2019-01-01

Time-Reflective Text Representations for Semantic Evolution Tracking and Trend Analytics

Roberto Camacho Barranco
University of Texas at El Paso

TIME-REFLECTIVE TEXT REPRESENTATIONS

FOR SEMANTIC EVOLUTION TRACKING

AND TREND ANALYTICS

ROBERTO CAMACHO BARRANCO

Doctoral Program in Computer Science

APPROVED:

M. Shahriar Hossain, Ph.D., Chair

Monika Akbar, Ph.D.

Christopher Kiekintveld, Ph.D.

Saurav Kumar, Ph.D.

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

to my son, Matías, to my wife, and to my parents

with love

TIME-REFLECTIVE TEXT REPRESENTATIONS FOR SEMANTIC EVOLUTION TRACKING AND TREND ANALYTICS

by

ROBERTO CAMACHO BARRANCO

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

December 2019

Acknowledgments

I would like to express my sincere gratitude to my advisor and committee chair, Dr. M. Shahriar Hossain. Throughout my doctorate, Dr. Hossain provided continuous support and guidance, which not only made this dissertation possible but also made me grow as a scientist and as a human being. I am very grateful for the opportunity of working with such an admirable person. Besides my advisor, I would also like to thank my committee co-chair Dr. Monika Akbar. Her support, technical expertise, and kind words were a significant boon throughout the last two years of my doctorate.

Thank you very much to the rest of my thesis committee: Dr. Christopher Kiekintveld and Dr. Saurav Kumar, for their insightful comments and encouragement, and for expressing their concerns and ideas in terms of the direction my research should take. Special thanks to Dr. Patricia J. Teller, who encouraged me to pursue a doctoral degree and who has been an extraordinary mentor since I met her.

I would not be where I am today without the fantastic support, encouragement, and love from my son Matias, my wife Vanezza, my parents, siblings, extended family, and friends. Thank you for always believing in me.

This material is based upon work supported by the U.S. Army Engineering Research and Development Center under Contract No. W9132V-15-C-0006. This work was also supported in part by the National Science Foundation under Grant No. HRD-1242122. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by the National Science Foundation grant number ACI-1548562, through the use of Bridges at Pittsburgh Supercomputing Center and Comet at San Diego Supercomputer Center, via allocation TG-CCR190030.

Abstract

The extraction of significant, relevant, and useful trends from massive document collections, such as a streaming newswire or scientific publications, is a challenging and significant problem in many different fields, including intelligence analysis, recommendation systems, and scientific research. However, techniques that tackle trend analytics of such large text corpora are limited because research that addresses the temporal nature of these publications is still in its early stages. In this work, we first show that it is possible to capture the evolution of a story (or trend) by connecting the dots between different documents in a text corpus. The observed results indicate that it is possible to transfer the idea of capturing evolution from a story level to a more general language-model level. Thus, we introduce a preliminary time-reflective frequency-based representation, which can capture the semantic evolution of a language model over time while being robust against the uncertainty and noise present in the real-world data. This preliminary representation has some shortcomings, including high dimensionality and lack of extensibility, which limit its potential use with trend-analytics techniques. We solve the shortcomings of the frequency-based representation by proposing a diffusion-based temporal word embedding model. The proposed technique generates low-dimensional word embeddings that emulate the temporal semantic changes observed in the frequency-based time-reflective representation. The proposed low-dimensional representation is suitable for trend-analytics algorithms. We apply several sequential modeling techniques on the temporal word embeddings to predict how the embedding space will look like in the future based on the current trends. We exploit the generated trend models for automatic hypothesis generation by finding potential future relationships between terms. The applications explored in this work include high impact areas such as *intelligence analysis* and *cybersecurity*, with analyses that study possible associations between malicious entities, as well as *medical sciences*, potentially discovering unexplored relations between substances and diseases.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

There is an overwhelming number of text documents published every day on the Internet, creating an endless wealth of information. Until recently, data mining researchers have focused mainly on the natural language aspects of these massive data sets, which include topic modeling [17], dynamic text clustering [140], and lower-dimensional representations of corpora [89, 11, 111, 149]. However, researchers have not exploited the dynamic nature of unstructured data to its full potential due to the compute- and resource-intensive requirements of temporal analyses. In particular, there is a need for new algorithms and systems that will allow us to mine the evolutionary aspects of text data.

Traditionally, temporal text models consider discrete chunks of documents for each time unit, e.g., a year [48, 74, 70]. The recent development of word embedding techniques has allowed the generation of low-dimensional vector space representations from document collections. These techniques have been extended to include the temporal dimension, via regression and alignment of single-timestamp models [70, 74, 48, 158]. These methods result in different vector spaces for each timestamp, and the process of aligning the vectors over time derives in unreliable results.

The state-of-the-art methods use Kalman filters and simulate Brownian motion to take into account all of the documents in a corpus while training the model, generating a unified multi-dimensional vector space [11, 111, 148]. In Chapter 4, we show that these methods suffer from far-sightedness, capturing only long-term changes in the context of a word. For example, these methods can identify that the word *computer* referred to a person that performed calculations in the 1900s, while now it is closer to meaning a *technological device* [11]. However, it would not be possible to detect that *computer* was closer in meaning

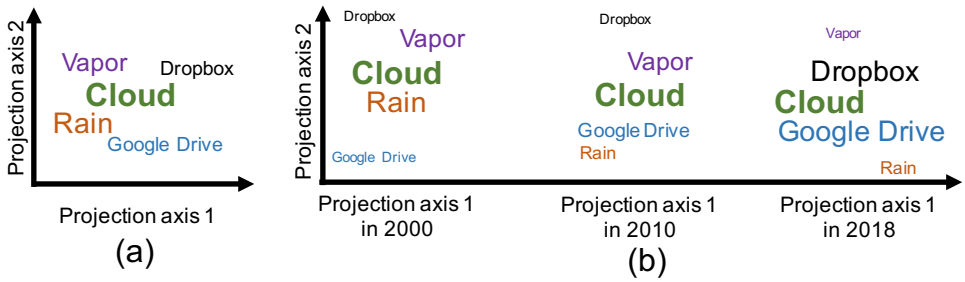


Figure 1.1: Difference between static representations and time-reflective representations ©2018 IEEE.

to *desktop* five years ago, and now it is closer in meaning to *cloud*.

The focus of this work is on studying time-reflective representations of natural language elements, e.g., entities, words, and phrases. We achieve this objective by introducing a time-aware vector-space model with strong predictive power that reduces the effect of the uncertainty and noise inherent to real-world data. This representation consists of a set of vectors for every word in a corpus over all the timestamps available (e.g., year or day).

A time-reflective vector space model enables tracking the neighborhood (or context) of each word over time to study the evolution of a concept. Figure 1.1 illustrates how using this approach would reveal how the word *cloud* changed its meaning over time, going from *condensed vapor* to a *technological* term. Figure 1.2 is another example that illustrates the expected context of the entity *Obama* as it evolves. The context changes over time based on the different stages in the life of Barack Obama, from studying at Harvard to being the President of the United States.

The main objectives of our research are:

- to define a robust low-dimensional time-reflective representation of text that captures the short- and long-term semantic evolution of a word.

- to use this representation to model the trajectory trends of the context of a word and generate predicted embeddings based on the observed trajectories.

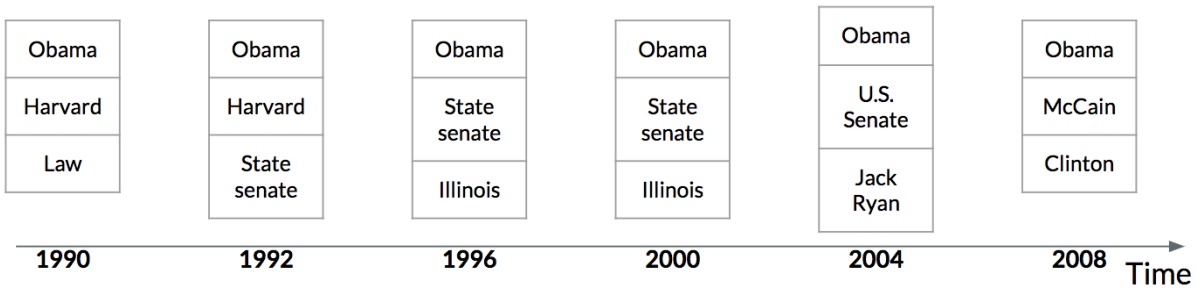


Figure 1.2: Expected neighborhood of the word Obama over time.

- to leverage these trend models to enhance the automatic generation of scientific hypotheses for literature-based discovery.

We focus on several high impact applications:

- *Intelligence Analysis.* The New York Times corpus [22] is used to find latent relations between entities of interest.
- *Cybersecurity.* A corpus based on the National Vulnerability Database [21] is used to learn possible relationships between attack methods and vulnerabilities as well as to study the evolution of cybersecurity threats.
- *Biomedicine and Pharmacology.* A corpus based on the MEDLINE database [16] of references and abstracts on life sciences and biomedical topics is analyzed to find hidden connections between treatments and diseases.

In the following sections, we motivate our work and define the research questions that we address in each chapter of this dissertation. Table 1.1 presents a summary of our research and contributions.

1.1 Tracking Story-Level Evolution of Entities

Studying the evolution of concepts over time in document collections to detect relevant stories within the corpus is called storytelling [97, 76, 119, 2, 40, 162, 120, 45, 141]. This

Table 1.1: Guzdial chart summarizing our research and contributions.

Topic	Research Questions	Data	Methods	Evaluation	Contributions
Story-level evolution	R1: How can we automatically discover chains of documents that describe the story-level evolution of a concept or entity? R2: What is the impact of modeling story-level evolution of a concept in improving predictive capabilities?	- New York Times	- Topic modeling - Objective function design, enhancement, and optimization	- Dispersion coefficient - Human-based study - Case-based study	- Captures story-level evolution - Diffusion vs similarity - Prediction
Time-reflective text representation	R3: How can we generate a temporal text representation that captures the semantic evolution of a language model over time? R4: How can we ensure that both long- and short-term changes in the meaning of natural language elements are captured?	- PubMed abstracts - National Vulnerability Database	- Tf-idf - Diffusion theory - Gaussian distribution	- Quantitative analysis: neighborhood monotony - Case-based study - Sensitivity analysis	- Captures language-model-level evolution - Captures short-term changes
Low-dimensional time-reflective word embeddings	R5: How can we generate a compressed representation of text that captures the semantic evolution of a language model over time? R6: How can the contemporary meanings of words be modeled over time to enrich predictive capabilities?	- New York Times - PubMed abstracts - National Vulnerability Database	- Approximate distance computation - Neural networks	- Quantitative analysis: similarity to baseline - Case-based study	- Low-dimensional representation - Suitable for trend analytics
Trend analytics for hypothesis generation	R7: How can we forecast future states of the generated time-reflective embeddings? R8: How can we generate hypotheses based on the current and predicted/extrapolated trends of the embedding space?	- New York Times - PubMed abstracts - National Vulnerability Database	- Recurrent Neural Networks - Attention-based transformer model	- Forecast: similarity to baseline - Hypothesis generation: Case-based study	- Sequential model of temporal word embeddings - Generation of future hypothesis

problem is a special case of language-model-level semantic evolution tracking, because the main objective of storytelling is to find a small subset of documents from a large corpus to summarize the evolution of a concept or entity.

In Chapter 3, we present an excerpt of an article we published in the International Journal of Data Science and Analytics [22], in which we propose a storytelling framework based on information diffusion. Our approach brings out the underlying diffusion of concepts and their progression over time, resulting in interesting and coherent stories. The framework discovers these stories over different periods, and the primary goal of our algorithm is to find different *segments* of time in a story, which can be related to changes in the context of a concept. The results of this research motivate transferring the idea of temporal diffusion to tackle the problem of tracking the semantic evolution of a language model.

The main research questions addressed in Chapter 3 are:

- How can we automatically discover chains of documents that describe the story-level evolution of a concept or entity?

- What type of predictive capabilities can be gained when finding the story-level evolution of a concept?

Storytelling is closely related to open and closed hypothesis generation [8]. The primary goal of both techniques is to build a chain of intermediate concepts that connect two disjoint but complementary concepts. Thus, storytelling techniques can be adapted for closed hypothesis generation since both follow the same principle, but for different applications.

Chapter 3 also presents a preliminary experiment that explores the possibility of predicting entities that might appear in the future given a set of recently published documents by studying the story-level evolution of entities. The prediction model considers how far in time a pair of entities may appear in the evolving chain of documents and estimates what are the tf-idf (term frequency inverse document frequency) weights [122, 115] of those entities when the terms appear in the future. Figure 1.3 depicts that some key terms of the Brussels bombings in 2016 could be predicted from the evolution of a news article relevant to a known Belgian-Moroccan terrorist named Abdelhamid Abaaoud who had masterminded the November 2015 Paris attacks. Using our storytelling framework, we were able to predict critical terms such as *Brussels*, *Belgium*, *Belgians* and *Paris attack* to appear near March 23, 2016. The Brussels bombings happened on March 22, 2016.

The modeling and predictive capabilities shown by this story-level technique lay the foundation for our language-model-level research.

1.2 Tracking Semantic-Level Evolution of Entities

Our work in storytelling demonstrates that concepts evolve over time and that it is possible to capture this evolution using the diffusion of concepts. However, while storytelling studies how information is diffused over time in a temporal corpus for a particular recent event, one of our primary goals is to investigate and model how a complete language model, e.g., the embedding space, evolves holistically over time.

In Chapter 4, we present an excerpt of an article titled "Tracking the Evolution of

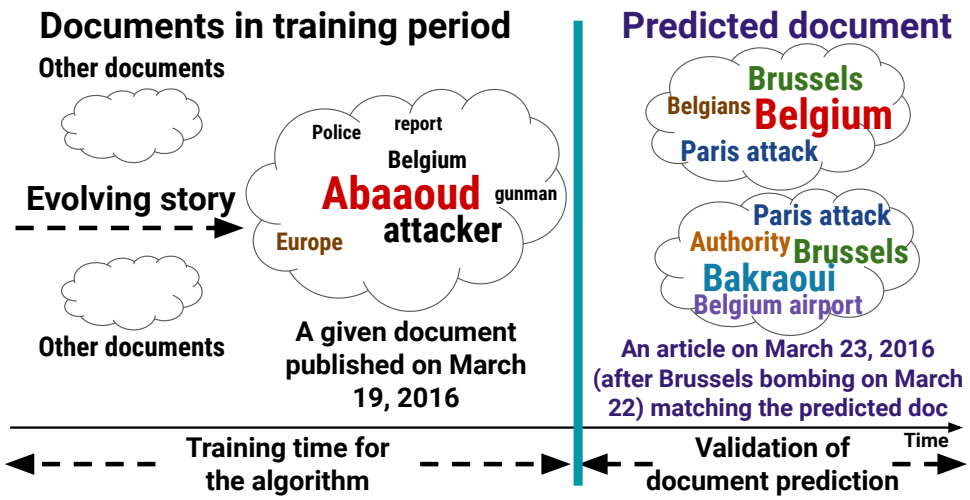


Figure 1.3: Predicted document using story evolution. The size of the words represent the predicted tf-idf weight.

Words with Time-reflective Text Representations" which was published in the Proceedings of the 2018 IEEE International Conference on Big Data [23](©2018 IEEE). In this work, we focus on the problem of tracking the semantic evolution of words or entities over time at the language-model level. We propose a preliminary high-dimensional time-reflective vector space representation of text documents that captures changes in the meaning of words. Throughout this work, we refer to *meaning* as what can be inferred by the context, or neighborhood, of a word at a particular point in time. We define the neighborhood of a term as the words that are considered *close* to the word of interest by some predefined measure, such as cosine similarity or Soergel distance [24].

The main research questions addressed in Chapter 4 are:

- How can we generate a temporal text representation that captures the semantic evolution of a language model over time?
- How can we ensure that both long- and short-term changes in the meaning of natural language elements are captured?

Using the generated word vectors, one can study the nearest neighbors of every word

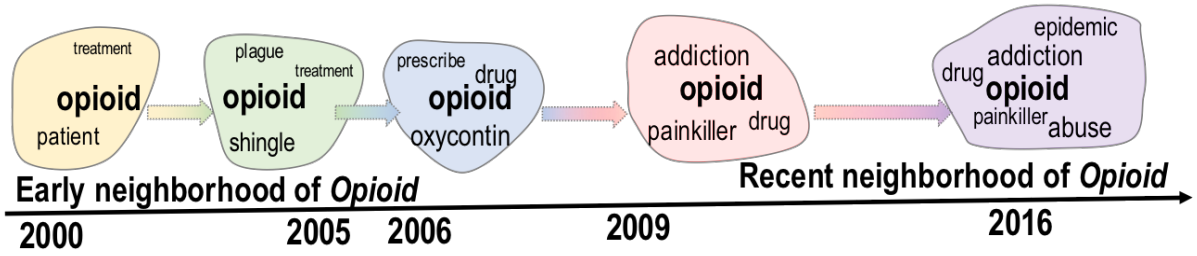


Figure 1.4: Evolution of the word *opioid*.

and track how the neighbors are changing over time, which is equivalent to tracking contextual or semantic similarity. Along with many case studies, we analyzed the word *opioid*, which is a class of drugs including *heroin*, *fentanyl*, and pain reducers like *OxyContin*. The nearest neighbors of *opioid*, based on New York Times news articles, as shown in Figure 1.4, conveyed the meaning that *opioid* was used to treat patients with painful diseases like shingles and plague in the early 2000s. Over time, the context of *opioid* started shifting towards terms like *OxyContin*, *addiction*, *drug*, *epidemic*, and *abuse*, which is a good representation of the current context of *opioid*.

The designed model is promising but not yet feasible for trend analytics or prediction of future word-vectors given that the vector-features (i.e., columns) are documents. That is, each vector has a length equal to the number of documents in the entire corpus. Having documents as vector-features results in two complications: (1) a high computation and storage complexity, and (2) since future documents are not seen, the vector-cells for unseen documents cannot be modeled or predicted. Our approach, as described in Section 1.3, focuses on overcoming these limitations by generating low-dimensional time-reflective embeddings that have a fixed number of features (columns).

1.3 Low-Dimensional Temporal Embeddings for Text Representation

In Chapter 5, we introduce a neural-network-based framework that generates low-dimensional temporal word embedding models that overcome the weaknesses of the time-reflective vector representation described in the previous chapter. There are two main reasons to obtain low-dimensional temporal embeddings based on our time-reflective representation and the observed distances between these vectors. First, as shown in Chapter 4, the obtained time-reflective representation significantly reduces the noise from the original data, and it can capture sudden changes in the corpus. Our temporal embedding model inherits these ideal characteristics and can capture sudden semantic changes.

The second reason for obtaining temporal embeddings is that having a lower-dimensional continuous representation of every word makes it feasible to use trend-analytics algorithms to predict/extrapolate the future trajectories of a word. With the projected trajectories of every word in the dataset, it is possible to estimate the future neighborhood of a particular word. Chapter 6 discusses our work in this area.

The main research questions that are addressed by Chapter 5 are:

- How can we generate a compressed representation of text that captures the semantic evolution of a language model over time?
- How can the contemporary meanings of words be modeled over time to enrich predictive capabilities?

To generate the new low-dimensional word embeddings, we use the neighborhoods of our time-reflective model (presented in Chapter 4) as training data and as a baseline. Our experiments show acceptable performance in terms of tracking the semantic evolution of words or entities over time at the language-model level. The vector size of the proposed temporal word embeddings in our experiments is 64, which is less than 3.0% of the vector size (3,000-5,000) required for the representation presented in Chapter 4.

1.4 Trend Analytics for Hypothesis Generation

Current hypothesis generation techniques [143, 52] focus on associating multiple concepts, e.g., *nicotinamide* and *cell fate decisions* to study *organismal aging*, by searching for statistically significant connections between these terms. While many hypothesis generation mechanisms have been successful in scientific domains, such closed analyses require domain knowledge about input pairs of concepts. One limitation of the closed analyses used in the current literature is that the generated scientific hypotheses are direct connections supported by the text corpus.

The use of a continuous temporal language representation makes it possible to predict how the vector space will be in the future even though the corpus does not contain future documents. Hypotheses generated from the predicted future information space, e.g., the embedding space, have the potential to be novel hypotheses that traditional association-based approaches might not reveal.

Estimating future trends using our temporal word embeddings could reveal that terms like *elderly care* and *Alzheimer's disease* may appear near the term *baby boomer* in the future embedding space. Multiple hypotheses, relationships, or questions can be generated from the neighborhood of the words in the predicted future embedding space. For example, "baby boomers receive elderly care," or "baby boomers present Alzheimer's disease at a high rate". Based on the trajectory of all relevant words, a projected scenario can aid the formation of such hypotheses.

The main research questions that we address in Chapter 6 are:

- How can we forecast future states of the generated diffusion-based temporal word embeddings?
- How can we generate hypotheses based on the current and predicted/extrapolated trends of the embedding space?

In Chapter 6, we leverage the predictive power of our low-dimensional diffusion-based temporal word embedding model for automatic hypothesis generation. We achieve this

by modeling the trends observed in the temporal embedding vectors of all words in the vocabulary using different sequential modeling algorithms, including different variants of recurrent neural networks as well as the attention-based transformer model. Once we create and train a model, we can use it to generate future word vectors and find pairs of words that are not related now but are related in the future embedding space. Using this approach, a scientist can hypothesize that this *latent relationship* is significant and verify it through experimentation. Our experiments show very promising results in terms of automatic hypothesis generation, as well as for the task of predicting the future neighborhood (or context) of a word.

Chapter 2

Related Work

In this chapter, we review the literature related to different aspects of this proposal to put our contributions in perspective.

2.1 Semantic Evolution

Language is ever-evolving, going through syntactic and semantic diachronic (temporal) changes, such as out of use words disappearing or being substituted by new words that better exemplify the contemporary meaning and intent of the expression [3, 154]. Many probabilistic approaches aim to track temporal changes in word meaning and semantic evolution by transforming a text corpus into a latent time series model [114, 108, 14, 150, 39, 128]. Mihalcea et al. [87] leverage Part of Speech (PoS) features from a word and its context to create a supervised model that predicts when that word was published. Other approaches describe a corpus as a graph, representing words as nodes, and generating edges between them that are weighted based on contextual information [91, 67]. None of these approaches focus on finding a vector space model that accurately models the corpus for tracking semantic evolution.

2.2 Storytelling

Storytelling in data mining refers to the problem of connecting two disjoint concepts via other intermediate concepts. It is often called the connecting-the-dots problem. There are several approaches that aim to solve the *storytelling* problem for different types of datasets,

such as scientific articles [58], entity networks [35, 36, 56], image collections [50, 123] and text corpus [97, 76, 119, 2, 40, 162, 120, 45, 141]. Many approaches in this problem space use graph-based representations of a document or entity set [35, 97, 57, 40, 162]. *Storytelling* has several different applications, such as intelligence analysis [56, 57], news recommendation [45], searching [36, 162], and social network analysis [35, 123, 102].

Solutions to the *connecting the dots* problem include the use of probabilistic approaches, such as random walks [119, 162, 120, 45, 141], Monte Carlo simulations [2] and determinantal point processes (DPP) [40]. Shahaf et al. [119] introduced the concept of coherence and coverage to assess the quality of a chain limited by two user-specified endpoints, and then extended this work by building *metro maps* [120] which are formed by several coherent chains that intersect at some points, forming a map-like structure. Gillenwater et al. [40] also obtain several coherent chains using a DPP-based model.

2.3 Dynamic Topic Models

Several different versions of dynamic topic models have been developed. These models obtain the distribution of a word over latent topics, effectively trying to identify the changes in the patterns of word usage [17, 137, 134, 46, 53, 140, 96]. Dynamic topic models can be combined with clustering techniques to identify significant changes in the clusters over time [140]. However, these methods obtain the distribution of a word over topics and thus are limited in their application to modeling the semantic changes of a word because the generated representations are not continuous over time.

2.4 Vector Space Models

A vector space model is a tool used to represent a text corpus in a continuous space. Vector space models can be classified into three classes: term-document, word-context, and pair-pattern [129]. The term-document matrix representation corresponds to the traditional

tf-idf model [122] and its many different variations in terms of weighting and normalization [115, 121] at a document level. The word-context model is obtained usually at a window or sentence level. The pair-pattern matrices are obtained by identifying patterns (columns) for a pair of words (rows) [129]. These vector models can be leveraged to compute semantic neighborhoods by using cosine similarity. Hamilton et al. [47] use a second-order vector which is obtained from the original vector plus the vectors of the neighborhood to compute the semantic similarity at each timestamp.

2.5 Word Embeddings

Word embeddings are, in general, vector space models obtained by leveraging the distributional statistics of the words in a corpus. Thus, word embeddings over time can flexibly be used to compute the semantic similarity between words at different timestamps. Mikolov et al. [89, 88] introduced the word2vec model, which obtains static embeddings, or low-dimensional representations of a corpus at a single point in time. Word2vec represents the foundation of most of the current state-of-the-art dynamic embeddings. Barkan [12] proposes a probabilistic version of the same algorithm using Bayesian inference. A different unsupervised learning approach is GloVe by Pennington et al. [106] in which word vectors are created using matrix factorization of a word-word co-occurrence matrix.

The introduction of the temporal dimension in word embeddings has also been explored. Several authors have trained word embeddings at every timestamp in their corpus, and then used a method such as regression to connect the embeddings over time artificially [70, 74, 48, 158, 30, 60]. The main drawback of this general approach in terms of the semantic evolution tracking task is that it requires having a substantial number of documents at each timestamp so that word2vec (or a different static method) can obtain a high-quality model, which is not always the case, especially with the introduction of new words at different points in time.

Rosin et al. [109] use a similar technique but introduce the supervised task of semantic

relatedness, which attempts to predict if two words are related at a certain point in time. Hu et al. [59] use pre-trained word embeddings to generate sense models, but their framework works at a word level.

Dynamic embedding models have been introduced to overcome the problem of data sparsity and lack of continuity of the previous approach, by using joint embedding generation. Bamler and Mandt [11] leverage a probabilistic Bayesian version of word2vec to infer the embedding vectors at each timestamp, but use Kalman filtering to connect the embeddings over time. Yao et al. [149] propose an approach that uses the pointwise mutual information (PMI) matrix instead of word2vec. To obtain the dynamic embeddings, the PMI matrix is factorized iteratively at each timestamp, and alignment is enforced through regularization. Finally, Rudolph and Blei [111] use Kalman filtering like Bamler and Mandt, but Rudolph and Blei use a non-Bayesian approach based on *exponential family embeddings*. According to Bamler and Mandt, using such a non-Bayesian approach makes the model more sensitive to noise for sparse data.

2.6 Embedding/Semantic Change Evaluation

The vast amount of work on word embeddings and semantic change analysis has resulted in significant interest in evaluating the quality of the results [117, 41, 51], as well as visualizing the semantic evolution of words in a comprehensible, useful way [61]. Schnabel et al. [117] introduce a *coherence* metric that is evaluated by showing a group of three words semantically related to a word at a certain point in time to a human evaluator, as well as an unrelated word. The expectation is that the evaluator can easily identify the non-related word [117].

