applied sciences

MDPI

Article

TrendFlow: A Machine Learning Framework for Research Trend Analysis

Tao Xiang [1], Sufang Chen [2], Yiwei Zhang [3] and Rui Zhu [4,*]

[1] Department of Informatics, Technical University of Munich, 85748 Munich, Germany; tao.xiang@tum.de
[2] School of Civil and Transportation Engineering, Southeast University Chengxian College, Nanjing 210088, China; sufangchen@cxxy.seu.edu.cn
[3] School of Mechanical Engineering, Southeast University, Nanjing 210000, China; 220221195@seu.edu.cn
[4] School of Engineering and Design, Technical University of Munich, 85748 Munich, Germany
* Correspondence: r.zhu@tum.de

Abstract: As various research fields continue to evolve, new technologies emerge constantly, making it challenging for scholars to keep up with the latest and most promising research directions. To address this issue, we propose TrendFlow, a framework that leverages machine learning and deep learning techniques for analyzing research trends. TrendFlow first searches relevant literature based on user-defined queries, then clusters the searched literature according to the abstracts, and finally generates keyphrases of the abstracts as research trends for each cluster. Our experimental results highlight the superior performance of TrendFlow compared to traditional literature analysis tools. We have released the beta version of TrendFlow on Huggingface.

Keywords: research trend analysis; machine learning; clustering; keyphrase generation

Citation: Xiang, T.; Chen, S.; Zhang, Y.; Zhu, R. TrendFlow: A Machine Learning Framework for Research Trend Analysis. *Appl. Sci.* **2023**, *13*, 7029. https://doi.org/10.3390/app13127029

Academic Editor: Krzysztof Koszela

Received: 6 May 2023
Revised: 2 June 2023
Accepted: 6 June 2023
Published: 11 June 2023

1. Introduction

The field of scientific research is in a constant state of flux, with new technologies emerging at a rapid pace. Consequently, scholars face the challenge of staying abreast of the latest and most promising research directions in their respective fields.

To provide a clearer understanding of this challenge, we introduce a novel machine learning task called research trend analysis (RTA). In RTA, the input consists of a set of literature and the output is comprised of pairs (cluster, keyphrases of the cluster). Each cluster represents a subset of original literature that shares the same research trend, and the keyphrases serve as indicators of that research trend.

It is natural to observe that the literature within related research areas tends to be grouped in the same cluster. The size of the cluster, determined by the number of included papers, reflects the popularity of the corresponding research trend. The RTA problem can be viewed as a more challenging version of a topic modeling task, since topic modeling can only find "keywords" of documents, for example "dogs", instead of the keyphrases [1]. To this end, we do not focus on topic modeling tasks in this work.

On the other hand, language models such as BERT [2] and GPT-3 [3] have revolutionized natural language processing (NLP) with their remarkable natural-language-understanding capabilities. They have been widely used for various text generation tasks, such as text summarization and keyphrase generation. Among them, KeyBART [4] is a generative pretrained language model that learns rich keyphrase representation and can improve many downstream NLP tasks, especially the keyphrase generation task.

Motivated by these remarkable achievements, we propose TrendFlow, an innovative framework for RTA. TrendFlow follows a similar pattern to BERTopic [5], a topic modeling framework that clusters documents and identifies keywords using traditional topic modeling algorithms. However, TrendFlow distinguishes itself by employing advanced language models for generating keyphrases instead of relying on topic modeling algorithms. For

this reason, we curate an RTA dataset containing 90 (abstract, trend) samples and present the KeyBART-adapter, a new model architecture that extends KeyBART using the adapter technique [6]. KeyBART-adapter freezes the parameters of KeyBART and only trains the adapters, which allows for efficient adaptation to research trend generation with limited training data and alleviates the forgetting problem of KeyBART. Furthermore, to enhance the user experience, TrendFlow supports automatic literature search on literature platforms (e.g., IEEE) based on user-specified queries and visualizes the final results.

The general working pipeline of TrendFlow Is given as follows:

1. TrendFlow starts by searching relevant literature on literature platforms such as IEEE using user-defined queries.
2. It then encodes the abstracts of the literature into embeddings (vectors) and clusters the embeddings.
3. It generates research trends from the abstracts for each cluster and visualizes the final results.

The key contributions of this work are the following:

I. We propose TrendFlow, a framework that solves the RTA problem. We have published the source code on GitHub (https://github.com/leoxiang66/research-trends-analysis, accessed on 5 June 2023).
II. We annotate an RTA dataset that contains 90 (abstract, trend) samples and publish it on Huggingface (https://huggingface.co/datasets/Adapting/abstract-keyphrases, accessed on 5 June 2023).
III. We present KeyBART-adapter, a new model that extends KeyBART with adapters for few-shot learning. We have published the source code on GitHub (https://github.com/leoxiang66/KeyBartAdapter, accessed on 5 June 2023).

The remainder of this paper is organized as follows: In Section 2, we review related work in recent years; in Section 3, we dive into details of our framework; in Section 4, we introduce the evaluation methods used in this work; in Section 5, we demonstrate and analyze the evaluation results; in Section 6, we highlight advantages and limitations of the proposed framework; and in Section 7, we conclude our work and discuss directions for future work.

2. Related Work

2.1. Text Encoding

Text encoding plays a critical role in Natural Language Processing (NLP) by enabling computers to comprehend and process textual data effectively. There are several text encoding techniques, ranging from traditional methods such as Bag-of-Words and TF-IDF to more advanced neural word embeddings.

Bag-of-Words (BoW). The Bag-of-Words model [7] is one of the simplest text encoding techniques. It represents each document as a "bag" of its words, disregarding grammar and word order but keeping track of word frequency. The BoW model builds a vocabulary comprising all unique words in the dataset and encodes each document as a vector. In this vector representation, each element corresponds to the frequency of a word in the document.

Term Frequency-Inverse Document Frequency (TF-IDF). TF-IDF [8] is an improvement over the Bag-of-Words model. It considers both the frequency of words in a document (Term Frequency) and their importance across all documents (Inverse Document Frequency). The idea is to give more weight to words that appear frequently in a document but rarely in the entire dataset, as they are more likely to carry important information. The resulting encoding is a weighted vector that better represents the semantic content of the document.

Neural Word Embeddings. Neural word embeddings, such as Word2Vec [9,10] and GloVe [11], are more advanced text encoding techniques that represent words as continuous vectors in a high-dimensional space. These embeddings are trained using neural networks

to capture semantic and syntactic relationships between words, resulting in similar words having similar vector representations. In contrast to traditional methods, neural word embeddings can capture the meaning and context of words more effectively. In recent years, neural word embeddings have been used in various NLP tasks, such as sentiment analysis, machine translation, and document classification. One notable application is the Transformer model, which has become the foundation for state-of-the-art NLP models such as BERT [2], a pre-trained language model that learns contextualized word embeddings by processing text bidirectionally.

