DEEP LANGUAGE MODELS FOR TEXT REPRESENTATION IN DOCUMENT CLUSTERING AND RANKING

by

Sima Rezaeipourfarsangi

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
November 2023

Table of Contents

List of Tables

List of Figures

List of Abbreviations Used

NLM Neural Language Model

LLM Large Language Model

NLP Natural Language Processing

GPT Generative Pre-trained Transformer

BERT Bidirectional Encoder Representations from Transformers

LightGBM Light Gradient Boosting Machine

ARIMA Autoregressive Integrated Moving Average

WCSS Within-cluster Sum of Squares

T-SNE T-Distributed Stochastic Neighborhood Embedding

ML Machine Learning

UMAP Uniform Manifold Approximation and Projection

K-NN K-Nearest Neighbor

RMSE Root Mean Square Error

MSE Mean Squared Error

WMAPE Weighted Mean Absolute Percentage Error

DNN Deep Neural Network

RNN Recurrent Neural Network

CNN Convolutional Neural Network

LSTM Long Short-Term Memory

BOW Bag of Word

ARS Adjusted Rand Score

LDA Latent Dirichlet Allocation

NMF Negative Matrix Factorization

LDC Lexical Double Clustering

AMI Adjusted Mutual Information

NMI Normalized Mutual Information

ARI Adjusted Rand index

StDev Standard Deviation

IR Information Retrieval

STS Semantic Text Similarity

TF-IDF Term Frequency-Inverse Document Frequency

WOA Whale Optimization Algorithm

KHM K-Harmonic Means

EWOA Enhanced Whale Optimization Algorithm

NER Named-Entity Recognition

MALNIS Machine Learning and Networked Information Spaces

SNN Siamese neural network

Abstract

Deep language models have become increasingly prominent in the field of machine learning. This thesis explores the potential of deep language models for text representation and their role in specific text mining applications such as interactive document clustering, sales forecasting, and document ranking. First, in interactive document clustering, we leverage deep language models and present a novel system that replaces key-term-based clustering with deep language models, allowing users to steer the clustering algorithm based on their domain knowledge through the system. Second, we introduced a novel approach for improving new product sales forecasting by incorporating product descriptions as an additional feature. By clustering products based on description similarity and using time series data from similar products, demand prediction is enhanced. Deep language models are utilized, along with dimensionality reduction methods. Cluster descriptions are obtained using Top2Vec, and new product forecasts are made based on historical sales data of related clusters and previously introduced products. Third, in document ranking, we proposed a novel approach for ranking resumes based on their similarity to specific job descriptions. By employing Siamese neural networks with integrated components like Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and attention layers, the model captures sequential, local, and global patterns to extract features and represent the documents. Deep language models are employed to encode the documents, serving as input for the network. Utilizing deep language models, the model achieves improved accuracy in document ranking and enhances the matching process between job descriptions and resumes, surpassing other comparative models. The versatility of deep language models arises from their ability to learn from vast amounts of text data, allowing them to extract meaningful patterns and insights. In our research, we utilized state-of-the-art deep language models such as SBERT, RoBERTa, Universal sentence encoder, Infer-Sent, and BigBird.

Acknowledgements

I would like to express my deepest gratitude and appreciation to all those who have supported and guided me throughout this journey. I would like to express my sincere gratitude to Dr. Milios for his invaluable guidance and support throughout my Ph.D. journey. I am also thankful to my family for their unwavering love, encouragement, and understanding throughout this challenging journey. I am indebted to the funding agencies and organizations that have provided financial support for my research. Their support has enabled me to pursue my studies and undertake research activities that would not have been possible otherwise.

Chapter 1

Introduction

Initially, we provide a concise overview of text representation methods and approaches to document embedding using deep language models. Document embeddings and language models are powerful tools in the field of Natural Language Processing (NLP) that enable the understanding and analysis of textual data, representing documents, such as sentences, paragraphs, or entire documents, as a vector. They are mapping data points into a lower-dimensional space, where each point is represented by a vector of continuous values. These embeddings capture the underlying semantic meaning and contextual details of the text, enabling more effective information retrieval, text classification, and similarity calculations. Language models are trained on vast amounts of text data to learn the patterns and relationships within the language. They are capable of generating coherent and contextually appropriate text based on the input provided. Language models have revolutionized various NLP tasks, including machine translation, text generation, sentiment analysis, and question-answering. By leveraging document embeddings and language models, we can unlock deeper insights and develop more advanced applications in the realm of language understanding and text processing. Various methods for generating sentence embeddings are proposed by Bengio in [11]. Standard sentence embedding models typically employ techniques such as averaging individual word embeddings for each sentence or document. However, in these approaches, the contextual information and word sequence within the sentence are disregarded, resulting in the loss of important linguistic nuances. In response to this concern, deep language models emerged as an alternative approach.

1.1 Deep Language Models

Large Language Model (LLM) are neural network models that have been trained on vast amounts of text data. Having an extensive number of parameters, up to

billions, enables these models to acquire knowledge of semantic and syntactic connections between words by processing vast amounts of text data. As a result, they effectively process textual information, enabling them to excel in diverse tasks such as document clustering, sales forecasting, and document ranking. Deep language models, such as Bidirectional Encoder Representations from Transformers (BERT) [27] and Generative Pre-trained Transformer (GPT) [77], demonstrate significant improvement over alternative NLP algorithms across various text mining tasks. These models are built upon transformer architectures, which allow them to capture the intricate relationships and contextual dependencies within the text. In this thesis, we will explore the potential of deep language models and their role in the above-mentioned applications.

1.1.1 Different Generations of Deep Language Models

There is no rigidly defined category for various generations of deep language models, and it can fluctuate depending on different perspectives [117]. In the following, we enumerate several generations of deep language models that have advanced the field of NLP.

1. First generation: In a vector space model, every word is depicted by a real-valued vector. Two model architectures, namely continuous Bag of Word (BOW) and continuous skip-gram, are utilized to generate word vectors [66]. The concept of representing words as low-dimensional vectors has a rich and extensive history [63]. There are some pioneering approaches to vector space modelling that have paved the way for advancements in this field, such as Glove [74], Word2vec [66] and Doc2vec [53]. Word vectors of unstructured text data were introduced by Mikolov [67]. He later extended his original skip-gram model to predict surrounding words in a sentence or a document [66].

2. Second generation: Transformer-based models, including BERT, marked a significant advancement in NLP. These models introduced attention mechanisms, enabling efficient parallel processing and capturing global context information. Transformers achieved state-of-the-art performance on various

tasks.

3. Third generation: GPT models introduced the concept of pre-training large-scale language models on massive amounts of data. GPT models demonstrated remarkable capabilities in generating coherent text and achieving outstanding performance in language understanding tasks.

4. Fourth generation: Fine-tuned for conversational tasks such as LLaMa2, GPT4 and Claude. These specialized models are designed to engage users in meaningful, context-aware interactions, providing more personalized and relevant responses.

5. Ongoing advancements: Ongoing research and development continue to bring forth new and improved models, incorporating techniques such as self-supervised learning, transfer learning, unsupervised pre-training and prompt engineering.

1.2 Text Mining Applications

Text mining utilizes NLP techniques and algorithms to process, analyze, and derive meaningful knowledge from textual sources. While the specific details may vary, the general process typically includes the following steps illustrated in Fig. 1.1. The

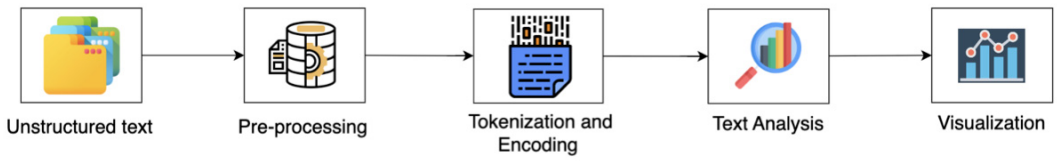

Figure 1.1: Comprehensive overview of the text mining workflow.

primary emphasis of this thesis lies in the encoding stage, utilizing deep language models for extracting features. By replacing traditional approaches with deep language models in text mining applications, we witness a significant leap in performance and accuracy. These models demonstrate exceptional proficiency in capturing intricate linguistic patterns, understanding context, and representing

semantic information, thereby facilitating more advanced analysis and interpretation of textual data. The underlying similarity among all the aforementioned applications lies in the utilization of deep language models as an alternative to traditional approaches. These models, including SBERT, RoBERTa, Universal Sentence Encoder, Infer-Sent, and BigBird, play a central role in improving the results of the applications. The incorporation of deep language models in these projects highlights their adaptability and efficacy in addressing diverse text mining challenges.

1.2.1 Interactive Document Clustering

The first application is interactive document clustering. Clustering is a crucial text mining technique for organizing digital document sets, enabling users to understand their data better. It has been demonstrated that involving users can often significantly improve clustering quality [6]. We propose a novel system that combines deep language models with interactive clustering enabling users to steer the clustering algorithm towards results meaningful to them through an interactive document and cluster visualizations. Our system is comprised of several visual components, each of which allows the user to apply their domain knowledge to the clustering process. The use of deep language models for representing sentences addresses the vocabulary mismatch problem that affects bag-of-words representations of documents. We employ sentence embeddings to obtain document embeddings as an input to the clustering algorithm, a modified version of Kmeans, considering the characteristics of document embeddings. Kmeans clustering is a popular choice for vector embeddings and continuous data in general due to its simplicity and efficiency. Here are some reasons why Kmeans might be suitable for our clustering task:

- Euclidean Distance Metric: Kmeans relies on the Euclidean distance metric to measure the similarity between data points. In vector embeddings, the Euclidean distance often makes sense because it captures the notion of similarity between vectors in a straightforward manner.
- Scalability: Kmeans is computationally efficient and scales well with the

number of data points, making it suitable for large datasets, which is often the case with embeddings generated from extensive corpora.

- Ease of Use: Kmeans is easy to understand and implement, which is advantageous for quick experimentation and prototyping.

Nevertheless, it's crucial to emphasize that the selection of a clustering algorithm depends on both the data's characteristics and our particular objectives. Below, we enumerate alternative clustering algorithms that we could employ in our research:

1. Density-Based Clustering: Density-based methods like DBSCAN or HDBSCAN are suitable when embeddings represent data with irregularly shaped clusters or varying densities. These methods do not require specifying the number of clusters in advance and can identify noise points.

2. Graph-Based Clustering: Graph-based methods like spectral clustering or hierarchical clustering can be effective when embeddings have an underlying graph structure, or when we want to capture hierarchical relationships between clusters.

3. Characteristics of Embeddings: When considering whether to use a partitioning approach like Kmeans, it's important to examine the characteristics of embeddings. If embeddings exhibit clear separations in vector space and clusters that can be well-defined by centroid-based partitioning, Kmeans might be a good choice. If the structure is more complex, density-based or graph-based methods might be more appropriate.

4. Dimensionality: The dimensionality of embeddings can also impact our choice. High-dimensional embeddings may require dimensionality reduction techniques before applying clustering, and the choice of clustering algorithm can depend on the reduced dimensionality.

We conduct a two-stage evaluation of our system. First, we evaluate the proposed clustering models in automatic clustering of various publicly available datasets, and we confirm that they are competitive with state-of-the-art. Second, we conduct a formal expert study of a specific dataset consisting of our research group's readings

(research papers in machine learning, text mining, and NLP) over several years. The domain expert is a graduate student whose thesis is in the above field. The expert study concludes that our system is significantly better at producing meaningful clusters than the baseline system (Vis-Kt) [93, 91].

1.2.2 Demand Forecasting

The second application is sales forecasting. It is challenging to predict the demand for new products due to the unavailability of historical sales data. A company's operational decisions depend on accurate forecasting. Poor forecasting can damage a company's reputation by causing customer dissatisfaction due to supply chain disruptions, stockouts, or unfulfilled orders. Additionally, inaccurate sales forecasts can result in financial problems such as excessive inventory costs, missed revenue targets, and inefficient resource allocation. These consequences can erode customer trust, impede business growth, and create financial instability for the organization. Demand forecasting has traditionally relied on historical sales data without considering the product description as a feature. We proposed a novel approach by incorporating product descriptions as an additional feature to improve new product sales forecasting since traditional demand forecasting only uses time series data. This approach involves clustering products based on the similarity of their descriptions and only considers time series data from similar products to predict demand. The product descriptions are encoded using different language models and embeddings such as RoBERTa, DistilBERT and SimCSE. We employed two commonly used methods for dimensionality reduction, namely Uniform Manifold Approximation and Projection (UMAP) and T-Distributed Stochastic Neighborhood Embedding (T-SNE), to transform encoded product descriptions into a 2D space. Subsequently, we applied either HDBSCAN or Kmeans. A Top2Vec model is trained on the product descriptions within each cluster, and the resulting topics were utilized as cluster descriptions as a novel feature in our model. We computed the mean of the embeddings for each cluster/topic and employed cosine similarity to allocate the description of each new product to a corresponding cluster. Between cosine and Euclidean distance, we opted for cosine similarity due to its suitability for measuring text-based document embeddings, emphasizing term

similarity and robustness to variations in document lengths. However, Euclidean distance could also serve as an alternative choice. The forecast is made based on historical sales data of previously introduced (current) products and new products, using state-of-the-art forecasting models including Light Gradient Boosting Machine (LightGBM) regressor, Catboost, and Facebook Prophet. According to our study, employing language models to cluster current product descriptions and allocate new products to those clusters enhances the accuracy of sales forecasting of new products. The research employs a publicly available dataset of sales data from an online retailer based in the UK, obtained from Kaggle, and evaluates the performance using metrics such as Root Mean Square Error (RMSE), Weighted Mean Absolute Percentage Error (WMAPE), Bias, and standard deviation.

1.2.3 Document Ranking

Document ranking is the third application which is a fundamental task in information retrieval, prioritizing relevant documents based on a user's query. Deep language models have emerged as valuable tools for document ranking by providing a deeper understanding of text context. In the realm of job searching, online matching engines are indispensable for both job seekers and employers. However, a poorly designed matching system can disadvantage competent candidates. While humans can easily match resumes to job descriptions, as job requirements become more specialized and applicants more skilled, manual resume matching by humans becomes more challenging. This is where a well-designed system can expedite decision-making processes while ensuring enhanced accuracy. We treated resumes and job descriptions as documents and calculate their similarity to determine the suitability of applicants. The objective is to rank a set of resumes based on their similarity to a specific job description. We employ Siamese Neural Networks, comprised of identical sub-network components, to evaluate the semantic similarity between documents. Our novel architecture integrates various neural network architectures, where each sub-network incorporates multiple layers such as CNN, LSTM and attention layers to capture sequential, local and global patterns within the data. The LSTM and CNN components are applied concurrently and merged together. The resulting output is then fed into a multi-head attention layer. These

layers extract features and capture document representations. The extracted features are then combined to form a unified representation of the document. We leverage deep language models like SBERT, BigBird, and RoBERTa to obtain embeddings for each resume and job description, which serve as a lower-dimensional representation of our input data. The model is trained on a private dataset of 268,549 real resumes and 4,198 job descriptions from twelve industry sectors, resulting in a ranked list of matched resumes. We performed a comparative analysis involving our model, Siamese CNN (S-CNNs), Siamese LSTM with Manhattan distance, and a BERT-based sentence transformer model. By combining the power of language models and the novel Siamese architecture, this approach leverages both strengths to improve document ranking accuracy and enhance the matching process between job descriptions and resumes. Our experimental results demonstrate that our model outperforms other models in terms of performance.

1.3 Objectives

The aim of this research is to investigate how the utilization of various document representation and deep language models, affects text mining tasks, with a particular focus on their influence on document clustering and ranking. By leveraging the capabilities of these models to comprehend context and represent text, we can unveil hidden patterns and connections within extensive text datasets, thus gaining deeper insights into the data. Using LLM as a document representation method offers a compelling opportunity to augment the efficiency of the clustering and ranking results. However, a significant research gap exists in understanding the optimal methodologies and fine-tuning techniques required for LLM to excel in this specific domain. Current approaches often employ traditional representation models such as key term-based clustering, but there is a need for more comprehensive investigations. Additionally, while LLM have shown remarkable performance on text classification tasks, their adaptability to clustering is not yet fully explored. Addressing this research gap entails developing novel approaches in document clustering and ranking using different representation models. In the following, we outline the specific goals and objectives of this thesis, followed by the research gaps associated with each and how this work contributes to closing them.

1. Investigate the effectiveness of deep language models as text representation models in improving the accuracy and efficiency of interactive document clustering systems:

    - Research Gap: Lack of user-centric clustering based on LLM in document clustering task: Traditional document clustering methods often rely on key-term-based approaches, which may not fully capture the user's domain-specific knowledge and preferences.
    - Contribution: The thesis introduces a novel system that replaces key-term-based clustering with deep language models, allowing users to steer the clustering algorithm based on their domain knowledge. This user-centric approach fills the gap by enhancing the effectiveness and relevance of document clustering.

2. Explore the potential of deep language models in enhancing sales forecasting tasks by encoding product descriptions using LLM and clustering the product based on the similarity of their descriptions:

    - Research Gap: Insufficient incorporation of product descriptions and representing them using LLM in sales forecasting: Existing sales forecasting techniques may not fully leverage product descriptions, potentially missing valuable information.
    - Contribution: The research proposes a novel approach that incorporates product descriptions as an additional feature for sales forecasting. By clustering products based on description similarity and utilizing deep language models, this work bridges the research gap, leading to more accurate demand predictions for new products.

3. Assess the impact of deep language models on document ranking, examining their ability to extract relevant information and rank documents/queries accurately. As a use case, sort resumes according to their similarities to a particular job description:

    - Research Gap: Traditional resume ranking methods often lack the ability to capture nuanced similarities between job descriptions and resumes,

resulting in suboptimal matching.

- Contribution: The thesis introduces a novel approach for ranking resumes using Siamese neural networks with integrated components like CNN, LSTM, and attention layers. By encoding documents with deep language models, this work significantly improves accuracy in document ranking, effectively addressing the gap in accurate job matching. After comparing our approach with different benchmarks, we achieved noticeably better results, underscoring the effectiveness of our method in enhancing document ranking precision for job matching.

4. Maximizing the potential of deep language models:

   - Research Gap: Although deep language models are versatile and powerful, there is a gap in effectively leveraging them for various text mining tasks.
   - Contribution: The research leverages state-of-the-art deep language models such as SBERT, RoBERTa, Universal sentence encoder, Infer-Sent, and BigBird across document clustering and ranking applications. By demonstrating their utility and effectiveness in these tasks, the thesis fills the gap by showcasing the potential of these models in practical applications.

5. Bridge between theory and practice:

   - Research Gap: There exists a notable disparity between theoretical foundations and practical application in this research domain.
   - Contribution: While emphasizing practical applications, the thesis also underscores the theoretical foundations behind the use of deep learning models in text mining. It bridges the gap between theoretical principles and real-world implementation, providing a valuable reference for researchers and practitioners in the field.

In summary, this thesis contributes to the field by addressing research gaps related to user-centric document clustering, enhanced sales forecasting through product

descriptions, improved document ranking for job matching, and the effective utilization of deep language models. It demonstrates how these models can be harnessed to improve the performance and efficiency of various document clustering and ranking applications, ultimately advancing the state-of-the-art in the field.

1.4 Practical Recommendations

Below, we offer a few pieces of advice for researchers seeking to employ deep language models in the context of text mining tasks.

- Understand the problem: Acquire a comprehensive understanding of the specific text mining task at hand, encompassing its inherent challenges, objectives, and desired outcomes. This foundational comprehension serves as a basis for picking the most appropriate deep language model and designing an effective approach.

- Choose the right model: Explore and evaluate different deep language models based on their architecture, capabilities, and pre-training objectives. Models like BERT, and GPT are popular choices, but consider factors such as task compatibility, available resources, and the nature of the text data when selecting the model.

- Pre-training and fine-tuning: Deep language models are often pre-trained on large corpora of text data, but fine-tuning on domain-specific or task-specific data is crucial for optimal performance. Determine whether you have enough labelled data for fine-tuning or consider transfer learning techniques to adapt pre-trained models to your task.

- Data preparation and preprocessing: Properly preprocess the text data by cleaning, normalizing, and tokenizing it to ensure compatibility with the chosen deep language model. Pay attention to factors like input length limitations, special characters, and language-specific requirements.

- Experiment and iterate: Experiment with different configurations, hyperparameters, and techniques to optimize the performance of your deep

language model. Continuously evaluate and iterate on your approach based on performance metrics and feedback to achieve the best results.

- Evaluate and interpret results: Assess the performance of your deep language model using appropriate evaluation metrics specific to your text mining task.
- Combine different models: To improve the accuracy and get better results try to combine different architectures of deep networks with different language models.
- Integrate diverse models: Enhance accuracy and achieve superior results by integrating distinct deep network architectures with various language models, thereby capitalizing on the strengths offered by their combination.

In this thesis, our overarching objective is to emphasize the importance of integrating LLM into document representation to enhance the outcomes of document clustering and ranking tasks significantly. Through the substitution of conventional key term-based representations with LLM, we have achieved notable enhancements in clustering results. Within the document ranking process, employing the fine-tuned LLM to tokenize and encode queries and documents serves to extract relevant information, thereby enhancing the quality of ranking outcomes. The recommendations in the thesis involve iterative experimentation and evaluation to improve performance in document clustering and ranking tasks, which is a common approach in machine learning and deep learning research. However, there are general principles and insights as to why deep learning models are well-suited for these tasks in principle:

1. Feature representation learning: Deep learning models excel at automatically learning meaningful representations from raw data. In text mining tasks, such as document clustering, sales forecasting, and document ranking, deep language models can capture complex patterns and relationships in text data. This is crucial because traditional methods often rely on handcrafted features, which may not capture the intricate nuances present in natural language.
2. Hierarchical and sequential information: Text data often contains hierarchical and sequential information. Deep learning models, particularly Recurrent

Neural Network (RNN) and CNN can process sequences and hierarchies effectively. For example, in document ranking, capturing sequential and local patterns in resumes and job descriptions is essential for accurate matching, and deep learning models are capable of doing this.

3. Contextual understanding: Deep language models, like BERT and RoBERTa, are pretrained on massive text corpora and are capable of understanding the contextual meaning of words and phrases. This contextual understanding is highly valuable in tasks like document clustering, where the relevance of documents may depend on nuanced language and context-specific information.

4. Transfer learning: Many deep learning models are pretrained on large, diverse datasets before being fine-tuned for specific tasks. This transfer learning approach allows them to leverage knowledge gained from one task (e.g., language modelling) to excel in related tasks (e.g., document ranking). This transfer of knowledge can significantly boost performance, even with limited task-specific data.

5. Flexibility and adaptability: Deep learning models are highly adaptable to different domains and tasks. They can be fine-tuned or customized to suit specific requirements, making them versatile tools for a wide range of text mining applications.

6. Scalability: Deep learning models can handle large-scale data efficiently, which is especially important in text mining, where datasets can be massive. This scalability ensures that these models can be applied to real-world datasets with millions of documents or data points.

In principle, deep learning models offer a powerful way to address text mining challenges by virtue of their ability to automatically learn complex representations, capture contextual information, and adapt to different tasks. While the thesis demonstrates their effectiveness through iterative experimentation, these general principles highlight why deep learning models are well-suited for text mining tasks in theory, providing a solid foundation for their application in practice.

1.5 List of Publications

- Sima Rezaeipourfarsangi, Ningyuan Pei, Ehsan Sherkat, Evangelos E. Milios: Interactive clustering and high-recall information retrieval using language models, Proceedings of the 2022 International Conference on Advanced Visual Interfaces (AVI 2022), June 6–10, 2022, Frascati, Rome, Italy, Pages 1–5, https://dl.acm.org/doi/abs/10.1145/3531073.3531174.

- Eric M. Cabral, Sima Rezaeipourfarsangi, Maria Cristina F. Oliveira, Evangelos E. Milios, Rosane Minghim: Addressing the gap between current language models and key-term-based clustering ACM Symposium on Document Engineering 2023 (DocEng '23), August 22–25, 2023, Limerick, Ireland, https://dl.acm.org/doi/10.1145/3573128.3604900.

- Sima Rezaeipourfarsangi, Evangelos E. Milios: AI-powered Resume-Job matching: A document ranking approach using deep neural networks ACM Symposium on Document Engineering 2023 (DocEng '23), August 22–25, 2023, Limerick, Ireland, https://dl.acm.org/doi/10.1145/3573128.3609347.

- Sima Rezaeipourfarsangi, Evangelos E. Milios: SciNLP 2021: 2nd Workshop on Natural Language Processing for Scientific Text, October 8, 2021, https://scinlp.org/#accepted-abstracts.

- Sima Rezaeipourfarsangi, Behrouz Haji Soleimani, Seyednaser Nourashrafeddin, Hossein Ghomeshi, Evangelos E. Milios: Demand forecasting of new products using language models on the product descriptions Journal paper submitted in Knowledge-based systems (under review): https://www.sciencedirect.com/journal/knowledge-based-systems.

In the following chapters, we will delve deeper into the specific methodologies and techniques employed when utilizing deep language models for each of the above-mentioned tasks (interactive clustering, document ranking and demand forecasting). We will explore the challenges and opportunities presented by deep language models, discuss best practices, and highlight the potential impact of these applications across various industries. By harnessing the power of deep language

models, we can unlock new possibilities for improving information retrieval, business forecasting, and text analysis.

Chapter 2

Interactive Document Clustering And High-Recall Information Retrieval Using Language Models

2.1 Introduction

Clustering is one of the main methods of unsupervised learning in the field of textual data. There are several clustering algorithms for placing documents in similar groups using different similarity measures[1]. However, automatic clustering, which best matches user goals, is not always possible, even with advanced clustering algorithms. Unsupervised techniques generate clusters, which often do not reflect the user's point of view regarding correct clustering. One way to overcome this problem is to engage humans in the clustering cycle. In this approach, users can generate clusters tailored to specific application domains and continuously modify them based on their preferences. We applied an interactive document clustering platform to involve the user in the clustering process and asked the domain expert to use our system (LM-Kt) to cluster her documents. We modified the clustering algorithm combined with the visual interface. The user interface is provided by a set of visual functionalities which assist the user in understanding and exploring the dataset and adjusting the clustering process. To conduct the expert study, we collected the dataset of scientific research papers in machine learning, text mining, and NLP. The data archived in Zotero [1] in our research group's readings over several years. The domain experts are selected from computer science graduate students in the above-mentioned fields. To validate the usefulness of the proposed system, we conducted a formal user study, which will be explained in the following sections in detail. The reported results demonstrate that the LM-Kt outperforms its predecessors and depict how visualizing user interactions with the corpus helps the user obtain insights into the corpus's content. Below we provide a summary of the

[1]This chapter is the extended version of the conference paper, presented at AVI2022: https://dl.acm.org/doi/abs/10.1145/3531073.3531174

research gaps presented in this chapter, followed by an exploration of the contribution related to each research gap and an examination of how this research serves to address and mitigate these gaps.

1. A novel document clustering algorithm:

    - Research gap: Investigating the effectiveness and advantages of using deep language model-based document representations compared to key term-based models as traditional methods, and identifying potential limitations.
    - Contribution: The proposed system: (a) uses deep language model-based document representations (b) a new objective function (c) a novel initialization of the number of clusters.

2. User interaction:

    - Research gap: Identifying the extent to which user interaction with deep language model-based representations enhances the clustering process and the user experience, and understanding the practical challenges and opportunities in implementing this interaction.
    - Contribution: Enabling user interaction with deep language model-based document representations in interactive clustering.

3. Expert study:

    - Research gap: Identifying the domain-specific factors that influence the performance of the LM-Kt model, as well as the potential variations in its effectiveness across different domains and contexts, and exploring strategies to address these domain-specific challenges.
    - Contribution: Evaluating the LM-Kt in a specific domain via an expert study.

4. Clustering algorithm evaluation:

    - Research gap: Assessing the performance and generalizability of the modified clustering algorithm without user interaction when applied to

standard datasets, and comparing its results with existing clustering methods to understand its strengths and weaknesses in typical scenarios.

- Contribution: Evaluate the modified clustering algorithm without interaction using deep language models on standard datasets.

2.2 Related Work

An interactive clustering system captures all necessary interactions between the user and the system. For the user to modify the final clustering results, various approaches have been introduced. In this research, we divide these different ways into three main groups as follows;

2.2.1 Interacting With The Clustering Result

In this type of interactive system, which is the most common, the user interacts with the clustering results through meaningful operations. Operations are different, although users are often expected to correct errors in clustering results. Based on these corrections by the user, the model receives hints about user preferences. In the end, analyzing this feedback leads to better overall results. In this approach, the domain expert directs the clustering process to improve the results, and there is no need to understand the internal performance of the algorithm. Such examples allow the user to move data samples from one cluster to another [8, 18]. This approach method, known as distance learning, involves training a model or algorithm to understand the relationships between data points in a way that reflects their true underlying similarities or differences. Users interact with the system by updating the similarity measure in [111]. This system allows users to interact with node-link diagrams, adjacency matrices, and tree-maps to filter clusters.

2.2.2 Interacting With The Model

The second group is similar, except that the interaction occurs at model or algorithm parameters instead of clustering results. In this type of research, the user would modify the specific parameters of the model to achieve the desired effect. The initial clustering is first executed and visualized. Then users can tweak the

parameters and re-run the clustering. In a few cases, the user updates the weights, data instances, or different features in her feedback [24].

2.2.3 Requesting Information From Users

In the third group, interactions with both the model and the results are allowed. Many papers in this group focus on presenting initial results to the user and then letting them interactively modify the clustering. For example, users can retouch a cluster by manipulating the weights dedicated to specific terms within various topics. They can be provided with other features such as delete, merge, move, re-cluster, and sub-cluster the results. Vis-Kt is a recent interactive clustering system [93, 91] based on key terms. They employ key terms as the basic semantic units for interaction. The user can explore the dataset to identify her desired clusters of terms indicative of relevant topics and guide the document clustering by providing good seeds. This research utilizes Vis-Kt as our baseline interactive clustering system and replaces BOW with deep language models. Models like BERT and RoBERTa have proven highly effective in grasping intricate connections among words, contextual cues, and the semantic significance within textual information. In contrast, BOW is a more simplistic representation that treats words independently, disregarding context and semantics. The gap that needs to be addressed is the inherent limitation of BOW in capturing the rich and contextually relevant information present in the text, especially in interactive clustering tasks. By introducing deep language models, the goal is to bridge this gap by enabling a more sophisticated and context-aware representation of text, which can significantly enhance the quality and precision of interactive clustering in Vis-Kt.