Hellrich and Hahn [51] show that *reliable* algorithms based on word2vec usually provide inconsistent word neighborhoods. Dubossarsky et al. [33] argue that the evidence of semantic laws given by recent literature is not enough and that some of those laws are merely an artifact of the characteristics of the data. Tahmasebi et al. [125] propose an evaluation

scheme for semantic evolution tracking.

2.7 Event Detection and Prediction

Event detection consists of detecting temporal bursts of highly correlated documents [146, 147, 72], while *event extraction* aims at obtaining useful knowledge regarding these events [77, 110]. *Event evolution* [153, 141] and *topic evolution* [64, 69, 136] study how these abstract concepts (*events* and *topics*) evolve over time, based on latent relationships within the corpus and temporal information encoded in each document.

The *event prediction* through text mining problem has also been addressed in the literature. Radinsky et al. [108] described a system that uses *causality extraction* to obtain pairs of terms that have a causal relation, and that are later used to train a prediction algorithm. Luo et al. [81] also address this problem by introducing the concept of *semantic uncertainty*, which is used to estimate the *most certain* next state based on the current state, or event. This approach is particularly useful when there is limited historical data available.

2.8 Hypothesis Generation

The existing primary application for hypothesis generation is to support Literature-Based Discovery [143, 52]. Current hypothesis-generation algorithms in the literature are based on co-occurrence models, semantic models, or distributional models [52]. Most of these techniques rely on creating evidence from text corpora to form a connection between entities [58, 116, 139] (e.g., gene and protein, cell and biochemical agents, or person and event). An unexplored area of hypothesis generation is to perform an extrapolation analysis and predict hypotheses that could be present in the embedding space of a future timestamp.

2.9 Applications of Predictive Text Analysis

Over the last few decades, several fields have benefited from text mining techniques for prediction. These applications include stock market prediction [20, 27, 98, 105, 118, 157], sentiment analysis for products and services [78, 83, 93, 95, 103], sentimental analysis in social media [19, 31, 94], inference of complementary products in recommender systems [15, 42, 84], and prediction of illegal activities in the dark web [126].

Researchers in the fields of biomedical sciences have utilized text datasets for numerous applications including medical recommendations [159], disease transmission analyses [1, 113], prediction of gene and protein association [34], and detection of medical misinformation in health forums [71]. While prediction using text is not new, prediction to generate a hypothesis that might later become apparent is unique.

2.10 Sequence Modeling

Performing predictive text analysis requires generating a *sequence model* of the phenomena under study. At a high level, a sequence model takes as input a sequence of elements and tries to predict the next element of the sequence. In our case, the order of the sequence represents the temporal dimension. Several researchers have focused on studying multi-dimensional time-series prediction [85, 82, 151, 152] using methods such as linear regression and accounting for temporal effects such as seasonality [82]. However, these models are not well-suited for the prediction of non-linear phenomena, which is the case of semantic evolution.

Recurrent neural networks (RNNs) [112] are the state-of-the-art in sequence modeling. An RNN is essentially a neural network that takes sequential data as input, with an architecture that simulates an *internal memory*. There are many variations of RNNs, but the most well known are Long Short Term Memory (LSTM) [54] and Gated Recurrent Units (GRU) [25] networks. The main difference between these architectures and vanilla RNN

is the addition of mechanisms to *forget* the initial states of the sequence to give more importance to the final states. The applications of RNNs and its different variations include many different areas such as sentiment analysis [138], image captioning [133, 13], speech recognition [43], language modeling [124] and machine translation [25].

In some tasks such as machine translation, it is imperative to be able to give more importance to the elements of the input sequence that are more relevant to the predicted value. Bahdanau, Cho and Bengio introduced the concept of *attention* in [9] to solve this issue. A regular RNN encodes a whole input sequence into a single fixed-length vector while an RNN with attention generates vectors for each element and then performs a weighted sum to produce the prediction, giving higher weights to the most relevant elements. The model learns these weights during the training process.

The sequential nature of the different versions of RNNs makes it impossible to train samples in parallel, which is particularly important for long sequences. Thus, Vaswani et al. introduced the *Transformer* model [132], which gets rid of the RNN and relies entirely on an attention mechanism. This model allows parallel training and surpasses the state-of-the-art performance in machine translation.

Chapter 3

Analyzing Evolving Stories in News Articles

3.1 Introduction

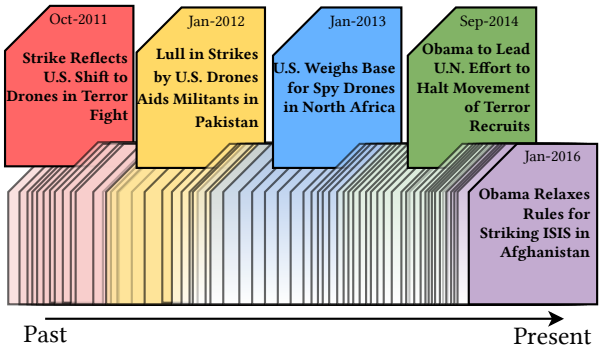
The pervasiveness of the Internet has greatly facilitated editors to publish a large number of news articles every day in any topic, practically without any restrictions of page-limits. Thus, the number of articles available to avid readers increases every day at a breakneck pace. Many business and government organizations track these news articles to study public sentiment, business directions, the progression of events, and many other aspects of economic and socio-political issues. However, tracking topics from different perspectives is a difficult task given that one news-story of today's interest could have evolved from another story of the past, thus forming a chain of relevant events. This leads to the concept of *diffusion theory* [7], which refers to the change of the distributional patterns of a phenomenon over time.

While capturing such diffusions over a timeline is still a challenge, researchers have targeted the problem of tracking stories in different guises, e.g., storytelling [57, 76], storyboarding [86], connecting the dots [119, 58], and metro maps [120]. Most of these methods find underlying connections between articles using a similarity-based network of documents. Their objective focuses mainly on generating a cohesive thread of a story. However, extensive usage of cohesion may result in stories that revolve around a single *chapter* of the evolving thread of the story.

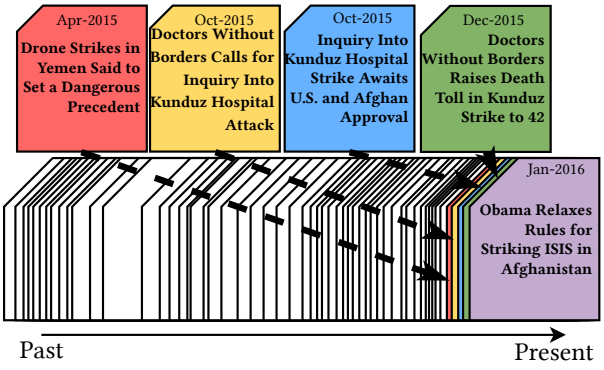
Figure 3.1 (a) illustrates a scenario generated using a diffusion-based approach described

in this chapter, which captures the articles published in the past reflecting the evolution of the story. The chain starts with an article that describes the increase in drone strikes from the U.S. against Al Qaeda. The next article describes a decrease in this type of strikes. The chain continues with an article about the possible use of drones in Africa to fight Al Qaeda and other extremist groups such as ISIS, followed by an article that discusses ISIS recruiting. Finally, the chain ends with an article that describes the use of drones in the fight against ISIS.

Figure 3.1(b) presents the results obtained for the same seed document, using a similarity-based approach to track past documents similar to the article of interest. While our ap-



(a) A diffusion-based story.



(b) A similarity-based story.

Figure 3.1: A diffusion-based approach captures documents from the past reflecting a smooth transition of the evolving story. In contrast, a similarity-based approach focuses on a narrow range of the timeline.

proach brings out the underlying diffusion of concepts and their progression over time, the similarity-based approach provides a story that is highly specific to a single event — a drone strike in Kunduz. Unlike the similarity-based model, our approach can include dissimilar documents from the distant past, which add critical context to the document of interest over time. An additional limitation of the similarity-based approach is that it tends to track back only to the recent past because similar documents to the current event of interest become much rarer as their historical timespan increases.

This chapter presents a novel storytelling framework that takes as input a large corpus of documents, along with a user-specified set of seed documents. The framework runs through a palette of natural language processing (NLP) and other preprocessing tasks before it reaches its core objective function, which generates a storyline by finding and combining turning points that lead to the seed story. The end-product of the framework is a short chain of documents from a multitude of news articles to help the user identify the evolution of a story, as shown in Fig. 3.1(a).

The contributions of the work presented in this chapter can be summarized as follows:

- **Evolution:** We propose a novel method to discover a chain of documents published in the past given a set of seed documents, where the chain reflects the evolution of the concepts in the seed documents.

- **Diffusion and similarity:** Our proposed technique captures the underlying diffusion of concepts and has the flexibility to incorporate the similarity concepts from the state-of-the-art methods.

- **Prediction:** We demonstrate the potential of the outcome generated by our approach as a tool to predict future states of an evolving story, which in turn shows that the available data has predictive power.

- **Evaluation:** We conduct a set of experiments to quantitatively and qualitatively evaluate different aspects of our model. Additionally, two human participant-based studies demonstrate that the discovered storylines were satisfactory.

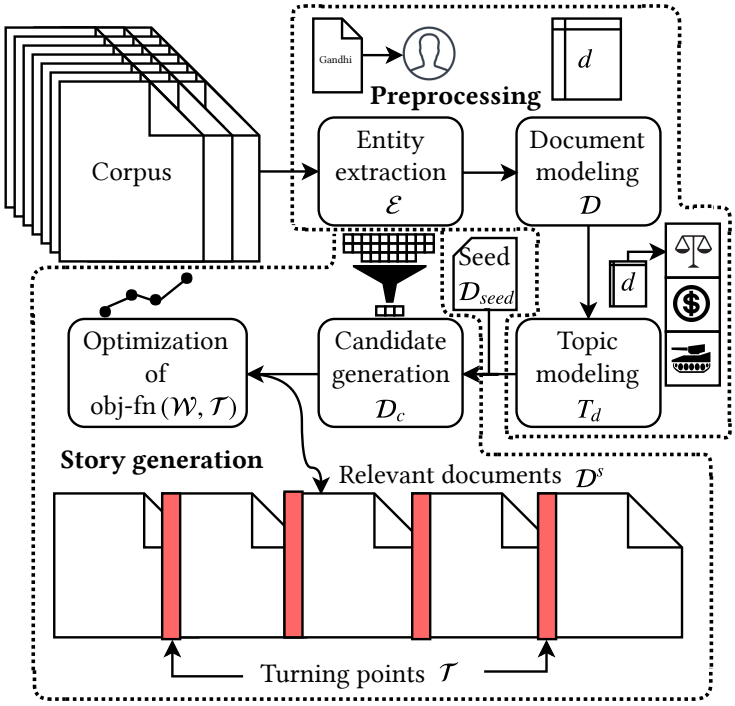


Figure 3.2: Proposed framework.

The work presented in this chapter has several differences with respect to the previously mentioned methods. First, we use an optimization-based approach similar to the one presented by Shahaf *et al.* [119], but we introduce the concept of *diffusion* to identify the evolution of the concepts in the seed documents. The use of diffusion allows our framework to create stories based on a combination of different events, which is not possible using previously proposed methods. Second, our work discovers the most relevant documents associated with each turning point to provide a concise summary of each turning point event. Third, we leverage the topic distribution of each document to reduce our search space while keeping essential documents for our optimization. Finally, we suggest the possibility of using the resulting relevant documents to build a regression framework for entity prediction.

3.2 Problem Description

The work presented in this chapter focuses only on news articles and the entities within the news. The entities detected are persons, organizations, and locations. Let $\mathcal{D} = \{d_1, d_2, \ldots, d_{|\mathcal{D}|}\}$ be the set of documents and $\mathcal{E} = \{e_1, e_2, \ldots, e_{\mathcal{E}}\}$ be the set of entities in the news corpus. Each document $d \in \mathcal{D}$ contains a set of entities $\mathcal{E}_d \subset \mathcal{E}$. Additionally, each document d has a publication date t_d. As part of the preprocessing steps, we apply topic modeling using Latent Dirichlet Allocation (LDA) [18] on $\mathcal{D}$, obtaining a topic distribution T_d for each document d.

3.2.1 What is the expected outcome?

Let the user input be a set of seed documents $\mathcal{D}_{seed} \subset \mathcal{D}$, where $|\mathcal{D}_{seed}| \geq 1$. We define a *turning point* in the story $\tau \in \mathcal{T}$ as a specific date in which the story under analysis has a significant change. Let $\mathcal{S}$ be a vector where each element, defined as *segment*, is a pair of consecutive turning points such that $|\mathcal{S}| = |\mathcal{T}| - 1$, and $\mathcal{S} = [(\tau_1, \tau_2), (\tau_2, \tau_3), \cdots, (\tau_{|\mathcal{T}|-1}, \tau_{|\mathcal{T}|})]$.

Our main goal is to split the set of documents into $|\mathcal{S}|$ segments by finding $|\mathcal{T}| - 2$ turning points since the first and last turning points are fixed. For each segment $s \in \mathcal{S}$ we want to find a subset of documents $\mathcal{D}^s \subset \mathcal{D}$ that help the user study the evolution of the prominent entities in $\mathcal{D}_{seed}$ through various different events. The value of $|\mathcal{S}|$ can be chosen by the user or set to different values automatically to obtain the optimal result.

3.3 Methodology

Our storytelling framework comprises two main stages: (1) preprocessing, which creates document and topic models from a text corpus and (2) story generation, which takes a set of seed documents and other constraints as inputs and, via an optimization routine, outputs a story formed by relevant documents that have a common thread, but that belong to different events. Fig. 3.2 illustrates this framework.

3.3.1 Preprocessing

In the preprocessing stage, our framework: (1) extracts entities (e.g., person, location, organization) from the documents, (2) represents each document as a vector of entities, and (3) obtains a topic distribution for each document in the corpus. We extract entities from the text corpus using standard Named Entity Recognizers [4, 37]. Our storytelling framework leverages a tf-idf model with cosine normalization [57] to generate weights $w(e, d)$ for each entity $e \in \mathcal{E}$ in each document $d \in \mathcal{D}$.

3.3.2 Story generation

In this stage, the storytelling framework furnishes a candidate set of documents $\mathcal{D}_c \subset \mathcal{D}$ from the entire dataset that satisfies some temporal and topical constraints with respect to the seed documents $\mathcal{D}_{seed}$. The temporal and topical constraints are driven by user input regarding how far back in time the algorithm should track to detect an origin, or, how much deviation the algorithm should allow in terms of topic distribution as compared with the seed set.

The temporal criteria establishes that all of the candidate documents must have been published before the most recent seed document and within a certain maximum threshold $t_{\max}$, i.e., d_i can only be part of $\mathcal{D}_c$ if $0 < \min(t_{seed}) - t_i < t_{\max}$, where t_i is the publication date of document d_i, and $\min(t_{seed})$ is the publication date of the oldest article.

The topical criteria expresses, in terms of topic distributions, how much deviation the candidate documents can have from the seed documents when optimizing for a diffusion and cohesion objective. For each document $d_{seed} \in \mathcal{D}_{seed}$, we compute the KL-divergence [75], or *topical divergence*, between the topic distribution of the seed document $T_{d_{seed}}$ and the T_d of all the documents $d \in \mathcal{D}$. Document d is included in $\mathcal{D}_c$, only if $\forall d_{seed} \in \mathcal{D}_{seed}\,(\text{KL-divergence}(T_d, T_{d_{seed}}) \leq \alpha)$, where T_d is the topic distribution of document D, obtained using the LDA [18] and α is a user-defined parameter. In this section, we will refer to $\mathcal{D}_c$ as $\mathcal{D}$, for brevity.

Finally, the storytelling framework generates a story model that identifies turning points of the story over a timeline and provides relevant documents within each segment based on an objective function optimization. The outcome of the optimization routine helps to understand the evolution of the news-story described in the seed document(s).

We formulate the objective function for evolution using four terms: *incoherence* (Eq. 3.1), *similarity* (Eq. 3.2), *overlap* (Eq. 3.6), and *uniformity* (Eq. 3.7). First, we have two penalties that aim to minimize the *incoherence* of documents within a segment, while also minimizing the *similarity* across segments. The expectation is that these terms promote having highly coherent segments that are independent of each other.

$$\text{incoherence}(s) = \frac{\sum_{i,j}^{|D|\times|D|} w_i * w_j * \Phi * \text{soergel}(d_i, d_j) * |t_i - t_j|}{\sum_{i,j}^{|D|\times|D|} w_i * w_j * \Phi} \tag{3.1}$$

$$\text{similarity}(s) = \frac{\sum_{i,j}^{|D|\times|D|} w_i * w_j * \phi * \mathrm{e}^{-\text{soergel}(d_i,d_j)}}{\sum_{i,j}^{|D|\times|D|} w_i * w_j * \phi} \tag{3.2}$$

where $w_i \in \mathcal{W}$ is the weight for document d_i and

$$\Phi = \gamma(t_i, t_L^s, t_H^s) * \gamma(t_j, t_L^s, t_H^s) \tag{3.3}$$

$$\phi = \gamma(t_i, t_L^s, t_H^s) * (1 - \gamma(t_j, t_L^s, t_H^s)) \tag{3.4}$$

where $t_{L,H}^s$ are the lower/upper turning points for a particular segment and γ is defined as:

$$\gamma(t, t_L^s, t_H^s) = \begin{cases} \frac{1}{\sqrt{2\pi\hat{\sigma}^2}} \mathrm{e}^{\left(-\frac{(t-t_L^s-\hat{\sigma}^2)^2}{2\hat{\sigma}^2}\right)} & \text{if } t \leq t_L^s \\ \frac{1}{\sqrt{2\pi\hat{\sigma}^2}} & \text{if } t_L^s < t < t_H^s \\ \frac{1}{\sqrt{2\pi\hat{\sigma}^2}} \mathrm{e}^{\left(-\frac{(t-t_H^s+\hat{\sigma}^2)^2}{2\hat{\sigma}^2}\right)} & \text{if } t_H^s \leq t \end{cases} \tag{3.5}$$

where $\hat{\sigma}$ is the standard deviation of a Gaussian distribution that indicates the degree of membership of a timestamp in a segment.

The third term in the objective function is the *overlap* penalty, which prevents having

two or more turning points very close to each other.

$$\text{overlap} = \left(1 + \sum_{i,j,i<j}^{(|\mathcal{S}|-1)\times(|\mathcal{S}|-1)} e^{\left(-\frac{(t_i-t_j)^2}{2\sigma^2}\right)}\right) \tag{3.6}$$

where σ is the standard deviation of a Gaussian distribution that defines the sensitivity of the overlap penalty.

A final term is added to avoid having high *uniformity* in the distribution of the weight probabilities that indicate if the documents are relevant or not. This penalty will make sure that not all of the weights are set to 0 or 1.

$$\text{uniformity} = \left(1 + \sum_{s=1}^{|\mathcal{S}|}\left(1 - \frac{\left(\left|\left|\frac{\mathcal{W}_s * \Gamma_s^\top}{\sum \mathcal{W}_s * \Gamma_s^\top}\right|\right|_2 * \sqrt{|\mathcal{W}_s|}\right) - 1}{\sqrt{|\mathcal{W}_s|} - 1}\right)\right) \tag{3.7}$$

where Γ_e is a vector of values returned by the membership function (Eq. 3.5 for the documents that fall in segment s).

The final objective function (Eq. 3.8 is minimized for two vectors: $\mathcal{S}$ and $\mathcal{W}$. The elements $s \in \mathcal{S}$ are bounded by the range of the turning points and the elements $w \in \mathcal{W}$ are bounded between $[0, 1]$. We use the quasi-newton limited memory algorithm for bound constrained optimization (L-BFGS-B) [160].

$$\mathcal{F}_(\mathcal{T}, \mathcal{W}) = \sum_{s=1}^{|\mathcal{S}|} (\text{incoherence}(s) * \text{similarity}(s)) * \text{overlap} * \text{uniformity} \tag{3.8}$$

3.4 Experimental Results

For our experiments, we use a corpus of 400,842 New York Times articles published between January 2000 and June 2016 on the U.S. and World news sections. The corpus contains 3,320,886 unique entities. To the best of our knowledge, there are no publicly available labeled benchmark datasets that can be used to perform a supervised evaluation of a storytelling framework. We base our evaluations on a combination of statistics- and human-participant-based studies. In this chapter, we seek to answer the following questions.

1. How well does our approach reflect the evolution of a story? (Section 3.4.1)

2. How does our method compare quantitatively to other methods such as clustering and similarity-based storytelling? (Section 3.4.2)

3. How does our method compare to a sophisticated storytelling algorithm? (Section 3.4.3)

4. How does the statistical significance for the position of the turning points change while modifying the different hyper-parameters of our optimization? (Section 3.4.4)

5. How does the dispersion coefficient change while modifying the number of segments $|\mathcal{S}|$? (Section 3.4.5)

6. What is the repeatability of the optimization? (Section 3.4.6)

7. Can we use the set of highly relevant documents with respect to a seed document to predict which entities are expected to appear in a future document? (Section 3.4.7)

3.4.1 Evaluation of chain continuity

The lack of a benchmark dataset makes it difficult to evaluate the performance of our storytelling framework. To address this issue, we designed a user study motivated by the work of Shahaf *et al.* [119]. We picked four different topics and two different random documents related to each topic as seed documents. The topics selected were *aviation*, *Brexit*, *ISIS*, and *tensions between North and South Korea*. Ten evaluators were asked to score the resulting chain for each seed document so that each topic had a total of twenty evaluations[1].

The sequence of documents presented to the evaluators consisted of one document per segment, which was selected based on the relevance weights. The number of segments $|\mathcal{S}|$

[1]Evaluation questions available at `https://storyeval.herokuapp.com/`

was set to five when running the optimization algorithm. Users evaluated four criteria on a scale of 1 to 5, with five being the best, for each sequence:

- *Familiarity:* How familiar is the evaluator with the topic?
- *Relevance:* Do the documents seem relevant to the topic?
- *Coherence:* Is the chain coherent? A document in the chain that does not belong to the topic results in a lower coherence score.
- *Broadness:* Does each document provide new information guaranteeing a slight shift in its topic over time?

The average scores per criterion are presented in Fig. 3.3. The results indicate that our algorithm performs, on average, above a score of 3.5, which indicates that the chains are coherent, relevant, and broad enough to capture evolution. An interesting observation is that the *Brexit* topic, which had the lowest average familiarity score, also has the lowest scores for coherence and relevance, and the topic with the highest average familiarity score *Korea*, had the highest scores for coherence and relevance, indicating that such a study with human participants might be biased toward familiarity with the topic.

3.4.2 Quantitative comparison with other methods

We quantitatively compare the outcomes of our framework with those of several other methods:

- similarity-based approach
- k-means clustering
- agglomerative clustering (using average, complete and Ward linkage)
- spectral clustering (with Gaussian kernel (RBF) and nearest neighbors affinity)

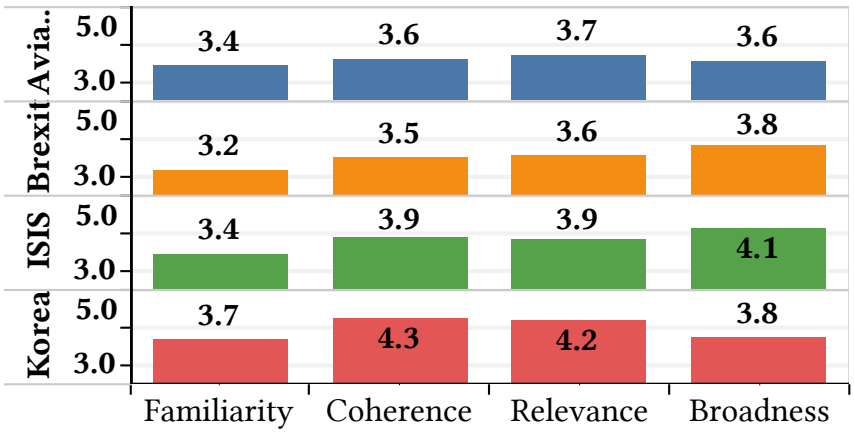


Figure 3.3: Average scores obtained for each metric per topic, a higher score is better (max. 5.0).

To evaluate our method, we selected the most relevant document per time-segment. For the similarity-based approach, the consecutive nearest neighbors were selected as the chain starting with the seed document. For all of the clustering-based experiments, we added time as a feature, and we use the number of time segments as the number of clusters generated. For the k-means clustering-based approach, we selected the document closest to the centroid of each cluster as a relevant document. For the rest of the clustering algorithms, we selected the document in the middle of the cluster (in terms of publication date). Notice that these alternative approaches focus on distance or similarity, while our objective function is designed to capture diffusion. Hossain *et al.* introduced the concept of *dispersion coefficient* in evaluating the Storytelling algorithm [57]. We use the same concept to evaluate the quality of a chain of documents $\{d_0, d_1, \ldots, d_{n-1}\}$, containing n articles.

$$\psi = 1 - \frac{1}{n-2} \sum_{i=0}^{n-3} \sum_{j=i+2}^{n-1} \text{disp}(d_i, d_j) \tag{3.9}$$

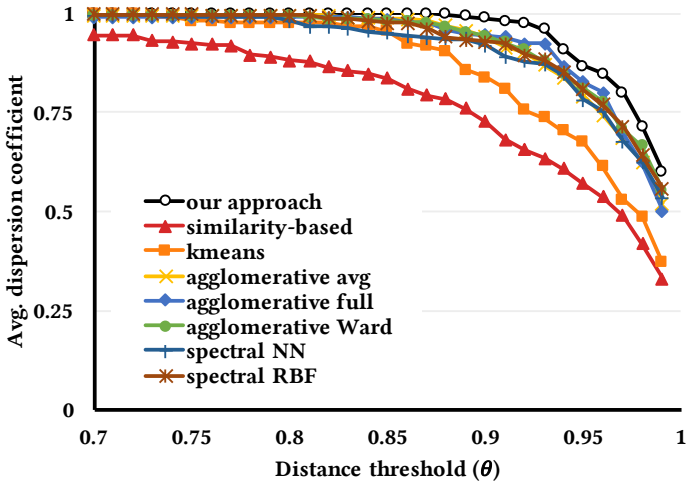


Figure 3.4: Comparison of average dispersion coefficient for stories created using our diffusion based approach, several clustering-based algorithms, and a similarity-based technique. Our approach results in the best (highest) dispersion.

where

$$\text{disp}(d_i, d_j) = \begin{cases} \frac{1}{n+i-j} & \text{if soergel}(d_i, d_j) < \theta \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

The dispersion coefficient of a chain of documents is 1.0 only if consecutive pairs of documents meet a given distance threshold (θ). The coefficient ψ is 0 when every pair of documents in the chain satisfies the distance threshold.

We generated chains for 32 different seed documents using all seven approaches under consideration. Figure 3.4 compares the average dispersion coefficient versus the distance threshold. It shows that the average dispersion of our diffusion-based method provides the highest dispersion coefficients for any distance threshold. This indicates that our method generates chains with a smooth transition of topics, which is one of our goals.