2.2. Dimension Reduction

To reduce computational costs, we aim to employ dimensionality reduction techniques on word embeddings. The primary objective of dimensionality reduction methods is to transform high-dimensional datasets into lower-dimensional representations while retaining most of the information. This trade-off involves finding the right balance between the number of dimensions and information preservation. Although reducing dimensions may lead to a slight loss of information, it helps mitigate the challenges associated with high dimensionality and simplifies tasks such as visualization, training, and clustering on lower-dimensional datasets. Some popular dimensionality reduction methods include principal component analysis (PCA) and uniform manifold approximation and projection (UMAP).

PCA: PCA [12] is a widely-used dimensionality reduction technique. It extracts the main feature components (principal components) of the original data by solving an eigen decomposition problem. PCA then changes the basis of high-dimensional data and generates a final low-dimensional representation. The technique assumes normalized input features, making it a general-purpose method applicable to any dataset.

UMAP: UMAP [13] is a dimensionality reduction technique based on Riemannian geometry, algebraic topology, and graph layout algorithms. Unlike PCA, UMAP relies on several assumptions about the data. Firstly, it assumes that the data is uniformly distributed on a Riemannian manifold. Secondly, the Riemannian metric is considered to be locally constant. Lastly, the manifold is assumed to be locally connected. The algorithm operates by constructing a high-dimensional graph that represents the original data. Subsequently, it identifies a low-dimensional graph that most closely resembles the high-dimensional counterpart through iterative optimization.

2.3. Clustering

Clustering techniques are crucial in NLP tasks as they facilitate the grouping of similar text or documents based on their content, allowing for effective organization, classification, and summarization of textual data. There are various types of clustering algorithms, including centroid-based, density-based, hierarchical, graph-based, and model-based clustering. In this section, we introduce one representative algorithm from each category.

Centroid-Based Clustering: K-means. K-means is a widely used unsupervised clustering algorithm that partitions data points into a specified number of clusters (k) based on their similarity [14]. It operates by iteratively assigning each data point to the nearest centroid and then updating the centroid's position by calculating the mean of all points within the cluster. This process is repeated until convergence or a maximum number of iterations is reached. K-means is particularly efficient for large-scale datasets and can be applied to NLP tasks such as text clustering, document categorization, and topic modeling [15].

Density-Based Clustering: DBSCAN. Density-based spatial clustering of applications with noise (DBSCAN) is a popular density-based clustering algorithm [16]. It groups data points based on the density of their neighborhoods, allowing it to discover clusters of varying shapes and sizes. DBSCAN is noise-tolerant, meaning it can effectively identify and separate noise from clusters. This property makes DBSCAN suitable for NLP tasks where the data distribution is non-uniform or contains outliers.

Hierarchical Clustering: Agglomerative Clustering. Agglomerative clustering, a hierarchical clustering algorithm, is employed to recursively merge clusters based on their pairwise similarities [17]. Initially, each data point is considered as a separate cluster. In each iteration, the two clusters with the highest similarity are merged, forming a dendrogram that represents the hierarchical structure of the data. The process continues until all data points belong to a single cluster or a specified stopping criterion is met. In NLP tasks, agglomerative clustering is effective for exploring the hierarchical relationships among documents, such as articles or emails, and for uncovering underlying themes.

Graph-based Clustering: Spectral Clustering. Spectral clustering is a graph-based clustering technique that leverages the eigenvectors of the Laplacian matrix derived from the similarity graph of the dataset [18]. The algorithm projects the data into a lower-dimensional space in which clusters are more easily separable. It then applies a clustering method such as K-means to partition the data in this new space. Spectral clustering is particularly useful for NLP tasks where data points may not be linearly separable in the original feature space, such as when dealing with complex semantic relationships or multi-modal data distributions.

Model-based Methods: GMM. Gaussian Mixture Model (GMM), on the other hand, is a generative probabilistic model that represents a dataset as a mixture of multiple Gaussian distributions [19]. The expectation-maximization (EM) algorithm is used to estimate the parameters of these Gaussian distributions. GMM clustering is soft, allowing each data point to belong to multiple clusters with varying probabilities. This characteristic can be advantageous in NLP tasks where overlapping topics or semantic ambiguity may be present.

2.4. Keyphrase Generation

Keyphrase generation (KG) is a fundamental task in natural language processing (NLP) that aims to generate a set of keyphrases given a document. Keyphrases are concise and informative phrases that capture the main topics or themes of a document. They can be used for various applications such as document summarization, indexing, retrieval, clustering, and classification.

Most traditional methods for Knowledge Graph (KG) construction primarily rely on extractive techniques, which involve selecting keyphrases from existing words or phrases in the document. Extractive methods typically follow a two-step process: candidate selection and ranking. In candidate selection, potential keyphrases are identified based on linguistic patterns, such as noun phrases or term frequency. Subsequently, in the ranking step, scores are assigned to the candidates using various features, including TF-IDF, position, length, or graph-based measures. Some examples of extractive methods are KEA (Witten et al., 1999) [20], TextRank (Mihalcea and Tarau, 2004) [21], and SingleRank (Wan and Xiao, 2008) [22].

However, extractive methods have some limitations. First, they cannot generate novel keyphrases that do not appear in the document but are still relevant to its content. Second, they may miss important keyphrases that are expressed implicitly or paraphrased in the document. Third, they may produce redundant or overlapping keyphrases that convey similar information.

To overcome these limitations, recent methods for KG are mostly generative, meaning that they can produce new words or phrases that are not necessarily present in the document.

Generative methods mainly adopt the transformer-based sequence-to-sequence (seq2seq) models, which employ encoder-decoder architectures with attention mechanisms to generate keyphrases [23]. Example models include T5 [24] and BART [25].

Recently, a novel pre-trained model, known as KeyBART, has been introduced for acquiring rich keyphrase representations, resulting in enhanced keyphrase generation performance by harnessing the robust capabilities of the BART architecture [4]. Experi-

mental findings demonstrate that KeyBART surpasses state-of-the-art methods in terms of performance for the keyphrase generation task.

In this work, we concentrate on the KeyBART model. Given that our dataset is relatively small, comprised of only 50 training samples, fully fine-tuning KeyBART may lead to a forgetting problem [24]. Consequently, we investigate the incorporation of adapter techniques into the KeyBART model to address this issue.

Adapters are lightweight modules that can be seamlessly integrated into the architecture of a pre-trained model. They enable efficient fine-tuning while addressing the forgetting problem. Comprising feed-forward neural networks, adapters adjust the activations of the model's hidden layers without modifying the original weights [6]. By leveraging adapters, the model can be fine-tuned on smaller datasets without compromising the crucial pre-trained knowledge, thereby preserving its generalization capabilities.