2.3 Methodology

Interactive document clustering systems consist of three different parts: An interactive clustering algorithm, document representation methods, and an interactive user interface (UI). The main focus of this chapter is on the interactive visualization part, and we use the interactive user interface system based on language models, LM-Kt. In the following, we will discuss each of the above-mentioned parts in detail.

2.3.1 Clustering Algorithm

In this study, we modified the Kmeans [47] and K-Harmonic Means (KHM) [115] clustering algorithms. KHM is a center-based clustering algorithm, and as a component of its performance function, it uses the harmonic averages of the distances from each data point to the centers. We applied an optimization method for the KHM algorithm called Whale Optimization Algorithm (WOA) developed by Mirjalili in [68]. WOA mimics the hunting behaviour of humpback whales to solve complex continuous problems and is a swarm-based stochastic optimization algorithm. A swarm refers to a number of potential solutions to the optimization problem and each potential solution is referred to as a search agent. The goal is to find the best evaluation of a given objective function, by identifying the position of the search agent.

In the context of clustering, assume that the search agent represents k cluster centers (k = number of clusters). Each search agent X_i is generated as follows: $X_i = (z_{i1}, z_{i2}, \ldots, z_{ik})$. where z_{ij} refers to the j_{th} cluster center vector of the i_{th} search agent in cluster c_{ij}. Thus, a swarm represents a set of clustering candidates for the dataset vectors. The fitness function is the intra-distance of clusters, measures the distance between cluster center and data vectors of the same cluster, calculated by the sum of the mean squared Euclidean distance. In the following, we discuss this algorithm in detail.

WOA/ EWOA

The inspiration for this method is the behaviour of humpback whales, as they prefer to chase a school of krill or small fish near the water's surface. Bubble-net feeding is a technique to catch the prey by generating a bubble around the prey and then swimming back to the surface. WOA has adopted the spiral bubble-net feeding manoeuvre as its primary strategy, which is the third step in $WOA's$ three-step algorithm for finding prey. In WOA, it is assumed that the current best candidate solution is the target prey in the search space that has not been previously known or is close to the optimum in terms of performance. Humpback whales use this to practice hunting for prey by simulating the process of locating and encircling their prey. After the best search agent is determined, the new agents will update their

positions across the best search agent.

The initial population of solutions is filled by the number of n-dimensional vectors with real values which have been randomly generated. Configurations of KHM, optimized by Enhanced Whale Optimization Algorithm (EWOA), are the hyper-parameters of the clustering algorithm which are maximum iterations, the minimum number of data points, and the initial centroids of KHM that significantly influence the final clustering result. So far, we have an optimization problem that determines these two parameters (P_1 and P_2) and the initial centroids (P_3, P_4, ...) in algorithm 1. In other words, each individual is a configuration set for clustering. To evaluate each individual and make EWOA iterations, the fitness function is defined as the sum of the mean squared distance of points to the corresponding center in a clustering made by that configuration (individual in EWOA). We use a subset of data points for less time overhead to evaluate the clustering and finally apply the best individual on the whole data. According to the WOA definition of *Exploitation phase*, humpback whales use the bubble-net strategy to attack their prey. This method can be summarised as follows:

1. *Shrinking encircling mechanism*: This is accomplished by decreasing the value of A, where A is a random value in the interval $[-1, 1]$.

2. *Spiral updating position*: A spiral equation is constructed between the position of the whale and prey to simulate the helix-shaped movement of humpback whales.

In the optimization model, the simultaneous behaviour of humpback whales pursuing the prey within a shrinking circle and along a spiral-shaped path simultaneously is modeled by choosing between either the shrinking encircling mechanism or the spiral model with a probability of 50% to update the position of whales during optimization.

Humpback whales search for prey following two behaviours, random search and the bubble-net procedure. The bubble-net procedure implementing the search for prey, where prey is a clustering (exploration phase). The exploration phase depends on the update of the position of a search agent by selecting a random search agent (a random whale) instead of the best search agent to move far away from a reference

whale. This procedure with the random values affirm the exploration. This exploration gives the WOA algorithm the ability to avoid getting trapped in the local minimum and achieves a global search. This is why we chose this algorithm to prevent the user from being trapped in local minimum during the interactive clustering.

EWOA [17] is an improvement made on WOA, which combines two strategies at the same time. The distinction between WOA and EWOA is that the WOA exploration stage relies on random entity selection from the population, which is not possible in some engineering applications with only one entity in the population. As a result, the WOA's exploration and surrounding stages change. EWOA is introduced to address the inherent disadvantages of traditional WOA to efficiently solve high-dimensional challenges. The following changes have been made to WOA to make the algorithm more efficient and resilient in its optimization:

- During the exploration phase, the search mechanism is adjusted to allow the whales to travel randomly throughout the search space. This phase allows the algorithm to exit the local solution.
- The range of the co-efficient vectors is changed to diversify the solution during exploration, as well as to enhance the exploitation process and accelerate convergence.
- To conduct a thorough search around the optimal solution, a nonlinear weight vector is developed and employed during the exploitation phase.

Then on each iteration, the performance of any combination of KHM (in the current population) will be assessed, and EWOA is updated for the next iteration. Like any evolutionary algorithm, these iterations are continued until reaching the desired quality of the population. In our work, this condition is to obtain accurate clustering.

Many methods have been used to optimize KHM, some of which are based on evolutionary algorithms. The main idea of our proposed method is the use of EWOA to optimize the parameters in KHM, which has not been used for this purpose so far. Given that the properties that are usually extracted have relative values in the text data, this algorithm can be helpful for optimization. In this study,

we used this algorithm variation on the top of KHM. In this algorithm, KHM is first executed on the data, and the resulting clustering is entered as a unit into the initial population of EWOA. Minimum, maximum, and average data are entered as another part of the initial population. The remainder of the original population is randomly generated. The goal of this initial population production strategy is to accelerate the achievement of optimal clustering.

We proposed a clustering method, which is a combination of KHM and EWOA. Similar to K-means, the Euclidean distance of the data is averaged based on the means. The KHM algorithm takes several random centers between the data and then improves these centers. Each sample is given a weight, and then data assigned to each center is calculated. The centers, weights, and their attributes are recalculated and updated. These steps repeat until it reaches convergence. The EWOA algorithm is used instead of randomly selecting the primary centers. The proposed algorithm is shown in Algorithm 1.

Algorithm 1: EWOA-KHM algorithm; A swarm-based algorithm.

1 Implement KHM in the dataset

2 Generate the initial population $P_1, P_2, \ldots, P_m$ for EWOA as follows:

- P_1 is the minimum samples in the dataset to create a cluster.
- P_2 is the maximum iterations.
- $P_3; \ldots; P_m$ are initial centroids.

Run EWOA

- Evaluate all individuals by objective function.
- Calculate the best current solution.
- Update the position to the optimal solution where the bait is desired.
- Update the position of the spiral update.
- Step 3 is repeated until the stop criterion (number of repetitions) is reached.

2.3.2 Keyword And Document Representation

There are many text representation models to represent keywords and documents. In the following, we will highlight the strategies we used in our project.

NLM

BOW became popular for many applications due to its simplicity, efficiency, and accuracy [40]. However, it fails to address the vocabulary mismatch problem [35], and it ignores word order and context. To mitigate these issues, we utilized *NLM* to represent words and documents as follows. *BERT* has state-of-the-art performance in several NLP tasks, from question answering to named-entity recognition [27]. In the following, we itemize the language models we applied in this work.

- **Word Embeddings**: *Word2Vec (WV)* and *Paragraph Vector* (*PV*), a generalization to full-length documents [66, 54].
- **InferSent**: InferSent uses a supervised learning approach to generate sentence embeddings [23]. It provides semantic representations for sentences. This model has two pre-trained versions, one trained with Glove and the other one trained with FastText. Each version is pre-processed using a different tokenizer.
- **SBERT**: Sentence BERT or SBERT is a modification of the pre-trained BERT network that uses Siamese network structures [80]. Instead of employing convolutions and recurrence, it is a sequence model that uses attention to gain sequence representation. It allows users to fine-tune their own sentence embedding methods, so it is a task-specific sentence embedding.
- **Universal Sentence Encoder (USE)**: The USE encodes sentences, phrases or short paragraphs into high-dimensional vectors [16]. It contains two possible models, the Transformer model and the Deep Averaging Network (DAN), and is trained on various datasets.

For representing documents using the models mentioned above, we apply the averaging word embeddings approach. In this way, we take the average of words and

sentences' vectors to construct document embeddings from meaningful word embeddings. To summarize the document in a single vector, average and sum are two standard summarization operators.
In summary, InferSent is 65% faster than SBERT because of its more straightforward neural network structure. However, by applying smart batching on a GPU, SBERT is about 9% faster than InferSent and about 55% faster than Universal Sentence Encoder.

2.3.3 Interactive User Interface (LM-Kt)

As mentioned before, clustering is not an easy task because of the noises and outliers in the data. One possible treatment to improve the clustering results could be to enhance the clustering process with interactive visualization strategies. We present our proposed system, LM-Kt [2], in the following;

User-guided Document Clustering

To involve the user in the clustering process, we introduce a user-centred framework to support a wide range of options for the underlying document representation models. The idea is that for each document d_i, first encode them using embedings, then we extract its key terms k_i using Doc2Vec, which are modified by the user. We generate a set of key terms $(k_1, k_2, k_3, ..)$ for each cluster C_i, which is assessed based on documents. To obtain a cluster-specific density, we take the union of these key terms in the set and name it K_c and for each element of this union, we compute its similarity with each one of the documents, and combine these similarities. We get the total, which represents the collective similarity of that key term to all the documents in the same cluster and assess based on that. By selecting the top N in this list, for which we used 5 as a threshold for N, we reach $K(i)$. We provide the key term clusters to the user, she manipulates these key term clusters to receive a new set of key term clusters $K(i')$. The next step is to map this set of key terms to a new cluster and re-cluster based on the embeddings of the documents $d_1, d_2, d_3, ...$ We implement this step by clustering documents around the key term clusters, computing the embedding of each key term cluster $k(i')$, and for each document,

[2] https://github.com/SimaRezaeipour/Interactive-clustering

computing the similarity of each one of those key term clusters and assigning it to the nearest set of terms. The overall process of our framework and how the user interacts with the clustering algorithm by modifying the cluster key terms are shown in both Algorithm 2 and Fig. 2.1.

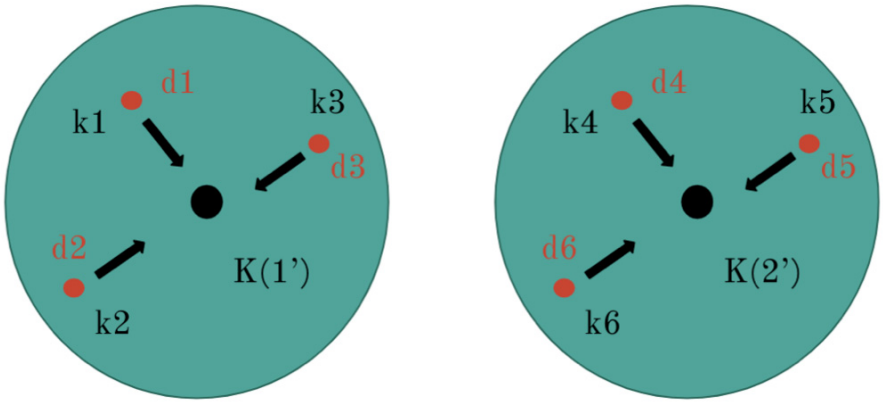

Figure 2.1: To actively engage the user in the clustering process, we introduce a user-centric framework. The concept revolves around extracting topics, referred to as key terms (k), for each document (d), which the user can modify. For every cluster (C), we create a set of key terms ($k_1, k_2, k_3, ..$) named K_c, which are evaluated in relation to the documents. To achieve cluster-specific density, we merge these key terms into a unified set K_c. For each element within this union, we calculate its similarity to every document and combine these similarities. The resulting total indicates the collective relevance of that key term to all the documents and serves as the basis for assessment.

After each user interaction to update the representative cluster key terms, the document clustering algorithm is re-executed taking into account user feedback.

Overview of The LM-Kt

The visual components and algorithm of our system are composed of different features such as re-clustering, sub-clustering, and mind map. The LM-Kt consists of two main parts: One part for processing the raw text, and one for visualizing the results. In the processing part, the user can log in to the system and upload CSV, PDF or text files. Then, the system empowers the user to choose from options like removing the numbers and non-English characters for pre-processing the documents. In this part, they can also select different clustering algorithms.

Algorithm 2: A framework of cluster key term modification

if *firstIteration* **then**
  For each document d_i, extract key terms k_i using Top2Vec;
  For each cluster C:
  Generate the set of cluster key terms $K_c = k_1 \cup k_2 \cup k_3$.. from all documents in C;
  Rank the key terms in K_c as follows:
  For each key term k in K_c:
  Compute its similarity with each one of the documents based on their embeddings;
  Compute the sum of these similarities s_k over all documents in C;
  Sort the key terms according to their s_k;
  The top N terms form a key term cluster (list) for document cluster C: $K'_C = ((k_1, s_1), (k_2, s_2), ..., (k_N, s_N))$
else
  $k(i') \leftarrow$ User modifies top key terms $k(i)$;
end
Re-cluster based on $(d_1, d_2, d_3, ..)$:
For each key term cluster (after user input) $K'c$:
Compute the similarity between $K'c$ and d using embeddings;
Assign d to the most similar $K'c$;
The result is a new set of document clusters C';

After the pre-processing, the user has access to document clusters in the visualization part. First, the user is asked to enter her desired number of clusters, albeit the system provides a default value generated by silhouette analysis and shown as a popup window. Using silhouette analysis, we can measure the similarity of an object to its cluster compared to its neighbours and find the distance between the resulting clusters. Silhouette analysis is used to select the optimal number of clusters. Accordingly, all visual components will be updated after the clustering process is completed. The overall view of the LM-Kt is shown in Fig. 2.2.

Figure 2.2: Overall view of the LM-Kt. Showing the expert interaction with 675 documents related to computer science. In the figure, the Document view (B): Displays the plain text of the focused document; the Document-cluster view (C): Shows the relatedness of the selected documents to each other; the Term cloud view (F): Displays the word cloud of the given selection (document selection, cluster, or focused document); Graph view (D): Depicts the projection of the documents using different techniques (T-SNE or force-layout graph), a document's title and top key-terms are shown on hovering the corresponding data point; Clusters view (E): Shows the top key-terms of each cluster inside the clustering boxes, where the user may provide further feedback about each clustering; Cluster key-terms view (G): Lists top terms of the selected cluster and depicts their level of importance in the bar charts; Term-cluster view (H): Showing the relatedness of the selected term(s) in the Cluster key-terms view to each cluster.

The user can interact with the system and gain the advantages of different views for better intuition into the data. The LM-Kt is comprised of several visual components

(views), see Fig. 2.2, where documents are shown as nodes, and the edges between nodes are specified based on the Cosine distance between nodes. Using the T-SNE algorithm [61], the position of nodes was computed, and the calculated clusters are shown in different colours. With the mouse hovering on the specific node, the user can view the details of that document, including the document name, the cluster it belongs to, and the top 5 keywords in the document. Clusters with major subjects of the collected data are shown in the clusters' view section on the top right. They contain the top five key phrases of each cluster. The term cloud view that displays a word cloud of top terms of a selected cluster is shown on the middle right. The cluster key term view lists the importance of the top terms of a selected cluster and their bar charts. The term-cluster view shows the relatedness of a selected term to each cluster on the bottom right. The cluster tree view is on the top left, representing the tree structure of the clustering results and showing the relationship between clusters and documents. On the middle left, the document view shows the content of the selected document, the list of top key phrases of the selected cluster, and the document cluster view on the bottom left displays the relatedness of the selected documents to other clusters. There are some buttons on the top of the screen, such as the save, delete, upload, and cluster buttons.

We can save the current session using the save button or go back to the pre-processing interface by the upload button. The user can select documents and remove them permanently from the original dataset. The new dataset is then processed, and the clustering process is repeated from the beginning. Another feature is re-clustering, which lets the user re-cluster some clusters without affecting other clusters. The user can choose multiple clusters by selecting the checkbox and re-cluster them into a new number of clusters. Using re-clustering, the user can apply both cluster union and split. Therefore, meaningful clusters, as specified by the user, are frozen by the algorithm and do not change in future re-clusterings. The next feature is sub-clustering which allows the user to split one cluster into multiple sub-clusters while keeping the original structure. In other words, the recently added sub-clusters are child clusters of the original one, and the structure of the parent layer remains the same. Like the re-clustering part, all the panels are subsequently updated. A view of the sub-clustering feature is shown in Fig. 2.3

2.4 Experimental Results

Evaluating the results of generated clusters is an important part of clustering projects. However, it is not an easy task since we do not have much prior knowledge of the dataset, especially for high-dimensional data. There are some metrics to measure the similarity of two clustering algorithm assignments such as Adjusted Rand index (ARI), Normalized Mutual Information (NMI), Adjusted Mutual Information (AMI), Homogeneity, Completeness and V-measure, which require knowledge of the ground truth classes and actual labels. On the other hand, some other metrics like the Silhouette Coefficient score do not need actual labels for evaluation. Higher scores mean that clusters are more dense and separated, which reflects the better clustering model.

In this section, we evaluate the performance of the clustering algorithm by employing different document representation methods to demonstrate the effect of the representation method on the clustering result. We chose five widely used benchmarks: IMDB movie reviews (IMDB), 20-Newsgroup (20NG), Reuters dataset, Twitter Sentiment Analysis, and YouTube Spam Collection. Then, represent the documents using both embeddings and basic methods like Term Frequency-Inverse Document Frequency (TF-IDF), and finally cluster them by our modified clustering algorithm. We employ various Python libraries for our implementations.[3] This study adopts ARI, NMI, and Silhouette for evaluating the different document representations. The performance comparison of document representations on all datasets using the modified Kmeans clustering algorithm indicates that InferSent achieves better results in most datasets. Results are presented in Table 2.1.

2.4.1 Expert Study

An expert study was conducted by two domain experts, including the author, to further evaluate the quality and effectiveness of the system. In the following, we introduce the dataset employed in the expert study, and next, we report the experimental results gathered.

[3]In our experiments, we use Gensim library for implementing Word2Vec, TF-IDF, and Doc2Vec, Pytorch for InferSent and SBERT, TensorFlow Hub for Universal Sentence Encoder.

Dataset	Metric	TF-IDF	Word2Vec	Doc2Vec	InferSent	SBERT	USE
20 Newsgroup	*ARI*	0.590	0.552	0.554	**0.627**	0.601	0.592
	NMI	0.500	0.615	0.434	**0.645**	0.623	0.612
	Silhouette	0.012	0.020	0.067	**0.128**	0.009	0.094
IMDB	*ARI*	0.481	0.527	0.695	**0.696**	0.691	0.571
	NMI	0.401	0.520	0.572	**0.635**	0.620	0.590
	Silhouette	0.015	0.047	0.001	**0.231**	0.012	0.193
Reuters	*ARI*	0.681	0.534	0.622	0.623	**0.687**	0.612
	NMI	0.643	0.484	0.571	0.635	**0.695**	0.589
	Silhouette	0.169	0.059	**0.192**	0.067	**0.192**	0.024
Twitter	*ARI*	0.531	0.586	0.594	**0.665**	0.624	0.637
Sentiment	*NMI*	0.520	0.570	0.588	**0.620**	0.612	0.629
Analysis	*Silhouette*	0.010	0.009	0.05	**0.283**	0.005	0.038
YouTube	*ARI*	**0.681**	0.576	0.589	0.616	0.593	0.596
Spam	*NMI*	**0.653**	0.520	0.541	0.605	0.580	0.588
Collections	*Silhouette*	**0.191**	0.029	0.013	0.032	0.002	0.109

Table 2.1: Performance comparison of modified K-Means clustering results on all datasets. Bold numbers indicate the best results. InferSent achieves better results in most datasets. USE stands for Universal Sentence Encoder.

Dataset Preparation

To conduct the expert study, we created a new dataset of scientific articles in computer science. The data has been gathered from a research group archive on text mining. The articles are archived in Zotero [1], an open-source bibliographic assistant for researchers. The dataset is comprised of 660 articles, including 12 different categories. First, we extracted the BibTex file from Zotero, which contains the articles' metadata, including the abstracts. We collected the following segments of each article; Abstract, author, title, journal, bibliography type, year and month of publication, and URL. Then, we converted the dataset to a CSV file. Each abstract is considered as a single document and saved by its title concatenated by the publication year as the document's name. To upload the dataset in the LM-Kt system, we saved each document in a TXT file. So, the user can upload those TXT files into the system and get the clusters.

To have an efficient data mining tool, well-prepared data is a prerequisite. Data preparation is about making the data ready for the application and gaining the necessary intuition to solve application problems. Accordingly, we put adequate

time and effort into pre-processing. After removing extra items, we converted the text to lowercase. We applied NLTK, Regular expressions and BeautifulSoup in Python.

Expert Study Results

The expert users are Ph.D. students in computer science, working on machine learning and NLP. The first expert started with 3 clusters. Then, based on the projection result and by looking at the Cluster view and her expert knowledge, she increased the number of clusters to 5. Since the new clustering result was preferable to her, she decided to increase the number of clusters to 8 to determine whether there would be better clusters. She found the clustering results with 8 clusters not as satisfying as the previous one. The expert searched for the different terms in the term similarity view during the study. These searches helped her retrieve the related documents, and she was also able to identify the most similar cluster to these terms. The second expert began with 2 clusters, but upon examining the projection results and referring to the Cluster view, along with her domain expertise, she decided to increase the cluster count to 4. Encouraged by the improved clustering outcome, she further expanded the clusters to 5 to explore potential enhancements and it met her expectations. During her analysis, the expert utilized the term similarity view to conduct searches, which assisted in locating relevant documents and identifying the cluster most closely associated with those terms. The summary of the overall final clustering of the dataset is presented in Fig. 2.4.
The experts considered the final clustering result compelling and informative. They mentioned that by using the system, they retrieved the related papers on a specific topic faster and better than a conventional search in Zotero or similar bibliography applications. The expert study was conducted in around 60 minutes, including 10 minutes introducing the system to the experts, 40 minutes executing the task, and 10 minutes interviewing the experts. We have summarized the Interview results as follows.

Representation Models. In the first scenario, the experts used our baseline interactive system (Vis-Kt). They were not satisfied with the projection result;

although they considered the words representative, they did not reflect on the projection. The clustering was acceptable, but the results depicted that this dataset needs domain-specific stop-word filtering in addition to the general English stop-word list. The second scenario was based on using NLM. As seen in the graph-view projection, the clusters were much more meaningful in this scenario. However, the top key terms in each cluster are too general compared to the BOW representation.

Interactive Visualization. Overall, the experts were satisfied with the LM-Kt and found most of the modules useful in practice. We automatically logged every operation that the experts conducted during the expert study. The most frequent operations were *mousehover* on the Graph view nodes, *Clickingacluster* in Cluster view, and *Highlightingdocuments* in Cluster view and Cluster Key terms view.

2.5 Conclusion

We applied a modified Kmeans clustering algorithm for unsupervised interactive text clustering and evaluated it on many widely used benchmarks. To represent the documents, we used (a) traditional approaches such as BOW and (b) state-of-the-art embedding methods like SBERT, InferSent, and USE and compared the results. Then, we evaluated the clustering results on a dataset by asking an expert to assess the interactive document clustering system (LM-Kt). The evaluation of the LM-Kt indicates that users want two capabilities from such a system. The first is the ability to interact with the clustering algorithm effectively. The second is a visualization that enables them to efficiently explore the document collection and the resulting clustering.

In conclusion, this chapter has made several significant contributions to the field of interactive document clustering systems. The primary contributions include the introduction of a novel document clustering algorithm that utilizes deep language model-based document representations, a new objective function, and a unique approach to initializing the number of clusters. This innovation addresses the research gap in understanding the advantages and effectiveness of employing deep language models compared to traditional key term-based methods. Furthermore,

the chapter presents a user interaction component that enables users to actively engage with deep language model-based document representations in the interactive clustering process. This aspect addresses the research gap concerning the impact of user interaction on clustering and user experience, shedding light on both challenges and opportunities in its implementation. The expert study conducted for LM-Kt in a specific domain serves to identify domain-specific factors influencing model performance and explores strategies to address these challenges. Lastly, the chapter evaluates the modified clustering algorithm without user interaction, employing deep language models on standard datasets, to assess its performance and generalizability, and to understand its strengths and weaknesses in typical clustering scenarios. Collectively, these contributions enrich our understanding of document clustering, interactive systems, and the application of deep language models, while providing insights into avenues for further research in this domain.

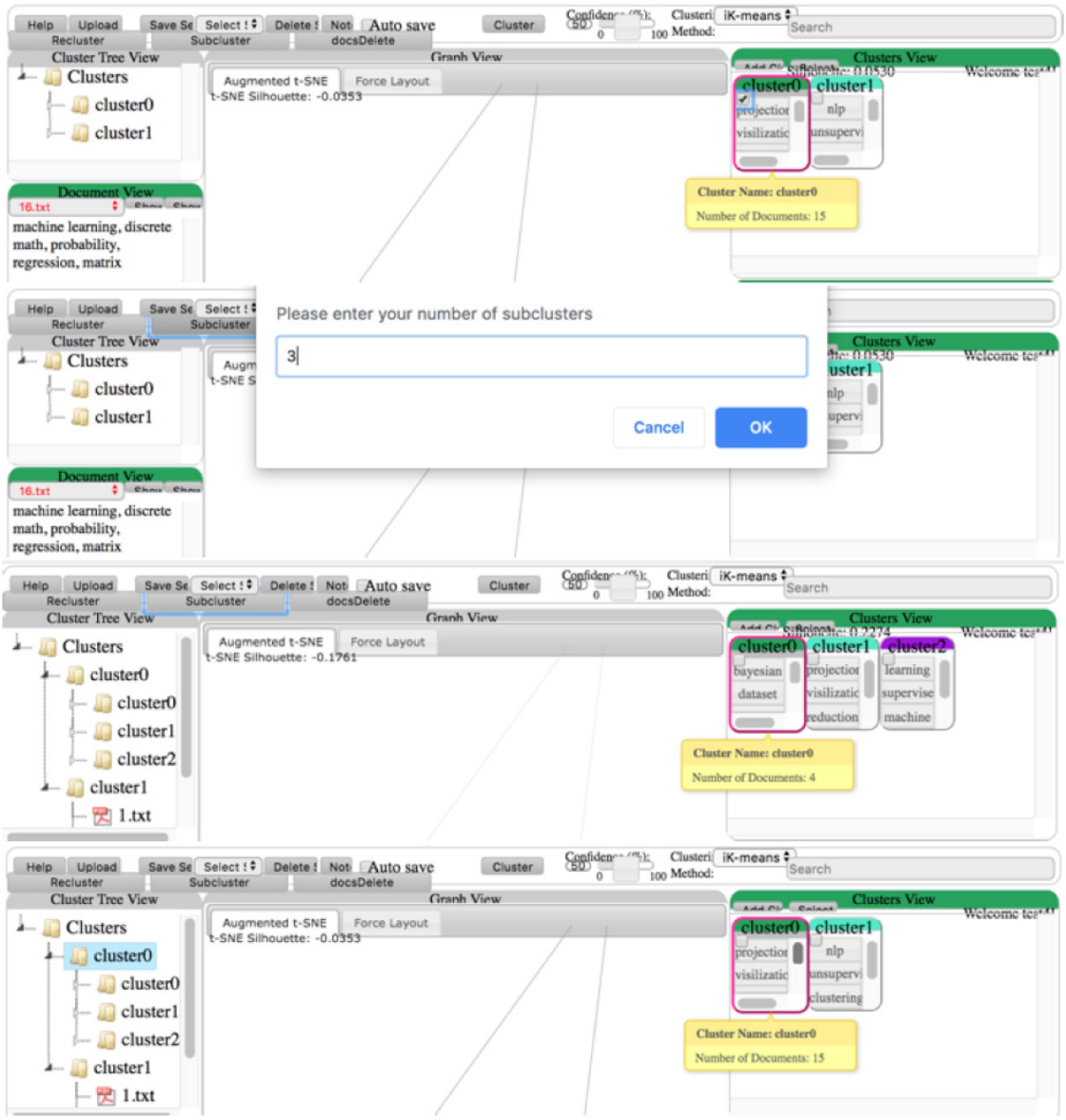

Figure 2.3: Sub-clustering feature

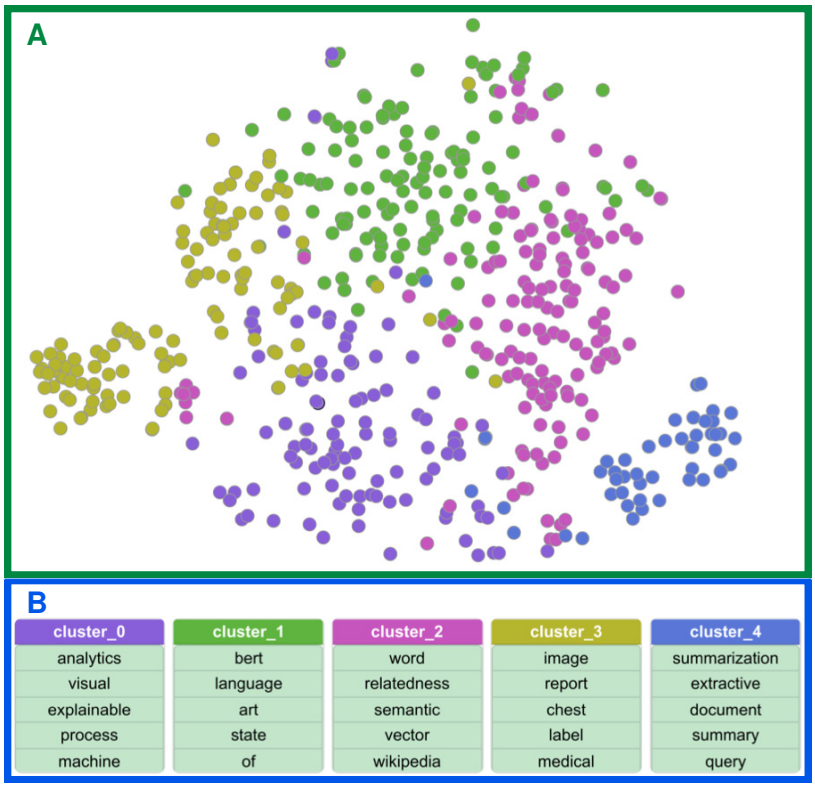

Figure 2.4: Result of the clustering of the dataset. In the figure: (A) a T-SNE projection of the corpus; (B) The top 5 key terms for 5 clusters identified throughout the expert study.