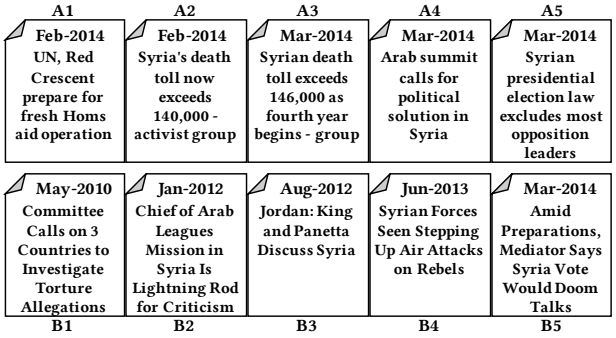
3.4.3 Comparison with *Metro Maps* [120]

Several algorithms and frameworks attempt to solve the *connecting the dots* problem, as presented in Chapter 2. It is difficult to perform a quantitative, or in many cases, qualitative, comparison of these methods with our algorithm because most of these frameworks are not publicly available or have a restricted dataset. Moreover, the objectives, scopes, and context of the algorithms vary widely. The SNAP library from Stanford University [79] provides a demo of the *metro maps* framework introduced by Shahaf *et al.* [120]. We conduct a comparative study between the results of *metro maps* and our framework.

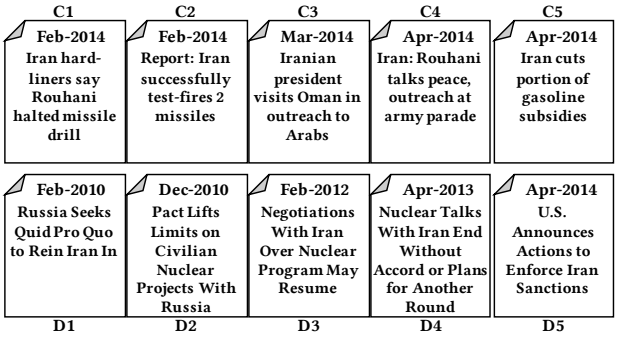
Metro maps chains are pre-generated. To generate chains under the same topic, we searched our database for similar news articles published around the same dates. These documents were used as seeds for our diffusion-based framework. We selected the document with the largest weight per segment.

We obtained two different chains of documents for two different queries: *Syria* and *Iran*. Fig. 3.5 (a) shows the results for the topic *Syria*. The row with prefix A was created with *metro maps* and the second row prefixed by B shows the result of our framework. Fig. 3.5 (b) depicts the results of the query *Iran*, where prefix C is used to identify documents in the *metro maps* chain and prefix D for the documents in the chain generated by our framework.

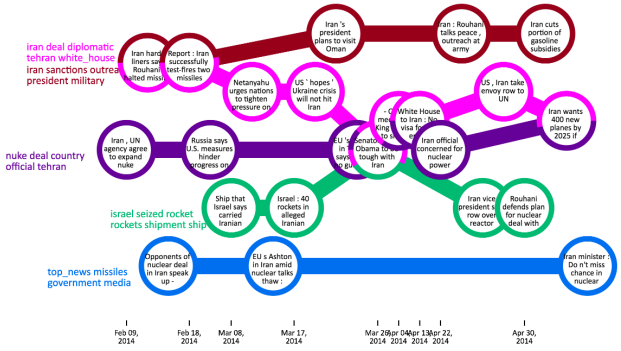
Fig. 3.5 (c) shows the actual output from *metro maps* for the *Iran* query. As can be seen, the format of the output from *metro maps* differs greatly from the output of our framework, with many different chains that intersect at several points, but not a clear definition of segments or turning points. This, along with the differences between the datasets used, makes a comparison between both methods almost impossible. It is unclear if the *metro maps* approach can handle longer time frames since the available demo only uses news articles from the prior three months. However, looking at Fig. 3.5 (c), it seems that extending the time frame would significantly increase the number of articles presented, making it harder for the user to obtain useful information. As opposed to *metro maps*, our approach is capable of building stories from articles published in any arbitrary time frame.



(a) Comparison between a single chain from the *metro maps* output (A) with a chain created by our framework (B) using the seed document on the right, related to the *Syria*.



(b) Comparison between a single chain from the *metro maps* output (C) with a chain created by our framework (D) using the seed document on the right, related to Iran.



(c) The output of the Stanford SNAP [79] implementation of *metro maps* for articles between 2013 and 2014 related to Iran.

Figure 3.5: Comparison between our method and the *metro maps* framework presented by Shahaf *et al.* [120]

Evaluation

Twelve evaluators were asked to score the resulting pair of chains for each topic. The chains did not have any information on which method was used to obtain them. The order in which the chains were presented to the evaluator was randomized across topics. Users compared each pair of chains and evaluated four criteria:

- *Relevance:* Which chain is more relevant to the topic?
- *Coherence:* Which chain is more coherent?
- *Broadness:* Which chain is broader?
- *Number of coherent and relevant documents:* How many documents from each chain form a coherent and relevant story?

For the first three criteria, each evaluator chose one of five options in terms of each criterion: one chain is better than the other (two options), one chain is slightly better than the other (two options), and both chains are similar. The fourth criterion was evaluated for each chain.

Table 3.1 presents the user study results for the first three criteria: coherence, relevance, and broadness. In this case, the results for each topic present a different picture of the quality of the stories generated by *metro maps* and our approach. Based on these results, we can observe that there is a trade-off between coherence and broadness, i.e., the more coherent a chain appears to the evaluator, the less broad it seems, since the topics do not seem to diverge significantly. Table 3.2 presents the percentage of users that selected a particular number of documents that are relevant and coherent w.r.t. the generated story.

Analysis of results for Syria topic

For the *Syria* topic, the evaluators preferred the story generated by *metro maps* in terms of coherence, with 66.7% of the users scoring this chain as slightly or more coherent, while the

Table 3.1: Results of user evaluation between two pairs of chains built using the *metro maps* approach and our diffusion-based algorithm comparing coherence, relevance and broadness.

Criterion	**Topic**	**Metro maps is better**	**Metro maps is slightly better**	**Both are similar**	**Our approach is slightly better**	**Our approach is better**
Coherence	Syria	50.0%	16.7%	25.0%	8.3%	0.0%
	Iran	0.0%	8.3%	8.3%	33.3%	50.0%
Relevance	Syria	16.7%	16.7%	66.7%	0.0%	0.0%
	Iran	0.0%	25.0%	41.7%	25.0%	8.3%
Broadness	Syria	0.0%	0.0%	41.7%	16.7%	41.7%
	Iran	41.7%	33.3%	25.0%	0.0%	0.0%

Table 3.2: Results of user evaluation for number of coherent and relevant documents in each chain using the *metro maps* approach and our diffusion-based algorithm.

Topic	**Syria**		**Iran**	
Score	**Metro maps**	**Our approach**	**Metro maps**	**Our approach**
0	0.0%	0.0%	0.0%	0.0%
1	0.0%	0.0%	0.0%	0.0%
2	0.0%	41.7%	16.7%	0.0%
3	33.3%	33.3%	66.7%	16.7%
4	25.0%	8.3%	16.7%	33.3%
5	41.7%	16.7%	0.0%	50.0%

rest is split between the same level of coherence and our approach being slightly better. In terms of relevance, 66.7% of the evaluators selected that both chains have the same level of relevance, while the rest considered the chain created with the *metro maps* approach to be slightly or more relevant. For broadness, the chain generated by our algorithm was chosen as slightly or more broad by 58.3% of the evaluators, while the rest considered both chains to have the same broadness.

To understand these findings, we performed an in-depth analysis of the contents of each article in both chains. For the *metro maps* approach, documents A1, A2, and A3 are very similar in that the central theme is the number of casualties of the civil war in Syria; document A4 discusses an effort to find a political solution to the conflict, while document A5 is about an upcoming election. The first three documents are very similar, but they revolve around very similar events that happened in a very short timespan, which is what we are trying to avoid.

In contrast, the chain generated by our approach starts with document B1, which discusses allegations of torture against the Syrian police. Regular acts of violence by the government against civilians are considered the main reason the war started. Document B2 describes the state of the civil war at its initial stage, as well as a controversy related to the then head of the Arab League observer mission in Syria. The next document, B3, discusses the flow of refugees from Syria to Jordan, which started in the late months of 2011, once the Syrian civil war was in full swing. Document B4 describes once again the actions of the government against the rebels. This article was published on the same day that the Syrian government was accused of using chemical weapons. Finally, document B5 discusses the prospect of a presidential election in Syria.

The chain of documents generated by our approach presents a larger picture that starts with a possible cause of the conflict, depicts its various manifestations and evolution, and ends in the possibility of having a presidential election that could end the conflict. However, it received low scores in the user evaluation. We suspect this is the result of the users not having sufficient background knowledge of the factors and events relevant to the Syrian civil

war. If the users devoted more time to analyze all of the provided evidentiary documents thoroughly or possessed a broader contextual understanding of the Syrian civil war, then our approach would have likely obtained higher scores. Table 3.2 shows the percentage of scores that each chain received. The chain generated by *metro maps* received an average score of 4.1, while our approach had an average score of 3.0. This supports our assertion that documents B1 and B4 might not seem relevant or coherent with the story at first glance, but after further analysis, we consider our story to have more value in terms of insight.

Analysis of results for Iran topic

For the *Iran* topic, the evaluators preferred the story generated by our approach in terms of coherence with 83.3% of the users scoring our chain as slightly or more coherent, while the rest is split between the same level of coherence and the *metro maps* approach being slightly better. In terms of relevance, 41.7% of the evaluators selected that both chains have the same level of relevance, and in this case, the rest of the answers are spread among the two methods. The *metro maps* chain was picked as slightly or more broad by 75% of the evaluators, while the rest considered both chains to have the same broadness. However, in terms of coherent and relevant documents, the chain generated by *metro maps* received an average score of 3.0, while our approach had an average score of 4.3. We consider these results as evidence that our approach can provide more insightful stories.

In this case, *metro maps* documents C1 and C2 discuss missile drills. Document C3 describes a presidential visit to improve the relationship between Islamic countries. Document C4 discusses the willingness of the Iranian President to talk with the Western leaders about their nuclear program. Finally, document C5 is about an increase in the price of gasoline in Iran, probably caused by the sanctions imposed because of their nuclear program. We believe that this chain is not very coherent, and thus the chain was scored as very broad.

The results from our diffusion-based method start with document D1 describing that

Russia was considering to condemn Iran's nuclear plan. Document D2 talks about a nuclear treaty between the United States and Russia but also mentions the assistance that the Russian government provided to develop Iran's nuclear program. The next article (D3) mentions that the negotiations over the nuclear program were set to resume, only to be stopped again less than a year later (D4), which resulted in several sanctions that the U.S. decided to enforce (D5).

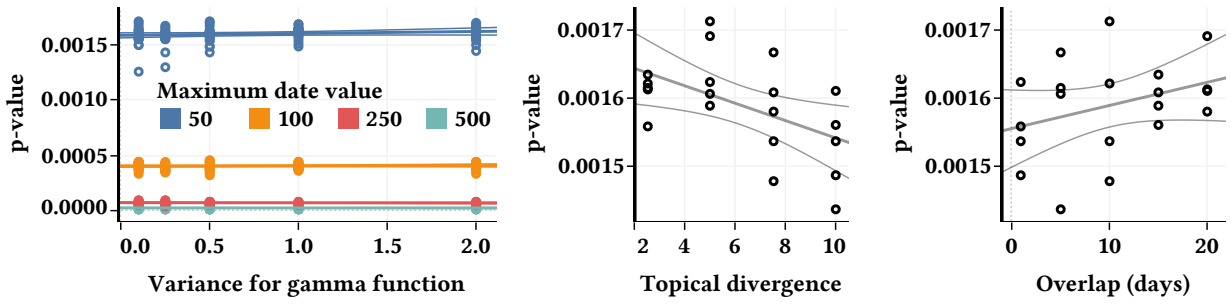
3.4.4 Statistical significance analysis

Evaluation using precision, recall, and other common metrics does not apply in the context of our work because of the lack of labeled datasets suitable for our task. An alternative approach, other than the user studies and the use of unsupervised metrics such as the dispersion coefficient, is to perform a statistical analysis of the significance (p-value) as a sanity check to make sure that our formulation obtains results that have a very low probability of being obtained by random chance. Our objective function has several user-modifiable parameters that can change analytic viewpoints. In this section, we explore how the significance changes with varying parameters.

For our problem, we defined p_{TP} as the p-value for the turning points vector $\mathcal{T}$. We generated m random samples $\mathcal{T}_S$ which are of the same size as $\mathcal{T}$. Then, we compared each of the m random samples with $\mathcal{T}$ by computing their element-wise difference and comparing it to a tolerance β, i.e., $(\mathcal{T}_s - \mathcal{T}) \leq \beta$. We counted the number of times this condition was satisfied in an auxiliary variable a. Finally, we returned $p_{TP} = \frac{a}{m}$.

The experiments were conducted by computing the p_{TP} of several different configurations. We precomputed 100,000 random samples. Then, we used several random seed document sets and ran the optimization routine for different configurations. We finally averaged the p values across the documents.

One of the configurations relates to the publication dates of the articles. The dates are scaled to a continuous range. In Fig. 3.6(a), it is evident that the p_{TP} value significantly decreases while increasing the *maximum date* value. This is because the range for the



(a) (b) (c)

Figure 3.6: Significance of turning points vector vs. (a) variance of gamma function, (b) topical divergence and (c) date overlap penalty. Each data series includes a predicted trend line as well as upper and lower 95% confidence lines.

turning points increases with the scaling. Fig. 3.6(a) also shows that our approach generates statistically significant results with any value of the variance for the gamma function $\hat{\sigma}^2$, and that the change in significance while varying this parameter is limited.

The standard deviation parameter for overlap σ controls the width of a Gaussian distribution. A penalty is added if any of the previous turning points fall within this curve. Figure 3.6(c) shows that as the overlap σ increases, the p_{TP} value increases. The maximum topical divergence parameter is used during the candidate generation stage to filter documents that are above this value. Figure 3.6(b) shows that when the topical divergence increases, the p_{TP} value decreases, albeit slightly. In general, our approach generates statistically significant results, even with varying parameters.

3.4.5 Dispersion coefficient versus number of segments

For the analysis presented in Section 3.4.4, the number of segments (or turning points) was also modified to observe the effect on the p_{TP} value. However, increasing the number of segments to more than three resulted in almost all cases giving a significance value of zero, which means not a single configuration matched with the 100,000 random samples.

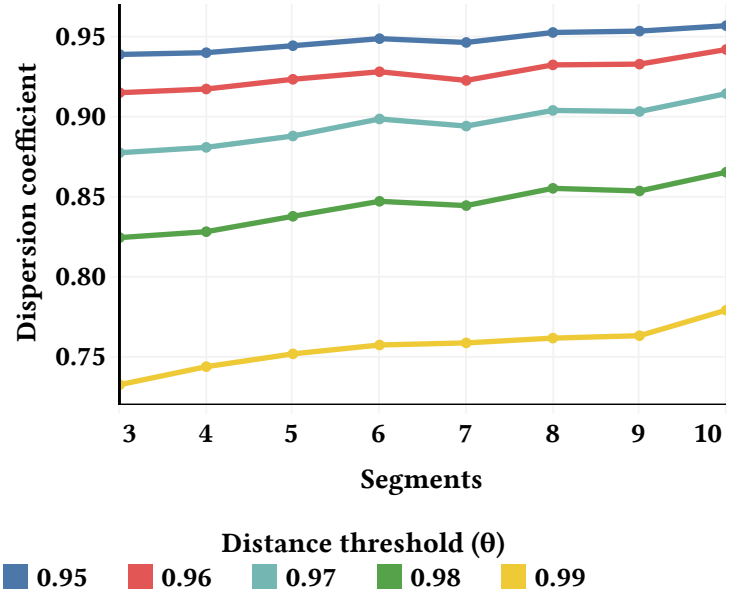


Figure 3.7: Dispersion coefficient versus number of segments $|\mathcal{S}|$ with $\theta = [0.95, 0.96, 0.97, 0.98, 0.99]$.

Thus, to evaluate how our algorithm performs while varying the number of segments $|\mathcal{S}|$, we performed a similar evaluation but measuring the dispersion coefficient presented in Section 3.4.2.

The threshold distance θ was varied from 0.95 to 0.99 in 0.01 increments. We obtained the dispersion coefficient for these values of θ as the average for 18 different seed documents and only using news articles from a window of five years from publication. Fig. 3.7 presents the average dispersion coefficient per number of segments $|\mathcal{S}|$. The figure indicates that an increase in the number of segments improves the dispersion coefficient, which indicates the existence of a high number of turning points, as expected in a five-year window.

3.4.6 Local optimization: repeatability of concepts

One of the crucial elements of local optimization, as used in our framework, is that it may produce multiple results with different executions for the same set of seed documents

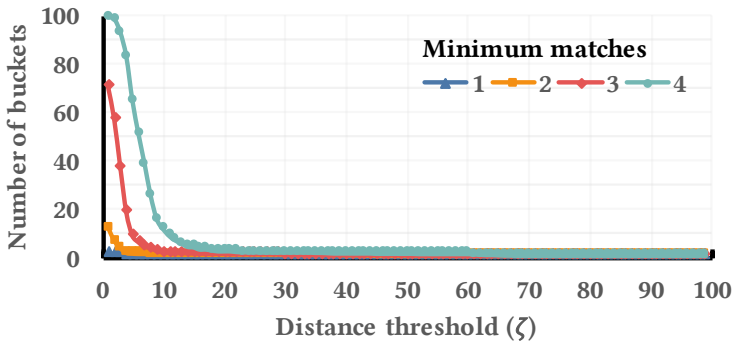


Figure 3.8: Number of buckets as a function of distance threshold ζ, with varying minimum matches.

and configurations. From an analytic perspective, multiple results for the same seed set provide a deeper understanding of evolution. However, too much diversity in the results for the same seed set may be overwhelming for users. In this experiment, we analyze the repeatability of the results, in particular for the turning points, for the same set of seed sets. We selected random seed documents from different topics and repeated the optimization 100 times for each seed, keeping track of the resulting turning-point vector, each time with a different initialization for the optimization routine. The collected vectors were compared pairwise and grouped into *buckets* based on similarity. To define similarity in this context, we first define a match as when the distance between two turning points from different vectors is below a threshold, ζ. Two vectors are similar if the number of matches is above a *minimum match* parameter. Ideally, the number of buckets should be small (close to 1). If the number of buckets is 100, this means that none of the pairs of vectors fulfill the conditions mentioned above.

In Fig. 3.8, we observe how the average of the number of buckets for the seed documents changes with respect to the distance threshold ζ and a *minimum matches* score. ζ is varied from one to 100, since this is the range of the publication dates in this experiment, while the number of *minimum matches* is changed from one to four. Figure 3.8 shows that as the distance threshold increases, the number of buckets rapidly decreases. This indicates that,

even with different initializations of the optimization routine with multiple executions for the same seed, we obtain similar chains and time segments. This helps the user by keeping the analysis focused on a relevant set of evolving concepts for the same seed document.

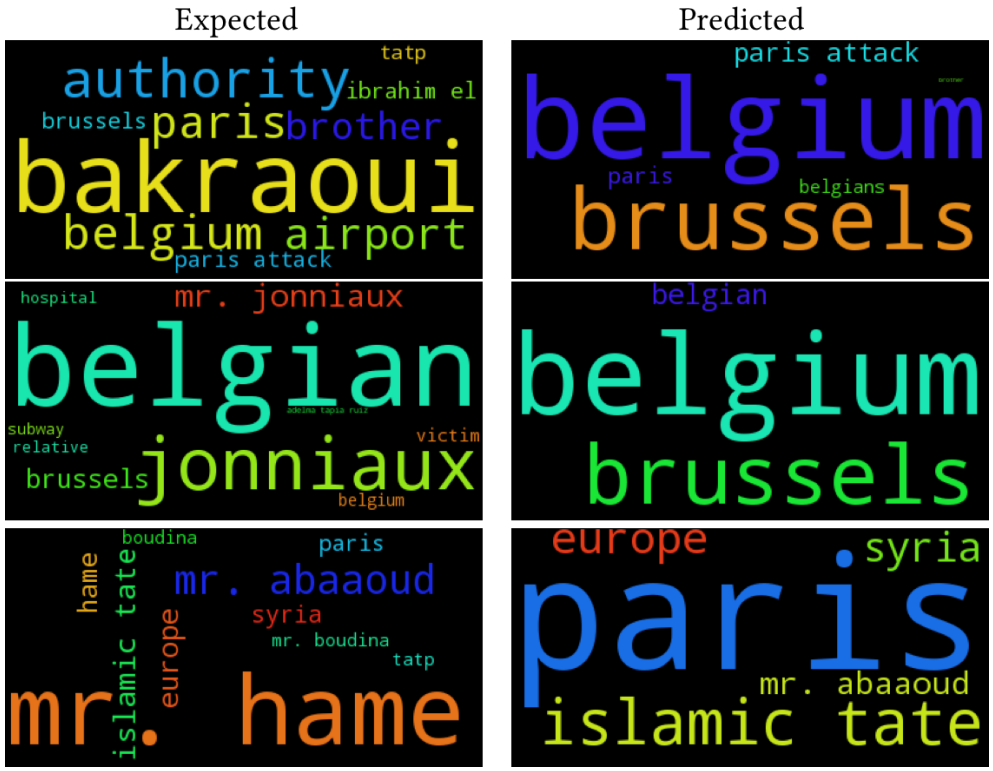
3.4.7 Prediction

Our storytelling framework provides past documents as an evolution of the story of a given seed document. All the past documents have associated weights reflecting their importance in the evolution. These relevant documents can be used to study the evolution over time of the prominent entities found in the seed document. In this experiment, we examine if we can leverage these relevant documents to not only study the evolution of the entities in the past but also to predict, using simple linear regression, the future evolution of these entities. A more specific question is: which entities will appear in an article published in the future given a seed document?

For this experiment, we divided the set of relevant documents based on the segments into a training dataset (four segments) and a testing dataset (one segment). Next, we generated a table that consists of all pairs of words in all pairs of the documents of the training set. This table contains the difference in publication dates, as well as a pair of entities and corresponding tf-idf weights that appear in a pair of documents. That is, the training data reflects how far in time a pair of entities may appear and what are the tf-idf scores of those entities when they appear. We built a linear regression model for each of the terms appearing in the *future* documents, taking into account the difference in publication dates as an extra feature.

When a new prediction is requested, we use the linear regression models that include the entities of the seed document, to predict the weight of the entities in the *future* document. In Fig. 3.9(a), we show the word-cloud of a sample seed document. We then use our algorithm to predict how the word-clouds of three future documents would look like in a range between 4 and 10 days after the publication date of the original article. The actual and predicted word-clouds are presented in Fig. 3.9(b). As can be seen, even with a simple

(a) A word-cloud of a seed document.



(b) On the left, we show the ground truth while on the right, we show the predicted word-clouds. The words are sized according to the original and predicted weights, respectively. The publication date difference between the seed document and the predicted documents ranges between 4 and 10 days.

Figure 3.9: An experiment on predicting a future document.

linear regression-based prediction model, for most of the cases, the predicted documents have a match of at least two of the entities from the ground truth (e.g., *Paris attack*, *Belgium*, and *Paris*). The advantage of using the output of our model for prediction is that we exploit both the similarity of documents within a segment and the diffusion of a topic across segments, to obtain better results.

We expect that a more sophisticated algorithm would be able to improve these results. The main disadvantage of this method is that if no entities from the seed document appear in the training set, then no entities can be predicted, and of course, unseen entities are not predicted.

3.5 Conclusions

In this chapter, we presented a storytelling framework that discovers the evolution of news stories from large news archives. The results show that the evolution of entities over time follows a diffusion process. We have presented a case study that demonstrates the predictive power of using the story-level context of a particular topic of interest as training data to build a supervised learning model. The generated model can be used to predict, to a certain degree, the future context of the event presented in the seed document.

Chapter 4

Tracking Semantic Evolution using Time-Reflective Text Representations

4.1 Introduction

With the current pace of publication of text data by every sector of society, the necessity to consider text publications as an evolving stream of data is increasing. There is no doubt that the current abilities to transform unstructured text collections into a structured representation provide us with ample analytic opportunities by leveraging many data mining and machine learning algorithms that have been designed exclusively for structured data. However, to serve even a larger set of analytic needs, modern text mining is slowly drifting toward the analysis of temporal aspects of text [48, 127]. The ability to represent unstructured text with a temporal context is in its infant stage. This chapter discusses the needs and expected properties of a time-reflective representation of text data along with a preliminary implementation that reveals the potential of such time-reflective representations.

In this chapter, we present a smooth and continuous time-reflective representation of temporal text data, which is diffusion-centric in the time dimension. We chose to use a diffusion-centric approach based on the results obtained in Chapter 3. The model takes into account the co-occurrence of words within the same context for a particular timestamp, thus enabling the model to capture drifts in short timespans. We incorporate *diffusion* of the word counts across timestamps to smooth the word vectors over time. The proposed representation allows us to smoothly track the meaning of every word in terms of their neighborhood over consecutive timestamps. While a static representation may provide

the context of each word of a corpus as a set of nearest neighbors, the time-reflective representation can capture how the context of a word changes over time.

Figure 1.1 (a) provides an example of the neighborhood of the word *cloud* using a static representation. Figure 1.1 (b) shows how the context words of the word *cloud* evolved over time. Prominent context words of *cloud* using a static representation are *vapor*, *rain*, *Google Drive*, and *Dropbox*. The history of how the neighboring words came into the context of *cloud* is not apparent when using static embeddings. In the time-reflective representation (Figure 1.1 (b)), it is evident that the terms *Google Drive* and *Dropbox* became more prominent in recent years.

To study semantic evolution, the corresponding text corpus must be large and timestamped so that enough evidence can be gathered by a model to form a chain of concepts over time. News articles and scientific papers are good examples of timestamped text corpora. In news articles, a time-reflective representation can help in studying how a particular event evolved in society. In scientific articles, e.g., in a biomedical corpus, scientists can discover how a specific medical concept evolved in the past.

The contributions of the work presented in this chapter are summarized as follows:

1. We present a new diffusion-based method of generating co-occurrence-based time-reflective vector space models for temporal text corpora.

2. We introduce the *neighborhood monotony metric*, which allows us to quantify the semantical or contextual change of a word over time.

3. We compare the new model with regular co-occurrence-based and dynamic-embedding-based vector space models.

4.2 An Ideal Temporal Text Representation

Semantic evolution is not yet quantifiable in the literature and cannot be easily modeled using data mining and machine learning algorithms. Moreover, the evolution of some words

can be slow, while in other words, it can be fast. For example, the word *husband* in the early 14th century meant *house-owner*, and in the next several hundred years, the meaning changed to *a marital status*. In contrast, words like *cloud*, *apple*, and *viral* have rapidly changed their meaning during the last two decades. The ideal temporal text representation model should have the following features:

1. The representation should include a temporal dimension to be able to model evolution over time.

2. The representation should support vector space modeling. That is, each word must have a vector in each timestamp so that the vectors can be tracked over time for each word.

