University of Texas at El Paso
ScholarWorks@UTEP

Open Access Theses & Dissertations

2024-05-01

Dynamic Storytelling Algorithms Using Contextual Aspects Of A Large Language Model

Alireza Pasha Nouri
University of Texas at El Paso

Follow this and additional works at: https://scholarworks.utep.edu/open_etd

 Part of the Computer Sciences Commons

Recommended Citation

Nouri, Alireza Pasha, "Dynamic Storytelling Algorithms Using Contextual Aspects Of A Large Language Model" (2024). *Open Access Theses & Dissertations*. 4129.
https://scholarworks.utep.edu/open_etd/4129

This is brought to you for free and open access by ScholarWorks@UTEP. It has been accepted for inclusion in Open Access Theses & Dissertations by an authorized administrator of ScholarWorks@UTEP. For more information, please contact lweber@utep.edu.

DYNAMIC STORYTELLING ALGORITHMS USING CONTEXTUAL ASPECTS OF A LARGE LANGUAGE MODEL

Alireza Pasha Nouri

Doctoral Program in Computer Science

APPROVED:

M. Shahriar Hossain, Ph.D., Chair

Monika Akbar, Ph.D.

Debzani Deb, Ph.D.

Stephen Crites, Ph.D.
Dean of the Graduate School

©Copyright

by

Alireza Nouri

2024

To my father who didn't wait for this moment

DYNAMIC STORYTELLING ALGORITHMS USING CONTEXTUAL ASPECTS OF A LARGE LANGUAGE MODEL

by

Alireza Pasha Nouri

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

THE UNIVERSITY OF TEXAS AT EL PASO

May 2024

Acknowledgements

I want to express my sincere gratitude to my mentor and committee chair, Dr. M. Shahriar Hossain, for his continuous support and guidance throughout my doctoral journey. His profound knowledge, wisdom, and guidance have been instrumental in helping me complete this dissertation and in supporting my professional and personal growth. His contributions to my education and personal development are immeasurable. Working under the direction of such a respected academic has been an honor, and I will carry forward the knowledge and values he taught me.

I would also like to thank my committee members, Dr. Monika Akbar and Dr. Debzani Deb, for their invaluable advice, insightful feedback, and constructive suggestions. Their guidance has been vital in helping me successfully complete my dissertation.

I want to give special thanks to the chair and staff of the Computer Science Department for their hard work, efforts, and commitment. Their support has been essential in providing the resources necessary to equip students like me for future careers in computer science.

My gratitude also extends to all the professors in the Computer Science Department who have played an instrumental role in my academic journey. Their expertise, dedication, and passion for teaching have profoundly impacted my transition from a student to a computer scientist.

Lastly, I have a deep appreciation for my wife, Melika Nouri. She always provided support and had never-ending faith in me. Her belief in me has been a source of encouragement and strength. I am forever grateful for her companionship and love.

Abstract

Storytelling is a set of algorithms used to create narratives by connecting documents in a sequence that accurately reflects the evolution of events and entities within a particular topic or theme. Early storytelling algorithms face challenges in encoding the progression and interconnections of information between consecutive texts, given that the conventional approaches rely primarily on connecting document pairs based on content overlap. They often neglect critical linguistic features, such as word contexts, semantics, the roles words play across different documents, and attention to the historical contexts of the underlying documents. Many existing storytelling models frequently produce story chains that, while connected by keywords, lack meaningful coherence in the chains. My dissertation introduces innovative LLM-driven storytelling algorithms to overcome the challenges traditional storytelling algorithms face and significantly enhance downstream tasks.

In my research, I propose a role-based contextual embedding algorithm using a large language model that provides a rich understanding of text in a document by considering the different roles of the same word in other documents. I also employ a generative diffusion model to seamlessly link documents within a narrative, even with gaps in the data, to ensure a smoother and more logical story progression. In my dissertation, I introduce a novel distributed attention similarity mechanism designed to control the narrative output of storytelling algorithms locally and globally. The techniques I have designed ensure that the generated stories are not merely connected by keywords but are also coherent and meaningfully sequenced. The experiments in my dissertation indicate that the proposed storytelling models generate more coherent, cohesive, and contextually rich narratives than existing approaches. In addition, I demonstrate that my proposed storytelling model has immense potential in vector data preparation for conventional machine-learning tasks.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Nowadays, with the exponential growth of data, the analyses of subjects and the evolution of topics have a significant place in helping us understand events from the past, present, and potentially the future [57]. Analyzing the development of an entity within a few documents is feasible; however, the task becomes considerably more complex as the number of documents increases. The challenge lies in the evolving nature of events that is more complex to capture across large datasets.

To analyze a subject or topic effectively, selecting the most relevant documents is crucial. Any irrelevant documents (noise) or missing key documents (gaps) can impact the performance of analytical models. This process for document selection ensures that the data provided in the model accurately reflect the evolution of a subject or topic over time. This process is crucial for allowing precise and comprehensive analysis of subject evolution or topic trends within large collections of documents. The process also provides a more accurate understanding of historical and contemporary events in a large data collection.

1.1 Storytelling Algorithm

A storytelling algorithm is a data mining task designed to construct a sequence of documents related to a topic, entity, or incident [34]. The storytelling algorithms use the temporal dimensions of a corpus to trace the evolution of entities and reveal obvious connections and highlight the relationships between entities that may not be immediately apparent [38]. Despite the essential role of storytelling in the NLP domain, there was not much research and study done in the past [92].

Selecting a chain of documents from data collection is one of the most challenging parts of analyzing the evolution of entities and subjects. When a narrative is constructed correctly, it enhances the performance of any downstream NLP models by offering a better understanding of the data [92, 4].

Fundamentally, a storytelling algorithm functions by taking a starting document and an ending document and then searching through a corpus to locate the documents that lie in between them to form a coherent narrative chain [77]. Some techniques initiate from a given document and collect other parts of the story while the end of the story is open. This process allows for the construction of narratives that maintain coherence and reflect the chronological or thematic evolution of events, making it possible to trace the development or evolution of specific subjects, entities, or events within a larger corpus.

This narrative may be segmented into chapters, each focusing on specific events, characters, locations, or time periods. Its length can vary depending on the narrative topic and the story's coverage.

The evolution of a story from its beginning, featuring particular entities and events, then follows the story's narrative to evolve to another point. These evolution, which occur smoothly and logically, form the backbone of different chapters within the story. A compelling story is characterized by these seamless transitions, allowing for a detailed analysis of how entities and themes evolve over time. This outcome provides analysts with a valuable understanding of the dynamics of a given subject.

1.1.1 Challenges and Gaps in Storytelling algorithms:

Despite recent studies on storytelling algorithms, these algorithms still face several challenges. We must address these challenges to construct coherent and focused narratives.

Topic Drifts: Despite recent advancements in storytelling algorithms, they are not entirely resistant to topic changes in a story. Some algorithms may lose the narrative line or focus while collecting documents to construct the narrative, leading to stories that include irrelevant or "noisy"

documents.

User-Defined Parameter Tuning: The quality of the stories generated relies heavily on user-defined parameters. Fine-tuning these settings is essential, yet it puts the burden on users to adjust them correctly based on their understanding of the textual data and the desired outcome, which makes the story's quality highly subjective.

Handling Missing Data: In large data collections, finding the missing documents that serve as supporting documents between two distanced documents in the timestamps is a significant challenge. The absence of these supporting documents can interrupt the flow of the story chain or lead to a deviation coming from unrelated topics.

Feature Enrichment: Extracting and enriching features from textual data is crucial for understanding the relationships and patterns between entities in a dataset. Improving feature extraction methods is important for enhancing the performance of story chains, as it provides a deeper understanding of the data and contributes to more coherent and engaging narratives.

Addressing these challenges is essential for developing and improving storytelling algorithms and their reliability, adaptability, and effectiveness in generating meaningful narratives.

1.1.2 Main Contributions of This Study

This dissertation introduces novel models designed to overcome the challenges of storytelling algorithms by enhancing their performance and utility.

Contextual Role-Based Storytelling (CoRBS) offers a new perspective on words' roles in texts. By recognizing and using these roles, CoRBS can extract more valuable features from the textual data. In addition to this contribution, CoRBS employs contextual documents and word embeddings alongside a contemporary-based search process to efficiently generate narratives from temporal corpora.

Diffusion-Based Storytelling (DifStoryGen), another novel model presented in this dissertation, uses a conditional diffusion model and a distributed attention mechanism. This approach effectively addresses the issues of missing content and topic drift, making the story chains more

applicable and valuable for various downstream NLP tasks.

1.2 Research Questions

I have formulated four research questions to address the challenges associated with storytelling algorithms. These questions are described in the following sections.

Research question 1: Besides the context and semantics of a text, what other aspects of natural language can be used to generate a better representation of a word, a paragraph, or a document?

Research question 2: What other possibilities of the contemporary aspect of natural language can be employed to help construct a story's evolution sufficiently?

Research question 3: How can a storytelling model handle the gaps between temporally distant articles and trace themes within stories in the absence of supporting documents in the corpus?

Research question 4: How does incorporating past documents in a storytelling algorithm during temporal progression impact the stories' coherence and thematic consistency?

1.2.1 Research Question 1

> Besides the context and semantics of a text, what other aspects of natural language can be used to generate a better representation of a word, a paragraph, or a document?

Text is a valuable resource for extracting information and analyzing the evolution of real-world entities and events. While textual data are naturally easy for humans to understand, machines face challenges in understanding this data. To address this issue, we must develop methods to transform words and sentences into representations that are easy for machines to understand.

In NLP, different techniques are used to transform words into numerical formats that machines can understand. One basic technique is Hot Encoding, which gives every word a unique binary vector. The length of this vector matches the total number of unique words in the text. In this vector, all parts are set to 0, except for one that corresponds to a specific word and is set to 1. This approach, while simple, provides a foundational step toward bridging the gap between human language and machine processing capabilities.

Term Frequency (TF) and Term Frequency-Inverse Document Frequency (TF-IDF) are two techniques that better represent words in a document collection by assigning a weighting to each word. TF measures how frequently a word appears in a document because the words appearing more frequently in a document are more important for understanding the document's contents. On the other hand, TF-IDF enhances this by reducing the weight of words that appear frequently across many documents in the corpus, effectively reducing the impact of common stopwords [43]. This method emphasizes the words that are crucial for differentiating the documents in a corpus.

Neural Networks introduce new approaches to word representation, allowing the encoding of words into lower-dimensional vectors that encapsulate much richer semantic information. Word2vec [53] and Global Vectors for Word Representation (GloVe) [49] are two of the main models in static word embedding algorithms. These techniques rely on the context in which words appear to generate embeddings reflecting semantic similarities and differences between words. These techniques offer a deeper semantic understanding of the text that is more aligned with how humans interpret language.

Contextual word representation techniques significantly advance the field of word embeddings by leveraging large language models that have been pre-trained on extensive collections of documents. These models capture the words' complex relationships and semantics in a language, offering a deeper understanding of word usage in different contexts. Embeddings from Language Models (ELMo) [63] and Bidirectional Encoder Representations from Transformers (BERT) [17] are two models for generating deep contextualized word representations used by different tasks and applications.

These approaches try to capture a word's semantics and contextual meaning in a text. Incorporating words' contexts and semantics will help us to obtain a richer representation of words and documents in a corpus. This contextual information generates a more suitable representation for different models to use. These embedding vectors help a machine better understand a text in a corpus [58]. I have provided a more detailed overview of different word embedding techniques in Chapter2.

Herein, I propose a novel approach, the Contextual Role-Based Storytelling algorithm **(CoRBS)**, to create an embedding space that captures the role of each word within its context. This role-based embedding technique captures a deeper understanding from textual data by uncovering aspects of language that previous methods may have missed. The effectiveness of this approach is analyzed by the empirical experiments presented in this dissertation. These experiments show how combining role-based vectors with contextual embedding vectors can significantly enhance the performance of storytelling models. As detailed in Section 3 of this dissertation, such models produce narratives with a stronger focus on the subjects and entities thus contributing to more coherent and topic-centered storytelling.

CoRBS aims to incorporate additional linguistic properties and enrich the word embedding vectors with a broader spectrum of language attributes. Such advancements promise to improve the capabilities of storytelling algorithms and provide new insights into the complex patterns of language that offer valuable contributions to NLP.

The main contributions of the CoRBS algorithm are as follows:

- **Inclusion of the roles of words:** CoRBS extracts the localized distribution of each word for

all its appearances in the same document. Based on the distribution of the vector similarities of the surroundings, CoRBS creates a localized cluster-based context that serves as a set for the roles of words in a text.

- **Lesser number of user-settable parameters:** Unlike previous storytelling models, our proposed algorithm has only one user-settable parameter. The parameter is easy to tweak, given that it is the Jaccard similarity threshold, varying between 0 and 1.

1.2.2 Research Question 2

> What other possibilities of the contemporary aspect of natural language can be employed to help construct a story's evolution sufficiently?

Dynamic word embedding refers to an embedding space where each vector has a time label. These labels indicate in which timestamp that word appeared. This new space allows us to track the changes in a word over different times. In Figure 1.1, Hamilton et al. [31] presented a visual example of a word embedding space and how the semantics of terms evolve over time. These are some words whose meanings have changed over the last decades. Semantic changes enable the study of how language evolves over time. It also helps analyze the evolution of specific words, entities, or events through time. Predicting an event or analyzing entities are some applications of this evolution. There are multiple studies on developing this temporal aspect of a text collection [36, 35, 31, 39].

A more detailed exploration of different methods in Temporal Word Embedding is presented in Chapter 2.

Traditional NLP techniques generally use the exact representation of the same word's multiple appearances. For example, Word2vec [53] and GloVe [49] consider global embeddings, even though the context of the word varies between different documents or even within different paragraphs of the same document. A word's context is localized by its surroundings, and each appearance of the same word may have a different context.

In addition, we cannot directly compare the embedding spaces formed on different temporal corpora. An embedding vector for a word in a specific timestamp is not comparable with an embedding vector of the same word from a different timestamp. There are research studies [91, 6, 11, 56, 26] that address this issue in temporal embeddings.

In CoRBS, two techniques are proposed to leverage the contemporary aspect of language and enable the model to trace the evolution of a story effectively. The first approach uses BERT to create distinct embedding vectors for each word's appearance within a corpus. This technique allows us to analyze all word appearances across different timestamps in one embedding space.

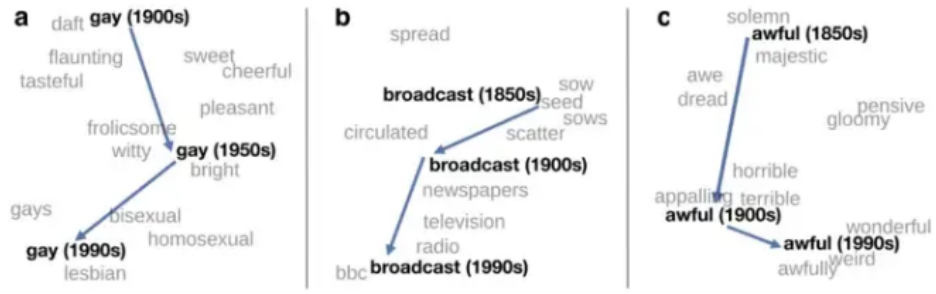

Figure 1.1: Form Hamilton et al. (2016) paper. "Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change."

This approach helps the model differentiate between word representations at different appearances. Hence, it facilitates a more efficient analysis of how entities evolve over time.

Previous storytelling algorithms employed various techniques to narrow the search field during the storytelling process. Hossain et al. [38] introduced a clique-based method to limit the search scope by focusing primarily on the connections between documents rather than considering the temporal dimension of the corpus.

CoRBS introduces a novel dynamic search process to enhance the storytelling algorithms' efficacy. This method uses a threshold parameter to determine whether to proceed with or halt the search effectively. The domain of this search process over a timestamped corpus is illustrated in Figure 1.2.

The main contributions to developing the contemporary aspect of a storytelling algorithm are as follows:

- **Contemporary context to capture evolution:** CoRBS uses BERT to capture the contemporary context of every word's appearance. Using this approach enables CoRBS to capture the evolution of a story better than other models.

- **Dynamic storytelling:** CoRBS introduces a new search process designed to create more coherent and cohesive story chains, including all relevant documents from a corpus in different timestamps. This approach significantly enhances the search mechanism and

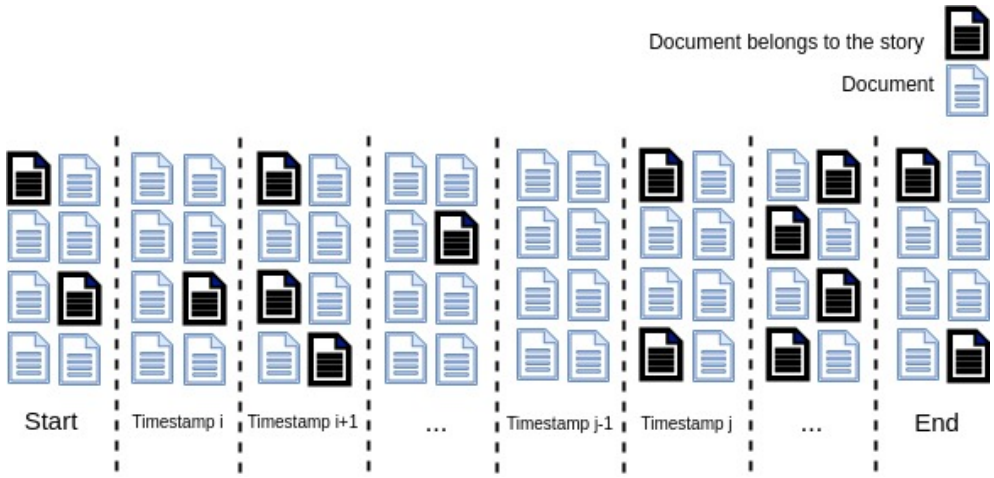

Figure 1.2: A series of documents belong to a story which is clustered by their timestamps used to dynamically search for a chain of documents

reduces the need for post-processing steps. Also, it makes the model more dynamic and less dependent on user-defined parameters.

1.2.3 Research Question 3

> How can a storytelling model handle the gaps between temporally distant articles and trace themes within stories in the absence of supporting documents in the corpus?

Many storytelling algorithms lack an effective mechanism to bridge the gaps in the text of document collections. This challenge often leads to the search process being prematurely halted or shifted into unrelated narratives, particularly in cases where documents that could logically connect parts of the story are missing.

Many algorithms employ a user-defined threshold to decide whether the search process should proceed or if there are no documents relevant to the narrative within the dataset. This parameter requires individual calibration for different tasks and datasets, making it necessary to define it specifically for each unique application.

If the threshold parameter is not defined correctly, this can significantly decrease the performance of a storytelling algorithm. Consequently, the model may generate shorter stories due to the absence of adequate supporting documents in the dataset or construct a document chain that shifts from the original topic midway through the narrative.

Storytelling algorithms require a mechanism capable of predicting missing documents during the search process and incorporating these hypothesized documents into the search process. This approach bridges the gap between two distanced documents, ensuring a continuous and coherent narrative flow, even in the absence of supporting documents within the corpus.

The Diffusion-Based Storytelling Algorithm (DifStoryGen) is a solution we developed to overcome the limitations faced by traditional storytelling models. This neural network incorporates a conditional diffusion-based module designed to generate new document embedding vectors that link two sequential timestamps. If a gap is caused by a missing supporting document between these timestamps, then the conditional network fills this gap with a hypothesis document embedding. Such a capability ensures that the model continues the search process smoothly without shifting the topic due to the absence of essential documents to maintain the narrative's coherence and continuity.

The main contributions of integrating a conditional diffusion model into a storytelling algorithm are as follows:

- **Bridging Narrative Gaps:** By employing a conditional diffusion-based document generation model, our algorithm effectively connects two articles in the absence of relevant supporting documents within the corpus. This capability is crucial for uncovering recurring themes and enriching the story's depth and continuity.
- **Less User-Defined Parameters:** The algorithm's capacity to use generated documents to determine the next document in the narrative allows it to navigate temporal shifts dynamically or define the story's evolution. This reduces the dependency on critical parameters that previously needed to be manually set by analysts, significantly simplifying the model's operation and making it more user-friendly.

1.2.4 Research Question 4

> How does incorporating past documents in a storytelling algorithm during temporal progression impact the stories' coherence and thematic consistency?

We have two types of storytelling algorithms. The first type is an open-ended algorithm in which the model receives a seed document and tries to build the story by looking at the rest of the chain from a given corpus. The second type is when the model receives two documents as start and end points and tries to find the documents between them from a corpus. The first task is an extrapolation, where a model tries to find the next possible document to form a chain, and the second task is an interpolation where a model looks for missed documents between those two given points.

Traditional storytelling algorithms focus on the last document in a story, one-way or two-way. Attention to the last element of a story chain forces the algorithm to find the next document by considering the last document it found in the previous iteration. In other words, the algorithm only depends on the last document when looking for the next one.

Challenges for Traditional Storytelling Algorithms

- **Noise Sensitivity and Robustness:** One notable problem in traditional algorithms is their vulnerability to noise – a common issue when a storytelling algorithm depends on a last document in the story chain to identify the next document among all potential candidates. If a document in the sequence weakly aligns with the theme of the narrative, it can easily change the story's direction.

- **Comprehensive Topic Coverage:** Another challenge derives from using just one document in the search process to advance the story's narrative and represent the story's multifaceted topics and aspects. A narrative often transits several sub-topics, not all of which may be presented in a single document. These sub-topics are distributed across different documents within the story chain, making it difficult to follow all story facets within the corpus when the algorithm only considers the last document in the story chain.

Implementing distributed attention across the last documents can maintain the narrative's integrity by ensuring continuity with the preceding entities and topics of the story. However, this approach may reduce the ability of a storytelling algorithm to evolve by distributing the focus on the documents collected from previous timestamps. Finding an optimal balance is crucial to avoid compromising either aspect of storytelling in order to maintain narrative consistency without losing the story's evolution over time.

Therefore, any solution to this challenge must maintain a consistent thematic focus and allow for the natural evolution of entities and themes within the story.

Moreover, for a distributed attention component in a storytelling algorithm, it is crucial to consider the unique contribution of each document in the story chain to the main narrative. Stories contain different documents and each influences the narrative to a varying degree. Some documents are pivotal in shaping the storyline, while others might have a lesser impact. Consequently, assigning different weights to each document, reflecting their individual contributions to the narrative, becomes essential. This differentiated weighting ensures that more influential documents impact the direction and development of the story more, while weak documents have a lesser impact. This approach guarantees the reflection of the significance of each narrative component.

The Distributed Attention component within a storytelling algorithm, such as DifStoryGen, is crucial in maintaining narrative's coherence and thematic consistency. This component is explained more in detail in Chapter 5. The main contributions of this component are as follows:

- **Thematic Consistency:** By applying distributed attention as the story progresses over time, DifStoryGen ensures the story remains faithful to the narrative's main theme. This aspect of distributed attention helps to construct a coherent narrative by considering the collective influence of previous documents rather than relying on a single document at a time.

- **Narrative Stability:** The Distributed Attention mechanism is key to keeping the story glued to its main narrative by preventing sudden shifts in the storyline caused by a singular document. This stability is key to maintaining the integrity and continuity of the narrative, ensuring that each document contributes meaningfully to the story's progression.

- **Flexible Adaptation:** This component offers adaptability through a user-defined parameter that controls the range of the last documents when selecting the following document for the story. This flexibility allows for modifying the storytelling process to fit specific applications and tasks, enabling stories to be customized according to the expected level of narrative evolution or consistency.

Together, these contributions of the Distributed Attention component highlight its importance in enhancing storytelling algorithms. This component ensures narrative coherence while also providing the flexibility to adapt the storytelling process to generate stories that are both engaging and faithful to their intended narrative

Chapter 2

Related Work

A storytelling model can be used to analyze events or entities for different applications, such as event forecasting [29, 41], intelligence analysis [38, 79], and knowledge extraction[8]. A storytelling algorithm can also discover relationships between entities, actors, or events [96, 65].These relationships can form a graph in many applications to discover deeper connections between entities. In addition, storytelling can be used as a pre-processing stage to select relevant documents for other downstream tasks.

This chapter reviews state-of-the-art research on storytelling algorithms and different aspects of this dissertation to put its contributions into perspective. The following sections review storytelling algorithms in different domains, including approaches and applications2.1, different approaches to generating embedding space representing words and documents 2.2, and the state-of-the-art text generative models 2.3.

2.1 Storytelling Algorithms

Storytelling forms a coherent chain of documents that belong to a story. These algorithms have been used to enhance news recommendations [30] and search engine outcomes [21, 98] by automatically excluding redundant articles from a story chain to create robust recommendation systems. In addition, storytelling algorithms aid in discovering relationships between entities, information flow among articles, and interactions in social media and news articles [20, 84, 59, 11].

D. Shahaf et al. [77] analyzed methods for automatically connecting documents by providing a structure based on the coherency of a story. The model integrated user feedback, refining and personalizing the generated stories. Hossain et al.[39] designed a storytelling algorithm by

adding a clique constraint into connecting chains to generate more meaningful stories. Some efforts leverage visualization systems to connect the dots to study stories formed from a corpus [40, 23, 41, 76].

Storytelling algorithms have also been applied in the domain of vulnerability analysis, particularly in assessing network risks [91]. For instance, Pascal et al. [61] aimed to automate the evaluation of the Internet of Things (IoT) networks by analyzing the security risks posed by IoT devices and tracking the progression of incidents to understand vulnerabilities better

2.1.1 Probabilistic Approaches

Different studies exist to discover valuable information from large textual data collections. Probabilistic techniques are a common approach to follow a narrative's flow, aiming to connect the dots in a story seamlessly. Most studies suggest algorithms designed to bridge two documents as two fixed points or to begin with a seed document and construct a narrative by gathering other related documents to the story from a corpus.

Several approaches are employed to determine the most efficient path between the start and end of a narrative. Random walks have been widely applied in multiple studies [77, 98, 30, 78, 93] for their effectiveness in navigating through complex narrative structures. Monte Carlo simulations [2] present another probabilistic technique known for its capacity to handle uncertainty and variability in predicting outcomes. The Structured Determinantal Point Processes (SDPPs) [25] technique is used for its ability to model diversity and repulsion, making it suitable for selecting a set of diverse and representative points (or documents) along the narrative path. These methods have collectively contributed to developing storytelling algorithms to construct a narrative by connecting relevant documents within a large corpus.

2.1.2 Optimization-Based Approaches

Identifying a relevant document sequence that tells a coherent story within a large corpus presents a classic search challenge. A common challenge in such search algorithms is the local optimum

problem, where the algorithm picks a local solution that is different from the global solution to that search problem.

D. Shahaf et al. [77] introduced a novel approach using a joint optimization over words and chains instead of assessing many chains along the local search path. This technique aims to mitigate the issue of local optima by considering a broader view of textual content and narrative structure.

Similarly, Camacho et al. [11] explored diffusion to trace entity evolution within seed documents. Their algorithm, focused on historical news articles, aims to pinpoint the origin of an event and highlight key documents that mark its progression over time.

2.1.3 Entity Extraction in Social Media datasets

Keyword extraction is crucial in discovering entity relationships within texts, especially in social media. Studies [96, 16, 45] have demonstrated its utility in revealing complex social dynamics and structuring narrative maps. Toraman et al. [88] introduced a dynamic approach to finding connections in social media news by using a zigzag search that revisits documents to refine the story as new information emerges. Additionally, Zhang's model [97] combines entity features with social network characteristics to automate storytelling in extensive datasets. These studies emphasize the effectiveness of integrating keyword extraction with advanced analytical techniques for narrative construction.

2.1.4 Entity Mining in Big Data

Exploring large information networks is challenging due to processing demands and noises. Si [80] proposed a method to engage audiences with narrative generation, while Chen [16] used keyword evolution in sentences for storytelling in NLP. Rigsby et al. [71] showed that signal injection does not harm model performance and instead helps analysts to concentrate on a story with personal knowledge. Shukla et al. [79] introduced the DISCRN, or distributed spatiotemporal concept search-based model that facilitates spatiotemporal storytelling by linking entities from

microblogs for structured narrative exploration. Koutrika et al. [47] proposed organizing entities on a tree for user-selected document sequences to allow users to select a reading sequence over documents.

2.2 Text Representation

Text representation in NLP involves transforming words and tokens from documents into numerical forms and vectors. This approach lets models capture the context and meaning behind these words and tokens. Different techniques have been developed for this purpose, each targeting different aspects of the linguistic properties embedded in textual data. This section reviews some of these methods and their applications.

2.2.1 Numerical Statistic Representation

Each token in a document contributes to its semantic and contextual meaning. NLP often considers this feature set, a collection of keywords for effectively summarizing the document in a more manageable form.

Term Frequency-Inverse Document Frequency (TF-IDF) is a method that evaluates the importance of a word or token of a document and its frequency across the entire corpus. The idea behind TF-IDF is that words identified by this method are more relevant to the document's content than words in the same document. These selected words offer a concise document representation that provides valuable features for classification and various NLP tasks.

TF-IDF operates by combining the term frequency (the count of a word's occurrence in a document) with the inverse document frequency (a measure of how unique a word is across all documents in the corpus) [42]. A higher inverse document frequency indicates a word's significance within a particular document [33]. This technique is widely applied in document classification to weigh terms effectively and highlight the relevance of specific words or tokens in capturing the essence of a document's content [69].

2.2.2 Word Embedding

Word embedding techniques use vector space models to represent words in a corpus based on their distributional properties, providing fixed-length vectors for each word. Key methods include:

Word2vec, introduced by Mikolov et al. [52, 53], offers static embeddings through two approaches: CBOW, predicting words based on context, and Skip-gram, predicting context from words. These models were enhanced with negative sampling for better performance [54].

GloVe, introduced by Pennington et al. [62], generates embedding vectors by analyzing global word co-occurrence matrices, capturing the semantic relationships between words.

However, these models need more temporal information, which is crucial for studying how the relevance of entities changes over time. Further discussions will explore how to address this limitation in word embedding models.

2.2.3 Dynamic Word Embedding

Over time, the meanings and uses of terms in a language evolve in different periods. Understanding this evolution helps us learn the historical context of terms and predict future shifts [3, 95]. Different studies [7, 7, 70, 73, 85, 94] proposed techniques to extract temporal changes in word meaning and semantic evolution by converting a text corpus into a latent time sequence model.