Chapter 3

Addressing The Gap Between Current Language Models And Key-Term-Based Clustering

3.1 Introduction

Text data is a valuable source of information in various domains, and text mining often relies on automatic clustering algorithms for effective analysis and knowledge discovery [3, 33][1]. However, the output of even the most advanced clustering algorithms may fall short of satisfying a user's requirements and expectations [43]. This gap between the outcome of an automated algorithm and the expectation of the analyst is particularly frequent in clustering text corpora, as a collection of documents could be organized into groups of $content - related$ documents in many different ways, all of which may make sense to distinct audiences. To address this issue, interactive or user-guided clustering methods have been developed, which ask analysts to provide input and personal insights into the corpus content to guide a clustering algorithm to produce a model consistent with their perspective [5, 43, 91]. Interactive clustering solutions vary in how they implement this general approach, but they typically implement an iterative pipeline that enables analysts to provide subjective feedback that reflects their domain knowledge, interests, and preferences. This feedback can take the form of explicit labels, key terms, or implicit judgments expressed through interactions with visualizations of the clustering results [43, 91]. Such feedback can be used to guide the clustering algorithm towards a more meaningful partitioning of the document collection, resulting in more accurate and relevant clusters. By tailoring algorithmic choices and incrementally modifying the outcome, the model will gradually converge to a final state that meets the user's needs.

[1] This chapter is the extended version of the conference paper, presented at DocEng2023, as one of the three papers nominated for the Best Student Paper Award: https://dl.acm.org/doi/10.1145/3573128.3604900

The current solutions for user-guided clustering have a practical limitation in their applicability due to their reliance on a fixed language representation model. Most systems use the BOW model, combined with TF-IDF scores, to capture term importance. While BOW has been a reliable technique for many NLP tasks, it is being surpassed by more recent NLM. NLM can capture not only term importance but also the relationships between terms in a text, based on their order, resulting in significant improvements in NLP tasks such as Named-Entity Recognition (NER), translation, and text generation. Thus, there is an urgent need to investigate the use of NLM in document clustering tasks, particularly in the context of semi-supervised document key-term-based clustering. However, incorporating NLM in this context is challenging due to the gap between the interpretability of the document representation (the embeddings) and the intuitive semantic elements guiding the clustering task (the key terms).

We introduce a testbed platform for document clustering that incorporates multiple language models, Mod-Kt. Our solution provides a flexible and extensible approach for assessing the performance of different language models in user-guided clustering tasks. With the ability to incorporate an arbitrary number of distinct language models, Mod-Kt can be useful for researchers investigating clustering setups, and also for various end-users, from researchers exploring scholarly corpora to journalists analyzing potentially interesting subjects in news datasets. The Mod-Kt offers an intuitive interaction paradigm, combined with the potential performance improvements of recent NLM, allowing users to derive maximum benefit from the system.

This chapter has made several significant contributions to the field of NLP and user-centered approaches. Below, we list research gaps along with the associated contributions.

1. A novel approach for interacting between language models for enhanced user-centric applications:

    - Research gap: The current literature lacks a detailed investigation into methodologies for effectively aligning NLM with user-guided tasks in knowledge discovery. There is a need for a more thorough exploration of

the mechanisms facilitating the seamless integration of NLM into user-driven knowledge exploration, an area that has not been sufficiently examined.

- Contribution: A novel communication layer bridging the gap between word-level and higher-level language representations.

2. A flexible and open-source testbed system, Mod-Kt:

   - Research gap: Mod_Kt incorporates similar features with LM-Kt, which was introduced in the preceding chapter, albeit with subtle modifications, aimed at enhancing system performance, speed, and flexibility.
   - Contribution: Providing a practical resource for researchers and developers to experiment, assess, and enhance user-centered applications driven by NLM. These contributions collectively propel the field forward, aiding in the creation of more user-friendly and adaptive language technologies.

3. Expert study:

   - Research gap: Investigating domain-specific factors influencing Mod-Kt model performance, exploring variations in effectiveness across diverse domains and contexts, and developing strategies to address unique challenges within each domain.
   - Contribution: Conducting an expert study to assess Mod-Kt within a specific domain, mirroring the approach taken with LM-Kt. Based on the expert study, Mod-Kt outperformed the previous systems.

This chapter is structured as follows: A concise overview of various document representation techniques and NLM, as well as prior studies investigating user-directed document clustering are provided in Section 3.2. We detail the Mod-Kt in Section 3.3. We describe three quantitative evaluations conducted to evaluate the effectiveness of distinct language models for document clustering, as well as an expert study conducted to assess its utility in a practical scenario from an end-user perspective in Section 3.4. Concluding remarks are in Section 3.5.

3.2 Related Work

This section provides a brief summary of document representation techniques and NLM, as well as the relevant literature on solutions for interactive document clustering.

3.2.1 NLM

Machine learning models for text mining require documents to be represented as fixed-size vectors. The representation should capture document content, context, and semantics to enable the identification of relevant information, such as groups, topics, and concepts. In the following, we provide an overview of the approaches used to represent words and documents in this study.

The standard *BOW* model gained popularity due to its simplicity and satisfactory accuracy in many text processing applications. However, the model does not take into account the order in which the words appear in a sentence. To address this issue, Mikolov [65, 64] introduced the ground-breaking *word embeddings*, a novel approach to represent words as vectors that are capable of capturing the contextual information surrounding each word. This is achieved by training a shallow neural network that maps words to vectors in a space where semantically similar words have similar vector representations. *Word2Vec* is the popular name for this first class of word embeddings.

An enhanced approach has been later introduced building upon Mikolov's skip-gram model [12, 46] that represents words as a bag-of-n-characters. Referred to as *FastText*, the approach has shown interesting results in various NLP tasks. Further research has sought to expand the capabilities of the word embedding model to encode not only individual words, but also larger textual units such as sentences, paragraphs, and complete documents.

Doc2vec [54] is an extension of Word2Vec designed to learn embeddings for variable-length documents. The core idea behind Doc2Vec's approach is to encode paragraphs as a single vector representation consisting of the average of their word vectors and incorporate the paragraph vectors as surrounding information to nearby words. In user-guided document clustering, Doc2vec has shown promising results in

capturing the semantic similarity between documents.
BERT is a stack of transformer encoders that achieved superior results in various NLP tasks [27]. It utilizes two crucial concepts in its learning process, namely, the Transformer attention model architecture [104] and unsupervised learning for predicting missing words given the surrounding context and next sentence prediction. SBERT, or Sentence BERT, is a variant that uses attention and Siamese network architectures to build a representation of a sequence of words [81]. It enables fine-tuning custom sentence embeddings and deriving task-specific sentence embedding models.

3.2.2 User Guided Document Clustering

Incorporating user guidance into the process has been shown to be effective in ensuring clustering results more in line with the analyst's perspective, as demonstrated in several studies [5, 7, 43, 92]. However, designing an intuitive visual knowledge discovery workflow without requiring the user to directly manage the algorithms or read extensive textual content is challenging. The choice of document representation impacts the clustering algorithm and also the visualization techniques and interactive features that can be employed. These, in turn, define how the user provides feedback on intermediate clustering results.
The *iVisClustering* visual text analytics system utilizes the Latent Dirichlet Allocation (LDA) algorithm for interactive document clustering [57]. *iVisClustering* allows users to modify the term weights in LDA to enhance their knowledge about the corpus and improve clustering results gradually. The system includes various functionalities such as soft-clustering visualization, force-based graph layout algorithms, and a hierarchical topic explorer tool to perform merge, split, and remove operations [30, 7]. Although LDA can be adapted to incorporate user interaction [78], results from LDA do not necessarily remain consistent along multiple iterations. *UTOPIAN* is a follow-up system [21] that replaces LDA by Non-Negative Matrix Factorization (NMF) to avoid the inconsistency problem [70]. The user interaction in both iVisClustering and UTOPIAN is hindered by the complexity of the underlying algorithms, which prevents a simple and intuitive user experience. Additionally, both systems face the issue of empirical convergence,

which refers to how quickly the algorithm converges from a practical human perspective.

The *Vis-Kt* system [69, 92] is aimed at supporting user-guided document clustering while avoiding the inconsistency and empirical convergence problems. An additional goal is to provide an intuitive interface that abstracts the complexity of underlying algorithms and document representations. It incorporates two clustering algorithms, namely *iKmeans* [92] and *Lexical Double Clustering (LDC) [69]*, which can be switched based on user requirements. The system executes key-term-based clustering incrementally, as shown in Algorithm 3. The user may interact to update

Algorithm 3: The framework for user-guided key-term based clustering (adapted from [90]).

```
1  if firstIteration then
2  |   termClusters ← Generate term Clusters;
3  |   Get the top terms of each term cluster;
4  else
5  |   termClusters ← User defined top terms;
6  end
7  Calculate the centroids of the term clusters;
8  Find the top key-term from each cluster;
9  Find document seeds based on the term clusters;
10 Doc clusters ← Use seed documents to guide the clustering algorithm and
     then cluster documents;
```

the representative key-terms and the document clustering algorithm is executed again taking this feedback into account. This incremental approach is user-focused and requires limited user effort. Nonetheless, Vis-Kt relies on BOW and TF-IDF for document representation, which is a limited representation compared to the capabilities of recent NLM. Vis-Kt's interface and interaction model inspired the visual interface developed for the framework described in this chapter.

A VA framework has been introduced [100] focused on assisting users in producing and evaluating multiple clustering outcomes based on their quality and attribute-based data. Users may create clusters using either automated algorithms

or manual methods, and visually evaluate them using cluster tendency scores and a parallel cluster view.

In a broader context, a testbed system for interactive visual exploration of dimensionality reduction and clustering of various data types, including images, documents, and vectors has been introduced [22]. The system allows the user to switch between different clustering and dimensionality reduction techniques, providing various combinations to choose from. While it is possible to compare the results yielded by different techniques, a user cannot provide direct input to the clustering algorithms.

Given the summary of current user-driven approaches to document clustering presented in this section, it is remarkable that no previous work has incorporated recent NLM-based representations. Yet, by adopting a BoW-based approach the user is presented and interacts with key-term-based abstractions, whereas in NLM representations no such abstractions are in place for direct user manipulation. Moreover, NLM word and document representations are represented in distinct vector spaces, which means they cannot be easily interpreted and compared with each other. All these factors indicate that employing NLM representations in user-guided clustering requires an intermediate communication layer. We propose implementing such a communication layer as a modular component for user-guided clustering that allows for the evaluation and comparison of various document representations, with the ultimate goal of improving clustering performance.

3.3 A Testbed For User-Guided Document Clustering (Mod-Kt)

In this section, we introduce the extensible framework for user-guided document clustering designed to support multiple word and document representation models, Mod-Kt. It is based on a modular architecture where word and document models must implement a protocol that facilitates seamless communication throughout the clustering algorithm's pipeline. As interactive clustering requires a proper interface, we also provide visual components to support user supervision by means of inspecting and modifying key-terms characteristic of the different clusters.

The Mod-Kt has been made freely available to the public as open-source software

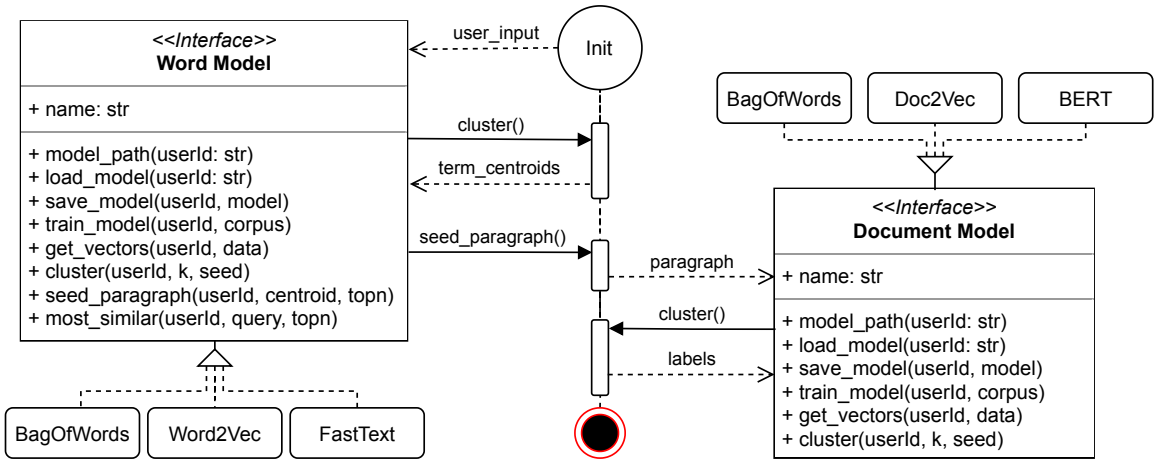

Figure 3.1: The Mod-Kt's architecture is illustrated by a UML diagram, showing the traditional class diagrams describing the object-oriented relations for word and document model strategies on the left and right sides, respectively. In the middle, sequence and state diagrams depict the message exchange between the word and document model interfaces. The method `seed_paragraph()` is emphasized, as it serves as a bridge between the word and document models.

under the GPLv3 license[2]. The source code can be accessed on the GitHub repository[3], allowing users to contribute and customize the software according to their specific needs.

3.3.1 The Modular Architecture

The extensible modular architecture for user-guided key-term-based clustering with varying word and document representation models is illustrated in Fig. 3.1. It has been designed following a behavioral *Design Pattern* called Strategy, which enables defining a set of algorithms, encapsulating each one, and making them interchangeable [36]. So far, we incorporated the word and document representation models depicted in Table 3.1.

The design pattern adopted ensures extensibility, so other representation models can be incorporated as long as each word or document model implements its specified interface, as illustrated in Fig. 3.1. Each representation model is fit to a particular clustering method that must be capable of handling its particularities. For instance, the BOW representation model yields a sparse matrix and the clustering algorithm

[2]https://www.gnu.org/licenses/gpl-3.0.en.html
[3]https://github.com/ericmacedo/i2DC

Word Representation	Document Representation
Bag-of-Words	Bag-of-Words
Word2Vec	Doc2Vec
Doc2Vec	SBERT

Table 3.1: Currently supported word and document representation models.

must handle sparse vector representations well. The clustering algorithms typically associated with BOW representations are not suitable to operate on dense vector representations. Our flexible approach tackles this inability to match a given model to a proper clustering algorithm, which may produce unsatisfactory results.
Dense and sparse vectors need to be handled differently because they have distinct characteristics when calculating distance. For the BOW models, we employ the iKmeans algorithm, which to the best of our knowledge yields the best results on user-guided document clustering. For the remaining models (Word2Vec, Doc2Vec, FastText, and SBERT), we incorporated a user-guided clustering algorithm inspired by previous work [14], which performs well on dense vector representations.
In the context of distance-based clustering algorithms, the distinction between dense and sparse vectors arises from the nature of the vector representations:

- Dense Vectors: Dense vector representations, like those generated by Word2Vec, Doc2Vec, FastText, and SBERT, typically have non-zero values in most of their dimensions. In these representations, the proximity or distance between vectors can be effectively measured using common distance metrics, such as cosine similarity or Euclidean distance. Distance-based clustering algorithms can be applied to such dense vector representations.

- Sparse Vectors: Sparse vectors, often encountered in BOW models, have a high number of zero values, as most dimensions correspond to the absence or presence of specific terms in a document. Traditional distance metrics may not perform well with sparse vectors because they tend to emphasize the differences caused by zero values, rather than capturing the underlying semantic similarity between documents. Therefore, specialized algorithms, like the iKmeans algorithm in the case of BOW models, are employed to address the unique challenges of clustering with sparse vectors effectively.

In the given context, iKmeans is selected for BOW models, as it is tailored to handle the sparsity of BOW representations efficiently and is known for yielding strong results in user-guided document clustering. For dense vector representations, a user-guided clustering algorithm inspired by prior work is used, benefiting from the dense nature of the vector representations. These choices reflect the need to adapt clustering algorithms to the specific characteristics of vector representations, whether dense or sparse, in order to achieve optimal clustering results.

The NLP pre-processing operations such as tokenization, stemming, and stop word removal are performed initially. This is followed by a two-phase inference process that generates a word embedding space X_{wv}, which creates word vectors, and a document embedding space X_{dv}, which creates document vector representations. The primary challenge addressed by this pipeline is: (i) establishing a meaningful relationship between the structures of both NLM embedding spaces, namely the word space and the document space; and (ii) incorporating user guidance to gradually refine such a relationship and incorporate it into the clustering outcome. In the first execution, prior to incorporating any user feedback, the corresponding clustering algorithm is executed on the word embedding space X_{wv} using a selected word representation model (WV). This step outputs K clusters represented by their centroids in $C_{WV}(K)$. Each centroid in $C_{WV}(K)$ establishes a reference point in the word vector space, allowing for the collection of a set of words that are used to construct an imaginary paragraph, referred to as $P_{seed}(K)$. The underlying assumption is that each $P_{seed}(K)$ provides a suitable representation for capturing the semantics of its corresponding word cluster. The $P_{seed}(K)$ seeds are input to the document clustering step. The clustering algorithm is executed with the selected document representation model (DV), taking the inferred document embeddings $P_{seed}(K)$ as the initial centroids.

After the initial clustering, the user has the opportunity to provide feedback to guide subsequent algorithm iterations. She can offer positive or negative feedback on each key-term based on what she considers to be the most relevant to defining each cluster, at the moment. Each interaction will result in updates to the representative terms (i.e., the word vectors) for each cluster. As a result, the cluster description is expected to gradually move towards a reference point in the word

embedding space, which should in turn affect the organization of the clusters. In summary, the user interacts with views of the clusters and their respective characteristic key terms discovered in the word representations. User inputs modify the structure of the term clusters, which impacts the resulting document clustering outcome, as outlined in Algorithm 4. This approach does not require the user a deep understanding of the underlying implementation strategies. In Section 3.4 we report results from a systematic quantitative evaluation of the clustering outcomes in publicly available text datasets, considering different representation models, plus a qualitative assessment of the platform with an expert user.

Algorithm 4: User-guided key-term-based clustering approach.

```
if firstIteration then
    termCentroids ← X_wv.cluster();
else
    termCentroids ← X_wv.cluster(seed);
end
for centroid in termCentroids do
    seedParagraphs[i] ← X_wv.seed_paragraph(centroid);
end
documentSeeds ← X_dv.get_vectors(seedParagraphs);
documentClusters ← X_dv.cluster(documentSeeds);
```

3.3.2 The User Interface

Three web pages provide interaction functions, accessed by means of a navigation bar. The primary one is for handling the corpus, and the others are a Dashboard of visual components and a Session Manager. An important concept behind the interaction paradigm adopted is that of a *Session*. A session stores the current state of each visual component and can be in one of four states: *staged* (currently in use), *saved* (stored on the server), *loaded* (retrieved from the server), and *deleted* (removed from the server).

In the Corpus page, the user informs a valid corpus (*TXT*, *PDF*, and *CSV* formats are accepted). One or multiple files can be uploaded and then pre-processed

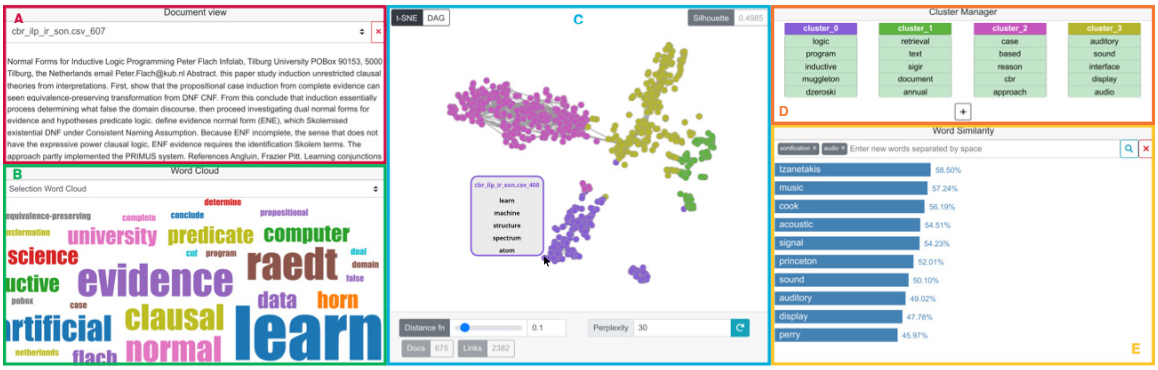

Figure 3.2: A view of the Mod-Kt, operating on 660 documents on computer science topics. The interface features as components a Document view (A) that displays the content of a focus document, a Word cloud view (B) of either a focus document or a cluster, a Graph view (C) that displays a similarity map of the documents in the corpus, a Cluster Manager (D) to inspect clusters and provide feedback to the clustering algorithm, and a Word similarity view (E) that displays a similarity bar chart for words identified as the most similar to user-provided query words.

on the server. This page also displays a table view of the corpus so the user can browse, view, select, or delete documents. Once a corpus is loaded the user can specify stop words, select the word and document language models, and the system is first trained in unsupervised mode.

Once the training is completed the user can move to the *Dashboard* page, which has the visual components that support interaction with the clustering, illustrated in Fig. 3.2. In the *Dashboard* the user will be prompted to either load an existing session or start a new one and the navigation bar will display a new set of features, namely:

- A + button to append new documents to the current session.
- A $re-cluster$ button to use the session's current state as seed to the clustering algorithm.
- A *Notes* popover and window to annotate the staged session.
- Buttons to act on sessions (start, save, load, or delete).

The Dashboard components present the corpus from various perspectives, with linked views so that changes resulting from interaction in one component will reflect on the others. The *document view* (see Fig. 3.2A) shows the content of one or multiple selected document(s). A *word cloud view* provides a rapid glimpse into the most significant terms of a selected document or cluster (see Fig. 3.2B). The *graph view* depicted in Fig. 3.2C presents a visual abstraction of the corpus as a document

similarity map, where documents are arranged as points in a two-dimensional space to reflect their (dis)similarity relationships. The similarity map is computed using either T-SNE [102] or a force-based graph layout and admits several interactions. The *cluster manager view* depicts a representation of the clusters and their top key terms (see Fig. 3.2D). Users may interact to select clusters and edit metadata (cluster name and color), add, remove, and change the key-terms' weights (positive or negative weighting), and delete or add clusters. Finally, the *word similarity view* (see Fig. 3.2E) features a query field where users can enter words to request their most similar words in the corpus, according to the underlying model. This resource is particularly useful for users unfamiliar with the vocabulary, as they can discover synonyms or identify uses of similar words in different contexts.
Users can save the current state as a *session* once they are satisfied and feel that the clustering reflects their perspective about the corpus. These saved sessions serve as *checkpoints* in the knowledge discovery process, as users may track changes in the clustering structure along the exploration. This is enabled on the Session Manager page for managing sessions, where it is possible to compare multiple sessions to analyze the changes made along the knowledge discovery process, e.g. observing how documents moved from and to different clusters, or the impact of user feedback on the outcome.

3.4 Evaluation And Experiments

In the following sections, we present experiments to evaluate the Mod-Kt and illustrate its flexibility as a testbed for assessing distinct language models in document clustering tasks. The initial clustering obtained with each representation strategy is evaluated relative to standard clustering metrics. Later, we report a study with a domain expert using the system in a real-world use case scenario.

3.4.1 Comparing Language Models For Clustering

We evaluate the clustering method of a selected range of representation strategies, adopting an evaluation methodology similar to that of related work [92, 69]. However, we explore the variety of possible combinations between the selected range of document representation models. In this quantitative evaluation, we measure the

clustering results with well-known clustering metrics and their processing time. The code for this experiment is publicly available [4].

	Random	Shape	Precision + Recall
S	✓	✗	✗
ARS	✓	✓	✗
AMI	✓	✗	✗
FMS	✓	✓	✓
V	✗	✓	✗

Table 3.2: Clustering properties captured by each metric considered. Random: Is the metric random clustering aware? Shape: Can it compare clusterings with different shapes? Precision + Recall: Does it measure precision and recall?

As summarized in Table 3.2, four metrics have been employed to evaluate different aspects of the clustering:

- **Silhouette score (S)**: Is an unsupervised metric of the consistency within clusters [85]. Values are in the range $[-1, +1]$, where the best value is 1, the worst value is -1, and values close to 0 indicate overlapping clusters. Negative values generally indicate an erroneous cluster assignment, with misplaced samples that are closer to another cluster.

- **Adjusted Rand Score (ARS)**: it is a similarity measure between the ground truth classes and the predicted clustering [45]. This measure ensures that a value close to 0 stands for random labelling and values close to 1 stand for identical clusterings. ARS is penalized by the number of false-positive and false-negative predictions.

- **AMI**: Measures the agreement between the ground truth and the clustering prediction assignments, ignoring permutations. AMI is an adjustment of the MI score to reduce the effect of the agreement by chance [106]. Values close to 0 stand for random clustering.

- **Fowlkes-Mallows Score (FMS)**: Measures the similarity between two cluster models, and it is defined as the geometric mean of the pairwise precision and recall [34]. Precision measures the proportion of correct cluster

[4] https://github.com/ericmacedo/DocEng2023-Experiment

membership assignments, whereas recall measures the proportion of all cluster members that were correctly assigned.

- **V-measure (V)**: A clustering algorithm has perfect *Homogeneity* (H) if each cluster contains only data points of a single class; a clustering has perfect *Completeness* (C) if all data points of a given class are assigned to the same cluster. Rosenberg and Hirschberg [84] define the *V-Measure* as the harmonic mean of H and C. The scores are in the range [0,1], where 1 stands for perfect clustering.

We execute the unsupervised clustering pipeline and collect the result, repeat it 100 times, and report the average and standard deviation of this set of observations. This experiment provides an assessment of the initial clustering, from which subsequent interactions will derive. We emphasize the silhouette score of the projected space with the low-dimensional representation obtained with T-SNE. We considered three publicly available datasets, henceforth referred to as D1, D2, and D3, respectively:

1. **CBR-ILP-IR-SON (D1)**: A corpus of 675 scientific papers in four computer science subjects [72], namely, Case-Based Reasoning (CBR), Inductive Logic Programming (ILP), Information Retrieval (IR) and Sonification (SON). The instances are described by the title, authorship, affiliation, abstract and references.

2. **NewsSeparate (D2)**: 381 news RSS feed digests manually labelled with 13 labels. This dataset was collected in 2006 [72] from 4 different news outlets' web pages (CNN, BBC, Reuters, and Associated Press).

3. **9NewsGroup (D3)**: A corpus of 1,080 newsgroup articles evenly partitioned across nine distinct labels, derived from the original dataset 20NewsGroup [52]. The dataset was fetched and curated with the *Scikit-learn* library [73].

The experiments were executed on a dedicated machine with the following configuration: OS Ubuntu 18.04 Bionic; Kernel x86_64 Linux 5.0.0-32-generic; CPU Intel Core i7-6850K, 12 cores (4GHz); GPU 2x GeForce GTX 1070 (SLI); 4x 16GB DDR4 memory (2400MHz).

Table 3.3: Results from experiments with D1. The strategy *BoW + BoW* shows the best values for most of the clustering metrics in the multidimensional space. *Doc2Vec* yields the best silhouette score both for the multidimensional and the projected data

	Document model: BoW	Document model: Doc2Vec	Document model: SBERT	Word model
S (MD)	0.0562 ± 0.0002	0.3897 ± 0.0152	0.1104 ± 0.0002	**BoW**
S (t-SNE)	0.3322 ± 0.0291	0.5049 ± 0.0193	0.4475 ± 0.0256	
ARS	0.6565 ± 0.0135	0.6228 ± 0.0206	0.6141 ± 0.0023	
AMI	0.6755 ± 0.0126	0.6361 ± 0.0322	0.6490 ± 0.0065	
FMS	0.7677 ± 0.0094	0.7404 ± 0.0224	0.7474 ± 0.0045	
V	0.6583 ± 0.0135	0.6247 ± 0.0205	0.6161 ± 0.0023	
Time	1.1267 ± 0.1467	1.3320 ± 0.1468	9.8179 ± 0.3803	
S (MD)	0.0559 ± 0.0002	0.3782 ± 0.0138	0.1110 ± 0.0009	**Word2Vec**
S (t-SNE)	0.3319 ± 0.0281	0.5113 ± 0.0151	0.4440 ± 0.0253	
ARS	0.6381 ± 0.0143	0.6219 ± 0.0216	0.6180 ± 0.0060	
AMI	0.6389 ± 0.0273	0.6561 ± 0.0296	0.6579 ± 0.0098	
FMS	0.7421 ± 0.0190	0.7538 ± 0.0209	0.7539 ± 0.0072	
V	0.6400 ± 0.0142	0.6239 ± 0.0215	0.6199 ± 0.0059	
Time	1.9488 ± 0.0540	3.1325 ± 0.0488	19.2984 ± 0.5818	
S (MD)	0.0559 ± 0.0002	0.3850 ± 0.0188	0.1108 ± 0.0017	**FastText**
S (t-SNE)	0.3312 ± 0.0278	0.5099 ± 0.0134	0.4420 ± 0.0258	
ARS	0.6391 ± 0.0146	0.6267 ± 0.0249	0.6168 ± 0.0089	
AMI	0.6408 ± 0.0261	0.6503 ± 0.0341	0.6560 ± 0.0140	
FMS	0.7434 ± 0.0182	0.7502 ± 0.0237	0.7525 ± 0.0100	
V	0.6410 ± 0.0145	0.6286 ± 0.0248	0.6188 ± 0.0089	
Time	1.7230 ± 0.0616	2.8604 ± 0.0687	18.9352 ± 0.4367	

Experiment 1: Dataset CBL_ILP_IR_SON For most of the clustering metrics, the strategy *BOW + BOW* yields better results, in general, as can be seen in Table 3.3. Moreover, it has a processing time considerably shorter than the remainder strategies.

