appeared in the text, but each word appeared only once. Any two words that appeared next to each other in a phrase were connected by an edge in the graph. We used periods to divide the sentences in the text and maintain the word order in each sentence. However, the presence of a dot sometimes prevented an edge between two words in the graph. As a result, the graphs we obtained were inconsistent.



Document	Cliques							Decision
document_1	0	1	0	0	1	0	1	Art
document_2	1	0	0	0	1	0	1	Math
document_3	0	1	1	0	0	0	0	Art
document_4	1	1	1	0	0	0	1	Biology
...								
document_n	0	0	0	1	1	1	0	Art

Figure 1. Diagram of the algorithm of the proposed approach.

Throughout our investigation, there were instances where a word was surrounded by dots, usually in website addresses, creating an inconsistent graph with only one node. To identify

every potential clique, we applied the Bron-Kerbosch algorithm to the network created by these instances, as seen in Step 3 of Figure 1. We also discovered frequent subgraphs with only one element. This method produced a collection of cliques and one-element subgraphs.

In Step 4 of Figure 1, we created a binary matrix that showed whether a specific set appeared in the book's text. We used "1" to indicate that it appeared and "0" if it didn't. We incorporated this finding into our method because our prior research has shown that term weighting (TF) yields better results than binary. Additionally, we produced matrices that weighted the frequency of occurrence of each element in the set. To count the instances of a specific element, we used three methods.

The term "maximum" refers to the highest number of times two elements appear together in the text among all the clique elements. The term "minimum" refers to the lowest number of times two elements appear together in the text among all the clique elements. The term "sum" refers to the total number of times two elements appear together in the text among all the clique elements.

It is important to consider the occurrence of two clique elements, as it may not always be the case that multiple words occur together in a phrase. However, for two words to be connected by an edge in the graph, they must occur together at least once. We determined the weight of a graph with a single vertex by counting how many times a particular word appeared in the text.

Additionally, we observed two important points. Firstly, we examined the vector that contained the common components in the set across all documents, as well as the unique elements specific to the class. Secondly, we aimed to determine whether the method of identifying common items in the matrix could potentially affect the resulting categorization outcomes. We established that common elements were solely within the class, not across all documents.

Additionally, we examined our method for generating vectors based on the inclusion of set elements, including cases of single-element subgraphs such as "all" (every piece in the set) and "clique" (groups of more than two objects). To create vectors, we utilized AdaBoost, Bagging, CART, Random Forest, and SVM techniques.

5. Experiments and Results

We conducted tests to evaluate the effectiveness of our proposed method using an actual dataset from Project Gutenberg books, as mentioned in Section 3.1. We aimed to determine how well our approach performed in classification based on quality metrics such as F1-score, accuracy, precision, and recall. We created a dataset with book material classified into distinct categories for this purpose. We chose two categories, "Philosophy" and "Technology," from Project Gutenberg to test our method. These categories were picked based on prominent scientific subjects and currents. Our dataset consisted of 320 books, with 218 books in the "Technology" category and 102 books in the "Philosophy" category. We processed the text of each book using the procedures outlined in Section 4. When compared to the algorithm employed in the traditional approach, our findings showed that the content of the dataset significantly influenced the classification outcomes. However, the novel text representation approach fared slightly worse than the traditional method in terms of classification results.

5.1. Text Representation

We conducted a comparison of our text representation with the traditional vector technique which includes binary, TF, and T-IDF. The traditional approach resulted in matrices with 63,280 features. However, we were able to obtain 112,869 cliques of features in the case of common members of the set within the class. Additionally, 140,465 was the matrix formed by the set's common elements from all of the documents.

Table 1. A list of the number of elements in each set.

Number of Elements in Set	**Number of Common Elements in Classes**	**Number of Common Elements in All Documents**
1 element	1469 single-elements	1757 single-elements
2 elements	104,383 cliques	130,997 cliques
3 elements	6294 cliques	6983 cliques
4 elements	643 cliques	648 cliques
5 elements	74 cliques	74 cliques
6 elements	5 cliques	5 cliques
7 elements	2 cliques	2 cliques

5.2 Classification

We used two sets of books to build our classifiers: a training set with 70% of the total books and a test set with 30% of the books. Our classification results are displayed for the following metrics: accuracy (Table 2), precision (Table 3), recall (Table 4), and F1-score (Table 5).