Mihalcea et al. [51] have utilized parts of speech and contextual information from a text to predict the publication era of words. Other methods [46, 55] represent a corpus as a graph, with words as nodes connected by edges representing their contextual relationships, to analyze how word meanings shift over time. Camacho et al. [22] proposed a diffusion model and tackled the challenge of sparsity in dynamic word embeddings, showing how incorporating temporal dynamics can enrich our understanding of semantic shifts. Moreover, the static representation for words at each timestamp in a corpus can be used to track the semantic evolution of terms in a corpus by employing regression or similar techniques [32, 72, 18]. Tikhomirov and Dobrov [87] used the temporal aspect of a corpus to build a query-oriented timeline summary. Camacho et al. [6] proposed a new time-reflective vector space model that used the concept of *diffusion* to

compensate for the possible sparsity of a dataset.

Recent approaches, including transformer-based models like BERT [17], have introduced novel word and document embedding representations that provide a unique vector representation for each word's appearance. This transformer-based approach allows us to embed each word in a document contextually.

2.3 Generative Models

In the field of generative artificial intelligence (AI), diffusion models are a class of generative models that use a diffusion process to create new samples resembling existing ones in many aspects. Denoising Diffusion Probabilistic Models (DDPMs) introduced by Sohl-Dickstein et al. [81] and extended by Ho et al. [37] are notable for generating realistic samples through forward and reverse processes. Innovations include Bit Diffusion by Chen et al. [15] for generating discrete data, Text2Tex by Zhenyu Chen et al. [13] for high-quality textures in 3D meshes from text prompts, and DiffUTE by Haoxing Chen et al. [14] for self-supervised text editing in images.

Conditional diffusion models, which use encoded input to control the output, have emerged as a significant research area. Diffusion-LM by Xiang Lisa Li et al. [48] exemplifies this with its ability to control complex NLP tasks. Applications extend to text-to-image synthesis, as demonstrated by the Imagen algorithm for photorealistic image generation from text [74], and to sequence-to-sequence text generation tasks, as seen in DiffuSeq [27].

Other noteworthy contributions include Austin et al.'s [5] discrete text diffusion model, the COLD framework by Lianhui et al. [68] for text generation under constraints, DeLorean by Qin et al. [67] for considering both past and future contexts in language models, Donahue et al.'s [19] text infilling approach, and Yang et al.'s [82] stochastic differential equation method for generative modeling. This dissertation uses a conditional diffusion model to connect documents separated by gaps, demonstrating the potential of diffusion models in storytelling algorithms.

Chapter 3

CoRBS: A Contextualization Approach for Documents Using BERT Features

3.1 Introduction

With many timestamped document collections available digitally over the internet, the use of conventional search engines is no longer sufficient for complex analyses of an event [28, 83, 10, 50]. Moreover, a single article may not represent an entire story that developed over a long period of time. For example, the news articles describing Elon Musk's intention for the Twitter acquisition spanned several months, resulting in thousands of documents (news articles, blog posts, and opinion/interview articles). A conventional search engine presents the relevant news articles to a surfer based on the keywords the person is using for the search. Oftentimes, the number of search results is overwhelming, and the user tends to surf only the top few documents, leaving with an incomplete idea about the evolution of a story. Similar trends are seen in the scientific literature review process [86, 75, 66, 64]. A researcher searching for a specific topic in a relevant digital library may end up with thousands of documents to study for a literature review. Another scenario of a conventional search resulting in a massive information overflow is how intelligence analysts attempt to find information from a large number of news articles, field reports, and social media discussions [59, 11, 79, 97]. In this chapter, we present a storytelling algorithm that outlines a story as a chain of documents illustrating the evolution of an event. Such a storytelling algorithm can be used in scientific studies, new search engine development, and intelligence analysis.

While different variations of storytelling algorithms have been developed over the past two decades, none of them is completely immune from sudden drifts of topics in a story. Some lose

the narrative of the stories in intermediate steps, some lose the story's focus, and some stories result in noisy documents. Most techniques have difficult-to-manage parameters to obtain more meaningful chains of articles. Table 3.1(a) shows a chain of news article titles regarding Elon Musk's intention for Twitter acquisition. The chain contains a coherent story flow with relevant documents representing the evolution of the topic. Table 3.1(b) is a story chain that keeps Elon Musk as the topic but fails to realize the relationship between Elon Musk and Twitter in the chain. The story lost the narrative right from the second document in the chain. Table 3.1(c) shows another story chain where the narrative lost the connection between Elon Musk and Twitter in the middle of the chain on May 19th, 2022. Table 3.1(d) is an example of a chain that does not recognize the connection between Elon Musk and Twitter in the entire chain but rather drifts from their interactions in the middle (on July 14). The story ends with a connection between a Twitter employee and a possible spying activity. Given a start document, Table 3.1 demonstrates what a good story should look like, such as Table 3.1(a), and how a story might lose the narrative, such as Table 3.1(b), (c), and (d).

The literature acknowledges that a good story must have the following characteristics [9, 38, 11, 77, 39]: (a) the surroundings of characters and events must evolve from the starting point to the end, while the entire story stays faithful to the main topic [11], (b) the evolution of the story should be smooth without any major leap from one article to another in the story[77, 9], and (c) a brief story does not provide valuable insights, and a long story is not analytically sound for examining the evolution of the entities and the events involved [39, 38]. Achieving these qualities in a story chain formed from text data is challenging in several different ways, which we outline in the following subsection.

3.1.1 Challenges

Challenges associated with a storytelling algorithm are listed below:

1. **Missing context:** In most of the existing storytelling algorithms [77, 39, 38, 11, 40], the semantics of a word are either represented by the co-occurrence of other words [40, 38, 77]

Table 3.1: Four different chains of news articles from the New York Times. These documents are relevant to Elon Musk's acquisition of Twitter. Each of these stories started with the same document published on April 14th and was followed by other documents generating a chain of supposedly connected events. (a) is an example story chain faithful to a specific theme connecting Elon Musk and his Twitter acquisition; (b) realizes Elon Musk but ignores his relationship with Twitter; (c) follows keyword overlaps in consecutive documents drifting to an irrelevant topic, and (d) does not realize the main theme of the starting document drifting to an irrelevant event.

date	a)
14 April 2022	Musk says he has the means to buy Twitter, but investors aren't so sure
14 April 2022	Twitter says it will review Musk's bid for the company.
14 April 2022	Elon Musk wants all of Twitter.
21 April 2022	Elon Musk details his plan to pay for a $46.5 billion takeover of Twitter.
29 April 2022	Elon Musk sells billions in Tesla stock as he prepares to buy Twitter.
4 May 2022	Elon Musk suggests Twitter could charge commercial and government accounts.
5 May 2022	Elon Musk has brought in new investors to fund his Twitter deal, a filling shows.
17 May 2022	Elon Musk says Twitter deal cannot move forward without more information.
8 July 2022	Twitter is ready for a legal battle to force Elon Musk to buy the company.
11 July 2022	twitter's stock falls further as doubts swirl over Elon Musk's takeover.
23 August 2022	Whistle-Blowers spam claims pose challenges for Twitter.
	b)
14 April 2022	Musk says he has the means to buy Twitter, but investors aren't so sure
23 May 2022	SpaceX executive defends Elon Musk against misconduct accusations.
1 June 2022	Elon Musk to workers: spend 40 hours in the office, or else.
7 June 2022	Elon Musk, Mars and the modern economy.
16 June 2022	SpaceX workers write letter to executives with concerns about Elon Musk's twitter
20 July 2022	Tesla profits falls in second quarter as supply chain problems hurt.
25 July 2022	Tesla will spend more to increase production at two new factories.
4 August 2022	Tesla prevails over most activist shareholder proposals
5 August 2022	California regulator accuses Tesla of falsely advertising autopilot
	c)
14 April 2022	Musk says he has the means to buy Twitter, but investors aren't so sure
14 April 2022	Elon Musk, after toying with Twitter, now wants it all.
18 April 2022	The big questions about what happens next in Elon Musk's bid for Twitter.
19 April 2022	Elon Musk race to secure finance for Twitter bid.
19 May 2022	G7 finance ministers race to secure more Ukraine aid
3 July 2022	Australia's new prime minister promises increased aid during visit to Ukraine.
16 August 2022	The secret powers of an Australian prime minister, now revealed.
28 August 2022	referendum seeks to mend the open wound at Australia's heart.
	d)
14 April 2022	Musk says he has the means to buy Twitter, but investors aren't so sure
4 May 2022	Elon Musk suggests Twitter could charge commercial and government accounts.
8 June 2022	Twitter said to agree to give Elon Musk access to stream of tweets.
16 June 2022	Elon Musk tells Twitter's employees he wants the service to contribute to a better.
11 July 2022	Twitter stock falls further as doubts swirl over Elon Musk's takeover.
14 July 2022	Twitter suffers a global outage at a delicate moment for the company.
20 July 2022	Twitter worker accused of spying for Saudi Arabia heads to trial.
21 July 2022	Trail begins for Ex-Twitter employee accused of spying for Saudis.
26 July 2022	Twitter security director says former employees viewed various Saudi accounts.
9 August 2022	Former Twitter employee convicted of charges related to spying for Saudis

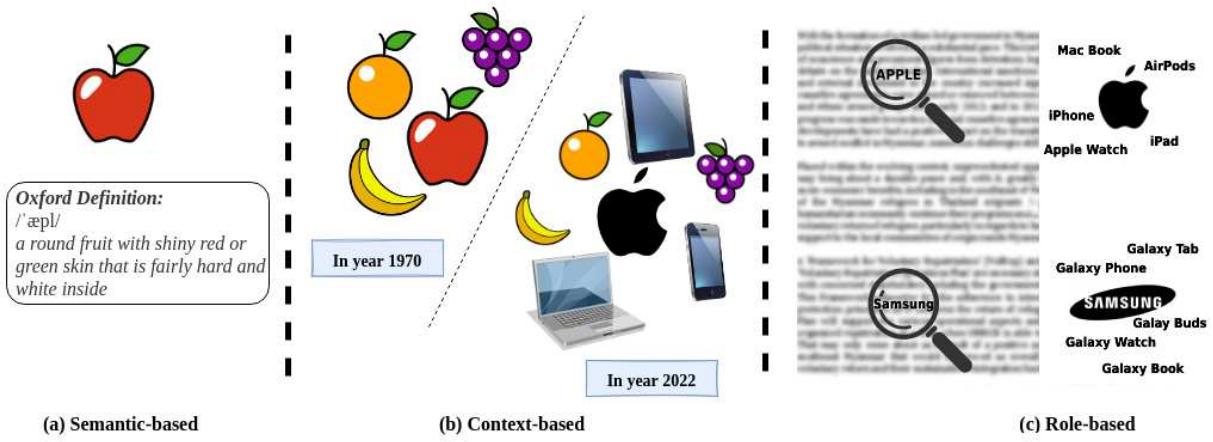

Figure 3.1: In (a), we have the semantic of the word Apple according to the Oxford dictionary. In (b), We have the contextual meaning of Apple in the past and the current time. In (c), we have the different roles of Apple in a text depending on the topic and the focus of the paragraph.

or by retrieved meanings from the dictionary or a knowledge base [39]. The state of the word representations in the literature has moved toward contextual embeddings, which are not present in storytelling algorithms. As a result of primitive non-contextual representations of words, most of the existing storytelling algorithms suffer from losing the main context of the topic in intermediate documents of the story.

2. **Missing evolution:** While some existing storytelling algorithms attempt to capture context to some extent using additional information, such as images in news articles [44], most do not include contemporary context to build on the evolution of the story. The algorithms sometimes rely on out-of-context words to jump from one document to another while forming the evolution of a story. As a result, a sudden leap from one article to another, using poor connections between overlapping words and ignoring the topical evolution, is observed in many stories. Generally, such low-quality stories are removed as a post-processing step, which can be time-consuming.

3. **Missing role of words:** The semantics of a word refer to its meaning and is a global concept (Figure 3.1(a)). The context of a word refers to other words that generally appear close to the word. Whereas before 1970, the word ”apple” was highly contextual with fruits, but

over time, the context of the word *Apple* spread to the technology area, as shown in Figure 3.1(b). A role of a word is localized to exactly when the word is mentioned in the text. It is a distribution of other words seen near a given word. For example, the localized distribution γ_{apple} of the word *Apple* in one paragraph relates to *iPhone*, *iPad*, and *Macbook Pro*, whereas in another paragraph, the localized distribution, γ_{samsung}, of the word *Samsung* relates to *Galaxy phone*, *Galaxy Tab*, and *Galaxy book*. If γ_{apple} and γ_{samsung} are similar, we call *Apple* and *Samsung* to have a similar role in a given document. The role concept is reflected in Figure 3.1(c).

4. **Difficulty in tuning user-settable parameters:** Existing storytelling algorithms require complex user-settable tuning parameters to retrieve coherent stories. It is difficult for users to study those parameters for different datasets, as the optimal values for the parameters may vary between datasets and even for events found in the same dataset.

5. **Missing temporal nature of stories:** Most existing storytelling algorithms [77, 39, 34, 11] are static in nature, ignoring the temporal nature and progression of events through time. While such static storytelling algorithms work well for hypotheses generation in scenting domains [39], event analysis using news data requires additional information and post-processing stages to prune non-evolving stories [44].

3.1.2 Contributions

The contributions of this chapter are listed below.

1. **Contemporary context to capture evolution:** We use Bidirectional Encoder Representations from Transformers (BERT) to capture the contemporary context of every word's appearance and better understand a story's evolution than other existing models. (See section 3.3.1)

2. **Inclusion of roles of words:** We extract the localized distribution of each word for all its appearances in the same document. Based on the distribution of vector-similarities of the

surroundings, we create a localized cluster-based context, which serves as the role of a word. (See the section 3.3.2.)

3. **Lesser number of user-settable parameters:** Unlike previous storytelling models, our proposed algorithm has only one user-settable parameter. The parameter is easy to tweak given that it is the Jaccard similarity threshold, varying between 0 and 1. (See the section 3.3.3.)

4. **Dynamic storytelling:** We propose a storytelling algorithm where the temporal evolution of the story is a combination of the underlying neural network embedding and the search algorithm that drives the story's progression. As a result, the generated stories smoothly progress in time, requiring no post-processing filtering. (See the section 3.3.4.)

3.2 Problem Description

Storytelling refers to an algorithmic framework to select a chain of documents from a corpus, such that the sequence of the documents in the chain represents the evolution of one or more connected events [77, 39, 16, 96, 92, 38, 34, 11, 8].

For a corpus D, a story S is defined as:

$$\exists S \subseteq D, \forall d_i \in S \implies f(d_i, d_{i+1}) > \theta \tag{3.1}$$

where $f()$ is a user-defined function and θ is an acceptable criteria to consider d_i and d_{i+1} in story S a part of the evolution of relevant events.

3.2.1 Temporal aspects of storytelling

A story is dynamic in nature. The entities evolve over time, and the narrative follows them. To generate a story, we need to assure each part of the story follows the contemporary context. Besides that, the algorithm must cover all documents that are related to the story in each timestamp:

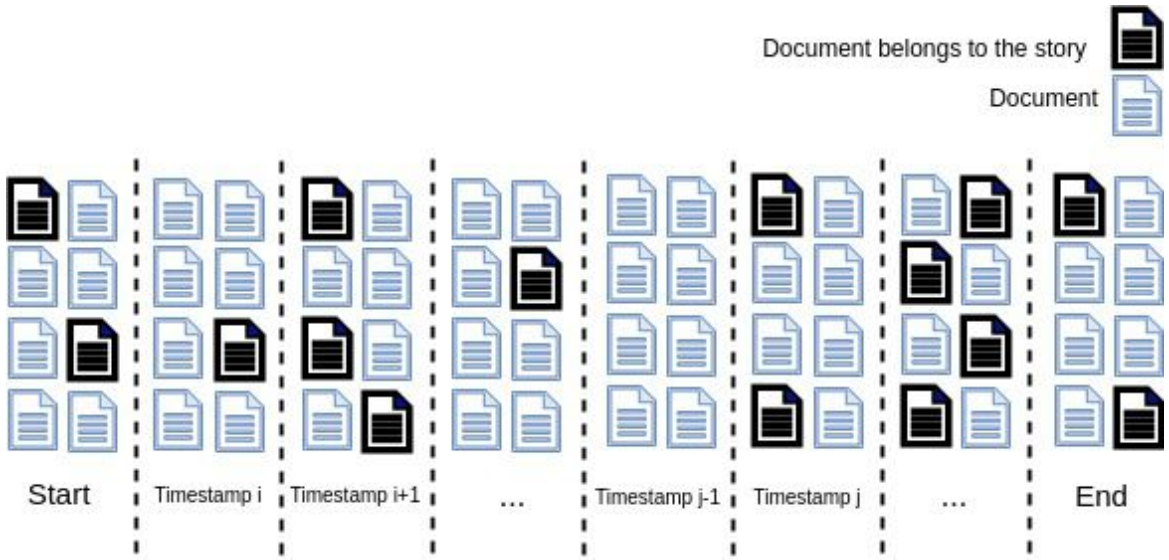

Figure 3.2: A series of documents belonging to a story that are clustered by their timestamps.

$$\left\{d_1^{t_1}, ..., d_i^{t_l}, ..., d_j^{t_k}, ..., d_n^{t_m}\right\} \in S, i < j \Rightarrow t_l \leqslant t_k \tag{3.2}$$

where d_i refers to the ith document in story S and $d_i^{t_l}$ is the representation of the document in timestamp t_l. That is, documents $d_i^{t_1}$ ... $d_n^{t_m}$ form a series of documents belonging to a story S spanning the time range $t_1...t_m$.

A storytelling algorithm can pick zero, one, or more documents from a timestamp (Figure 3.2). The temporal aspect of storytelling is that the timestamps of the documents in a story cannot go back in time but rather only move forward.

3.2.2 The Contemporary contextual aspect of storytelling

Conventional natural language processing techniques generally use the same representation for the same word's multiple appearances. For example, word2vec [52], and GloVe [62] consider global embeddings as the context of a word even though the context of the word can vary between documents and within different paragraphs of the same document. A word's context is localized by its surroundings, and each appearance of the same word may have a different context:

$$\forall m \neq n, \exists w_k^{im}, w_k^{in} \in d_i, \Rightarrow E(w_k^{im}) \neq E(w_k^{in}) \tag{3.3}$$

where E is the context (e.g., embedding vector) for a word w that was generated for document i. m and n are two different appearances of the kth word, w_k, of the vocabulary.

Ignoring the local context and summarizing each word to give it a global context lead to incorrect overlaps of concepts while joining two consecutive documents of a story. To capture contemporary contextual aspects of a story, an algorithm must follow the constraint provided in Equation 3.3.

3.2.3 Role-based aspects of words in storytelling

The context of a word is conventionally represented by words in the neighborhood formed inside the mathematical space of the data, such as the embedding E. A role is different than a context. Two words may have the same role but a different contextual neighborhood. For example, the words *Apple* and *Samsung* have different neighborhoods of keywords, but both words have a similar role (Figure 3.1). The role of a word is computed from the similarity distribution of the words in the neighborhoods (details are in Section 3.3.2). Traditional storytelling algorithms focus on contextual overlaps of consecutive documents. For high-quality story chains, we need to incorporate the role aspect of storytelling so that the story can progress with relevant documents even without a direct match of keywords.

The problem of generating a new representation γ for an appearance of a word in document d_k by incorporating the word's role, r, with its context e is formalized below:

$$\left\{\gamma \in \Gamma : r_i \in R, e_j \in E, F\left(r_i, e_j\right) = \gamma_j^i\right\} \in d_k \tag{3.4}$$

where F() is a function to integrate the context e with role r.

3.3 Methodology

This section proposes an algorithmic framework called Contextual Role-Based Storytelling (CoRBS). Given a corpus and a document of interest, CoRBS generates a chain of documents starting from the given one explaining the evolution of an event, addressing **role** and **contemporary context** issues of existing methods.

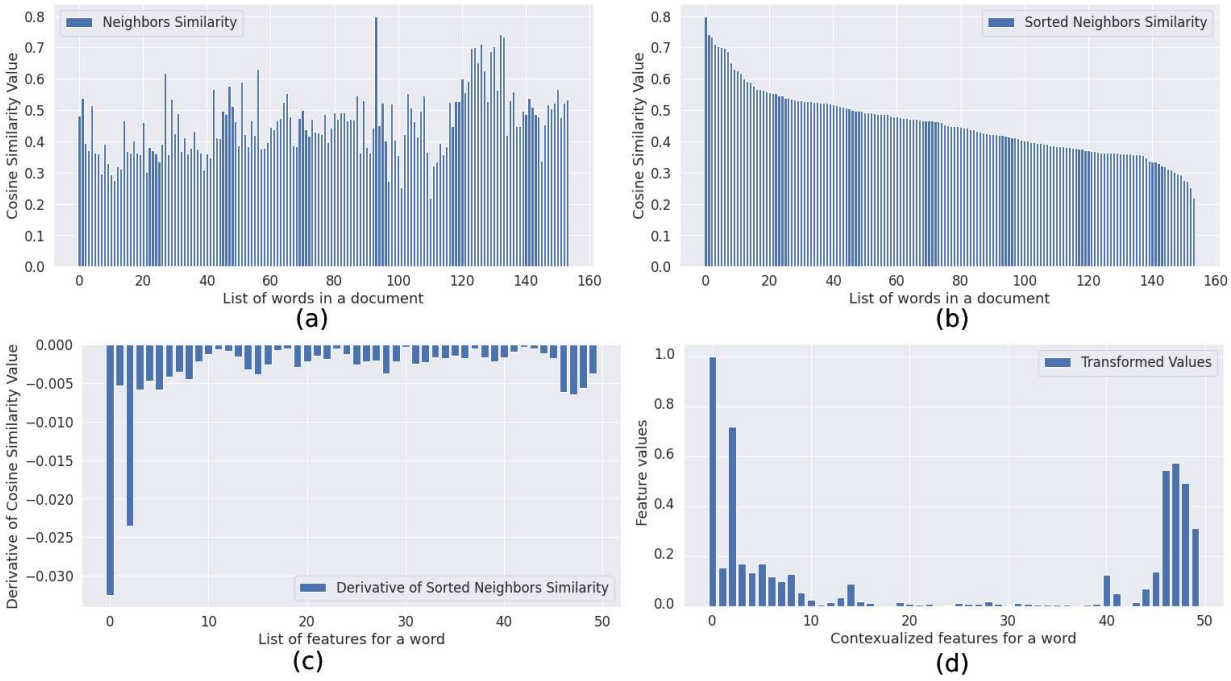

Figure 3.3: The enhancement process to improve contextual features that represent a word in a text.

3.3.1 Generate contemporary context for each word of the corpus

CoRBS stitches together role and contemporary context-driven documents to represent the evolution of the events that appear in the given document. CoRBS uses the Bidirectional Encoder Representations from Transformers (BERT) model as its mechanism to extract contemporary contextual features. BERT generates an embedding vector for each word in a document. That is, a word appearing in two different contexts in the same document may have two different embedding

vectors. We use this unique feature vector to capture the dynamic nature of evolution in our storytelling algorithm.

We leverage BERT's classification node (CLS) to extract the embedding for each document. CoRBS uses document embedding for its similarity search for articles. CoRBS uses the base version of BERT, which has 12 layers of transformer blocks and 12 attention heads. The base version has 110 million parameters and 768 tokens as the input. Each token is a vector of 768 values, which is the embedding vector for that token [17].

3.3.2 Role generation for each word

The role of a word depends on how the nearest neighbors are becoming distant rather than what exact nearest neighbors are there. For example, for the word *Apple*, if the embedding vectors of the three nearest neighbors are losing similarity with *Apple* in δ_1, δ_2, δ_3 rates, and another word *Samsung* also has three nearest neighbors with similarity rate-changes of δ_1, δ_2, δ_3, then we call *Apple* and *Samsung* to be sharing the same role, even though their three nearest neighbors are not exactly the same.

The role generation steps for a given word's appearance in a document are outlined below:

Step 1: For the given word's appearance in the given document, we computed the cosine similarity of all appearances of all words of that document (Figure 3.3a), based on their BERT-based embedding vectors.

Step 2: We organized the words based on descending order of their similarity with the given word's appearance (Figure 3.3b). That is, the similarities form a monotonic distribution.

Step 3: For every three points of the monotonic distribution, we applied linear regression to compute the slope of the descending line of Figure 3.3b, which results in Figure 3.3c. That is, Step 1, Step 2, and Step 3, generate Figure 3.3c, as can be explained using the following formula:

$$DNSD = \frac{\partial}{\partial E} Sorted \left(\left[\forall w_i, w_j \in d_k, \frac{E(w_i) \cdot E(w_j)}{\|E(w_i)\| \, \|E(w_j)\|} \right] [: \mu] \right) \tag{3.5}$$

where DNSD indicates the *Derivative of Neighborhood Similarity Distribution*. μ is the user-

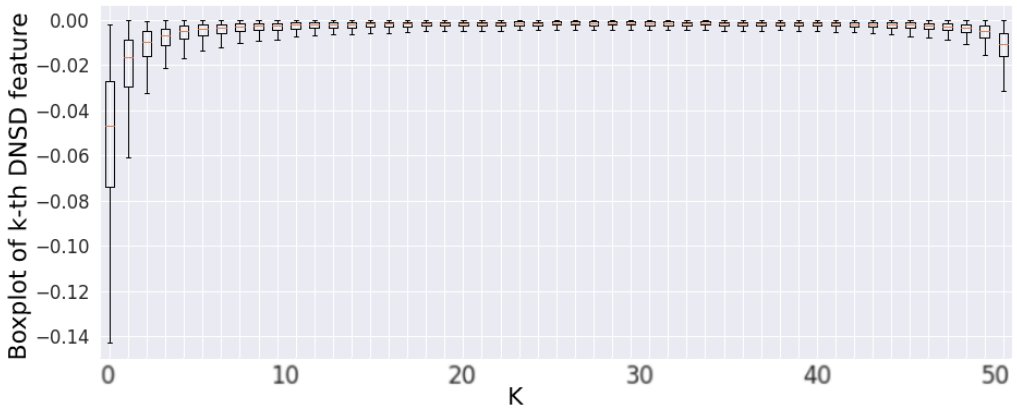

Figure 3.4: Boxplots of DNSD features generated from 1,219 words of 5,700 documents. These documents were selected because they had the same length.

defined maximum number of words in the starting distribution of Figure 3.3(a), which had $\mu = 150$. In the distribution of Figure 3.3c, we compute DNSD with every three consecutive points; as a result, we have a length of 50 (one-third of 150). $E(w_i)$ and $E(w_j)$ are the embedding vectors for words w_i and w_j in document d_k.

Step 4: The DNSD of a word indicates the similarity change of neighboring words in the embedding space, as they are situated in a distant location. The left part of DNSD contains the closest neighbors, which are words contextually connected to the given word. On the other hand, the words in the right part of DNSD are the words that are contextually far away. DNSD represents changes in similarity but not the similarity itself. The similarity is reflected in the ordering alone in DNSD (hence Figure 3.3c is not monotonic, but Figure 3.3b is).

Our empirical study demonstrates that changes in similarity occur more often in the nearest neighbor region and far neighbor region. We selected documents with n words and found that the number of documents in our dataset is the highest when n=147. The 5,700 documents have exactly 147 words. With three consecutive nearest neighbors to compute the DNSD vector, we ended up with a DNSD vector length of 49. Figure 3.4 shows DNSD boxplots for kth DNSD feature of all 1,219 words. It demonstrates that the boxes (i.e., ranges of the kth DNSD feature) are larger in the beginning and tend to become smaller in the middle. In the latter part of the distribution, the boxes again tend to become larger. Taking the difference between the minimum and maximum values

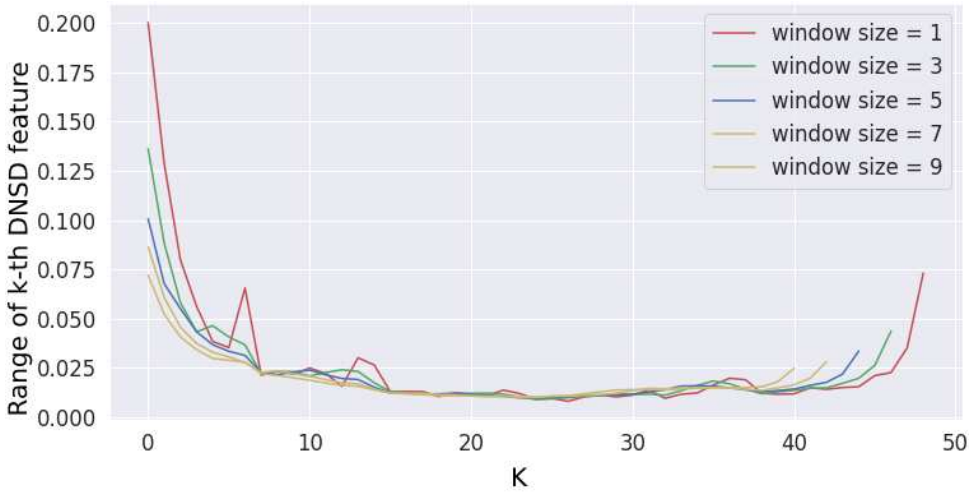

Figure 3.5: The range of DNSD changes distribution values for each feature vector.

of the boxplots, we constructed the red line (window size=1) of Figure 3.5. Moving a running window of different lengths (3, 5, 7, 9) and computing the average within the window, we find the holistic pattern of the DNSD features. The pattern is that variations of changes occur in the closest nearest neighbors and furthest nearest neighbors compared to the ones in the middle.

Motivated by the fact from the empirical study, we smooth the DNSD vector of Equation 3.6 using a funnel-shaped filter to generate the role of a word r_w.

$$r_w = \frac{-1}{1 + e^{-(|DNSD| + \frac{1}{2}\mu)}} - \frac{1}{2} \tag{3.6}$$

Figure 3.3(d) shows the generated role from the DNSD of Figure 3.3(c). Note here that the starting and ending have higher weights in Figure 3.3(d), and the amplitudes are all normalized between 0 to 1, using equation 3.6.