Whereas the *BOW + BOW* strategy has shown better performance in general, both in the multidimensional space and the projected space with T-SNE, the document representation with *Doc2Vec* yielded better results in terms of silhouette score, suggesting this model yields better-defined clusters. For user-guided clustering, having clear cluster boundaries is highly desirable, since it plays an important role in the user process of obtaining gradual knowledge along this process.

Experiment 2: Dataset NewsSeparate Similar to the results reported in the previous experiment, the strategy *BOW + BOW* requires less processing time when compared with the remaining strategies, due to its computational simplicity.

However, in this second experiment the strategy *FastText+ Wor2Vec* as the word representation model yields better clustering results in the multidimensional space. In this scenario, *SBERT* as a document representation model resulted in better silhouette scores for the data projected with T-SNE.

Table 3.4: Results from experiments with D2. The best results for the clustering in the multidimensional space are yielded by the combination *BOW + BOW*. The data projected with T-SNE *Doc2Vec* yields the best silhouette score.

		Document model			
		BoW	**Doc2Vec**	**SBERT**	
Word model	**S (MD)**	0.1777 ± 0.0129	0.1027 ± 0.0081	0.1666 ± 0.0093	**BoW**
	S (t-SNE)	0.2683 ± 0.0565	-0.1077 ± 0.0390	0.3999 ± 0.0244	
	ARS	0.8819 ± 0.0249	0.3632 ± 0.0289	0.6963 ± 0.0265	
	AMI	0.7669 ± 0.0563	0.2135 ± 0.0274	0.5560 ± 0.0422	
	FMS	0.7899 ± 0.0512	0.2879 ± 0.0243	0.5985 ± 0.0380	
	V	0.8922 ± 0.0227	0.4191 ± 0.0266	0.7230 ± 0.0241	
	Time	0.5071 ± 0.0306	0.8848 ± 0.0624	9.9006 ± 1.2098	
	S (MD)	0.1891 ± 0.0061	0.0993 ± 0.0078	0.1620 ± 0.0090	**Word2Vec**
	S (t-SNE)	0.3114 ± 0.0340	-0.0986 ± 0.0296	0.3838 ± 0.0352	
	ARS	0.9016 ± 0.0112	0.3692 ± 0.0286	0.6935 ± 0.0195	
	AMI	0.8090 ± 0.0288	0.2198 ± 0.0274	0.5747 ± 0.0315	
	FMS	0.8283 ± 0.0258	0.2932 ± 0.0237	0.6151 ± 0.0284	
	V	0.9102 ± 0.0103	0.4247 ± 0.0261	0.7202 ± 0.0178	
	Time	0.6432 ± 0.0465	1.5988 ± 0.1059	18.9370 ± 1.5795	
	S (MD)	0.1886 ± 0.0049	0.0981 ± 0.0090	0.1643 ± 0.0089	**FastText**
	S (t-SNE)	0.3152 ± 0.0290	-0.1033 ± 0.0340	0.3968 ± 0.0296	
	ARS	0.9027 ± 0.0107	0.3625 ± 0.0305	0.6969 ± 0.0193	
	AMI	0.8076 ± 0.0262	0.2144 ± 0.0289	0.5734 ± 0.0313	
	FMS	0.8271 ± 0.0235	0.2881 ± 0.0254	0.6138 ± 0.0283	
	V	0.9112 ± 0.0098	0.4186 ± 0.0278	0.7234 ± 0.0176	
	Time	0.5521 ± 0.0251	1.5041 ± 0.1116	18.1232 ± 0.5175	

It is possible to infer that the small dataset size of only 381 documents had a negative impact on the *BOW* word model, resulting in a very sparse representation. In contrast, the *Word2Vec* and *FastText* models produce dense vector representations and are not impaired by the dataset size. In conclusion, the impact of dataset size becomes apparent in our study, with the relatively small dataset of only 381 documents having a discernibly negative effect on the BOW model. This impact is primarily attributed to the sparsity of the representation generated by the BOW model, where many dimensions in the vectors remain zero due to the limited vocabulary size in the context of a small dataset. This sparseness can be

detrimental to the performance of distance-based clustering algorithms, as the sparse vectors emphasize differences caused by zero values rather than effectively capturing the underlying semantic similarity between documents. In contrast, models like Word2Vec and FastText, which produce dense vector representations, are better equipped to handle the dataset size and do not suffer from the same level of impairment, as the dense vectors provide more robust and meaningful representations. Therefore, this study underscores the importance of considering dataset size and vector sparseness when selecting and applying vector representations in clustering tasks, as these factors can significantly impact the quality of clustering results.

Experiment 3: Dataset 9NewsGroup In this third experiment, one can spot a different behavior from the previous ones. Given that *Word2Vec* and *FastText* are machine learning models, as the number of training instances increases, the model will yield better results in general, which is observed in this experiment.
The dataset D3 is the largest we experimented with (1,080 documents). One observes in Table 3.5 that the strategy *BOW + FastText* has shown the best performance, in general, for the multidimensional space. Whereas the document representation strategy *SBERT* still shows better results for the projected data with T-SNE.

Dicussion From these experiments, it is possible to draw some insight regarding the selected representation models. Concerning the document representation models:

- **BOW:** produces better clustering results for multidimensional data. However, the clustering performance drops as the target dataset dimensionality increases.
- **Doc2Vec:** delivers satisfying silhouette scores for multidimensional and projected data with a reasonable number of documents. However, this model falls short when it is employed with fewer documents.
- **SBERT:** shows consistent results in all explored scenarios, yielding not state-of-the-art but average and sufficiently accurate results.

Table 3.5: Results from experiments with D3. The strategy *BOW + BOW* shows the best values for most of the clustering metrics in the multidimensional space. *Doc2Vec* yields the best silhouette score both for the multidimensional and the projected data.

	Document model: BoW	Document model: Doc2Vec	Document model: SBERT	Word model
S (MD)	0.0562 ± 0.0002	0.3897 ± 0.0152	0.1104 ± 0.0002	**BoW**
S (t-SNE)	0.3322 ± 0.0291	0.5049 ± 0.0193	0.4475 ± 0.0256	
ARS	0.6565 ± 0.0135	0.6228 ± 0.0206	0.6141 ± 0.0023	
AMI	0.6755 ± 0.0126	0.6361 ± 0.0322	0.6490 ± 0.0065	
FMS	0.7677 ± 0.0094	0.7404 ± 0.0224	0.7474 ± 0.0045	
V	0.6583 ± 0.0135	0.6247 ± 0.0205	0.6161 ± 0.0023	
Time	1.1267 ± 0.1467	1.3320 ± 0.1468	9.8179 ± 0.3803	
S (MD)	0.0559 ± 0.0002	0.3782 ± 0.0138	0.1110 ± 0.0009	**Word2Vec**
S (t-SNE)	0.3319 ± 0.0281	0.5113 ± 0.0151	0.4440 ± 0.0253	
ARS	0.6381 ± 0.0143	0.6219 ± 0.0216	0.6180 ± 0.0060	
AMI	0.6389 ± 0.0273	0.6561 ± 0.0296	0.6579 ± 0.0098	
FMS	0.7421 ± 0.0190	0.7538 ± 0.0209	0.7539 ± 0.0072	
V	0.6400 ± 0.0142	0.6239 ± 0.0215	0.6199 ± 0.0059	
Time	1.9488 ± 0.0540	3.1325 ± 0.0488	19.2984 ± 0.5818	
S (MD)	0.0559 ± 0.0002	0.3850 ± 0.0188	0.1108 ± 0.0017	**FastText**
S (t-SNE)	0.3312 ± 0.0278	0.5099 ± 0.0134	0.4420 ± 0.0258	
ARS	0.6391 ± 0.0146	0.6267 ± 0.0249	0.6168 ± 0.0089	
AMI	0.6408 ± 0.0261	0.6503 ± 0.0341	0.6560 ± 0.0140	
FMS	0.7434 ± 0.0182	0.7502 ± 0.0237	0.7525 ± 0.0100	
V	0.6410 ± 0.0145	0.6286 ± 0.0248	0.6188 ± 0.0089	
Time	1.7230 ± 0.0616	2.8604 ± 0.0687	18.9352 ± 0.4367	

And finally, concerning the word representation models:

- **BOW**: As in the document representation, BoW's performance is highly affected by the corpus size. The model yields better clustering results when the number of documents is in the middle ground (as in dataset D1, with nearly 600 documents). When presented with few documents the matrix representation will be very sparse, and when presented with a larger number of documents it will not scale properly.

- **Word2Vec**: This model has a similar performance to *FastText*, but the latter outperforms when combined with *BOW* document representation.
  Since *Doc2Vec* is a Word2Vec generalization, it is possible to observe cases where this particular combination can deliver satisfying silhouette scores.

- **FastText**: When combined with the *BOW* document representation, this

model can deliver satisfying clustering results, outperforming the remainders. It is also possible to observe that FastText scales well as the corpus size increases.

We have a selection of representation models that can be incrementally explored and combined to fit a user's needs in each particular case.

3.4.2 Expert Study

As an additional assessment, we conducted a study with a domain expert to obtain some preliminary information on the quality and effectiveness of the proposed solution in a specific domain, from an end-user perspective. Moreover, we leverage the Mod-Kt to visualize the dataset's content. In the following, we detail the experimental expert study and its results.

MALNIS Dataset Preparation To conduct the expert study, we created a new dataset of scientific articles in computer science. The data has been gathered from Machine Learning and Networked Information Spaces (MALNIS) research group archive from different subjects in text mining. The articles are archived in Zotero [1], an open-source bibliographic assistant for researchers. The MALNIS dataset [5] is comprised of 660 articles from 12 different categories, listed in Table 3.6. First, we extracted the BibTex file from Zotero, which contains different parts of the article. We collected from each article the Abstract, authors, title, journal, bibliography type, publication year and month, and URL. Then, we converted the dataset to a CSV file. Each abstract is considered as a single document and saved by its title concatenated by the publication year as the document's name. To upload the dataset in the interactive clustering system, we saved each document in a TXT file. So, the user can upload those TXT files into the system and get the initial clusters.

Expert Study Results The expert user in this study is a female Ph.D. student in computer science, with 10 years of experience in machine learning and NLP. The expert user was selected to conduct the study due to her familiarity with the dataset and domain, which enabled her to effectively evaluate the system's

[5] https://github.com/jarobyte91/malnis_data

Category	# of articles
AI and Grammar checking	4
COVID	12
Classification	30
Deep Active Learning	64
Embeddings, Language models	248
Multi-word terms and short-noisy text	5
Information Retrieval	118
Neural IR	11
Radiology Report	4
Visual analytics and deep networks	74
Interactive clustering	24
Attention mechanism	66
	660

Table 3.6: Topical categories of the MALNIS dataset, which contains papers in computer science.

capabilities. The user started by initializing the system with some initial setups based on her knowledge of the dataset. For instance, she employed the T-SNE projection and started with a setup of 3 clusters, as suggested by the initial unsupervised clustering. She then increased the number of clusters to 5, based on observing the projection result and on her knowledge about the corpus, while examining the Cluster view. The expert user can inspect the top words in each cluster, as shown in Fig. 2.4, to determine whether the current clusters match the topics of her interest. She found the new clustering result preferable and increased the number of clusters to 8 to further explore the possibility of better clusters. However, she was not satisfied with the 8-cluster result and reverted to the 5-cluster model. Most documents in this dataset pertain to language models, information retrieval, visual analytics, attention mechanism, and deep learning, as presented in Table 3.6. As a result, it is reasonable to expect better results with 5 clusters. Throughout the study, the expert resorted to the term similarity view to search for alternative terms, which helped her to retrieve additional related documents and identify the clusters most related to these terms. As an illustration, for the language model topic, she used the term similarity view to search for terms like BERT, find similar articles, and verify them in the corresponding cluster. Regarding the

difficulties faced by the expert while locating the appropriate article in the correct clusters, it is worth noting that some articles had been mistakenly categorized in Zotero. In such cases, the Mod-Kt successfully assigned those articles to the correct clusters based on their content similarity.

One possible limitation of the expert study is the potential for confirmation bias, where the expert's prior knowledge or expectations may influence the interpretation of the results.

The expert considered the final clustering result compelling and informative, with an organization reflecting her topics of interest. She was impressed by the clustering results and mentioned that using the Mod-Kt she managed to retrieve the related papers on a specific topic faster and more precisely than with a conventional search in Zotero or similar bibliography management applications. For example, she obtained larger clusters for topics related to language models and information retrieval, which was in line with her expectations. The expert study was completed in approximately one hour, of which 10 minutes were spent introducing the system, 40 minutes with the user performing the task, and 10 minutes conducting an interview with the expert. Results from the interview are summarized below.

Representation Models. In the first scenario, the expert user chose BOW as both the word model and the document model. Despite considering the individual words in the dataset representative of their respective topics, she was dissatisfied with the projection result. The clustering was acceptable, but the expert user found the dataset required stop-word filtering in addition to the general English stop-word list to improve the results. The second scenario was based on using FastText as the word mode and BERT as the document model. The projection obtained with this setup was considered much more convincing, but the top words in each cluster were too general since FastText failed to capture the most relevant words. In the final scenario, the expert user selected Word2vec for the word model and BERT for the document model. The top words in each cluster are too general, and the stop words problem remained. As a general finding regarding the representation models, BOW has better key-term mining compared with other word models. Doc2vec projection was the least promising model in terms of perceived projection quality, and BERT

was the best model choice for document embedding and low-dimensional projection.

Overall Goal. In general, the expert user expressed satisfaction with the interactive visualization and found the majority of the modules useful in practice. The Mod-Kt aided her in identifying various topics, including some that were unexpected in the dataset. For instance, the expert could use the Mod-Kt to separate the language model topics she was looking for. Additionally, the expert user fine-tuned the model to obtain improved clusters and top words.

3.5 Conclusion

In this chapter, we explore and experiment with document and word representation models applied to user-centered document clustering to better understand how such models behave and relate to each other in distinct controlled scenarios. In order to achieve this goal, we produced three contributions to user-guided clustering.
First, we address the gap between document and word representation models by introducing a novel modular framework that enables seamless interoperability between various representation models. This innovative framework serves as an intermediate communication layer between any given word and document representation models, provided they implement a common interface. This contribution points towards a better understanding of how to bring NLM closer to user-guided tasks required in knowledge discovery processes.
Second, we conducted experiments to evaluate the quality of clustering with different combinations of word and document language models. The results of the quantitative experiments indicate that the *BOW* word model is faster and yields better clustering results for handling corpora with an intermediate number of documents. However, when the number of documents is small (e.g., 381 documents in dataset D2) or larger (e.g., 1,080 documents in dataset D3), more recent models such as *Word2Vec* and *FastText* yielded better clustering results. As for document representation, the *BOW* model generally yields better clustering results for the multidimensional space, whereas the *SBERT* model yields better silhouette scores for the projected data with T-SNE.
The reason why better-separated multidimensional data does not yield the best

projections, despite being well-suited for clustering, is rooted in the different characteristics and objectives of multidimensional data and projected data. In the context of document representation, the BOW model performs well in the multidimensional space because it excels at capturing the intricate relationships and nuances between documents, which is essential for clustering tasks. It provides a richer representation of the data in its original multidimensional form, allowing for more accurate and meaningful clustering. On the other hand, SBERT, a model known for generating high-quality document embeddings, may not perform as well in the original multidimensional space. However, when the data is projected using T-SNE, SBERT excels in producing better silhouette scores. This indicates that SBERT is adept at reducing the dimensionality of data while preserving the semantic information that aids in the visualization of data in lower-dimensional spaces. In this context, the aim is not necessarily to cluster the data in its projected form, but rather to create a lower-dimensional representation that maintains the relative similarity and structure of the original data. So, the discrepancy arises from the different purposes of the multidimensional space, which is ideal for clustering, and the projected space, which is more suitable for visualization and maintaining semantic relationships in a lower-dimensional form.

This preliminary assessment indicates that none of the selected models outperforms its counterparts in the general case. Instead, we observed that different models exhibit better performance in specific scenarios. This suggests a need for a visual solution that allows a user to explore alternatives and select the representation model that best fits her expectations.

Third, we implemented a proof-of-concept testbed for interactive document clustering based on key terms, called Mod-Kt. The Mod-Kt supports the modular framework implementing visual representations and real-time interaction functions for the user to experiment with the various representation models available. It also allowed us to conduct an expert study to qualitatively evaluate the system's utility for a real user, with encouraging results.

As for future steps, we intend to perform a formal user study to assess the Mod-Kt's usability and utility to a broader range of potential users. Conducting experiments on additional document sets, including larger ones, is also important to gain further

insight into the interplay between the choice of language models and clustering quality. Moreover, the Mod-Kt is an extensible open-source tool that can be enhanced with additional representation models, allowing contributions by the research community. Thus, incorporating additional word and document models, such as GPT3 [13] and LLaMA (*Large Language Model Meta AI*) [98] would be another relevant future contribution.

Chapter 4

Demand Forecasting of New Products Using Language Models on The Product Descriptions

4.1 Introduction

Forecasting is a crucial element in various aspects of our daily lives, and sales forecasting is particularly significant for companies to enhance their business strategies and attain a competitive edge[1]. Every business has an underlying need to anticipate future revenue and sales. Predictive analytics and business intelligence are essential tools for boosting sales and facilitating strategic planning across all industries. The fundamental concept involves gathering historical sales data and utilizing it to predict future sales, which is then used to make informed decisions. The majority of studies focused on demand forecasting utilize historical sales data and time series analysis. Time series forecasting involves predicting how a set of data points will evolve over time [10]. Nevertheless, forecasting is a complex undertaking owing to the following factors: Firstly, dealing with large quantities of historical data can lead to the loss of crucial information from the past. Conversely, making decisions based on a limited history can also be challenging. Secondly, there can be difficulties in handling related but independent data such as holidays, locations, and significant events. Additionally, the accuracy of the forecast is crucial for a company's future decision-making process. If the forecast is too high, it may lead to over-investment, resulting in a decline in revenue. Conversely, if the forecast is too low, the company may miss opportunities due to under-investment. Furthermore, there are several external factors, such as weather, time, and location, that can influence the accuracy of the forecast.
This chapter introduces a forecasting method [2] for estimating the sales of new

[1] This chapter is the extended version of the journal paper submitted in Knowledge-based systems (under the review): https://www.sciencedirect.com/journal/knowledge-based-systems

[2] https://github.com/SimaRezaeipour/Demand-Forecasting

products with a limited sales history. Predicting the sales of recently launched products can be difficult due to their shorter life cycle and a lack of extensive historical sales data for a considerable number of products in numerous companies. To address this challenge, we incorporate product descriptions into the forecasting process by utilizing state-of-the-art language models to encode them. Our proposed approach involves adding product descriptions as a new feature and initially clustering listed products in stock based on the similarities in their descriptions. Next, the approach assigns new products with limited sales history to the relevant clusters and reruns the forecast model to predict sales. During this phase, we employed time series data, similar to what is conventionally done in traditional demand forecasting. We also compare and utilize various widely used forecasting models. To assess the effectiveness of our approach, we carried out a comprehensive case study on newly launched products using data obtained from an online retailer. Our experimental findings indicate that the proposed approach can effectively enhance the accuracy of sales forecasting.
In the following, we itemize the contributions of this study:

1. Incorporating product descriptions as a new feature: Introducing a novel way of using product descriptions for sales forecasting, which is not typically used in traditional forecasting methods.

2. Overcoming data limitations for new products: Addressing the challenge of limited sales history data for new products, which can make sales forecasting difficult.

3. Cluster-based forecasting: Clustering similar products based on their descriptions and assigning new products to relevant clusters, can help to identify demand patterns and enable more accurate predictions.

4. Case study validation: Validated through a comprehensive case study on newly launched products using data obtained from an online retailer, which confirms its effectiveness in enhancing the accuracy of sales forecasting.

5. Enhanced accuracy: Utilizing product descriptions enhances sales forecasting by providing a deeper understanding of the product's characteristics.

The structure of this chapter is as follows: We review relevant literature and discuss related work in Section 4.2. We then present our proposed method including the language models, clustering algorithms, and forecasting methods employed in Section 4.3. An overview of the data and case study is provided in Section 4.4. We present the experimental results and evaluation metrics in Section 4.5. Finally, we conclude our research in Section 4.7.

4.2 Related Work

Accurate predictions are critical for businesses in numerous application domains. In this section, we will provide an overview of relevant research on demand forecasting in various fields, including food, fashion, transportation, tourism, energy, weather, and the economy. Numerous studies have been conducted on demand forecasting using time series data. Sales forecasting tasks can be approached using expert knowledge, statistical techniques, or Machine Learning (ML) methods. Compared to existing product demand forecasting, forecasting demand for new products is more challenging due to the lack of historical data. To overcome this issue, numerous quantitative methods have been proposed for forecasting demand for new products. These models leverage similar products to predict demand for new products, as similar products tend to exhibit comparable demand patterns.
DemandForest is a novel machine learning-based method for predicting demand for new products, proposed by Van in [103]. This method combines Kmeans clustering, Random Forest, and Quantile Regression Forest. It leverages the historical sales data of current products and incorporates new product features to make predictions. A categorical data embedding representation is used to map similar values close to each other in the embedding space in [38]. The authors employ conventional neural networks to identify patterns in the data and use entity embedding for visualizing categorical data and data clustering. A survey to review ML approaches for short-term sales prediction in the field of food sales forecasting is reported in [99]. ML models have acceptable performance, but their quality heavily relies on the availability and quality of data, which can be limited for new products. Statistical and ML models alone are not sufficient to obtain high-quality forecasts for new products, and expert knowledge can improve forecast quality [48]. However,

employing biased experts is costly and time-consuming. To address this problem, Transfer Learning has been applied in some research as an analytical approach for ML models to imitate human behaviour and transfer information between similar products to improve prediction in new products [107, 48]. Product descriptions are split into a sequence of tokens and used to predict sales, brand, and price for a Japanese e-commerce marketplace [75]. Albeit, their dataset is not public.
The fashion retail industry faces challenges in forecasting sales for new products due to the lack of historical sales data, and the short life cycles of fashion products. A deep learning framework for forecasting sales in the fashion industry, utilizing various features such as the physical characteristics of the products and the domain expert's view is presented in [60]. Word embeddings, convolutional layers, and attention mechanisms are used to combine text data with time series to address the taxi demand forecasting problem in [83]. They collect some information from the web such as pickups from the previous days, weather and events, to predict the number of pickups of the next day. Their goal is to improve time-series forecasts by combining temporal data with information in unstructured text format.
Deep learning architectures are also utilized for online media text mining using CNN, RNN and LSTM for oil price forecasting in [58, 110]. They grouped online news media by their topics in price forecasting to enhance accuracy. They used the LDA Topic modelling technique to identify different topics. Tourism forecasting using online news data mining and Google trend data is discussed in [71, 42, 89, 118]. The use of LSTM and FB Prophet for predicting air temperature is explored in [97]. The effect of customer reviews on demand forecasting in flash sale platforms is investigated in a few research works, where data streams from customer reviews are found to contain useful information to improve sales forecasting [116, 88].

4.3 Methodology

In this section, we present the research methodology utilized in this chapter. Numeric features, such as historical sales data, are widely used in e-commerce to forecast sales performance in the upcoming period. Since historical sales data provides a baseline, these features are crucial for accurate forecasting. The main

objective of our research is to incorporate product descriptions as a new feature in addition to historical and time series data. We can obtain a complete picture of the current sales performance by combining these two features. To add product descriptions as a new feature, we extract textual features from product descriptions, group them into clusters and employ a regression model to predict the sales for each product within each cluster. This approach has not been previously employed or explored in the context of sales forecasting. We evaluate various regression algorithms, such as LightGBM, Autoregressive Integrated Moving Average (ARIMA), Catboost, and Facebook Prophet, to develop a demand prediction model. The following summarizes the seven primary research questions addressed in our study.

1. How to extract the features from the product description?
2. Will the addition of product description as a new feature to time series data result in improved sales forecast accuracy?
3. How does clustering the data before regression affect the sales forecast accuracy?
4. Can clustering products based on their descriptions enhance sales forecast accuracy for new products?
5. Which language model is optimal for extracting features from product descriptions?
6. What is the most precise forecasting model for sales?
7. Which dimensionality reduction techniques provide the best performance in terms of sales forecasting accuracy?

The clustering-based forecasting model is based on the concept of employing a clustering algorithm to divide the entire training data into multiple clusters and create a forecasting model for each cluster. In our case, the test data, which includes new products, are assigned to a specific cluster based on the similarity of their product descriptions, and the prediction model of that cluster is utilized to obtain the forecasting results. Since the data in the same cluster share similar data

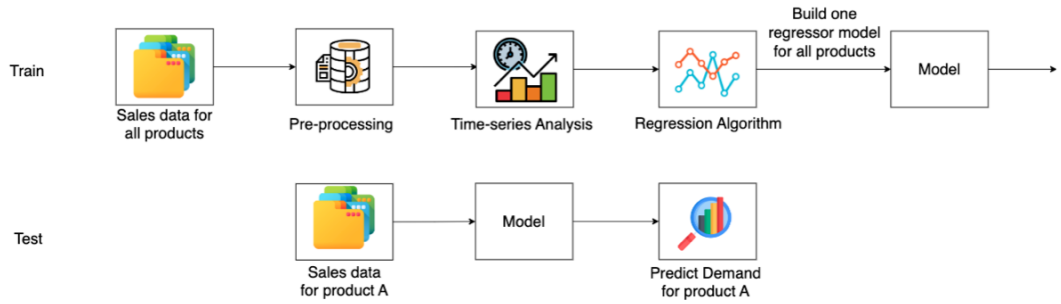

Figure 4.1: Traditional approaches for time series forecasting. The training phase is illustrated in the top figure, while the bottom figure represents the testing phase. Traditionally, demand forecasting models have primarily relied on historical sales data to predict future demand for a product, without considering the product description as a feature. Such models use time series data from all products to train a model to forecast demand for a given product. However, our approach incorporates product description as a feature and only considers time series data from similar products to predict demand. In traditional approaches, during the pre-processing phase, the relevant information is extracted from the Date feature while applying noise cancellation and data reduction techniques. In the time series analysis phase, important features are selected. Subsequently, different regression algorithms, such as ARIMA, are used to train prediction models. During the testing phase, the model receives a series of historical sales data for a specific product, which it uses to predict demand for that product.

patterns, the clustering-based forecasting model can produce more accurate forecasts than the forecasting model built using the complete dataset. In this study, we employed Kmeans or HDBSCAN to cluster the data, while LightGBM, ARIMA, Catboost, and Prophet were utilized as predictors.

The forecasting task can be generally divided into the following fundamental steps: Defining the problem, data analysis, data pre-processing, feature engineering, selecting the prediction method, fitting the model, prediction, and evaluation. The workflow of traditional approaches is illustrated in Fig. 4.1. In contrast, the methodology developed in this study is depicted in Fig. 4.2 and Fig. 4.3, and it consists of two phases: Training and testing. A detailed description of each phase is provided in the following subsections.

4.3.1 Training Phase

The training dataset consists of items that contain historical data. The primary objective of the training phase is to construct a regression model for each cluster obtained by the clustering algorithms. To achieve this goal, we applied several interconnected steps. A description of each of these steps and how they are linked during the model's development is provided below.

- *Data Pre-processing*

  In the first step, the input data which are sales records, are pre-processed. As depicted in boxes 1, 1-a and 1-b in Fig. 4.2. we divide our input data into *ProductDescription* and *Salesdata*. We applied different pre-processing techniques for each part. To pre-process the product descriptions, which are text data, we utilized different Python packages such as NLTK, and Texthero (box 1-a). On the other hand, to pre-process the sales data, we removed the outliers from the product quantities by eliminating records with the number of quantities outside a certain range (box 1-b). In both parts of the data, we employed additional pre-processings, such as dealing with missing descriptions or zero unit prices, focusing on transactions with prices and quantities that fall within the specific range, and so on.

- *Feature Extraction*

  In this step, we start by defining the target variable, which is the product quantity in our empirical study. Next, we employ various language models, such as RoBERTa, DistilBERT, and SimSCE, to encode the product description feature, as illustrated in box 2 of the figure. We then aggregate the data at the daily, and monthly levels by computing daily, and monthly product sales as the features. To achieve this, we extract temporal attributes from the date feature, as shown in box 4.

- *Cluster training data using Kmeans or HDBSCAN algorithms*

  The next step involves applying clustering algorithms such as Kmeans or HDBSCAN to the encoded product descriptions to group them into distinct clusters, as shown in box 3. For the Kmeans clustering algorithm, we

determine the number of clusters, denoted as k, by using the elbow method and in our specific instance, this value is set to 10. Once the training data have been clustered based on the similarity of their product descriptions, we extract the sales data for each cluster, which is presented in box 5. After clustering, we calculate the average embeddings of each cluster to represent the cluster vector.