3. The changes of a word vector over time should be continuous and smooth, e.g., the vectors should not completely shift in space from one timestamp to the other. This property allows the usage of time series analysis algorithms.

4. The representation should be able to capture significant changes in the semantic and contextual similarity of a word in a short period (e.g., between consecutive timestamps).

5. The representation should be as compact (i.e., low-dimensional) as possible to make it suitable/feasible for use with big data analytics algorithms.

There are two main types of vector space models for text data: co-occurrence- and embedding-based models. There are important properties associated with each of these representations, as illustrated in Table 4.1. Word2vec [89] and dynamic Bernoulli embeddings [111] are embedding-based methods, while tf-idf [122] and the model presented in this chapter are co-occurrence based models. As noted in Table 4.1, this chapter targets most of the expected properties of an ideal time-reflective representation except for the low-dimensionality of the vectors.

Table 4.1: Feature comparison between vector space models ©2018 IEEE.

Method	**Includes temporal dimension**	**Smooth evolution of a single vector**	**Captures drifts in long periods**	**Captures drifts in short periods**	**Results are low-dimensional**
word2vec [89]					✓
Dynamic Bernoulli embeddings [111]	✓		✓		✓
tf-idf [122]				✓	
Chapter 4 approach	✓	✓	✓	✓	
Ideal representation	✓	✓	✓	✓	✓

To highlight the potential of the proposed time-reflective representation of a natural language, we use the task of tracking the semantic evolution of words. Throughout this document we use the phrase *semantic evolution* and *contextual evolution* interchangeably. The term *semantic evolution* in our work does not refer to *meaning* as seen in the dictionary. Rather, *semantic evolution* between a pair of words refers to how similar the context of the words are. *Context* is more time-dependent, as illustrated in Figure 1.1.

Pros and cons of co-occurrence-based models: Co-occurrence-based approaches are known to be accurate regarding inferring the meaning of a given word based on the words that co-occur with the word of interest. The disadvantage of a co-occurrence-based approach — when using it in tracking temporal semantic evolution — is that the distribution of words that appear at every timestamp is usually sparse. As a result, it is possible that some words do not appear at all for a particular timestamp, making the representation discontinuous. A word might have a neighborhood that does not describe an accurate context of the word when the appearance of a word is infrequent in a particular timestamp. Another disadvantage of this approach is that it requires $O(mnt)$ storage, where m is the number of documents, n is the number of words, and t is the number of timestamps in the

corpus.

Pros and cons of embedding-based models: Word embeddings are a low-dimensional representation of a text corpus. The model is suitable for analyzing large text datasets due to the compression of these representations. Static embeddings require $d \times n$ storage where d is the length of the vector of each word, and n is the number of unique words in the corpus, resulting in a $O(n)$ space requirement. If time-reflective embeddings are created, the storage requirement is $O(nt)$, given t is the number of timestamps. One drawback of using word embeddings is that the vectors do not have interpretable values, unlike co-occurrence based approaches. The vectors provide a holistic information space in which the nearest neighbors of each word represent the context of the word.

4.3 Problem Description

Throughout this proposal, we focus exclusively on timestamped text corpora, such as collections of news articles or scientific publications. Let $\mathcal{D} = \{d_1, d_2, \ldots, d_{|\mathcal{D}|}\}$ be a corpus of $|\mathcal{D}|$ text documents and $\mathcal{W} = \{w_1, w_2, \ldots, w_{|\mathcal{W}|}\}$ be the set of $|\mathcal{W}|$ noun phrases extracted from the text corpus $\mathcal{D}$. Each document d contains noun phrases from the vocabulary ($\mathcal{W}_d \subset \mathcal{W}$) in the same order as they appear in the original text of d. Every document $d \in \mathcal{D}$ is labeled with a timestamp $t_d \in \mathcal{T}$, where $\mathcal{T}$ is the ordered set of timestamps.

4.3.1 Expected outcome

The main goal of this chapter is to obtain a time-reflective vector space model from corpus $\mathcal{D}$. Thus, for every timestamp $t \in \mathcal{T}$, we seek to obtain a vector representation of every word $w \in \mathcal{W}$. Ideally, the vectors of words that appear frequently in the same *context* of word A should be *spatially closer* to the vector of word A, than those that appear less often.

4.4 Methodology for Diffusion-Based Semantic Tracking

In this work, we focus on a specific application of vector space models — tracking semantic or contextual evolution. We focus on such a task because it allows the user to determine the meaning of a particular word at different times. The context of a word can be retrieved from the nearest neighbors of a word vector. Our approach consists of two main components: (1) temporal diffusion of words and (2) temporal neighborhood retrieval.

4.4.1 Temporal diffusion

The distribution of the frequency of a word over time can be severely irregular, in particular for words or noun phrases that suddenly appear at a particular timestamp, or that are used sporadically. Furthermore, based on *diffusion theory* [7], which refers to the change of the distributional patterns of a phenomenon over time, we assume that the meaning of a word, and consequently its vector representation, diffuses over time.

To smooth the word vectors over time, we assume that every document is present in every timestamp but with a higher probability for the timestamps closer to when the document was initially published. We use a Gaussian filter to *diffuse* the contribution of the document smoothly before and after the publication date of the document. The filter uses a sliding window, going from the first to the last timestamp. The contribution of a document d increases the closer its timestamp t_d is to the current timestamp t. The tf-idf weight of a word can be modified at each timestamp with Equation 4.1.

$$\hat{w}(w, d, t_d, t, \varsigma) = \left(\frac{1}{\sqrt{2\pi\varsigma^2}} e^{-\frac{(t_d - t)^2}{2\varsigma^2}} \right) \cdot \left(\frac{(1 + \log(f_{w,d})(\log \frac{|\mathcal{D}|}{\delta_w})}{\sum_{w' \in \mathcal{W}_d} \left((1 + \log(f_{w',d})(\log \frac{|\mathcal{D}|}{\delta_{w'}}) \right)^2} \right), \tag{4.1}$$

where $\hat{w}$ is the weighted value at timestamp t for the noun phrase $w \in \mathcal{W}$ in document $d \in \mathcal{D}$, which was published at timestamp t_d. The term $f_{w,d}$ represents the term frequency of noun phrase w in document d, δ_w is the number of documents that contain noun phrase

This is an example of window-level context .

Figure 4.1: Resulting window-level context (underlined words) of the word *window-level* assuming a window size of 1 ©2018 IEEE.

w, and $\mathcal{W}_d$ is the set of noun phrases that appear in document d. ς represents the standard deviation of the Gaussian distribution, and is set by the user. A large value of ς means that the diffusion of concepts will be slow over time. A small standard deviation will allow capturing short-term changes in meaning, but makes the model more susceptible to noise.

4.4.2 Computing the nearest neighbors of a word per timestamp

To track the semantic evolution of a given word, we perform an analysis from one year to another and construct a set of k-nearest neighbors for each year. Analyzing the changes in a word's neighborhood in consecutive years helps us understand the semantic change of a word over time.

The *context* of a word conveys the meaning and intent of the word. Selecting a context that filters out unimportant or irrelevant words can improve the quality of the retrieved neighborhoods significantly. In this chapter, we filter out every word occurrence that is not within the context of the word under analysis, and we evaluate the following contexts:

- **document-level context.** Every word in document d is part of the context of the other words in the same document.
- **window-level context.** Only the words that are within a window of a particular size from the word of interest are part of its context. For example, if the window size is 2, then the two words before and the two words after the word of interest form its context. Figure 4.1 illustrates this concept by showing the context (underlined words) of the word *window-level* with a window size of 1.

We divide the neighborhood retrieval task into three main subtasks. First, we find all the occurrences of words that belong to the context of the word of interest. Next, we set

all of the tf-idf vector entries that do not belong to the context to 0. Finally, we compute the cosine similarity between the resulting vectors for every timestamp. The neighborhood is formed by the terms that have the highest values of cosine similarity at each timestamp.

4.4.3 Evaluation of semantic evolution

The meaning of a word tends to evolve and change over time. In this chapter, we introduce the concept of *neighborhood monotony* to evaluate, quantitatively, how much the meaning of a word changes over time, based on the word's neighborhood over consecutive timestamps. This metric helps in evaluating semantic evolution by allowing to estimate the degree to which the semantical or contextual meaning remains steady. We name such steadiness of the meaning of a word throughout a timeline *neighborhood monotony*. The neighborhood monotony, $\bar{F}$, is computed by taking the average of Jaccard similarities between the neighborhoods of a word in every consecutive pair of timestamps. Equation 4.2 formulates the average neighborhood monotony of a word w.

$$\bar{F}(M, w, k) = \frac{1}{|T| - 1} \sum_{i=0}^{|T|-1} \frac{M(t_i, w, k) \cap M(t_{i+1}, w, k)}{M(t_i, w, k) \cup M(t_{i+1}, w, k)}, \tag{4.2}$$

where $M(t_i, w, k)$ is the vector space model with respect to word w at timestamp t_i, and k is the size of the neighborhood.

If the k-neighborhood of a word remains unchanged across all timestamps, the word is considered to exhibit a completely *monotonous* evolution, and the average neighborhood monotony will be 1.0. If the neighborhood changes completely for every pair of consecutive timestamps, then the average neighborhood monotony will become 0.0.

Additionally, we introduce the concept of *minimum* and *absolute* neighborhood monotony. *Minimum neighborhood monotony* refers to the pair of consecutive timestamps where the Jaccard similarity between neighborhoods is lowest. It identifies significant changes in the meaning of a word in a single point in time. *Absolute neighborhood monotony* refers to the Jaccard similarity between the neighborhoods of the first and last timestamps. It helps

identify very monotonous concepts that did not evolve.

4.5 Experimental Results

For the experiments in this chapter we use a corpus of 613,545 PubMed abstracts [16] related to the query *anticancer agents*, published between January 1998 and April 2018. We preprocessed each abstract by extracting its noun-phrases using the SpaCy Python library [55]. The corpus contains 15,988,890 unique noun-phrases. We selected the top-10,000 noun-phrases based on frequency.

We evaluate our method by comparing its performance with that of a regular tf-idf model and the state-of-the-art dynamic Bernoulli embedding model.

In the tf-idf model, each document $d \in \mathcal{D}$ is represented as an $|\mathcal{W}|$-length sparse vector, which contains the normalized *tf-idf* weights of each noun phrase $w \in \mathcal{W}$. To compute these weights we use tf-idf with cosine normalization [57].

For dynamic Bernoulli embeddings, we use the code provided by Rudolph and Blei [111]. The parameters of the model are selected as described in [111]. The optimal configuration for the results presented in [111] is not provided in the paper. Therefore, we performed a thorough exploration of different configurations and selected the model that provides the best test log-likelihood, which was -2.71. The parameters used in our experiments are:

- embedding size: 100
- window size: 2 (i.e. 2 words on each side)
- negative samples: 20
- number of batches: $100000 \in [10000, 100000]$

For our own model (Equation 4.1), we set the temporal diffusion parameters to the following values:

- standard deviation of the diffusion over time: 1.0

- context type: window
- window size: 2 (i.e. 2 words on each side)

In this chapter, we seek to answer the following questions.

1. How well does our algorithm track the evolution of specific medical terms? (Section 4.5.1)
2. How does our method compare qualitatively to other methods in terms of tracking semantic evolution? (Section 4.5.2)
3. How sensitive is our vector space model to changes in the hyperparameters? (Section 4.5.3)
4. How does our method compare quantitatively to other methods such as regular tf-idf and dynamic embeddings in terms of neighborhood monotony? (Section 4.5.4)

4.5.1 Case study

In this section, we present a study in tracking semantic evolution using our algorithm for two medical terms: *leukemia* and *gastric cancer*. We limit our analysis to the top-16 nearest neighbors for each timestamp, from which we present the most interesting words that show significant changes over time. The results are promising because, as we describe next, the neighborhoods help explain the context or contemporary state-of-the-art related to a particular term.

***Leukemia* neighborhood evolution**

Figure 4.2 shows the evolution of the neighborhood of the term *leukemia* in medical abstracts over the last twenty years. The early neighborhood includes the term *GVHD*, which is a disease frequently related in the literature to bone marrow transplantation and has been identified as the most critical issue in the treatment of relapsing leukemia patients [90].

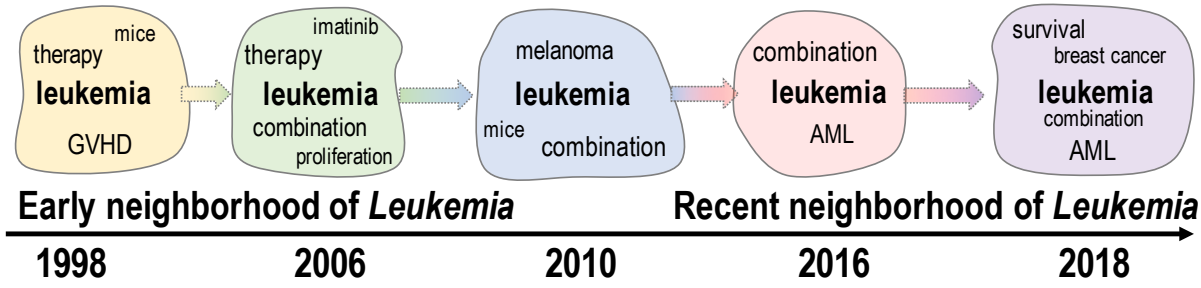


Figure 4.2: Neighborhoods of the word *leukemia* in biomedical abstracts over time, derived from our time-reflective model ©2018 IEEE.

Imatinib is a drug approved by the FDA as a treatment for leukemia in 2001 [29]. Around 2006, several articles were published that included long-term studies of the effects and success rates of *Imatinib* [32, 104, 66], as well as several experiments that *combine* this drug with some other treatments [145]. This could also explain the appearance of the word *combination* in the neighborhood starting in 2006.

From the observed results, we can infer that *combining* two or more treatments to try to cure leukemia has been the trend since 2006 [38, 131]. There has also been an important recent effort on curing *Acute Myeloid Leukemia* (*AML*), which is a very common, fast-growing, and known to be deadly cancer [99]. In 2018, the FDA approved a combination treatment targeting *AML* [131].

Finally, in 2018, several articles have been published exploring the possibility that the presence of Bovine *Leukemia* Virus (BLV) in humans can significantly increase the risk of *breast cancer* [10, 28]. In this case, we can infer that the word *leukemia* is more closely related to the bovine virus associated *breast cancer* and not to the commonly known human blood cancer.

Evolution of *Gastric cancer*

Figure 4.3 presents the evolution of *gastric cancer* for the last two decades. Between 1997 and 2003, many studies of *liver metastases* caused by *gastric cancer* were reported [80],

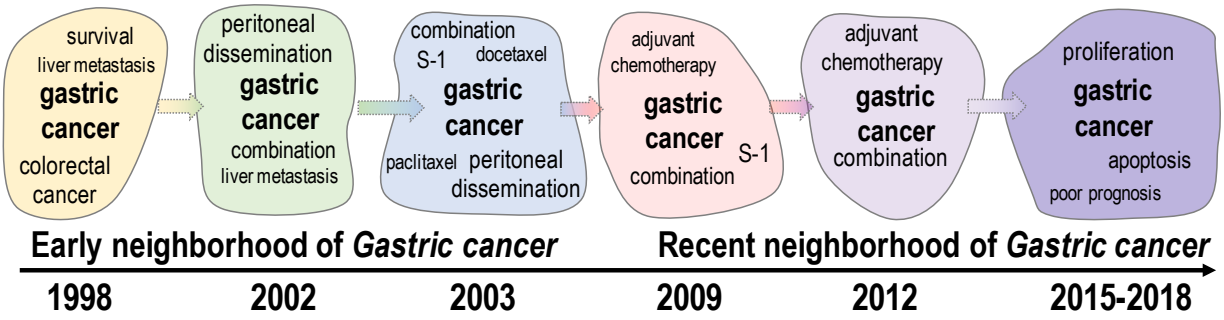


Figure 4.3: Neighborhoods of the term *gastric cancer* in different years, computed by using our time-reflective model ©2018 IEEE.

which could explain its appearance in the early neighborhood of *gastric cancer*. From 2000 to 2004, there were several findings related to the role of integrins in the*peritoneal dissemination* of gastric cancer cells, which is relevant because there is no effective treatment for *gastric cancer* after this cancer stage is reached [65].

To treat early stages of *gastric cancer*, several different *combinations* of treatments have been studied but currently there is no single treatment that is considered the most effective [6, 5]. *Docetaxel*, *placitaxel* and *S-1* are drugs that have been used to treat gastric cancer, either separately or as a *combination* [73, 92, 144]. Many of the publications between 2003 and 2005 about these drugs are phase 1 or 2 studies, which means that the drugs were in the early stages of clinical research at that time [130]. Thus, we can infer that the novelty of these drugs caused a peak in the interest of the clinical research community during this period.

In 2007, *S-1* was approved as an effective option for *adjuvant chemotherapy* for patients with resected gastric cancer. Thus, it is possible that the term *adjuvant chemotherapy* appeared within the neighborhood of *gastric cancer* because many published studies evaluated its effectiveness [68].

Recently, there has been a significant interest on inducing *apoptosis* in gastric cancer cells. *Apoptosis* refers to *auto-destruction* [142] of the harmful cells. There has also been contemporary interest on identifying substances or conditions that suppress *apoptosis* [62].

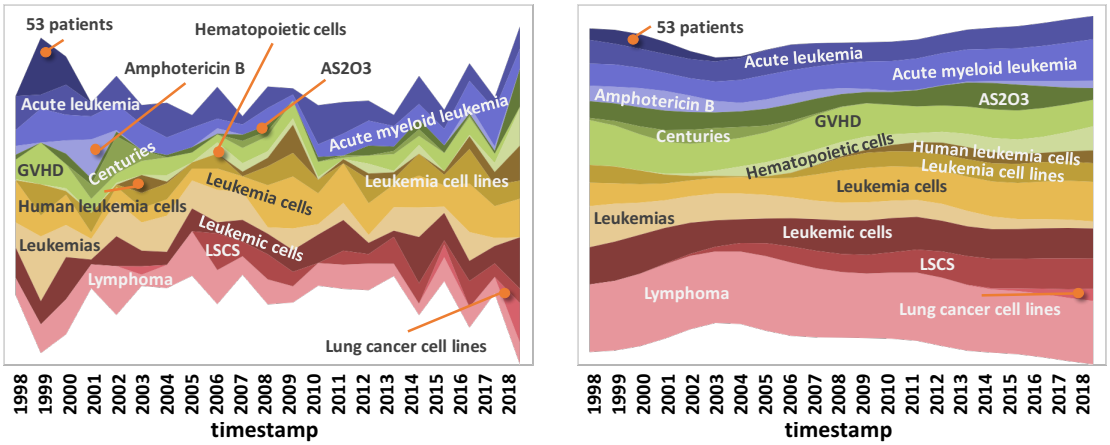
Finally, the terms *proliferation* and *poor prognosis* seem to be related to *gastric cancer* because of many recent publications that correlate substances with the proliferation or suppression of gastric cancer cells [155, 135, 161], as well as with indicators of a poor prognosis or outcome [156, 63].

4.5.2 Qualitative evaluation

In this experiment, we use the task of tracking semantic evolution to illustrate the advantages of using our model. We compare the neighborhoods over time obtained using our method with those obtained using the tf-idf model and the dynamic-embedding-based model. For this experiment, we selected the top-16 words such that their tf-idf-based vectors have the highest cosine similarity at any point in time with the tf-idf vector for the word of interest. Next, we obtained the evolving trends for these 16 words using the dynamic Bernoulli embeddings and the vectors generated by our time-reflective model (Eq. 4.1).

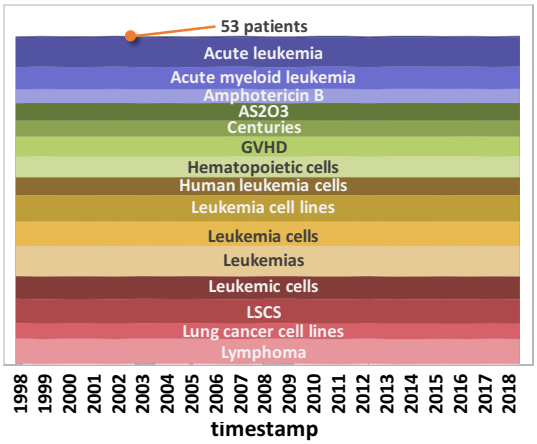
Fig. 4.4 shows the evolution of the neighborhood of the word *leukemia* at different points in time, for (a) the tf-idf method, (b) our time-reflective method (Equation 4.1), and (c) the dynamic Bernoulli embeddings method. The height of a stream at a particular point represents the cosine similarity of that word vector with the vector for *leukemia*. It is quite noticeable that the tf-idf method provides a noisy signal, while the dynamic Bernoulli method produces almost *stale* results, and clearly, none of the noticeable variations from the tf-idf model are captured. In contrast, our method captures the most noticeable variations from the tf-idf methods, e.g. for the noun phrases *1000 mg*, *lscs*, *lung cancer cell lines*, and *centuries*. Furthermore, we can observe that the trends are clearer in our method because the noise of the original data is filtered out.

Fig. 4.5 shows the evolution of the neighborhood of the term *colon cancer* using three different methods. We observe that the overall trends of similarity of most of the words are preserved by our method, in particular for terms such as *SW480 cells*, *hepatic arterial infusion chemotherapy*, and *cyclins*. The dynamic Bernoulli embeddings fail to capture



(a) Tf-idf with cosine normalization

(b) Our time-reflective model (Equation 4.1)



(c) Dynamic Bernoulli embeddings

Figure 4.4: Evolution of *leukemia* using different vector representations ©2018 IEEE.

changes in the neighborhood of the word of interest.

Fig.4.6 shows the neighborhoods of the term *diffuse large B-cell lymphoma (DLBCL)* obtained using the evaluated methods. As can be seen from Fig. 4.6a, for the first three years of data, the term *DLBCL* did not appear at all. Thus, the initial neighborhood based on the tf-idf method does not make sense at all because of the sparsity of the original data. However, our method can estimate what the neighborhood would look like based on the diffusion of the terms over time. However, the influence of the terms is diminished as the data is more sparse. In such a case, it would be a good idea to remove from the analysis the years where the word does not appear. This is of particular importance because of the 10,000 words in the selected vocabulary, 1,137 words do not appear at least in one timestamp, with an average of 3.6 timestamps where a word does not appear.

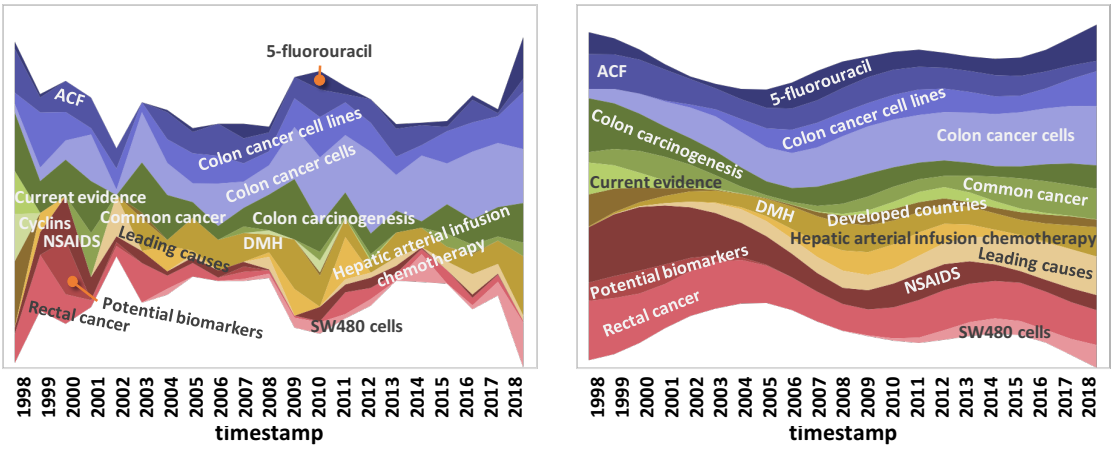
We performed another experiment using data from the National Vulnerability Dataset to verify that our approach could be extended to other datasets. Figure 4.7 shows the evolution of the term *Adobe flash player* from 2003 to 2018. In the figure we can identify two interesting trends: in the past, *user-assisted remote attackers* and *arbitrary code* where prominent, while recently terms such as *exploitable memory corruption vulnerability* (with *successful exploitation*) and *user-after-free vulnerability* are more related to *Adobe Flash*. The results for the dynamic Bernoulli embeddings and tf-idf are similar to those shown in Fig. 4.4a and Fig. 4.4c.

Our webpage[1], presents more plots for other *terms* from both datasets. All of these case studies demonstrate that our time-reflective tracking of semantic evolution smoothly captures changes in the meanings of words.

4.5.3 Sensitivity analysis

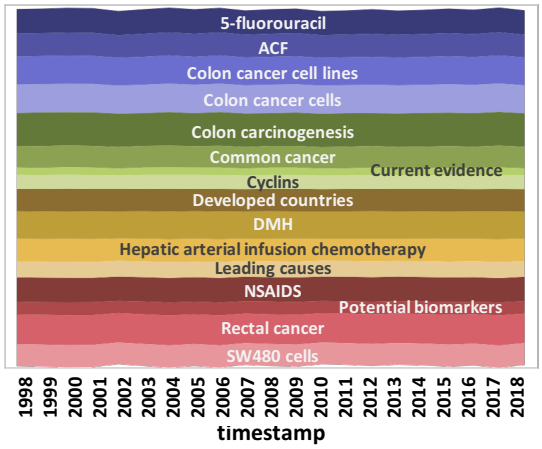
In this experiment, we evaluate the effect of performing a sweep of different values for (a) the standard deviation and (b) the word context on our model using our neighborhood

[1]https://sites.google.com/view/tracking-evolution/home



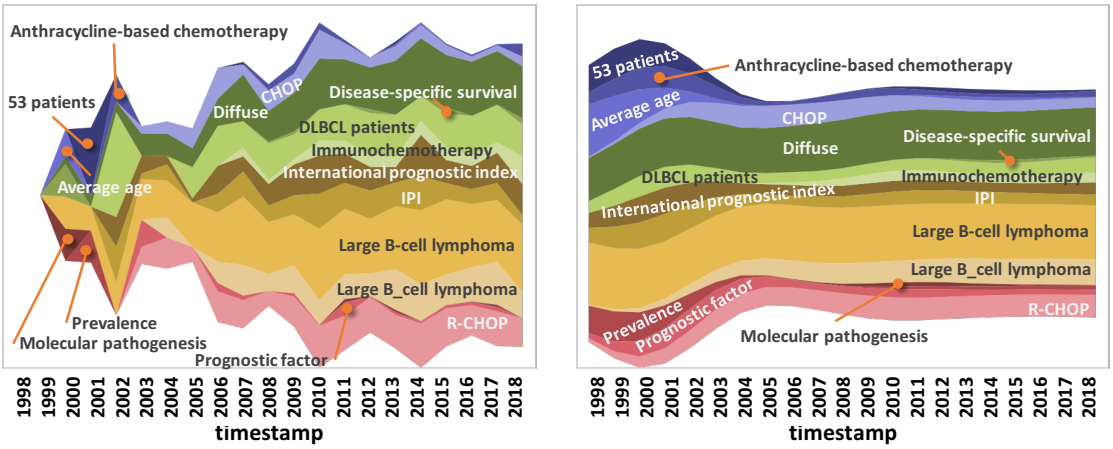
(a) Tf-idf

(b) Our time-reflective model (Equation 4.1)



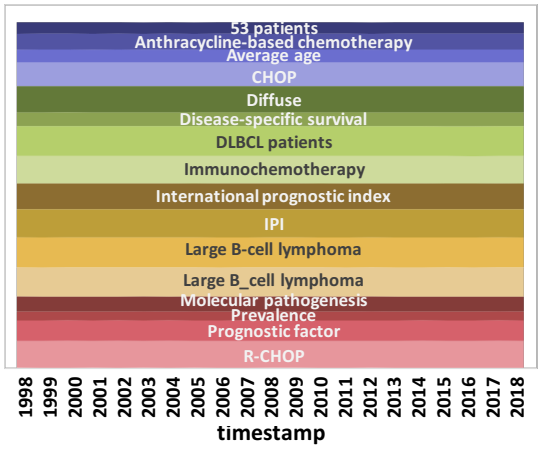
(c) Dynamic Bernoulli embeddings

Figure 4.5: Neighborhood evolution over time for the term *colon cancer* using different vector representations.