To explain the concept of the *role* of a word's appearance further, we provided an example with four words – *communist*, *party*, *popular*, and *new* – from a document. Figure 3.6 (a) shows the KL-divergence between each pair of words' role vectors' probability distribution (each role vector is normalized such that the sum of the vector becomes 1.0). So far, we have discussed two types of representations for words. For each word that appears in a document, one representation is the contextual representation vector generated by BERT, and the second is the role vector generated

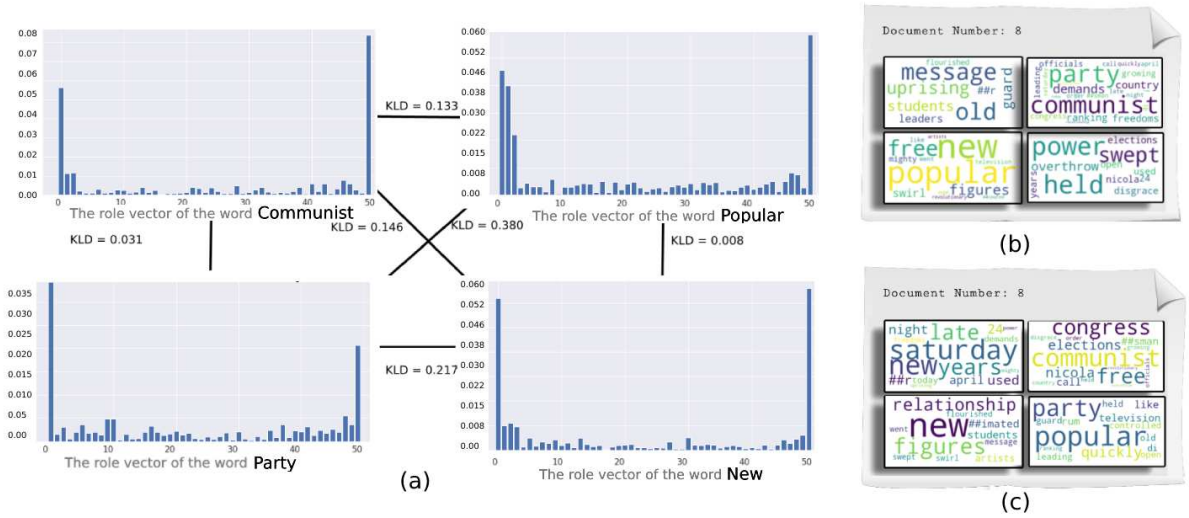

Figure 3.6: the comparison of Kl-divergence values for words Communist, Party, Popular, and New.

by equation 3.6 for each word's appearance. Figure 3.6 (b) shows clusters of words from the document using role-based vectors. Figure 3.6 (c) contains clusters using BERT-provided vectors.

Figure 3.6 (a) shows that pairs of words — (popular, new) and (communist, party) — have low KL-divergence (0.008 and 0.031, respectively) between the corresponding role vectors. This is also reflected in Figure 3.6 (b), where the pair (popular, new) is in the same cluster. Also, the pair of words (communist, party) are in the same group. The other pairs – (communist, popular), (communist, new), (new, party), and (popular, party) – exhibiting higher KL-divergence (0.133, 0.146, 0.217, and 0.380, respectively) are in different clusters. Therefore, the role vector representation seems to reflect in a clustering algorithm properly.

The roles and contexts of words might not always form the same cluster because contextual vectors focus on the nearest neighborhood of a word to form a context, whereas our role-based representation considers a holistic role of the word's appearance in the entire neighborhood of the document both at near and far distances, and how sharply or slowly the neighbors are drifting apart. Figure 3.6 (c) shows clusters formed by the BERT-generated contextual vectors. The pair (popular, party) is in the same cluster of Figure 3.6 (c), even though the words are in different clusters in their roles, as shown in Figure 3.6 (b). The pair (communist, party) is in the same

cluster of the role-based cluster in Figure 3.6 (b), but the words are in two different clusters in the context-based representation space of the document shown in Figure 3.6 (c). The role-based and context-based representations may not always disagree. For example, the pair of words (free, party) are in two different clusters in both Figure 3.6 (b) and (c).

Contextual representation is biased toward the neighborhood, sometimes resulting in findings that do not directly have a meaningful perspective. For example, the words *free* and *communist*, even though appearing in the neighborhood in the text, resulting in solid connectivity between them in the contextual space, they are known to have opposing semantics. That is, their roles might be different. The observation is evident in Figure 3.6, where the pair of words (free, communist) are in the same cluster of contextual groups (Figure 3.6 (c)) but in different role-based clusters in Figure 3.6 (b). Our role-based representation complements contextual representations by distinguishing such relationships automatically.

3.3.3 Generation of role-based document representation

For the appearance of each word in a document, we now have role vectors. In this section, we outline how we can represent a document using the roles of the words in that document. We use the role vectors of the appearances of all words to form groups of roles. Application of any clustering technique could provide us with a general sense of the role groups in a document. However, automating how many strong groups of roles there are in a document is tricky. To automate the process of generating a role-based representation, we apply k-means clustering on the role vectors of a document with different k, varying from 2 to k'. If a cluster of words, C, remains almost the same with two runs of the k-means clustering algorithm, with $k = i$ and $k = i + 1$, then C is considered to be a strong group of roles that did not break even when k was increased. Varying the number of clusters for the k-means clustering algorithm, from $k = 2$ to k', allows us to find the strongest groups of words in a document in terms of the role vectors of the words.

Figure. 3.8 demonstrates how we vary $k = 2$ to 6. Based on a group-similarity threshold, we chose groups of words that do not change much, with an increased number of clusters.

$$RoleGroupSim(C_i, C_j) = 1.0 - \frac{min\left(\left[\sum (M_{C_i} - M_i) \circ (M_{C_i} - M_i)\right]^{\circ\frac{1}{2}}, \left[\sum (M_{C_j} - M_j) \circ (M_{C_j} - M_j)\right]^{\circ\frac{1}{2}}\right)}{max\left(\left[\sum (M_{C_i} - M_i) \circ (M_{C_i} - M_i)\right]^{\circ\frac{1}{2}}, \left[\sum (M_{C_j} - M_j) \circ (M_{C_j} - M_j)\right]^{\circ\frac{1}{2}}\right)} \tag{3.7}$$

Figure 3.7: Weighted Jaccard index between two clusters formed using role-vectors of words in a document.

To compute the role-based similarity between two groups of words, let us assume that M_{C_i} contains all the role vectors of cluster C_i and a centroid matrix M_i has the center of cluster C_i copied in each row, where the number of rows and columns in both matrices M_{C_i} and M_i are equal. Equation 3.7 in Figure 3.7 demonstrates how we compute the role-based similarity between two groups of words, C_i and C_j. We use the distance between the role vector of a word and the center of the group of the word as the feature value of the word in that group. $\circ$ in Equation 3.7 is a Hadamard product, and $\circ\frac{1}{2}$ refers to the square-rooting of every element of a vector.

To determine which clusters to keep in the representation of a document, we computed the role-based group similarity ($RoleGroupSim$) between every two clusters generated using $k = i$ and $k = i + 1$ for the k-means clustering algorithm.

A document in our data contains around 150 to 200 words. In our observation, setting $k' = 6$ is sufficient to form groups of roles from all documents. Moreover, having $k' = 6$ may generate as many as 20 groups (2+3+4+5+6) to represent a document. Figure 3.9 demonstrates how role-based group similarity thresholds may impact the number of selected groups of words in a document for different k, varying between 4 to 8. With k=6, with low role-based similarity requirements, such as 0.1, on average, there are 15 groups of words per document. With a more stringent threshold, such as 0.9, the number of surviving clusters is less than 5.

With lower k, the number of selected role groups per document will be lower. With a larger k, the number of selected role groups per document will be more. In a document containing around 150 to 200 words, our observation is that there can be six or seven paragraphs containing different roles. Setting K=6, and the $RoleGroupSim$ threshold to 0.7, on average, gives us around seven role groups per document.

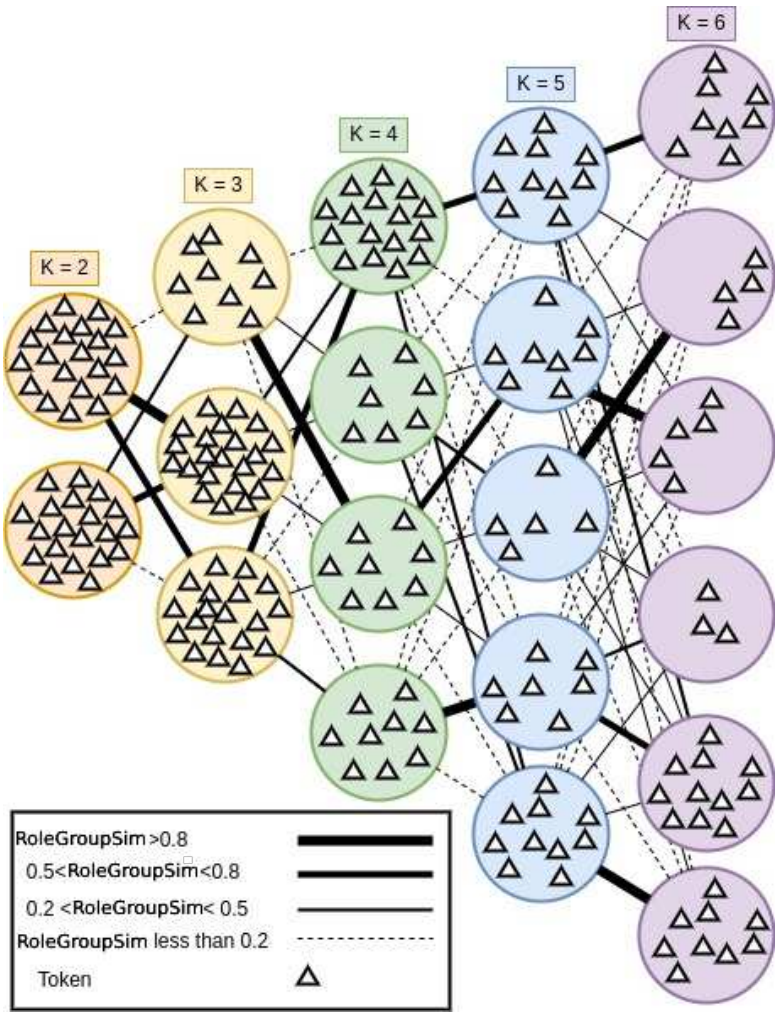

Figure 3.8: The transition of tokens over different clustering. K-clusters of tokens k=2,3,..,6 for a document.

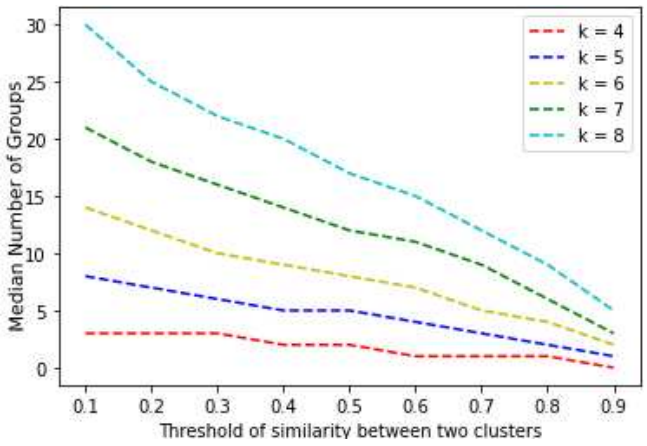

Figure 3.9: The median of different numbers of group representation of documents for different values of the threshold similarity between two clusters.

A text collection containing larger documents with many different topics requires a larger k to capture a sufficient number of roles on average to represent the document of that collection.

$$RoleDocSim(d_k, d_l) = \frac{1}{|C^{d_k}||C^{d_l}|} \sum_{i \in d_k} \sum_{j \in d_l} RoleGroupSim(C_i^{d_k}, C_j^{d_l}) \tag{3.8}$$

A role-group is given a higher weight when it appears in multiple k and k+1 clusterings.

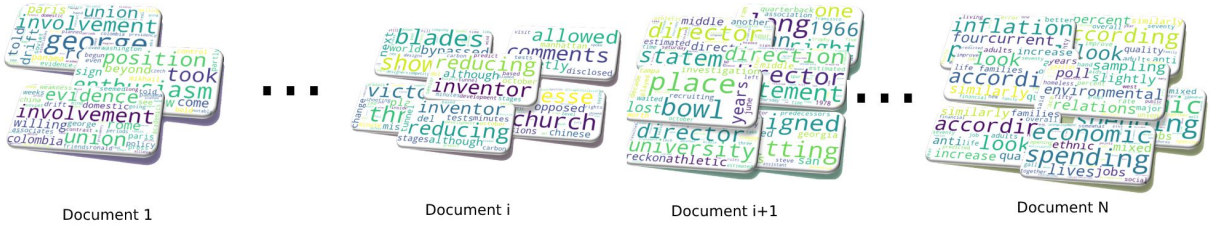

Figure 3.10: Text's representation using clusters of words for each document.

3.3.4 Finding the best narrative

Equations 5.1 and 3.2 of Section 3.2 define a story as a chain of documents with an acceptable similarity between each consecutive pair of documents, where the chain monotonically moves forward in time. To allow an acceptable similarity between two consecutive documents, so far,

we have two types of similarity: contextual similarity (from BERT) and role-based similarity (equation 3.8). The acceptable similarity can be a combination of both.

Give an initial document d_{seed}, we generated a story based on an algorithm, $StoryGen$ (Algorithm 2), that takes both contextual and role-based similarities into account with α and β weights, respectively. The time span T is divided into n time groups, and each time group is composed of m consecutive time stamps. As outlined in Algorithm 2, $StoryGen$ starts from d_{seed} and iterates over groups of timestamps to construct a story.

The for loop in line 12 of Algorithm 2 iterates over the time groups. Within each time group T_j, which is representative of a short time span, the algorithm attempts to find a set of consecutive documents satisfying equations 5.1 and 3.2. θ_{min} and θ_{max} in Line 16 of the algorithm ensure an acceptable similarity range between consecutive documents in the story. The function *docMax* on Line 22 of Algorithm 2 returns the document from the $CandidateList$ that has the maximum similarity with document $d_{selected}$.

If a time group does not have any acceptable connecting documents, the algorithm moves forward to the next time group. This happens in real-world stories where a topic spikes for a few days, then dies down, and then again reappears after some time. Our $StoryGen$ algorithm is able to capture such scenarios. Figure 3.2 shows that a story can contain multiple documents from a time group and might not pick any document at all from another group.

We allowed $StoryGen$ to be open-ended, unlike a destination-based story generator [77, 39, 11, 44, 40], to ensure that the stories are a natural flow of connected events. Our objective is to construct natural open-ended stories that are not used for hypothesis generation where a forceful directional flow of a story is required. $StoryGen$ uses temporal shifts (unlike the other approaches), and contextual and role-based similarities as its basis for open-ended exploration. As a result, $StoryGen$ is a suitable tool for news event analysis.

Algorithm 1 $StoryGen(D, d_{seed}, M_{role}, M_{emb}, T, \alpha, \beta, \theta)$

D is the document collection, $\{d_1, .., d_k\}$.

d_{seed} is the initial document for the story.

M_{role} is the role similarity matrix for all pairs of documents.

M_{emb} is the contextual similarity (embedding-based) matrix for all pairs of documents.

T contains the time groups: $T_1, .., T_n$.

α is the weight coefficient for role-based similarity.

β is the weight coefficient for contextual-based similarity.

θ is a range between θ_{min} and θ_{max}.

$M_{Sim} = \alpha M_{role} + \beta M_{emb}$

$d_{selected} \leftarrow d_{seed}$

$Story \leftarrow \phi$

for $j = 1$ to n **do**

    $T_j \leftarrow j^{th}$ time group from T containing m timestamps

    while t_j not in $T_j[-1]$ **do**

        for $i = 1$ to m **do**

            if $\theta_{min} < M_{sim}[d^{t_i}, d_{selected}] < \theta_{max}$ **then**

                $CandidateList \leftarrow d^{t_j}$

            end if

            for s in $Story$ **do**

                $CandidateList.remove(s)$

            end for

            $d_{selected} \leftarrow docMax(M_{sim}[d_{selected}, CadidateList])$

            $Story.append(d_{selected})$

        end for

    end while

end for

3.3.5 Time and space complexity

The proposed algorithm has multiple stages. In the preprocessing step, which is mostly composed of contextual vector generation using BERT, the time complexity is $O(n^2)$. The reason behind the quadratic time complexity of the vector generation is that the transformer-based model we leveraged was a pre-trained model; therefore, the training from scratch was not involved in our process. The number of transformer layers was 12 in BERT, and the embedding length of the vectors was 768.

The role-based document representation is a combination of computing derivatives, generating role-vectors for each appearance of every word in a given corpus and finally forming the document representation. Computing derivatives is a linear process over documents and a quadratic process over the words of each document. Since each document is small, the quadratic complexity over words per document results in a feasible computation. The computation of role vectors (DNSD) is a linear process over the generated derivatives. A role-based document representation uses groups of role vectors to represent each document. In our approach, we applied k-means clustering with k varying from 2 to k' for role vectors of each document, which results in the complexity of the k-means clustering algorithm, $O(n^2)$, over the words of each document. All of these steps are one-time computations for the entire storytelling system and can be considered amortized establishment costs.

The search component of the algorithm is the only part that changes given a seed document. Theoretically, a breadth-first-search variant has a time complexity of $O(b^d)$, where b is the branching factor, and d is the depth of the search. Our $StoryGen$ algorithm (Algorithm 2) has a feasible computation given that its branches are heuristically contained by timestamp groups and a combination of role and contextual similarity. The search behaves like a polynomial one as if it is operating over a tree, making $StoryGen$ a suitable algorithm for storytelling.

3.4 Conclusions

This chapter introduces an algorithmic framework named Contextual Role-Based Storytelling (CoRBS). CoRBS is designed to construct a sequence of documents from a starting point named seed from a corpus, enabling models to trace the evolution of events and entities. This approach addresses the challenges related to understanding the role of terms and their contemporary context, which are often missed by conventional storytelling algorithms.

The role of a term in a document is defined by analyzing the distribution of similarities among its nearest neighbors using BERT contextual embedding vectors for all words in a document. Contemporary contexts are incorporated to ensure a proper chain of documents as the story develops to ensure a narrative's progression remains coherent.

Chapter 4

Experimental Analysis: CoRBS

4.1 Introduction

In this chapter, we discuss the experimental analysis of the CoRBS model using *the New York Times* dataset containing 5,700 articles covering a wide range of topics, including politics, sports, entertainment, economics, and medicine. This dataset is used to evaluate the different challenges highlighted in the problem description from the previous chapter 3.

This chapter is structured into the following sections: Evaluation Mechanism4.2, Experimental Results 4.3, and Conclusion 4.4, providing a comprehensive overview of the CoRBS model's effectiveness and performance.

4.2 Evaluation Mechanism

Usually, generated stories are evaluated based on inherent objectives of the storytelling algorithm [77, 11]. Based on our storytelling algorithm, our evaluation mechanism is driven by four core metrics: dispersion coefficient, temporal continuity of the story, story stretch, and story evolution. The four metrics are described in the following subsections.

4.2.1 Dispersion coefficient

Hossain et al. [38] introduced the concept of the dispersion coefficient in evaluating a storytelling algorithm. The dispersion coefficient reflects whether two consecutive story documents have sufficient overlaps. Additionally, the dispersion coefficient metric penalizes if non-consecutive

documents of the story have overlapped. A dispersion coefficient value of 1.0 indicates that only each consecutive pair of documents has certain overlaps in the story. The score reduces as non-consecutive documents start to have content overlaps. A value of 0.0 indicates that no consecutive pair of documents overlaps, but all other non-consecutive pairs overlap the content of a certain threshold.

The dispersion coefficient for a story, $\{d_0, d_1, ..., d_{n-1}\}$, is given by the following formula.

$$\psi = 1 - \frac{1}{n-2} \sum_{i=0}^{n-3} \sum_{j=i+2}^{n-1} disp(d_i, d_j) \tag{4.1}$$

Where

$$disp(d_i, d_j) = \begin{cases} \frac{1}{n+i-j}, & \text{if normdist}(d_i, d_j) < \eta. \\ 0, & \text{otherwise.} \end{cases} \tag{4.2}$$

η is an acceptable distance threshold for the evaluation. For computing the $normdist$, one can use Soergel distance, cosine dissimilarity, or any other normalized dissimilarity between two documents d_i and d_j. d_i and d_j are vector representations of two documents. We used BERT-generated contextual vectors for all methods in our experiments for a fair comparison.

Note that the dispersion coefficient evaluates a story based on content or context overlaps but does not check if the story progresses over time. The next evaluation metric assesses the temporal continuity of a story.

4.2.2 Temporal coverage coefficient, $TempoCover$, for stories

While the dispersion coefficient helps in measuring content overlaps in the story chain, whether the documents in the story cover different timestamps is not reflected in the dispersion coefficient. To evaluate the temporal coverage, we use a story-length normalized value of the number of timestamps covered by each story of a set. In equation 4.3, the function $Cover(s_i, t_j)$ returns 1 if the story s_i has a document in timestamp t_j. Otherwise, the function returns 0.

$$Cover(s_i, t_j) = \begin{cases} 1, & \text{if } \exists\ d \in s_i\ \&\ d \in t_j\ . \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

Equation 4.4 provides the length-normalized temporal coverage coefficient, $TempoCover$, for a set of m stories, S.

$$TempoCover(S) = \frac{1}{m} \sum_{i=1}^{m} \frac{\sum_{j=1}^{n} Cover(s_i, t_j)}{|s_i|} \tag{4.4}$$

$TempoCover$ becomes 1.0 when all the stories in the given set of stories S have an equal number of timestamps covered, and each document of a story appears in a unique timestamp. $TempoCover$ tends to 0.0 when each story in set S covers close to one timestamp only.

4.2.3 Evaluation of the stretch of stories, $StoryStretch$

Short stories are not eventful in the sense that they express sudden events. Our objective is to find stories that evolve over time. In contrast, long stories are difficult to analyze and might contain obvious relationships in the chain of documents available throughout the timeline. For a set of stories generated from random seed documents, shorter and longer stories should be less than stories of average length.

To measure if stories are homogeneously stretched, we use a property named $StoryStretch$ for a set of stories, S.

$$StoryStretch(S) = 1 - \frac{\sqrt{\sum_{i=1}^{|S|}(l_i - \bar{L})^2}}{|S|\bar{L}} \tag{4.5}$$

l_i is the number of total documents in $s_i \in S$, and $\bar{L}$ is the average length of all the stories in S. If S contains a set of stories with the same length, then $StoryStretch(S) = 1$. With an increasing number of shorter and longer stories, $StoryStretch(S)$ tends to become smaller.

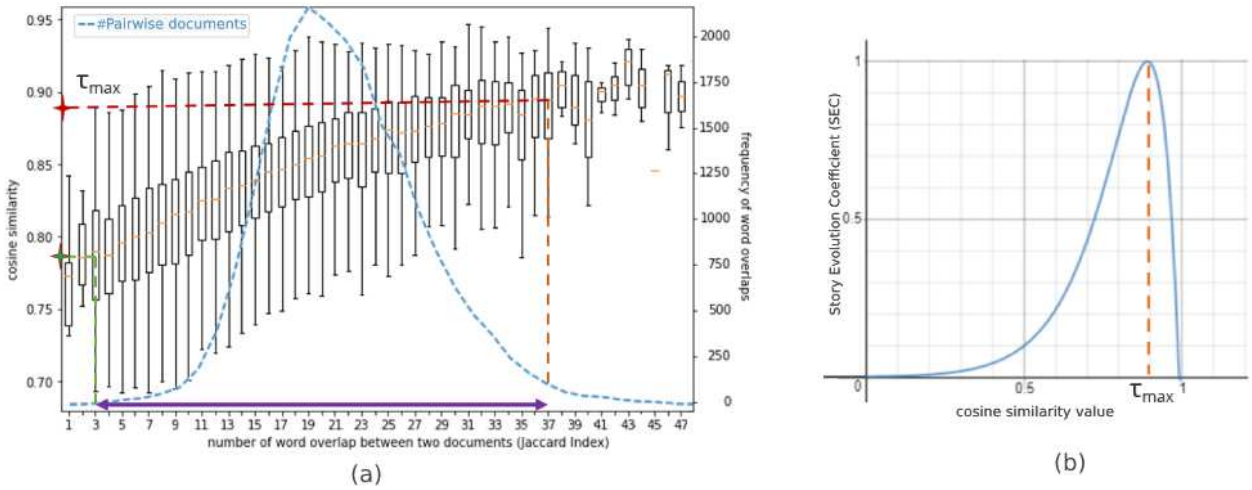

Figure 4.1: (a) Number of terms overlapping vs. cosine similarity between pairs of documents. A boxplot of contextual cosine similarities is shown at each term overlap. The blue dashed line shows how many pairs of documents are found with a certain number of term overlaps. (b) The functional distribution of Story Evaluation Coefficient (SEC of Equation 6.3) with respect to cosine similarity values.

$$SEC(S) = \frac{1}{(n-1)|S|} \sum_{s=1}^{|S|} \sum_{i=1}^{n-1} (\varphi\tau_{max} - \varphi Cosine(E(d_i^s), E(d_{i+1}^s)) + \rho) \times exp(-\varphi(\tau_{max} - Cosine(E(d_i^s), E(d_{i+1}^s)))) \tag{4.6}$$

4.2.4 Story evolution coefficient (SEC)

To evaluate the evolution of a story, we need to examine two items.

- Do consecutive documents in a story have enough overlap? If two consecutive documents in a story do not have sufficient overlap, then the story is broken and does not reflect evolution.
- Do consecutive documents in a story exhibit enough change to foster evolution? If consecutive documents have too much similar content, the change might not be sufficient to construct an evolving story.

To reflect these two criteria in an evaluation metric, we design equation 6.3, which we call *Story Evolution Coefficient (SEC)*. The definition of *enough overlap* and *enough change* between two consecutive documents of a story can be set as a function of the similarities between their contextual vectors. SEC should be the highest for a certain similarity, before and after which the function should decay sharply to penalize lower or higher similarities. The function's peak is τ_{max}. τ_{max} and determined empirically, where random pairs of documents have reasonable overlap in terms of content and contextual vector similarity.

Figure 6.1(a) shows the number of terms (words) overlapping vs. cosine similarity between pairs of documents. A boxplot of contextual cosine similarities is shown at each term overlap. The blue dashed line shows how many pairs of documents are found with a certain number of term overlaps. This experiment was done by randomly selecting 200 documents, where each document had at least one word in common with another document on this list. The number of pairs from this list is 9,900. Our observation is that with an overlap of one word, the median cosine similarity between the contextual vectors is around 0.79. The median cosine similarities between the contextual vectors tend to become higher with more overlaps of terms as we go from

left to right of Figure 6.1(a). The dashed blue line shows that pairs of documents with around 19 words in common are most frequent. Note here that the frequency of these pairs forms a Gaussian distribution.

The maximum word overlap is 48, and only around 1% of the pairs have more than 37-word overlaps. Based on this empirical study, we consider that more than 37-word overlaps can be considered as *evolution being stuck* and less than three overlaps being *not enough content overlap*. 37-word overlaps are equivalent to a median contextual similarity of 0.89, as shown by τ_{max} in Figure 6.1(a).

SEC is an evaluation metric that should be the highest when the contextual similarity between two consecutive documents in a story is τ_{max}. Figure 6.1(b) shows the function we design in Equation 6.3 to compute SEC for a set of stories.

In Equation 6.3, S is a set of stories, and e_i is the contextual embedding vector for document i. ρ is the parameter to set the scale of the metric SEC from zero to a desired maximum value. We set ρ to = 1.0 to vary SEC within the range of $[0, 1]$. SEC would vary between 0 to 2 with $\rho = 1.7$, and 0 to 10 with $\rho = 3.3$. The choice depends on analytic needs. In our experiments later, we use the range $[0, 1]$ for SEC.

φ is a parameter to tune the metric for changing the slope of the curve from zero to τ_{max}. In our case, $\varphi = 10$ gives a reasonable rise to the maximum value of SEC. Greater values for φ make SEC sharper (the width of the bell shape reduces).

4.3 Experimental Results

We performed some experiments using the *New York Times* articles dataset [1]. The dataset contains more than 140,000 articles about different subjects, including *politics, sports, entertainment, economics, medicine*. Each article has a title, publication date, author name, and text. We used more than 5700 articles from this dataset in our experiments. We split articles into 84 different time intervals based on their publication date. We randomly selected 20 different articles from the first timestamp as a seed set to generate separate stories based on them. For this dataset, we

applied some restrictions, such as the size of an article must be 200 to 500 words, and all special characters and links must be removed from the text.

We used different evaluation metrics to examine the performance of our proposed model compared with three other methods:

1. a storytelling algorithm based on the Doc2Vec contextual embedding model.

2. a storytelling algorithm based on the document embedding in BERT. This document embedding is extracted from the classification token in the last layer of BERT.

3. a storytelling algorithm based on the entity overlap between two documents. This algorithm uses the Jaccard Index to calculate the overlap between pairs of documents to select the maximum overlap to generate the narrative of the story (entity-set-based method).

We seek to answer the following questions:

- How well does the proposed model select the relevant documents over irrelevant documents in a search algorithm? We used the *Dispersion coefficient* to calculate this aspect of a story.

- How well does the story generated by the proposed model distribute over different time spans? We used the *TempoCover* coefficient to measure the temporal coverage aspect of a story.

- How well does the proposed model generate stories that are homogeneously stretched? We used the *StoryStretch* metric to generate the normal length of chain documents.

- How well does the proposed model reflect the evolution of a story? We use the *SEC* metric to measure the quality of the evolution in a story.