- *Build a regressor model for each cluster*

  In this step, a regression model is constructed for each cluster as shown in box 6. As previously mentioned, the selected regression models in this study are LightGBM, ARIMA, Catboost, and Prophet. The LightGBM model is optimized using Optuna. The output of this step is a regressor model for each cluster as illustrated in box 7.

4.3.2 Testing Phase

In the absence of historical sales data for newly introduced products, forecasting their sales becomes challenging. However, by analyzing the sales of historical items and the characteristics of current and new products and comparing them, we can identify similarities that enable us to forecast sales. The objective of the testing phase is to assign new products to the most appropriate clusters based on their description similarity. This process allows us to leverage the sales history of similar products and use it to generate sales forecasts for new items. By using this approach, we can improve the accuracy of our sales forecasts and provide valuable insights for businesses introducing new products.

- *Collect testing data*

  In this study, we consider products that have been introduced in the last two months as *newproducts*, which we define as test data. Predicting the future sales of new products can be challenging due to their limited historical data. To address this issue, we employ a clustering approach that assigns new products to clusters based on similarities in their descriptions. This enables us to group new products with similar products that have more historical data, thus improving our ability to make accurate sales predictions.

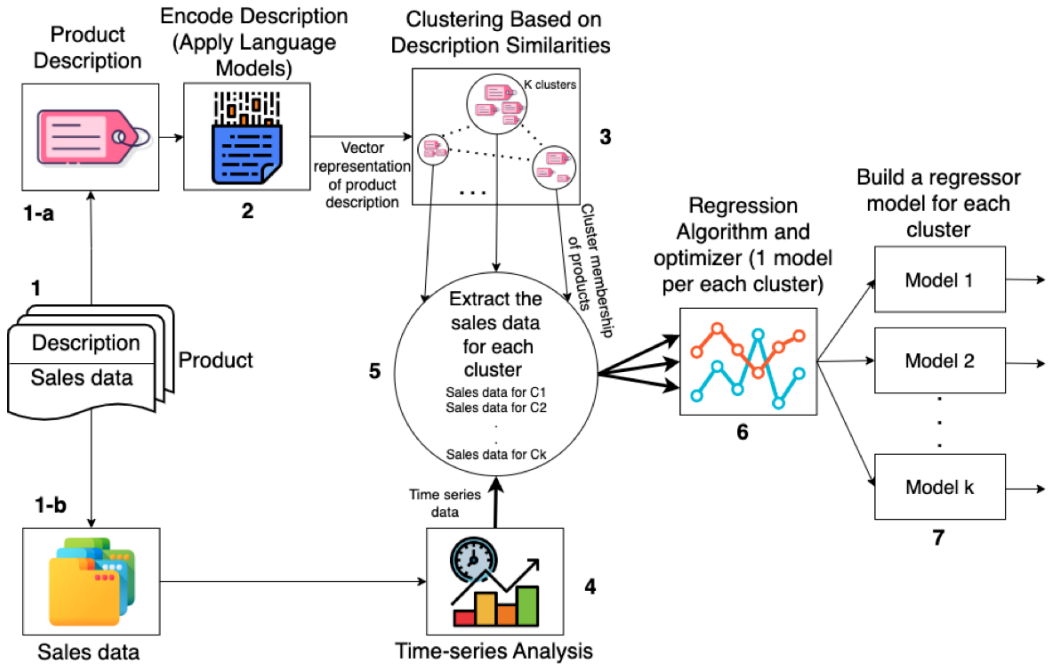

Figure 4.2: The training phase of the proposed methodology for predicting demand. The process includes data pre-processing (#1), feature extraction (#2 and #4), data clustering (#3), and model construction (#6 and #7). The workflow starts with data pre-processing (#1, #1-a, and #1-b), where product descriptions are cleaned (#1-a) and sales data is processed (#1-b). Feature extraction is then performed using various language models (#2) and time-series analysis techniques (#4). The next step involves clustering the vector representations of product descriptions using Kmeans or HDBSCAN algorithms (#3) and extracting the time-series data associated with each cluster (#5). For each cluster, regression algorithms such as LightGBM, ARIMA, Catboost, and Prophet are applied to the time-series data (#6) to build a regressor model (#7).

- *Feature extraction*

  This step is similar to the feature extraction process in the training phase. As previously mentioned, we use several language models to convert the product descriptions into vectors, which are then used to determine their similarities to each cluster. This approach allows us to effectively measure the degree of similarity between new products and products in existing clusters, which is an essential step in our sales prediction model.

- *Assign new products to appropriate clusters*

  To determine the similarity between the new products and the clusters obtained during the training phase, we first attain a vector for each cluster by computing the average of the embeddings of its product descriptions. Similarly, we calculate a vector for each new product description. Next, we use cosine similarity to assign new products to a cluster, selecting the cluster with the highest similarity score. The most appropriate cluster for a new product is the one with the lowest cosine similarity score. We use cosine similarity as a measure of similarity because it is a widely used and effective similarity metric for vectorized data.

- *Demand prediction*

  As previously stated, we constructed a model to predict demand for each cluster. Once we have determined the most appropriate cluster for each new product, we rerun the models that were built during the training phase for that particular cluster. These models are then used to forecast sales for the new products, based on the historical data of products that belong to the same cluster. By leveraging the historical sales data of similar products in the same cluster, we can improve the accuracy of our sales predictions for new products.

4.3.3 NLM

To approach the feature mining problem, we first encode product descriptions using language models that directly extract features from text data. Typically, language models require significant computational resources to train due to their large and

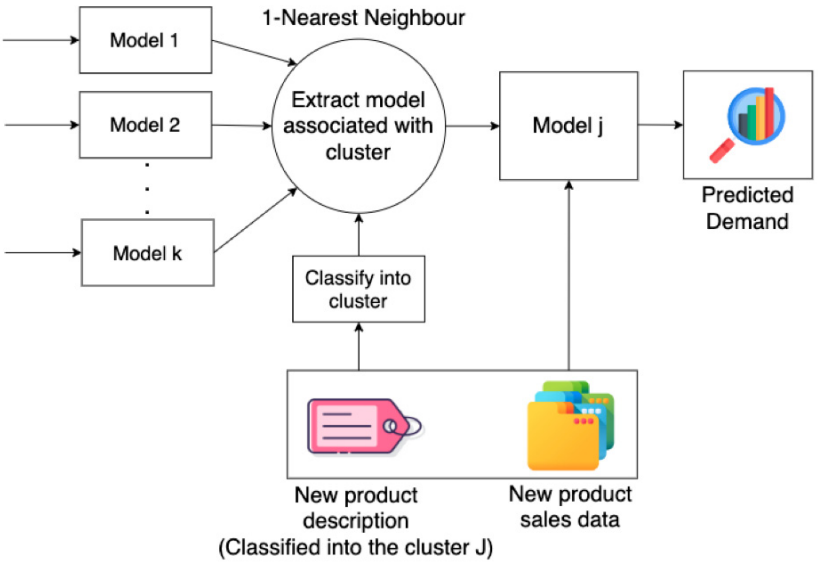

Figure 4.3: Demand prediction for a new product. The input consists of the new product description and its corresponding sales data. The new product description is converted to a vector using the same language model used during training. This vector is then used to classify the new product into the closest cluster (cluster J in the figure). Models 1 to k represent the output of the training, from Fig. 4.2 (#7). The corresponding Model j for cluster J is utilized to predict demand based on the available sales data for the new product.

complex nature. This necessitates access to powerful GPUs, a large memory and storage capacity, high-performance CPUs, and a potent cluster or cloud computing resources. However, fine-tuning pre-trained language models for specific tasks like text classification, question answering, or language generation requires fewer computational resources than training them from scratch. Fine-tuning only requires adapting the pre-trained weights to the new task rather than training the entire model. Pre-trained language models have significantly improved performance in recent years. Since product descriptions are typically sentence-sized, we must use different sentence embedding techniques to map them to a vector. Thus, we have not trained any language model on our data, and we are utilizing pre-trained versions to generate sentence embeddings and capture the semantic similarity between product descriptions. Once we choose a pre-trained model, we must recognize that various language models have different structures and parameters. Additionally, they typically use specific tokenization techniques to convert input text into a format that their model can interpret.

The output of the encoded product descriptions using pre-trained language models is represented as fixed-length vectors. These models require a specific input and output format. First, we pre-process the product descriptions and send them as input for the pre-trained language models. We obtain the vector representation of the product description by using the output from the last hidden state of the model. The last hidden state of the model typically has a dimension equal to the number of the model's hidden layers, which ranges from 512 to 1024. In the following, we describe three different sentence embedding algorithms employed in this research, to capture the distance between individual words. Additionally, we used TF-IDF as a baseline feature encoding model [79]. This encoding method assigns greater weight to more discriminative words in the corpus.

RoBERTa [59] is an optimized transformer-based model that shares the same architecture as BERT[26]. It uses a byte-level BPE as a tokenizer and is pre-trained with a vast dataset by Facebook. RoBERTa has 12 layers, 768 hidden neurons, 512 sequence lengths, 12 self-attention heads, and a total of 175M parameters, which enables it to achieve significant results on widely-used NLP benchmarks. In this research, we utilized RoBERTa as one of the pre-trained language models to encode

product descriptions in our dataset. Specifically, we used the RoBERTa-base-v2 model from the Hugging Face sentence transformers library, which provides an implementation of RoBERTa and handles all the pre-processing, tokenization, and conversions necessary to properly format the input and output. We set the max_seq_length to 100 and do_lower_case to False. The output of RoBERTa is a dense 768-dimensional vector space.

DistilBERT is a distilled (approximate) version of BERT that reduces the size of the BERT model by 40 percent by learning with only half the number of parameters [86]. DistilBERT is smaller and faster than RoBERTa on prediction metrics. Since DistilBERT does not have token_type_ids, it uses the separation token tokenizer to separate segments and only needs to indicate which token belongs to which segment. DistilBERT has 6 layers, 768 hidden neurons, 12 attention heads, and 66M parameters and can be fine-tuned to improve performance on a wide range of tasks. In this research, we used the distilbert-base-cased model from the Hugging Face library with the same output size and number of parameters as RoBERTa.

SimCSE is a simple contrastive learning framework that is used to train sentence embeddings by combining the word embeddings of the words in a sentence to generate a fixed-length sentence embedding [37]. SimCSE can handle both labelled and unlabelled data and is based on the idea that sentence embedding should capture the similarity relationships among the words in the sentence. In this research, we used unsupervised SimCSE, specifically the unsup-simcse-bert-base-uncased model developed by the Princeton NLP group. This model utilizes a single-layer neural network with a hidden size of 512 to encode product descriptions as sentence embeddings.

4.3.4 Clustering Algorithm

In this study, we used KMeasns as a partitional clustering algorithm and HDBSCAN as a hierarchical clustering algorithm. To cluster the products we used Kmeans, as the most common clustering algorithm introduced by Kanungo in [47]. By using Kmeans, we aim to minimize the inter-cluster distance. To identify the optimal number of clusters for Kmeans clustering, we applied the elbow method. It is a heuristic method involving plotting the Within-cluster Sum of Squares (WCSS)

as a function of the number of clusters and selecting the number of clusters where the curve starts to level off or form an elbow. Based on the resulting plot, we set the number of clusters (or K) to 10. We used random initialization to choose the initial centroids randomly from the set of product descriptions.

HDBSCAN is a hierarchical density-based clustering algorithm which uses the density feature to cluster the data and works well on applications with noise, developed by Campello in [15]. The reason that we used HDBSCAN in this research is that it implements soft clustering and supports outlier detection. In our implementation of the HDBSCAN algorithm, in addition to specifying the Euclidean distance metric, we fine-tuned the hyperparameters min_cluster_size and min_samples to control the density of the resulting clustering. Specifically, we set min_cluster_size to 50 and min_samples to 10. The min_samples hyperparameter focuses on the density of individual points, while min_cluster_size controls the minimum size of a cluster. By adjusting both parameters, we aimed to achieve the desired level of granularity in the resulting clusters while ensuring that the noise and outliers are detected accurately. In addition, we utilized the allow_single_cluster hyperparameter, which allowed us to form a single cluster even if it contained fewer samples than the min_cluster_size. This hyperparameter is particularly useful when dealing with datasets that contain a large number of small clusters, as it allows us to group them together into a single cluster, thus reducing the overall complexity of the clustering results.

When dealing with high-dimensional vectors, dimension reduction techniques are helpful for embedding the high-dimensional data into lower dimensions, while preserving as much of the original information as possible. In the following, we will refer to the dimension reduction techniques that we applied in this project to improve the clustering results and for visualizing high-dimensional data in 2D. First, T-SNE is considered a gold standard for 2D projection. The goal is to maximize the likelihood that similar samples are placed close together [101]. Second, UMAP is competitive with T-SNE for visualization quality, produces more clustered results, is faster than T-SNE and is introduced by McInnes in [62]. We applied the following parameters for UMAP; n_neighbors=15, n_components=2, min_dist=0.0, metric=*cosine* and set the parameters for T-SNE as below. n_components=2,

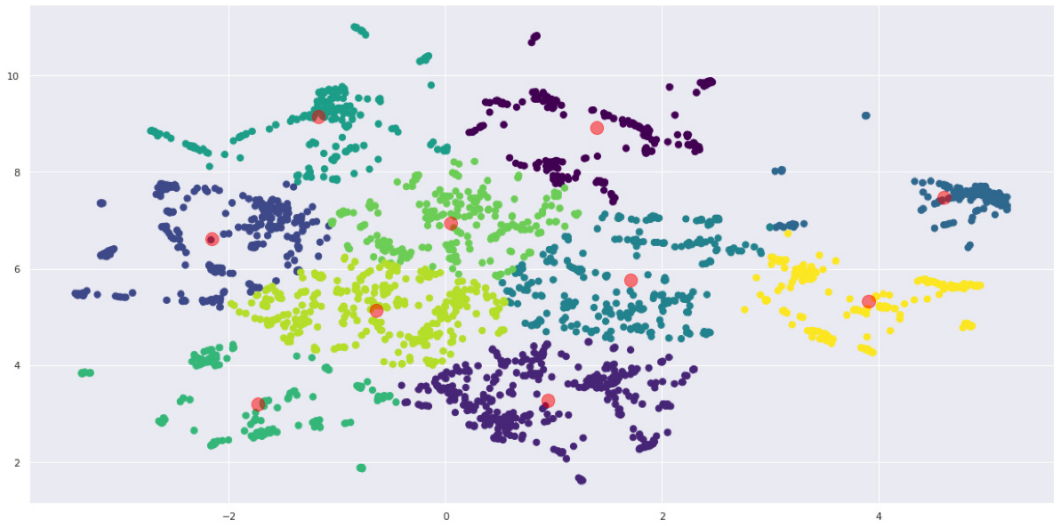

Figure 4.4: Put products in different clusters based on their description's similarity. First, encode the product descriptions using RoBERTa language model. Then, applying UMAP to project encoded product descriptions to 2D. In the end, passing the 2D array as an input of the Kmeans clustering and setting K=10. The red dots in the centre of each cluster depicts the centroid of each cluster. We assume that similar products show the same trend in their sales data, and they have closer vectors in the 2D projection. We use this in the testing phase to predict the demand for new products with similar descriptions.

learning_rate= 700, init=*random*. A simple example of how to fit data in a two-dimensional space with UMAP and utilizing the Kmeans clustering algorithm is depicted in Fig. 4.4. The same example, using HDBSCAN and T-SNE is illustrated in Fig. 4.5

It is important to choose an appropriate distance measure since it influences the result of clustering. In our experiments, we use cosine distance as a similarity measure for Kmeans and Euclidean distance for the HDBSCAN clustering algorithm. In a large document collection, the goal is to discover latent semantic structures or topics. Latent Dirichlet Allocation and Probabilistic Latent Semantic Analysis are the most popular topic modellings which rely on the bag-of-words representation of documents. In this research, a topic modelling tool is utilized to obtain a topic for each cluster, named Toc2Vec [4]. Top2Vec automatically detects the number of topic vectors that are indicating semantic similarity between the document and word vectors that are mutually embedded. To create a joint embedding of the document and word vectors, we used Doc2Vec. In the next step,

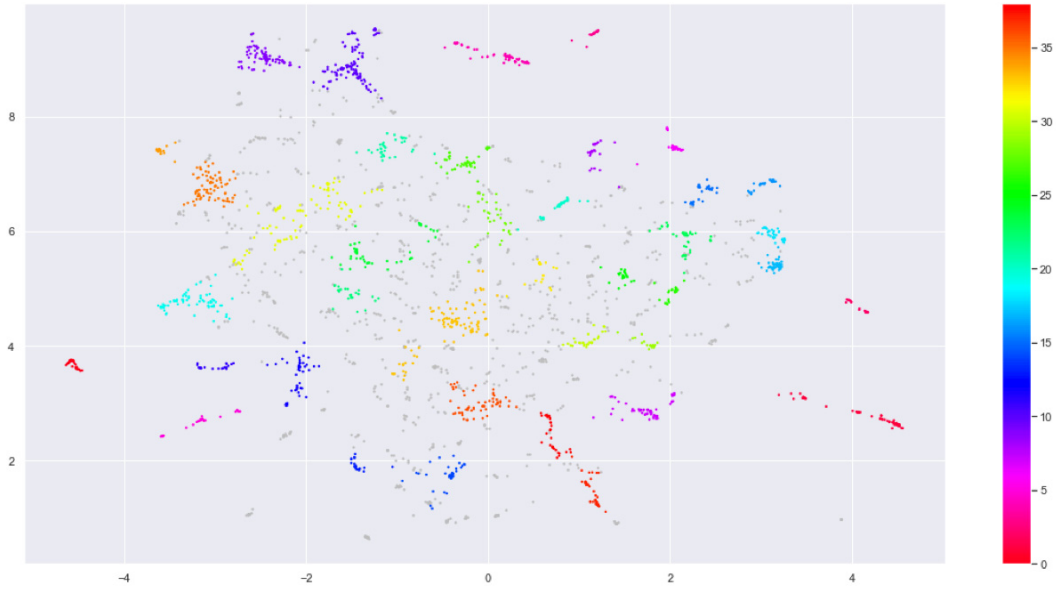

Figure 4.5: Visualizing the results of HDBSCAN clustering algorithm using RoBERTa language model to encode the product descriptions. We applied T-SNE as a dimensionality reduction technique to project encoded product descriptions to 2D.

UMAP is used as a dimensionality reduction technique. HDBSCAN is utilized in the next step to finding dense areas of documents. Then, the centroid of document vectors is calculated for each dense area, which is the topic vector. In the final step, the topic vector is obtained from the n-closest word vectors. The input of our topic modelling is a list of strings from each product description. Two important parameters used for topic modelling are speed and workers. The speed determines how fast the model will train. The fast-learn option generates low-quality vectors. We picked the deep-learn option which takes a longer time to train, although produces high-quality vectors. To train the model we utilized a large number of worker threads that led to faster training. As an example, the word cloud of one of the topics is presented in Fig. 4.6.

As we mentioned before, we applied another approach to group products using their descriptions. Here we use unsupervised K-Nearest Neighbor (K-NN) as an alternative clustering algorithm simultaneously. It can act as an unsupervised dimensionality reduction tool capable of performing dimensionality reduction and clustering [76]. The data points can be represented by the set of their k nearest neighbours, effectively reducing the dimensionality of the data from the original

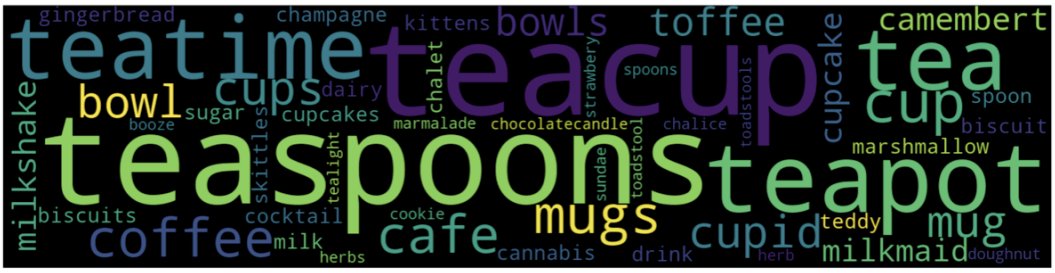

Figure 4.6: Illustrating the word cloud of one of the topics generated by the Top2Vec topic modelling approach. As demonstrated here, the products in this topic are mostly associated with tea cups and related items.

feature space to the space of the k nearest neighbours. It takes a bunch of product descriptions' vectors and tries to group them into K number of clusters. The goal is to maximize inter-cluster separability in a set of clusters. As a distance metric, we utilized Euclidean distance to find the k nearest neighbours.

4.3.5 Time Series Feature Engineering

Extracting useful features from time series data is an essential part of the forecast before running the model. Therefore, in this section, we examine applied approaches in our research to extract and select the important features for forecasting to increase the forecast accuracy. Many time series show trends, cycles or seasonal patterns which are the components of time series data. Over a long period of time, the trend is a pattern that indicates a movement, caused by an external event, in time series data that changes the regular pattern. The trend happens for a while and does not repeat. A periodic variation in time series data is called a cycle. It occurs when the data show rises and falls without a fixed frequency and it happens due to various external conditions. Seasonality is a recurring pattern over a regular period of time. Time series data may have multiple seasonal patterns such as daily, weekly and monthly [9].

In this research, we extract the following patterns from the date feature in our dataset: Day, week, month, quarter, and year. Time series usually demonstrate a combination of trends and/or seasonal patterns. Selecting the most important features can improve the forecasting model. Reducing the number of features, not only can prevent the model from over-fitting, but also speed up the training time.

There are several methods to evaluate the importance of features such as the Chi-square test, Correlation Coefficient, and Recursive feature elimination (RFE). We utilize a t-test which is a statistical test to compare the means of two groups. We will discuss the results of this test later in Section 4.5. After extracting adequate features, we utilize different methods to predict sales. Since there is no one forecasting method that performs best for all time series, we use various state-of-the-art forecasting methods which we will introduce in the following sections.

4.3.6 Forecasting Models

Sales forecasting is the process of estimating future sales revenue for a business or organization. The goal of sales forecasting is to provide a realistic forecast of future sales that can be used to make informed business decisions and plan for future growth. A sales forecasting problem is usually defined by a set of historical sales data that is used as input to a forecasting model. The model is trained to learn patterns in the data and predict future sales. The problem definition also includes the specific time horizon for the forecast and level of granularity, which is the sales quantity per product in this research. The output is a single number representing the total expected sales for a given time period, which in our experiment includes both daily and monthly sales for each product. In this section, we describe the applied forecasting models and relate them to our methodology. The models of this section are represented in boxes 6, 7, and 8 in Fig. 4.2. The procedure for predicting events based on their past and present values is a time series forecasting technique. In this subsection, we demonstrate detailed characteristics of the applied forecasting methods that we adapted to our forecasting scenario and have a superior history in the field of time series forecasting. A brief description of applied forecasting models and their corresponding optimization algorithms are presented below.

ARIMA We use ARIMA as a baseline, which is one of the most popular and widely used statistical analysis models for predicting future trends using time series data [41]. We implemented the general ARIMA class with three distinct input parameters:

- p: Order; The number of lag observations in the model. To capture temporal dependencies in the data we used 13 lagged observations as predictors in our experiment.
- d: Degree of differencing; the number of times that the raw observations are differenced.
- q: The size of the moving average window.

To tune the forecasting algorithm and find the best set of parameters, we applied grid search and set various combinations of values for the above-mentioned parameters p, d and q. The optimization criteria used by grid search select the parameters as follows to obtain the lowest Root Mean Square Error to make predictions; p = 13, d = 1, q = 5.

LightGBM LightGBM is a fast gradient boosting framework based on a decision tree algorithm [49]. This algorithm is relatively new and has been used for various machine-learning tasks such as ranking and classification. In this study, we employed the LightGBM regressor model to forecast sales. To guarantee high performance and avoid over-fitting, LightGBM set a limit on the maximum depth of the leaf nodes during its leaf-wise approach [19]. In this chapter, to tune the hyper-parameters of the LightGBM model, the Optuna optimization algorithm has been applied [2]. Optuna is an open-source hyper-parameter tuning optimization algorithm that is efficient for pruning and searching procedures. In general, Optuna is an efficient optimization algorithm in terms of searching and performance estimation strategy. The applied parameters of the model are presented as follows.

- metric: RMSE
- random_state: 48
- n_estimators: (200, 300, 500)
- reg_alpha: (1e3, 10.0)
- reg_lambda: (1e3, 10.0)

- colsample_bytree: [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
- subsample: [0.4, 0.5, 0.6, 0.7, 0.8, 1.0]
- learning_rate: [0.007, 0.01, 0.014, 0.017, 0.02]
- max_depth: [10, 20, 100]
- num_leaves: (30, 200, 1000)
- min_child_samples: (10, 300, 1000)
- cat_smooth: ($min\_data\_per\_groups$,1 ,100)

Catboost Catboost is an open-source machine learning library introduced by Dorogush in [29]. One of the remarkable features of Catboost is its ability to handle missing data and categorical data, which makes Catboost a fitting model for feature engineering tasks. One of the main aspects of CatBoost is its ability to merge different types of data into one framework. The reason that we picked this model as one of the forecasting algorithms in our experiments is that CatBoost does not follow similar gradient boosting models in the growing procedure and the leaf index is calculated with bit-wise operations and find an optimal solution and avoid over-fitting. Below, some of the parameters used are shown.

- metric: RMSE
- loss: RMSE
- iterations: 1,000
- max_depth: 4
- l2_leaf_reg: 3
- learning_rate: 0.5
- seed: 0

Prophet Prophet is an open-source software developed by Facebook [96]. It is a generalized additive model for time series data and contains three main model components: Trends, seasonality, and holidays. As Prophet was precisely designed for data with non-linear trends and contains weekly and daily seasonality seems like a proper candidate for our sales data. We didn't apply any optimizations for this model. The parameters of the models are shown below.

- changepoint_range: 0.8
- growth: linear
- seasonality_mode: [*additive*, *multiplicative*]
- seasonality_prior_scale: [0.01, 1, 5, 10, 12]

4.4 Data Preparation- Case Study

The application domain of this research is to forecast the sales of an online retail company in the E-commerce industry in the UK. The company carries a diverse range of products and has provided sales data through Kaggle [3]. The dataset used in this study consists of 541,909 records with six features including *Country*, *Description*, *StockCode*, *InvoiceDate*, *Quantity*, and *UnitPrice*. The aim is to assist the retailer by predicting daily and monthly sales of products. The data was collected between January 2010 and February 2012 and includes records from 4,315 customers across 37 countries. The majority of records are from the UK, which is also the primary market for the retailer, followed by several other European countries, as shown in Fig. 4.7. The uniqueness of this dataset is that the product descriptions are the only textual attribute in the data, and there are no product categories provided to classify products. Thus, we leveraged product descriptions for clustering, as there are 3,000 unique product descriptions available. In the retail industry, product descriptions refer to the textual information provided by retailers about their products. These descriptions typically include details such as the product's name, features, dimensions, materials, and other relevant information that can help customers make informed purchasing decisions.

[3] Kaggle data

Examples of product descriptions	# of character	# of words
SET OF 6 3D KIT CARDS FOR KIDS	30	8
SET OF 10 LANTERNS FAIRY LIGHT STAR	35	7
SET/5 RED RETROSPOT LID GLASS BOWLS	35	6
BUNDLE OF 3 ALPHABET EXERCISE BOOKS	35	6
CHRISTMAS STAR WISH LIST CHALKBOARD	35	5
CLEAR DRAWER KNOB ACRYLIC EDWARDIAN	35	5
JUMBO BAG SCANDINAVIAN BLUE PAISLEY	35	5
SKULLS STICKERS	16	2
POPCORN HOLDER	14	2
PARTY BUNTING	13	2
SPACE OWL	9	2

Table 4.1: Random samples of product descriptions with different sizes are illustrated here. As shown in these samples, product descriptions comprise essential details such as the product's name, characteristics, size, and material composition. The length of the descriptions varies from 1 to 8 words, with a character count of 9 to 35.

A few product description samples along with the number of characters and words in each description are presented in Table 4.1. The table illustrates that the retailer offers a wide range of products, and most product descriptions are predominantly in capital letters. As part of the preprocessing, we removed product descriptions with lowercase letters. Additionally, there are some variations in the product descriptions, which may result from typos or missing words. The product description's length and the corresponding number of products with that length are demonstrated in Fig. 4.8. The figure displays two sets of information. On the left, it shows the length of the product descriptions in characters, while on the right, it illustrates the number of words in each description. The product descriptions predominantly consist of 3 to 6 words and have a character length of around 22 to 35 characters. The target attribute in our data is Quantity, which has a wide range of distribution. Outliers with highly unrealistic quantities were identified and removed from the dataset. Most quantities in our records are between 1 and 12. To predict daily and monthly sales, we extracted the temporal features from the Invoice Date, including $Year$, $Quarter$, $Month$, $Week$, $Weekday$, Day, $Dayofyear$, and $Date$. We used these features to calculate the daily and monthly aggregation of product sales. Besides, we can use the features Unit Price and Quantity to estimate revenue.