Table 2. Classification Table for measuring the Accuracy

	AdaBoost	**Bagging**	**CART**	**Random Forest**	**SVM**
classic_Binary	0.97	0.96	0.91	0.96	0.97
classic_TF	0.98	0.94	0.90	0.95	0.90
classic_TF-IDF	0.97	0.94	0.89	0.96	0.95
binary_all_true	**0.95**	**0.94**	0.87	0.87	0.79
binary_all_false	**0.95**	**0.90**	0.87	0.85	0.76
binary_clique_true	0.75	0.72	0.73	0.72	0.71
binary_clique_false	0.75	0.73	0.75	0.72	0.70
sum_all_true	**0.95**	**0.91**	0.88	0.86	0.83
sum_all_false	**0.95**	**0.93**	0.88	0.86	0.84

sum_clique_true	0.75	0.73	0.75	0.72	0.68
sum_clique_false	0.75	0.72	0.75	0.71	0.68
max_all_true	**0.95**	**0.92**	0.88	0.87	0.83
max_all_false	**0.95**	**0.93**	0.87	0.86	0.84
max_clique_true	0.76	0.73	0.74	0.72	0.68
max_clique_false	0.75	0.73	0.74	0.71	0.68
min_all_true	**0.95**	**0.92**	0.87	0.89	0.83
min_all_false	**0.95**	**0.91**	0.88	0.87	0.83
min_clique_true	0.74	0.73	0.73	0.72	0.72
min_clique_false	0.75	0.73	0.74	0.71	0.72

Table 3. Classification Table for measuring the precision

	AdaBoost	**Bagging**	**CART**	**Random Forest**	**SVM**
classic_Binary	0.98	0.95	0.90	0.96	0.98
classic_TF	0.98	0.94	0.90	0.96	0.93
classic_TF-IDF	0.97	0.94	0.88	0.97	0.96
binary_all_true	**0.95**	**0.93**	0.86	**0.92**	0.88
binary_all_false	**0.95**	0.89	0.86	**0.91**	0.87
binary_clique_true	0.81	0.79	0.75	0.82	0.85
binary_clique_false	0.80	0.81	0.78	0.82	0.85
sum_all_true	**0.95**	**0.90**	0.87	**0.90**	0.87
sum_all_false	**0.95**	**0.91**	0.86	**0.90**	0.87
sum_clique_true	0.82	0.82	0.82	0.82	0.76
sum_clique_false	0.81	0.81	0.80	0.81	0.76
max_all_true	**0.95**	**0.91**	0.86	**0.91**	0.87
max_all_false	**0.95**	**0.92**	0.86	0.89	0.87
max_clique_true	0.83	0.86	0.79	0.84	0.84
max_clique_false	0.83	0.85	0.81	0.84	0.84
min_all_true	**0.95**	**0.91**	0.86	**0.92**	0.87
min_all_false	**0.95**	**0.90**	0.86	**0.91**	0.87
min_clique_true	0.79	0.81	0.76	0.82	0.81

min_clique_false	0.81	0.81	0.78	0.81	0.81

Table 4. Classification results for the measure recall.