(a) Tf-idf

(b) Our time-reflective model (Equation 4.1)



(c) Dynamic Bernoulli embeddings

Figure 4.6: Neighborhood evolution over time for the term *DLBCL* using different vector representations.

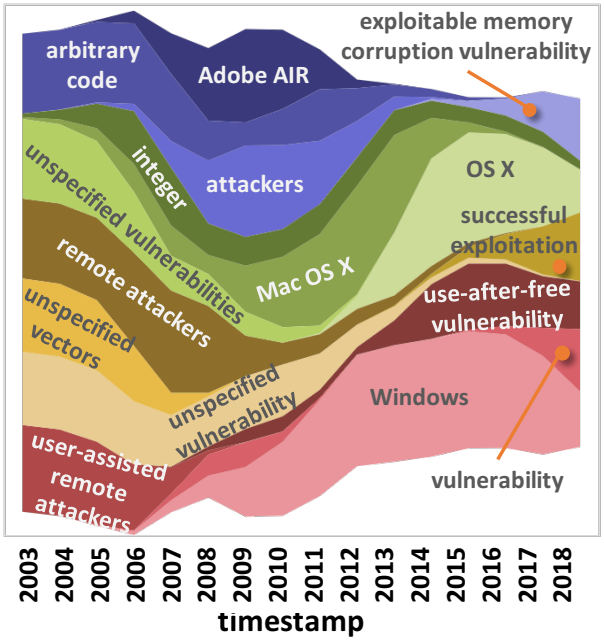


Figure 4.7: Evolution of *Adobe flash player* using our time-reflective model (Eq. 4.1) ©2018 IEEE.

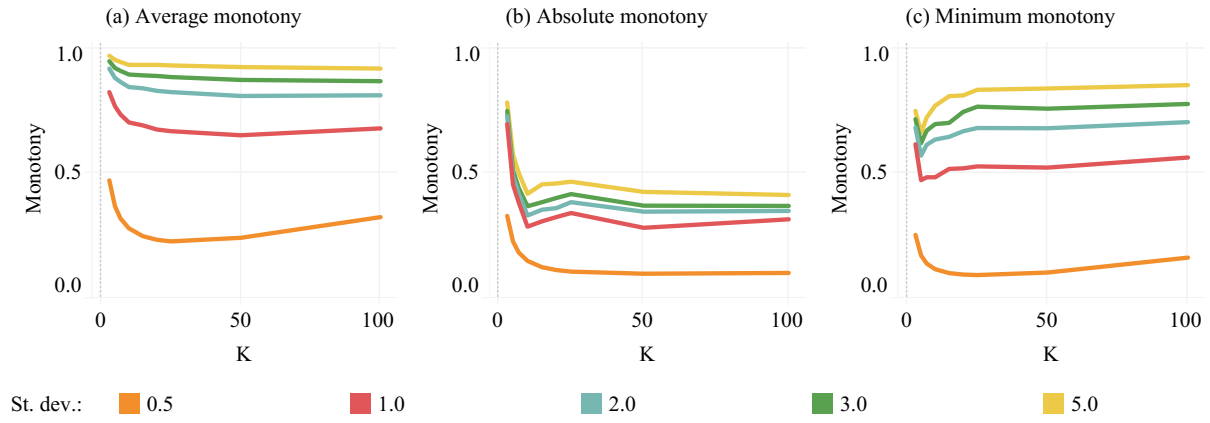


Figure 4.8: Average(a), absolute(b) and minimum(c) neighborhood monotony for different values of standard deviation ©2018 IEEE.

monotony metric (Equation 4.2). Ideally, we would like to capture a pattern that is not too monotonous (i.e., not close to 1.0), and not too unstable (i.e., not close to 0.0). We used both the document-level context and the window-level context. The window-level context was varied between window sizes of 1 to 4. We used multiple standard deviations in our model (Equation 4.1): 0.5, 1.0, 2.0, 3.0, 5.0.

Figure 4.8 shows how the neighborhood monotony metric reacts with varying neighborhood sizes (K) and varying standard deviation. As we increase the value of the standard deviation, the neighborhood monotony increases significantly. This is to be expected because increasing the standard deviation means that we are increasing the contribution of documents from distant timestamps to the current timestamp. Based on the observed results, we select the optimal standard deviation value of 1.0 since this results in an average neighborhood monotony level that is not too monotonous, and not too unstable.

Figure 4.9 shows how the different metrics of neighborhood monotony change for different neighborhood sizes (K) while varying the context of the selected representation. As we use different contexts, the neighborhood monotony values do not present a significant variation. We believe that this behavior results from the idea that the words that are closer

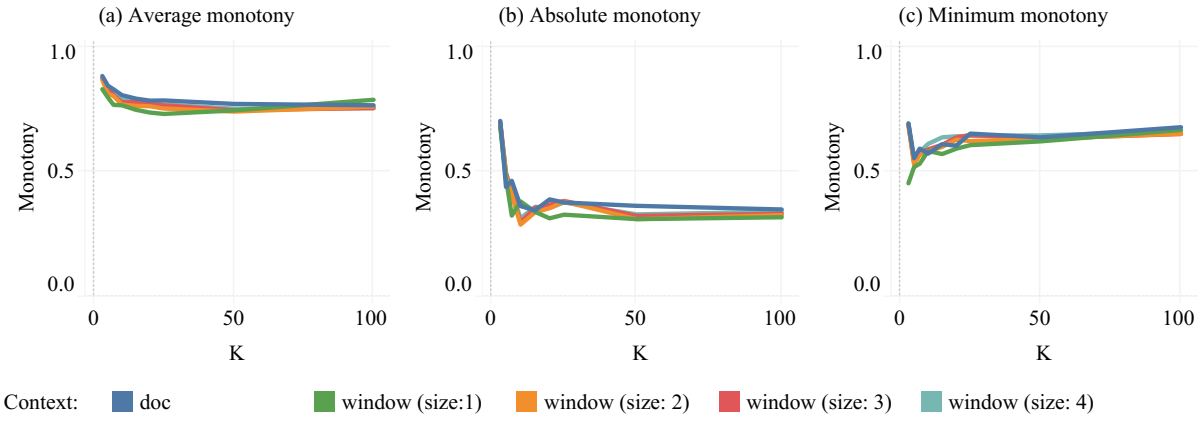


Figure 4.9: Average(a), absolute(b) and minimum(c) neighborhood monotony for different contexts ©2018 IEEE.

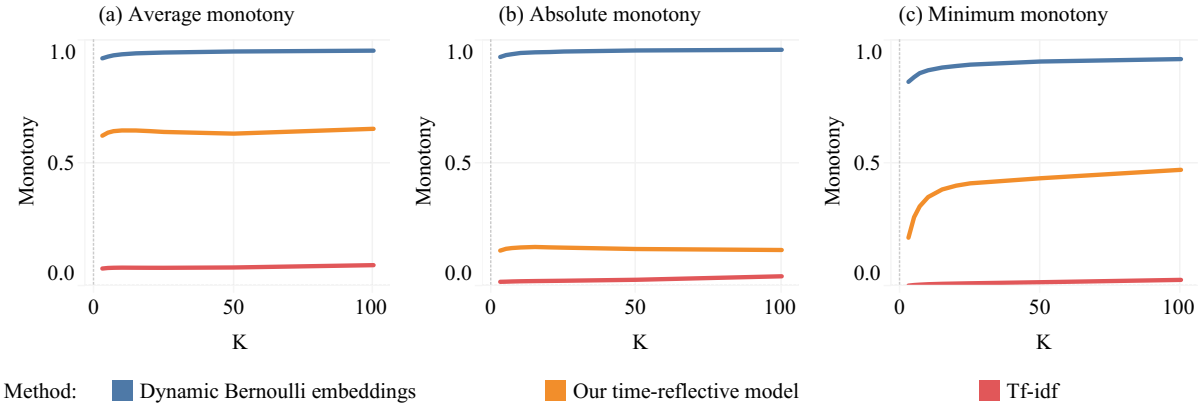


Figure 4.10: Average(a), absolute(b) and minimum(c) monotony values for different neighborhood sizes (K) obtained using three different methods ©2018 IEEE.

to the word of interest are more relevant and often appear together in many different documents. This means that the immediate context of a word is probably part of its nearest neighbors set. Thus, increasing the size of the window does not improve the detection of changes in the neighborhood significantly. Based on these results, we decided to use a window size of two words on each side of the center word, since this is also the value used by Rudolph and Blei [111].

4.5.4 Quantitative comparison with other methods

A quantitative evaluation is significantly difficult because no method can be considered a *ground truth*. Ideally, the model of a word should capture the underlying structure of the original data as well as the temporal semantic evolution. The structure should aid neighborhood selection in every timestamp and hence capture how the neighborhood changes. The changes are best represented by performing neighborhood retrieval using the tf-idf model, but tf-idf suffers from sparsity- and noise-related issues along with continuity in the time dimension. Thus, in this chapter we target construction of a model that, for evolution tracking, is good at all the following aspects: (a) filtering noisy data, (b) keeping the evolution patterns similar to the underlying original data, (c) not as monotonous as when using dynamic embeddings. We again use the previously introduced neighborhood metrics to compare our method with the neighborhoods obtained using tf-idf and dynamic Bernoulli embeddings.

Figure 4.10 illustrates the trends of the three different methods while changing the size (K) of the retrieved neighborhoods. As can be seen from the figure, our method achieves results that strike a balance between the highly monotonous results from the dynamic Bernoulli embeddings and the highly unstable results provided by tf-idf. The figure also shows the significant differences between the average, absolute, and minimum neighborhood monotony values for different neighborhood sizes when using our method. Of particular interest is the behavior of the minimum monotony, which significantly increases as the neighborhood size increases. This is the expected behavior since when the neighborhood

size increases, the probability that a high percentage of words change from one timestamp to the other decreases.

4.6 Conclusions

In this chapter, we presented a new time-reflective vector space model representation. We compared our model with other existing text representations through extensive experiments. Our method obtains a representation that: (1) can track significant changes that are observed within a short period, (2) provides a smooth evolution of the vectors over a continuous temporal vector space, and (3) uses the concept of *diffusion* to compensate for the possible sparsity of a dataset.

Chapter 5

Low-Dimensional Temporal Embeddings for Text Representation

5.1 Introduction

Word embeddings are low-dimensional vector space models obtained by training a neural network using contextual information from a large text corpus. There are many variants of word embeddings with different features, such as word2vec [89][88] and GloVe [106]. In this chapter, we focus on generating word embeddings that account for and take advantage of, the temporal nature of a timestamped document collection. Examples of timestamped document collections are news feeds, scientific publications, and blog posts. Our goal is to obtain a **low-dimensional temporal** vector space representation that allows us to study the semantic evolution of the vocabulary in the corpus.

Using the word embeddings generated by our framework, we demonstrate the task of tracking the semantic evolution of a text corpus. Fig. 5.1 shows an example of tracking the nearest neighbors of the word *president* using New York Times news articles. Section 5.4.6 contains further details about Fig. 5.1.

To generate word embeddings, our framework uses the same diffusion-mechanism for evolving *concepts* within a large text corpus presented in Chapter 4. *Diffusion* [7] refers to the changes that a phenomenon experiences over time in terms of its distributional patterns. As summarized in Table 5.1, the current temporal low-dimensional language representations fail to integrate the concept of temporal diffusion into language models effectively. Moreover, existing temporal language models [11][149][111] cannot capture both the short-

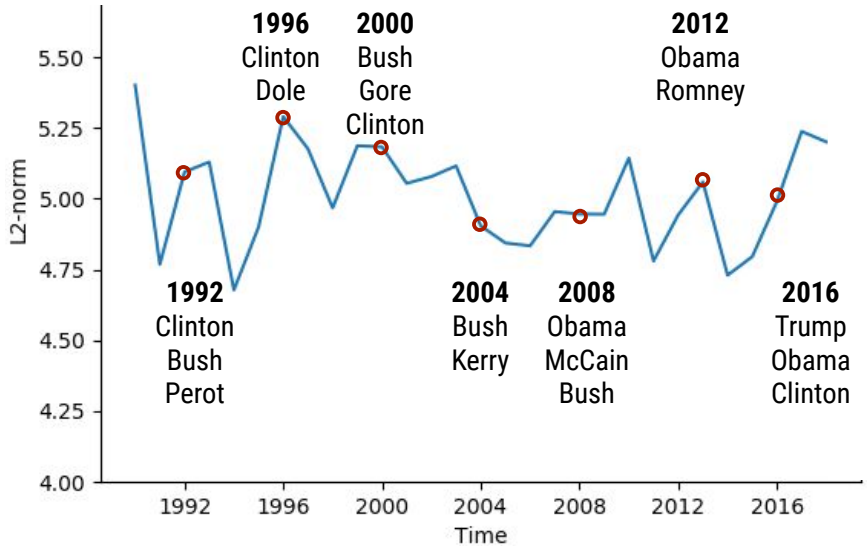


Figure 5.1: Evolution of the neighborhood of the word *president* in the New York Times using our temporal embedding method (embedding size = 64).

term and long-term drifts in the meaning of words within the same model. The advantages of a low-dimensional representation that supports both short- and long-term diffusion are as follows: (1) modeling the temporal evolution of the data is more feasible in terms of complexity and storage requirements, and (2) **low-dimensional temporal** embeddings can be used for other text mining problems such as sentiment analysis [78][83][19], inference of complementary products in recommender systems [42][84], and prediction of illegal activities in the dark web [126].

Our approach relies on using our temporal high-dimensional tf-idf representation, introduced in Chapter 4, as a baseline. This temporal tf-idf representation significantly reduces the noise from the original data. Additionally, it can capture sudden short-term changes in the corpus. Our goal is to reduce the dimensionality of this representation without losing the essential temporal diffusion information encoded in the vectors. The framework presented in Chapter 4 generates smooth tf-idf vectors for each word of a corpus at every timestamp. However, we did not use any mechanism to create a low-dimensional embedding space. As a result, each word vector has a length equal to the number of documents

Table 5.1: Feature comparison between vector space models ©2018 IEEE.

Method	Includes temporal dimension	Smooth evolution of a single vector	Captures drifts in long periods	Captures drifts in short periods	Results are low-dimensional
word2vec [89]					✓
Dynamic Bernoulli embeddings [111]	✓		✓		✓
tf-idf [122]				✓	
Chapter 4 approach	✓	✓	✓	✓	
Chapter 5 approach	✓	✓	✓	✓	✓

in the corpus. In this chapter, we introduce a neural-network-based framework (Fig. 5.2) which generates temporal word embeddings while optimizing for multiple key objectives. The temporal tf-idf representation obtained using Eq. 4.1 is used to obtain the **expected cosine distance** between pairs of word vectors, which is the output layer of our neural network.

The experimental results show that the proposed method performs significantly better than a state-of-the-art dynamic embedding model in capturing short-term changes in word semantics. Results show that our approach improves the continuity between the vectors across different timestamps. As a result, embeddings for different timestamps combine to a homogeneous space, unlike the state-of-the-art models.

5.2 Problem Description

In this work, we focus on timestamped text corpora, such as collections of news articles or scientific publications. Let $\mathcal{D} = \{d_1, d_2, \ldots, d_{|\mathcal{D}|}\}$ be a corpus of $|\mathcal{D}|$ text documents and $\mathcal{W} = \{w_1, w_2, \ldots, w_{|\mathcal{W}|}\}$ be the set of $|\mathcal{W}|$ noun phrases and entities extracted from the

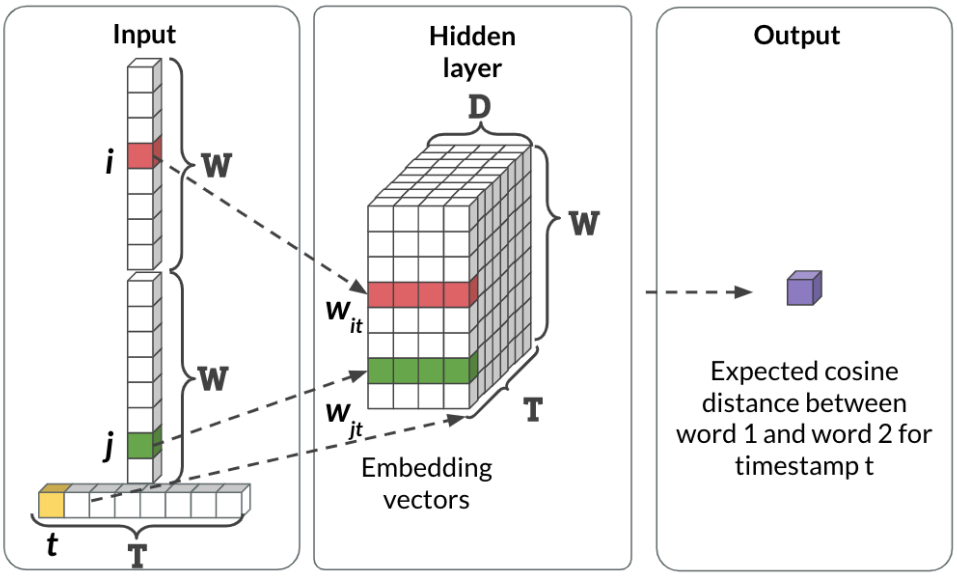


Figure 5.2: Neural network architecture for temporal embedding generation.

text corpus $\mathcal{D}$. We consider each of the noun phrases and entities a word. Each document d contains words from the vocabulary ($\mathcal{W}_d \subset \mathcal{W}$) in the same order as they appear in the original document of d. Every document $d \in \mathcal{D}$ is labeled with a timestamp $t_d \in \mathcal{T}$, where $\mathcal{T}$ is the ordered set of timestamps.

The goal of this chapter is to obtain a temporal word embedding model $\mathcal{U}$ from corpus $\mathcal{D}$. Thus, for every timestamp $t \in \mathcal{T}$, we seek to obtain a vector representation u_{it} for every word $w_i \in \mathcal{W}$. The word embeddings $\mathcal{U}$ are represented as a 3-dimensional matrix of size $|\mathcal{W}| \times |\mathcal{T}| \times |u|$ where $|u|$ is a user-given parameter that indicates the size of a vector for a particular word at a particular time. We use the shorthand U_i to describe the 2-dimensional matrix of size $|\mathcal{T}| \times |u|$ that represents word $w_i \in \mathcal{W}$ over time.

5.3 Methodology

5.3.1 $\mathcal{D}$-dimensional temporal text representation (the baseline model)

Initially, we run the framework presented in Chapter 4 to obtain time-reflective text representations of size $|\mathcal{W}| \times |\mathcal{T}| \times |\mathcal{D}|$. The vectors are formed using the temporal tf-idf weight of a word for every document in every timestamp. The temporal tf-idf weight of a word is computed using Equation 4.1.

Next, we compute the cosine distance between every pair of words and store these as a distance matrix Δ, where each element can be addressed as $\delta_{ijt} \in \Delta$, which is the baseline cosine distance between a particular pair of words $(w_i, w_j) \in \mathcal{W}$ at time $t \in \mathcal{T}$. We also use δ_{ij} to represent a vector of size $|\mathcal{T}|$ with the baseline cosine distance between $(w_i, w_j) \in \mathcal{W}$ for all the timestamps.

5.3.2 Optimizing for similarity: Creation of low-dimensional embeddings that mimic the neighborhood of a high dimensional space

One of our objectives is to obtain a low-dimensional word embedding model $\mathcal{U}$ such that computing the cosine distance between the word vectors results in a distance matrix that closely resembles Δ. Equation 5.1 formulates this objective as ϑ. In this case, we are optimizing the vectors in $\mathcal{U}$ to minimize the difference between the cosine distance of each pair of word vectors for every timestamp and the baseline cosine distance in Δ, the distance matrix formed by vectors resulting from Equation 4.1.

In this chapter, the term $dist(A, B)$ refers to the cosine distance between vector A and vector B. The cosine distance between two word vectors is bounded between $[0, 1]$. A cosine distance of 0 between two word vectors means that both words share the same context, while a cosine distance of 1 means that the vectors are completely orthogonal. The variable

α is introduced as a scaling factor to avoid numerical stability issues with values close to zero. The simplest form of our objective function is as follows.

$$\vartheta_{1a}(\mathcal{U}) = \sum_{i=1}^{|\mathcal{W}|} \sum_{j=1}^{|\mathcal{W}|} \sum_{t=1}^{|\mathcal{T}|} |\alpha \cdot dist(u_{it}, u_{jt}) - \alpha \cdot \delta_{ijt}| \tag{5.1}$$

or

$$\vartheta_{1b}(\mathcal{U}) = \sum_{i=1}^{|\mathcal{W}|} \sum_{j=1}^{|\mathcal{W}|} \sum_{t=1}^{|\mathcal{T}|} (\alpha \cdot dist(u_{it}, u_{jt}) - \alpha \cdot \delta_{ijt})^2 \tag{5.2}$$

5.3.3 Weighing relevance: Giving more importance to the neighborhood of each word

In our work, we focus on the task of studying the semantic evolution of a word based on changes to its context. Thus, it is more important that our word embedding model correctly captures the *relevant* neighborhood of a word. Our experiments demonstrated that each word has a small number of *relevant* neighbors. That is, each word shares context with a small number of words. To take this into account in the objective function, we introduce a penalty when the original cosine distance δ_{ijt} in the baseline model is small, ensuring that our word embedding model captures the *relevant* context as accurately as possible.

$$\vartheta_{2a}(\mathcal{U}) = \sum_{i=1}^{|\mathcal{W}|} \sum_{j=1}^{|\mathcal{W}|} \sum_{t=1}^{|\mathcal{T}|} \frac{(\alpha \cdot dist(u_{it}, u_{jt}) - \alpha \cdot \delta_{ijt})^2}{\delta_{ijt}} \tag{5.3}$$

This formulation has the issue of a potential division by zero, so to prevent this, we explored different alternatives:

$$\vartheta_{2b}(\mathcal{U}) = \sum_{i=1}^{|\mathcal{W}|} \sum_{j=1}^{|\mathcal{W}|} \sum_{t=1}^{|\mathcal{T}|} \frac{(\alpha \cdot dist(u_{it}, u_{jt}) - \alpha \cdot (\delta_{ijt} + \epsilon))^2}{\delta_{ijt} + \epsilon} \tag{5.4}$$

$$\vartheta_{2c}(\mathcal{U}) = \sum_{i=1}^{|\mathcal{W}|} \sum_{j=1}^{|\mathcal{W}|} \sum_{t=1}^{|\mathcal{T}|} (\alpha \cdot dist(u_{it}, u_{jt}) - \alpha \cdot \delta_{ijt})^2 \cdot e^{-\beta \delta_{ijt}} \tag{5.5}$$

$$\vartheta_{2d}(\mathcal{U}) = \sum_{i=1}^{|\mathcal{W}} | \sum_{j=1}^{|\mathcal{W}|} \sum_{t=1}^{|\mathcal{T}|} \log\left((\alpha \cdot dist(u_{it}, u_{jt}) - \alpha \cdot \delta_{ijt})^2 + 1\right) - \log\left(\delta_{ijt} + 1\right) \tag{5.6}$$

where β is a scaling parameter to increase/decrease the importance given to the samples with a smaller distance. Notice that $e^{-\beta\delta_{ijt}}$ in Equation 5.5 imposes a higher penalty when the baseline distance is small. The penalty is less when the baseline distance is large. This equation supports the phenomenon that, for a specific word, most of the words in the vocabulary are at a relatively large distance. The large distances need not be a part of the penalty because the objective function is only concerned about neighbors that appear in the vicinity for the baseline model.

5.3.4 Temporal diffusion filter

Diffusion theory [7] refers to the change of the distributional patterns of a phenomenon over time. Based on this theory, we assume that the meaning of a word, and consequently its vector representation, diffuses (or drifts) over time. Thus, the word embeddings should evolve smoothly over time.