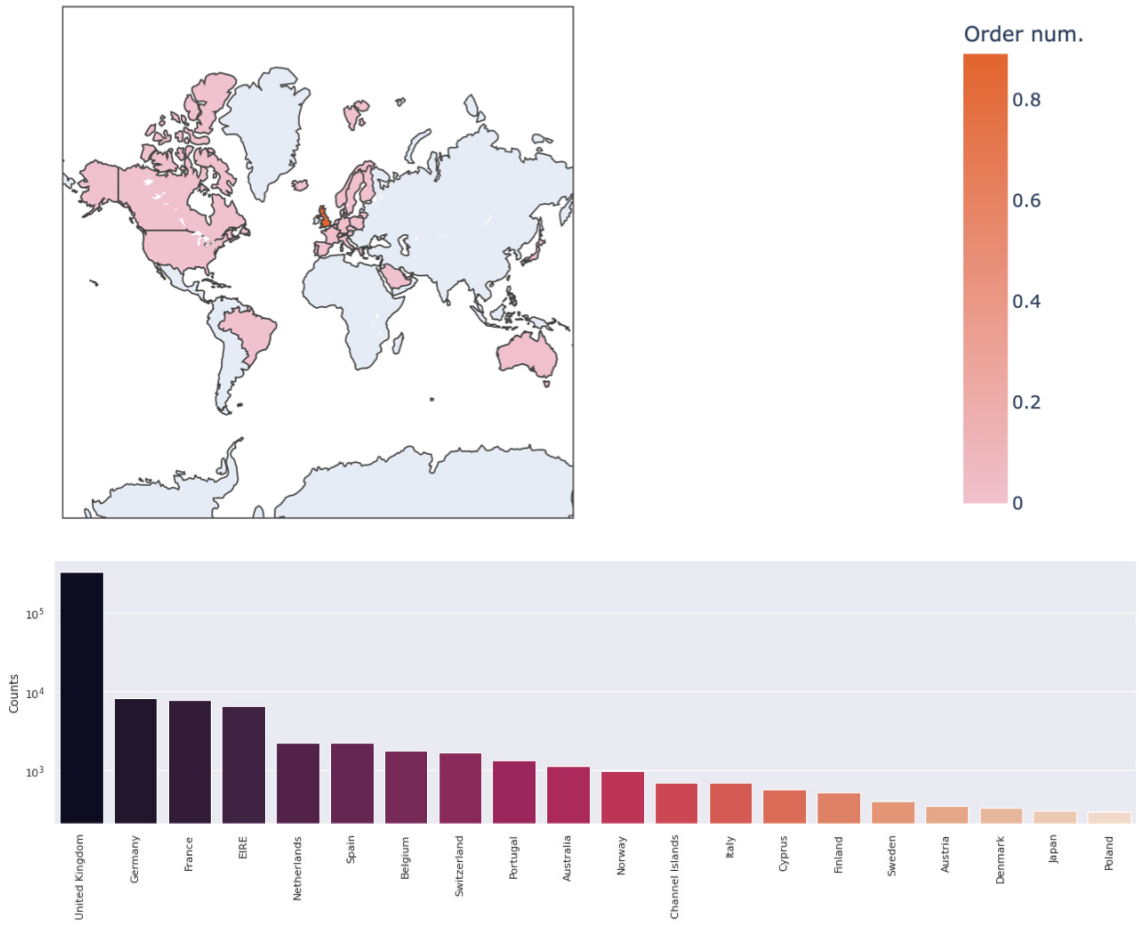

Figure 4.7: This figure depicts the most common countries in our data and the total number of orders per each. As can be seen, the UK is the largest in terms of the number of orders, followed by Germany, France, and some other European countries.

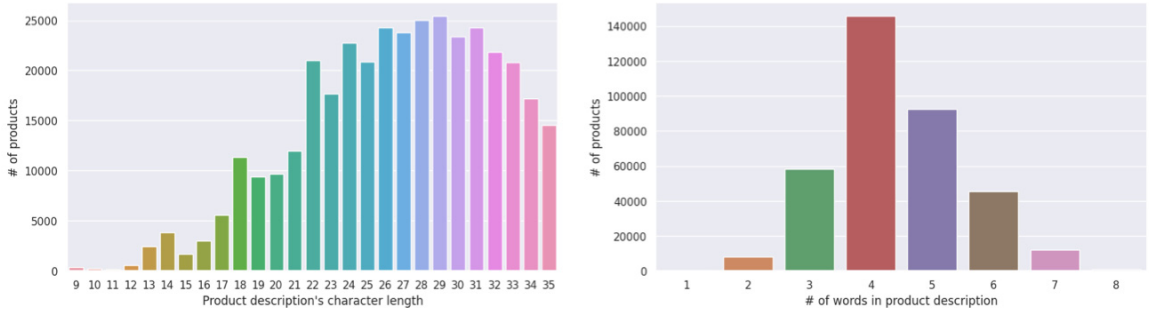

Figure 4.8: Distribution of product descriptions' length according to both the number of words and characters counts. The y-axis represents the number of products with that length. The left figure represents the character length of the descriptions, while the right figure indicates the number of words in each description. The figure highlights that most descriptions fall within the 3 to 6 word range, equivalent to 22 to 35 characters.

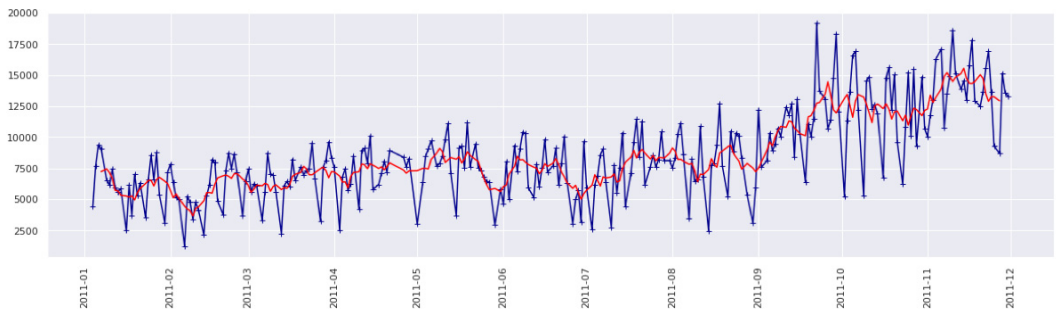

Figure 4.9: The number of daily sales over a year, from Jan 1 to Nov 30, 2011. The overall sales rise from September to pre-Christmas time. The X and Y axis shows the time period and the number of sales, respectively. The blue line shows how many products are sold daily, and the red line is the daily sales trend over the given time. We apply the moving average method and set the size of the rolling window to 7 days. We calculate the mean of each window.

4.4.1 Data Pre-processing

In this step, we perform data cleaning by removing redundant records and columns and addressing missing values, including 25 percent of customers and 0.26 percent of product descriptions. When the description is missing, the customer ID and unit price are also missing, and the price and quantities of records without customer ID may indicate outliers. Additionally, we filter out records with zero quantity or price, as well as cancelled orders. To handle outliers, we remove these occurrences from the data. As mentioned earlier in the training phase section, additional prepossessing has been applied to the product descriptions by cleaning the text using Python libraries such as NLTK and Texthero before applying language models and embeddings.

4.4.2 Exploratory Data Analysis

In this section, we present visualizations of the collected data to help understand the patterns and trends. The daily sales trend of the distributed sales data in 2011 is shown in Fig. 4.9. The blue pluses represent the number of daily sales, while the red line shows the overall sales trend during a particular period. As seen in the figure, there is a significant increase in sales during the pre-Christmas period. This is in line with the observation that important events during that period had a considerable impact on sales.

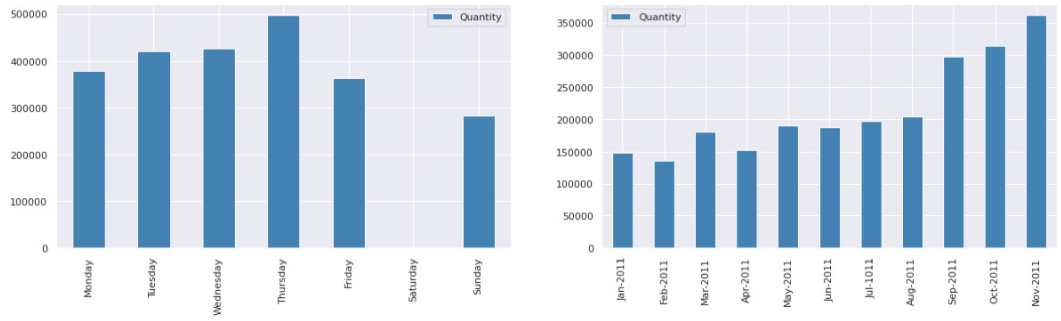

Figure 4.10: The figure on the left displays the total amount of sales per weekday across all the data, with no transactions recorded on Saturdays. The figure on the right shows the monthly sales data from January to December 2011. The X and Y axes represent the time period (weekdays on the left and months on the right plot), and sales quantity for all the data respectively.

Further insights into the sales data by displaying the total sales per weekday and month are provided in Fig. 4.10. The bar chart on the left shows the total sales quantity per day of the week, revealing that Thursday has the highest number of product sales, whereas the number of transactions declines on Friday and Sunday, and there are no transactions on Saturday. The bar chart on the right illustrates the monthly sales, which shows a peak in November that correlates with the pre-Christmas season which typically starts in September. February and April are the months with the lowest sales volume.

4.5 Experimental Study

In this section, we present the results of our experimental analysis and compare the performance of different forecasting models. We conducted two scenarios to evaluate the effectiveness of using product descriptions to improve sales forecasts for new products. In the first scenario, we compared the performance of various forecasting models using daily and monthly sales data, both with and without the incorporation of product descriptions and clustering algorithms. In the second scenario, we assessed the performance of various language models and dimensionality reduction techniques following the implementation of product descriptions and clusterings. To evaluate the models, we considered records of the last two months as new products, as the test set and the remaining data as the training set, with the test set comprising about 20 percent of the data. To assess

the feature importance of the forecasting models, we performed SHAP value analysis as an explainable AI tool [51]. Additionally, we used a statistical analysis known as a t-test which compares the mean values between two sample groups of data to determine if they are statistically different from each other [50, 32]. We conducted a t-test to compare the mean error values of two approaches: The traditional method that utilizes only time-series data for all products, and our proposed method that clusters products based on their descriptions and uses time-series data specific to each cluster for forecasting. The test yielded a t-value of 2.43 with 25 degrees of freedom (df) and a p-value of 0.022. Given the p-value of 0.022, we reject the null hypothesis that the means are equal and conclude that there is a statistically significant difference between the mean values of errors in the traditional approach and our approach, indicating that using product descriptions and clustering improved the performance of the forecasting models.

4.5.1 Evaluation Metrics

When comparing forecasting methods, evaluating the accuracy of the models is crucial. The forecast error is the difference between the predicted and actual values, and smaller errors indicate higher accuracy. Several metrics are available to measure forecast accuracy, and in this research, we utilized three widely used performance criteria: RMSE, WMAPE, Bias, and Standard Deviation (StDev). To provide a synthetic overview of the forecasting performance of each method, we computed the mean of error distribution for each forecasting method over the entire set of monthly data points in each cluster.

RMSE is the square root of the Mean Squared Error (MSE) [109]. It is a useful metric for calculating forecast accuracy, which considers the error between the target value y_n and the predicted value $\hat{y}_n$ in each sample. Squaring it, taking the mean, dividing by the total amount N of entries into the data, and then taking the root provides an idea of the error for individual predictions. The RMSE metric is heavily affected by outliers, and higher errors are obtained when some predictions are far from the targets. In our experiments, we used the LightGBM model after applying clustering and achieved the lowest RMSE of 3.01 for this forecast model, indicating that, on average, the forecast values were 3.01 units away from the actual values.

$$\text{RMSE} = \sqrt{\frac{1}{N}\Sigma_{n=1}^{N}(y_n - \hat{y}_n)^2} \tag{4.1}$$

N is the number of samples, y is a vector of the actual data, and $\hat{y}$ represents the forecast in the formula in Eq. 4.1.

WMAPE is a metric used to measure the prediction accuracy of a forecasting method [20]. Unlike MAPE and WAPE, WMAPE takes into account differences in the importance of products or moments in time by weighting errors by sales volume. WMAPE can be useful when trying to identify the most critical day to predict, for example. The formula for WMAPE is shown in Eq. 4.2:

$$\text{WMAPE} = \frac{\Sigma_{n=1}^{N}|y_n - \hat{y}_n|}{\Sigma_{n=1}^{N}|y_n|} \tag{4.2}$$

In this formula, weights can be assigned to important factors, such as specific days of the week, to reflect their relative importance. Like RMSE, the LightGBM model had the lowest WMAPE compared to the other forecasting models in our experiments.

Bias is a measure of systematic error in the forecast and occurs when there is a consistent difference between actual sales and the forecast [94]. If the bias value is positive, it means that the forecast is consistently higher than the actual values. Conversely, if the bias value is negative, it means that the forecast consistently underestimates the actual values. When the bias value is zero, it implies that the forecast matches the actual values on average, although there may still be random errors present. The formula for calculating Bias is shown in Eq. 4.3:

$$Bias = \frac{1}{N}\Sigma_{n=1}^{N}(y_n - \hat{y}_n) \tag{4.3}$$

The StDev is a widely used statistical tool that is used to measure the variability of historical sales data around the mean. In the formula presented in Eq. 4.4, n is the time period (which is the months in our experiments), y_i is the sales data for the i-th time period, and $\bar{y}$ is the mean sales over all time periods.

$$StDev = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4.4}$$

The resulting value of StDev indicates how much the actual sales deviate from the average, which can be useful in determining the confidence level of sales forecasts or identifying potential outliers.

4.5.2 Results And Discussion

The evaluation results obtained from various time-series prediction algorithms, and product description processing including multiple clustering algorithms, dimensionality reduction techniques, and language models are presented in Table 4.2. Based on the findings, there are several important aspects to discuss. Firstly, the performance of each forecasting method is compared (#6 in Fig. 4.2). LightGBM+Optuna is the best-performing method for this time-series forecasting problem, using different variations and hyperparameters based on our experimental results in Table 4.2. This suggests that the performance of various forecasting models depends on fine-tuning their parameters. Since Catboost and Prophet consistently performed the poorest, we have omitted them from Table 4.2 and have provided details of their results in Section 4.6. Secondly, the results of different language models used to embed product descriptions are compared (#2 in Fig. 4.2). As shown in the table, RoBERTa has the best performance among other language models. Thirdly, the role of different clustering algorithms in the final results is explored (#3 in Fig. 4.2). Among the two clustering algorithms applied, Kmeans obtained superior results compared to HDBSCAN. By setting K=10 as the number of clusters, we achieved fewer errors. Fourthly, the effect of dimensionality reduction techniques on top of clustering algorithms is studied. The difference between UMAP and T-SNE is not significant in all evaluation measures, indicating that neither of the dimensionality reduction techniques is necessarily better than the other in our experiments, as indicated by the results in Table 4.2. However, T-SNE slightly outperformed UMAP.

Last but not least, the effect of using product descriptions and clustering on the prediction outcomes is examined. The forecasting results only based on time-series data without considering any product descriptions, labelled as *Noproductdescriptions*, are presented in the left-hand columns in Table 4.3. The corresponding results from applying product descriptions and clustering are

displayed in the right-hand columns for comparison. Interestingly, the results indicate that considering product descriptions and clustering leads to lower prediction errors in the forecasting approaches. Clustering products (as shown in #3 in Fig. 4.2) and assigning new products to clusters based on their similarity in descriptions reduces the prediction errors of new products. We obtained the best outcomes by incorporating product descriptions, which are highlighted in bold font. These outcomes were achieved using a combination of the LightGBM model, RoBERTa (LM1), Kmeans, and T-SNE from Table 4.2. To evaluate the effectiveness of our approach compared to traditional methods (labelled as No product descriptions), we calculated the mean values of RMSE, WMAPE, Bias, and StDev across all monthly data points over a period of 12 months and 50 trials. The standard deviation for each metric is shown within the brackets, with a lower value indicating better performance. Overall results, shown in Table 4.3, confirm our claim about the advantages of using product descriptions and clustering to improve sales forecasting models.

We utilized SHAP [4] value analysis for the LightGBM model, as it outperformed other forecasting models, based on the results obtained from Table 4.2. We calculated the SHAP values for each instance of the dataset to analyze the importance of each feature of the forecasting model. The SHAP values indicate how each feature affects the prediction, as illustrated in Fig. 4.11. In this concept, the SHAP value for each feature is shown by the length of the bar. For instance, $StockCode$ is the feature that contributed the most, along with $Weekday$ and $Month$. Day, on the other hand, had the lowest contribution to the prediction, according to Fig. 4.11. Lower values at the $Weekday$ feature (0 to 3) correspond to Monday, Tuesday, Wednesday, and Thursday, which are the days with the highest number of product sales, leading to high quantity target values. These values are shown in blue in the figure and move towards higher SHAP values and thus higher predicted quantity values. In contrast, Friday, Saturday, and Sunday are shown in red and move toward lower forecast values, which aligns with our results based on the discovery of the weekdays and the sum of the daily quantities in Fig. 4.10, demonstrating that these days only made a few sales. As mentioned earlier, we have

[4]It should be noted that SHAP value analysis is a widely used and effective method for interpreting the predictions of machine learning models.

over 4,000 unique products and descriptions that are important but highly complex features in our data. In this figure, the *StockCode* is shown as an important feature, and since it is not numerical, it has no colour or low or high values.

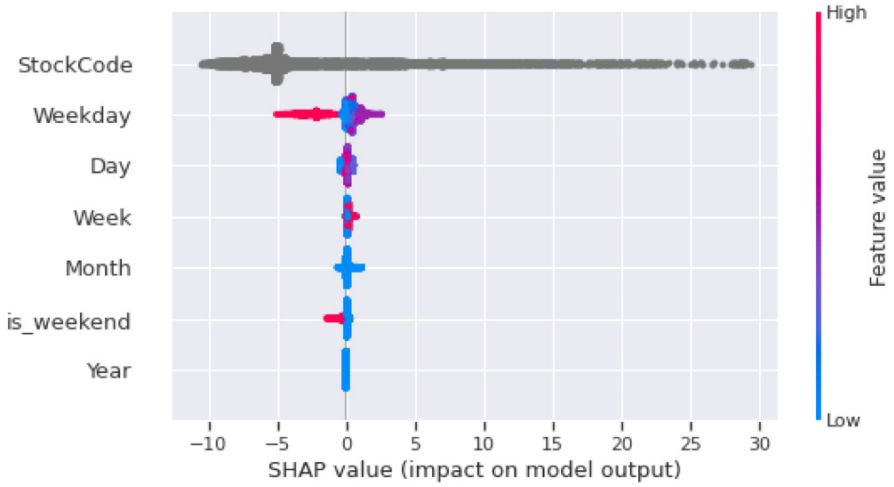

Figure 4.11: SHAP values provide a way to understand the contribution and importance of each feature in a prediction, by assigning a value to each feature that represents its impact on the model's output. The SHAP values related to individual features are presented to interpret the results provided by the LightGBM model. The higher the value of a feature, the more important the feature is, and the more it impacts the results. *Weekday* as an example in this figure depicts the importance of days of the week in the sales results. Friday to Sunday are shown in red, which are the days with the lowest quantity of sales. On the other hand, the blue area shows values corresponding to Monday, Tuesday, Wednesday, and Thursday, which are related to the days with the high number of product sales. As the *StockCode* is not a numerical feature, it is not shown in colours in the figure.

4.6 Performance of Other Applied Forecasting Models

In this section, we provide additional tables related to the detailed results presented in the section Experimental Study. The results from the forecasting models (Catboost and Prophet) that performed the least accurately are presented in Table 4.4. We have presented their respective forecasting results in Table 4.5. The left-hand columns display the forecasting results that are solely based on time-series data without considering any product descriptions, which we have labelled as *Noproductdescriptions*. We have also included the corresponding results obtained

Methods		LightGBM				ARIMA			
		TF-IDF	LM1	LM2	LM3	TF-IDF	LM1	LM2	LM3
Kmeans + T-SNE	RMSE	5.78	**3.01** (±**0.01**)	4.20	5.06	8.06	**5.02** (±**0.05**)	6.61	7.50
	WMAPE	6.27	**4.80** (±**0.02**)	5.02	6.77	9.52	**5.64** (±**0.07**)	7.01	8.98
	Bias	-0.90	**0.02** (±**0.04**)	0.06	0.08	1.03	**0.05** (±**0.09**)	-0.09	-1.10
	StDev	±0.10	±**0.01**	±0.03	±0.04	±0.12	±**0.10**	±0.14	±0.16
Kmeans + UMAP	RMSE	6.83	3.65	4.31	5.02	8.73	5.46	6.70	7.69
	WMAPE	7.01	5.44	5.88	6.85	9.71	6.02	7.32	9.01
	Bias	-0.84	0.03	0.10	0.14	1.12	0.72	0.13	-1.02
	StDev	±0.23	±0.02	±0.05	±0.06	±0.18	±0.12	±0.17	±0.20
HDBSCAN + T-SNE	RMSE	6.90	3.49	4.62	5.34	8.90	5.50	6.74	7.78
	WMAPE	6.77	5.63	5.90	6.82	9.85	6.47	7.56	9.31
	Bias	-1.30	0.05	0.14	0.23	-1.42	1.01	0.21	1.05
	StDev	±0.25	±0.03	±0.07	±0.08	±0.19	±0.16	±0.21	±0.25
HDBSCAN + UMAP	RMSE	7.02	3.67	4.81	6.03	9.02	5.66	6.79	7.93
	WMAPE	7.03	5.82	6.13	6.95	9.89	6.50	7.66	9.35
	Bias	-0.89	-0.06	-0.12	-0.25	1.46	1.03	-0.33	1.12
	StDev	±0.29	±0.07	±0.09	±0.11	±0.23	±0.26	±0.28	±0.29

Table 4.2: Results obtained by utilizing product descriptions as an additional feature. Comparing the results of different combinations of time-series prediction algorithm and product description processing (including the clustering algorithm, dimensionality reduction technique and language model), using different evaluation metrics. The evaluation metrics for assessing the accuracy of monthly sales forecasting are computed for each cluster using the mean of RMSE, WMAPE, Bias and StDev across 50 trials over the entire monthly period of data points in that cluster, during 12 months. A positive bias overestimates actual values, a negative bias underestimates them, and a bias of zero indicates an equal average forecast but possible random errors. A smaller StDev(Eq. 4.4) is generally preferred as it suggests that the sales data is more consistent and predictable, which is beneficial for making accurate predictions. The time-series prediction algorithms are LightGBM and ARIMA (as shown in #6 in Fig. 4.2). The clustering algorithms (#3 in Fig. 4.2) are Kmeans (with K=10) and HDBSCAN (with min_cluster_size= to 50 and min_samples=10). For dimensionality reduction, we used UMAP and T-SNE. The language models for embedding the product descriptions (#2 in Fig. 4.2) are RoBERTa (LM1), DistilBERT (LM2) and SimCSE (LM3). The numbers shown in bold font are the best results obtained for each evaluation metric across combinations. We observed that the optimal pairing involves using LightGBM as the preferred forecasting model, Roberta as the most effective language model, and Kmeans as the superior clustering algorithm, with T-SNE as the most suitable dimensionality reduction technique. As a result, we provide the standard deviation for each evaluation metric in brackets for the top-performing combinations.

Methods	No product descriptions		With product descriptions	
	LightGBM	ARIMA	LightGBM	ARIMA
RMSE	8.23 (±0.51)	8.90 (±0.67)	**3.01** (±**0.01**)	5.02(±0.05)
WMAPE	7.05 (±0.46)	8.11 (±0.58)	**4.80** (±**0.02**)	5.64(±0.07)
Bias	-1.09 (±0.70)	-1.30 (±0.73)	**0.02** (±**0.04**)	0.05 (±0.09)
StDev	±0.44	±0.59	±**0.01**	±0.10

Table 4.3: The outcomes of the prediction models with and without using product descriptions are compared. The results under the *Noproductdescriptions* columns are based on solely time-series data (same approach shown in Fig. 4.1), without incorporating any product descriptions. The columns on the right, under *Withproductdescriptions*, display the results obtained by applying product descriptions and clustering techniques for prediction. The optimal outcomes, achieved through the utilization of product descriptions, are highlighted in bold font. These results are associated with the combination of the LightGBM model, along with Roberta (LM1), Kmeans, and T-SNE from Table 4.2. To evaluate the efficacy of our approach in comparison to traditional methods, we computed the mean values of RMSE, WMAPE, Bias, and StDev across all monthly data points for a period of 12 months, in 50 different trials. The standard deviation for each metric is presented within the brackets, with a lower value indicating better performance. These outcomes justify the privilege of utilizing product descriptions to significantly improve the accuracy of sales forecasts.

from applying product descriptions and clustering in the right-hand columns for comparison.

4.7 Conclusion

Sales forecasting is important for business financial planning. In this study, we used real sales data from a retail company and compared the performance of different forecasting models, including LightGBM, Catboost, ARIMA, and Facebook Prophet. We encoded the product descriptions using various language models such as RoBERTa, DistilBERT, and SimCSE and clustered them based on similarity using Kmeans or HDBSCAN algorithms with UMAP and T-SNE for dimensionality reduction. We then applied the forecasting models to each cluster to predict the sales of new products. The results showed that the RoBERTa language model and LightGBM forecasting model had the best performance, with Kmeans (K=10) and T-SNE providing the best clustering results. We also found that using the Optuna optimizer improved the performance of the LightGBM model. Overall, clustering

Methods		Catboost				Prophet			
		TF-IDF	LM1	LM2	LM3	TF-IDF	LM1	LM2	LM3
Kmeans + T-SNE	RMSE	13.60	12.27	12.95	13.56	7.34	**5.77** (±**0.44**)	6.20	7.01
	WMAPE	13.97	12.11	13.02	13.78	8.85	**5.42** (±**0.31**)	6.86	7.59
	Bias	-1.47	0.81	0.55	1.42	0.93	**0.13** (±**0.53**)	0.23	-0.45
	StDev	±0.53	±0.51	±0.53	±0.58	±0.32	±**0.30**	±0.34	±0.36
Kmeans + UMAP	RMSE	13.71	**12.10** (±**0.59**)	13.03	13.69	7.55	5.81	6.33	7.12
	WMAPE	14.02	**11.60** (±**0.48**)	13.45	13.99	8.98	5.60	6.91	7.73
	Bias	1.50	**0.36** (±**0.50**)	0.98	1.50	0.99	0.18	-0.47	-0.52
	StDev	±0.55	±**0.47**	±0.56	±0.58	±0.35	±0.33	±0.36	±0.39
HDBSCAN + T-SNE	RMSE	13.75	12.34	13.36	13.72	7.67	5.91	6.44	7.34
	WMAPE	14.34	12.29	13.48	14.02	9.01	5.78	6.94	7.16
	Bias	1.56	0.92	1.01	1.57	-1.0	0.24	-0.56	0.67
	StDev	±0.57	±0.58	±0.61	±0.63	±0.36	±0.39	±0.41	±0.45
HDBSCAN + UMAP	RMSE	13.83	12.55	13.40	13.81	7.81	6.04	6.91	7.55
	WMAPE	14.40	12.30	13.52	14.37	9.12	5.95	7.05	7.81
	Bias	-1.62	-1.05	0.99	-1.62	-0.96	0.31	-0.66	-0.71
	StDev	±0.69	±0.59	±0.60	±0.62	±0.47	±0.43	±0.45	±0.46

Table 4.4: The analysis of using product descriptions as an additional feature for sales forecasting. We compared the performance of Catboost and Prophet as time-series prediction algorithms, using the same product description processing (clustering algorithms, language models, and dimensionality reduction techniques) as presented in Table 4.2. With the same evaluation metrics and model parameters. The bold numbers in the table indicate the best results achieved for each evaluation metric across all combinations.

Methods	No product descriptions		With product descriptions	
	Catboost	Prophet	Catboost	Prophet
RMSE	15.20 (±0.91)	9.33 (±0.79)	12.10 (±0.59)	**5.77** (±**0.44**)
WMAPE	14.70 (±0.88)	8.58 (±0.56)	11.60 (±0.48)	**5.42** (±**0.31**)
Bias	1.88 (±0.76)	-1.24 (±0.69)	0.36 (±0.50)	**0.13** (±**0.53**)
StDev	±0.73	±0.68	±0.47	±**0.30**

Table 4.5: Comparing the prediction results of Catboost and Prophet, with and without considering any product descriptions. The evaluation metrics and parameters are the same as Table 4.3. These results demonstrate the value of utilizing product descriptions to greatly enhance the accuracy of sales forecasts.

the product descriptions and assigning new products to clusters significantly improved the accuracy of the sales forecasting results.

Chapter 5

AI-Powered Résumé-Job Matching: A Document Ranking Approach Using Deep Neural Networks

5.1 Introduction

Semantic Text Similarity (STS) estimation is a crucial task in NLP applications, including document classification, text summarization, and question answering[1]. The primary objective of STS is to determine the degree of similarity between two texts. In recent decades, there has been significant progress in the development of NLP methods aimed at automatically scoring the similarity between pairs of documents. DNN [55] have played a crucial role in automatically scoring the similarity between pairs of documents, enabling more accurate and effective evaluation of document similarities. Although individual network components or architectures such as CNN [56], LSTM [87], attention [105], and Siamese neural network (SNN) [112] have demonstrated their effectiveness in representing documents and measuring similarity when utilized independently, the potential of combining all these components into a unified network remains relatively unexplored.

Consequently, we propose a novel SNN architecture that combines LSTM, CNN, and multi-head attention layers to measure document similarity. Our primary objective is to assess the similarity between resumes and job descriptions, and subsequently rank them accordingly. In our approach, we leverage the power of both a CNN and an LSTM to extract features from the document. We utilize CNN to extract local features, while simultaneously employing LSTM to capture global features. By employing these parallel components, we can effectively capture both local and global information within the document. Then, we concatenate these

[1]This chapter is the extended version of the conference paper presented at DocEng 2023. https://dl.acm.org/doi/10.1145/3573128.3609347. And a workshop paper presented in SciNLP 2021. https://scinlp.org/#accepted-abstracts.

extracted features to create a unified representation of the document. By combining both types of features, we capture both the finer details captured by the local features and the broader context captured by the global features, resulting in a comprehensive representation of the document. Ultimately, we apply a multi-head attention layer, that allows token representations to be influenced by the most relevant tokens within a document, providing a more fine-grained and context-aware representation. By considering the relationships and dependencies between tokens, attention allows for a more contextually aware representation of the input.
We evaluate our approach by comparing it against standard methods, including Siamese CNN (S-CNNs), Siamese LSTM with Manhattan distance, and a BERT-based sentence transformer model. We put forward the hypothesis that our proposed model outperforms the current state-of-the-art DNN in accurately measuring document similarity.
Below, we highlight the primary contributions and outline the corresponding research gaps.