In the context of KeyBART, integrating adapter techniques can potentially enhance the model's ability to generate meaningful keyphrases from a limited dataset. By preserving the pre-trained knowledge and adjusting the activations with adapters, KeyBART can effectively learn from the small dataset without losing its ability to generate coherent and relevant keyphrases.

2.5. Topic Modeling

Topic modeling is a technique used in natural language processing and information retrieval to discover the underlying themes or topics within a collection of documents. Topic modeling algorithms such as latent dirichlet allocation (LDA) [26] and non-negative matrix factorization (NMF) [27] can automatically identify and group words that frequently co-occur, thus revealing the hidden semantic structure of the text corpus.

LDA: LDA, first introduced by Blei, Ng, and Jordan (2003) [26], is a generative probabilistic model that assumes each document is a mixture of a predefined number of topics, and each topic is a distribution over words. The model learns these topic distributions and document-topic proportions using a Bayesian framework, incorporating Dirichlet priors to capture uncertainty and smoothness. LDA has been widely used and acclaimed for its ability to discover coherent and interpretable topics in large text corpora [28].

NMF: NMF, initially proposed by Lee and Seung (1999) [27], is a linear algebra-based technique that decomposes a non-negative data matrix into two lower-dimensional nonnegative matrices. One matrix captures the underlying topics, while the other represents the weights of the topics for each document. NMF is widely applied in domains where data sparsity and non-negativity play a crucial role, such as text mining and image processing [29]. By imposing non-negativity constraints, NMF results in a parts-based representation, which can lead to more interpretable topics [30].

Traditional topic modeling techniques have certain drawbacks when compared to deep-learning-based methods. For example, they may have limited semantic understanding] as they rely on word co-occurrence statistics, making it challenging to capture more profound semantic relationships between words and phrases. Additionally, most traditional topic modeling techniques necessitate specifying the number of topics in advance, which may not be optimal or straightforward to determine [31].

3. Framework Design

3.1. Problem Statement & Notation

In this section, we present the primary notations utilized in this paper and provide a formal description of the task. The central objective of TrendFlow is to perform research trend analysis. The formal definition is as follows:

Definition 1. *(Research Trend Analysis (RTA)). Consider a collection of literature papers* $L = L_1, L_2, \ldots, L_n$. *The goal of research trend analysis is to identify k research trends* $T_1, T_2, \ldots, T_k$. *Each research trend* T_i *is characterized by a trend text* W_i, *which consists of phrases like "reinforcement learning", and a subset of literature papers* $C_i \subseteq L$ *(i.e., a cluster) that belong to this trend.*

To rephrase and enhance: Essentially, when provided with a set of scholarly abstracts, the RTA task is to identify all research trends and their corresponding literature. It is important to note that the number of trends (represented by k) is not predetermined; rather, it is ascertained throughout the clustering process.

TrendFlow utilizes three distinct modules to address the RTA task: (1) text encoder, (2) clustering, and (3) keyphrase generator. The text encoder transforms each literature abstract into a vector representation. Subsequently, these vectors are clustered based on their relative distances. After the clusters have been established, the keyphrase generator produces research trends for each cluster. A comprehensive structure of the TrendFlow framework is available in Figure 1.

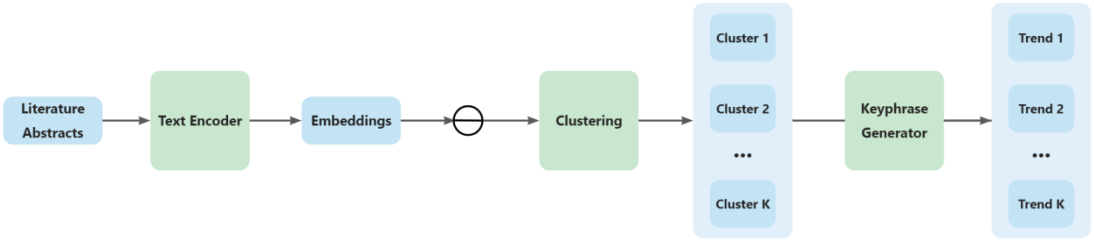


Figure 1. The framework structure of TrendFlow. The ⊖ symbol represents dimensionality reduction.

3.2. Text Encoder Module

Given a collection of literature abstracts (i.e., texts), our initial task is to transform them into vector representations to facilitate subsequent clustering. We have decided to employ transformer-based encoder models for this purpose owing to their exceptional performance. At the time of conducting this research, numerous high-performing encoder models have emerged, including BERT [2], RoBERTa [32], and DistilBERT [33].

Recently, Muennighoff et al. published the Massive Text Embedding Benchmark (MTEB) Leaderboard [34], which evaluates the performance of various transformer encoder models across multiple NLP downstream tasks. According to the results, the all-mpnet-base-v2 model [35] demonstrates strong clustering performance while maintaining a moderate word-embedding dimension of 768, resulting in lower computational costs. Due to limited training resources, we employ the all-mpnet-base-v2 model to encode abstracts in this work. However, as TrendFlow is a flexible framework, we can easily substitute the encoder model with a superior alternative as it becomes available in the future.

The all-mpnet-base-v2 model is a sentence-transformers model that maps sentences and paragraphs to a 768-dimensional dense vector space. It is based on MPNet [36], which is a transformer model that combines masked language modeling (MLM) and permutation language modeling (PLM) objectives for pre-training.

According to sentence-transformers documentation (https://www.sbert.net/, accessed on 5 June 2023), this model provides the best quality among all evaluated models for sentence embeddings. It achieves state-of-the-art results on several benchmarks, such as STS-B [37], SICK-R [38], TREC-QA [39], and Quora-Question-Pairs [40].

3.3. Clustering Module

The clustering process is a vital component of the TrendFlow framework, as it aims to group related literature abstracts based on their vector representations. This section introduces the clustering algorithm employed in this study, which plays a crucial role in identifying research trends within the literature collection. Additionally, we discuss the implementation of a dimensionality reduction technique to enhance computational efficiency and performance during the clustering process.

3.3.1. Clustering Algorithm

In this work, we employ and compare five clustering algorithms: K-Means, DBSCAN, agglomerative clustering, the Gaussian mixture model (GMM), and spectral clustering.