The source code of the project is available on this GitHub repository: `https://github.com/AlirezaPNouri/CoRBS`

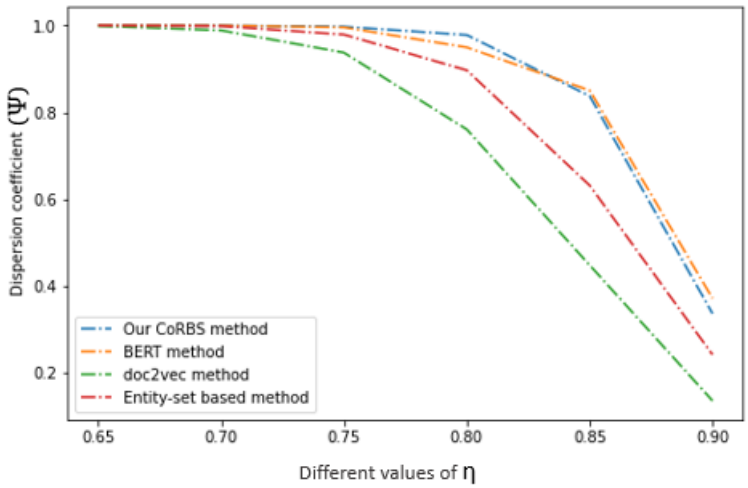

Figure 4.2: Dispersion coefficient for different values of $normdist$ requirements η. We set θ_{max} for StoryGen to a high value of 0.9 for this experiment.

4.3.1 Evaluation of local content overlap in a story using dispersion coefficient

The dispersion coefficient (Equation 4.1) provides a measure of how a story carries content from its start to the end. With the content overlap between only consecutive pairs of documents of a story chain, the dispersion coefficient is 1. With overlaps between non-consecutive documents, the dispersion coefficient becomes smaller. As stated in the literature on storytelling algorithms [39, 38], stories of higher quality should exhibit a higher dispersion coefficient.

In this subsection, we evaluate four different models for the same storytelling framework (Algorithm 2) for a fair comparison. The models are Doc2Vec, entity-set representation, BERT, and our proposed CoRBS-based representation that combines role-based and contextualized vector representations.

The earliest timestamp (early January 2018) of our data contains 93 documents. Using each of these 93 documents as a seed document, Figure 4.2 demonstrates the average dispersion coefficient of all the stories with different $normdist$ requirements η. Our CoRBS model and BERT have similar dispersion coefficient values, whereas the use of Doc2vec and the direct entity-set-based models exhibit lesser average dispersion coefficients. This indicates that storytelling using the

CoRBS and BERT models results in better stories in terms of the dispersion coefficient. That is, the stories generated with CoRBS and BERT have evolving content as they progress over a chain of documents.

4.3.2 Temporal coverage coefficient (*TempoCover*)

Temporal coverage coefficient, *TempoCover*, evaluates a set of stories based on how unique each story in the set is to the timestamps covered and how much all the stories agree in terms of coverage length. A set of stories that is well-distributed over time has a higher value of *TempoCover*. A story with many documents covering a limited number of timestamps accompanied by similar stories exhibits a smaller value of *TempoCover*.

A comparison of the temporal coverage coefficient of four models, including our CoRBS model, is shown in Figure 4.3. Our CoRBS model has the highest TempoCover value among models with different maximum similarity thresholds (θ_{max} values) of the StoryGen algorithm (Algorithm 2).

This value does not change when we increase the maximum similarity filter to pick the candidate document in the *StoryGen* algorithm. Notice here that some of the models exhibit a straight-line *TempoCover* after a certain similarity threshold. This occurs because, after a certain θ_{max}, there were no pair of documents with a cosine similarity greater than θ_{max}, leading to the same set of stories for the rest of the thresholds provided in the experiment. For Doc2vec, this started with a small θ_{max} value of around 0.12. With the entity-set-based representation, the set of stories (and hence *TempoCover*) kept changing until around 0.9.

Overall, regardless of the values of θ_{max}, stories generated with our CoRBS model provided the highest *TempoCover* compared to those generated with all the other three models.

4.3.3 Evaluation of the stretch of stories (*StoryStretch*)

StoryStretch (Equation 4.5) measures the homogeneity of the lengths of stories. Stories that are shorter than the average length of a set of stories are likely to lack evolution and represent sudden

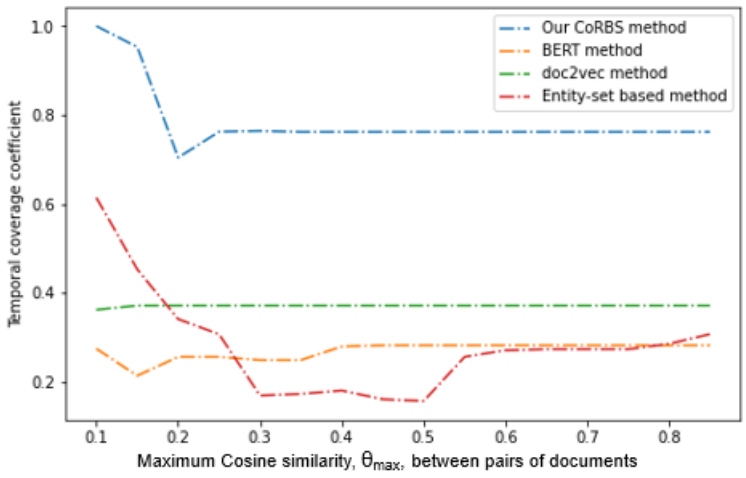

Figure 4.3: Temporal Coverage Coefficient for different maximum similarities, θ_{max}, in the storytelling algorithm.

events. Stories that are longer than the average length of a set of stories might have superfluous and obvious words connecting the events. A higher *StoryStretch* of a set of stories indicates that most of the stories in the set have lengths closer to the average length of the stories.

Figure 4.4 demonstrates that CoRBS-driven stories exhibit higher *StoryStretch* with higher maximum similarity thresholds (θ_{max} values greater than 0.28) of the StoryGen algorithm (Algorithm 2), compared to stories driven by the other three models: BERT, Doc2vec, and entity-set-based representations. The allowable maximum similarity, θ_{max}, is generally set higher rather than lower to ensure enough contextual match between consecutive documents. Therefore, it is essential that a model performs better with higher θ_{max} values. Compared to other models, our CoRBS-driven stories have better homogeneous lengths with higher θ_{max} values.

Similar to *TempoCover*, *StoryStretch* also does not change after certain θ_{max} values, for some methods, and for reasons explained in the previous subsection (Subsection 4.3.2)

As shown in Figure 4.4 the most changes of StoryStretch vs. Maximum Cosine similarity appear on the range of the densest embedding area for each model.

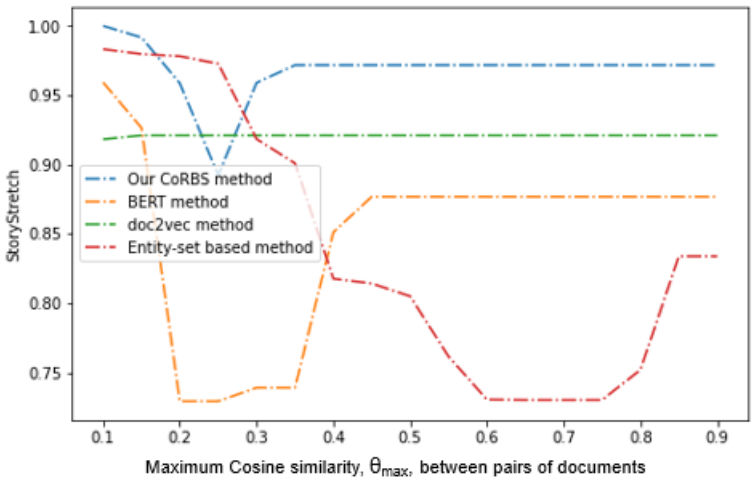

Figure 4.4: Story stretch coefficient for different ranges of maximum similarity in storytelling algorithm.

4.3.4 Story evolution coefficient *(SEC)*

Story Evolution Coefficient *(SEC)* (Equation 6.3) is a measure to evaluate a set of stories in terms of how much overlap consecutive documents have in a story and how much changes are propagating from one document to another in each chain. That is, evolution is considered a combination of preservation and modification of some concepts. A set of stories with high SEC value represents chains that have evolving concepts. We evaluate different models provided with the StoryGen algorithm (Algorithm 2) using SEC to discover which model provides the most evolving stories.

Figure 4.5 shows boxplots for stories generated with the same seeds using four models: CoRBS, BERT, Doc2Vec, and entity-set-based representations. CoRBS provides the highest SEC (the box's median line is higher than any other medians of the other methods). The thinnest boxplot is seen for CoRBS, which indicates that the variance of SEC for CoRBS is lower than any other model. Doc2Vec exhibits the lowest SEC, indicating that the generated stories driven by the Doc2Vec model are less evolving. Our CoRBS model is superior in terms of SEC compared to the three other methods, as observed in Figure 4.5.

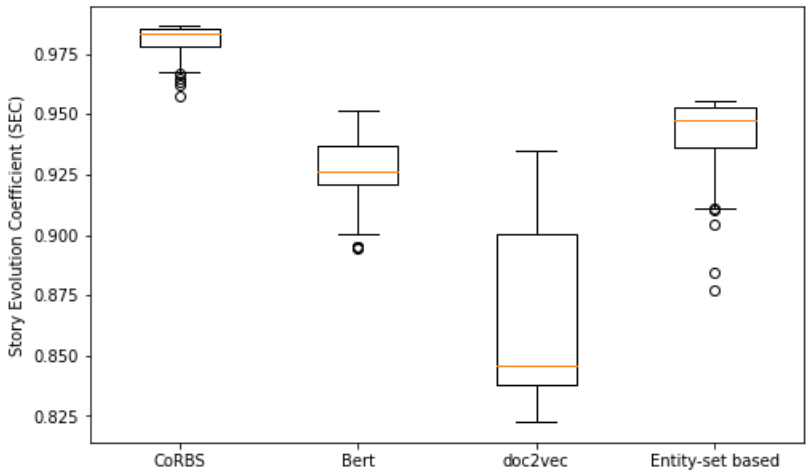

Figure 4.5: Story Evaluation coefficient boxplots for different models using $\theta_{max} = 0.9$ for StoryGen.

4.3.5 Model comparison Summary

In Table 4.1, we provide the average SEC, Dispersion, StoryStretch, and TempoCoverage of all the methods. We allowed *StoryGen* to use a wide range of similarities [θ_{min}, θ_{max}] of [0.001, 0.9] for consecutive documents. Table 4.1 shows that our model, CoRBS, provides the best (highest) average evaluation scores. All the models were provided with the same seed documents to generate stories using the *StoryGen* algorithm.

Table 4.1: Comparison between the CoRBS, Doc2Vec,Entity-set based model, and BERT.

Models	SEC	Dispersion	StoryStretch	TempoCoverage
Doc2Vec	0.86	0.76	0.92	0.37
Entity-set based	0.94	0.89	0.83	0.31
BERT	0.93	0.94	0.88	0.28
CoRBS	**0.98**	**0.97**	**0.97**	**0.76**

4.3.6 Case study

In this case study, as a seed document, we selected an article that was on the front-page news for a while in early 2018. It is a peripheral event associated with President Trump's election campaign and possible obstruction of justice. The peripheral event is about a Republican memo claiming that the FBI surveilled Carter Page, the foreign policy advisor of President Trump's election campaign. The allegation was that the FBI used information collected by a former British spy, Christopher Steele, to request the court for a surveillance warrant against Carter Page in 2016. The news had different actors and episodes over a time range, making it a suitable event for analyzing results generated by different storytelling models.

All the storytelling models we used for this case study contained the same StoryGen algorithm 2 to form a story, given the seed document. All the models are tuned to the same parameters, which are minimum acceptable contextual similarity, maximum acceptable contextual similarity, and minimum acceptable entity overlap in each pair of consecutive documents. Having different document representations is the only distinction among these models. The CoRBS model uses contextual and role-based embedding representations, while BERT and Doc2Vec use their native embeddings, and the entity-based model uses a weighted TF-IDF representation for documents.

Figure 4.6 shows TF-IDF-based word clouds of the stories generated by each method for the same seed news article (Document ID 24058). Corresponding article links, publication dates, and titles for stories generated by CoRBS, BERT, Doc2Vec, and entity-based models are provided in Tables 4.2, 4.4, 4.3, and 4.5.

As can be seen from the word clouds for the story documents of all methods (Figure 4.6) and the respective Tables, CoRBS provides the longest story, spanning a wider time range (January 29 to February 26 of 2018), compared to all other methods. CoRBS sets the chain of documents in such a way that it connects the event of the use of British Intelligence in approval of surveillance for the Russia inquiry, relevant Mueller investigation, democratic rebuttal in support of the Russia investigation, possible influence on the Justice Department, and Trump's reaction toward Barack Obama administration as well as the Attorney General.

Compared to other baseline models, the CoRBS model performs better by selecting all the relevant articles related to the seed document. Remarkably, the CoRBS model correctly includes article 263962, titled ”The Real Aim of the Nunes Memo Is the Mueller Investigation,” which none of the other models consider relevant to the story. Furthermore, the CoRBS model effectively continues the story by incorporating article 221778, titled ”Trump Attacks Obama, and His Own Attorney General, Over Russia Inquiry,” article 252336, titled ”5 Takeaways From the Release of the Democratic Memo,” and article 241855 titled ”2 Weeks After Trump Blocked It, Democrats Rebuttal of G.O.P. Memo Is Released.” In contrast, other models cease progression at this stage. The CoRBS model's ability to utilize both role-based and contextual-based embeddings allows it to identify and capture the significance of various roles played by entities such as ”President Trump,” ”the attorney general,” ”Russian interference,” ”Justice Department,” and more. This comprehensive approach enhances the storytelling model's capacity to distinguish and incorporate relevant information within the story.

The BERT model (Table 4.4) creates a chain that starts on January 29 but ends quickly by February 4, 2018. This chain is present almost entirely in the CoRBS model. BERT missed the connection of the story with the democratic rebuttal in support of the Russia investigation and the connection of the story with the Justice Department. Since BERT is not role-based, it relies on contextual similarity and might miss significantly important connections if the content does not bring words into the contextual neighborhood.

Despite its initial success, the BERT model falls short of maintaining the story's coherence. It mistakenly selects article 261196, titled ”How Trump's Allies Fanned an Ember of Controversy Into Flames of Outrage,” as the final article, disregarding other important entities mentioned in subsequent articles such as article 226858, titled ”Senate Letter Echoes House Republicans' Accusations of Bias”, article 262729, titled ”No. 3 Official at the Justice Department Is Stepping Down,”, article 262821, titled ”Pressure From Trump May Lead to Revision of Democratic Memo”, article 221778, titled ”Trump Attacks Obama, and His Own Attorney General, Over Russia Inquiry”, article 252336, titled ”5 Takeaways From the Release of the Democratic Memo,” and article 241855, titled ”2 Weeks After Trump Blocked It, Democrats' Rebuttal of G.O.P.

Memo Is Released." The BERT model fails to recognize the connection between the first and final chapters of the story, primarily because it ignores the roles played by entities such as the "House Intelligence Committee," "F.B.I.," "Justice Department," and "Trump campaign" in articles 261196 and 266389. This limitation in entity role identification contributes to the model's inability to track the story's narrative flow effectively.

The Doc2Vec model (Table 4.3) was able to move up to February 11, 2018, but could not bring Trump's reaction toward the Barack Obama administration and the Attorney General during Trump's own administration.

The entity-based model was only able to include one document in the story after the seed document and failed to generate a coherent chain because the method is content similarity-based. Such a chain might not have content similarity; rather, context similarity, along with the role, is the key to generating meaningful stories.

Due to its static embedding nature, the Doc2Vec model fails to encompass all documents about the story, such as articles 263962, 285583, 221778, 252336, and 241855. This limitation arises from the model's inability to establish contextual connections between entities within the story. While the model maintains narrative continuity by leveraging semantic relationships found through static embedding, it overlooks related documents. Doc2Vec struggles to recognize the similarity between a person and a title in a news article, as exemplified by the instances of "Stephen E. Boyd" and "The Republican chairman of the committee" in article 268651. In contrast, contextual-based models like BERT and CoRBS effectively identify them as the same entity due to their ability to extract richer connection information from the document.

Tables 4.6, 4.7, and 4.8 show the average of Contextual Embedding, Entity overlap Role similarity for all stories generated by different models.

The entity-based model encountered difficulties identifying the subsequent document following article 254266, "House Republicans Vote to Release Secret Memo on Russia Inquiry," in the story news. The potential candidates for continuing the story were article 232340, "F.B.I. Condemns Push to Release Secret Republican Memo," and article 263962, "The Real Aim of the Nunes Memo Is the Mueller Investigation." However, despite article 254266 containing more frequent

mentions of entities such as Russia, Republicans, and the committee compared to article 232340, the model failed to recognize the latter as a viable candidate. This suggests that the model's reliance on entity frequency alone led to an erroneous decision. Furthermore, the entity-based model struggled to establish the semantic connections between certain words, such as Trump, President, and he. Unlike contextual-based models that recognize these words as different forms of the same entity, the entity-based model failed to identify their similarity. This limitation further restricted its ability to follow the story chain appropriately. In summary, the entity-based model's shortcomings in both entity frequency analysis and semantic understanding have impeded its performance in determining the next document in the storyline.

Our observation regarding stories is that consecutive pairs of documents have similar average similarity using all methods, whereas CoRBS has the highest standard deviation. Also, CoRBS generates stories that have a trail of documents with more deviation from the seed compared to other methods. Table 4.6 demonstrates that the highest standard deviation of average similarity, both for consecutive documents and with seed, is obtained by CoRBS, indicating a better evolution compared to other methods. Additionally, CoRBS generates longer stories compared to other methods. Table 4.6 contains averages of a total of 18 stories with different seed documents.

Tables 4.7 and 4.8 provide similar information as Table 4.6 but for entity overlaps and role-vector similarities of consecutive documents in each story and similarities with the seed. Among all models, the standard deviation is the highest with CoRBS. The tables demonstrate that even though StoryGen attempts to find similar documents fairly for all models, the models are responsible for picking up documents that provide better evolution. CoRBS provides better evolution in terms of embedding similarity, content overlap, and role.

Another note is that CoRBS provides high-quality similarity as well as better evolution, which is difficult to maintain for longer stories. Yet, CoRBS provides longer stories compared to other models.

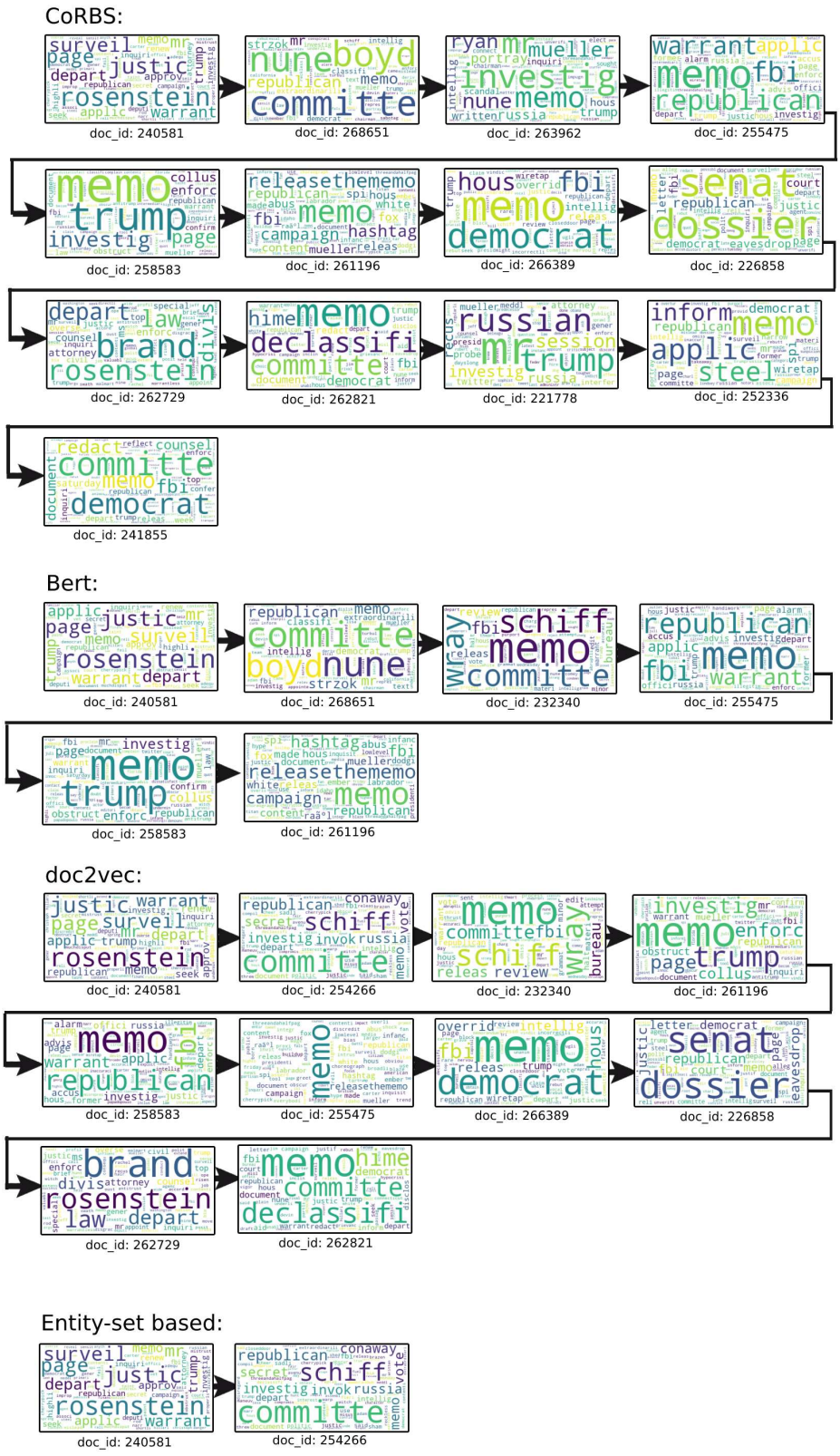

Figure 4.6: The word cloud of a story generated by CoRBS, BERT, doc2vec, and Entity-set based models from seed document 240581.

4.4 Conclusion

This chapter describes a novel algorithmic framework to generate a chain of documents forming a storyline given a start document from a corpus. This framework uses embedding vectors generated by a bidirectional transformer to address issues in relevant legacy models. The localized role-driven context of each word helps the framework bring latent connections into the scenario, in addition to the semantics of a token. Experimental results demonstrate that our approach extracts evolving stories that smoothly change from some events and actors to others over time, outperforming baseline techniques.

Table 4.2: Stories generated from seed document 240581 by CoRBS model.

Document ID	Publish date	Title of the document
240581	Jan 29 2018	Secret Memo Hints at a New Republican Target: Rod Rosenstein
268651	Jan 30 2018	F.B.I. Texts and Dueling Memos Escalate Fight Over Russia Inquiry
263962	Feb 01 2018	The Real Aim of the Nunes Memo Is the Mueller Investigation
255475	Feb 03 2018	House Republicans Release Secret Memo Accusing Russia Investigators of Bias
258583	Feb 04 2018	Trump Says Republican Memo Totally Vindicates Him
261196	Feb 04 2018	How Trump s Allies Fanned an Ember of Controversy Into Flames of Outrage
266389	Feb 06 2018	Committee Votes to Release Democratic Rebuttal to G.O.P. Russia Memo
226858	Feb 07 2018	Senate Letter Echoes House Republicans Accusations of Bias
262729	Feb 10 2018	No. 3 Official at the Justice Department Is Stepping Down
262821	Feb 11 2018	Pressure From Trump May Lead to Revision of Democratic Memo
221778	Feb 22 2018	Trump Attacks Obama, and His Own Attorney General, Over Russia Inquiry
252336	Feb 26 2018	5 Takeaways From the Release of the Democratic Memo
241855	Feb 26 2018	2 Weeks After Trump Blocked It, Democrats Rebuttal of G.O.P. Memo Is Released

Table 4.3: Stories generated from seed document 240581 by doc2vec model.

Document ID	Publish date	Title of the document
240581	Jan 29 2018	Secret Memo Hints at a New Republican Target: Rod Rosenstein
254266	Jan 30 2018	House Republicans Vote to Release Secret Memo on Russia Inquiry
232340	Feb 02 2018	F.B.I. Condemns Push to Release Secret Republican Memo
255475	Feb 03 2018	House Republicans Release Secret Memo Accusing Russia Investigators of Bias
261196	Feb 04 2018	How Trump s Allies Fanned an Ember of Controversy Into Flames of Outrage
258583	Feb 04 2018	Trump Says Republican Memo Totally Vindicates Him
266389	Feb 06 2018	Committee Votes to Release Democratic Rebuttal to G.O.P. Russia Memo
226858	Feb 07 2018	Senate Letter Echoes House Republicans Accusations of Bias
262729	Feb 10 2018	No. 3 Official at the Justice Department Is Stepping Down
262821	Feb 11 2018	Pressure From Trump May Lead to Revision of Democratic Memo

Table 4.4: Stories generated from seed document 240581 by Bert model.

Document ID	Publish date	Title of the document
240581	Jan 29 2018	Secret Memo Hints at a New Republican Target: Rod Rosenstein
268651	Jan 30 2018	F.B.I. Texts and Dueling Memos Escalate Fight Over Russia Inquiry
232340	Feb 02 2018	F.B.I. Condemns Push to Release Secret Republican Memo
255475	Feb 03 2018	House Republicans Release Secret Memo Accusing Russia Investigators of Bias
258583	Feb 04 2018	Trump Says Republican Memo Totally Vindicates Him
261196	Feb 04 2018	How Trump s Allies Fanned an Ember of Controversy Into Flames of Outrage

Table 4.5: Stories generated from seed document 240581 by Entity-set based model.

Document ID	Publish date	Title of the document
240581	Jan 29 2018	Secret Memo Hints at a New Republican Target: Rod Rosenstein
254266	Jan 30 2018	House Republicans Vote to Release Secret Memo on Russia Inquiry

Table 4.6: Average and Standard Deviation of similarities between embedding vectors of consecutive pairs of documents and between the seed and other documents of a story.

Model	Avg. Length	Avg. Sim.	Std. dev. of Sim.	Avg. Sim. with seed	Std. dev. of Sim. with seed
CoRBS	**4.6**	0.88	**0.012**	0.88	**0.015**
Bert	2.8	0.88	0.009	0.87	0.009
doc2vec	2.5	0.87	0.010	0.87	0.007
Entity-set based	1.3	0.87	0.002	0.87	0.003

Table 4.7: Average and Standard Deviation of entity overlaps of consecutive pairs of documents and between the seed and other documents of a story.

Model	Avg. Length	Avg. Sim.	Std. dev. of Sim.	Avg. Sim. with seed	Std. dev. of Sim. with seed
CoRBS	**4.6**	0.11	**0.023**	0.11	**0.023**
Bert	2.8	0.11	0.018	0.10	0.014
doc2vec	2.5	0.10	0.014	0.10	0.011
Entity-set based	1.3	0.10	0.001	0.11	0.003

Table 4.8: Average and Standard Deviation of similarities between role-vectors of consecutive pairs of documents and between the seed and other documents of a story.

Model	Avg. Length	Avg. Sim.	Std. dev. of Sim.	Avg. Sim. with seed	Std. dev. of Sim. with seed
CoRBS	**4.6**	0.73	**0.150**	0.31	**0.13**
Bert	2.8	**0.74**	0.092	0.30	0.10
doc2vec	2.5	0.69	0.076	0.32	0.06
Entity-set based	1.3	0.66	0.003	**0.34**	0.01

Chapter 5

DifStoryGen: Diffusion-Based Storytelling Algorithm with Distributed Attention

5.1 Introduction

As news outlets and scientific archives composed of text data grow, analyzing events and entities in a collection of timestamped documents by conventional search engines falls short [28, 83, 10, 50]. A singular article often fails to represent the knowledge that has accumulated over the past or explain how an event evolved. Accessing the history of an event is critical in understanding the root and, as such, making decisions in consideration of the past contexts of the event. A storytelling algorithm is a method designed to create chains of consecutive documents using a corpus representing the evolution of an event or topic. Storytelling algorithms aim to generate cohesive and coherent stories, usually by stitching together events and entities from documents with timestamps. They are used in various domains, including search algorithms [21, 98], recommendation systems [30, 90, 12], vulnerability analysis [91, 61], intelligence analysis[40, 38], entity relations discovery [20, 84, 59, 11], and text summarization [23, 41, 76, 24, 70, 89].

The effectiveness of a storytelling algorithm depends on the interpretability of the generated sequence. Clearly, a story that fails to track an event's evolution is of little to no use. Also, a story that changes topics drastically is misleading. Figure 5.1 (a) shows an example where the story chain starts from the Russia-Ukraine War, changes the topic to the vaccination of astronauts, and then ends with Elon Musk's appearance on a TV show on Mother's Day. This story was generated based on entity overlaps between consecutive pairs of documents. While the contents of each pair of consecutive documents had entity overlaps, the overall story changed the topics so drastically

(a) Entity overlap-based storytelling

Date	Title of Article	Entities Overlap	Topic Overlap
March 30 2021	Fighting Escalates in Eastern Ukraine, Signaling the End to Another Cease-Fire.		
April 15 2021	U.S. Imposes Stiff Sanctions on Russia, Blaming It for Major Hacking Operation.	Russia, Ukraine, War	Russia-Ukraine War
April 21 2021	Putin outlines a populist economic policy for the post pandemic era.	Russia, Sanction, Economy	——
April 23 2021	Now launching to the space station: Vaccinated astronauts.	Pandemic, Vaccine, Putin	Pandemic
May 2 2021	SpaceX Makes First Nighttime Splashdown With Astronauts Since 1968.	Space, Astronauts, Vaccine	——
May 9 2021	Elon Musk Hosts a Mother's Day Episode of 'Saturday Night Live'.	SpaceX, Astronauts, Elon Musk	——

(b) Semantic overlap-based storytelling

Date	Title of Article		
March 30 2021	Fighting Escalates in Eastern Ukraine, Signaling the End to Another Cease-Fire.		
April 9 2021	Russian Troop Movements and Talk of Intervention Cause Jitters in Ukraine.	Russia, Ukraine, Fighting	Russia-Ukraine War
April 19 2021	A Quiet Arms Race Is Rapidly Heating Up Between the Two Koreas.	——	War
April 22 2021	The U.S. War in Afghanistan: How It Started, and How It Ended.	——	War
April 30 2021	The British Army's Legacy in Iraq and Afghanistan.	Afghanistan	War

(c) Our method, DifStoryGen

Date	Title of Article		
March 30 2021	Fighting Escalates in Eastern Ukraine, Signaling the End to Another Cease-Fire.		
April 9 2021	Russian Troop Movements and Talk of Intervention Cause Jitters in Ukraine.	Russia, Ukraine, Fighting	Russia-Ukraine War
April 16 2021	In Russia, a Military Buildup That Can't Be Missed.	Russia, Ukraine, War	Russia-Ukraine War
April 21 2021	Ukraine's president warns that war may be coming, and vows that his nation will stand 'to the last man.'	Russia, Ukraine, Military	Russia-Ukraine War
May 8 2021	Where Ukrainians Are Preparing for All-Out War With Russia.	Russia, Ukraine, War	Russia-Ukraine War

Figure 5.1: Stories generated by three different storytelling algorithms: (a) Entity overlap-based storytelling algorithm, (b) Semantic overlap-based storytelling algorithm, and (c) Our method, DifStoryGen. Entity overlap and topic overlap between each consecutive document are shown on the right side. The words with the highest TF-IDF in the title are colored for each news article.

that it lost coherence. The story in Figure 5.1 (b) started with the same document as Figure 5.1 (a) and used a semantic technique to curate a cohesive series of documents. The chain of documents covers the concept of war but with different regions. From the Russia-Ukraine War, the story shifted to the conflicts between the two Koreas in East Asia, then to the crisis in Afghanistan, and finally ended with an article covering some entities and events in Afghanistan and Iraq. While the story in (b) is more coherent than the story in (a) in covering wars across the globe, the temporal shifts are not topically connected. Finally, the story in Figure 5.1 (c) consistently adheres to its central theme, the Russia-Ukraine War, while allowing the entities to evolve over time. The story is generated by our proposed DifStoryGen, which is a Distributed Attention-based storytelling algorithm. In addition to remaining on topic, the discovered storyline of Figure 5.1 (c) covered the main characters and events, allowing the evolution of various incidents over time. This chapter focuses on a method that constructs coherent stories, as presented in Figure 5.1 (c).