1. Propose a novel SNN architecture for matching resumes and job descriptions:

    - Research gap: Previous studies employing neural networks for resume and job description ranking have been limited in their ability to comprehensively capture the nuances of both local and global features.
    - Contribution: By integrating LSTM, CNN, and multi-head attention layers, we improved the accuracy of the task involving the ranking of resumes and job descriptions.

5.2 Related Work

In this section, we provide a concise overview of the relevant literature that explores the measurement of document similarity using DNN. The traditional approaches for measuring document similarity often rely on representing documents using word frequency vectors, commonly referred to as the BOW representation. The similarity between these vectors can be computed using different techniques, such as cosine similarity and Euclidean distance. A search engine specifically designed for

retrieving similar documents, named SimSeerx, is introduced by Williams in [108]. It accepts complete documents as queries and provides a ranked list of documents that exhibit the highest similarity to the query document. The ranking is determined using cosine similarity, which serves as the general ranking function in SimSeerx. Word embeddings have been introduced as a solution to overcome the limitation of traditional approaches [63].

There are multiple approaches to capturing the similarity of sentences/documents, such as summing or averaging the word embeddings within a sentence or document. However, this approach does not take into account crucial factors such as word order and syntax, which play a significant role in determining the meaning and structure of the text. As a result, researchers have developed various DNN approaches that address this limitation by leveraging word embeddings while also considering the sequential nature of language. A notable example of a DNN architecture that maintains the sequential order of words in a sentence is the RNN [31] such as LSTM. On the other hand, CNN employs convolution filters (CFs) to capture informative local features from a document. In contrast, contemporary transformer models such as sentence-BERT [82] utilize feedforward architectures. These models incorporate positional embeddings and multiple self-attention layers, facilitating the development of document representations that are sensitive to word order. A comparison between CNN and LSTM across a diverse range of NLP tasks has been conducted by Yin in [113]. Their claim suggests that there is a lack of consensus when it comes to selecting a DNN for a particular NLP problem. According to the findings of their experiments, CNN and RNN offer distinct and complementary information.

An SNN can be constructed using any of the aforementioned types of neural networks or different combinations of these DNN. SNN is designed to compare and measure the similarity or dissimilarity between pairs of inputs and consists of two or more identical sub-networks. It encodes the input into a fixed-length embedding. These embeddings are then compared using a similarity metric or distance measure to determine the similarity between the inputs. SNN have been successfully applied to various tasks such as text similarity [25], face recognition [95], and signature verification [28]. SNN offers advantages such as efficient inference, scalability to

large datasets, and the ability to handle imbalanced data. Various research studies have proposed diverse architectures for SNN. A Siamese structure incorporating bidirectional LSTM and a CNN with a weighted-pooling attention layer is introduced in [44]. Their architecture enables the extraction of an attention vector and improves information extraction and learning capabilities. A novel Siamese Attention-augmented Recurrent Convolutional Neural Network (S-ARCNN) that combines bidirectional LSTM, convolution, pooling, and attention layers is presented by Han in [39]. They evaluated the performance of their model by applying it to rank document similarity in the Quora Question Pairs dataset.

5.3 Methodology

Previous studies have demonstrated the strong capability of LSTM, CNN, and Attention-based architectures in capturing enriched patterns of semantic representation for entire sentences. In our research, we proposed a model that significantly enhances the capacity for information extraction and learning by combining LSTM, CNN, and multi-head attention architectures. Below, we provide a summary of the DNN and transformer-based language models that we employed in our study.

5.3.1 DNN

In this section, a concise overview of CNN and LSTM is provided.

CNN

The first encoding strategy is CNN which excels at capturing local patterns through the use of convolutional, pooling, and fully connected layers. Convolutional layers apply filters to extract relevant features, while pooling layers reduce spatial dimensions, improving computational efficiency and translational invariance. The input consists of the text-based information found in a resume or job description, which is represented as a sequential arrangement of words. The process involves mapping the words to word vectors using an embedding matrix to generate a document matrix.

LSTM

The alternative encoding approach is the LSTM model. LSTM is a popular type of RNN that addresses the vanishing gradient problem. It uses memory cells and gating mechanisms to capture long-term dependencies in sequential data. LSTM have shown excellent performance in various NLP tasks due to their ability to retain and forget information from previous time steps. In our model, we employ an LSTM with 256 units (128 units in each direction) to attain context-aware word embeddings.

Attention

A multi-head attention layer allows the model to attend to different parts of the input simultaneously and learn diverse representations. Each head in the multi-head attention mechanism attends to different parts of the input and produces a separate attention output. These outputs are then concatenated or combined to obtain the final representation. By incorporating a multi-head attention layer, the model can capture different types of dependencies and relationships within the input data, leading to improved performance in tasks such as document classification, machine translation, or summarization.

5.3.2 Transformer-Based Models

This section briefly summarizes the NLM we utilized as document representation techniques. All of the subsequent models that we utilized are constructed based on the BERT [26] framework with the aim of enhancing its performance.

SBERT

Unlike traditional BERT models, Sentence-BERT [82] considers entire sentences as input and encodes them into fixed-length vectors. SBERT captures fine-grained sentence representations and enables efficient similarity calculations. In this study, we employ the method of averaging the vectors of each sentence to generate a comprehensive vector representation for the entire Resume or Job description. For our configuration, we set the max_seq_length to 100 and do_lower_case to False.

BigBird

To handle longer sequences, BigBird [114] introduces a novel global attention mechanism called *SparseAttention*. It uses a combination of global and local attention patterns, reducing the computational complexity for long sequences. This makes it highly suitable for handling long documents or sequences, such as Resumes or Job descriptions in our specific scenario. We configured the num_attention_heads to be 16 and set the intermediate_size to 3072.

RoBERTa

By employing larger training data, longer training duration, and more diverse pre-training objectives, RoBERTa [59] improves BERT. We employed the RoBERTa-base-v2 model from the sentence transformers library developed by Hugging Face. This library offers an implementation of RoBERTa, taking care of pre-processing, tokenization, and necessary conversions to ensure proper formatting of input and output. We used the same configurations as those used for SBERT. The output vectors generated by RoBERTa and SBERT are dense vector spaces with 768 dimensions, whereas BigBird produces output vectors with a dimensionality of 4096.

5.3.3 Proposed Model

In this section, we present our proposed approach that combines the strengths of the methods discussed in Section 5.3.1 and 5.3.2. Initially, a Siamese structure incorporating an LSTM and a CNN is utilized to generate a representation for the document. Following that, an attention vector is derived by applying a multi-head attention layer. The model takes in two input sequences, representing the pair of resumes/job descriptions to compare for similarity. Transform input sequence into a dense vector representation using pre-trained document embeddings such as SBERT, RoBERTa, and BigBird. LSTM component processes the embeddings of each input sequence, capturing contextual information and sequential dependencies. CNN component applies convolutional filters to the embeddings, extracting local patterns and features. Then, concatenate the outputs of the LSTM and CNN

components to combine their respective representations. In the next step, we apply a multi-head attention mechanism to the concatenated representation. This layer attends to different parts of the concatenated features, allowing the model to focus on relevant information and capture dependencies across different elements. Then, we aggregate the output of the multi-head attention layer, using max pooling, to summarize the attended information. In the next step, we pass the pooled representation through a fully connected dense layer, to process the combined features. Finally, a similarity score is generated using the cosine measure, which serves as an indicator of the degree of similarity or dissimilarity between the Resume and the job description. An overview of the proposed model, showcasing its key components and their interactions is provided in Fig. 5.1.

5.4 Dataset

Data preparation plays a vital role in creating an effective tool, as it involves not only formatting the data for the application but also gaining a comprehensive understanding of the underlying application problem. Therefore, we invested significant time and effort into pre-processing and eliminating irrelevant information from the dataset which was obtained from a Canadian recruitment company specializing in electronic recruitment, comprising a collection of resumes and authentic job descriptions in JSON format. The JSON files consist of various sections, including resumes, job descriptions, and stages. Resumes are obtained from applicants' profiles, while job descriptions are provided by employers. The stage section indicates the status of each resume and job description, such as whether they have been hired or rejected. To create a unified record, we extracted the relevant information from each file and combined them. This record includes the job description, key resume field information, and the corresponding stage label. The dataset encompasses 12 distinct industry sectors and comprises 4,198 job descriptions and 268,549 resumes.

5.4.1 Data Pre-processing

To process our data, which contained both semi-structured and textual fields, our initial task was to extract relevant information. The resumes consisted of itemized

lists, such as education, experience, and certificates, along with text fields like cover letters. Similarly, the job descriptions were represented as textual fields. We consolidated the extracted text from each resume into a single field. We assigned a value of *zero* for rejected resumes and *one* for hired resumes in the stage field. This resulted in a CSV file with three columns, where each row represented the content of an applicant's resume, the corresponding job description, and the stage. The job descriptions were repeated for each resume under the same job advertisement. Subsequently, we executed data cleaning and normalization techniques on our dataset utilizing well-known Python libraries like SpaCy and Gensim. The process comprised multiple steps, including the removal of punctuation, stop words, and HTML tags from the plain text. Moreover, we converted all letters to lowercase and filtered out non-English words to concentrate solely on pertinent content. Table 5.1 shows the data distribution across different industry sectors.

Industry sector	# of resumes	# of job ads
Energy&Utilities	76,748	1,200
Government&Military	53,840	487
Other&Not_Classified	35,217	810
Construction	31,249	663
Residential&Commercial	25,889	436
Insurance	14,200	298
Nonprofit_Charitable_Orgs	11,228	233
Computer&IT_Services	2,804	58
Engineering_Services	183	2
$Real_Estate\&Property\_Management$	154	1
Advertising&PR_Services	16,955	7
Agriculture, _Forestry&Fishing	82	3
Total	268,549	4,198

Table 5.1: Distribution of data among various industry sectors.

5.5 Evaluation And Experiments

Subsequent sections of this chapter will showcase experiments conducted to assess the effectiveness of our proposed model in the task of document query ranking. We evaluated our model alongside Siamese CNN (S-CNNs), Siamese LSTM using Manhattan distance, and a BERT-based sentence transformer model.

5.5.1 Results

We employed a 10-fold cross-validation methodology to train and test three DNN: S-CNNs, S-LSTM, and the BERT-based transformer model, in addition to our proposed model. Initially, we partitioned the dataset into ten stratified folds. Subsequently, we trained a model in nine of these folds and evaluated it in the remaining fold. This process was repeated ten times, allowing us to compute average performance metrics, including accuracy, precision, recall, and F1 score. When it comes to choosing NLM for encoding input documents, the most favourable outcomes were achieved by utilizing BigBird. Therefore, for all models we used pre-trained BigBird-large and in the results table, we exclusively present the outcomes achieved with this transformer-based model. The maximum sequence length was set to 4096 tokens, such that shorter and longer questions were padded or trimmed, respectively. We designated 20 percent of the job-resume dataset as our test set and assessed the accuracy of various models to establish a comparative benchmark against our proposed model. The outcomes are illustrated in Table 5.2, showcasing the obtained results. As evident in Table 5.2, our model outperforms the other three deep learning models in terms of performance.

Models	F1	Precision	Recall	Accuracy
S-CNNs	73.51	73.10	73.94	74.67
S-LSTM	70.92	70.83	71.03	71.48
BERT-based	72.43	72.06	72.81	73.51
Proposed model	**82.90**	**82.77**	**83.05**	**84.16**

Table 5.2: Model comparison for resume-job matching. As demonstrated in the table, our proposed method outperformed other models, yielding superior results.

5.6 Conclusions

The current leading method for automatically ranking document queries involves utilizing DNN with attention, recurrence, convolution, and Siamese networks. While these components are believed to have complementary roles, their integration within a single architecture remains relatively unexplored. To address this gap, we have introduced a unique SNN that combines these components and conducted an

evaluation using a real-world dataset consisting of resumes and job descriptions. By integrating LSTM and CNN structures, our model effectively captures comprehensive contextual information. These two layers are applied simultaneously and merged together. The resulting output is subsequently passed through a multi-head attention layer, further enhancing the model's performance. To obtain a lower-dimensional representation of our input data, resumes, and job descriptions, we utilize pre-trained embeddings such as SBERT, BigBird, and RoBERTa. Through a comparative analysis, we evaluated our model alongside S-CNNs, S-LSTM with Manhattan distance, and a BERT-based sentence transformer model. Our experimental findings clearly indicate that our model surpasses the performance of the other models in terms of various evaluation metrics.

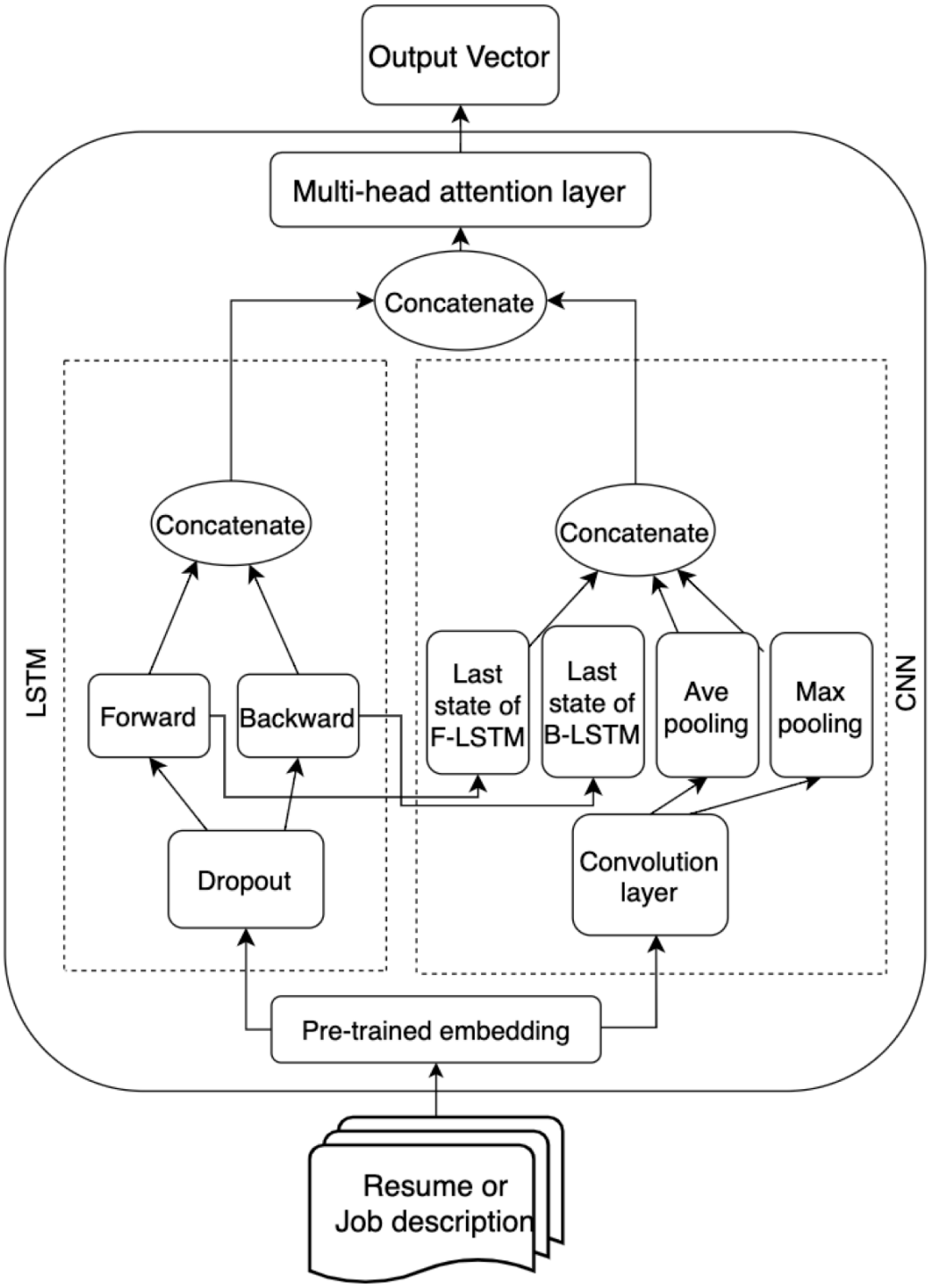

Figure 5.1: The overall view of the proposed SNN architecture, showing the combination of LSTM, CNN, and Attention layers. The input to our system can be either a resume or a job description, and our goal is to generate a vector representation of the input document. This vector representation is then passed to the ranking function, which calculates the similarity score between the input documents.

Chapter 6

Conclusion

Deep language models have emerged as indispensable tools for various machine-learning tasks. This thesis highlights the significant potential of document representation using deep language models in various machine-learning applications, specifically document clustering and ranking. By leveraging deep language models, we have demonstrated their effectiveness in enhancing interactive document clustering, allowing for efficient organization and navigation of large text datasets. Additionally, the application of diverse deep language models has improved traditional document ranking algorithms, resulting in more accurate and efficient search results. Moreover, deep language models have proven valuable in sales forecasting by analyzing historical data and market trends, enabling businesses to make informed decisions and optimize their operations. The versatility of language models, facilitated by their ability to learn from vast amounts of text data, allows them to extract meaningful patterns and insights. By employing state-of-the-art models such as SBERT, RoBERTa, Universal Sentence Encoder, Infer-Sent, and BigBird, this research has provided valuable insights into the potential and impact of language models in these specific domains. Future research can further explore the capabilities and limitations of language models in other machine-learning applications, fostering advancements in this rapidly evolving field. In conclusion, deep language models have transformed document ranking and clustering by providing deeper understanding, improved predictions, and efficient data organization. As we embrace these advancements, we must also strive to mitigate potential risks and promote responsible AI practices to fully leverage the benefits of language models in shaping a more intelligent and inclusive future. As language models continue to evolve, it is crucial to address challenges such as model biases, ethical considerations, and the responsible use of these powerful tools. The development of robust evaluation metrics, ongoing research, and collaborations

between academia, industry, and policymakers is essential to harness the full potential of language models while ensuring fairness, transparency, and accountability.

6.1 Future Research

The next steps involve further research and exploration in several areas. First, apply various deep language models to each task and perform a comprehensive comparison of the results. By systematically evaluating the performance of different models, we can gain insights into their strengths, weaknesses, and suitability for specific tasks. This analysis will enable us to make informed decisions regarding the selection and utilization of deep language models in future applications. Second, employ diverse use cases and datasets in the evaluation process. By considering a wide range of real-world scenarios and datasets, we can assess the generalizability and robustness of deep language models across different domains and data characteristics. This approach will provide a more comprehensive understanding of the models' performance and enable us to identify potential limitations and areas for improvement. Third, there is a need for continuous model advancements and developments to keep pace with the evolving nature of text data and emerging challenges in the field of text mining. Fourth, the emergence of new generations of LLM, such as GPT-4 or subsequent iterations, and the continuous developments in ChatGPT technology have significant implications for the projects mentioned earlier. Below, we list a few of them.

- No traditional training: With the emergence of conversational LLM such as ChatGPT, the need for traditional machine learning model training methods is significantly reduced. These models, pre-trained on extensive datasets, possess the capacity to adapt and generate context-aware responses without the extensive manual training typically required in traditional machine learning approaches. For instance, in clustering tasks, topic extractions from documents or product descriptions can be accomplished by instructing ChatGPT with relevant prompts.

- Improved performance: Newer generations of LLM often come with

enhancements in terms of language understanding, context handling, and the ability to generate more coherent and contextually relevant responses. This means that projects utilizing these advanced models may experience improved performance in tasks like document clustering and ranking. Furthermore, it's likely that we won't require conventional training methods for feature extraction using these technologies.

- Enhanced customization: As language models evolve, they tend to offer more flexibility and options for fine-tuning. This allows developers to customize the models more effectively to suit the specific requirements of their projects. For example, in a document ranking project, it becomes easier to fine-tune the model for the purpose of retrieving relevant information from documents that match the query.

- Reduced data requirements: Advanced models can often perform better with smaller amounts of training data. This is particularly valuable for projects with limited access to large datasets. The ability to achieve good results with less data can save time and resources.

- Ethical considerations: As language models become more powerful, there is an increased focus on ethical and responsible AI. Developers and project managers need to consider issues related to bias, fairness, and the responsible use of AI in their projects. This includes addressing potential biases present in training data and ensuring that AI systems are used in ways that align with ethical principles.

- Challenges in integration: While newer LLM offer significant advantages, they may also introduce challenges in terms of integration into existing systems or workflows. Compatibility issues, deployment complexities, and the need for robust infrastructure can be factors to consider.

- Adaptation and learning: Language models like ChatGPT are designed to adapt and learn from user interactions. This means that projects using these models may benefit from ongoing improvements in performance as the models accumulate more data and user interactions.

- Community and support: As the field of AI and LLM advances, a growing community of developers and researchers contributes to knowledge sharing, best practices, and support. This can be a valuable resource for project teams looking to leverage the latest advancements effectively.

In summary, the advent of new LLM generations and developments in ChatGPT have the potential to significantly impact the mentioned projects by offering improved performance, customization options, and ethical considerations. However, it also introduces challenges related to integration and requires careful consideration of ethical and responsible AI use.
Last, in our future work for the document ranking project, we aim to improve the model by incorporating a variety of transformer architectures to represent the resumes and job descriptions, while also making modifications to the existing model architecture. Also, an alternative avenue for future research in the domain of query-document ranking involves exploring the concept of query expansion.

Query expansion Query expansion is a technique used in information retrieval and search engines to improve the quality of search results by broadening the scope of the user's query. It involves adding additional terms or phrases to the original search query to retrieve more relevant documents. In our project, the query will be a job description. There are two main types of query expansion:

- Relevance-Based: In this approach, additional terms or phrases are added to the query based on the content of the top-ranked documents in the initial search results. These terms are assumed to be related to the user's query and can help in finding more relevant documents. For example, if a user searches for "climate change," relevance-based query expansion might add terms like "global warming" or "carbon emissions" based on the content of highly ranked documents.

- Thesaurus-Based: Thesaurus-based query expansion relies on predefined synonyms or related term lists. When a user submits a query, the system looks up synonyms or related terms for the query terms in a thesaurus or synonym database and adds them to the query. For instance, if a user searches

for "car," the system might expand the query to include synonyms like "automobile" or "vehicle."

Query expansion aims to overcome limitations in users' query formulations. Users might not always use the most relevant or precise terms when searching for information. By expanding the query, the search engine attempts to capture the user's intent more accurately and retrieve a wider range of potentially relevant documents. However, query expansion is not without challenges. It can introduce noise if irrelevant terms are added to the query, and managing the expanded query's length is crucial to avoid overwhelming users with too many results. Additionally, evaluating the effectiveness of query expansion techniques and ensuring they genuinely improve search results is an ongoing area of research in information retrieval.

Appendix A

ACM Copyright (AVI 2022, Interactive clustering and high-recall information retrieval using language models)

ACM Copyright and Audio/Video Release Title of the Work: Interactive clustering and high-recall information retrieval using language models Submission ID:66 Author/Presenter(s): Sima Rezaeipourfarsangi:Dalhousie University;Ningyuan Pei:University of Alberta;Ehsan Sherkat:Salesforce;Evangelos Milios:Dalhousie University Type of material:short paper Publication and/or Conference Name: Proceedings of the 2022 International Conference on Advanced Visual Interfaces Proceedings I. Copyright Transfer, Reserved Rights and Permitted Uses * Your Copyright Transfer is conditional upon you agreeing to the terms set out below. Copyright to the Work and to any supplemental files integral to the Work which are submitted with it for review and publication such as an extended proof, a PowerPoint outline, or appendices that may exceed a printed page limit, (including without limitation, the right to publish the Work in whole or in part in any and all forms of media, now or hereafter known) is hereby transferred to the ACM (for Government work, to the extent transferable) effective as of the date of this agreement, on the understanding that the Work has been accepted for publication by ACM. Reserved Rights and Permitted Uses (a) All rights and permissions the author has not granted to ACM are reserved to the Owner, including all other proprietary rights such as patent or trademark rights. (b) Furthermore, notwithstanding the exclusive rights the Owner has granted to ACM, Owner shall have the right to do the following: (i) Reuse any portion of the Work, without fee, in any future works written or edited by the Author, including books, lectures and presentations in any and all media. (ii) Create a "Major Revision" which is wholly owned by the author (iii) Post the Accepted Version of the Work on (1) the Author's home page, (2) the Owner's institutional repository, (3) any repository legally mandated by an agency funding the research on which the Work is based,

and (4) any non-commercial repository or aggregation that does not duplicate ACM tables of contents, i.e., whose patterns of links do not substantially duplicate an ACM-copyrighted volume or issue. Non-commercial repositories are here understood as repositories owned by non-profit organizations that do not charge a fee for accessing deposited articles and that do not sell advertising or otherwise profit from serving articles. (iv) Post an "Author-Izer" link enabling free downloads of the Version of Record in the ACM Digital Library on (1) the Author's home page or (2) the Owner's institutional repository; (v) Prior to commencement of the ACM peer review process, post the version of the Work as submitted to ACM ("Submitted Version" or any earlier versions) to non-peer reviewed servers; (vi) Make free distributions of the final published Version of Record internally to the Owner's employees, if applicable; (vii) Make free distributions of the published Version of Record for Classroom and Personal Use; (viii) Bundle the Work in any of Owner's software distributions; and (ix) Use any Auxiliary Material independent from the Work. (x) If your paper is withdrawn before it is published in the ACM Digital Library, the rights revert back to the author(s). When preparing your paper for submission using the ACM TeX templates, the rights and permissions information and the bibliographic strip must appear on the lower left hand portion of the first page. The new ACM Consolidated TeX template Version 1.3 and above automatically creates and positions these text blocks for you based on the code snippet which is system-generated based on your rights management choice and this particular conference. NOTE: For authors using the ACM Microsoft Word Master Article Template and Publication Workflow, The ACM Publishing System (TAPS) will add the rights statement to your papers for you. Please check with your conference contact for information regarding submitting your source file(s) for processing. NOTE: For authors using the ACM Microsoft Word Master Article Template and Publication Workflow, The ACM Publishing System (TAPS) will add the rights statement to your papers for you. Please check with your conference contact for information regarding submitting your source file(s) for processing. If you are using the ACM Interim Microsoft Word template, or still using or older versions of the ACM SIGCHI template, you must copy and paste the following text

block into your document as per the instructions provided with the templates you are using: Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org. AVI 2022, June 6–10, 2022, Frascati, Rome, Italy © 2022 Association for Computing Machinery. ACM ISBN 978-1-4503-9719-3/22/06…15.00 https://doi.org/10.1145/3531073.3531174 NOTE: Make sure to include your article's DOI as part of the bibstrip data; DOIs will be registered and become active shortly after publication in the ACM Digital Library. Once you have your camera ready copy ready, please send your source files and PDF to your event contact for processing. A. Assent to Assignment. I hereby represent and warrant that I am the sole owner (or authorized agent of the copyright owner(s)), with the exception of third party materials detailed in section III below. I have obtained permission for any third-party material included in the Work. B. Declaration for Government Work. I am an employee of the National Government of my country/region and my Government claims rights to this work, or it is not copyrightable (Government work is classified as Public Domain in U.S. only) Are any of the co-authors, employees or contractors of a National Government? Yes N o Country/Region: Canada II. Permission For Conference Recording and Distribution * Your Audio/Video Release is conditional upon you agreeing to the terms set out below. I hereby grant permission for ACM to include my name, likeness, presentation and comments in any and all forms, for the Conference and/or Publication. I further grant permission for ACM to record and/or transcribe and reproduce my presentation as part of the ACM Digital Library, and to distribute the same for sale in complete or partial form as part of an ACM product on CD-ROM, DVD, webcast, USB device, streaming video or any other media format now or hereafter known. I understand that my presentation will not be sold separately as a

stand-alone product without my direct consent. Accordingly, I give ACM the right to use my image, voice, pronouncements, likeness, and my name, and any biographical material submitted by me, in connection with the Conference and/or Publication, whether used in excerpts or in full, for distribution described above and for any associated advertising or exhibition. Do you agree to the above Audio/Video Release? Yes N o III. Auxiliary Material Do you have any Auxiliary Materials? Yes No IV. Third Party Materials In the event that any materials used in my presentation or Auxiliary Materials contain the work of third-party individuals or organizations (including copyrighted music or movie excerpts or anything not owned by me), I understand that it is my responsibility to secure excerpts or anything not owned by me), I understand that it is my responsibility to secure any necessary permissions and/or licenses for print and/or digital publication, and cite or attach them below. We/I have not used third-party material. We/I have used third-party materials and have necessary permissions. V. Artistic Images If your paper includes images that were created for any purpose other than this paper and to which you or your employer claim copyright, you must complete Part V and be sure to include a notice of copyright with each such image in the paper. We/I do not have any artistic images. We/I have any artistic images. VI. Representations, Warranties and Covenants The undersigned hereby represents, warrants and covenants as follows: (a) Owner is the sole owner or authorized agent of Owner(s) of the Work; (b) The undersigned is authorized to enter into this Agreement and grant the rights included in this license to ACM; (c) The Work is original and does not infringe the rights of any third party; all permissions for use of third-party materials consistent in scope and duration with the rights granted to ACM have been obtained, copies of such permissions have been provided to ACM, and the Work as submitted to ACM clearly and accurately indicates the credit to the proprietors of any such third-party materials (including any applicable copyright notice), or will be revised to indicate such credit; (d) The Work has not been published except for informal postings on non-peer reviewed servers, and Owner covenants to use best efforts to place ACM DOI pointers on any such prior postings; (e) The Auxiliary Materials, if any, contain no malicious code, virus, trojan horse or other software routines or

hardware components designed to permit unauthorized access or to disable, erase or otherwise harm any computer systems or software; and (f) The Artistic Images, if any, are clearly and accurately noted as such (including any applicable copyright notice) in the Submitted Version. I agree to the Representations, Warranties and Covenants DATE: 04/14/2022 sent to sima.rezaei@dal.ca at 14:04:38