These algorithms were selected because they can either inherently determine the number of clusters or utilize additional techniques to estimate the optimal number of clusters without prior specification. More specifically:

I. For K-means, we apply the elbow method [41] to automatically determine the optimal number of clusters. This technique involves plotting the explained variation as a function of the number of clusters and identifying the point where adding more clusters does not significantly improve the explained variation. This point, referred to as the "elbow," indicates the optimal number of clusters.
II. For GMM, we use the Bayesian information criterion (BIC) [42] to estimate the optimal number of clusters. BIC evaluates the trade-off between model complexity and goodness-of-fit, providing a measure that helps select the best model from a set of candidate models with different numbers of clusters.
III. DBSCAN inherently determines the number of clusters based on the density of the data points in the feature space. The algorithm identifies dense regions in the data and groups them into clusters without requiring a predetermined number of clusters.
IV. Agglomerative clustering is a hierarchical clustering algorithm that automatically determines the number of clusters through the construction of a hierarchical tree. By analyzing the tree structure, an optimal number of clusters can be determined by selecting a suitable cut-off point that effectively represents the underlying data.
V. Spectral clustering inherently determines the number of clusters by leveraging the eigenvectors of the graph Laplacian matrix. The eigengap heuristic [43] can be used to automatically estimate the optimal number of clusters. This method involves computing the eigenvalues of the Laplacian matrix and selecting the number of clusters corresponding to the largest gap between consecutive eigenvalues.

3.3.2. Dimensionality Reduction

When performing vector clustering, the dimensionality of a vector is a critical issue. Generally, higher-dimensional vectors can express richer information but increase computational complexity and storage space requirements. In this work, we explore to use PCA [12] to reduce the dimensionality of the word embeddings. We opt for PCA over other techniques because it only assumes normalized input data and retains the maximum amount of variance in the lower-dimensional representation [44].

3.4. Keyphrase Generator Module

Once we have k clusters $C_1, \ldots, C_k$, the keyphrase generator generates keyphrases for each cluster C_i. In this work, we use KeyBART [4] as the keyphrase generator. KeyBART is a task-specific language model that learns rich representation of keyphrases from text documents by using different masking strategies for pre-training transformer language models. It achieves state-of-the-art performance on keyphrase generation and near state-of-the-art performance on other NLP tasks such as named entity recognition, relation extraction, question answering, and summarization.

In order to enhance KeyBART's comprehension and generation of research trends, we manually annotate an RTA dataset of 90 samples, each containing an abstract and its corresponding research trend, for the purpose of fine-tuning. To be more specific, the first author examined the most recent papers on Papers with Code (https://paperswithcode.com/, accessed on 5 June 2023) and Arxiv (https://arxiv.org/, accessed on 5 June 2023), providing annotations for the identified research trends. The research trends follow the concatenated keyphrase pattern, as shown in Figure 2. The majority of the chosen papers belong to the fields of machine learning and deep learning, which align with the first author's area of expertise. Subsequently, we partitioned the dataset into training (50 samples), validation (20 samples), and test (20 samples) subsets. We made this dataset public on Huggingface.

"text-to-image synthesis;diffusion model;text-to-3D synthesis;probability density distillation"

Figure 2. An example representation of the labeled research trends, where each research trend is separated by a semicolon.

Given the small size of our dataset, fully fine-tuning KeyBART (consisting of 560 M parameters) may result in a loss of previously acquired knowledge. Consequently, we investigate the integration of the adapters technique [6] with KeyBART. By adding adapter layers between KeyBART's decoder layers and freezing all original parameters, we train only the newly introduced adapter layers. This approach aims to retain KeyBART's original knowledge while enhancing its understanding of research trends.

More specifically, we follow the methodology presented in [45]. Each adapter layer comprises a layer normalization layer, succeeded by a down-projection linear layer, a ReLU activation function, an up-projection linear layer, and a residual connection. This configuration enables compression. The input and output dimensions of the adapter layers are adjusted to match KeyBART's decoder layer, while the middle hidden layer's dimension serves as a hyperparameter. In this work, we experiment with hidden dimensions of 32, 64, 128, 256, and 512. We refer to this model architecture as KeyBART-adapter-d, where d denotes the hidden dimension. In our implementation, KeyBART-adapter-32 encompasses 0.8 M trainable parameters, while KeyBART-adapter-512 comprises 12 M trainable parameters. The considerable decrease in parameter number implies that the fine-tuning process could potentially be much more efficient. The detailed structure of the adapter layer can be found in Figure 3.

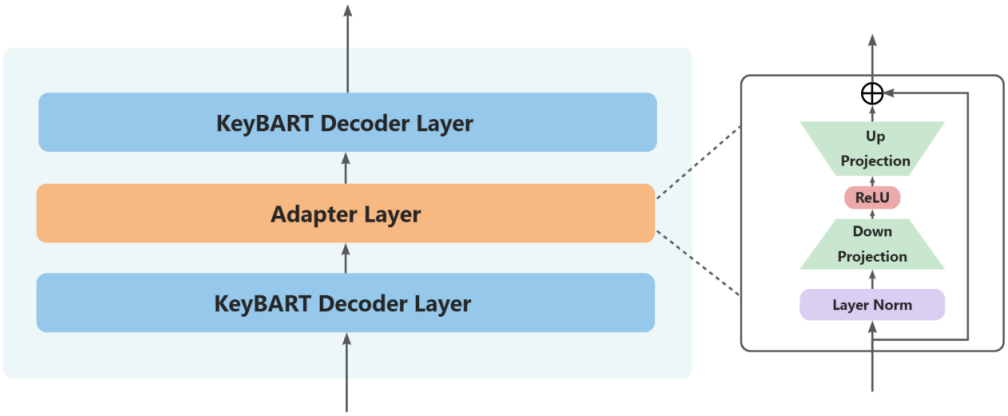


Figure 3. The layer structure of adapter layers in KeyBART-adapter.

Assuming cluster C_i comprises m pieces of literature, the keyphrase generator generates research trends for each literature within the cluster, yielding m predicted research trend strings. The final step in TrendFlow involves grouping similar keyphrases. For example, the keyphrase "Natural Language Processing" should be recognized as identical to "natural language processing" or "Natural Language Processing (NLP)."

Initially, we split the m raw research trend strings using a semicolon delimiter to obtain a list of keyphrases. Subsequently, we apply the union-find algorithm [46]—with the Ratcliff/Obershelp algorithm [47] serving as the similarity measurement—to group keyphrases similar in token level. We set the similarity threshold at 0.8, which implies that only keyphrases with similarity scores exceeding 0.8 will be grouped together. Eventually, the groups obtained through the union-find algorithm represent the research trends associated with a single cluster. An example group could be {"natural language processing", "Natural Language Processing"}.

4. Evaluation Methods

This chapter presents the evaluation methods utilized to assess the performance of the TrendFlow framework. To evaluate the quality of the generated research trends and the effectiveness of literature abstract clustering, a combination of qualitative and quantitative evaluation techniques is employed. Furthermore, the datasets utilized for evaluation purposes are described in detail.