Different literature explains that a good storytelling algorithm should possess the following characteristics:

1. The narrative context, including characters and events, must progressively evolve from start to end while remaining loyal to the topic [11],

2. The storyline should transition smoothly, avoiding any drastic shifts between consecutive documents in a story [77, 9], and

3. The narrative length must be sufficient to provide valuable insights. A brief story lacks depth, and a lengthy one may compromise clarity in tracing the evolution of events and entities [39, 38].

5.1.1 Challenges

The challenges faced by storytelling algorithms include:

- **Missing Context:** Existing storytelling algorithms often rely on static, non-contextual representations of words based on co-occurrence or definitions from a knowledge base. This

approach leads to losing the main topic's context in the middle of the narrative.

- **Missing Content:** Many storytelling algorithms lack an effective mechanism to bridge gaps in the text of document collections. This issue either halts the search process or shifts into unrelated narratives due to missing documents.

- **Complexity in Parameter Tuning:** The need for a detailed set of user-defined parameters to control the narrative makes conventional storytelling algorithms biased towards user preference. Each dataset requires its specific settings, adding to the complexity and potential impartiality of the algorithm.

- **Incorporating Historical Context:** Most storytelling algorithms use only the last document in the story chain for the search process, missing the influence of earlier documents selected in the story. This issue, most of the time, leads to a sudden shift in the narrative, distanced from the main flow of a story.

- **Consistency of Main Entities and Concepts:** Storytelling algorithms that focus on keywords often distract from the story's core concept. On the other hand, those concentrating on the main concept may shift focus between groups of entities, losing the narrative's main characters.

5.1.2 Contributions

Narratives generated by our proposed storytelling algorithm achieve these attributes by constructing a framework of generative models and machine learning algorithms. The main contributions of the chapter are:

- DifStoryGen employs Bidirectional Encoder Representations from Transformers (BERT) to capture the temporal context of each document. This approach significantly enhances the ability to track the evolution of entities in a story, offering excellent performance in capturing narrative evolution compared to other existing storytelling models.

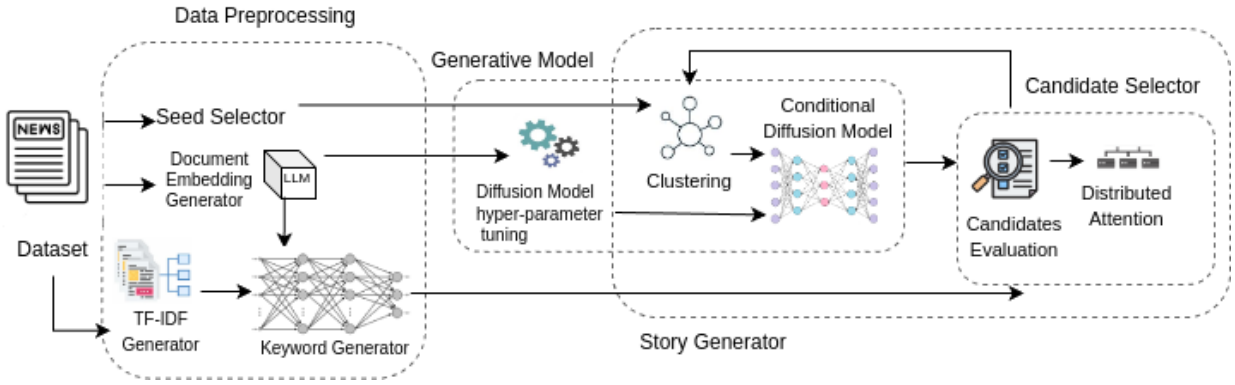

Figure 5.2: DifStoryGen consists of four vital modules: 1) the data preprocessing unit, 2) a generative model, 3) the story generator, and (4) the candidate selector.

- DifStoryGen helps a human analyst by generating intermediate documents, using a diffusion-based document generation model, to connect the dots between two articles in the absence of relevant supporting documents in the corpus. This helps discover stories with re-appearing themes.

- Given the involvement of generated documents to identify the next real document, the temporal shift or the definition of evolution need not be set by the analyst, reducing critical user-settable parameters and enhancing the model's ease of use.

- DifStoryGen uses distributed attention while temporally progressing during story construction to ensure the story's overall theme. The distributed attention helps the model generate a coherent story (as illustrated in Figure 5.1 (c)).

- To ensure a smooth transition between two consecutive candidate documents in a story, DifStoryGen relies on two types of similarities: (a) embedding similarity and (b) overlap of generated keywords for two consecutive candidate documents. Keywords are generated for each document based on an encoder-decoder model between embedding and words, rather than on the words appearing in the documents.

5.2 Problem Description

Storytelling is the task of creating a sequence of documents from a corpus, such that the ordering of the documents in the chain represents the evolution of events and entities in that story [77, 39, 16, 96, 92, 38, 34, 11, 8].

For a document collection D, a story S is defined as:

$$\exists S \subseteq D,\ s.t.\ \forall d_i \in S \implies f(d_i, d_{i+1}) > \theta,\ s.t.\ T(d_i) \leqslant T(d_{i+1}) \tag{5.1}$$

where $f()$ is a user-specified function and θ is an acceptable criteria to consider to consecutive documents, d_i and d_{i+1}, connected in story S. $T(d_i)$ is the timestamp of document d_i.

Earlier storytelling formulations in the literature [77, 39, 38, 11] did not include the timestamps in the definition of a story as a part of the story generation algorithm, instead considering pruning approaches to select stories that follow a reasonable timeline.

5.3 Methodology

The pipeline of the DifStoryGen framework is illustrated in Figure 5.2. DifStoryGen starts with extensive preprocessing involving document embedding generation, TF-IDF computation, and preparing a keyword generator neural network from embeddings. The story generator involves (1) a diffusion-based generative model to generate a new document in a coming timestamp, (2) a clustering component that is used as a condition of the generative model, and (3) a candidate document selector to find the next most suitable document, if any. The following subsections explain the components of DifStoryGen.

5.3.1 Data preprocessing

Before involving the core of the DifStoryGen model, the dataset goes through a series of processing: (a) generate an embedding vector e_d for each document d, using a pre-trained large language model $E(d)$, BERT, in this chapter, (b) to fit the embeddings in compact and reasonable memory

space, construct an encoder-decoder arrangement of layers to reduce the size of the embeddings, (c) compute a TF-IDF vector, $TF(d)$ for each document d, and (d) train a neural network to generate TF-IDF vector $TF(d)$ given an embedding vector e_d of document d.

The neural network to generate TF-IDF from embedding vectors is later used in the core of DifStoryGen to realize prominent words of a generated embedding vector where no original document exists.

Our method incorporates contextual document embeddings to capture a document's subtle semantic and syntactic differences. It leverages the Bidirectional Encoder Representations from Transformers (BERT) framework to generate a contextual representation for each document. BERT is adept at generating a comprehensive embedding vector for each word and provides a distinct e_{cls} vector from the final layers of its encoders, encapsulating the entire document. Our storytelling algorithm uses this e_{cls} vector to represent each document contextually.

$$\begin{aligned} &d = \{w_1, ..., w_n\} \rightarrow E(d_i) = (e_{cls}, e_1, ..., e_n, e_{sep}) \\ &\forall d_i, d_j \in D, i \neq j \Leftrightarrow E_{cls}(d_i) \neq E_{cls}(d_j) \end{aligned} \tag{5.2}$$

Where d_i and d_j are two documents from the corpus D. The contextual representation for these two documents $E_{cls}(d_i)$ and $E_{cls}(d_j)$ cannot be equal as far as these two documents are not identical. This requirement guarantees a unique contextual representation for each document in the corpus.

Incorporating keywords in addition to document embedding enriches the representation of a text. Keywords act as beacons, emphasizing core themes and offering a direct, interpretable route to determining the document's main ideas. Although embedding yields a rich, subtle textual content characterization, keywords can help recognize the stories' central themes.

We use an encoder-decoder architecture to reduce the size of contextual document embeddings generated by this large language model to distill the most crucial information from the high-dimensional embeddings into a more compact form. By compressing embeddings, the encoder-decoder model also reduces the noise in the document embedding. This transformation leads to a more robust contextual representation of documents in the corpus.

Given these advantages, we have developed a multi-layer neural network specifically designed to extract keywords from a document embedding.

$$\begin{aligned} &\forall d_j \in D, E_{cls}(d_j) = e_{cls} \ \wedge \ \forall w_i \in V, (w_i, [0..1]) \in v_{tf-idf} \\ &mlp(\mu) : e_{cls} \rightarrow v_{tf-idf} \end{aligned} \tag{5.3}$$

Where w is a term in a document d, and D is the corpus, which is a collection of documents. $mlp()$ represents a deep neural network with a set of parameters μ to generate a TF-IDF vector: v_{tf-idf} for document d using contextual document embedding e_{cls}.

Our multi-layer neural network is designed to take a document's contextual embedding vector and generate a corresponding TF-IDF vector as its final output. The input to this model is the output from the contextual document embedding, and it returns in a TF-IDF vector with the size of the corpus's vocabulary, V. The network employs three dense layers and leverages the Adam optimizer to accomplish this. It learns to transform a contextual document embedding into a TF-IDF vector through training on all the documents in the corpus.

Later in our process, this model will extract key terms from a hypothesis text by processing its document embedding as generated by a conditional diffusion model. These extracted keywords will function as features to enhance our proposed algorithms, offering an additional dimension that complements the rich detail encoded within the document embedding.

5.3.2 Conditional generative diffusion model

DifStoryGen trains a generative diffusion model to predict potential future documents in a narrative sequence. The algorithm will need a model that can generate a document embedding of Monday (as an example of a future timestamp), given a document from Sunday and an indication of a topic for the generated document embedding of Monday.

A forward diffusion process (a standard Markov process) involves the following data: an embedding vector $e_{d^{t_i}}$ from timestamp t_i (Sunday), the embedding, $e_{d^{t_{(i+1)}}}$ (Monday), and the centroid of the cluster where $e_{d^{t_{(i+1)}}}$ exists. The noise is only gradually added to $e_{d^{t_{(i+1)}}}$, keeping $e_{d^{t_i}}$ and the centroid fixed in the Markov process. For s steps, there will be s pairs of input and output combinations for each document $e_{d^{t_{(i+1)}}}$. Figure 5.3 illustrates the process.

The forward process in the conditional diffusion model is defined as:

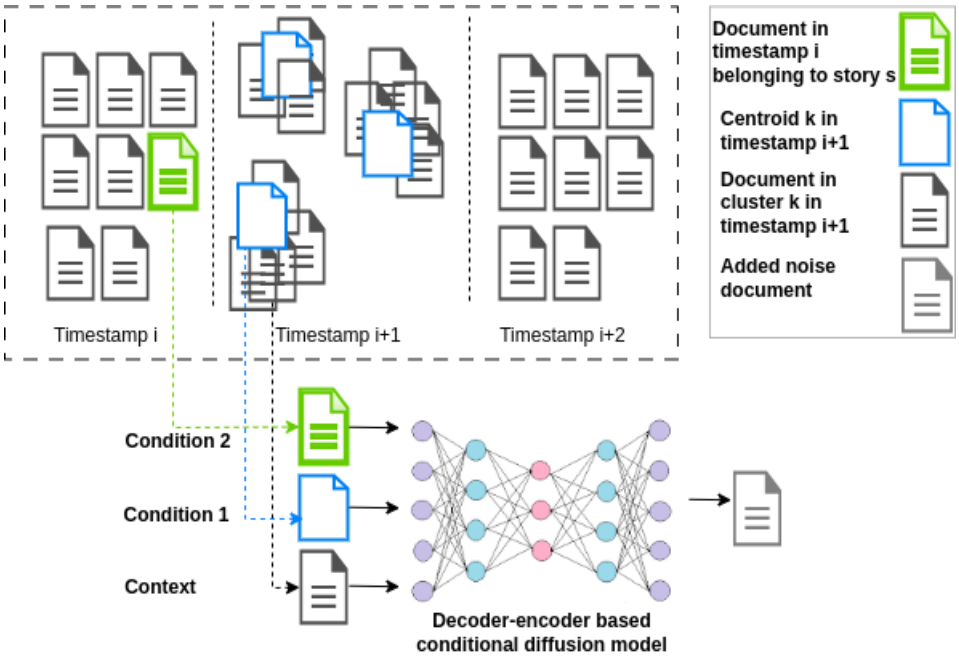

Figure 5.3: Illustration of the conditional generative diffusion model.

$$q(d_s^{i+1}|d_{s-1}^{i+1}, C^{i+1}, E(d^i)) = \mathcal{N}(d_s^{i+1}; \beta_s + d_{s-1}^{i+1}) \tag{5.4}$$

In the forward diffusion process, the model gradually adds noise to the data d_0^{i+1} (original document in t_{i+1}) over a sequence of steps s to produce d_s^{i+1}, and a reverse diffusion process which generates samples from the noise by conditioning on the cluster center C^{i+1} and $E(d^i)$. β_s are noise levels at each step, and $\mathcal{N}$ represents the normal distribution.

For training the model, we provide the noisy version of the data prepared during forward diffusion in the input and the previous, less noisy version of forward diffusion in the output. Note here that the model trains on how a Monday document can be generated from noise given a Sunday document and a centroid of Monday. The centroid reflects the topic of the generated document because otherwise a document from the corpus can be generated.

The reverse process, which is what we train the model over the documents on the corpus, is:

$$p_\theta(d_{s-1}^{t+1}|d_s^{t+1}, C^{i+1}, E(d^i)) = \mathcal{N}(d_{s-1}^{t+1}; \mu_\theta(d_s^{t+1}, C^{i+1}, E(d^i)), \sigma_s^2\mathbf{I}) \tag{5.5}$$

Here, μ_θ is a neural network with parameters θ, typically an encoder-decoder architecture for text data. The function predicts the mean of the distribution of d_{s-1}^{t+1} given d_s^{t+1} and conditions C^{i+1} and $E(d^i)$, while σ_s^2 is the variance of the reverse process at step s, which is often learned or fixed as a function of β_s.

To sample from the model, we start with noise $d_S \sim \mathcal{N}(0, \mathbf{I})$ and apply the reverse process conditioned on C^{i+1} and $E(d^i)$ to obtain d_0 in timestamp $i+1$.

$$d_0^{i+1} \sim p_\theta(d_0^{i+1}|C^{i+1}, E(d^i)) \tag{5.6}$$

In the context of a diffusion model, conditions should ideally offer a concise yet comprehensive overview of the data they represent. Centroids from k-means clustering fit this criterion.

Determining a reasonable value for *K* in k-means clustering often requires analytic efforts in the absence of previous knowledge about data. Centroids that are distinct from one another better represent the corpus. Conversely, centroids that closely resemble each other represent similar topics and hence indicate redundant clusters.

In Figure 5.4, the distribution (with a boxplot) of cosine similarity between all-pair centroids for different k averaged over each data timestamp. The plot demonstrates that as the number of clusters increases, the average similarity between clusters has the tendency to decrease in the beginning. With a large number of clusters, the average similarity does not change much, but the variation of similarities starts to increase. That is, the boxes tend to grow bigger with larger k, from k=6. This necessitates a process to select a reasonable value for the number of clusters, k.

To find a reasonable value for k at timestamp t, we maximize the intra-cluster distance and minimize the inter-cluster distance at each timestamp t. Maximizing Λ in Equation 5.7 ensures that we have high-quality clusters that are distant from each other in each timestamp.

$$\Lambda(d, c, C) = \frac{1}{k|C^i|} \frac{\sum_{i=1}^{k} \sum_{d \in D_k^i} sim(d, c^i)}{\sum_{i \neq j} sim(c^i, c^j)} \tag{5.7}$$

where C^i is the set of documents in cluster i. $|C^i|$ is the number of documents in cluster i, and d is a document in cluster i. c^i is the centroid of cluster i in timestamp t and $sim(d, c^i)$ is the cosine similarity between document d and centroid c^i.

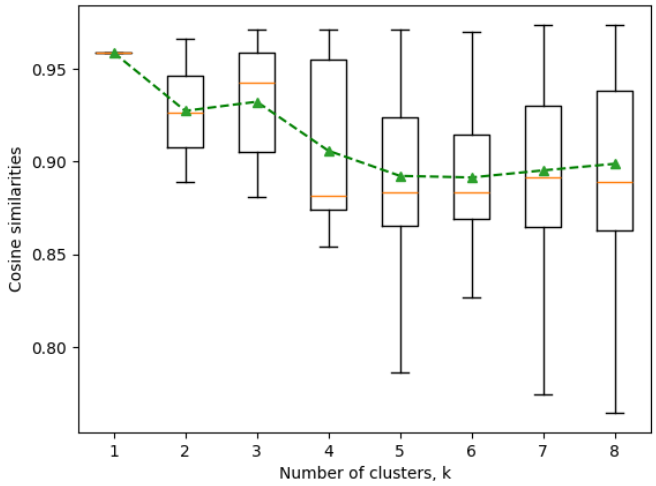

Figure 5.4: The average cosine similarity between all pairs of cluster centroids against different numbers of clusters, k. The green line is average. Each boxplot represents the distribution of the centroid similarities. $k$means clustering was used to cluster the documents of each timestamp with different k.

Noising process: In a diffusion model, the number of steps for noise addition and the magnitude of the noise itself are two critical hyper-parameters that significantly affect the quality of the generated text. Careful selection of a number of steps can ensure that the training phase later is not over-trained with too much noise-to-noise data in the input. That is, we expect our training data to capture how to transition text in a small amount. A large number of iterations in the noising process will result in more noise-to-noise data rather than capturing the text into noisy-text transition. To define when to stop creating more pairs of instances of data and noisy data, we design a formula that computes the ratio of two elements for each document: (1) similarity between the original document and complete noise and (2) similarity between the original document and computed noise added so far iteratively. The ratio is then averaged over all documents. Equation 5.8 provides the ratio.

$$\frac{1}{K}\sum_{i=1}^{K}\frac{Cosine(e_i,[\mathcal{N}(0,\mathbf{I})])}{Cosine(e_i,[e_i+(s-1)\eta\times\mathcal{N}(0,\mathbf{I})])}\sim 1 \tag{5.8}$$

Where e_i is a document embedding of document d_i and η is a scaling factor (usually much smaller than 1.0) applied to each step of noising. s is the iteration number, and K is the number of

documents in the training set. The ratio is less than 1 in the beginning and will increase over iterations because the denominator decreases. The blue line in Figure 5.5 shows the denominator, reflecting that the similarity between the original document and the noisy document monotonically decreases. The green line marks where the average ratio for all documents becomes 1.0. The orange line marks the number of steps where the blue and green lines are crisscrossed. For Figure 5.5, we used more than 8k documents and a noise scaling factor, $\eta = 0.025$, that resulted in 115 steps to train the diffusion model. After creating the data for diffusion, we proceed to the training phase of the conditional diffusion model.

5.3.3 Story generator

Algorithm 2 outlines the generator, StoryGen, which is a document search mechanism that starts at document m within timestamp i and navigates towards document n at timestamp j, where $j > i$. The sequence of documents in this path, starting from document m and ending in document n, constitutes a story spread across different timestamps. The story generator traces a path and strategically selects documents that exhibit high similarity to the story's main narrative, maintaining evolution and coherence simultaneously.

The conditional generative diffusion model aids in predicting the following document in the sequence, effectively forecasting the narrative's progression. The distributed attention similarity mechanism, as seen in Figure 5.6, is another crucial feature in our search algorithm, which ensures that selecting the upcoming documents in the story is not just based on the last documents in the story but all documents in the story, weighted by their contribution to the core storyline and their temporal distance. We explain the components below.

Initializing the StoryGen algorithm:

The search process in the StoryGen algorithm is initiated with a seed document. StoryGen selects the initial timestamp i, corresponding to the seed document, and proceeds to continue the story through documents associated with the following timestamps, all of which are greater than or equal to i. As the story develops, the algorithm constructs and continually updates a set of

Algorithm 2 $StoryGen(D, d_{seed}, E_i, TF, T, C_x, \theta, \alpha, \beta)$

D=$\{d_1, .., d_k\}$ is a document collection.
S=$\{s_1, ..., s_m\}$ is a chain of documents in story s.
d_{seed} is the initial document for the story.
E_i=$\{e_1, ..., e_k\}$ is a set of document embeddings of documents in timestamp i.
TF=$\{tf_1, ..., tf_k\}$ is a collection of keywords in D.
T=$\{t_1, ..., t_p\}$ is a collection of timestamps in D.
C_x=$\{c_1, ..., c_z\}$ is a collection of centroids in t_x.
θ is the minimum acceptable similarity between d_i and d_j in story S.
α maximum similarity accepted for the generative model.
β the constant value for the temporal drift.

$S \leftarrow \phi$
$d_{selected} \leftarrow d_{seed}$
$S \leftarrow d_{selected}$
$storyLine \leftarrow TF(d_{selected})$
$harvesting \leftarrow True$
for t_i in $T = \{t_1, ..., t_n\}$ **do**
  while $harvesting$ **do**
    $pool = Diffusion(C_i, E_i, d_{selected})$
    for x, y in $pool$ **do**
      if $Sim(x, y) > \alpha$ **then**
        $pool.remove(y)$
      end if
    end for
    $candidate = Max(pool)$
    $similarity = 0$
    for d_j in D^{t_i+1} **do**
      for s_i in $S = \{s_1, ..., s_n\}$ **do**
        $w_i = TF(s_i) \times TF(storyLine)$
        $w_i = w_i \times (1 - \beta)^{(n-i)}$
        $similarity+ = w_i \times Cosine(s_i, candidate)$
      end for
      $distAttnResults \leftarrow (similarity, d_j)$
    end for
    sim, d_j = Max($distAttnResults$)
    if $sim > \theta$ **then**
      $d_{selected} \leftarrow d_j$
      $S \leftarrow d_j$
      $Update(storyLine, d_j)$
    else
      $harvesting \leftarrow False$
    end if
  end while
end for

keywords that represent the story's main entities and characters.

This keyword set is derived from the top-N TF-IDF scores, with $TF_{[:N]}(S)$ across all documents in the story S that has been constructed so far. With each new document d_i added to the story, this set is updated—a weighted update that assigns a τ weight to the existing keywords in the past documents. This weighted update technique (Equation 5.9) allows the narrative elements and entities within the story to evolve dynamically as the story is forming.

$$TF(S) = (1 - \tau) \times TF_{[:N]}(S) + \tau \times TF_{[:N]}(d_i) \tag{5.9}$$

Generating hypothetical document: In the next step, StoryGen clusters documents within the next timestamp and retrieves the cluster centroids. As explained earlier, these centroids represent the various topics or ideas that might emerge in the following timestamps. The algorithm dynamically determines the number of clusters using the method described earlier (Equation 5.7).

The diffusion model utilizes the centroids with the last document's embedding vector in the story as conditions to generate hypothetical document embeddings representing the potential next documents in the story as shown in Equation 5.5. The conditions ensure that cohesive new documents are generated that flow well with the ongoing story.

The conditional diffusion Network ψ generates M document embeddings from the two given conditions, C^{i+1} and $E(d^i)$. To maintain diversity and reduce redundancy, StoryGen filters generated embeddings, eliminating highly similar generated document embedding vectors, as formulated in Equation 5.10.

$$\forall e_i, e_j \in \psi(C^{i+1}, E(d^i)) \Leftrightarrow Sim(e_i, e_j) < \vartheta \tag{5.10}$$

where, ϑ is a similarity threshold.

StoryGen generates TF-IDF of the newly generated document embeddings from the neural network we trained over corpus documents (Subsection 5.3.1). We pick the generated document with the highest TF-IDF-based weighted Jaccard similarity with the last document. The generative model ensures contextual similarity at the embedding level, and our selection using TF-IDF ensures content overlap between the last document and the generated document.

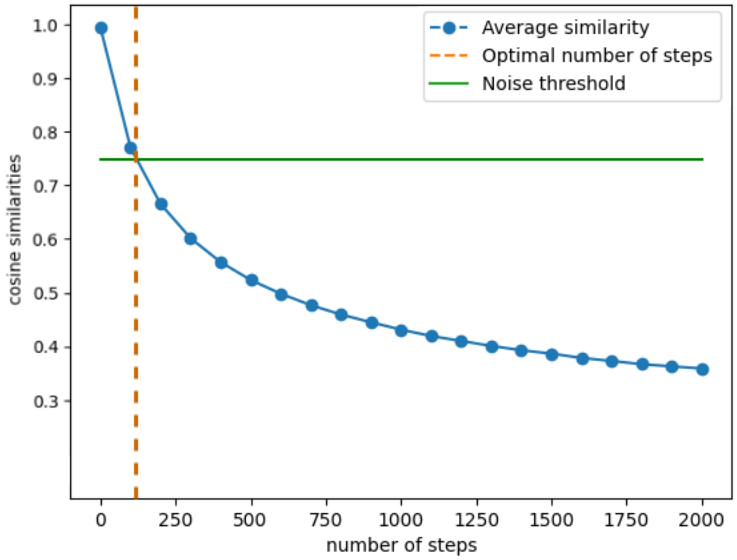

Figure 5.5: The detection of an optimal number of steps for the generative diffusion model during training data creation. The cosine similarities between the original documents and documents with added noise at each step are plotted.

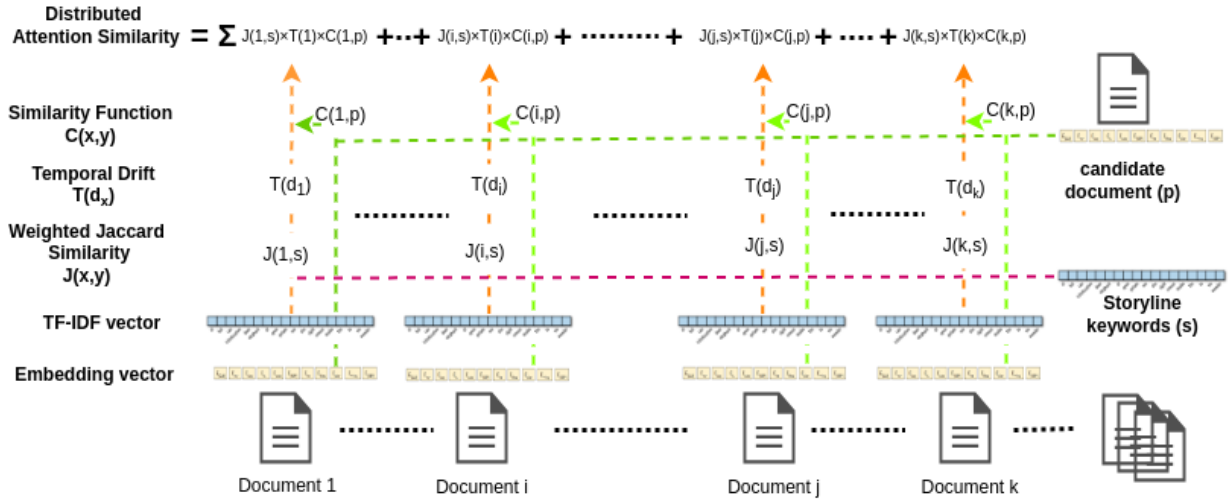

Figure 5.6: Distributed Attention Similarity calculates the contribution of each document to the story generated so far.

StoryGen uses the selected generated document to find the most relevant document in the following timestamps to add to the story.

To select the next document, argmax of distributed attention similarity between the documents in the next timestamp and the selected generated document embedding vector is used (Equation 5.11).

$$d_i \in S \Leftrightarrow f(d_c, S) > \gamma \wedge \ d_i = argmax(distAttn(S, d_c)) \tag{5.11}$$

Where d_c is a document embedding vector generated by diffusion model ψ, and d_i is the best candidate picked by distributed attention in timestamp t, which satisfies requirements γ. Next, we explain the distributed attention similarity function.

Distributed attention similarity mechanism:

Unlike previous storytelling algorithms in the literature, which applied similarity measurements only to the latest document during story progression, ignoring the storyline's historical context, DifStoryGen uses a distributed attention similarity mechanism. The distributed attention incorporates other documents within the story constructed so far based on their contribution to the main story and in which timestamps they are positioned.

Distributed attention is a weight vector to quantify each document's impact on a story's ongoing narrative. Leveraging these weights helps StoryGen maintain a coherent narrative with a smooth evolution of entities.