Appendix B

ACM Copyright (DocEng 2023, Addressing the gap between current language models and key-term-based clustering)

ACM Publishing License and Audio/Video Release Title of the Work: Addressing the gap between current language models and key-term-based clustering Submission ID:fp_27 Author/Presenter(s): Eric M. Cabral:Universidade de São Paulo;Sima Rezaeipourfarsangi:Dalhousie University;Maria Cristina F. De Oliveira:Universidade de São Paulo;Evangelos E. Milios:Dalhousie University;Rosane Minghim:University College Cork
Type of material:full paper Publication and/or Conference Name: DocEng '23: ACM Symposium on Document Engineering 2023 Proceedings 1. Glossary 2. Grant of Rights (a) Owner hereby grants to ACM an exclusive, worldwide, royalty-free, perpetual, irrevocable, transferable and sublicenseable license to publish, reproduce and distribute all or any part of the Work in any and all forms of media, now or hereafter known, including in the above publication and in the ACM Digital Library, and to authorize third parties to do the same. (b) In connection with software and "Artistic Images and "Auxiliary Materials, Owner grants ACM non-exclusive permission to publish, reproduce and distribute in any and all forms of media, now or hereafter known, including in the above publication and in the ACM Digital Library. (c) In connection with any "Minor Revision", that is, a derivative work containing less than twenty-five percent (25ACM all rights in the Minor Revision that Owner grants to ACM with respect to the Work, and all terms of this Agreement shall apply to the Minor Revision. (d) If your paper is withdrawn before it is published in the ACM Digital Library, the rights revert back to the author(s). A. Grant of Rights. I grant the rights and agree to the terms described above. B. Declaration for Government Work. I am an employee of the national government of my country/region and my Government claims rights to this work, or it is not copyrightable (Government work is classified as Public Domain in U.S.

only) Are you a contractor of your National Government? Yes N o Are any of the co-authors, employees or contractors of a National Government? Yes N o 3. Reserved Rights and Permitted Uses.
(a) All rights and permissions the author has not granted to ACM in Paragraph 2 are reserved to the Owner, including without limitation the ownership of the copyright of the Work and all other proprietary rights such as patent or trademark rights. (b) Furthermore, notwithstanding the exclusive rights the Owner has granted to ACM in Paragraph 2(a), Owner shall have the right to do the following: (i) Reuse any portion of the Work, without fee, in any future works written or edited by the Author, including books, lectures and presentations in any and all media. (ii) Create a "Major Revision" which is wholly owned by the author (iii) Post the Accepted Version of the Work on (1) the Author's home page, (2) the Owner's institutional repository, (3) any repository legally mandated by an agency funding the research on which the Work is based, and (4) any non-commercial repository or aggregation that does not duplicate ACM tables of contents, i.e., whose patterns of links do not substantially duplicate an ACM-copyrighted volume or issue. Non-commercial repositories are here understood as repositories owned by non-profit organizations that do not charge a fee for accessing deposited articles and that do not sell advertising or otherwise profit from serving articles. (iv) Post an "Author-Izer" link enabling free downloads of the Version of Record in the ACM Digital Library on (1) the Author's home page or (2) the Owner's institutional repository; (v) Prior to commencement of the ACM peer review process, post the version of the Work as submitted to ACM ("Submitted Version" or any earlier versions) to non-peer reviewed servers; (vi) Make free distributions of the final published Version of Record internally to the Owner's employees, if applicable; (vii) Make free distributions of the published Version of Record for Classroom and Personal Use; (viii) Bundle the Work in any of Owner's software distributions; and (ix) Use any Auxiliary Material independent from the Work. When preparing your paper for submission using the ACM TeX templates, the rights and permissions information and the bibliographic strip must appear on the lower left hand portion of the first page. The new ACM Consolidated TeX template Version 1.3 and above automatically creates and positions these text blocks for you based on the code

snippet which is system-generated based on your rights management choice and this particular conference. When creating your document, please make sure that you are only using TAPS accepted packages. (If you would like to use a package not on the list, please send suggestions to acmtexsupport@aptaracorp.com RE: TAPS LaTeX Package
evaluation.) NOTE: For authors using the ACM Microsoft Word Master Article Template and Publication Workflow, The ACM Publishing System (TAPS) will add the rights statement to your papers for you. Please check with your conference contact for information regarding submitting your source file(s) for processing.
NOTE: For authors using the ACM Microsoft Word Master Article Template and Publication Workflow, The ACM Publishing System (TAPS) will add the rights statement to your papers for you. Please check with your conference contact for information regarding submitting your source file(s) for processing.
If you are using the ACM Interim Microsoft Word template, or still using or older versions of the ACM SIGCHI template, you must copy and paste the following text block into your document as per the instructions provided with the templates you are using: Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org. DocEng '23, August 22–25, 2023, Limerick, Ireland © 2023 Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0027-9/23/08...15.00
https://doi.org/10.1145/3573128.3604900 NOTE: Make sure to include your article's DOI as part of the bibstrip data; DOIs will be registered and become active shortly after publication in the ACM Digital Library.
Once you have your camera ready copy ready, please send your source files and PDF to your event contact for processing.

Appendix C

ACM Copyright (DocEng 2023, AI-powered Resume-Job matching: A document ranking approach using deep neural networks)

ACM Publishing License and Audio/Video Release Title of the Work: AI-powered Resume-Job matching: A document ranking approach using deep neural networks Submission ID:sp_9 Author/Presenter(s): Sima Rezaeipourfarsangi:Dalhousie University Type of material:short paper Publication and/or Conference Name: DocEng '23: ACM Symposium on Document Engineering 2023 Proceedings 1. Glossary 2. Grant of Rights (a) Owner hereby grants to ACM an exclusive, worldwide, royalty-free, perpetual, irrevocable, transferable and sublicenseable license to publish, reproduce and distribute all or any part of the Work in any and all forms of media, now or hereafter known, including in the above publication and in the ACM Digital Library, and to authorize third parties to do the same. (b) In connection with software and "Artistic Images and "Auxiliary Materials, Owner grants ACM non-exclusive permission to publish, reproduce and distribute in any and all forms of media, now or hereafter known, including in the above publication and in the ACM Digital Library. (c) In connection with any "Minor Revision", that is, a derivative work containing less than twenty-five percent (25ACM all rights in the Minor Revision that Owner grants to ACM with respect to the Work, and all terms of this Agreement shall apply to the Minor Revision. (d) If your paper is withdrawn before it is published in the ACM Digital Library, the rights revert back to the author(s). A. Grant of Rights. I grant the rights and agree to the terms described above. B. Declaration for Government Work. I am an employee of the national government of my country/region and my Government claims rights to this work, or it is not copyrightable (Government work is classified as Public Domain in U.S. only) Are you a contractor of your National Government? Yes N o 3. Reserved Rights and Permitted Uses. (a) All rights and permissions the author has not

granted to ACM in Paragraph 2 are reserved to the Owner, including without limitation the ownership of the copyright of the Work and all other proprietary rights such as patent or trademark rights. (b) Furthermore, notwithstanding the exclusive rights the Owner has granted to ACM in Paragraph 2(a), Owner shall have the right to do the following: (i) Reuse any portion of the Work, without fee, in any future works written or edited by the Author, including books, lectures and presentations in any and all media. (ii) Create a "Major Revision" which is wholly owned by the author (iii) Post the Accepted Version of the Work on (1) the Author's home page, (2) the Owner's institutional repository, (3) any repository legally mandated by an agency funding the research on which the Work is based, and (4) any non-commercial repository or aggregation that does not duplicate ACM tables of contents, i.e., whose patterns of links do not substantially duplicate an ACM-copyrighted volume or issue. Non-commercial repositories are here understood as repositories owned by non-profit organizations that do not charge a fee for accessing deposited articles and that do not sell advertising or otherwise profit from serving articles. (iv) Post an "Author-Izer" link enabling free downloads of the Version of Record in the ACM Digital Library on (1) the Author's home page or (2) the Owner's institutional repository; (v) Prior to commencement of the ACM peer review process, post the version of the Work as submitted to ACM ("Submitted Version" or any earlier versions) to non-peer reviewed servers; (vi) Make free distributions of the final published Version of Record internally to the Owner's employees, if applicable; (vii) Make free distributions of the published Version of Record for Classroom and Personal Use; (viii) Bundle the Work in any of Owner's software distributions; and (ix) Use any Auxiliary Material independent from the Work. When preparing your paper for submission using the ACM TeX templates, the rights and permissions information and the bibliographic strip must appear on the lower left hand portion of the first page. The new ACM Consolidated TeX template Version 1.3 and above automatically creates and positions these text blocks for you based on the code snippet which is system-generated based on your rights management choice and this particular conference. When creating your document, please make sure that you are only using TAPS accepted packages. (If

you would like to use a package not on the list, please send suggestions to acmtexsupport@aptaracorp.com RE: TAPS LaTeX Package evaluation.) NOTE: For authors using the ACM Microsoft Word Master Article Template and Publication Workflow, The ACM Publishing System (TAPS) will add the rights statement to your papers for you. Please check with your conference contact for information regarding submitting your source file(s) for processing. information regarding submitting your source file(s) for processing. NOTE: For authors using the ACM Microsoft Word Master Article Template and Publication Workflow, The ACM Publishing System (TAPS) will add the rights statement to your papers for you. Please check with your conference contact for information regarding submitting your source file(s) for processing. If you are using the ACM Interim Microsoft Word template, or still using or older versions of the ACM SIGCHI template, you must copy and paste the following text block into your document as per the instructions provided with the templates you are using: Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org. DocEng '23, August 22–25, 2023, Limerick, Ireland © 2023 Copyright is held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 979-8-4007-0027-9/23/08…15.00 https://doi.org/10.1145/3573128.3609347 NOTE: Make sure to include your article's DOI as part of the bibstrip data; DOIs will be registered and become active shortly after publication in the ACM Digital Library. Once you have your camera ready copy ready, please send your source files and PDF to your event contact for processing. 4. ACM Citation and Digital Object Identifier. (a) In connection with any use by the Owner of the Definitive Version, Owner shall include the ACM citation and ACM Digital Object Identifier (DOI). (b) In connection with any use by the Owner of the Submitted Version (if accepted) or the Accepted Version or a

Minor Revision, Owner shall use best efforts to display the ACM citation, along with a statement substantially similar to the following: "© [Owner] [Year]. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive version was published in Source Publication, https://doi.org/10.1145/number." 5. Livestreaming and Distribution You are giving a presentation at the annual conference. This section of the rights form gives you the opportunity to grant or deny ACM the ability to make this presentation more widely seen, through (a) livestreaming of the presentation during the conference and/or (b) distributing the presentation after the conference in the ACM Digital Library, the "Conference Presentations" USB, and media outlets such as Vimeo and YouTube. It also provides you the opportunity to grant or deny our use of the presentation in promotional and marketing efforts after the conference. Not all conference presentations are livestreamed; you will be notified in advance of the possibility of your presentation being livestreamed. The permissions granted and/or denied here apply to all presentations of this material at the conference, including (but not limited to) the primary presentation and any program-specific "fast forward" presentations. ACM's policy on the use of third-party material applies to your presentation as well as the documentation of your work; if you are using others' material in your presentation, including audio, you must identify that material on the ACM rights form and in the presentation where it is used, and secure permission to use the material where necessary. Livestreaming. I grant permission to ACM to livestream my presentation during the conference (a "livestream" is a synchronous distribution of the presentation to the public, separate from the presentation distributed to conference registrants). Yes No Post-Conference Distribution. I grant permission to ACM to distribute the recording of my presentation after the conference as listed above. Yes No 6. Auxiliary Material Do you have any Auxiliary Materials? Yes No 7. Third Party Materials In the event that any materials used in my presentation or Auxiliary Materials contain the work of third-party individuals or organizations (including copyrighted music or movie excerpts or anything not owned by me), I understand that it is my responsibility to secure any necessary permissions and/or licenses for print and/or digital publication, and cite or attach

them below. We/I have not used third-party material. We/I have used third-party materials and have necessary permissions. 8. Artistic Images If your paper includes images that were created for any purpose other than this paper and to which you or your employer claim copyright, you must complete Part IV and be sure to include a notice of copyright with each such image in the paper. We/I do not have any artistic images. We/I have any artistic images. 9. Representations, Warranties and Covenants The undersigned hereby represents, warrants and covenants as follows: (a) Owner is the sole owner or authorized agent of Owner(s) of the Work; (b) The undersigned is authorized to enter into this Agreement and grant the rights included in this license to ACM; (c) The Work is original and does not infringe the rights of any third party; all permissions for use of third-party materials consistent in scope and duration with the rights granted to ACM have been obtained, copies of such permissions have been provided to ACM, and the Work as submitted to ACM clearly and accurately indicates the credit to the proprietors of any such third-party materials (including any applicable copyright notice), or will be revised to indicate such credit; (d) The Work has not been published except for informal postings on non-peer reviewed servers, and Owner covenants to use best efforts to place ACM DOI pointers on any such prior postings; (e) The Auxiliary Materials, if any, contain no malicious code, virus, trojan horse or other software routines or hardware components designed to permit unauthorized access or to disable, erase or otherwise harm any computer systems or software; and (f) The Artistic Images, if any, are clearly and accurately noted as such (including any applicable copyright notice) in the Submitted Version. I agree to the Representations, Warranties and Covenants. I agree to the Representations, Warranties and Covenants. 10. Enforcement. At ACM's expense, ACM shall have the right (but not the obligation) to defend and enforce the rights granted to ACM hereunder, including in connection with any instances of plagiarism brought to the attention of ACM. Owner shall notify ACM in writing as promptly as practicable upon becoming aware that any third party is infringing upon the rights granted to ACM, and shall reasonably cooperate with ACM in its defense or enforcement. 11. Governing Law This Agreement shall be governed by, and construed in accordance with, the laws of the

state of New York applicable to contracts entered into and to be fully performed therein. DATE: 07/14/2023 sent to sima.rezaei@dal.ca at 19:07:40

Bibliography

[1] K Mueen Ahmed and Bandar Al Dhubaib. Zotero: A bibliographic assistant to researcher. *Journal of Pharmacology and Pharmacotherapeutics*, 2(4):303, 2011.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery; Data Mining*, New York, NY, USA, 2019. Association for Computing Machinery.

[3] Aretha B. Alencar, Maria Cristina F. de Oliveira, and Fernando V. Paulovich. Seeing beyond reading: a survey on visual text analytics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6):476–492, nov 2012.

[4] Dimo Angelov. Top2vec: Distributed representations of topics. *arXiv preprint arXiv:2008.09470*, 2020.

[5] Pranjal Awasthi, Maria Florina Balcan, and Konstantin Voevodski. Local algorithms for interactive clustering. *Journal of Machine Learning Research*, 32(2):550–558, 2014.

[6] Juhee Bae, Tove Helldin, Maria Riveiro, Sławomir Nowaczyk, Mohamed-Rafik Bouguelia, and Göran Falkman. Interactive clustering: A comprehensive review. *ACM Computing Surveys (CSUR)*, 53(1):1–39, 2020.

[7] Maria-Florina Balcan and Avrim Blum. Clustering with Interactive Feedback. In Yoav Freund, László Györfi, György Turán, and Thomas Zeugmann, editors, *Algorithmic Learn. Theory*, pages 316–328, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[8] Sumit Basu, Danyel Fisher, Steven Drucker, and Hao Lu. Assisting users with clustering tasks by combining metric learning and classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.

[9] André Bauer, Marwin Züfle, Nikolas Herbst, Samuel Kounev, and Valentin Curtef. Telescope: An automatic feature extraction and transformation approach for time series forecasting on a level-playing field. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1902–1905. IEEE, 2020.

[10] André Bauer, Marwin Züfle, Nikolas Herbst, Albin Zehe, Andreas Hotho, and Samuel Kounev. Time series forecasting for self-aware systems. *Proceedings of the IEEE*, 108(7):1068–1093, 2020.

[11] Yoshua Bengio. Deep learning of representations: Looking forward. In *International conference on statistical language and speech processing*, pages 1–37. Springer, 2013.

[12] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *Trans. Assoc. Comput. Linguist.*, 5:135–146, dec 2017.

[13] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[14] Eric Macedo Cabral. Interactive keyterm-based document clustering and visualization via neural language models, jun 2020.

[15] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[16] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[17] Sanjoy Chakraborty, Apu Kumar Saha, Sushmita Sharma, Seyedali Mirjalili, and Ratul Chakraborty. A novel enhanced whale optimization algorithm for global optimization. *Computers & Industrial Engineering*, 153:107086, 2021.

[18] Nitesh Chawla and Wei Wang. *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 2017.

[19] Tingting Chen, Jun Xu, Haochao Ying, Xiaojun Chen, Ruiwei Feng, Xueling Fang, Honghao Gao, and Jian Wu. Prediction of extubation failure for intensive care unit patients using light gradient boosting machine. *IEEE Access*, 7:150960–150968, 2019.

[20] Mark Chockalingam. Forecast accuracy and inventory strategies. *Demand Planning LLC. Recuperado de: https://demandplanning. net/documents/dmdaccuracywebVersions. pdf*, 2009.

[21] Jaegul Choo, Changhyun Lee, Chandan K. Reddy, and Haesun Park. UTOPIAN: User-Driven Topic Modeling Based on Interactive Nonnegative Matrix Factorization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):1992–2001, dec 2013.

[22] Jaegul Choo, Hanseung Lee, Zhicheng Liu, John Stasko, and Haesun Park. An interactive visual testbed system for dimension reduction and clustering of large-scale high-dimensional data. In Pak Chung Wong, David L. Kao, Ming C. Hao, Chaomei Chen, and Christopher G. Healey, editors, *Proc. SPIE 8654, Vis. Data Anal. 2013*, page 865402, Burlingame, California, United States, feb 2013. SPIE.

[23] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[24] Geraldo N Corrêa, Ricardo M Marcacini, Eduardo R Hruschka, and Solange O Rezende. Interactive textual feature selection for consensus clustering. *Pattern Recognition Letters*, 52:25–31, 2015.

[25] João Vitor Andrioli de Souza, Lucas Emanuel Silva E Oliveira, Yohan Bonescki Gumiel, Deborah Ribeiro Carvalho, and Claudia Maria Cabral Moro. Exploiting siamese neural networks on short text similarity tasks for multiple domains and languages. In *Computational Processing of the Portuguese Language: 14th International Conference, PROPOR 2020, Evora, Portugal, March 2–4, 2020, Proceedings 14*, pages 357–367. Springer, 2020.

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[27] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. 2019 Conf. North*, pages 4171–4186, Stroudsburg, PA, USA, oct 2019. Association for Computational Linguistics.

[28] Sounak Dey, Anjan Dutta, J Ignacio Toledo, Suman K Ghosh, Josep Lladós, and Umapada Pal. Signet: Convolutional siamese network for writer independent offline signature verification. *arXiv preprint arXiv:1707.02131*, 2017.

[29] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018.

[30] Wenwen Dou, Xiaoyu Wang, Remco Chang, and William Ribarsky. ParallelTopics: A probabilistic approach to exploring document collections. In *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 231–240, Providence, RI, USA, oct 2011. IEEE.

[31] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[32] Ajla Elmasdotter and Carl Nyströmer. A comparative study between lstm and arima for sales forecasting in retail, 2018.

[33] Paolo Federico, Florian Heimerl, Steffen Koch, and Silvia Miksch. A Survey on Visual Approaches for Analyzing Scientific Literature and Patents. *IEEE Transactions on Visualization and Computer Graphics*, 23(9):2179–2198, 2017.

[34] E. B. Fowlkes and C. L. Mallows. A Method for Comparing Two Hierarchical Clusterings. *J. Am. Stat. Assoc.*, 78(383):553–569, sep 1983.

[35] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971, 1987.

[36] John M. Gamma, Erich and Helm, Richard and Johnson, Ralph and Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, USA, 1 edition, 1994.

[37] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.

[38] Cheng Guo and Felix Berkhahn. Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737*, 2016.

[39] Sifei Han, Lingyun Shi, Russell Richie, and Fuchiang R Tsui. Building siamese attention-augmented recurrent convolutional neural networks for document similarity scoring. *Information Sciences*, 615:90–102, 2022.

[40] Zellig S. Harris. Distributional Structure. *WORD*, 10(2-3):146–162, aug 1954.

[41] Steven Craig Hillmer and George C Tiao. An arima-model-based approach to seasonal adjustment. *Journal of the American Statistical Association*, 77(377):63–70, 1982.

[42] Wolfram Höpken, Tobias Eberle, Matthias Fuchs, and Maria Lexhagen. Google trends data for analysing tourists' online search behaviour and improving demand forecasting: the case of åre, sweden. *Information Technology & Tourism*, 21(1):45–62, 2019.

[43] Yuening Hu, Jordan Boyd-Graber, Brianna Satinoff, and Alison Smith. Interactive topic modeling. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 248–257, Portland, Oregon, USA, jun 2011. Association for Computational Linguistics.

[44] Degen Huang, Anil Ahmed, Syed Yasser Arafat, Khawaja Iftekhar Rashid, Qasim Abbas, and Fuji Ren. Sentence-embedding and similarity via hybrid bidirectional-lstm and cnn utilizing weighted-pooling attention. *IEICE TRANSACTIONS on Information and Systems*, 103(10):2216–2227, 2020.

[45] Lawrence Hubert and Phipps Arabie. Comparing partitions. *J. Classif.*, 2(1):193–218, 1985.

[46] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *15th Conf. Eur. Chapter Assoc. Comput. Linguist. EACL 2017 - Proc. Conf.*, 2:427–431, 2017.

[47] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002.

[48] Tristan Karb, Niklas Kühl, Robin Hirt, and Varvara Glivici-Cotruta. A network-based transfer learning approach to improve sales forecasting of new products. *arXiv preprint arXiv:2005.06978*, 2020.

[49] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[50] Tae Kyun Kim. T test as a parametric statistic. *Korean journal of anesthesiology*, 68(6):540–546, 2015.

[51] Mateusz Krzyziński, Mikołaj Spytek, Hubert Baniecki, and Przemysław Biecek. Survshap (t): Time-dependent explanations of machine learning survival models. *Knowledge-Based Systems*, page 110234, 2022.

[52] Ken Lang. NewsWeeder: Learning to Filter Netnews. In *Mach. Learn. Proc. 1995*, pages 331–339. Elsevier, San Francisco (CA), 1995.

[53] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

[54] Quoc V. Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. *Proceedings of the 31 st International Conference on Machine Learning*, 32(2):1188–1196, may 2014.

[55] Yann LeCun, Yoshua Bengio, Geoffrey Hinton, et al. Deep learning. nature, 521 (7553), 436-444. *Google Scholar Google Scholar Cross Ref Cross Ref*, page 25, 2015.

[56] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[57] Hanseung Lee, Jaeyeon Kihm, Jaegul Choo, John Stasko, and Haesun Park. iVisClustering: An Interactive Visual Document Clustering via Topic Modeling. *Computer Graphics Forum*, 31(3pt3):1155–1164, jun 2012.

[58] Xuerong Li, Wei Shang, and Shouyang Wang. Text-based crude oil price forecasting: A deep learning approach. *International Journal of Forecasting*, 35(4):1548–1560, 2019.

[59] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[60] Ana LD Loureiro, Vera L Miguéis, and Lucas FM da Silva. Exploring the use of deep neural networks for sales forecasting in fashion retail. *Decision Support Systems*, 114:81–93, 2018.

[61] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[62] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[63] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, abs/1301.3781, 2013.

[64] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space, jan 2013.

[65] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality, oct 2013.

[66] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, Red Hook, NY, USA, 2013. Curran Associates Inc.

[67] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013.

[68] Seyedali Mirjalili and Andrew Lewis. The whale optimization algorithm. *Advances in engineering software*, 95:51–67, 2016.

[69] Seyednaser Nourashrafeddin, Ehsan Sherkat, Rosane Minghim, and Evangelos E. Milios. A Visual Approach for Interactive Keyterm-Based Clustering. *ACM Transactions on Interactive Intelligent Systems*, 8(1):1–35, feb 2018.

[70] Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, jun 1994.

[71] Eunhye Park, Jinah Park, and Mingming Hu. Tourism demand forecasting with online news data mining. *Annals of Tourism Research*, 90:103273, 2021.

[72] F.V. Paulovich, L.G. Nonato, Rosane Minghim, and Haim Levkowitz. Least Square Projection: A Fast High-Precision Multidimensional Projection Technique and Its Application to Document Mapping. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):564–575, may 2008.

[73] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, jan 2012.

[74] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[75] Reid Pryzant, Youngjoo Chung, and Dan Jurafsky. Predicting sales from the language of product descriptions. In *eCOM@ SIGIR*, 2017.

[76] Chen Qin, Shiji Song, Gao Huang, and Lei Zhu. Unsupervised neighborhood component analysis for clustering. *Neurocomputing*, 168:609–617, 2015.

[77] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. —, 2018.

[78] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 248–256, Singapore, 2009. Association for Computational Linguistics.

[79] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer, 2003.

[80] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP/IJCNLP*, pages 3982–3992, Hong Kong, China, 2019. Association for Computational Linguistics.

[81] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proc. 2019 Conf. Empir. Methods Nat. Lang. Process. 9th Int. Jt. Conf. Nat. Lang. Process.*, pages 3980–3990, Stroudsburg, PA, USA, 2019. Association for Computational Linguistics.

[82] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[83] Filipe Rodrigues, Ioulia Markou, and Francisco C Pereira. Combining time-series and textual data for taxi demand prediction in event areas: A deep learning approach. *Information Fusion*, 49:120–129, 2019.

[84] A. Rosenberg and J. Hirschberg. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, volume June, pages 410–420, 2007.

[85] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(C):53–65, nov 1987.

[86] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[87] Jürgen Schmidhuber, Sepp Hochreiter, et al. Long short-term memory. *Neural Comput*, 9(8):1735–1780, 1997.

[88] Eric WK See-To and Eric WT Ngai. Customer reviews for demand distribution and sales nowcasting: a big data approach. *Annals of Operations Research*, 270(1):415–431, 2018.

[89] Jamal Shahrabi, Esmaeil Hadavandi, and Shahrokh Asadi. Developing a hybrid intelligent model for forecasting problems: Case study of tourism demand time series. *Knowledge-Based Systems*, 43:112–122, 2013.

[90] Ehsan Sherkat. *Interactive Text Analytics for Document*. Phd dissertation, Dalhousie Universuty, 2018.

[91] Ehsan Sherkat, Evangelos E. Milios, and Rosane Minghim. A Visual Analytics Approach for Interactive Document Clustering. —, 10(1):1–33, aug 2019.

[92] Ehsan Sherkat, Seyednaser Nourashrafeddin, Evangelos E. Milios, and Rosane Minghim. Interactive Document Clustering Revisited. In *Proceedings of the 2018 Conference on Human Information Interaction&Retrieval - IUI '18*, pages 281–292, New York, New York, USA, 2018. ACM Press.

[93] Ehsan Sherkat, Seyednaser Nourashrafeddin, Evangelos E. Milios, and Rosane Minghim. Interactive document clustering revisited: A visual analytics approach. In *23rd International Conference on Intelligent User Interfaces*, IUI '18, page 281–292, New York, NY, USA, 2018. Association for Computing Machinery.

[94] Souhaib Ben Taieb and Amir F Atiya. A bias and variance analysis for multistep-ahead time series forecasting. *IEEE transactions on neural networks and learning systems*, 27(1):62–76, 2015.

[95] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[96] Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

[97] Toni Toharudin, Resa Septiani Pontoh, Rezzy Eko Caraka, Solichatus Zahroh, Youngjo Lee, and Rung Ching Chen. Employing long short-term memory and facebook prophet model in air temperature forecasting. *Communications in Statistics-Simulation and Computation*, pages 1–24, 2020.

[98] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models, 2023.

[99] Grigorios Tsoumakas. A survey of machine learning techniques for food sales prediction. *Artificial Intelligence Review*, 52(1):441–447, 2019.

[100] Cagatay Turkay, Julius Parulek, Nathalie Reuter, and Helwig Hauser. Integrating cluster formation and cluster evaluation in interactive visual analysis. In *Proceedings of the 27th Spring Conference on Computer Graphics*, pages 77–86, New York, NY, United States, 2011. Association for Computing Machinery.

[101] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[102] G.E. van der Maaten, L.J.P. and Hinton. Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, aug 2008.

[103] RM van Steenbergen and Martijn RK Mes. Forecasting demand profiles of new products. *Decision support systems*, 139:113401, 2020.

[104] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017-Decem(Nips):5999–6009, 2017.

[105] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[106] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11:2837–2854, 2010.

[107] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[108] Kyle Williams, Jian Wu, and C Lee Giles. Simseerx: a similar document search engine. In *Proceedings of the 2014 ACM symposium on Document engineering*, pages 143–146, 2014.

[109] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.

[110] Binrong Wu, Lin Wang, Sheng-Xiang Lv, and Yu-Rong Zeng. Effective crude oil price forecasting using new text-based and big-data-driven model. *Measurement*, 168:108468, 2021.

[111] Panpan Xu, Nan Cao, Huamin Qu, and John Stasko. Interactive visual co-cluster analysis of bipartite graphs. In *2016 IEEE Pacific Visualization Symposium (PacificVis)*, pages 32–39. IEEE, 2016.

[112] Wen-tau Yih and Christopher Meek. Learning vector representations for similarity measures. *Microsoft Research*, 2010.

[113] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.

[114] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

[115] Bin Zhang, Meichun Hsu, and Umeshwar Dayal. K-harmonic means-a data clustering algorithm. *Hewlett-Packard Labs Technical Report HPL-1999-124*, 55, 1999.

[116] Mingyang Zhang, Yixin Wang, and Zhiguo Wu. Data mining algorithm for demand forecast analysis on flash sales platform. *Complexity*, 2021, 2021.

[117] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[118] Binggui Zhou, Yunxuan Dong, Guanghua Yang, Fen Hou, Zheng Hu, Suxiu Xu, and Shaodan Ma. A graph-attention based spatial-temporal learning framework for tourism demand forecasting. *Knowledge-Based Systems*, 263:110275, 2023.