	AdaBoost	**Bagging**	**CART**	**Random Forest**	**SVM**
classic_Binary	0.96	0.96	0.90	0.94	0.96
classic_TF	0.97	0.93	0.87	0.93	0.85
classic_TF-IDF	0.96	0.93	0.87	0.95	0.92
binary_all_true	**0.94**	**0.92**	0.84	0.80	0.67
binary_all_false	**0.94**	0.88	0.84	0.77	0.63
binary_clique_true	0.63	0.58	0.61	0.57	0.55
binary_clique_false	0.62	0.58	0.63	0.57	0.55
sum_all_true	**0.93**	0.89	0.87	0.80	0.76
sum_all_false	**0.93**	**0.92**	0.85	0.79	0.76
sum_clique_true	0.63	0.60	0.63	0.57	0.52
sum_clique_false	0.63	0.58	0.62	0.57	0.52
max_all_true	**0.94**	**0.91**	0.85	0.80	0.76
max_all_false	**0.94**	**0.92**	0.85	0.79	0.76
max_clique_true	0.63	0.58	0.62	0.57	0.51
max_clique_false	0.62	0.58	0.62	0.56	0.51
min_all_true	**0.94**	**0.91**	0.85	0.83	0.76
min_all_false	**0.94**	**0.90**	0.86	0.81	0.76
min_clique_true	0.62	0.60	0.61	0.57	0.57
min_clique_false	0.63	0.58	0.61	0.57	0.57

Table 5. Classification table for the measuring the F1 score.

	AdaBoost	**Bagging**	**CART**	**Random Forest**	**SVM**
classic_Binary	0.97	0.96	0.90	0.95	0.97
classic_TF	0.97	0.94	0.88	0.94	0.87
classic_TF-IDF	0.96	0.94	0.88	0.96	0.94

binary_all_true	**0.95**	**0.93**	0.85	0.83	0.69
binary_all_false	**0.95**	0.88	0.84	0.80	0.63
binary_clique_true	0.63	0.56	0.60	0.54	0.51
binary_clique_false	0.63	0.56	0.63	0.55	0.50
sum_all_true	**0.94**	**0.90**	0.87	0.83	0.78
sum_all_false	**0.94**	**0.92**	0.86	0.82	0.79
sum_clique_true	0.63	0.58	0.63	0.54	0.44
sum_clique_false	0.63	0.56	0.62	0.53	0.44
max_all_true	**0.95**	**0.91**	0.86	0.83	0.78
max_all_false	**0.95**	**0.92**	0.85	0.82	0.79
max_clique_true	0.63	0.56	0.62	0.54	0.42
max_clique_false	0.62	0.56	0.61	0.52	0.42
min_all_true	**0.94**	**0.91**	0.85	0.86	0.78
min_all_false	**0.94**	**0.90**	0.86	0.84	0.78
min_clique_true	0.62	0.58	0.61	0.54	0.54
min_clique_false	0.63	0.56	0.61	0.53	0.54

The best results were achieved using conventional procedures for each of the measures. AdaBoost and the TF measure showed the best overall results in terms of accuracy and recall. AdaBoost also scored the highest precision scores for the TF and Binary measurements. The two models with the highest F1 score were AdaBoost using a TF or Binary measure and SVM using a Binary measure.

When it comes to the technique proposed, the strategy that considered all components of the set (cliques and one-element graphs) and the AdaBoost algorithm produced results closest to the traditional approaches for each metric. Additionally, the Bagging method and every technique that included every element in the set received scores higher than 90% (refer to Table 2 for details).

After analyzing the accuracy measure presented in Table 2, it was observed that the SVM algorithm performed the worst when paired with the strategy that considers cliques greater than two elements. Conversely, the AdaBoost and Bagging algorithms performed the best when working together with the strategy that included every component of the set. The accuracy results for different clique weights (sum, max, and min) were similar, with all of them ranging from 90% to 95%. This suggests that using more cliques in this specific situation may result in less accurate classification.

In Tables 3 and 4, AdaBoost and Bagging both achieved scores above 90% for the precision and recall metrics. Additionally, the Random Forest algorithm, when using every element in the set, scored higher than 90% for the precision metric. High precision indicates that the

system can correctly identify affirmative examples in the categorization, which is a significant observation. However, the CART algorithm combined with a binary approach, which considered common cliques greater than two elements in the class, produced the worst results. On the other hand, the SVM algorithm combined with the max approach, which considered cliques larger than two-element size, produced the worst results for the recall measure. This indicates that the precision and recall of the algorithms were reduced since they were unable to accurately identify all affirmative cases for these combinations.

By examining the F1-score measure presented in Table 5, it was observed that the SVM algorithm, when combined with the max method, produced the worst results. This strategy considered cliques that had more than two elements, indicating that the algorithm had difficulty achieving high recall and precision for this specific combination, resulting in a lower F1 score.