Equation 5.12 calculates the distributed attention similarities, which is a vector of similarities between a candidate document d_c for the story S constructed so far.

$$distAttn(S, d_c) = \frac{1}{|S|} \sum_{i=1}^{|S|} (1-\alpha)^{|S|-i} \times \frac{\sum \text{TF}(S) \times \text{TF}(s_i)}{|V|} \times Cosine(E(d_c), E(s_i)) \tag{5.12}$$

where $TF()$ is a function to retrieve the $TF-IDF$ vector of a document d with vocabulary V. α is a temporal drift parameter to assign weights based on the distance of the head of the story. $Cosine(E(d_c), E(s_i))$ calculates the similarity between document embedding vectors of s_i in story S and candidate document d_c.

Harvesting documents for the story chain:

At the last step of the StoryGen algorithm, the model utilizes the computed distributed attention similarity vector to select the document that most aligns with the hypothetical sample generated by the diffusion model. If this document's similarity is greater than a certain threshold, which we call the StoryGen threshold, the document is considered the next document in the story. Then, the algorithm updates the storyline keyword sets (Equation 5.9) and repeats the process for the following timestamps. If the similarity is less than the threshold, the model ignores that timestamp, repeating the process with the next timestamp to look forward to collecting other relevant documents.

The StoryGen threshold is adjustable and can be modified in various storytelling applications. Assigning for a lower threshold allows the story to cover broader topics, including a variety of entities. On the other hand, a higher threshold narrows the focus, keeping the storyline tightly knit around a few entities and characters.

5.4 Conclusions

This chapter presented DifStoryGen (Diffusion-based Story Generator), a novel storytelling algorithm that leverages a diffusion model and distributed attention mechanism to construct a story chain from a collection of documents. DifStoryGen provides a narrative that maintains a consistent theme, tracks the evolution of events and entities, and generates hypothetical intermediate documents to connect temporally distant documents due to missing a supporting document. Also, it uses contextual similarity rather than direct word overlap in its search process to form the story chain. The next chapter 6 will evaluate the algorithm's effectiveness using diverse metrics and analyze the impact of its individual components.

Chapter 6

Experimental Analysis: DifStoryGen

6.1 Introduction

In this chapter, I demonstrate a series of experiments designed to evaluate the effectiveness of DifStoryGen and its individual components in constructing cohesive story chains. We introduce different evaluation metrics and discuss DifStoryGen's performance across two datasets, *The New York Times* articles and PubMed abstracts. Additionally, we conduct an ablation study to determine the contribution of each component in this algorithm. Also, we analyze how integrating DifStoryGen into different classification and clustering tasks can enhance the performance of these downstream tasks.

6.1.1 Evaluation metrics

We use different evaluation metrics to assess the quality of stories: the Hit@K metric, the Dispersion Coefficient, the Story Evolution Coefficient, and storyStretch, which are described below.

6.1.1.1 Hit@K for content overlap between consecutive documents

To determine the flow of entities in a story from start to end, we use an average of Hit@K of all consecutive pairs of documents in the story, where a hit refers to the overlap of any of the top k highest TF-IDF entities of each of i-th and $(i+1)$-th document in the story. Hit@K is either 1 or 0, based on whether there is a hit or not.

For example, hit@10 determines whether any shared keywords exist among the top 10 highest

TF-IDF keywords between a pair of consecutive documents. If the story length is 15, there are 14 such consecutive pairs, and if hit@10 is 1 for 11 pairs and 0 for the rest, then the average Hit@K will be 11/14. In our experiments, we used hit@10, hit@30, and hit@50 to examine the overlap of entities in three different levels between consecutive documents in a story.

6.1.1.2 Dispersion coefficient for coherence and separation

Dispersion coefficient [11] measures the extent of overlap between consecutive documents within a narrative while penalizing overlaps between non-consecutive documents. A dispersion coefficient of 1.0 indicates that overlaps are exclusively found between each consecutive pair of documents in the story. The coefficient value decreases as the narrative begins to exhibit content overlaps among non-consecutive documents. Contrarily, a coefficient of 0.0 indicates a complete absence of overlaps in consecutive document pairs, with all overlaps occurring only between non-consecutive pairs of a certain threshold.

The dispersion coefficient of a story chain $\{d_0, d_1, ..., d_{n-1}\}$ is defined by the following two formulae 6.1 and 6.2.

$$disp(d_i, d_j) = \begin{cases} \frac{1}{n+i-j}, & \text{if normdist}(E(d_i), E(d_j)) < \theta. \\ 0, & \text{otherwise.} \end{cases} \tag{6.1}$$

where,

$$\Omega = 1 - \frac{1}{n-2} \sum_{i=0}^{n-3} \sum_{j=i+2}^{n-1} disp(d_i, d_j) \tag{6.2}$$

Where θ is an adequate distance threshold for evaluating a story. For calculating the $normdist$, Soergel distance, cosine dissimilarity, or any other normalized dissimilarity between two documents, $E(d_i)$ and $E(d_j)$ can be used.

In our experiments, we calculate the dispersion coefficient by measuring the similarity between two contextual document embedding vectors within a story.

6.1.1.3 Story Evolution Coefficient (SEC) for measuring evolution

In order to assess a story's evolution, two main aspects must be considered:

- The degree of overlap between consecutive documents in a story: Insufficient overlap between two consecutive documents results in a discontinuity in the story, indicating a lack of evolution.

- The degree of change between consecutive documents: If consecutive documents are too similar, there might not be enough progress to support the development of an evolving narrative.

While the dispersion coefficient incorporates both overlaps between consecutive documents (coherence) and the difference of non-consecutive documents (separation and hence evolution), the change of concepts is not incorporated.

The Story Evolution Coefficient (SEC) [60] reflects these criteria as an evaluation metric. SEC reaches its maximum value at a specific level of similarity between consecutive documents, before and after which the function should decrease rapidly to penalize both lower and higher levels of similarity between consecutive documents. The function's peak is represented by τ_{max}, a parameter empirically determined where random pairs of documents have reasonable content and contextual vector similarity overlap.

Figure 6.1(a) illustrates the relationship between the overlap in terms (words) and the cosine similarity among document pairs. For each level of term overlap, there is a boxplot that represents the distribution of contextual cosine similarities. The blue dashed line indicates the number of document pairs corresponding to various amounts of term overlap. This experiment randomly selected 200 documents, each sharing at least one word with another document (after removing all stopwords) in the set, resulting in 9,900 pairs. The observation demonstrates that a single word overlap corresponds to a median contextual vector cosine similarity of approximately 0.79. As the overlap in terms increases from left to right in Figure 6.1(a), the median cosine similarities also tend to increase. The dashed blue line further reveals that pairs of documents sharing around 19 words are the most common, with the distribution of these pairs resembling a Gaussian curve.

The maximum word overlap is 48, and only about 1% of the pairs have more than 37 word overlaps. Based on this observed study, we consider that more than 37-word overlaps can be

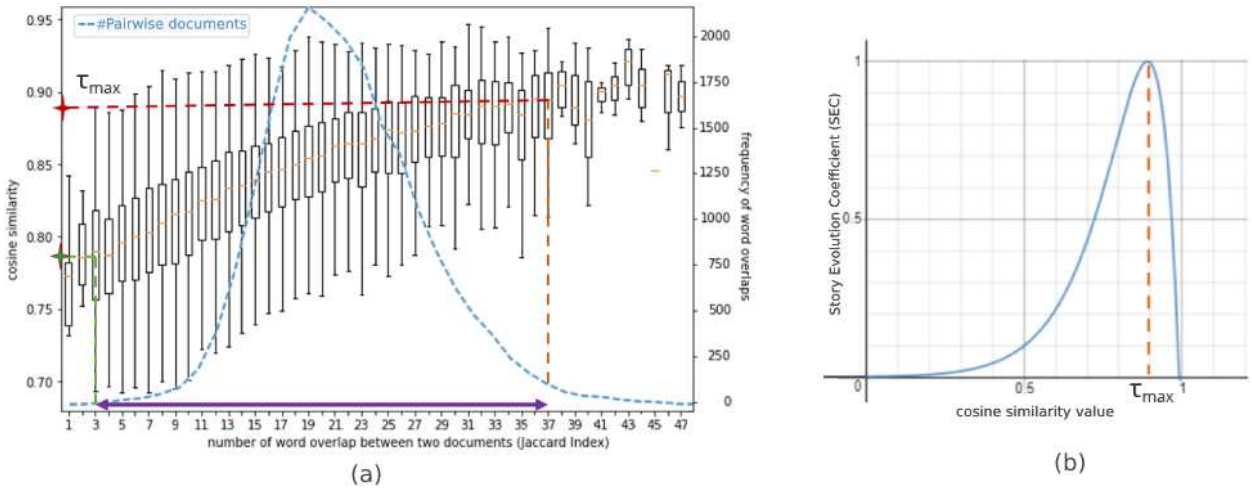

Figure 6.1: (a) Number of term overlapping vs. cosine similarity between pairs of documents. A boxplot of contextual cosine similarities is shown at each term overlap. The blue dashed line shows how many pairs of documents are found with a certain number of term overlaps. (b) The functional distribution of Story Evaluation Coefficient (SEC of Equation 6.3) with respect to cosine similarity values.

considered as *evolution being stuck* and less than three overlaps being *not enough content overlap*. A 37-word overlap is equivalent to a median contextual similarity of 0.89, as shown by τ_{max} in Figure 6.1(a).

The Story Evolution Coefficient (SEC) is a metric designed to peak at τ_{max}, representing the optimal contextual similarity between two consecutive documents in a narrative. Figure 6.1(b) presents the formula outlined in Equation 6.3, which we use to calculate SEC for a collection of stories.

Story Evolution Coefficient (SEC) includes the evolution of entities in a story by incorporating overlaps between consecutive documents and variations between them. Story Evolution Coefficient is story S and it is defined by:

$$SEC(S) = \frac{1}{(n-1)|S|} \sum_{s=1}^{|S|} \sum_{i=1}^{n-1} (\varphi\tau_{max} - \varphi Cosine(E(d_i^s), E(d_{i+1}^s)) + \rho) \times exp(-\varphi(\tau_{max} - Cosine(E(d_i^s), E(d_{i+1}^s)))) \tag{6.3}$$

Where e_i is the contextual embedding vector for document i and ρ is the parameter to set the scale SEC from zero to a desired maximum value.

Our experiments use the range $[0, 1]$ for SEC. φ is a parameter to tune SEC to ensure a reward for the similarity between a pair of consecutive documents if the similarity is between zero to τ_{max}. For similarities greater than τ_{max}, SEC penalizes the overall score of a story for any consecutive document pair with high similarity because high similarity might prevent evolution.

6.1.1.4 Evaluation of the stretch of stories

Short stories usually do not cover the evolution of entities, as our goal is to identify narratives that demonstrate gradual evolution over time. Conversely, analyzing long stories can be challenging due to their complexity and the potential for obvious connections within the document sequence over the entire timeline. In a collection of narratives originating from randomly selected seed documents, both shorter and longer stories are considered inadequate compared to medium-length stories.

To evaluate whether narratives are distributed normally in terms of length, it is essential to evaluate all stories generated from seed documents and analyze their length distribution. An effective storytelling algorithm should mostly produce stories whose lengths are close to the average, with only a few narratives shorter or longer than this mean value. This approach ensures a balanced distribution of story lengths, reflecting the characteristics of real-world stories.

6.2 Experimental Results

To evaluate the performance of our storytelling model, we used two different datasets: 1) the *New York Times* articles dataset and 2) the *PubMed* dataset. In our experiments, 10,000 New York Times articles from 2022 under the "international" and "US" news categories are used to compare the performance of DifStoryGen against other storytelling baseline models. Meanwhile, we used Pubmed dataset to benchmark the impact of incorporating DifStorygen into various classification and clustering baseline models. This dataset contains 50,000 abstractions of biomedical and life sciences literature, with 14 different MeSH terms used as labels.

In this experiment, we compare DifStoryGen with other storytelling algorithms using three approaches: (1) static Doc2Vec embedding model, (2) contextual BERT embedding model, and (3) entity-set overlap-driven similarities (Jaccard).

Our research aims to address several fundamental questions to evaluate the effectiveness of the DifStoryGen model:

1. How well does DifStoryGen perform compared to other models in terms of content flow, dispersion, and evolution? (Section 6.2.2)

2. What is the impact of distributed attention on the stories generated by DifStoryGen? (Section 6.2.3)

3. What is the impact of DifStoryGen on enhancing the performance of classification and clustering models? (Section 6.2.4)

4. Are all major components of DifStoryGen absolutely required? (Section 6.2.5)

5. How does a story change when distributed attention is applied? (Section 6.2.6)

6. How does a narrative of a story generated by DifStoryGen impact LLMs applications? (Section 6.2.7)

Addressing these questions is crucial for advancing the field of automated storytelling and enhancing models' abilities to generate coherent, engaging, and robust narratives.

6.2.1 Implementation details

The DifStoryGen utilizes the base version of the BERT model, which has 12 transformer blocks and 12 layers of attention heads. This base model, containing more than 110 million parameters and receiving 768 tokens, then generates a 768-dimensional embedding vector for each corresponding token. In our storytelling algorithm, we use the CLS token of BERT as the document embedding vector for every document, representing a text uniquely with all contextual and semantics information encoded into that vector.

To align with the requirements of the contextual document embedding generator, every document is truncated to a maximum of 512 tokens. This resizing is accomplished by applying a WordPiece tokenizer, which reduces the length of each document to fit within the specified token limit.

Initial documents (seed documents) serve as the narrative's foundation. The algorithm identifies and selects temporally relevant documents to these initial points. We randomly selected a thousand articles as seed documents for this experiment. These seed documents are selected from the first quarter of 2022 to construct a narrative containing various events and entities.

6.2.2 Quality evaluation of stories

In this subsection, we use the story quality evaluation metrics that we outlined in Section 6.1.1. Plots in Figure 6.2 portray the results using Hit@K, dispersion coefficient, and SEC.

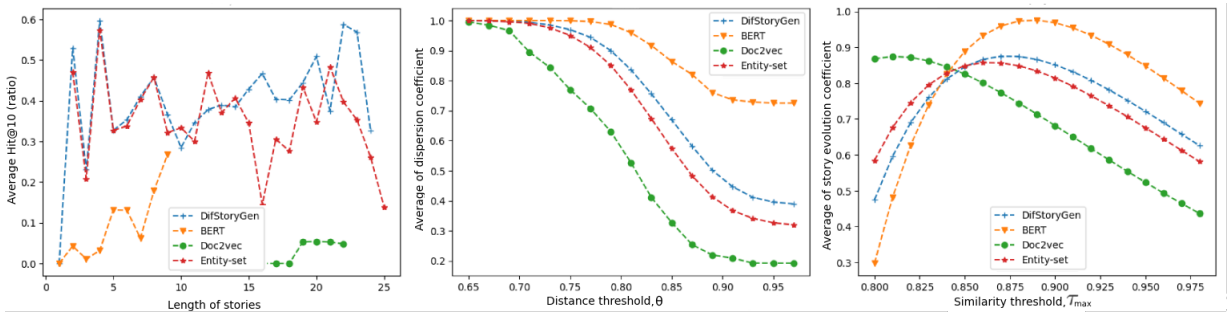

Figure 6.2: (left) Average Hit@10 ratio – the sum of Hit@10 for all consecutive pairs in a story divided by the number of consecutive pairs. (middle) Dispersion coefficient averaged over all stories for different ranges of thresholds (right) Story Evolution Coefficient, SEC, averaged over all stories at different similarity threshold τ_{max}.

For the experiment in this section, we generated 1000 stories, but some of the generated stories were sub-stories of longer stories. We removed any sub-story generated by the same storytelling method. DifStoryGen, Doc2Vec, BERT, and entity-set-based approaches ended up with 471, 467, 459, and 415 stories, respectively.

Table 6.1: A comparison of different storytelling methods in terms of Hit@K, dispersion coefficient, and SEC.

Model	H@10	H@30	H@50	$disp_{32}$	$disp_{64}$	$disp_{128}$	SEC_{32}	SEC_{64}	SEC_{128}
BERT Storytelling	0.09	0.18	0.46	0.53	**0.88**	**0.95**	0.76	**0.95**	**0.97**
doc2vec Storytelling	0.04	0.05	0.23	0.48	0.47	0.53	0.70	0.65	0.71
Entity-set based Storytelling	0.36	0.61	0.79	0.76	0.76	0.76	**0.85**	0.85	0.85
DifStoryGen w/o Attn	**0.42**	**0.69**	**0.86**	**0.92**	**0.94**	**0.95**	**0.93**	**0.94**	**0.94**
DifStoryGen	**0.40**	**0.64**	**0.81**	**0.83**	0.82	**0.84**	**0.85**	0.86	0.87

Figure 6.2 (left) shows an average Hit@10 against the length of stories. BERT and Doc2Vec have low Hit@10, indicating that the consecutive pairs of documents in the stories had less commonality in entities. BERT produced shorter stories, while Doc2Vec produced longer ones. The entity-set-based approach has a high Hit@10, but that is to be expected because entity-set-

based storytelling focuses on maximizing entity overlaps for constructing stories. Our approach, DifStoryGen (the blue line), exhibits an even higher Hit@10 than entity-set-based storytelling, demonstrating that DifStoryGen does not compromise basic story properties despite consideration of context and other crucial elements.

Figure 6.2 (middle) demonstrates the average dispersion coefficients of all the stories produced by each method as a function of distance threshold, θ. A higher dispersion coefficient refers to coherent (better) stories with lesser distance between consecutive documents and more distance between non-consecutive documents. BERT has the highest dispersion coefficient. This is because the downstream storytelling algorithm focused on lower context embedding distance when selecting the next document from a set of candidates. DifStoryGen focused on several factors, including embedding distance and how generated hypothetical documents can diffuse to the concept of the next document and attention. To maintain the evolution aspect of DifStoryGen, it compromises the dispersion coefficient but not as much as the Doc2Vec and entity-set-based approaches. Note that the underlying embedding space of our DifStoryGen is still the BERT embedding. On top of BERT similarity, DifStoryGen considers other essential elements to generate evolving stories rather than stories that merely exhibit high dispersion.

Figure 6.2 (right) demonstrates a similar trend as Figure 6.2 (middle). BERT has the highest SEC, indicating high evolution. Our DifStoryGen method has the second highest one. However, this evolution metric relies heavily on the embedding change, thereby favoring BERT more than DifStoryGen. Referring back to Figure 6.2 (left), BERT has a very low average Hit@K, indicating that even though stories generated by BERT have high evolution content-wise, the evolution is not reflected in consecutive documents in the BERT-based storytelling approach.

DifStoryGen has a balance of Hit@K, dispersion, and evolution, generating more meaningful evolving stories than other approaches. This subsection includes the results of DifStoryGen with distributed attention. The following subsection provides a comparative study on the impact of attention in DifStoryGen.

The Story Stretch metric evaluates the length distribution of stories created by a storytelling algorithm. Stories shorter than average may lack sufficient detail for event evolution, while those

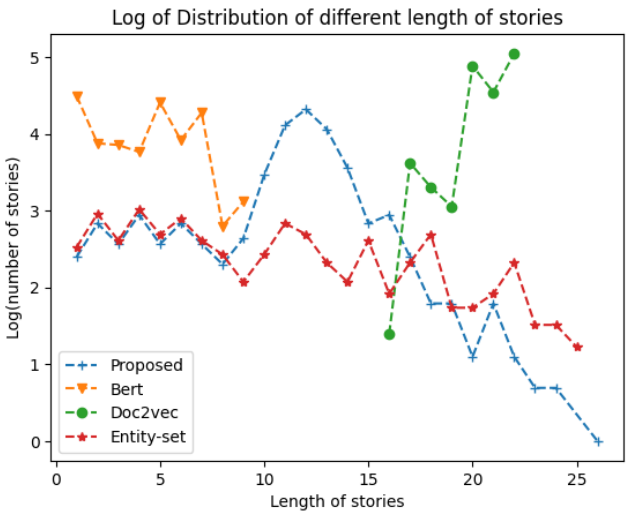

Figure 6.3: Log of distribution of stories length by different models for stories containing less than 25 documents

exceeding average length could include irrelevant content, complicating entity analysis. Stories naturally vary in length, with most following a normal distribution known as Gaussian distribution. Figures 6.3 and 6.4 present the length distribution for different models. Analysis of length distributions for different models, like BERT and Doc2Vec, shows these models typically produce stories within a narrow length range. Conversely, the entity-set-based and our proposed model cover a wider range of story lengths. The entity-set model shows a uniform distribution across lengths, whereas the proposed model's distribution resembles a normal (Gaussian) distribution, with more stories close to the average story length. This finding suggests that stories generated by the DifStoryGen model reflect real-world narrative structures more accurately, making them more suitable for various downstream applications.

6.2.3 Impact of *distributed attention* in DifStoryGen

Table 6.1 compares the baseline models against two versions of DifStoryGen, one with and the other without distributed attention. The table provides story quality metrics using Hit@10, Hit@30, Hit@50, Dispersion Coefficient, and Story Evolution Coefficient using BERT vectors compressed

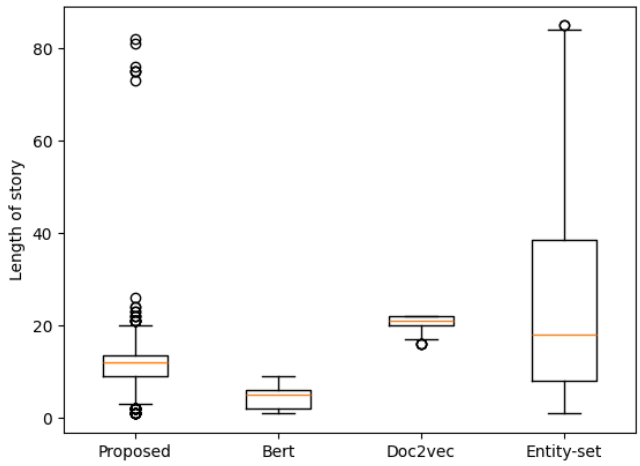

Figure 6.4: The different lengths of stories generated by different models

to lengths of 32, 64, and 128 using an encoder-decoder neural network. In the previous section (Section 6.2.2), we used vectors of length 128.

As shown in Table 6.1, DifStoryGen performs better with and without distributed attention in Hit@K evaluation with k=10, 30, and 50. DifStoryGen without distributed attention performs slightly better than DifStoryGen with distributed attention, which is to be expected because the attention mechanism ensures content overlap with past documents of the storyline generated thus far in the process, resulting in slightly lesser content overlap between consecutive pairs of documents.

When embedding vectors of length 32 are used, DifStoryGen exhibits the highest dispersion (0.92 for DifStoryGen without attention and 0.83 with attention.) Computationally, we desire to design methods that will provide superior results using smaller vectors, and DifStoryGen seems to provide better dispersion with short vector lengths. As the vector size increases to 64 and 128, DifStoryGen, without attention, exhibits an equal or higher dispersion coefficient than BERT. However, DifStoryGen, with attention, exhibits a slightly lower dispersion coefficient but still reasonably high values (0.82 with a vector length of 64 and 0.84 with a vector length of 128).

Similar trends are observed with SEC. At a vector length of 32, DifStoryGen with and without attention is superior. With increased vector size, BERT starts to exhibit higher evolution scores. Note that BERT's content overlap (Hit@K) values are quite small compared to DifStoryGen's,

indicating that DifStoryGen is the superior model in terms of balancing all the quality metrics.

6.2.4 DifStoryGen Downstream Applications

In this subsection, we want to analyze the impact of incorporating the DifStoryGen model into downstream tasks such as classification and clustering. In this way, we can show the capabilities of DifStoryGen to support downstream tasks.

In this subsection, we analyze the impact of integrating the DifStoryGen model on downstream tasks, including classification and clustering. We highlight DifStoryGen's potential to strengthen these tasks. This analysis will demonstrate how DifStoryGen enhances the performance and efficiency of downstream applications, proving its utility in supporting a range of analytical techniques.

First, we store all stories generated from initial documents. Then, we identify the adjacent articles for each document to develop a new embedding vector composed of the original contextual embedding vector and the average contextual embedding of these neighbors. A neural network is designed to map each document's embedding to these new embedding vectors containing the information from neighbors. During the classification and clustering process, we generate these vectors for every document in a dataset. In the following subsections, we show that this approach boosts the performance of different models by leveraging the enriched context of each document's neighbors.

6.2.4.1 Experiments on Classification

In our study, we deployed the DifStoryGen algorithm to augment the contextual embeddings of documents with the goal of boosting the performance of various classification models. Our experiments utilized the PubMed dataset, which contains 50,000 abstracts from the biomedical and life sciences literature. Given its rich collection of biomedical literature, the PubMed dataset was an excellent choice for evaluating the impact of our embedding augmentation approach in an area where precision is important.

Table 6.2: Accuracy, Precision, Recall, and Accuracy for different classification models with and without incorporating DifStoryGen algorithm

	Accuracy		Precision		Recall		F1	
DifStoryGen embedding	w/o	w/	w/o	w/	w/o	w/	w/o	w
SVM	0.80	**0.84**	0.71	**0.82**	0.80	**0.84**	0.74	**0.82**
KNeighbors Classifier	0.77	**0.82**	0.74	**0.81**	0.77	**0.82**	0.75	**0.81**
Decision Tree	0.79	**0.82**	0.71	**0.80**	0.79	**0.82**	0.75	**0.80**
Logistic Regression	0.79	**0.85**	0.73	**0.82**	0.79	**0.85**	0.74	**0.83**
Gaussian Naive Bayes	0.78	**0.82**	0.73	**0.80**	0.78	**0.82**	0.74	**0.81**
Stochastic Gradient Descent	0.79	**0.84**	0.71	**0.82**	0.79	**0.84**	0.73	**0.82**
Random Forest	0.78	**0.84**	0.75	**0.82**	0.78	**0.84**	0.76	**0.82**
Gradient Boosting	0.79	**0.83**	0.74	**0.81**	0.79	**0.83**	0.75	**0.81**

First, we selected 10,000 articles from this dataset for story generation using DifStoryGen, which were used to train our encoder-decoder model. This model was then employed to predict the embedding vectors for the neighboring articles of the remaining 40,000 articles, which were used for classification and clustering tasks. We used a split ratio of 75% for training and 25% for testing in our classification experiments. The enhanced embeddings, derived from the neighbors within the stories generated by DifStoryGen, were merged with the original context embeddings produced by LLMs.

This fusion was designed to create a more comprehensive representation of the textual documents in our dataset, incorporating additional historical context of the text. Our innovative embedding strategy was aimed at providing more precise and reliable classification results in our experiments.

In this study, we evaluated eight diverse classification models to determine their performance, with and without integrating the DifStoryGen algorithm. These models were selected due to their wide range of applicability across different classification tasks. The configurations and parameters for each model were carefully selected to optimize performance, and all used Min-Max Scalar to

Table 6.3: False-positive, true-positive, false-negative and True-negative rates for all classification models with and without DifStoryGen

	FPR		TPR		FNR		TNR	
DifStoryGen embedding	w/o	w/	w/o	w/	w/o	w/	w/o	w
SVM	0.19	**0.12**	0.28	**0.43**	0.72	**0.57**	0.80	**0.88**
KNeighbors	0.21	**0.17**	0.38	**0.51**	0.62	**0.48**	0.79	**0.83**
Decision Tree	0.14	**0.11**	0.24	**0.37**	0.76	**0.63**	0.86	**0.87**
Logistic Regression	0.24	**0.14**	0.30	**0.48**	0.70	**0.52**	0.76	**0.86**
Gaussian Naive Bayes	0.27	**0.16**	0.33	**0.46**	0.66	**0.53**	0.73	**0.84**
Stochastic Gradient Descent	0.25	**0.11**	0.30	**0.41**	0.70	**0.59**	0.75	**0.89**
Random Forest	0.19	**0.13**	0.36	**0.46**	0.64	**0.54**	0.80	**0.87**
Gradient Boosting	0.20	**0.15**	0.31	**0.48**	0.69	**0.52**	0.80	**0.85**

normalize the input features. These models are detailed as follows:

- **Support Vector Machine (SVM) with RBF Kernel:** This model utilizes the Radial Basis Function (RBF) kernel, known for its effectiveness in handling non-linear data.
- **K-Neighbors Classifier:** Set with three neighbors for its classification.
- **Decision Tree Classifier:** It is configured with a maximum depth of 3 to prevent overfitting.
- **Logistic Linear Regression Classifier:** This model applies a logistic regression approach and is optimized for binary classification tasks.
- **Gaussian Naive Bayes Classifier:** Leveraging the assumptions of Gaussian distributions in feature likelihoods.
- **Stochastic Gradient Descent (SGD) Classifier:** Employing a linear Support Vector Machine (SVM) loss function for an efficient classification.

- **Random Forest Classifier:** With 100 estimators and a minimum sample split of 2, this ensemble model combines multiple decision trees to improve classification accuracy and control over-fitting.

- **Gradient Boosting Classifier:** Set with a maximum depth of 1 and 100 estimators, this model focuses on boosting weak learners, optimizing for both bias and variance.

Each model was selected to represent a range of machine learning approaches, from simple to complex and from linear to non-linear classifiers. By applying these models to the same dataset and comparing their performances with and without DifStoryGen's augmented embeddings, we aimed to measure the impact of enriched contextual information on classification accuracy across various algorithmic strategies.

Table 6.3 shows the performance metrics of incorporating the DifStoryGen algorithm with various classification models such as Accuracy, Precision, Recall, and F1 score. The results highlight notable improvements across all metrics when the models are integrated with DifStoryGen. Specifically, the accuracies of the classification models were enhanced up to **0.06**, precisions were increased up to **0.11**, recalls were increased up to **0.06**, and F1 scores improved up to **0.09** in some experiments. It is important to note that these classification models were already fine-tuned, where the original accuracies were around **0.80** even before utilizing DifStoryGen. These enhancements are significant considering the already high performance of the models and show the impact of the DifStoryGen model in various classification models. This demonstrates not only the effectiveness of DifStoryGen in enriching the models' input data but also highlights its potential to refine the accuracy of highly tuned classification models.