4.1. Datasets

For our experiments, we use two datasets:

1. **Abstract-Only Dataset:** We gather abstracts from Arxiv in the domains of natural language processing (NLP), computer vision (CV), and audio processing (AP). Specifically, we randomly sampled 67 abstracts from the NLP domain, 92 from the CV domain, and 59 from the AP domain. To enable domain-based clustering experiments, we labeled each abstract with a corresponding domain code (e.g., "0" for CV). The resulting dataset is publicly available on Huggingface.
2. **RTA Dataset:** We used the RTA dataset introduced in Section 3.4.

4.2. Quantitative Evaluation

For the quantitative evaluation, we use both the abstract-only dataset and the RTA dataset to perform the following evaluations:

1. **Clustering Evaluation:** We performed clustering experiments on the abstract-only dataset using various clustering algorithms such as K-means, GMM, DBSCAN, agglomerative clustering, and spectral clustering. To evaluate the clustering performance, we used standard clustering evaluation metrics, such as the adjusted rand index (ARI) [48], silhouette score [49], and Calinski-Harabasz index (CHI) [50]. We computed these metrics for each clustering algorithm and compared their performance.
2. **Dimensionality Reduction Evaluation:** To evaluate the impact of dimensionality reduction on clustering performance, we compared the clustering results obtained using the original high-dimensional word embeddings with those obtained using the reduced-dimensional embeddings. We used the same clustering evaluation metrics mentioned above (ARI, Silhouette Score, and CHI) for this comparison.
3. **Keyphrase Generation Evaluation:** To assess the comparative performance of KeyBARTadapter and KeyBART, we initially trained both models on the training set of the RTA dataset, and subsequently fine-tuned their hyperparameters using the validation set. In the case of KeyBART-adapter, we fine-tuned the hidden dimension of adapter layers over a set of five logarithmically-spaced values, ranging from 32 to 512. Additionally, the learning rate was fine-tuned over logarithmically-spaced ranges, i.e., 1×10^{-6} to 1×10^{-4} with 20 equally-spaced steps, resulting in 100 configurations for the KeyBART-adapter. For KeyBART, we solely fine-tuned the learning rate within the same range as the KeyBART-adapter, resulting in a total of 20 configurations. Eventually, we selected the best configuration for KeyBART-adapter and KeyBART on the validation set and evaluated them on the test set.

We adopt the same evaluation metric as used in the original KeyBART paper [4] to select the best configuration on the validation set and for evaluation on the test set. The chosen metric is the F-score@M, which is a harmonic mean of precision and recall, considering the top-M generated keywords. By employing the F-score@M, we can compare our KeyBART-adapter with the KeyBART, ensuring a fair evaluation and allowing us to identify potential improvements or drawbacks in our approach.

4.3. Qualitative Evaluation

For the qualitative evaluation, we conducted a human evaluation involving five researchers with expertise in various domains, such as NLP and CV, to assess the quality of

the generated research trends and the clustering of abstracts. The evaluation consists of the following steps:

1. Participants are provided with a set of research trends and corresponding clusters of literature abstracts from the abstract-only dataset generated by the TrendFlow framework using two different configurations. The first configuration makes use of the Gaussian Mixture Model (GMM) clustering algorithm without applying dimensionality reduction, and employs the KeyBART-adapter model for research trend generation. The second configuration is similar, except that it uses the KeyBART model for research trend generation instead.
2. Participants are asked to rate the relevance ("Relevance of clusters" refers to the degree to which the clusters obtained from a clustering algorithm are meaningful or useful for the RTA task) and coherence ("Coherence of clusters" refers to how internally consistent the data points within each cluster are) of the clusters and the accuracy ("Accuracy of research trends" refers to the degree to which the identified trends represent the actual state of the research in a certain field or topic) of the generated research trends on a five-point Likert scale [51], where one represents the lowest score and five represents the highest score.

This qualitative evaluation aims to compare the performance of research trend generation using the KeyBART and KeyBART-adapter models in the TrendFlow framework, offering insights into the effectiveness of the models and the framework's overall performance.

5. Results

In this chapter, we present the results obtained from the evaluation methods described in Section 4.

5.1. Quantitative Evaluation Results

5.1.1. Clustering Evaluation

The clustering evaluation metrics for the TrendFlow framework are presented in Table 1. The best score is made bold from each metric. DBSCAN does not have any results, since it failed to detect the number of clusters correctly.

Table 1. Clustering evaluation results for the clustering algorithms used in TrendFlow framework.

	Adjusted Rand Index	Silhouette Score	Calinski-Harabasz Index
K-means	0.6944	**0.0843**	**17.362**
GMM	**0.7202**	0.0839	17.356
DBSCAN	-	-	-
Agglomerative	0.6201	0.0782	16.261
Spectral	0.7087	0.0836	17.349

In Table 1, we present a comparison of five different clustering algorithms employed in the TrendFlow framework. The evaluation of the clustering results is based on three metrics: the adjusted rand index (ARI), the silhouette score, and the Calinski-Harabasz index (CHI).

The Gaussian mixture model (GMM) achieved the highest ARI with a score of 0.7202, indicating that it has the best performance in terms of producing similar cluster assignments to a ground truth labeling. Meanwhile, spectral clustering yielded the second-highest ARI score of 0.7087, followed closely by the K-means clustering algorithm at 0.6944. Agglomerative clustering had a lower ARI score of 0.6201. Unfortunately, the DBSCAN algorithm did not yield any results for this evaluation, as it failed to detect the correct number of clusters, which is three in this context.

In terms of the silhouette score, which measures the intra-cluster similarity and inter-cluster dissimilarity, K-means obtained the highest value at 0.0843, followed by GMM

with a score of 0.0839 and spectral clustering with 0.0836. Agglomerative clustering had the lowest silhouette score of 0.0782, suggesting that it might produce clusters with lower intra-cluster similarity or higher inter-cluster similarity compared to the other algorithms.

Lastly, the Calinski-Harabasz index assesses the quality of clusters by considering both intra-cluster variance and inter-cluster variance. Here, K-means performed the best with a score of 17.362, closely followed by GMM at 17.356 and the spectral clustering algorithm at 17.349. Agglomerative clustering had the lowest CHI score of 16.261.

In conclusion, the Gaussian mixture model performed the best in terms of the adjusted rand index, while K-means had the highest scores for the silhouette score and Calinski-Harabasz index. The choice of the most suitable clustering algorithm for the TrendFlow framework would depend on the desired qualities of the resulting clusters, such as higher intra-cluster similarity, lower inter-cluster similarity, or a closer alignment with a ground truth labeling.

5.1.2. Dimensionality Reduction Evaluation

The dimensionality reduction evaluation results are shown in Table 2. The best score is made bold for each metric. DBSCAN still does not have any results due to its failure in detecting the correct number of clusters.

Table 2. Clustering evaluation results for the clustering algorithms used in TrendFlow framework with dimensionality reduction.