The study found that the AdaBoost and Bagging algorithms, when used in combination with a method that considers every element of the set with various clique weights (sum, max, min), produced the best F1-score results. These algorithms were able to achieve high recall and precision simultaneously, which increased the F1 score.

These findings suggest that the AdaBoost and Bagging algorithms can be more effective in classifying text and producing a higher F1 score when used with a diverse range of dataset elements and clique weights. By dynamically adjusting the influence of different-sized cliques on the classification score using varying clique weights, a better balance between recall and precision can be achieved.

The results presented in Tables 2, 3, 4, and 5 demonstrate the effectiveness of strategies that utilize all of the dataset's information. The inclusion of cliques and single-element graphs provides a more comprehensive depiction of the data and a deeper understanding of its structure. Additionally, good results were also obtained from algorithms like AdaBoost and Bagging, which combine multiple models to improve overall classification performance.

If a score is more than 90%, it means that our methods are comparable to traditional categorization techniques. This also opens the door to further research and development to improve these techniques and apply them in other fields where categorization is necessary. Our research has concluded that using all the components of the dataset and appropriate classification methods can lead to higher scores and better performance in text classification tasks.

6. Conclusions

This research aimed to analyze the text representation approach we proposed. Our suggested text representation approach was predicated on a graph representation, from which we extracted a set of cliques and single-element graphs. The matrix was constructed from a vector that was formed based on the set. Furthermore, we employed our text representation to carry out a classification and verified if the outcomes aligned with those obtained with the conventional text representation. To categorize the books, we employed machine learning techniques (CART, Bagging, Random Forest, AdaBoost, SVM). The following metrics were

chosen to evaluate the quality of the classification: accuracy, precision, recall, and F1-score. Using traditional methods, we contrasted the outcomes with the categorization outcomes.

When a set had both cliques and single-element graphs, the number of features was more than twice as large, according to an analysis of the matrices obtained using both the traditional and our proposed methods of text representation. The number of features obtained in the case of a set including more than two-element cliques, however, was 89% lower in the case of cliques common in the class and 88% lower in the case of cliques common in all documents when compared to the number of features produced from the conventional approaches. The reason for such a large difference is that the vectors from the conventional techniques had more two-element cliques than features. Furthermore, the research revealed that, in comparison to the common set elements in the class, there were 27,596 more common set elements in all texts. Of these elements, 26,614 were two-element cliques.

The results of the trials that were carried out verified that we could get over 90% accuracy, precision, recall, and F1-score when we used a new text representation method based on content analysis to predict book categories. When the AdaBoost algorithm was applied in conjunction with every element in the set, this resulted. It is important to acknowledge that the standard text representation yielded superior levels of accuracy, precision, recall, and F1 score. It is worth mentioning that the subsequent categorization results were exacerbated when a collection consisting solely of cliques with more than two elements was used. This supports the idea that the set's content matters during the classification process.

It is intended to apply parameter selection in research in the future. Better results are obtained with a collection that includes both cliques and single-element graphs, according to the conducted analyses. To evaluate if the set's content is important when the document has fewer words, it is also worthwhile to test our text representation for a collection of shorter texts. It is also worthwhile to investigate how text representations can be used for other NLP tasks, such as summarising. The way the text is presented in this book includes word occurrence information, which is quite helpful for creating a text summary.