Table 6.3 presents the results of incorporating DifStoryGen into different classification models, focusing on the False Positive (FP), False Negative (FN), True Positive (TP), and True Negative (TN) rates. The results from these experiments indicate that applying DifStoryGen across all tested classification models leads to improvements in both TP and TN rates up to **18%** in some classification models while concurrently decreasing FP and FN rates in some models by **20%**. Such enhancements in these rates are significant in fields where precision and recall are

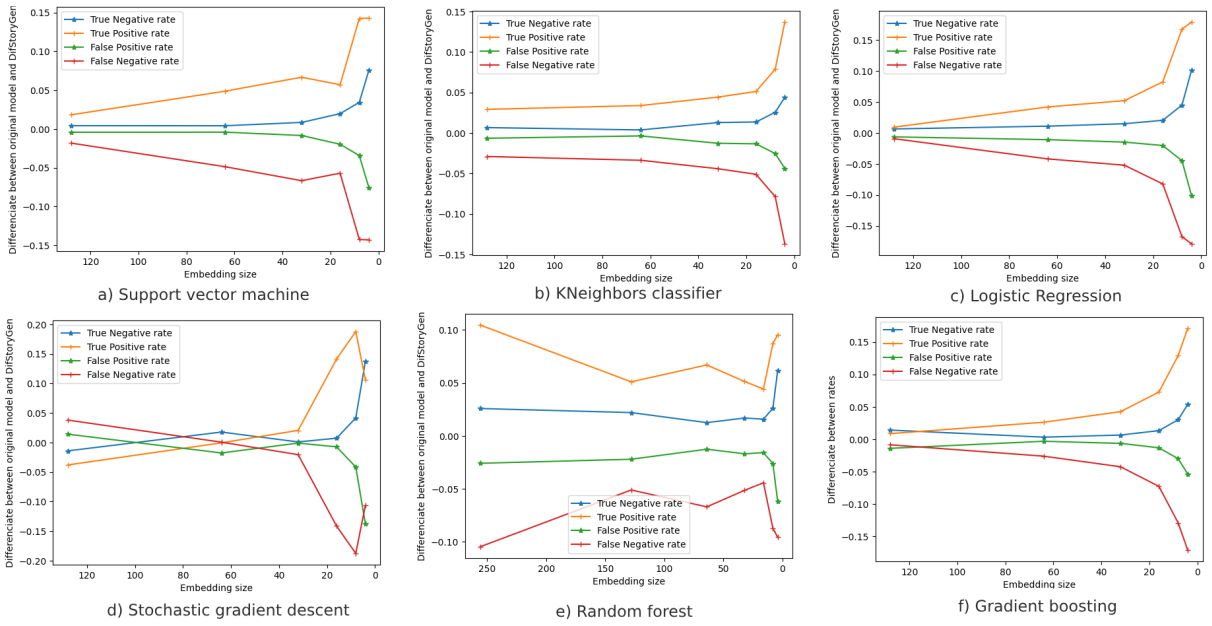

Figure 6.5: The impact of different embedding sizes in True-positive and False-positive True-negative, and False-negative rates in Classifier models

critical, highlighting the importance of DifStoryGen's contribution. By enriching the models with additional information, DifStoryGen not only helps in accurately identifying relevant instances (TP) and correctly rejecting irrelevant ones (TN) but also minimizes the instances of mistakenly flagged irrelevant instances (FP) and overlooked relevant instances (FN). This improvement in performance metrics highlights DifStoryGen's value, especially in applications where the cost of errors is high, and the quality of classification decisions is crucial.

6.2.4.2 Time and Space Complexity of DifStoryGen

The DifStoryGen framework integrates different components to construct cohesive and coherent narratives. Notably, BERT is essential to constructing contextual embedding vectors for texts. This feature optimizes the computational efficiency of DifStoryGen by employing a pre-trained Large Language Model instead of training a Deep Neural Network for generating embedding vectors. The time complexity for this operation is denoted as $O(L \times N^2 \times d)$, where L represents the number of layers in the BERT model (set at 16), N represents the maximum number of tokens

in the input sequence, and d indicates the dimensionality of the hidden states within the BERT model. This approach reduces the computational load, as generating contextual embeddings costs $O(N^2)$, which is significantly less than training a neural network for the same purpose.

Deep neural networks, such as the TF-IDF generator and the conditional diffusion model, are employed throughout the storytelling process to generate the most appropriate candidates from a corpus. Despite the significant training expenses associated with the diffusion model, its inclusion in DifStoryGen is crucial as it connects temporal distant documents to enhance the coherence of the generated stories.

The time complexity of the TF-IDF generator involves both encoding and decoding stages, with costs of $O(L_{enc} \times N)$ and $O(L_{dec} \times M)$, respectively. M and N are the sizes of input and output vectors, and L is the number of layers in each component. This complexity directly depends on the size of the embedding vectors, which can be minimized due to the model's capability to encode information in shorter vectors.

The conditional diffusion model is the most process-consuming component in this model, but it is an essential component due to its importance in bridging the gaps in temporally distanced documents. The time complexity of the training stage is $S \times E \times N \times (O(L_{enc} \times N) + O(L_{dec} \times M))$, where N is the size of the training dataset, M is the size of the latent representation, E is the number of epochs, S is the number of steps to add noises, and L_{enc} and L_{dec} are the number of layers in encoder and decoder. The time complexity to generate a new embedding vector is $O(S \times (O(Lenc \times N) + O(Ldec \times M)))$, less than the training step.

It is essential to mention that applying DifStoryGen for data analysis typically focuses on analyzing the entities offline. Given the critical nature of these applications, the priority is to improve the accuracy of the story chains rather than the processing time. This approach guarantees that the narratives align with sensitive applications' needs.

Also, the distributed attention similarity component, which includes past documents during the candidate selection process, offers flexible parameters to balance the depth of context and processing efficiency. This parameter can be adjusted to optimize its current DifStoryGen's time complexity of $O(N^2)$.

We analyzed the impact of embedding size on these models' performances to analyze the time and space complexity of DifStoryGen incorporated into downstream classification and clustering tasks. Figure 6.5 shows the True Positive Rate (TPR) and False Positive Rate (FPR) across three different classification models, with embedding sizes ranging from 128 to 4. The figure shows that the TPR and FPR for models employing contextual embeddings are near those of DifStoryGen embeddings incorporated when the embedding size is 256. However, a significant difference appears as the embedding size is reduced. These experiments highlight the strength of DifStoryGen models in maintaining performance even with reduced embedding sizes compared with models relying just on contextual embedding vectors.

This discovery is essential, especially for applications constrained by computational resources or storage capacities. The DifStoryGen model's ability to retain effectiveness with smaller embeddings makes it advantageous for scenarios requiring efficiency without compromising model accuracy. This characteristic of DifStoryGen models opens up new possibilities for deploying advanced classification models in environments where utilizing larger embedding vectors might not be feasible.

6.2.4.3 Experiments on Clustering

In this part, we focus on the performance of clustering algorithms using contextual document embedding vectors produced by BERT compared to those augmented with DifStoryGen embedding vectors. Figure 6.6 shows the silhouette scores for k-mean clustering at embedding sizes of 512 and 256. Recognizing the variability of clustering outcomes. In this experiment, we performed each clustering ten times for various values of k and presented the results in a boxplot format to capture the range of outcomes.

The results indicate that clustering with DifStoryGen embedding vectors achieves significantly higher silhouette scores for all cluster sizes (k) in both embedding sizes, 512 and 256. This enhancement indicates that DifStoryGen embedding vectors contribute to a more defined and cohesive cluster structure, improving the contextual embedding vectors generated by LLMs alone.

These findings highlight the value of the DifStoryGen algorithm outcome in the domain of

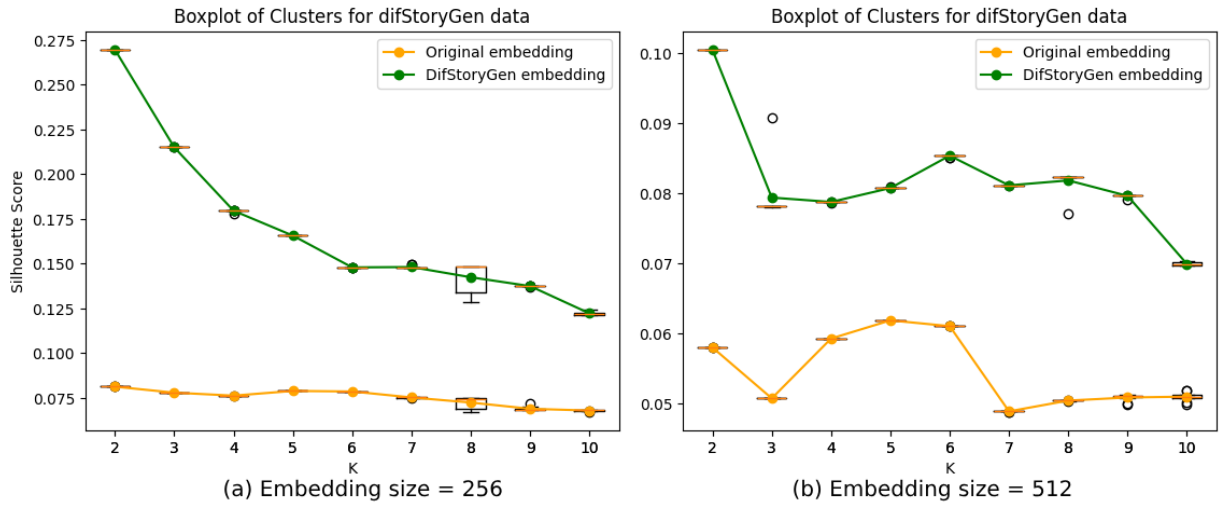

Figure 6.6: Comparison Silhouette of original and DifStoryGen embeddings. The left figure compares the average silhouette scores for different clustering K for embedding size 512. The right figure is using embedding 256 in clustering

clustering by showing that it significantly enhances the clustering structure. This improvement suggests that augmenting DifStoryGen embedding vectors makes the representation more helpful in capturing the underlying patterns and relationships within the data, making them essential for tasks that rely on the quality and interpretability of cluster formations, such as document categorization and data organization.

6.2.5 Ablation study

To analyze the effectiveness of different components of DifStoryGen, we compare it with several of its variations: without distributed attention, without diffusion-based generative model, and without the dynamic keyword set. Table 6.4 shows Hit@K with k=10, 30, and 50, overall average dispersion coefficient, and overall average SEC. DifStoryGen without distributed attention similarity has the highest average values in all evaluation metrics. The embedding vectors had a length of 128 for this ablation study.

DifStoryGen, without a diffusion-based generative model, exhibits lesser evaluation values than DifStoryGen in terms of all Hit@K and SEC but not in dispersion coefficient. DifStoryGen

Table 6.4: Evaluation with different variants of DifStoryGen. H@K in the table refers to Hit@K.

Variations	H@10	H@30	H@50	$disp$	SEC
w/o Attention	0.42	0.69	0.86	0.95	0.94
w/o Diffusion	0.33	0.61	0.79	0.90	0.86
w/o keyword set	0.16	0.34	0.55	0.95	0.96
DifStoryGen	0.40	0.64	0.81	0.84	0.87

without a diffusion-based generative model has a 0.9 dispersion coefficient compared to 0.84 of DifStoryGen. This is because the number of longer stories was reduced when we inactivated the diffusion from DifStoryGen. The diffusion-based document vector helps connect two documents to generate longer stories. Therefore, the model without diffusion with a slight improvement in the dispersion coefficient and deteriorating Hit@K and SEC is undesirable.

The approach without the dynamic keyword set has lower Hit@K but a higher dispersion coefficient and SEC compared to DifStoryGen. Despite its higher dispersion coefficient and SEC values, its extreme lack of content coherence (the lowest Hit@K values) makes the removal of the dynamic keyword set undesirable.

Table 6.4 suggests that DifStoryGen has the best evaluation metrics without distributed attention. The inclusion of attention (which is the regular DifStoryGen) will reduce the values a bit. The purpose of distributed attention in the story is to ensure that the overall content seen so far is not forgotten during the story generation process.

To examine the impact of attention in DifStoryGen, we need another measure that reflects content remembrance as the story progresses. We use average Hit@K with the start document for each document in the story to measure the content remembrance of a story. With K=10, 30, and 50 for Hit@K, DifStoryGen (with attention) exhibits average Hit@K values of 0.28, 0.64, and 0.91, respectively, whereas its attention-removed version has the values 0.17, 0.53, and 0.88. That is, DifStoryGen carries the content themes more toward the story's end than its distributed attention-less version. Depending on requirements, an analyst might choose to use DifStoryGen

with distributed attention or without attention.

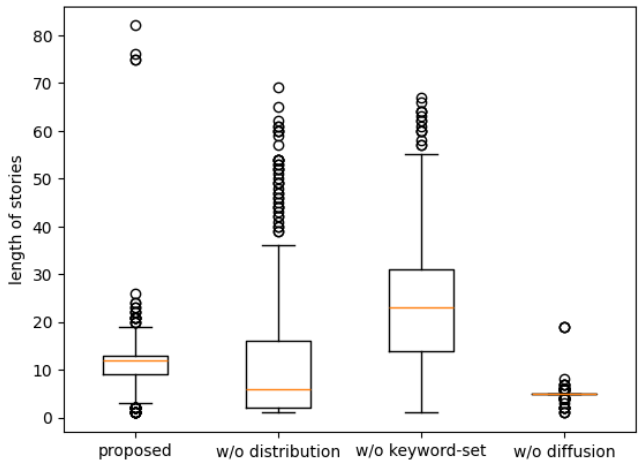

Figure 6.7: The length of stories generated by different variants of the proposed model

Figure 6.7 shows the story length distribution produced by different model versions. The variant without a diffusion network generates shorter stories, primarily because it terminates the narrative when essential documents are missing from the dataset, or there is a temporal gap in the story coverage. This results in shorter stories generated by a storytelling algorithm without a diffusion component. Contrarily, the model version that does not utilize a keyword set constructs narratives without adhering to a specific set of entities. This approach leads to longer stories as it lacks commitment to the presence of entities and characters in the story.

The DifStoryGen and DifStoryGen without distributed attention similarity construct diverse and comprehensive stories, leading to more reliable narratives. However, the stories from DifStoryGen generally have a higher average length than those from DifStoryGen without distributed attention. This is due to the constraints applied by the distributed attention similarity component in DifStoryGen, which focuses the narrative more narrowly on the previous documents of the story.

Figure 6.8 shows that the version of the storytelling algorithm that does not include keyword sets loses its focus on the evolution of entities, concentrating mainly on document embedding. This approach leads to poor performance in the Hit@K metrics across various entities and characters, although it does enhance the embedding coherence between consecutive documents. This outcome

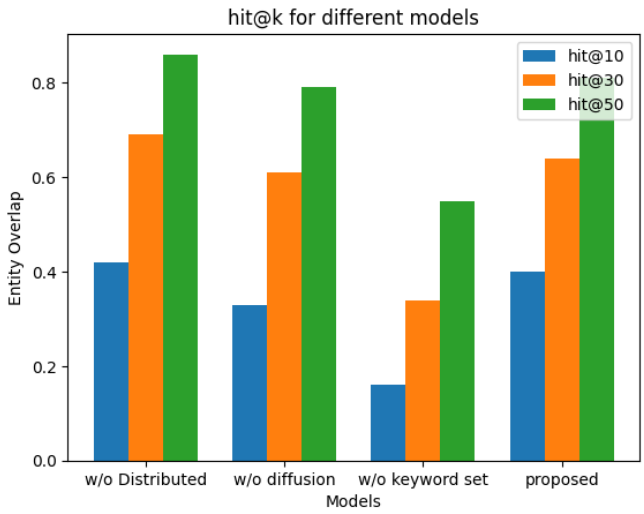

Figure 6.8: The hit@k, k= 10, 30, 50 for different variants of DifStoryGen

illustrates the critical role of incorporating keyword sets in maintaining a balanced focus on entity evolution and embedding consistency.

Figure 6.9 illustrates the performance of our TF-IDF generator in capturing keywords from a document's contextual embedding. It shows the overlap between keywords from the original text and the keywords for the exact text generated from document embeddings for over 2000 documents. The plot indicates that, as the top K words increase, there is a notable rise in overlap, which signifies that many keywords from the embeddings align with the original text. The terms that do not match are often synonyms or semantically related to the main topic, indicating that the contextual embedding effectively captures related concepts, even if not a direct match. The boxplot elements also suggest a variance in the keyword overlap for different documents, which is natural considering the diversity of language and context across various texts.

The following subsection provides an example of how distributed attention impacts a story with the same start.

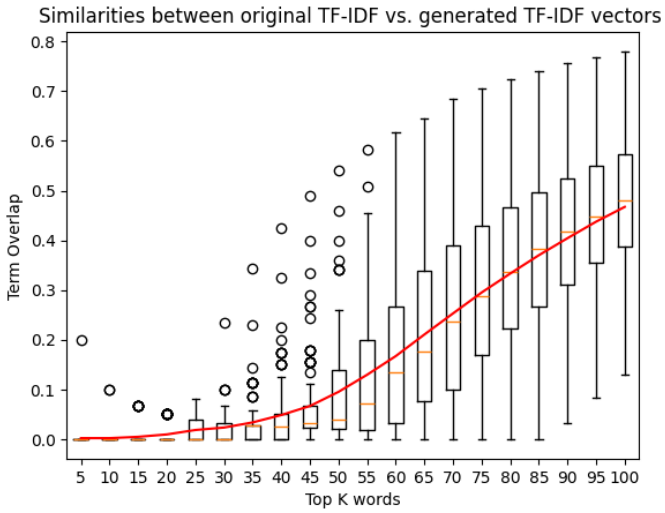

Figure 6.9: Keyword overlap between Original TF-IDF and generated TF-IDF vectors from document embedding vectors for different ranges of K

6.2.6 Case study with topic – Russia-Ukraine

In this case study, we create two stories using *DifStoryGen* with and without distributed attention. The seed document we selected covered an article from the *New York Times*, published on January 10, 2022, about the start of the Russia-Ukraine War, originally titled *"Can the West Stop Russia From Invading Ukraine?"* The word cloud of the document is reflected on the left of both the stories in Figure 6.10. Figure 6.10 (top) shows that the last document of the story generated by DifStoryGen without distributed attention was published on September 13, 2022, which was way after the start of the war. The original title of the end document in the story is *"Challenges for Russia and China Test a 'No-Limits' Friendship."*

On the other hand, Figure 6.10 (bottom), which DifStoryGen generates with distributed attention, ends with an article published on February 21, 2022, titled *"The U.S. still sees an invasion as imminent, dimming hopes for a Biden-Putin summit"*. The story ended three days before the war started on February 24, 2022.

Since the topic of the starting document was about stopping the war and involving NATO and the West, the attention mechanism stopped the story when there was no more continuation of what

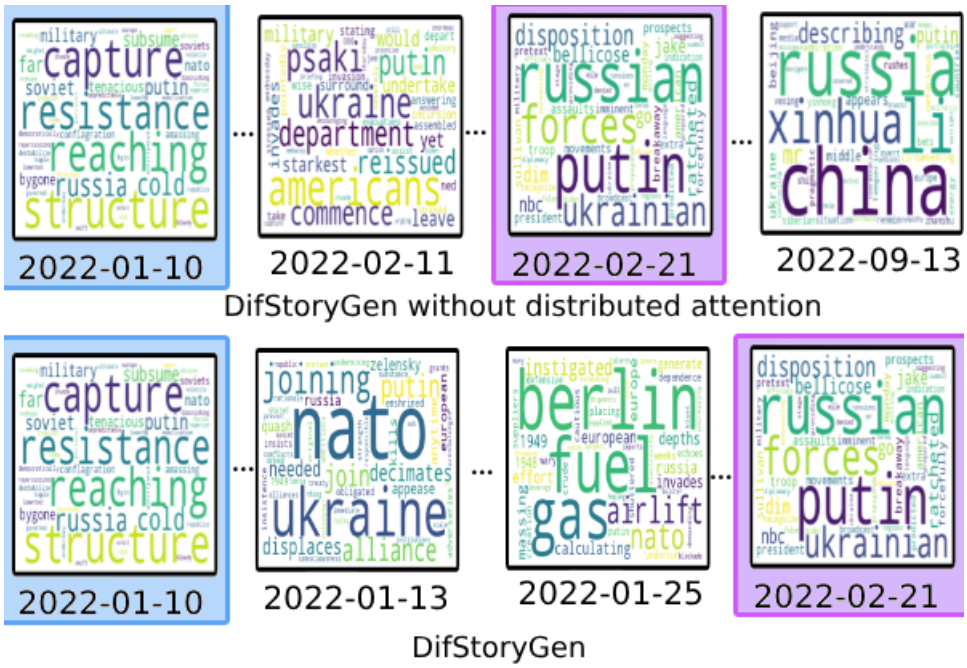

Figure 6.10: Word clouds of documents in two stories using DifStoryGen without and with distributed attention. The first documents are the same in both stories. The last document of the story generated by DifStoryGen is an intermediate document in the story without distributed attention.

had been propagated from the start. The story at the top, which did not include distributed attention, continued through the start of the war and many other documents, ending with a document seven months later.

Depending on the application, whether the user or the analyst needs to progress through a story or not, DifStoryGen can deactivate or activate its distributed attention mechanism.

6.2.7 Case study with large language model – Explanatory by LLM

This case study highlights the impact of Large Language Models (LLMs) like the Generative Pre-trained Transformer (GPT). These models understand, generate, and summarize text, exhibiting unparalleled capabilities that can radically change content creation and curation processes.

LLMs can generate concise, coherent, and engaging summaries of complex narratives fed

Table 6.5: Stories generated by different algorithms from same initial document

a) Storytelling algorithm: DifStoryGen without Attention
NYT23941(2022-01-10)→ NYT24459(2022-01-13)→ NYT25005(2022-01-18)→ NYT25770(2022-01-24)→ NYT26008(2022-01-25)→ NYT26622(2022-01-30)→ NYT11389(2022-02-02)→ NYT11852(2022-02-05)→ NYT12216(2022-02-08)→ NYT12696(2022-02-11)→ NYT13145(2022-02-14)→ NYT13234(2022-02-15)→ NYT14054(2022-02-20)→ NYT14149(2022-02-21)→ NYT14497(2022-02-24)→ NYT15080(2022-02-28)→ NYT521(2022-03-03)→ NYT785(2022-03-05)→ NYT1453(2022-03-10)→ NYT1544(2022-03-11)→ NYT2273(2022-03-16)→ NYT2644(2022-03-18)→ NYT3046(2022-03-22)→ NYT3308(2022-03-24)→ NYT3731(2022-03-27)→ NYT4175(2022-03-30)→ NYT7445(2022-04-02)→ NYT7970(2022-04-06)→ NYT8138(2022-04-07)→ NYT9071(2022-04-14)→ NYT9429(2022-04-18)→ NYT9787(2022-04-20)→ NYT10179(2022-04-23)→ NYT10396(2022-04-25)→ NYT20030(2022-05-09)→ NYT20165(2022-05-10)→ NYT21055(2022-05-16)→ NYT21468(2022-05-19)→ NYT22015(2022-05-24)→ NYT22841(2022-05-30)→ NYT28233(2022-06-12)→ NYT28764(2022-06-15)→ NYT29611(2022-06-21)→ NYT30183(2022-06-24)→ NYT30843(2022-06-29)→ NYT15412(2022-07-02)→ NYT15484(2022-07-03)→ NYT16265(2022-07-10)→ NYT16450(2022-07-12)→ NYT17542(2022-07-19)→ NYT18942(2022-07-30)→ NYT31533(2022-08-03)→ NYT32194(2022-08-08)→ NYT32601(2022-08-11)→ NYT33543(2022-08-18)→ NYT33940(2022-08-22)→ NYT34566(2022-08-27)→ NYT34979(2022-08-31)→ NYT5787(2022-09-12)→ NYT5909(2022-09-13)
b) Storytelling algorithm: DifStoryGen
NYT23941(2022-01-10)→ NYT24459(2022-01-13)→ NYT25005(2022-01-18)→ NYT25579(2022-01-21)→ NYT25830(2022-01-24)→ NYT26008(2022-01-25)→ NYT26622(2022-01-30)→ NYT11297(2022-02-02)→ NYT11852(2022-02-05)→ NYT12216(2022-02-08)→ NYT12696(2022-02-11)→ NYT13145(2022-02-14)→ NYT13357(2022-02-15)→ NYT14066(2022-02-20)→ NYT14149(2022-02-21)
c) Storytelling algorithm: BERT
NYT23941(2022-01-10)→ NYT26544(2022-01-29)→ NYT11630(2022-02-04)→ NYT28461(2022-06-14)→ NYT16873(2022-07-14)→ NYT32899(2022-08-13)→ NYT4724(2022-09-04)
d) Storytelling algorithm: doc2vec
NYT23941(2022-01-10)→ NYT24912(2022-01-18)→ NYT26567(2022-01-29)→ NYT11252(2022-02-01)→ NYT11447(2022-02-03)→ NYT12562(2022-02-10)→ NYT14358(2022-02-23)→ NYT979(2022-03-07)→ NYT1791(2022-03-12)→ NYT7288(2022-04-01)→ NYT9729(2022-04-20)→ NYT10030(2022-04-22)→ NYT21517(2022-05-19)→ NYT28379(2022-06-13)→ NYT29449(2022-06-20)→ NYT30133(2022-06-24)→ NYT15555(2022-07-05)→ NYT16545(2022-07-12)→ NYT31257(2022-08-02)→ NYT32684(2022-08-11)→ NYT4907(2022-09-06)
e) Storytelling algorithm: Entity-set based
NYT23941(2022-01-10)→ NYT24421(2022-01-13)→ NYT24772(2022-01-16)→ NYT25579(2022-01-21)→ NYT25770(2022-01-24)→ NYT25963(2022-01-25)→ NYT26358(2022-01-28)→ NYT11277(2022-02-01)→ NYT11789(2022-02-04)→ NYT11896(2022-02-06)→ NYT12640(2022-02-10)→ NYT13145(2022-02-14)→ NYT13652(2022-02-17)→ NYT14054(2022-02-20)→ NYT14189(2022-02-22)→ NYT14895(2022-02-25)→ NYT15065(2022-02-27)→ NYT495(2022-03-03)→ NYT784(2022-03-05)→ NYT1340(2022-03-09)→ NYT1734(2022-03-11)→ NYT1938(2022-03-14)→ NYT2689(2022-03-19)→ NYT3037(2022-03-22)→ NYT3587(2022-03-25)→ NYT3818(2022-03-28)→ NYT3992(2022-03-29)→ NYT7443(2022-04-02)→ NYT8003(2022-04-06)→ NYT8403(2022-04-09)→ NYT8751(2022-04-12)→ NYT9165(2022-04-15)→ NYT9276(2022-04-16)→ NYT9561(2022-04-19)→ NYT10241(2022-04-24)→ NYT10479(2022-04-26)→ NYT11125(2022-04-30)→ NYT19356(2022-05-03)→ NYT19499(2022-05-04)→ NYT19855(2022-05-07)→ NYT20542(2022-05-12)→ NYT20860(2022-05-14)→ NYT21055(2022-05-16)→ NYT21675(2022-05-20)→ NYT21807(2022-05-22)→ NYT22307(2022-05-25)→ NYT22841(2022-05-30)→ NYT27086(2022-06-02)→ NYT27224(2022-06-03)→ NYT28176(2022-06-11)→ NYT28309(2022-06-13)→ NYT28769(2022-06-15)→ NYT29314(2022-06-19)→ NYT29786(2022-06-22)→ NYT30183(2022-06-24)→ NYT30568(2022-06-28)→ NYT15412(2022-07-02)→ NYT15550(2022-07-04)→ NYT15823(2022-07-07)→ NYT16368(2022-07-11)→ NYT16498(2022-07-12)→ NYT16917(2022-07-15)→ NYT17708(2022-07-20)→ NYT18093(2022-07-23)→ NYT18149(2022-07-24)→ NYT18661(2022-07-28)→ NYT18942(2022-07-30)→ NYT31726(2022-08-04)→ NYT32122(2022-08-07)→ NYT32510(2022-08-10)→ NYT32908(2022-08-13)→ NYT33051(2022-08-15)→ NYT33317(2022-08-17)→ NYT33940(2022-08-22)→ NYT34412(2022-08-25)→ NYT34534(2022-08-26)→ NYT34979(2022-08-31)→ NYT4401(2022-09-01)→ NYT4882(2022-09-06)→ NYT5379(2022-09-08)→ NYT5814(2022-09-12)→ NYT5933(2022-09-13)→ NYT6383(2022-09-16)→ NYT6804(2022-09-20)

Table 6.6: ID, Date, and Title of News articles selected by DifStoryGen as a story from seed document 23705.

ID	Date	Title
23705	2022-01-07	The Anguish of Afghans in Sacramento
11394	2022-02-02	Over a Million Flee as Afghanistan's Economy Collapses
12804	2022-02-11	Taliban Free 2 Westerners Working for U.N., Days After Quiet Detention
9362	2022-04-17	Death Toll From Pakistani Airstrike Rises to 45, Afghan Officials Say
19060	2022-05-01	With Spate of Attacks, ISIS Begins Bloody New Chapter in Afghanistan
32859	2022-08-12	We Can't Claim Mission Accomplished': A Long Road for Afghan Refugees
34605	2022-08-27	Judge Recommends Rejecting Bid by Sept. 11 Families to Seize Frozen Afghan Funds

Figure 6.11: Word clouds of documents in a story generated by DifStoryGen to use in LLMs. This story was generated from seed document 23705 covering the events after the U.S. withdrew its forces from Afghanistan

Table 6.7: The first level of summaries generated out of 7 documents

Summaries generated by GPT model by using stories from DifStoryGen
Summary level 1
In the wake of the United States' withdrawal from Afghanistan in the summer of 2021, a series of reports have emerged, chronicling the profound impact on Afghans both within and outside their homeland. These stories, published between January and August 2022, provide a window into the varied experiences of Afghan individuals navigating a new reality post-U.S. exit. Yousafzai, a refugee who has made a new home for himself and his family in the Sacramento suburb of Elk Grove, embodies the resilience and adaptation of Afghans resettling in the U.S. Despite establishing a successful insurance business, Yousafzai and his community feel the acute pain of separation from loved onesstill in Afghanistan, compounded by a collapsing banking system that hampers their ability to provide financial support. The dire economic situation in Afghanistan has forced individuals like Najaf Akhlaqi to undertake perilous journeys in search of a better life, highlighting the desperation that drives many to flee their homeland. Meanwhile, incidents of Taliban detaining foreign and Afghan workers, despite subsequent releases, underscore the volatile security situation and the unpredictable governance by the Taliban. Abdul Sultani, another refugee who found sanctuary in London, represents the hope and determination of Afghans to rebuild their lives abroad, even as they contribute back to their families and communities still facing uncertainty in Afghanistan. Cross-border tensions between Afghanistan and Pakistan have escalated, with Pakistan urging the Taliban to control militant activities, reflecting the broader regional security implications of the Taliban's return to power. The resurgence of terrorist attacks, notably by the Islamic State's affiliate targeting civilians, signals a troubling escalation of violence, threatening the fragile peace that followed the Taliban's takeover. Abdul Wajed Ahadi's story of escape and subsequent anxiety over his family's safety back in Afghanistan encapsulates the personal toll and ongoing crisis faced by those who served alongside U.S. forces, now vulnerable to Taliban reprisals. Together, these reports paint a complex picture of hope, resilience, despair, and ongoing challenges for Afghans, as they navigate the aftermath of a two-decade-long conflict and the sudden shift in their country's fate.