To introduce this concept in our objective function, we assume that every word vector affects every timestamp to some degree. We use a Gaussian filter (Equation 5.7) to *diffuse* the contribution of each vector smoothly before and after the timestamp for the current sample. The filter uses a sliding window, going from the first to the last timestamp. σ is a user-settable parameter representing the standard deviation of the Gaussian distribution. A large value of σ means that the diffusion of word vectors is slow over time. A small standard deviation allows capturing short-term changes in meaning but makes the model more susceptible to noise.

$$\gamma(t, \sigma) = \left\langle \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t_i - t)^2}{2\sigma^2}} \right) \text{with } t_i = 1, \ldots, |\mathcal{T}| \right\rangle \tag{5.7}$$

Equation 5.8 presents the updated objective ϑ which includes the temporal diffusion of the word embeddings. It is worth noting that in this particular equation, the *expected*

cosine distances are also weighted using the γ function. This is only required if we compute the cosine distance of the vectors obtained using the vanilla tf-idf method.

$$\vartheta_{3a}(\mathcal{U}) = \sum_{i=1}^{|\mathcal{W}|}\sum_{j=1}^{|\mathcal{W}|}\sum_{t=1}^{|\mathcal{T}|}(\alpha \cdot dist(\gamma(t,\sigma)U_i, \gamma(t,\sigma)U_j) - \alpha \cdot \gamma(t,\sigma) \cdot \delta_{ij})^2 \cdot e^{-\gamma(t,\sigma)\cdot\delta_{ij}} \tag{5.8}$$

It would be redundant to use the temporal diffusion filter when using the temporal tf-idf method for training since these distances are already filtered. Thus, we use Equation 5.9 which does not include the temporal diffusion in the cosine distance terms.

$$\vartheta_{3b}(\mathcal{U}) = \sum_{i=1}^{|\mathcal{W}|}\sum_{j=1}^{|\mathcal{W}|}\sum_{t=1}^{|\mathcal{T}|}(\alpha \cdot dist(\gamma(t,\sigma)U_i, \gamma(t,\sigma)U_j) - \alpha \cdot \delta_{ijt})^2 \cdot e^{-\beta\delta_{ijt}} \tag{5.9}$$

5.3.5 Smoothness penalty: Creating a homogeneous temporal embedding space

The second important goal that our word embedding model should achieve is to be spatially smooth over time. Continuous or smooth temporal embeddings are those where the distance (e.g., Manhattan or Euclidean) between two vectors of the same word for consecutive timestamps is small. Equation 5.10 captures the expected behavior by penalizing significant spatial changes.

$$\varepsilon_{1a}(\mathcal{U}) = \sum_{i=1}^{|\mathcal{W}|}\sum_{t=1}^{|\mathcal{T}|-1}||u_{it+1}, u_{it}||_2 \tag{5.10}$$

The main issue with this expression is that by forcing consecutive vectors to be very close together, we might be losing important information when the vectors drift apart in the original data. Thus, we introduce weights, ω_ϑ, and ω_ε to control the effect of each objective. The final objective function takes the form of Equation 5.11.

$$\mathcal{F}_a(\mathcal{U}) = \vartheta_3(\mathcal{U})^{\omega_\vartheta}\varepsilon_1(\mathcal{U})^{\omega_\varepsilon} \tag{5.11}$$

An alternative form would be:

$$\mathcal{F}_b(\mathcal{U}) = \omega_\vartheta \log \vartheta_3(\mathcal{U}) + \omega_\varepsilon \log \varepsilon_1(\mathcal{U}) \tag{5.12}$$

or

$$\mathcal{F}_c(\mathcal{U}) = \omega_\vartheta \vartheta_3(\mathcal{U}) + \omega_\varepsilon \varepsilon_1(\mathcal{U}) \tag{5.13}$$

5.3.6 Implementation

We implemented a neural network-based model using Tensorflow to generate our low-dimensional temporal embeddings. An overall view of the architecture of our neural network is shown in Fig. 5.2. The goal of the neural network is to minimize Eq. 5.11. We initialize the weights randomly. The data used for training the model contains three inputs and one target value. The inputs are the indices for two random words w_{it} and w_{jt}, at timestamp t. The target value is the expected cosine distance between w_{it} and w_{jt}, obtained using the temporal tf-idf representations of Equation 4.1.

5.4 Experimental Results

In this section, we present the experiments performed using four different datasets. For the first experiment, we generated **a synthetic dataset** that consists of 10,000 words and ten timestamps. The actual neighborhoods are known up to a size of $k = 10$ for every timestamp. Neighborhoods of larger sizes contain random words starting at the 11th nearest neighbor. We use this dataset to: (1) evaluate the different versions of our objective function and (2) as a sanity check for the generated models.

For the rest of the experiments we used: (1) a corpus of **613,949 PubMed cancer-related abstracts** [16] published in a 21 year span from which we extracted a vocabulary of 3,000 words based on frequency; (2) a corpus of **812,857 New York Times articles** published in a 29 year period from which we extracted the top-5,000 words based on frequency; and (3) a corpus of **101,273 National Vulnerability Database (NVD)**

bulletins [21] published in a 20 year period from which we extracted a vocabulary of 3,000 words based on frequency.

We evaluate our temporal embedding method by comparing its performance with that of a regular tf-idf model, the temporal tf-idf model presented in Chapter 4 and the dynamic embeddings introduced by [111]. For the baseline methods, we use the same parameter values given in Section 4.5. We empirically selected the following values for the hyper-parameters of our model: batch size $= 16,384$, $|u| = 64$, learning rate $= 0.05$, $\alpha = 1.0$, $\sigma = 0.25$, $\beta = 0.25$, $\omega_\vartheta = 1.0$ and $\omega_\varepsilon = 0.01$.

In this section, we seek to answer the following questions.

1. What is the effect of introducing different penalty terms in our objective function? (Section 5.4.1)

2. How well does our algorithm track the temporal tf-idf representation presented in Chapter 4? (Section 5.4.2)

3. What effect does the neighborhood size have on our model generation? (Section 5.4.3)

4. How sensitive is our temporal word embedding model to changes in the hyperparameters? (Section 5.4.4)

5. How does our temporal embedding model compare qualitatively to other methods in terms of tracking semantic evolution? (Section 5.4.5)

6. How well does our algorithm track the evolution of a specific term? (Section 5.4.6)

5.4.1 Effect of penalty terms

In this experiment, we study the effect of the different versions of our objective function on the quality of the temporal word embedding model, focusing on the task of tracking semantic evolution. The versions under study correspond to Eq. 5.2 (ϑ_{1b}), Eq. 5.6 (ϑ_{2c}), Eq. 5.9 (ϑ_{3b}), Eq. 5.11 ($\mathcal{F}_a$), Eq. 5.12 ($\mathcal{F}_b$), and Eq. 5.13 ($\mathcal{F}_c$). We quantify the quality of

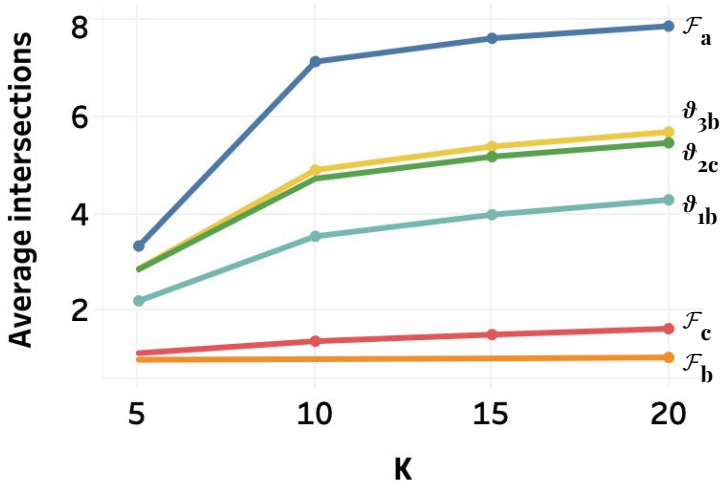


Figure 5.3: Average number of intersections per timestamp for different neighborhood sizes (k) between the neighborhoods obtained with the baseline method and those obtained using the different versions of our objective function (embedding size = 64).

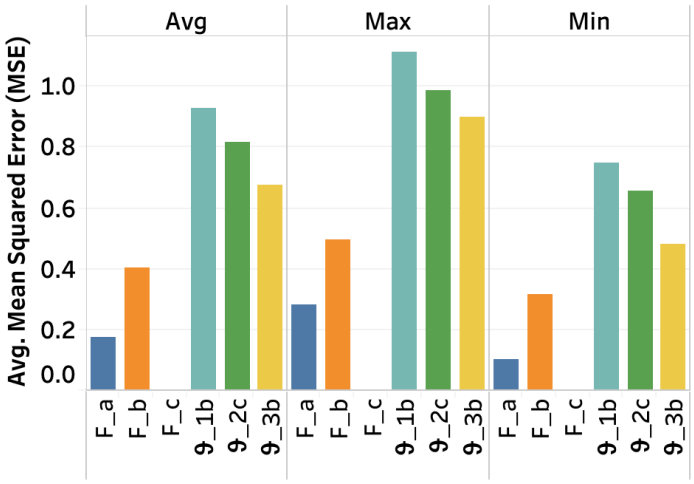


Figure 5.4: Average mean squared error (MSE) for different versions of our objective function. The average MSE is computed from obtaining the squared difference between vectors for the same word for every pair of consecutive timestamps (embedding size = 64).

the resulting vectors with two different metrics: similarity and continuity. The similarity is measured as the number of intersections between the word neighborhoods obtained using the baseline and each of the different versions of our objective function. The continuity is measured using the average, maximum and minimum mean squared errors (MSE) across consecutive timestamps for the word vectors obtained using the different versions of our objective function.

The goal of the similarity evaluation is to quantify how well our model mimics the original temporal tf-idf model. It must be noted that we did not expect to have a perfect match in the neighborhoods of words since the original representation does not take into account latent relationships between words. Figure 5.3 shows the results for the similarity evaluation with the synthetic dataset described at the beginning of this section. The objective function labeled as $\mathcal{F}_a$ performs significantly better than the other formulations. If we discard $\mathcal{F}_b$ and $\mathcal{F}_c$, it is possible to see how the similarity improves with the progression in which we developed our objective function. Furthermore, taking into account that only the top-10 neighbors are known, having an average of 8 neighbor intersections means that our model can correctly capture the semantic evolution of this synthetic dataset.

Evaluating continuity is required to ensure that there is a smooth transition between timestamps for the vectors of the same word. A high average or minimum MSE value indicates that there is a significant movement of the word vectors over time. However, a small maximum MSE value would mean that the word embeddings are not following the trends observed in the baseline. Thus, the best model is one that has high similarity with the baseline while maintaining a low MSE value. Fig. 5.4 shows the results for the continuity evaluation. In this case, F_c has a continuity of zero, which, in conjunction with the similarity results, indicates that this objective function produces static, unusable vectors. The second smallest average MSE value is obtained with $\mathcal{F}_a$, which also showed the best performance in terms of similarity. Thus, the **final objective function** is $\mathcal{F}_a$ (Eq. 5.11), and we confirm that the smoothness penalty (Eq. 5.10) has a positive effect both on the similarity and continuity results.

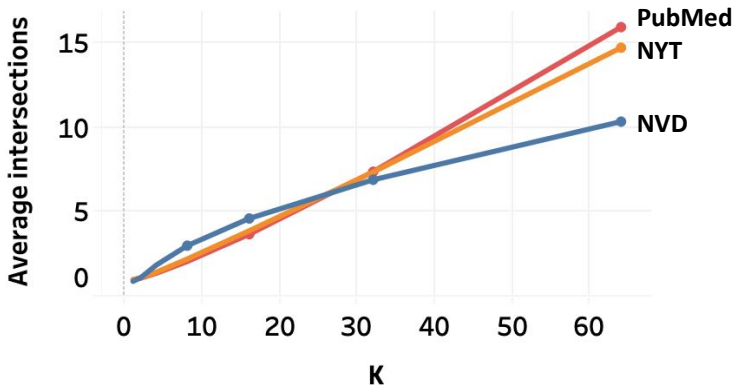


Figure 5.5: Average number of intersections per timestamp between the neighborhoods obtained with temporal tf-idf (Eq. 4.1) and the temporal embedding method (Eq. 5.11) for the NVD, PubMed and New York Times datasets (embedding size = 64).

5.4.2 Neighborhood similarity using real-world datasets

The previous experiment shows promising results in terms of tracking the semantic evolution of a synthetic dataset. However, it is important to evaluate if the neighborhoods obtained using our temporal word embedding model resemble those obtained using the baseline temporal tf-idf model presented in Chapter 4. Figure 5.5 presents the average number of word intersections between the neighborhoods of size k obtained using the baseline model and those obtained using the proposed model for the three real datasets described previously. The nearest neighbors, which form the neighborhood, are those with the smallest cosine distance between their word vectors and the base word vector. As can be seen from the plot, the results show a decline in performance when compared to the synthetic dataset. On average, we need to look at a neighborhood of size 20 to get the top-5 neighbors from the baseline using real-world datasets.

The reason for this is the following – with the synthetic dataset, we made sure that there are ten nearest neighbors for each word. With real-world datasets, most of the words have a small number of words in the neighborhood. In our experience, most of the words have two, three, or four neighbors that are closeby and share a context. The rest of the neighbors

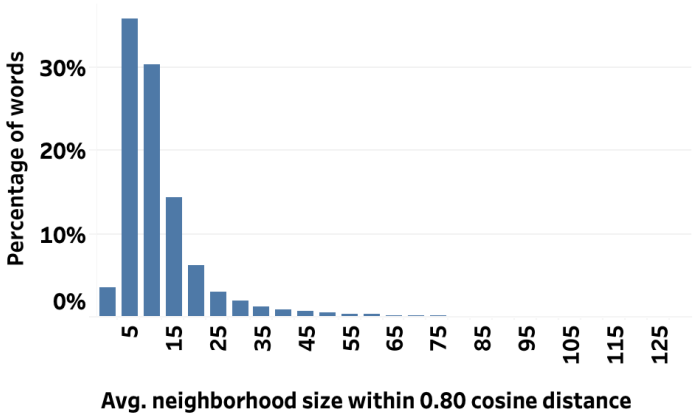


Figure 5.6: Percentage of words in the NVD dataset that have X number of neighbors within a cosine distance threshold of 0.80.

are too distant for a meaningful comparison. Therefore, finding the top-5 neighbors from the baseline in the top-20 neighbors using real-world datasets is a significantly good result because, in reality, the average number of meaningful nearest neighbors is around 3, in our experience.

5.4.3 Effect of neighborhood sizes

In this section, we analyze the temporal tf-idf (baseline) representation for the NVD dataset. In particular, we look at the distribution of the number of neighboring words that are within a given distance threshold of the base word. Figure 5.6 presents the results for a cosine distance threshold of 0.80. The graph shows that 40% of the words have, on average, less than ten neighboring words that are under the 0.80 threshold, and 85% of the words have less than 20 neighbors under the same threshold. We observed a similar distribution in the other datasets.

The number of contextual words is minimal in natural language, especially when the language model is developed using a corpus, not a thesaurus. If we evaluate a neighborhood using 10-nearest neighbors, but, on average, each word has less than five nearest neighbors

in their vicinity, then a performance metric will give a low accuracy in evaluating the outcome because less than five of the neighbors out of 10 will be correct in most cases. To avoid this situation, we propose using a density-based dynamic neighborhood size (K) for every word at every timestamp. The dynamic neighborhood size is given by the number of nearest neighbors i for which the distance from the origin to the i-th neighbor is larger than double the average distance of the previous neighbors (1 through $i-1$).

Figure 5.7 shows the effect of using the dynamic values of K versus using a static version in terms of the average intersections between the neighborhoods computed using our method and those generated using the baseline model. In the dynamic K version, we only use the words that have at least the given value of K at every point in the x-axis. That is, if a word has only two nearest neighbors that fulfill the dynamic K condition at every timestamp, then this word is not taken into account when computing the average intersections for a dynamic neighborhood size of 4. The plot shows that there is a significant improvement when using the dynamic K values, in particular for the PubMed dataset, where the number of average intersections almost doubles for $K = 8$.

Tables 5.2, 5.3, and 5.4 contain the evolution of the neighborhoods of three words from the NVD dataset obtained using our temporal word embedding model with dynamic K values. Table 5.2 illustrates the neighborhood of the term *fulfillment component*. The table shows that there is only one *relevant* neighbor per timestamp, which means that there is a strong connection between these terms, probably because they refer to a particular vulnerability related to Oracle. However, Tables 5.3 and 5.4 present the neighborhoods of the terms *itunes* and *winzip*, which are more general and thus include more neighbors per timestamp.

Appendices A, B, and C contain more samples of the resulting neighborhoods using the dynamic K method with the PubMed, NVD, and New York Times datasets, respectively. These tables show that every word at every timestamp can have a different number of *relevant* neighbors based on the density of its neighborhood, which seems to be correlated with how specific a term is.

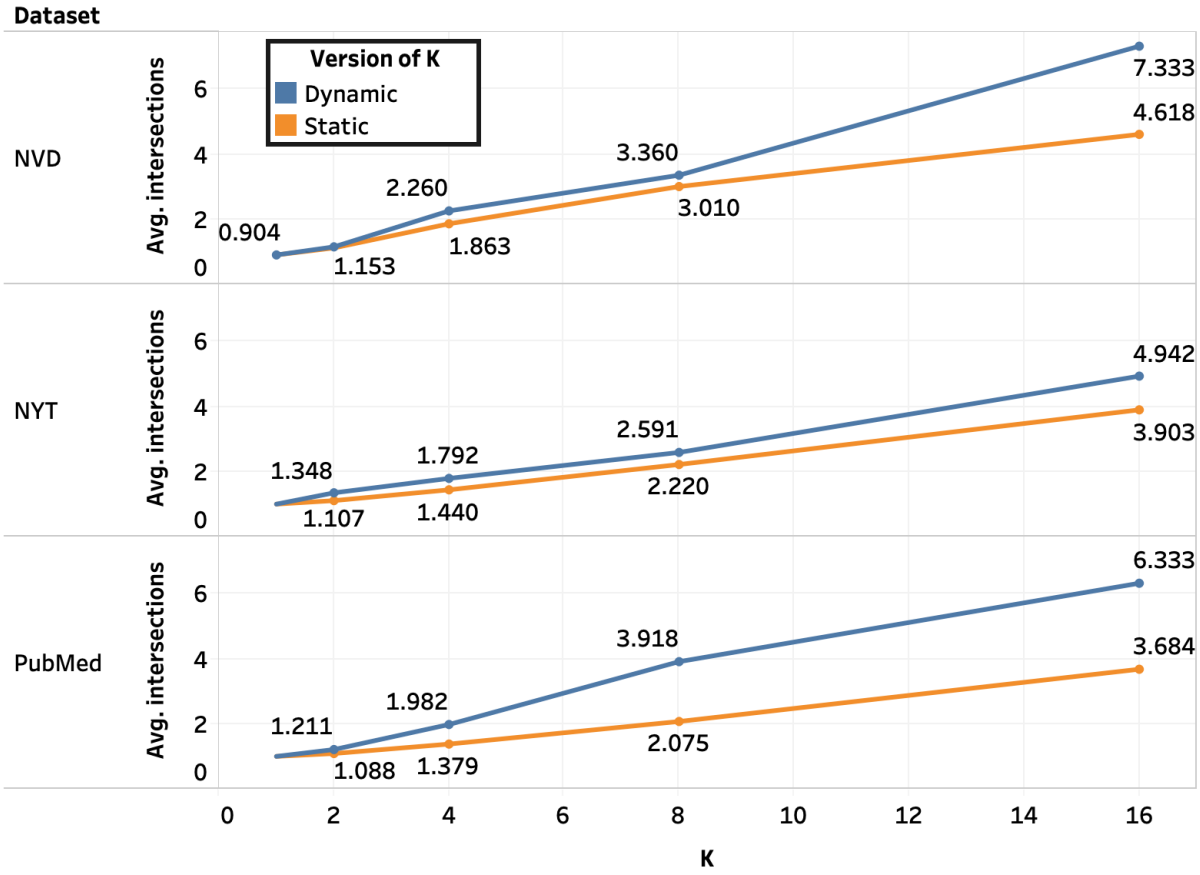


Figure 5.7: Average number of intersections per timestamp between the neighborhoods obtained with temporal tf-idf (Eq. 4.1) and the temporal embedding method (Eq. 5.11) for the NVD, PubMed and New York Times datasets (embedding size = 64), using static and dynamic values of K.

Table 5.2: Nearest neighbors of the term *fulfillment component.*.

1999	**2002**	**2005**	**2008**	**2011**	**2014**	**2017**
oracle	oracle	oracle	oracle e business suite	oracle e business suite sub-component user interface	oracle e business suite	oracle e business suite sub-component user interface

Table 5.3: Nearest neighbors of the term *itunes*..

1999	**2002**	**2005**	**2008**	**2011**	**2014**	**2017**
unquoted windows search path vulnerability	unquoted windows search path vulnerability	trojan horse dll	apple quicktime	apple	webkit component	safari
libpng	insufficient filtration	unquoted windows search path vulnerability		iphone ipod touch	safari	
login.asp	user supplied data	crafted serialized java object		webkit		
kernel component	multiple heap based buffer	attempt		coregraphics		
command line arguments	replacement	unspecified denial				
.html cwe untrusted search path	dll	environment				

Table 5.4: Nearest neighbors of the term *winzip*.

1999	**2002**	**2005**	**2008**	**2011**	**2014**	**2017**
powerzip	powerzip	powerzip	powerzip	authenticity	powerzip	updates
bitzipper	bitzipper	bitzipper	case	updates	authenticity	powerzip
winrar	winrar	winrar	components			
	moinmoin		contacts			
	drivers		cce			
			application			
			updates			
			authenticity			
			privilege boundaries			

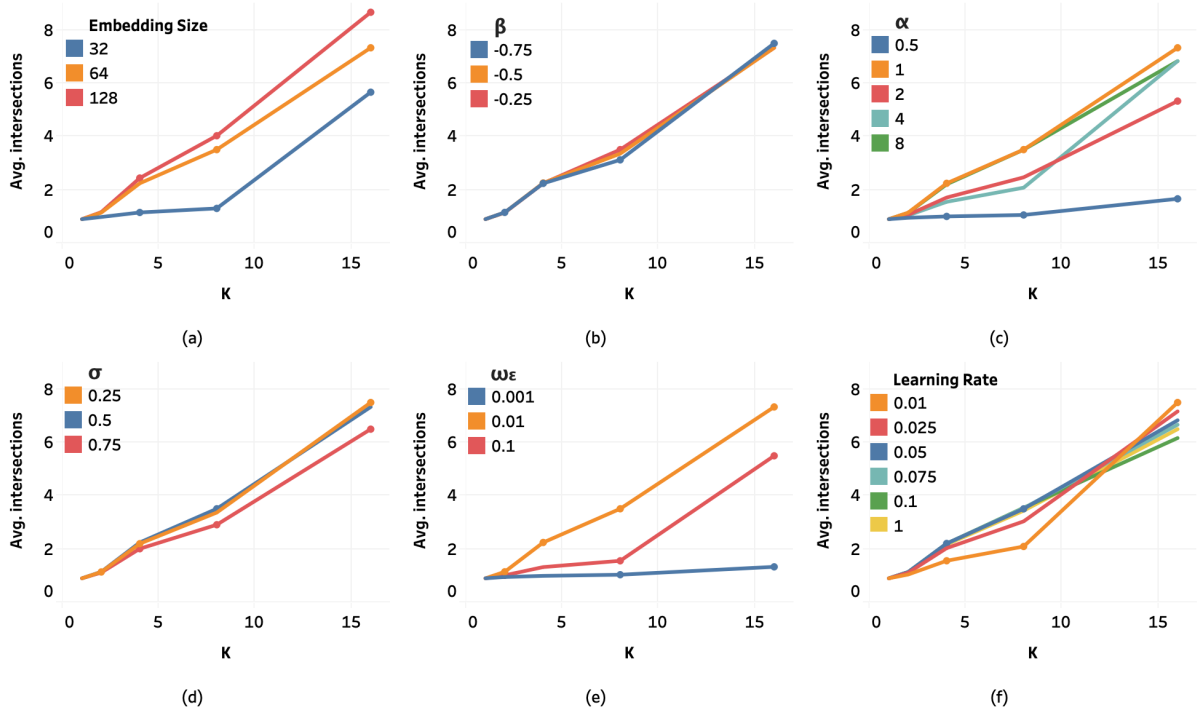


Figure 5.8: Average number of intersections per timestamp between the neighborhoods obtained with temporal tf-idf (Eq. 4.1) and the temporal embedding method (Eq. 5.11) for the NVD dataset while changing (a)embedding size, (b)β, (c)α, (d)σ, (e)ω_ϵ, and (f) learning rate.

5.4.4 Sensitivity analysis

In this experiment, we evaluate the effect of performing a sweep of different values for (a) the embedding size, (b) the exponential factor β, (c) the scale factor α, (d) the temporal diffusion filter standard deviation σ, (e) the smoothness penalty factor ω_ϵ, and (f) the learning rate of our model. We use the average number of intersections per timestamp between the neighborhoods obtained using our method and those generated using the baseline method as our accuracy metric.

Figure 5.8 presents the effect of changing the parameters of interest on the accuracy of our technique. The results show that the embedding size, the scale factor (α), and the smoothness penalty factor (ω_ϵ) have a significant effect on the accuracy of our neigh-

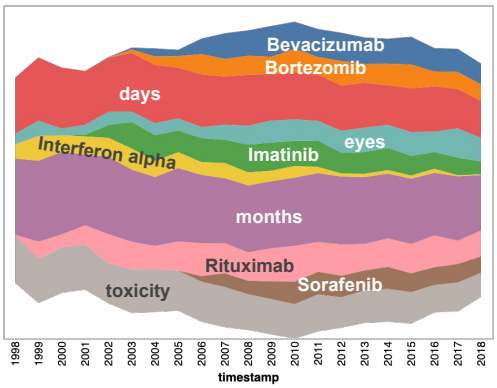
borhoods. In this plot, we only present the results obtained using the NVD dataset, but we performed similar analyses for the other datasets, focusing only on the variables that showed higher sensitivity. It is important to note that Fig. 5.8(a) shows that using an embedding size of 128 resulted in slightly better performance than a size of 64. However, this slight improvement does not justify the significant increase in terms of training time and computational/storage complexity (training time increases from 18 hours to 50 hours per experiment in a GPU-enabled cluster), so we decided to keep an embedding size of 64.

5.4.5 Qualitative evaluation

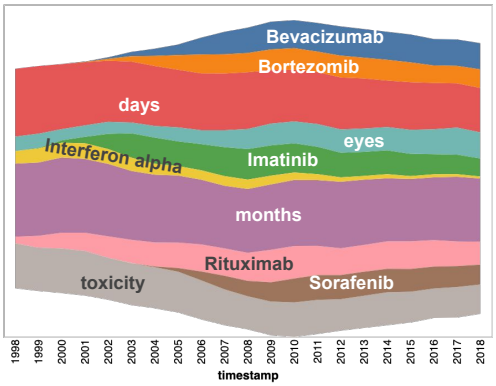
In this section, we illustrate the advantages of using our temporal word embedding model for the task of tracking the semantic evolution of the PubMed dataset. The neighborhoods are obtained by computing the cosine distance between the base word and the rest of the vocabulary. For this experiment, we selected the top-10 words such that their regular tf-idf-based vectors have the highest cosine similarity at any point in time with the tf-idf vector for the word of interest. We compare, over time, the neighborhoods obtained using our temporal embedding method with those obtained using a (vanilla) tf-idf model, the temporal tf-idf model presented in Chapter 4, and one of the dynamic embedding models [111].

Fig. 5.9 presents the evolution of the neighborhood of the word *treatment* at different points in time, for (a) the tf-idf method, (b) the baseline temporal tf-idf method (Eq. 4.1), (c) our temporal embedding method (Eq. 5.11) using a static K, (d) our temporal embedding method (Eq. 5.11) using a dynamic K, and (e) the dynamic Bernoulli embedding method [111]. The height of each stream represents the cosine similarity (i.e., 1.0-cosine distance) between the word vector for *treatment* and the vector of the stream label.