References

1. Bales, M.E.; Wright, D.N.; Oxley, P.R.; Wheeler, T.R. *Bibliometric Visualization and Analysis Software: State of the Art, Workflows, and Best Practices*; Cornell University: Ithaca, NY, USA, 2020.
2. Jaradeh, M.Y.; Oelen, A.; Farfar, K.E.; Prinz, M.; D'Souza, J.; Kismihók, G.; Stocker, M.; Auer, S. Open research knowledge graph: Next generation infrastructure for semantic scholarly knowledge. In Proceedings of the 10th International Conference on Knowledge Capture, Marina Del Rey, CA, USA, 19–21 November 2019; pp. 243–246.
3. Ali, S.M.; Noorian, Z.; Bagheri, E.; Ding, C.; Al-Obeidat, F. Topic and sentiment aware microblog summarization for Twitter. *J. Intell. Inf. Syst.* **2020**, *54*, 129–156.
4. Wanigasooriya, A.; Silva, W.P.D. *Automated Text Classification of Library Books into the Dewey Decimal Classification (DDC)*; University of Kelaniya: Kelaniya, Sri Lanka, 2021.
5. Hirschberg, J.; Manning, C.D. Advances in natural language processing. *Science* **2015**, *349*, 261–266.
6. Liddy, E.D. *Natural Language Processing*; Syracuse University: Syracuse, NY, USA, 2001.

7. Sharifi, B.; Hutton, M.A.; Kalita, J.K. Experiments in microblog summarization. In Proceedings of the 2010 IEEE Second International Conference on Social Computing, Minneapolis, MN, USA, 20–22 August 2010; pp. 49–56.
8. Mosa, M.A.; Hamouda, A.; Marei, M. Graph coloring and ACO-based summarization for social networks. *Expert Syst. Appl.* **2017**, *74*, 115–126.
9. Mosa, M.A.; Hamouda, A.; Marei, M. Ant colony heuristic for user-contributed comments summarization. *Knowl.-Based Syst.* **2017**, *118*, 105–114.
10. Rumagit, R.Y.; Setiyawati, N.; Bangkalang, D.H. Comparison of graph-based and term weighting method for automatic summarization of online news. *Procedia Comput. Sci.* **2019**, *157*, 663–672.
11. Crossley, S.A.; Kim, M.; Allen, L.; McNamara, D. Automated summarization evaluation (ASE) using natural language processing tools. In Proceedings of the International Conference on Artificial Intelligence in Education, Chicago, IL, USA, 25–29 June 2019; pp. 84–95.
12. Liang, B.; Su, H.; Gui, L.; Cambria, E.; Xu, R. Aspect-based sentiment analysis via affective knowledge enhanced graph convolutional networks. *Knowl.-Based Syst.* **2022**, *235*, 107643.
13. Belwal, R.C.; Rai, S.; Gupta, A. A new graph-based extractive text summarization using keywords or topic modeling. *J. Ambient. Intell. Humans. Comput.* **2021**, *12*, 8975–8990.
14. Dai, Y.; Shou, L.; Gong, M.; Xia, X.; Kang, Z.; Xu, Z.; Jiang, D. Graph fusion network for text classification. *Knowl.-Based Syst.* **2022**, *236*, 107659.
15. Probierz, B.; Kozak, J.; Hrabia, A. A comparative study of classification and clustering methods from text of books. In Proceedings of the Intelligent Information and Database Systems: 14th Asian Conference, ACIIDS 2022, Ho Chi Minh City, Vietnam, 28–30 November 2022; Proceedings, Part II. Springer: Cham, Switzerland, 2022; pp. 13–25.
16. Chou, C.; Chu, T. An Analysis of BERT (NLP) for Assisted Subject Indexing for Project Gutenberg. *Cat. created. Q.* **2022**, *60*, 807–835.
17. Betts, T.; Milosavljevic, M.; Oberlander, J. The utility of information extraction in the classification of books. In Proceedings of the European Conference on Information Retrieval, Rome, Italy, 2–5 April 2007; pp. 295–306.
18. Brooke, J.; Hammond, A.; Hirst, G. GutenTag: An NLP-driven tool for digital humanities research in the Project Gutenberg corpus. In Proceedings of the Fourth Workshop on Computational Linguistics for Literature, Denver, CO, USA, 4 June 2015; pp. 42–47.
19. Bean, R. The use of Project Gutenberg and hexagram statistics to help solve famous unsolved ciphers. In Proceedings of the 3rd International Conference on Historical Cryptology HistoCrypt 2020, Budapest, Hungary, 15–17 June 2000; Linköping University Electronic Press: Linköping, Sweden, 2020; Volume 171, pp. 31–35.
20. Chowdhary, K. *Fundamentals of Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2020.