Table 6.8: The second level of summaries generated out level one summarize by GPT model total 129 words

Summaries generated by GPT model by using stories from DifStoryGen
Summary level 2
Following the U.S. withdrawal from Afghanistan in 2021, stories from January to August 2022 reveal the diverse challenges and adaptations of Afghans, both within their country and as refugees abroad. Afghans like Yousafzai in Sacramento and Abdul Sultani in London have built new lives, yet remain deeply connected to those they've left behind, facing a collapsed economy and limited means to offer support. The economic desperation has pushed individuals to dangerous migrations, while the Taliban's unpredictable governance raises concerns over security and freedom. Cross-border tensions with Pakistan and a resurgence of terrorist attacks underscore the region's volatile security landscape. Despite these adversities, stories of hope and resilience emerge as Afghans abroad strive to rebuild and support their communities, reflecting a complex narrative of struggle, adaptation, and the enduring spirit of the Afghan people in the face of ongoing crisis.

Table 6.9: The third level of summaries generated out level two summarize by GPT model total 70 words

Summaries generated by GPT model by using stories from DifStoryGen
Summary level 3
After the U.S. withdrawal from Afghanistan in 2021, Afghans face varied challenges. Resettled Afghans like Yousafzai and Abdul Sultani adapt to new lives abroad but grapple with concerns for relatives in an economically crippled Afghanistan. Migrations, Taliban governance issues, regional security tensions, and terrorist attacks highlight the ongoing instability. Despite hardships, the resilience and hope among Afghans persist, showcasing their efforts to rebuild and support their communities amidst the crisis.

with a series of documents produced by a storytelling algorithm. This ability revolutionizes how content is consumed and curated, making it easier for users to access and engage with a vast collection of stories without dedicating the time required for long-form reading. Hence, this enhances discoverability and engagement, offering quick and informative summaries for content consumers and providing content creators with an effective tool to present their work to wider audiences.

The integration of LLMs into storytelling platforms marks the beginning of a new chapter in content consumption, where the efficiency of algorithmic summarization is integrated with storytelling algorithms. Our case study includes a demonstration, represented in Figure 6.11 and Table 6.6, showing how a document chain generated by the DifStoryGen algorithm can be effectively summarized at three different levels. The results of summarizing a story in three different levels by a GPT model are present in Tables 6.7, 6.8, and 6.9. These results show the models' capability to maintain text coherency and continuity across varying summarization levels, illustrating the potential of LLMs to enhance the storytelling domain.

In this scenario, a chain of documents generated by DifStoryGen has different lengths, ranging from 151 to 348 words. The analysis of entity overlap, explicitly focusing on the top 75 keywords across each document and each summarization level, revealed overlaps of 0.19, 0.11, 0.08, 0.09, 0.13, 0.17, and 0.08 for the first level, indicating an acceptable level of keyword retention for a summary with a size of 342 words. The second level of summarization showed overlaps of 0.09, 0.04, 0.05, 0.08, 0.09, 0.08, and 0.08 as the size of the summary decreased to 139 words, while the third level presented overlaps of 0.09, 0.05, 0.04, 0.07, 0.04, 0.05, and 0.07, with a more brief summary of just 70 words.

The stories produced by DifStoryGen displayed significant keyword overlap with the GPT-generated summaries: 0.85 with the first level, 0.52 with the second, and 0.41 with the third. This suggests a strong coherence between the original documents and the most detailed GPT summary, with reduced overlap as the summaries become more concise.

The keywords overlap with different documents in the story, and summary levels decrease with a similar ratio, indicating the importance of each document generated by DifStoryGen.

This demonstrates that no redundant text exists in the story and highlights the capability of both DifStoryGen and GPT models to generate coherent and relevant summaries of narratives.

6.3 Conclusions

Our proposed model introduces a storytelling framework, DifStoryGen, for constructing a story chain from an archive of timestamped documents. This framework leverages a contextual embedding mechanism, a diffusion-based generative model, and a distributed attention technique to generate coherent and evolving stories. Experiments demonstrate the effectiveness of DifStoryGen in building stories from a corpus and incorporating it in downstream tasks such as clusterings and classifications.

Chapter 7

Concluding Remarks

Current research focuses on enhancing storytelling algorithm techniques to produce more coherent and cohesive narratives. This trend extends significant opportunities across various domains. (1) In intelligence analysis, storytelling algorithms play a crucial role in identifying potential threats by uncovering hidden connections between entities and events. (2) In the field of scientific research, these algorithms assist in discovering previously unrecognized evolutions among studies. (3) Moreover, within trend analysis, they are instrumental in forecasting future events by tracking the evolution of a particular entity or event. The growing field of storytelling algorithms promises to improve classification and clustering in downstream tasks, highlighting the critical role of storytelling models in different domains.

This dissertation has achieved notable progress in tackling existing challenges and filling the research gaps in storytelling algorithms. It introduces innovative techniques, including utilizing contextual embedding from large language models, presenting the role-based aspects of words, bridging data collection gaps with generative diffusion models, and employing distributed attention mechanisms to effectively control a story's narrative. These advancements offer a new perspective on storytelling and language model utilization for future research.

7.1 Key achievements of this dissertation

In this dissertation, I introduce an innovative approach to storytelling by leveraging advanced computational models to enhance narrative coherence and thematic consistency. The following points present the main contributions of this work:

1. **Contemporary Context with BERT:** Utilizing the Bidirectional Encoder Representations

from Transformers (BERT), this approach captures the contemporary context of every word, significantly improving the ability to trace a story's evolution over conventional models.

2. **Role Identification of Words:** By analyzing the localized distribution of words within documents and the vector similarities of their neighbors, I create localized, cluster-based contexts that define the roles of words within the narrative, adding depth and precision to the story's structure.

3. **Lesser number of user-settable parameters:** My models require less user intervention, with just a single set of user-settable parameters to be defined. This simplicity enables ease of use without compromising the algorithm's effectiveness.

4. **Dynamic Storytelling Mechanism:** The proposed algorithms dynamically generate narratives by integrating neural network embeddings with a newly designed storytelling search process, ensuring seamless temporal progression without the need for any post-processing techniques.

5. **Intermediate Document Generation:** To bridge gaps between documents, DifStoryGen employs a diffusion-based document generation model. This innovative component generates intermediary texts, enabling the discovery of omitted concepts in a story even when direct connections are not apparent in the existing corpus.

6. **Distributed Attention Similarity for Theme Consistency:** In the storytelling process, DifStoryGen uses distributed attention similarity to maintain thematic consistency, ensuring that the evolving narrative remains coherent and focused. This mechanism engages all documents belonging to a story in the search process based on their contribution to the story's core.

7. **Smooth Transition Between Documents:** The transition between documents is facilitated by evaluating both embedding similarities and the overlap of generated keywords, which are derived not directly from the document's text but through an encoder-decoder model.

This dual similarity approach ensures a natural flow and logical progression between story segments.

Together, these features represent a significant advancement in automated storytelling algorithms. By addressing the limitations of previous models and introducing mechanisms for deeper contextual analysis, role identification, generative diffusion model, distributed attention similarity, and dynamic narrative construction, my dissertation sets a new standard for automated storytelling systems.

7.2 Limitations

Storytelling models are great tools in data mining and machine learning. However, like other models in these fields, they have some limitations. If we want to use these models to extract knowledge from text, track the evolution of an entity or event, or integrate them into other downstream tasks, it's essential to understand their limitations.

- **Parameter sensitivity:** Storytelling models often depend on user-defined parameters, which can be challenging to optimize and may need to adapt better to different datasets or applications. I introduced two storytelling models that are less dependent on user-defined parameters in this dissertation. However, having an expert in the field is still crucial to fine-tuning these parameters for specific tasks. Incorrect parameter settings can result in poorly constructed narratives.

- **Limited scalability:** Many storytelling models have difficulty managing large datasets due to computational limitations or inefficiencies in processing large volumes of text. The CoRBS and DifStoryGen models use pre-trained large language models to create contextual embeddings, which reduces time complexity and improves scalability. However, the conditional diffusion model and role generator still pose challenges. We recommend selecting a diverse, representative subset of data from extensive collections for the training phase in these models to make them more compatible for a large scale of data collection.

- **Ethical and bias challenges:** The proposed models use large language models that may have ethical and other types of biases inherent in their training. These biases can be transferred to any task utilizing these embedding spaces. To mitigate these biases, we can employ techniques like normalizing the space by adjusting the vectors or retraining these large language models using diverse data.

- **Ambiguity in natural language:** In natural language processing, ambiguity presents a significant challenge for storytelling algorithms. Words can have multiple meanings based on their context, making it difficult to identify the actual meaning in a narrative. Additionally, when trying to analyze or trace the evolution of an entity, collecting similar words or synonyms will improve the task. These issues can impact the accuracy and coherence of storytelling models, as they may struggle to distinguish between different senses of a word or appropriately handle entities with overlapping characteristics. Addressing these challenges is crucial for developing robust storytelling algorithms that can accurately capture and convey narratives.

- **Incorporating knowledge base:** The CoRBS and DifStoryGen models do not utilize a knowledge base for general tasks, which is a limitation. Incorporating additional information from a knowledge base could enhance these models for specific applications. We recommend using a knowledge base to optimize these models more efficiently for specialized tasks.

Addressing these limitations is crucial for improving the effectiveness and applicability of storytelling models, enhancing their ability to generate coherent, engaging, and accurate narratives from diverse document collections.

7.3 Future works

My research highlights the abilities of storytelling algorithms to construct more cohesive and coherent document chains, which are crucial in analyzing the evolution of entities across a narrative. The methods and components described in my dissertation significantly enhance the

utility of storytelling models for various downstream tasks – clusterings, classifications, and predictions–, proposing a robust framework for narrative analysis. Despite these advancements, the field of automated storytelling algorithms, like all areas of scientific research, offers continuous opportunities for further development and improvement. Potential directions for future research and improvement are suggested:

- **Segment-Based Storytelling Algorithms:** Storytelling algorithms can achieve a finer narrative by analyzing individual paragraphs or text segments and identifying connections. This approach allows for more reliable text analysis, potentially leading to more applicable and coherent stories by connecting segments in different documents.

- **Enhanced Contextual Understanding:** Incorporating other aspects of language, such as sentiment analysis, entity recognition, and more, can significantly improve narrative coherence and the tracking of entity evolution. Incorporating these linguistic features could lead to a more profound understanding of the text, enabling algorithms to construct more meaningful and engaging narratives.

- **Scalability and Efficiency:** Enhancing storytelling models' scalability and computational efficiency are crucial for handling larger datasets. This improvement would make narrative analysis more feasible for big data applications.

- **Integration of Multimodal Data:** By incorporating various data types, including images, videos, and audio, storytelling algorithms can offer a more enriched narrative outcome. This integration could transform storytelling from a purely textual analysis into a comprehensive multimodal narrative construction.

- **Advanced Language Models Integration:** Leveraging the latest developments in language models and natural language processing technologies can further improve the precision and depth of the narratives generated. Such integration promises to enhance the storytelling process, making it more accurate and detailed.

- **Tuning Parameters for Storytelling Algorithms:** For future work, we recommend analyzing different text sources to tune the parameters used in the CoRBS and DifStoryGen models. These algorithms employ various parameters to balance different language aspects, such as contextual semantics and role aspects of terms. Since these parameters vary across different types of texts, such as news articles, medical literature, or cyber-security reports, it's crucial to identify the appropriate range of values for each specific context. A detailed study examining how these parameters should be adjusted for different domains would enhance the effectiveness and adaptability of storytelling algorithms in diverse applications.

- **Expert Analysis for Future Storytelling Algorithm Research:** For future work, we suggest incorporating expert analysis into evaluating stories generated by the CoRBS and DifStoryGen models. While quantitative metrics are currently used to assess the quality of these narratives, feedback from subject matter experts in different fields could identify areas where these models are underperforming. Researchers can improve storytelling algorithms by addressing these shortcomings, ensuring they generate more accurate, coherent, and contextually appropriate narratives across various domains.

- **Ethical and Bias Considerations:** It is necessary to address ethical issues and biases within storytelling algorithms to ensure the fairness and inclusivity of narratives. By actively mitigating biases and ethical concerns, the field can ensure that stories reflect diverse and balanced views in describing events and entities.

Addressing these potential research and development areas expands storytelling algorithms' technical capabilities and application domains. Future progress in these directions promises to enrich automated storytelling, making it more reliable, inclusive, and effective in generating coherent and comprehensive narratives for various purposes.

References

[1] The new york times article search api. `https://developer.nytimes.com/apis`.

[2] Amr Ahmed, Qirong Ho, Jacob Eisenstein, Eric Xing, Alexander J Smola, and Choon Hui Teo. Unified analysis of streaming news. In *Proceedings of the 20th international conference on World wide web*, pages 267–276, 2011.

[3] Jean Aitchison. *Language change: Progress or decay?* Cambridge university press, 2001.

[4] Emilia Apostolova and R Andrew Kreek. Training and prediction data discrepancies: Challenges of text classification with noisy, historical data. *arXiv preprint arXiv:1809.04019*, 2018.

[5] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

[6] Roberto Camacho Barranco, Raimundo F Dos Santos, M Shahriar Hossain, and Monika Akbar. Tracking the evolution of words with time-reflective text representations. In *2018 IEEE international conference on big data (big data)*, pages 2088–2097. IEEE, 2018.

[7] Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. Analysing word meaning over time by exploiting temporal random indexing. *Analysing word meaning over time by exploiting temporal Random Indexing*, pages 38–42, 2014.

[8] Amin Beheshti, Alireza Tabebordbar, and Boualem Benatallah. istory: Intelligent storytelling with social data. In *Companion Proceedings of the Web Conference 2020*, pages 253–256, 2020.

[9] Lauren Bradel, Jessica Zeitz Self, Alex Endert, M Shahriar Hossain, Chris North, and Naren Ramakrishnan. How analysts cognitively "connect the dots". In *2013 IEEE International Conference on Intelligence and Security Informatics*, pages 24–26. IEEE, 2013.

[10] Jan Brophy and David Bawden. Is google enough? comparison of an internet search engine with academic library resources. In *Aslib proceedings*, volume 57, pages 498–512. Emerald Group Publishing Limited, 2005.

[11] Roberto Camacho Barranco, Arnold P Boedihardjo, and M Shahriar Hossain. Analyzing evolving stories in news articles. *International Journal of Data Science and Analytics*, 8:241–256, 2019.

[12] Mario Casillo, Dajana Conte, Marco Lombardi, Domenico Santaniello, and Carmine Valentino. Recommender system for digital storytelling: A novel approach to enhance cultural heritage. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10-15, 2021, Proceedings, Part VII*, pages 304–317. Springer, 2021.

[13] Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. *arXiv preprint arXiv:2303.11396*, 2023.

[14] Haoxing Chen, Zhuoer Xu, Zhangxuan Gu, Jun Lan, Xing Zheng, Yaohui Li, Changhua Meng, Huijia Zhu, and Weiqiang Wang. Diffute: Universal text editing diffusion model.

[15] Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv preprint arXiv:2208.04202*, 2022.

[16] Yuhua Chen. An automatic storytelling system based on natural language processing, 2021.

[17] 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.

[18] Valerio Di Carlo, Federico Bianchi, and Matteo Palmonari. Training temporal word embeddings with a compass. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6326–6334, 2019.

[19] Chris Donahue, Mina Lee, and Percy Liang. Enabling language models to fill in the blanks. *arXiv preprint arXiv:2005.05339*, 2020.

[20] Christos Faloutsos, Kevin S McCurley, and Andrew Tomkins. Fast discovery of connection subgraphs. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 118–127, 2004.

[21] Lujun Fang, Anish Das Sarma, Cong Yu, and Philip Bohannon. Rex: explaining relationships between entity pairs. *arXiv preprint arXiv:1111.7170*, 2011.

[22] Ahnaf Farhan, Roberto Camacho Barranco, Mahmud Shahriar Hossain, and Monika Akbar. Diffusion-based temporal word embeddings. In *SDU@ AAAI*, 2021.

[23] Tong Gao, Jessica R Hullman, Eytan Adar, Brent Hecht, and Nicholas Diakopoulos. Newsviews: an automated pipeline for creating custom geovisualizations for news. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 3005–3014, 2014.

[24] Samira Ghodratnama, Amin Beheshti, Mehrdad Zakershahrak, and Fariborz Sobhanmanesh. Intelligent narrative summaries: From indicative to informative summarization. *Big Data Research*, 26:100257, 2021.

[25] Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. Discovering diverse and salient threads in document collections. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 710–720, 2012.

[26] Hila Gonen, Ganesh Jawahar, Djamé Seddah, and Yoav Goldberg. Simple, interpretable and stable method for detecting words with usage change across corpora. *arXiv preprint arXiv:2112.14330*, 2021.

[27] Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*, 2022.

[28] Ryan Gould. Best search engine alternatives, January 2019.

[29] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[30] Wanrong Gu, Shoubin Dong, and Mingquan Chen. Personalized news recommendation based on articles chain building. *Neural Computing and Applications*, 27:1263–1272, 2016.

[31] William L Hamilton, Jure Leskovec, and Dan Jurafsky. Cultural shift or linguistic drift? comparing two computational measures of semantic change. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2016, page 2116. NIH Public Access, 2016.

[32] William L Hamilton, Jure Leskovec, and Dan Jurafsky. Diachronic word embeddings reveal statistical laws of semantic change. *arXiv preprint arXiv:1605.09096*, 2016.

[33] BS Harish and MB Revanasiddappa. A comprehensive survey on various feature selection methods to categorize text documents. *International Journal of Computer Applications*, 164(8):1–7, 2017.

[34] Richard F Helm and Malcolm Potts. Algorithms for storytelling. *IEEE Transactions on Knowledge and Data Engineering*, 20(6):1, 2008.

[35] Gerhard Heyer, Florian Holz, and Sven Teresniak. Change of topics over time-tracking topics by their change of meaning. *KDIR*, 9:223–228, 2009.

[36] Martin Hilpert and Stefan Th Gries. Assessing frequency changes in multistage diachronic corpora: Applications for historical corpus linguistics and the study of language acquisition. *Literary and Linguistic Computing*, 24(4):385–401, 2009.

[37] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[38] M Shahriar Hossain, Patrick Butler, Arnold P Boedihardjo, and Naren Ramakrishnan. Storytelling in entity networks to support intelligence analysts. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1375–1383, 2012.

[39] M Shahriar Hossain, Joseph Gresock, Yvette Edmonds, Richard Helm, Malcolm Potts, and Naren Ramakrishnan. Connecting the dots between pubmed abstracts. *PloS one*, 7(1):e29509, 2012.

[40] Mahmud Shahriar Hossain, Christopher Andrews, Naren Ramakrishnan, and Chris North. Helping intelligence analysts make connections. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[41] Jessica Hullman, Nicholas Diakopoulos, and Eytan Adar. Contextifier: automatic generation of annotated stock visualizations. In *Proceedings of the SIGCHI Conference on human factors in computing systems*, pages 2707–2716, 2013.

[42] Li-Ping Jing, Hou-Kuan Huang, and Hong-Bo Shi. Improved feature selection approach tfidf in text mining. In *Proceedings. International Conference on Machine Learning and Cybernetics*, volume 2, pages 944–946. IEEE, 2002.

[43] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

[44] Md Abdul Kader, Arnold Priguna Boedihardjo, and Mahmud Shahriar Hossain. F2context: how to extract holistic contexts of persons of interest for enhancing exploratory analysis. *Knowledge and Information Systems*, 61(1):363–396, 2019.

[45] Brian Felipe Keith Norambuena and Tanushree Mitra. Narrative maps: An algorithmic approach to represent and extract information narratives. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW3):1–33, 2021.

[46] Tom Kenter, Melvin Wevers, Pim Huijnen, and Maarten De Rijke. Ad hoc monitoring of vocabulary shifts over time. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 1191–1200, 2015.

[47] Georgia Koutrika, Lei Liu, and Steve Simske. Generating reading orders over document collections. In *2015 IEEE 31st International Conference on Data Engineering*, pages 507–518. IEEE, 2015.

[48] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343, 2022.

[49] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[50] Yuqing Mao, Haifeng Shen, and Chengzheng Sun. Episose: An epistemology-based social search framework for exploratory information seeking. In *Human-Computer Interaction: Second IFIP TC 13 Symposium, HCIS 2010, Held as Part of WCC 2010, Brisbane, Australia, September 20-23, 2010. Proceedings*, pages 211–222. Springer, 2010.

[51] Rada Mihalcea and Vivi Nastase. Word epoch disambiguation: Finding how words change over time. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 259–263, 2012.

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

[53] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[54] 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.

[55] Sunny Mitra, Ritwik Mitra, Suman Kalyan Maity, Martin Riedl, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. An automatic approach to identify word sense changes in text media across timescales. *Natural Language Engineering*, 21(5):773–798, 2015.

[56] Syrielle Montariol, Matej Martinc, Lidia Pivovarova, et al. Scalable and interpretable semantic change detection. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies*. The Association for Computational Linguistics, 2021.

[57] Antonio Moreno and Teófilo Redondo. Text analytics: the convergence of big data and artificial intelligence. *IJIMAI*, 3(6):57–64, 2016.

[58] Usman Naseem, Imran Razzak, Shah Khalid Khan, and Mukesh Prasad. A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models. *Transactions on Asian and Low-Resource Language Information Processing*, 20(5):1–35, 2021.

[59] Yue Ning, Sathappan Muthiah, Ravi Tandon, and Naren Ramakrishnan. Uncovering news-twitter reciprocity via interaction patterns. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 1–8, 2015.

[60] Alireza Nouri and Mahmud Shahriar Hossain. Corbs: A dynamic storytelling algorithm using a novel contextualization approach for documents utilizing bert features. 2023.

[61] Pascal Oser, Rens W van der Heijden, Stefan Lüders, and Frank Kargl. Risk prediction of iot devices based on vulnerability analysis. *ACM Transactions on Privacy and Security*, 25(2):1–36, 2022.

[62] 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.

[63] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[64] Qing Ping and Chaomei Chen. Litstoryteller+: an interactive system for multi-level scientific paper visual storytelling with a supportive text mining toolbox. *Scientometrics*, 116(3):1887–1944, 2018.

[65] Maria Polo, Umberto Dello Iacono, Giuseppe Fiorentino, and Anna Pierri. A social network analysis approach to a digital interactive storytelling in mathematics. *Journal of e-Learning and Knowledge Society*, 15(3):239–250, 2019.

[66] E Popoff, M Besada, JP Jansen, S Cope, and Steve Kanters. Aligning text mining and machine learning algorithms with best practices for study selection in systematic literature reviews. *Systematic reviews*, 9(1):1–12, 2020.

[67] Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. *arXiv preprint arXiv:2010.05906*, 2020.

[68] Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. Cold decoding: Energy-based constrained text generation with langevin dynamics. *Advances in Neural Information Processing Systems*, 35:9538–9551, 2022.

[69] Shouning Qu, Sujuan Wang, and Yan Zou. Improvement of text feature selection method based on tfidf. In *2008 International Seminar on Future Information Technology and Management Engineering*, pages 79–81. IEEE, 2008.

[70] Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. Learning causality for news events prediction. In *Proceedings of the 21st international conference on World Wide Web*, pages 909–918, 2012.

[71] JT Rigsby and Daniel Barbará. Storytelling with signal injection: Focusing stories with domain knowledge. In *Machine Learning and Data Mining in Pattern Recognition: 14th International Conference, MLDM 2018, New York, NY, USA, July 15-19, 2018, Proceedings, Part II 14*, pages 425–439. Springer, 2018.

[72] Guy D Rosin, Eytan Adar, and Kira Radinsky. Learning word relatedness over time. *arXiv preprint arXiv:1707.08081*, 2017.

[73] Eyal Sagi, Stefan Kaufmann, and Brady Clark. Tracing semantic change with latent semantic analysis. *Current methods in historical semantics*, 73:161–183, 2011.

[74] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[75] Gholam Reza Samadzadeh, Tahereh Rigi, and Ali Reza Ganjali. Comparison of four search engines and their efficacy with emphasis on literature research in addiction (prevention and treatment). *International Journal of High Risk Behaviors & Addiction*, 1(4):166, 2013.

[76] Ishrat Rahman Sami. Automatic contextual storytelling in a natural language corpus. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3249–3252, 2020.

[77] Dafna Shahaf and Carlos Guestrin. Connecting the dots between news articles. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–632, 2010.

[78] Dafna Shahaf, Carlos Guestrin, Eric Horvitz, and Jure Leskovec. Information cartography. *Communications of the ACM*, 58(11):62–73, 2015.

[79] Manu Shukla, Raimundo Dos Santos, Feng Chen, and Chang-Tien Lu. Discrn: A distributed storytelling framework for intelligence analysis. *Big data*, 5(3):225–245, 2017.

[80] Mei Si. Facilitate knowledge exploration with storytelling. *Procedia Computer Science*, 88:224–231, 2016.

[81] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[82] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[83] Tim Soulo. Rethinking conventional keyword research, January 2023.

[84] Caroline Suen, Sandy Huang, Chantat Eksombatchai, Rok Sosic, and Jure Leskovec. Nifty: a system for large scale information flow tracking and clustering. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1237–1248, 2013.

[85] Xuri Tang, Weiguang Qu, and Xiaohe Chen. Semantic change computation: A successive approach. *World Wide Web*, 19:375–415, 2016.

[86] J Teitelbaum. An improved forensic science information search. *Forensic Science Review*, 27(1):41–52, 2015.

[87] Mikhail Tikhomirov and Boris Dobrov. News timeline generation: Accounting for structural aspects and temporal nature of news stream. In *International Conference on Data Analytics and Management in Data Intensive Domains*, pages 267–280. Springer, 2017.

[88] Cagri Toraman and Fazli Can. Discovering story chains: A framework based on zigzagged search and news actors. *Journal of the Association for Information Science and Technology*, 68(12):2795–2808, 2017.

[89] Piek Vossen, Tommaso Caselli, and Roxane Segers. A narratology-based framework for storyline extraction. *Computational Analysis of Storylines: Making Sense of Events*, 125:125–140, 2021.

[90] Kodzo Wegba, Aidong Lu, Yuemeng Li, and Wencheng Wang. Interactive movie recommendation through latent semantic analysis and storytelling. *arXiv preprint arXiv:1701.00199*, 2017.

[91] Mark A Williams, Sumi Dey, Roberto Camacho Barranco, Sheikh Motahar Naim, M Shahriar Hossain, and Monika Akbar. Analyzing evolving trends of vulnerabilities in national vulnerability database. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3011–3020. IEEE, 2018.

[92] Sarah E Worth. Storytelling and narrative knowing: An examination of the epistemic benefits of well-told stories. *Journal of Aesthetic Education*, 42(3):42–56, 2008.

[93] Chunzi Wu, Bin Wu, and Bai Wang. Event evolution model based on random walk model with hot topic extraction. In *International Conference on Advanced Data Mining and Applications*, pages 591–603. Springer, 2016.

[94] Dani Yogatama, Chong Wang, Bryan R Routledge, Noah A Smith, and Eric P Xing. Dynamic language models for streaming text. *Transactions of the Association for Computational Linguistics*, 2:181–192, 2014.

[95] George Yule. *The study of language*. Cambridge university press, 2022.

[96] Chenhan Zhang, Qingpeng Zhang, Shui Yu, JQ James, and Xiaozhuang Song. Complicating the social networks for better storytelling: An empirical study of chinese historical text and novel. *IEEE Transactions on Computational Social Systems*, 8(3):754–767, 2021.

[97] Xuchao Zhang, Zhiqian Chen, Weisheng Zhong, Arnold P Boedihardjo, and Chang-Tien Lu. Storytelling in heterogeneous twitter entity network based on hierarchical cluster routing. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 1522–1531. IEEE, 2016.

[98] Xianshu Zhu and Tim Oates. Finding story chains in newswire articles using random walks. *Information Systems Frontiers*, 16:753–769, 2014.

Curriculum Vitae

Alireza Pasha Nouri earned a bachelor's degree in information technology and a master's in artificial intelligence in 2011 and 2014, respectively. Following his graduate studies, Alireza joined the industry as a software engineer in a startup in the San Fransico Bay Area to gain more skills and experience for four years.

In 2019, Alireza joined the doctoral program in computer science at the University of Texas at El Paso. He joined the Discovery Analytic Laboratory, where he worked under the supervision of Dr. M. Shahriar Hossain. While in the doctoral program, he took different roles as a research assistant, teaching assistant, and instructor in the Department of Computer Science. This dual commitment to research and education has helped Alireza to develop his research and teaching skills.

Alireza received scholarships throughout his academic journey, including government scholarships for his bachelor's and master's degrees and the Dean of Engineering Department award during his doctoral studies.

During his Ph.D. program, Alireza's experience was further enhanced by joining Wells Fargo & Company as a data scientist intern in the R& D department for two years. This position allowed him to apply his academic knowledge to real-world problems, preparing him for his future role as a full-time researcher with the company after he completes his Ph.D.

His dissertation, titled "Dynamic Storytelling Algorithms Using Contextual Aspects of a Large Language Model," supervised by Dr. Hossain, exhibits Alireza's innovative approach to artificial intelligence. Through this work, he contributes to the evolving domain of AI, specifically in developing algorithms that enhance the capability and efficiency of large language models in storytelling.

Permanent address: 1700 Hawthorne St

El Paso, Texas, United States 79902