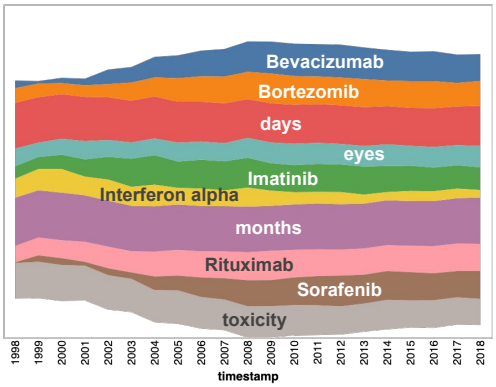
Fig. 5.9 (a) shows that using only tf-idf results in a noisy signal. The dynamic embedding method, shown in Fig. 5.9 (e), does not follow any of the trends exhibited by the tf-idf model of Fig. 5.9 (a), producing mostly static streams. However, the temporal tf-idf model of Fig. 5.9 (b) and our temporal embedding models of Figs. 5.9 (c) and (d) can capture the



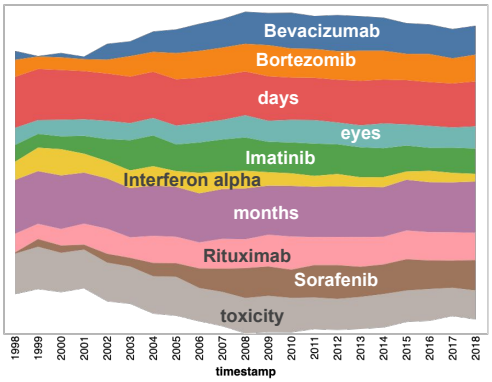
(a) Tf-idf (vector size = 3,000)



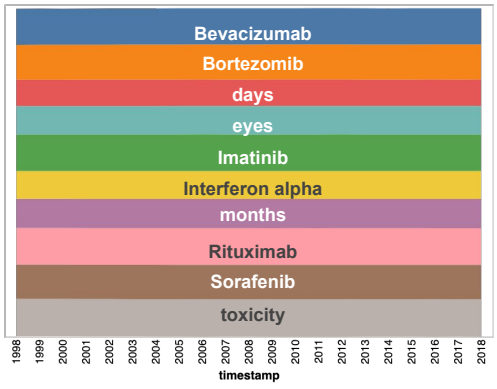
(b) Temporal tf-idf baseline model (vector size = 3,000)



(c) Our model (Equation 5.11, vector size = 64, with static K values)



(d) Our model (Equation 5.11, vector size = 64, with dynamic K values)



(e) Dynamic Bernoulli embeddings

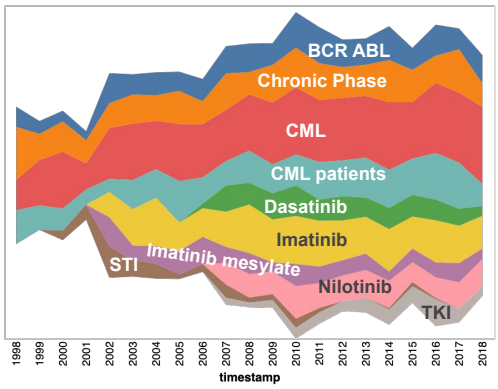
Figure 5.9: Evolution of *treatment* using different vector representations.

most significant trends in the tf-idf model of Fig. 5.9 (a). There are no significant differences between the streamgraphs obtained using a static K (Fig. 5.9 (c)) and a dynamic K (Fig. 5.9 (d)), which is expected. At the same time, the temporal tf-idf model and our temporal embedding model filter some of the noise of the vanilla tf-idf model. It is important to remember that our model has a significantly smaller word vector size of 64 versus 3,000 from the temporal tf-idf model.

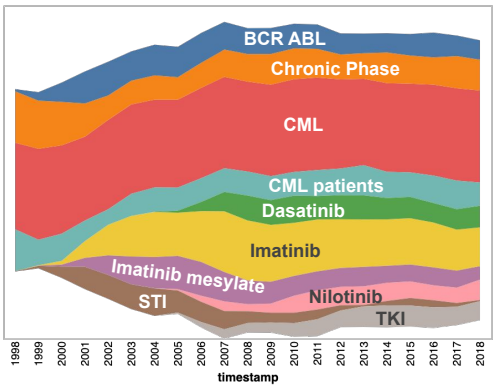
Fig. 5.10 presents the evolution of the neighborhood of the term *chronic myeloid leukemia (CML)* at different points in time. Fig. 5.10 (a) shows once again that using only tf-idf results in a noisy signal. The dynamic embedding method, shown in Fig. 5.10 (e), results in completely static neighborhoods. Again, our temporal embedding models (Figs. 5.10 (c) and (d)) mimic the behavior of the baseline model (Fig. 5.10 (b)). Of special interest is the appearance of the term *dasatinib* around 2005. This drug was introduced in 2004/2005 as an alternative to *imanitib*, the *gold standard* in CML treatment, because some patients presented *imatinib* resistance [49]. A similar trend is observed with the term *TKI* (*tyrosine kinase therapy*), which was introduced in the literature in 2007 [26, 107]. In this case, Figs. 5.10 (c) and (d) show that, using our temporal embeddings, the term *TKI* appears slightly earlier than 2007, but this is expected because of the diffusion process followed by our method.

5.4.6 Case study

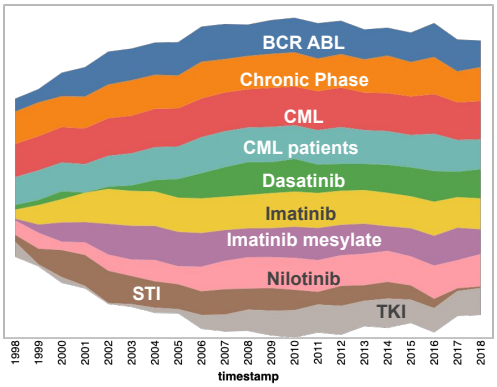
In this experiment, we analyze the changes in the neighborhood of the term *president* from the New York Times dataset. Figure 5.1 (in the Introduction Section) presents how the L2 norm and the nearest neighbors of the vectors for the word *president* change over time, with call-outs shown for every election year starting in 1992. The resulting neighborhoods seem very accurate because they always include the seating President for that particular year, along with the presidential candidates, except for 1996, when there was a three-party election in which Ross Perot was also a candidate. It is important to note that the terms that are shown in Figure 5.1 are only the entities representing persons from the



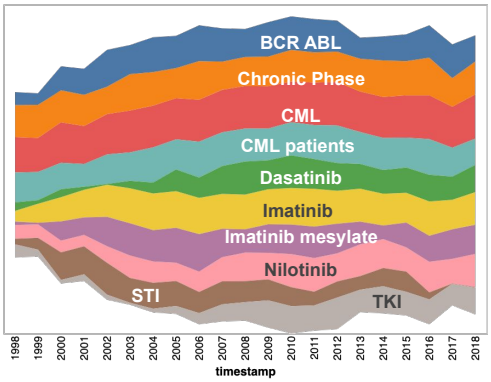
(a) Tf-idf (vector size = 3,000)



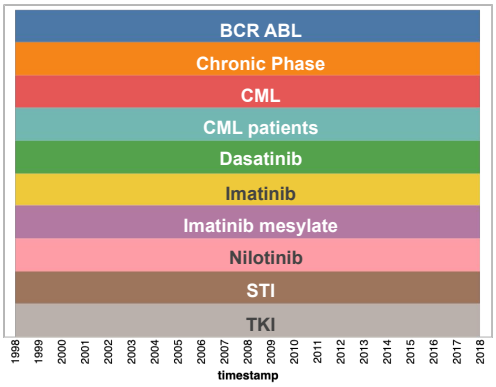
(b) Temporal tf-idf baseline model (vector size = 3,000)



(c) (Equation 5.11, vector size = 64, with static K values)



(d) (Equation 5.11, vector size = 64, with dynamic K values)



(e) Dynamic Bernoulli embeddings

Figure 5.10: Evolution of *chronic myeloid leukemia* using different vector representations.

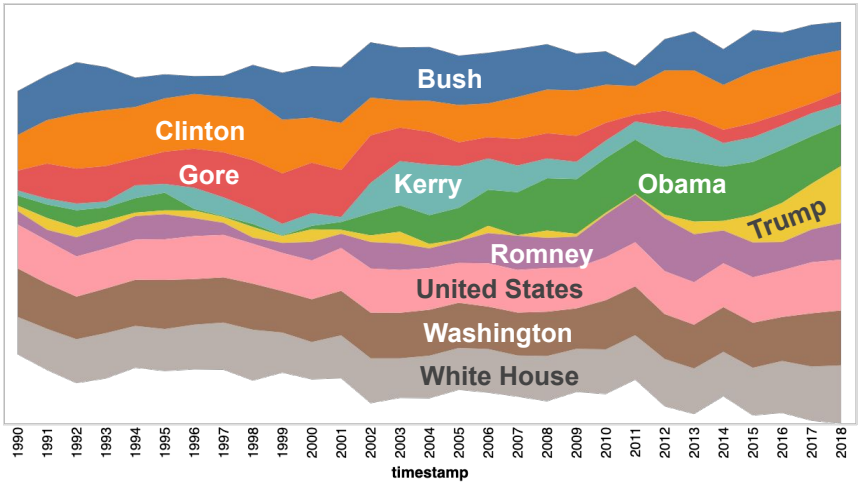


Figure 5.11: Streamgraph of the word *president* using the New York Times corpus (embedding size = 64).

top-10 neighbors for each year. Several contextual entities such as White House, Chief Executive, Vice President, Washington, democrats, and republicans also appear in these top-10 neighbors.

Figure 5.11 shows a streamgraph for the same term (*president*). Using this plot, we can see how the words *Obama* and *Trump* started getting closer to the word *president* only a few years before their presidency. We also can observe the particular cases of political families, such as the Bush and Clinton families that have been relevant to the presidency and the presidential elections for the last three decades.

We also analyze the changes in the neighborhood of the term *arbitrary code* from the National Vulnerability Database (NVD) dataset. Figure 5.12 presents how the L2 norm and the nearest neighbors of the vectors for *arbitrary code* change over time, with call-outs shown every couple of years. Figure 5.13 shows the normalized number of co-occurrences between *arbitrary code* and a selection of the nearest neighbors presented in Fig. 5.12. We obtained these statistics using the advanced search function from the NVD website [21].

When we compare both Figs. 5.12 and 5.13, it is possible to see a high co-occurrence between *arbitrary code* and *buffer* up to the year 2012, when it starts declining, which

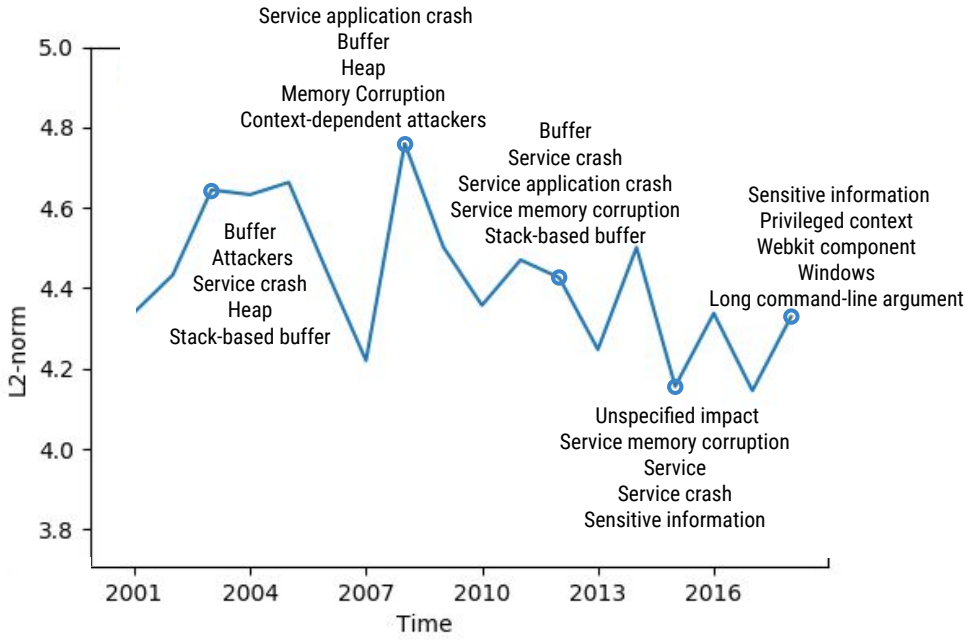


Figure 5.12: Evolution of the neighborhood of the term *arbitrary code* in the National Vulnerability Database using our temporal embedding method (embedding size = 64).

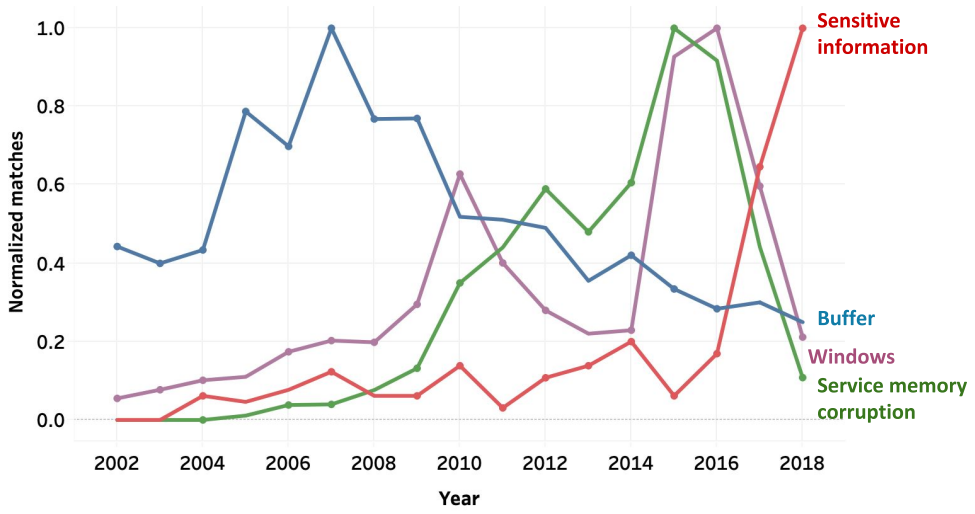


Figure 5.13: Normalized frequency of co-occurrences of the term *arbitrary code* and different terms in the National Vulnerability Database.

correlates with the word *buffer* disappearing from the nearest neighbors in 2015. A similar correlation can be observed with *service memory corruption*, *Windows* and *sensitive information* in terms of the spikes in co-occurrence and the appearance in the nearest neighbors of *arbitrary code*. This indicates that our method can capture the trends observed in the original data correctly.

5.5 Conclusions

This chapter introduced a new technique to generate low-dimensional temporal word embeddings for text corpora. We compared our model with other existing text representations through extensive experiments. Our method obtains a representation that: (1) can track significant changes that are observed within a short period, (2) provides a smooth evolution of the vectors over a continuous temporal vector space, (3) uses the concept of *diffusion* to compensate for the possible sparsity of a dataset, and (4) is low-dimensional, using an inner dimension of 64 for all of the experimental results, compared with 3,000 or 5,000 for the different datasets.

Unlike previous models, our proposed model creates a homogeneous space over every timestamp of the embeddings. As a result, the generated vectors over timestamps can be used for prediction using conventional algorithms for predicting signals. In the next chapter, we leverage the proposed representation to predict embeddings for upcoming years, and hence automatically identify changes of meanings of words in the vocabulary of a corpus.

Chapter 6

Trend Analytics for Hypothesis Generation

6.1 Introduction

The low-dimensional space-continuous temporal word embeddings introduced in Chapter 5 facilitate performing trend analysis of these vectors by using existing sequential modeling techniques. At a high level, a sequential model takes as input a sequence of elements, e.g., word embeddings for each timestamp, and outputs a single fixed-size vector. The meaning of the output vector can change depending on the application. In this work, we train the models so that the output is the *expected* vector for the next timestamp. The generated models can be leveraged to predict the future trends of all the word vectors in the vocabulary. The predicted patterns can then be used to aid in the detection of latent relationships between pairs of terms, which we can use as the basis for automatic hypothesis generation.

Current hypothesis generation techniques [143, 52] focus on associating multiple concepts, e.g., *nicotinamide* and *cell fate decisions* to study *organismal aging*, by searching for statistically significant connections between these terms. While some hypothesis generation techniques have been successfully used in scientific domains, they require a significant amount of domain knowledge. Another limitation of statistics-based analyses is that only direct connections within the text corpus support the generated scientific hypotheses. The use of a continuous temporal language representation makes it possible to project how the word vector space would be in the future even though the corpus does not contain future

documents. Hypotheses generated from the predicted future embedding space have the potential to be novel hypotheses that traditional association-based approaches might not reveal.

In this chapter, we leverage the predictive power of our low-dimensional diffusion-based temporal word embedding model to generate temporal models of the semantic evolution of words, as well as for automatic hypothesis generation. We achieve this by modeling the trends observed in the temporal embedding vectors of all words in the vocabulary using different sequential modeling algorithms, including different variants of recurrent neural networks as well as the attention-based Transformer model [132]. Once we create and train a model, we can use it to generate future word vectors and find pairs of words that are not related now but appear to have a relationship in the future embedding space. Using this approach, a scientist can hypothesize that this *latent relationship* is real and verify it through experimentation.

There are two different types of hypothesis discovery problems [8]: open and closed. For open discovery, the algorithm starts with a single term A. The main goal is to find a set of terms C that are somehow related to term A, and each of these relationships are defined as a hypothesis. A method used frequently to automatically generate open hypotheses is to find a set of intermediate terms B that appear in both the context of A and C [8]. In contrast, in closed discovery, an initial hypothesis is given, i.e., term A and term C are related to each other. Given this initial hypothesis, closed discovery scans the corpus for evidence that supports the initial hypothesis by searching for intermediate terms B that are related to both term A and term C [8].

We focus on open hypothesis discovery in this chapter. For example, given the word *ischemia*, if we observe over time that its word vector is becoming closer to *IL-1 alpha* and *PARP inhibitors*, which are commonly known to be associated with *Alzheimer's disease*, we may generate the hypothesis that genetic treatments of *Alzheimer's disease* may be applicable in treating *post-burn ischemia* injuries.

The experimental results explained in this chapter show that the proposed sequential

models trained with our temporal word vectors can generate predicted embeddings that can capture semantic changes that have been *observed previously*. The generated models perform particularly well when predicting only embeddings for the next timestamp. Our models do not target the replication of sudden changes that are only observed in the test data since the model accounts only for the trends in the training data. Additionally, the further away the predicted embeddings are from the current timestamp, the lower the expected accuracy of the neighborhoods, which is to be expected in all predictions. Results show that our automatic hypothesis generation approach can identify pairs of words that might have a latent relationship that is not evident in the original text corpus.

6.2 Problem Description

Let $\mathcal{U}$ be a temporal word embedding model generated using the framework presented in Chapter 5 from a timestamped document corpus $\mathcal{D} = \{d_1, d_2, \ldots, d_{|\mathcal{D}|}\}$, with a vocabulary $\mathcal{W} = \{w_1, w_2, \ldots, w_{|\mathcal{W}|}\}$ of size $|\mathcal{W}|$. The word embeddings $\mathcal{U}$ are represented as a 3-dimensional matrix of size $|\mathcal{W}| \times |\mathcal{T}| \times |u|$ where $\mathcal{T}$ represents the ordered set of timestamps when the original corpus was published, and u is a vector for a particular word at a particular time and its size is a user-given parameter. For every timestamp $t \in \mathcal{T}$, we have a vector representation u_{it} for every word $w_i \in \mathcal{W}$. We use the shorthand $\mathbb{U}^t$ to describe the 2-dimensional matrix of size $|\mathcal{W}| \times |u|$ that represents the vectors for timestamp t for all words $\mathcal{W}$.

The goal of this chapter is to generate a sequence model $\mathcal{P}$ that can capture the trends observed in the original embedding model $\mathcal{U}$. More specifically, we want to train a model for which $\mathcal{P}(\mathbb{U}^{t_a:t_b}) \approx \mathbb{U}^{t_{b+1}}$, i.e., a model that takes as input the temporal embedding vectors for every word between t_a and t_b and predicts as output embedding vectors for every word in the vocabulary that are as similar as possible to the vector representations for timestamp t_{b+1}. The output of $\mathcal{P}$ can be used to extrapolate the future trends of the words in the vocabulary. Based on these trends, we can search for pairs of words that have a stronger

relationship and hence appear closer in the predicted embedding space than in the existing one, allowing us to generate hypotheses in connection to the predicted relationship.

6.3 Methodology of Time-Series Modeling

Several researchers have focused on studying multi-dimensional time-series prediction [85, 82, 151, 152] using methods such as linear regression and accounting for temporal effects such as seasonality [82]. However, these models are not well-suited for prediction of high-dimensional non-linear phenomena, which is the case of semantic evolution. To model non-linear phenomena, several neural-network based sequence modeling techniques have been introduced recently [54, 25, 9, 132].

In this chapter, we leverage state-of-the-art techniques in terms of sequence modeling, such as (1) the Long Short Term Memory (LSTM) [54] and (2) Gated Recurrent Units (GRU) [25] recurrent neural networks (RNNs). We also explore the effect of adding an attention mechanism [9] to both LSTM- and GRU-based RNNs, which allows the network to focus on the most important elements of the sequence. Finally, we evaluate the non-recurrent attention-based *Transformer* model [132] which, in contrast to RNNs, can be trained in parallel. In this section, we give a brief overview of the sequential modeling techniques used, but the reader is encouraged to consult the references for a detailed description of each method.

6.3.1 Long Short Term Memory (LSTM) network

The Long Short Term Memory (LSTM) network [54] is a particular form of a recurrent neural network (RNN) that takes as input a sequence of fixed-dimensional vectors and outputs a (hidden) state vector. LSTM networks have three weighted gates: *input*, *forget*, and *output* gates, which control the influence of the current input, output, and previous hidden states on the state of the current cell. The weight of each gate goes from 0 to 1, which is achieved by a sigmoid activation function.

The *input* gate controls how much of the latest word vector should the model take into account. The *forget* gate defines how much information of the old cell states should the LSTM retain. If the previous information is not useful, the training process sets the *forget* gate to 0, which essentially means that the model *forgets* the previous information. Finally, the *output* gate controls how much information passes to the next state or the predicted embeddings.

Figure 6.1 shows the architecture of a single LSTM cell. Each LSTM cell has two inputs and two outputs. The inputs are: (1) the previous cell state C_{t-1}, and (2) a concatenation of the previous hidden state H_{t-1} and the current word embedding w_t. The output layer of the LSTM only contains the hidden state H_t, and the cell state is for internal use only. The LSTM networks are trained automatically using backpropagation [44].

The main advantage of LSTM-based networks over vanilla RNNs is that they are better at capturing long- and short-term dependencies. For example, if an element at the beginning of the sequence is very relevant to the last element of the sequence, the model can learn to *remember* this important element. On the other hand, if there is a drastic change in the observed trends in the middle of the sequence, the training process can *reset* the current state, so that it only takes into account the most relevant timestamps.

Model structure

Most of the progress in RNN-based sequence modeling has been oriented towards machine translation, which uses an encoder-decoder architecture [124, 25, 9]. The objective of the encoder part of the network is to *summarize* the input data as state vectors. The state vectors pass to the decoder layer, which is in charge of generating the outputs that best fit the input state. In the particular case of machine translation, a complete sentence such as "I love you" would be transformed into a single vector by the encoder, and then a decoder trained to generate text in Spanish would output "Te amo".

In our case, we only focus on the encoder part of the model, since our primary goal is to train the RNN in such a way that we can predict the next element in the sequence.

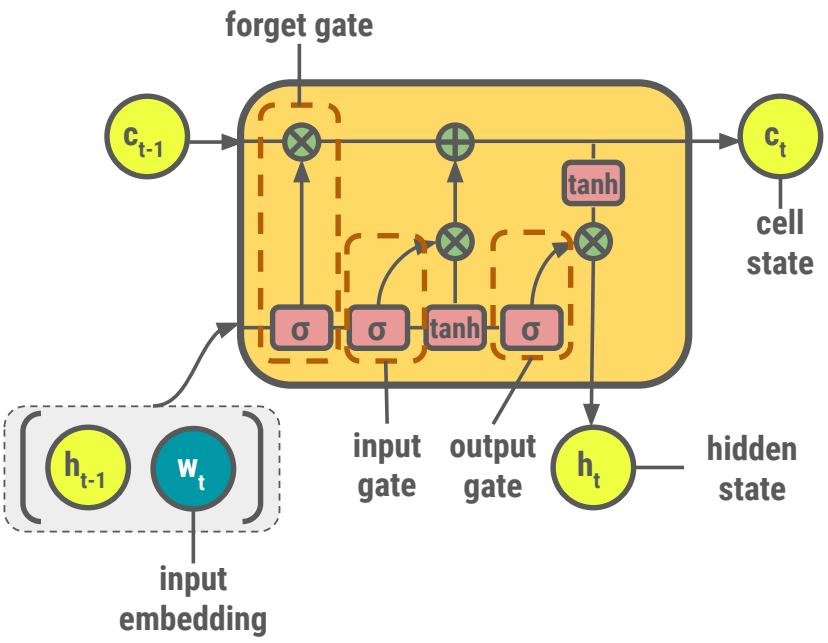


Figure 6.1: Structure of a Long Short Term Memory (LSTM) [54] cell.

Each element of the sequence is a word embedding for timestamp t, or, more generally, a fixed-size vector.

Figure 6.2 illustrates the structure of the LSTM-based network we use for word embedding prediction. In this particular diagram, we are using three LSTM cells. This means that we predict the embedding vector for the next timestamp using the word embeddings of the previous three timestamps. The number of LSTM cells, or *sequence length*, is a user-defined parameter. The dense layer before the predicted embeddings is required because the output layer of the LSTM is always between -1 and 1 due to the tanh activation of the hidden state.

6.3.2 Gated Recurrent Unit (GRU) network

The Gated Recurrent Unit (GRU) network [25] was introduced more recently to address the same RNN issues that LSTM tackles. However, GRU cells have a simpler structure than LSTM cells, which makes the training process significantly faster, and studies show

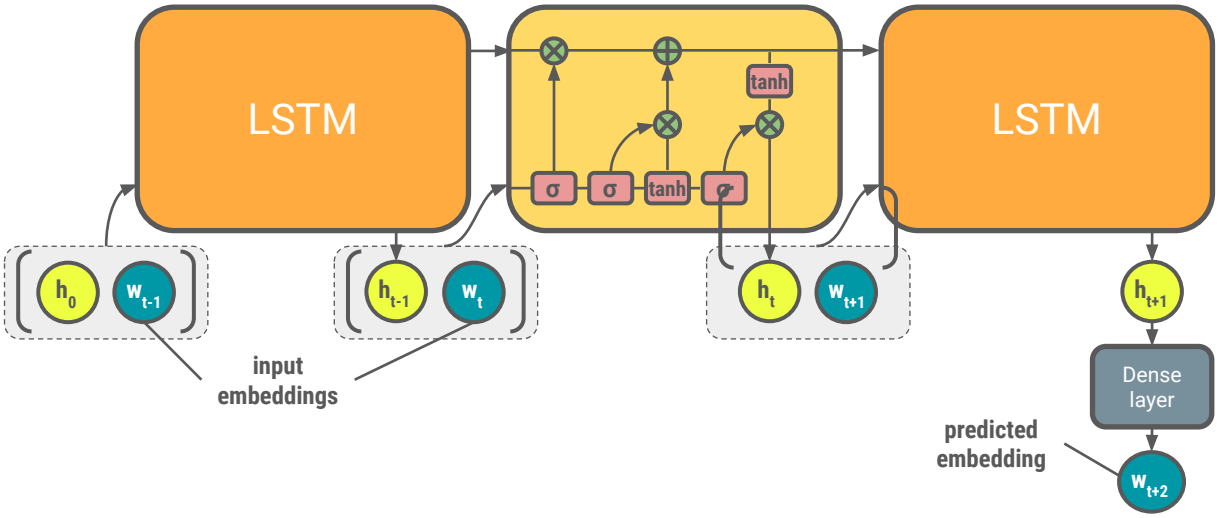


Figure 6.2: Structure of the LSTM-based network used for word embedding prediction.