	Adjusted Rand Index	Silhouette Score	Calinski-Harabasz Index
K-means + PCA	0.6944	0.09061	**18.410**
GMM + PCA	0.7065	**0.09064**	18.402
DBSCAN + PCA	-	-	-
Agglomerative + PCA	0.6209	0.0841	17.212
Spectral + PCA	**0.7087**	0.0900	18.395

In Table 2, we present the evaluation results of the same clustering algorithms used in the TrendFlow framework, but with an additional dimensionality reduction step using principal component analysis (PCA) applied beforehand. The three metrics, adjusted rand index (ARI), silhouette score, and Calinski-Harabasz index (CHI), are once again employed to assess the performance of the clustering algorithms.

An interesting observation is that the application of PCA has a minimal impact on the adjusted rand index. In fact, for GMM, there is a slight decrease in the ARI score from 0.7202 to 0.7065. The other algorithms show little to no change in their ARI scores, indicating that the addition of PCA has a limited effect on the cluster assignments' similarity to ground truth labeling. On the other hand, the silhouette score has improved across all clustering algorithms when PCA is applied. GMM achieves the highest silhouette score of 0.09064, followed closely by K-means at 0.09061 and spectral clustering at 0.0900. Agglomerative clustering also sees an increase in its silhouette score to 0.0841.

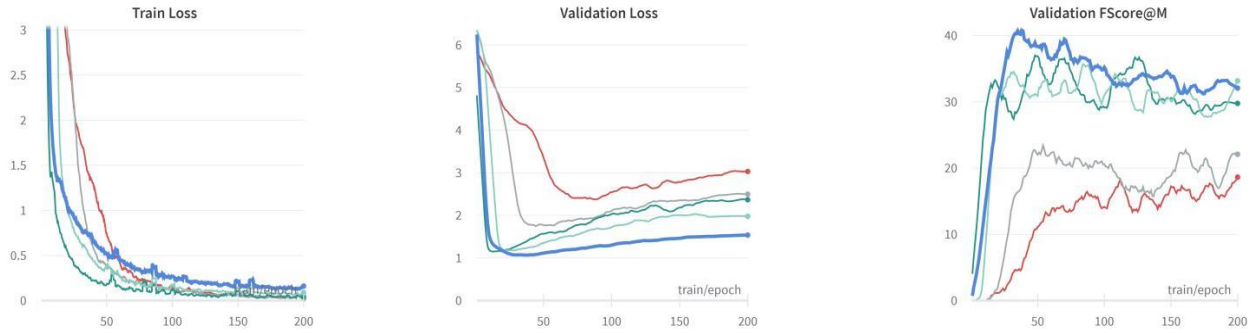
Similarly, the Calinski-Harabasz index demonstrates an improvement for all clustering algorithms after applying PCA, where the K-means still obtains the highest CHI score of 18.410.

In summary, the addition of PCA as a dimensionality reduction technique has minimal impact on the adjusted rand index, with a slight decrease observed for GMM. However, the other two evaluation metrics, the silhouette score and Calinski-Harabasz index, show consistent improvements across all clustering algorithms when PCA is applied. This suggests that incorporating PCA may enhance the overall clustering quality in terms of intra-cluster similarity and inter-cluster dissimilarity.

5.1.3. Keyphrase Generation Evaluation

The training and validation results for KeyBART-adapter and KeyBART are illustrated in Figures 4 and 5, respectively. In Figure 4, the best configuration (lr = 7.3×10^{-5},

hidden = 32) is highlighted in bold blue, and we only display a subset of five configurations for clarity. These selected configurations have the same learning rate and only vary from hidden dimensions. In Figure 5, the best configuration (lr = 2.9×10^{-5}) is highlighted in bold sky-blue, and we only display a subset of five configurations for clarity. The complete set of results is available on wandb (https://api.wandb.ai/links/leoxiang66/uo9odosw, accessed on 5 June 2023).



(**a**) Train loss (**b**) Validation loss (**c**) Validation FScore@M

Figure 4. Training and validation results for KeyBART-adapter, with running average smoothing applied for better visualization.

(**a**) Train loss (**b**) Validation loss (**c**) Validation FScore@M

Figure 5. Training and validation results for KeyBART, with running average smoothing applied for better visualization.

Following the training and fine-tuning of hyperparameters, we have identified the optimal configurations for both KeyBART-adapter and KeyBART, as detailed below:

I. KeyBART-adapter: A learning rate of 7.3×10^{-5} was used, with a hidden dimension of 32 for the adapter layers. The model was trained for 34 epochs on the training set.

II. KeyBART: A learning rate of 2.9×10^{-5} was applied, and the model was trained for 45 epochs on the training set.

Subsequently, we load the model weights for these two configurations and assess their performance on the test set. The evaluation results are presented in Table 3. The KeyBART-adapter outperforms KeyBART in terms of F-score@M (33.25% vs. 32.75%) and Precision@M (40.33% vs. 32.92%), indicating that the KeyBART-adapter model generates more accurate and relevant keyphrases. However, when it comes to Recall@M, KeyBART demonstrates a slightly higher value (33.08%) compared to the KeyBART-adapter (30.07%). This suggests that KeyBART is better at capturing a larger portion of relevant keyphrases within the top M generated keyphrases, albeit with a lower precision.

Table 3. Keyphrase generation evaluation results for KeyBART-adapter and KeyBART.

	F-score@M (%)	Precision@M (%)	Recall@M (%)
KeyBART-adapter	**33.25**	40.33	30.07
KeyBART	32.75	32.92	**33.08**

In summary, the KeyBART-adapter exhibits an overall superior performance in keyphrase generation, particularly in terms of F-score@M and Precision@M. Although KeyBART achieves a marginally higher Recall@M, it is essential to consider the trade-off between precision and recall when selecting the most suitable model for a specific application. In the realm of research trend analysis, scholars are likely to prioritize a smaller number of precise trends over a larger collection of relevant but less accurate trends. This preference stems from the desire to focus on the most critical issues and developments within a given field, enabling researchers to receive targeted guidance and make more informed decisions. As such, placing emphasis on precision would be a prudent approach.

5.2. Qualitative Evaluation Results

The average scores obtained from the human evaluation are presented in Table 4.

Table 4. Human evaluation results for the TrendFlow framework with KeyBART-adapter (Conf1) and KeyBART (Conf2) configurations.

Configuration	Relevance of Clusters	Coherence of Clusters	Accuracy of Research Trends
Conf1	4.66	4.33	**4.33**
Conf2	4.66	4.33	4.16

The results in Table 4 show that the domain experts rated both configurations of the TrendFlow framework highly in terms of the relevance and coherence of the clusters. The identical scores for the clustering metrics in both configurations can be attributed to the same clustering settings being used for both Conf1 and Conf2. However, the raters perceived the research trends generated by the KeyBART-adapter model (Conf1) to be more accurate than those generated by the KeyBART model (Conf2).

This qualitative evaluation demonstrates the effectiveness of both models in clustering research topics and highlights the superior performance of the KeyBART-adapter configuration in generating accurate research trends. The results suggest that the TrendFlow framework, particularly when employing the KeyBART-adapter model, is a valuable tool for analyzing and understanding research trends in a given domain.

6. Discussion

In this chapter, we delve into the advantages and limitations of the TrendFlow framework, building upon the findings discussed in Section 5. Our objective is to offer a well-rounded understanding of the framework's performance, highlighting its limitations and identifying potential areas for improvement.

6.1. Advantages

TrendFlow boasts the following notable advantages:

Advanced text encoding. TrendFlow employs cutting-edge text encoders, leveraging the MTEB leaderboard [35] to significantly enhance subsequent clustering processes. By utilizing state-of-the-art encoding methods, the framework can accurately capture and represent textual information, which is crucial for effective clustering and trend identification.

High-quality research trend generation. The KeyBART-adapter model, after being finetuned on the meticulously annotated RTA dataset, demonstrates exceptional performance in keyphrase generation. It surpasses the baseline KeyBART model in terms of

F-score@M and Precision@M, showcasing the framework's ability to generate accurate and pertinent keyphrases, which is essential for pinpointing research trends.

Flexibility and adaptability. TrendFlow has been designed with a modular architecture, facilitating the effortless replacement of its components to accommodate the integration of more robust models as needed. This framework offers support for a wide array of customizable clustering algorithms, enabling users to select the most suitable algorithm for their specific research domains or datasets. This flexibility and adaptability empowers users to generate accurate research trends by leveraging the algorithm that best aligns with their requirements.

User-friendly UI experience and illustrative visualization. Our web application streamlines the configuration of TrendFlow, enabling users to effortlessly query papers across various literature platforms and automatically receive analyzed results. The visualized outcomes are not only accessible but also easy to comprehend, facilitating user understanding and interpretation of research trends.

6.2. Limitations

However, there are still several limitations for TrendFlow.

Constrained summarization of generated research trends. The existing TrendFlow framework generates research trends for each paper within a single cluster, which may lead to an overwhelming number of trends if the cluster contains a large number of papers. This necessitates refined post-processing of the generated research trends. At present, TrendFlow utilizes a union-find algorithm to group similar research trends, which is a basic and straightforward approach. Nevertheless, exploring more advanced methods (e.g., neural-network-based techniques) is essential for handling potentially larger datasets.

Domain-specific adaptations. In this study, the RTA datasets were annotated by sampling literature exclusively from machine learning and deep learning-related domains. Consequently, the generalizability and adaptability of the TrendFlow framework to diverse research domains remain largely unexplored. It is important to recognize that the framework's performance might vary across different domains. Future research should investigate the necessity of domain-specific adaptations or customizations to enhance its performance in various fields of study.

Scalability. The TrendFlow framework's performance and scalability in handling largescale datasets have not been thoroughly evaluated. The increasing volume of research publications may pose challenges in terms of computational resources, processing time, and memory requirements. Future work should consider addressing these issues by optimizing the framework's performance and testing its scalability with more extensive datasets.

In addition to the framework limitations, the human evaluation of TrendFlow's performance should be taken with caution due to the small dataset size and subjective nature of manual ratings. A more extensive evaluation with a larger dataset and a more diverse pool of domain experts would help obtain more reliable results.

7. Conclusions

In this work, we introduce TrendFlow, a novel framework for understanding research trends. TrendFlow is designed to address the challenges faced by scholars in keeping up with the latest and most promising research directions in their respective fields. We propose the research trend analysis (RTA)task, which involves clustering literature and generating keyphrases to represent the research trends within each cluster. To support the development of TrendFlow, we annotate an RTA dataset containing 90 (abstract, trend) samples and present the KeyBART-adapter, a new model architecture that extends KeyBART with adapter techniques. We have performed both quantitative and qualitative evaluations on different configurations of TrendFlow. Our evaluation demonstrates that TrendFlow with KeyBART-adapter as a research trend generator performs well in identifying research trends and generating keyphrases that effectively represent these trends.

Despite its advantages, the TrendFlow framework has some limitations, such as constrained summarization of generated research trends, potential domain-specific adaptations, and scalability concerns. Future work should address these challenges by: (1) Exploring more advanced methods for trend summarization to improve the framework's ability to capture the essence of research trends; (2) Investigating the generalizability of TrendFlow across different domains and, if necessary, annotating additional data across various domains to enhance its adaptability; and (3) Testing and optimizing the framework's performance when handling large-scale datasets to ensure its scalability. Moreover, the TrendFlow framework could be extended by: (1) Incorporating additional data sources, such as citations, author affiliations, and conference proceedings, to complement the information obtained from abstracts and further refine research trend analysis; and (2) Introducing temporal analysis, as the current version of TrendFlow does not explicitly consider the temporal aspects of research trends. By incorporating time-based analysis, the framework could help identify the evolution of research trends over time and provide insights into emerging and declining areas of interest.

In conclusion, TrendFlow represents a significant step forward in the field of research trend analysis, offering researchers a valuable tool to stay updated with the latest advancements in their domains. As research continues to grow and evolve rapidly, tools such as TrendFlow will become increasingly vital in assisting scholars to navigate the ever-expanding landscape of scientific literature.

Author Contributions: T.X. and S.C. contributed equally to this work. Conceptualization, T.X., S.C. and R.Z.; Methodology, T.X. and S.C.; Software, T.X. and S.C.; Validation, T.X., S.C. and Y.Z.; Formal analysis, T.X., S.C. and Y.Z.; Investigation, Y.Z. and R.Z.; Resources, R.Z.; Writing—original draft, T.X.; Supervision, R.Z.; Funding acquisition, S.C.; Writing—review & editing, T.X., S.C., Y.Z. and R.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by TUM Open Access Publishing Fund, the National Science Research Program Cultivation Fund of Southeast University Chengxian College (No. 2022NCF005), and in part by the research General Project of Natural Science Research in Jiangsu Universities (No. 20KJB410001) and School Level First-Class Curriculum Construction Project (No. yIkc2105).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset proposed in this paper can be obtained from the author with a reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kherwa, P.; Bansal, P. Topic modeling: A comprehensive review. *ICST Trans. Scalable Inf. Syst.* **2018**, *7*, 159623. [CrossRef]
2. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2018**, arXiv:1810.04805.
3. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. *arXiv* **2020**, arXiv:2005.14165.
4. Kulkarni, M.; Mahata, D.; Arora, R.; Bhowmik, R. Learning rich representation of keyphrases from text. *arXiv* **2021**, arXiv:2112.08547.
5. Grootendorst, M. BERtopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv* **2022**, arXiv:2203.05794.
6. Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; Gelly, S. Parameter-Efficient Transfer Learning for NLP. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019.
7. Harris, Z.S. Distributional structure. *Word* **1954**, *10*, 146–162. [CrossRef]
8. Jones, K.S. A statistical interpretation of term specificity and its application in retrieval. *J. Doc.* **1972**, *28*, 11–21. [CrossRef]
9. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
10. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed Representations of Words and Phrases and their Compositionality. *arXiv* **2013**, arXiv:1310.4546.

11. Pennington, J.; Socher, R.; Manning, C. Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014.
12. Pearson, K. Liii. On lines and planes of closest fit to systems of points in space. *Lond. Edinb. Dublin Philos. Mag. J. Sci.* **1901**, *2*, 559–572. [CrossRef]
13. McInnes, L.; Healy, J.; Saul, N.; Großberger, L. UMAP: Uniform Manifold Approximation and Projection. *J. Open Source Softw.* **2018**, *3*, 861. [CrossRef]
14. MacQueen, J.B. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Davis Davis, CA, USA, 21 June–18 July 1965; 27 December 1965–7 January 1966.
15. Steinbach, M.; Karypis, G.; Kumar, V. A Comparison of Document Clustering Techniques. In Proceedings of the KDD Workshop on Text Mining, Boston, MA, USA, 20–23 August 2000; 400, pp. 525–526.
16. Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, OR, USA, 2–4 August 1996; pp. 226–231.
17. Defays, D. An efficient algorithm for a complete-link method of hierarchical clustering. *J. ACM* **1973**, *20*, 634–638.
18. von Luxburg, U. A tutorial on spectral clustering. *Stat. Comput.* **2007**, *17*, 395–416. [CrossRef]
19. Dempster, A.P.; Laird, N.M.; Rubin, D.B. Maximum likelihood from incomplete data via the em algorithm. *J. R. Stat. Soc. Ser. B* **1977**, *39*, 1–22.
20. Witten, I.H.; Paynter, G.W.; Frank, E.; Gutwin, C.; Nevill-Manning, C.G. Kea: Practical automatic keyphrase extraction. In Proceedings of the fourth ACM conference on Digital libraries, Berkeley, CA, USA, 11–14 August 1999.
21. Mihalcea, R.; Tarau, P. TextRank: Bringing Order into Text. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, 25–26 July 2004; pp. 404–411.
22. Wan, X.; Xiao, J. Single Document Keyphrase Extraction Using Neighborhood Knowledge. In Proceedings of the National Conference on Artificial Intelligence, Chicago, IL, USA, 13–17 July 2008.
23. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
24. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **2020**, *21*, 5485–5551.
25. Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; Zettlemoyer, L. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv* **2019**, arXiv:1910.13461.
26. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
27. Lee, D.D.; Seung, H.S. Learning the parts of objects by non-negative matrix factorization. *Nature* **1999**, *401*, 788–791. [CrossRef]
28. Blei, D.M. Probabilistic topic models. *Commun. ACM* **2012**, *55*, 77–84. [CrossRef]
29. Xu, W.; Liu, X.; Gong, Y. Document Clustering Based on Non-negative Matrix Factorization. In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Toronto, ON, Canada, 28 July–1 August 2003; pp. 267–273.
30. Gillis, N. The why and how of nonnegative matrix factorization. *Connections* **2014**, *12*, 257–291.
31. Dieng, A.B.; Ruiz, F.J.R.; Blei, D.M. Topic Modeling in Embedding Spaces. *Trans. Assoc. Comput. Linguist.* **2020**, *8*, 439–453. [CrossRef]
32. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692.
33. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv* **2019**, arXiv:1910.01108.
34. Muennighoff, N.; Tazi, N.; Magne, L.; Reimers, N. MTEB: Massive Text Embedding Benchmark. *arXiv* **2022**, arXiv:2210.07316. [CrossRef]
35. Reimers, N.; Gurevych, I. Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation. *arXiv* **2020**, arXiv:2004.09813.
36. Song, K.; Tan, X.; Qin, T.; Lu, J.; Liu, T.Y. Mpnet: Masked and permuted pre-training for language understanding. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 16857–16867.
37. Cer, D.; Diab, M.; Agirre, E.; López-Gazpio, I.; Specia, L. SemEval-2017 Task 1:Semantic Textual Similarity-Multilingual and Cross-lingual Focused Evaluation. In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), Vancouver, BC, Canada, 3–4 August 2017; pp. 1–14.
38. Marelli, M.; Bentivogli, L.; Baroni, M.; Bernardi, R.; Menini, S.; Zamparelli, R. SICK through the SemEval glasses. Lesson learned from the evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), Reykjavik, Iceland, 26–31 May 2014.
39. Aitao, W. TREC QA track overview. *TREC* **2003**, *3*, 16–26.
40. Shankar, I. First Quora Dataset Release: Question Pairs. *Quora Engineering Blog*, 24 January 2017.

41. Thorndike, R.L. Who belongs in the family? *Psychometrika* **1953**, *18*, 267–276. [CrossRef]
42. Schwarz, G. Estimating the dimension of a model. *Ann. Stat.* **1978**, *6*, 461–464. [CrossRef]
43. Ng, A.Y.; Jordan, M.I.; Weiss, Y. On spectral clustering: Analysis and an algorithm. *Adv. Neural Inf. Process. Syst.* **2001**, *14*, 849–856.
44. Shlens, J. A tutorial on principal component analysis. *arXiv* **2014**, arXiv:1404.1100.
45. Wang, W.; Zhang, Z.; Guo, J.; Dai, Y.; Chen, B.; Luo, W. Task-Oriented Dialogue System as Natural Language Generation. *arXiv* **2021**, arXiv:2108.13679.
46. Tarjan, R.E. Efficiency of a Good But Not Linear Set Union Algorithm. *J. ACM* **1975**, *22*, 215–225. [CrossRef]
47. Ratcliff, J.E.; Obershelp, J.H. A pattern-matching program that saves time listing all occurrences of a pattern. *Commun. ACM* **1988**, *31*, 776–779.
48. Hubert, L.; Arabie, P. A Comparison of External Validity Indices for Clustering. *Psychol. Bull.* **1985**, *98*, 238.
49. Rousseeuw, P.J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **1987**, *20*, 53–65. [CrossRef]
50. Calinski, T.; Harabasz, J. A dendrite method for cluster analysis. *Commun. Stat. -Theory Methods* **1974**, *3*, 1–27. [CrossRef]
51. Likert, R. A Technique for the Measurement of Attitudes. *Arch. Psychol.* **1932**, *22*, 1–55.