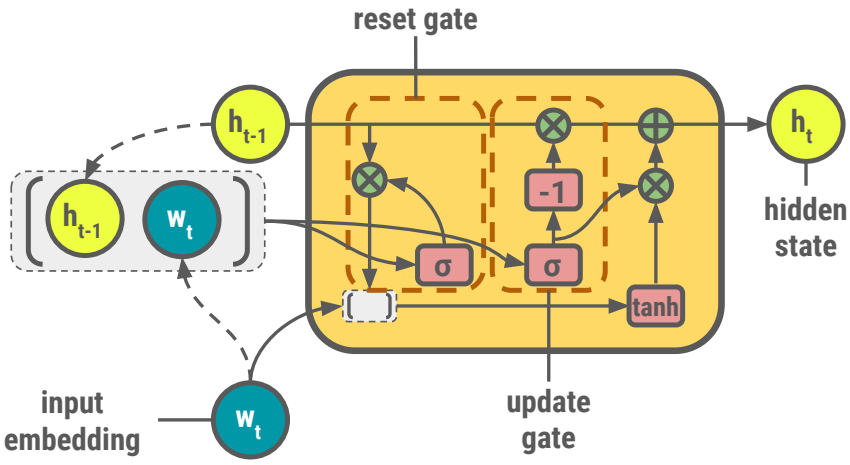


Figure 6.3: Structure of a Gated Recurrent Unit (GRU) [25] cell.

that the resulting models often are of comparable quality [25].

GRU-based networks have two weighted gates: *reset* and *update*. The *reset* gate controls how much information from the previous states should be used as input to the current cell (or forgotten) and helps capture short-term dependencies in the input sequence. The *update* gate controls how much of the old state should be used for the current hidden state, which is useful to capture long-term dependencies. These gates can take values between 0 to 1.

Figure 6.3 shows the architecture of a single GRU cell. Each GRU cell has two inputs and one output. The inputs are the previous hidden state H_{t-1}, and (2) the current word embedding w_t. The cell contains two sigmoid activation functions that control the influence of the two gates. If the *reset* gate is set to 0.0, then the hidden state only depends on the current input w_t, while having a value of 1.0 results in similar behavior to vanilla RNNs. When the *update* gate value is 1.0, the cell ignores the current input, while a gate value of 0.0 means that the cell ignores the previous information. The GRU networks are also trained automatically using backpropagation [44].

Model structure

Figure 6.4 illustrates the structure of the GRU-based network used to model the evolution of our temporal word embeddings. In this particular diagram, we are using two GRU cells, but the user-defined *sequence length* parameter can be used to change this number. This model also requires a dense layer before the predicted embeddings layer because of the tanh activation function in the GRU cell structure.

6.3.3 Attention mechanism

The goal of the LSTM- and GRU-based networks presented previously is to generate a prediction of future word embedding vectors based on an input sequence. To achieve this goal, the networks use a recurrent unit state to store the previous states. Both techniques include gates that allow them to *forget* the earlier information, in which case the cell forgets

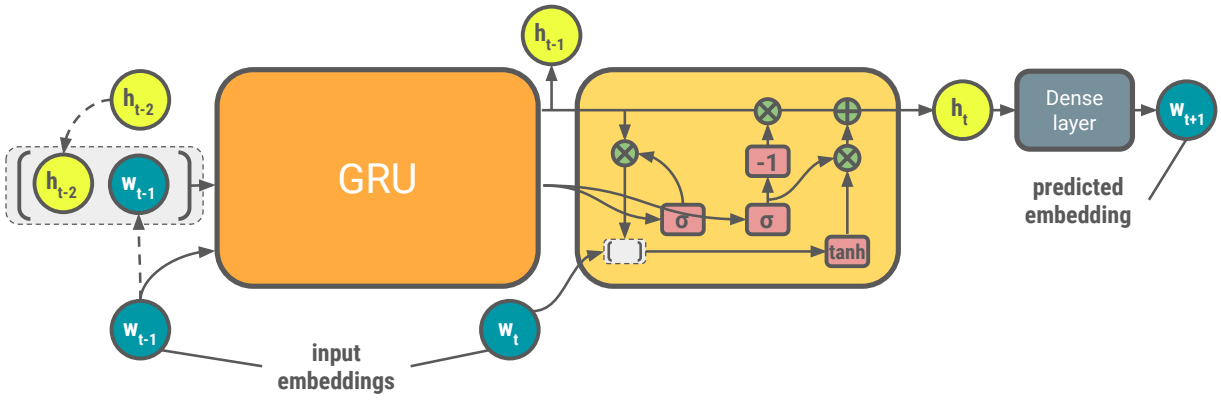


Figure 6.4: Structure of the GRU-based network used for word embedding prediction.

all of the previous states up to that point. However, it is possible that only some elements of the sequence are useful to estimate the output vector.

The *attention* mechanism [9] was introduced to allow the model to consider all of the previous information by performing a weighted linear combination of all the hidden states. This model explicitly assigns higher weights to the hidden states of the timestamps that are closely related to the predicted word vector. The *attention* mechanism significantly improves the accuracy of the machine translation task [9].

The main component of the attention mechanism is the attention layer, which contains a set of *key-value* pairs. The attention layer receives as input a *query*, and compares it against the existing *keys* using a score function. The score function that we use in this work is called Bahdanau attention [9]. The output of the score function is a measure of the similarity between the *query* and each *key*. Once the attention mechanism computes all the scores, the layer normalizes them using the softmax function. The normalized values are called *attention weights*. Finally, the output is computed by calculating the sum of the product between the attention weights and the *values* for the corresponding key. This output is called the *context vector*. The original model introduced in [9] further computes an *attention vector*, but this is the input to the decoder part of their network, so we do not need to perform this step.

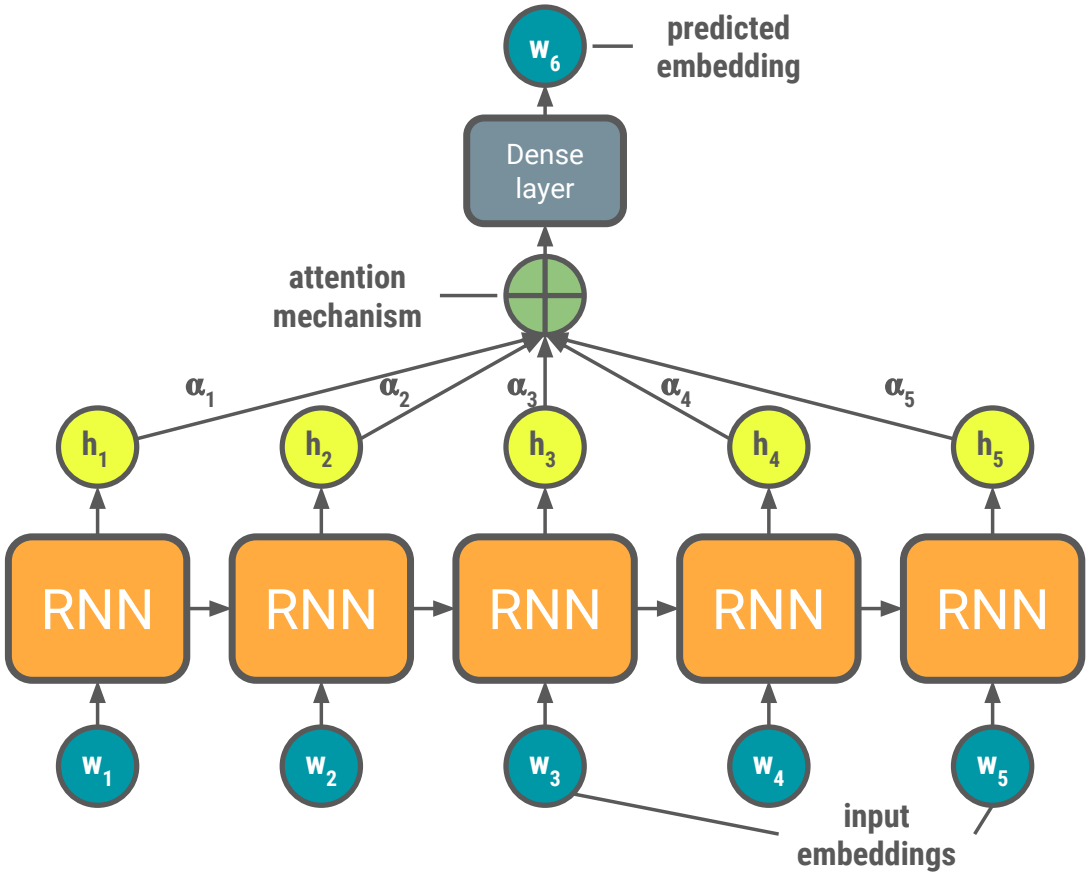


Figure 6.5: Structure of the attention-based networks [9] used for word embedding prediction, with LSTM and GRU cells instead of vanilla RNNs.

Model structure

Figure 6.5 illustrates a generic version of an attention-based network with RNNs. In our experiments, we replace the generic RNNs with LSTM and GRU cells. The α values in the figure refer to the *attention weights*. The diagram shows only one line going from one RNN to the next, but as we have explained in the LSTM section, it is possible to have more than one state passed from one cell to the next. Similar to the LSTM- and GRU-based networks, it is possible to change the training sequence length, which is set to 5 in this example for illustrative purposes.

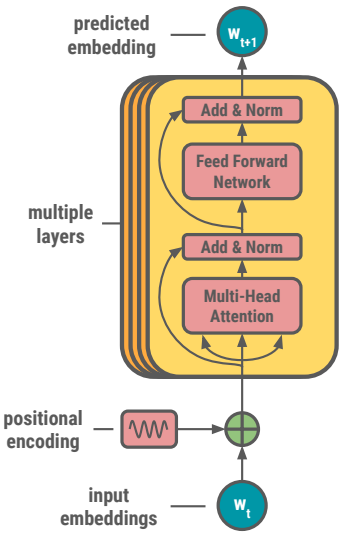


Figure 6.6: Structure of the network used for word embedding prediction using only the encoder layer from the Transformer architecture [132].

6.3.4 Transformer-based approach

The sequential nature of the different versions of RNNs prohibits parallel training. The *Transformer* model [132] gets rid of the recurrence part of the previous networks and relies completely on a self-attention mechanism. This model allows for parallel training, and it is also good at learning long-term dependencies. Furthermore, distant elements can affect each other without running into the vanishing gradients issue [132].

The *Transfomer* model uses a stack of self-attention layers to handle sequential inputs. The idea of self-attention is to be able to generate a compressed representation of a sequence by studying (or *attending* to) different positions of the input. The original *Transformer* has an encoder-decoder architecture, but as in the previous cases, we only use the encoder portion of the model.

Figure 6.6 presents a diagram of the resulting network, which consists of a stack of encoder layers, a positional encoding element, and the inputs and outputs. Each encoder layer contains:

- A multi-head attention element, which is the most important (and complex) element of the encoder, so we explain it below.
- A feed-forward dense network, which consists of a dense layer with a ReLU activation function followed by a regular dense layer.
- A normalization of the sum of the residual connection and the output of each of the previous two elements. This is introduced to avoid the vanishing gradients issue.

The positional encoding is required to give the model information about the temporal dimension of the input word embedding vectors. There are different positional encoding functions, but we use the one presented by Vaswani et al. [132], which consists of a vector of sine-cosine pairs at each position that rotate at different frequencies.

The multi-head attention layer performs four different steps:

1. The first sublayer splits the input into multiple attention and consists of a dense network.
2. The next sublayer contains an attention mechanism similar to the one presented in Section 6.3.3.
3. The following sublayer concatenates the context vectors (output of the attention mechanism) of each head.
4. The final dense sublayer receives the concatenated context vectors and provides a single fixed-dimensional vector as output.

It is necessary to split the input into multiple heads so that the model can jointly learn from different elements of the sequence at different scales. However, having multiple heads does not increase the complexity of training the model because the first step of the multi-head attention layer reduces the dimensionality of the input.

6.4 Experimental Results

In this section, we evaluate the selected time-series modeling techniques to generate temporal word embedding predictions. We perform the experiments on three different datasets: (1) PubMed abstracts [16], (2) New York Times articles, and (3) National Vulnerability Database (NVD) bulletins [21].

The PubMed corpus consists of 21 years of data with **613,949** abstracts that contain the keyword cancer. The New York Times corpus contains **812,857** news articles that were published over 29 years. The NVD dataset includes **101,273** bulletins published in the last 20 years. The analysis of the PubMed and NVD datasets was limited to the top-3,000 words based on their term frequency-inverse document frequency (tf-idf) weights, while the analysis of the New York Times corpus was limited to the top-5,000 words, based on their tf-idf weights.

The temporal word embeddings used as baseline data were generated using the method presented in Chapter 5. We split the embeddings for each dataset into training and test datasets based on their timestamps. The word embeddings of the first X years out of $|\mathcal{T}|$ are the training data. The generated models were tested using the word vectors for the last $|\mathcal{T}| - X$ years, which are not part of the training data. The X parameter is user-defined.

We evaluate the performance of the time-series modeling techniques using two different metrics: (1) average mean squared error (MSE) between the predicted and the actual word vectors, and (2) neighborhood similarity (explained next). We define *neighborhood similarity* as the average of the average number of intersections between the neighborhoods generated using the actual word embeddings and those generated using the predicted word embeddings, divided by the neighborhood size k where $k \in [1, 2, 4, 8, 16]$. The neighborhood similarity is computed only for the test data timestamps. We formalize the concept of neighborhood similarity as follows:

$$\text{neighborhood similarity}(\mathcal{N}_a, \mathcal{N}_b) = \sum_{w \in \mathcal{W}} \sum_{k \in [1,2,4,8,16]} \sum_{t \in \mathcal{T}_{\text{test}}} \frac{\mathcal{N}_a(w,t,k) \cap \mathcal{N}_b(w,t,k)}{k}$$

where $\mathcal{N}_a(w, t, k)$ returns the k nearest neighbors of word w at time t obtained from word embeddings generated using method a.

We use two different methods to generate the word vectors for future timestamps: (1) **predict-next** and (2) **predict-*i*th**. The **predict-next** approach uses only information that is already available in the form of baseline word embeddings, so effectively this method only predicts embeddings for the *next* timestamp. The **predict-*i*th** technique can predict up to the $|\mathcal{T}| + i$ timestamp by iteratively generating embeddings starting at timestamp $|\mathcal{T}| + 1$ and using these newly obtained vectors as part of the input to estimate the next element of the sequence.

The expectation is that the **predict-next** method will have a significantly higher neighbor prediction accuracy than the **predict-*i*th** approach as the value of i increases. However, reusing the predicted embeddings allows the model to extrapolate word vectors for more than one unseen timestamp, which is a desirable feature for the automatic hypothesis generation task.

In this section, we seek to answer the following questions.

1. Which sequence modeling technique is most well-suited to predict future word embeddings? (Section 6.4.1)

2. What is the relationship between the average MSE and the neighborhood similarity of the predicted word embeddings? (Section 6.4.2)

3. How sensitive is the selected time-series modeling technique to changes in the hyperparameters? (Section 6.4.3)

4. How well do the predicted word embeddings capture the neighborhood trends observed in the temporal representation introduced in Chapter 5? (Section 6.4.4)

5. How well does our algorithm predict the evolution of a specific term? (Section 6.4.5)

6. How can we leverage the predicted temporal word embeddings for hypothesis generation? (Section 6.4.6)

6.4.1 Model selection

The main goal of this experiment is to identify the sequence modeling technique that has the best performance in terms of predicting the semantic evolution of the given corpora. First, we identify the best hyperparameters by performing a sensitivity analysis for (1) LSTM, (2) GRU, (3) LSTM with attention, (4) GRU with attention, and (5) the Transformer model. Section 6.4.3 describes this sensitivity analysis in more detail.

For each model, we generate predicted word embeddings for every timestamp of the test dataset using the **predict-next** method described earlier. We use the neighborhood similarity metric to measure the performance of each model. Section 6.4.2 explores if there is a correlation between MSE and neighborhood similarity.

Figures 6.7 and 6.8 present a comparison, for each dataset, between the best versions of each sequence modeling technique. Fig. 6.7 presents the results in terms of the neighborhood similarity, while Fig. 6.8 shows the effect of changing the neighborhood size K on the average number of intersections between the baseline and the predicted embeddings for both the **predict-next** and **predict-*i*th** variants.

The results show that the GRU-based network outperforms the more complex sequential models in every dataset and prediction type. We believe that this is because we are using a small sequence length for the input given that we have a hard limit on the number of timestamps of each dataset. The attention mechanism was explicitly designed to model long sequences [132], and the regular LSTM model is consistently the second-best performer, so we think that having a small sequence length is a reasonable explanation for such a poor performance of the attention-based models.

As expected, there is also a significant difference between the neighborhood similarity obtained using the **predict-next** and **predict-*i*th** alternatives. For example, for the New York Times dataset, on average, 7.4 words out of 10 appear both in the baseline and the predicted neighborhoods using **predict-next**, while for the **predict-*i*th** method that number decreases to an average of 6.3 words out of 10. The worst performance is observed using the **predict-*i*th** approach for the NVD dataset, where the average number

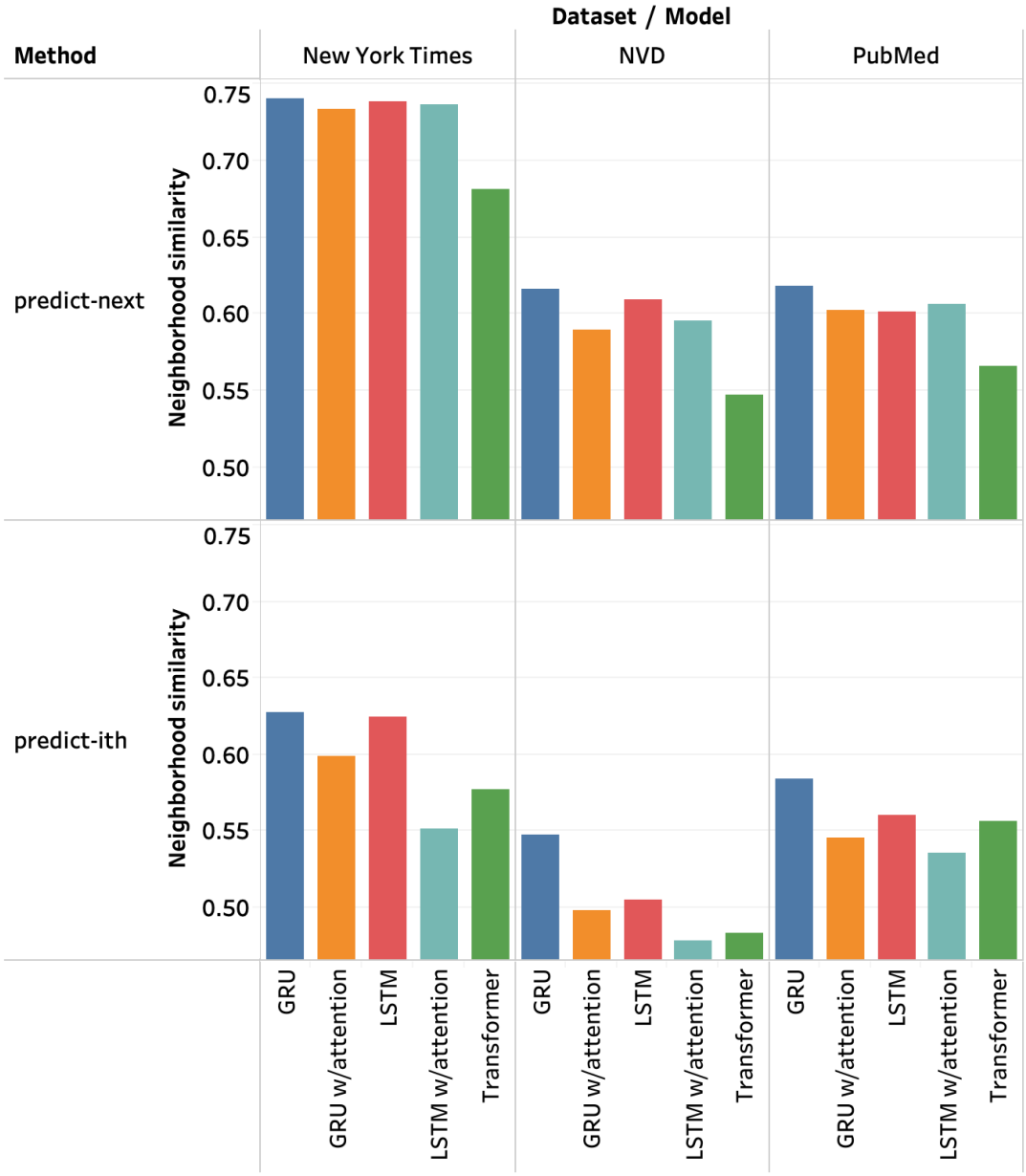


Figure 6.7: Neighborhood similarity between the neighborhoods obtained with the baseline temporal embedding method (Eq. 5.11) and neighborhoods obtained using the predict-next and predict-ith techniques with different sequential models for the NVD, PubMed and New York Times datasets, using dynamic values of K.

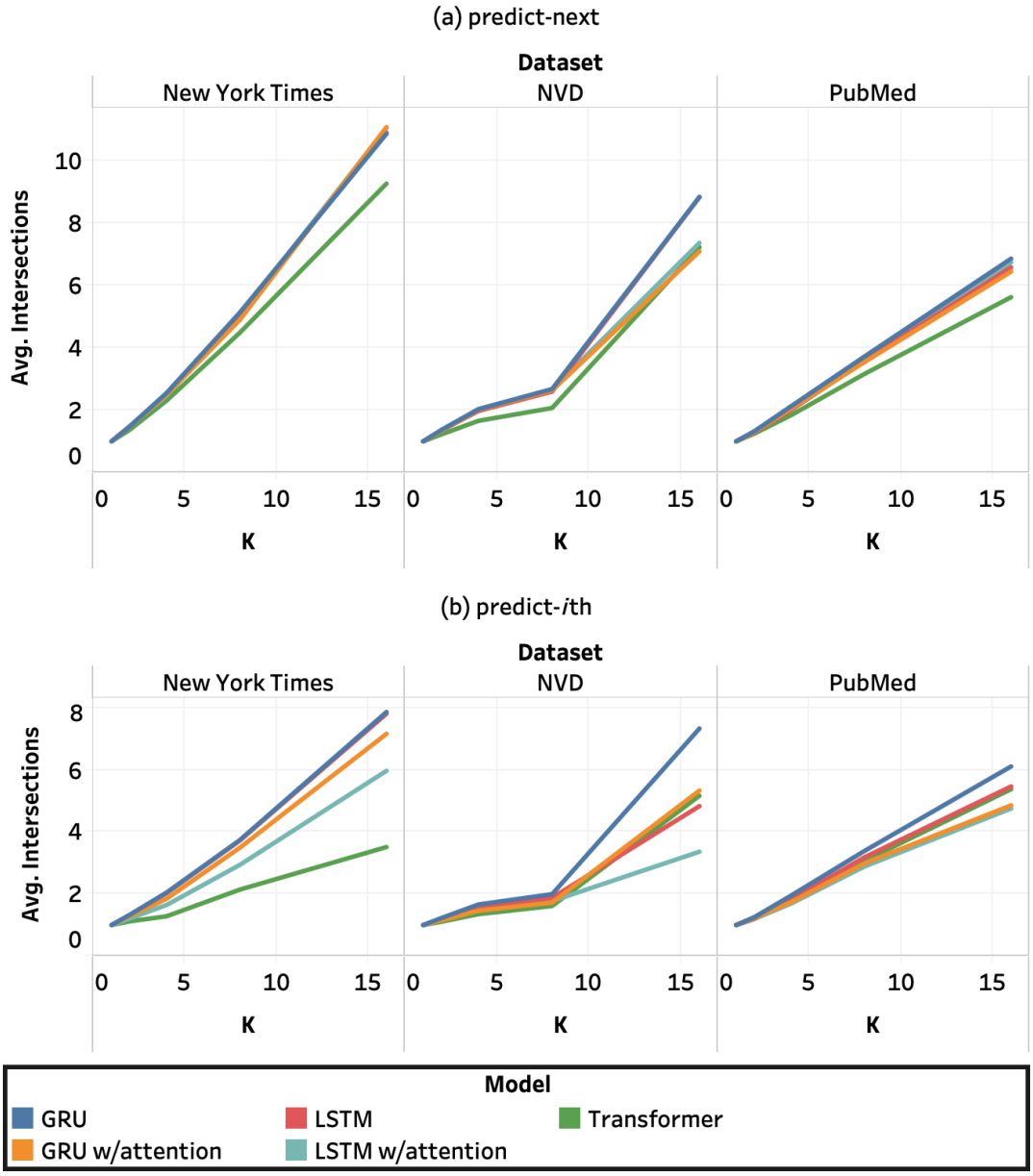


Figure 6.8: Average number of intersections for different neighborhood size k per timestamp between the neighborhoods obtained with the baseline temporal embedding method (Eq. 5.11) and neighborhoods obtained using the (a) predict-next and (b) predict-ith techniques with different sequential models for the NVD, PubMed and New York Times datasets (embedding size = 64), using dynamic values of K.

of matching words is 5.4 out of 10. We consider that a model that, on average, can capture more than half of the baseline neighborhood has good potential in terms of capturing the dataset trends.

6.4.2 Correlation between MSE and neighborhood similarity

In this section, we investigate if there exists a correlation between the MSE values and neighborhood similarity obtained using the test dataset. The assumption is that there exists a negative correlation since a lower error in the predicted word embeddings should result in a higher neighborhood similarity. It is important to confirm that this correlation exists to confidently use the train MSE metric as our optimization metric since the targeted application is to capture the semantic trends in the dataset. We quantify the performance of our model in terms of semantic trend tracking using the neighborhood similarity metric.

Figure 6.9 presents a scatterplot of the MSE values versus the neighborhood similarity computed between the neighborhoods generated using the baseline embeddings and those generated using the **predict-next** and **predict-*i*th** methods. The data points include all of the different experiments performed for the hyperparameter selection on the different techniques for the three datasets. The scatterplot suggests that there is a very strong negative correlation between the two metrics when using the **predict-next** method. We confirm this very strong negative relationship by computing the correlation coefficient between the two metrics, which results in -0.84.

For the **predict-*i*th** approach, Fig. 6.9 indicates a slightly weaker relationship between the variables. The correlation coefficient, in this case, is -0.61, which indicates a moderate to strong negative correlation. This weaker correlation can be explained because the MSE can increase significantly as we increase the number of predicted steps, while the neighborhood similarity might not decrease as much if the relative distances are kept. We further explore this behavior with a use case in Section 6.4.5, but we consider that the observed correlation is sufficient to use the MSE metric as our optimization metric confidently.

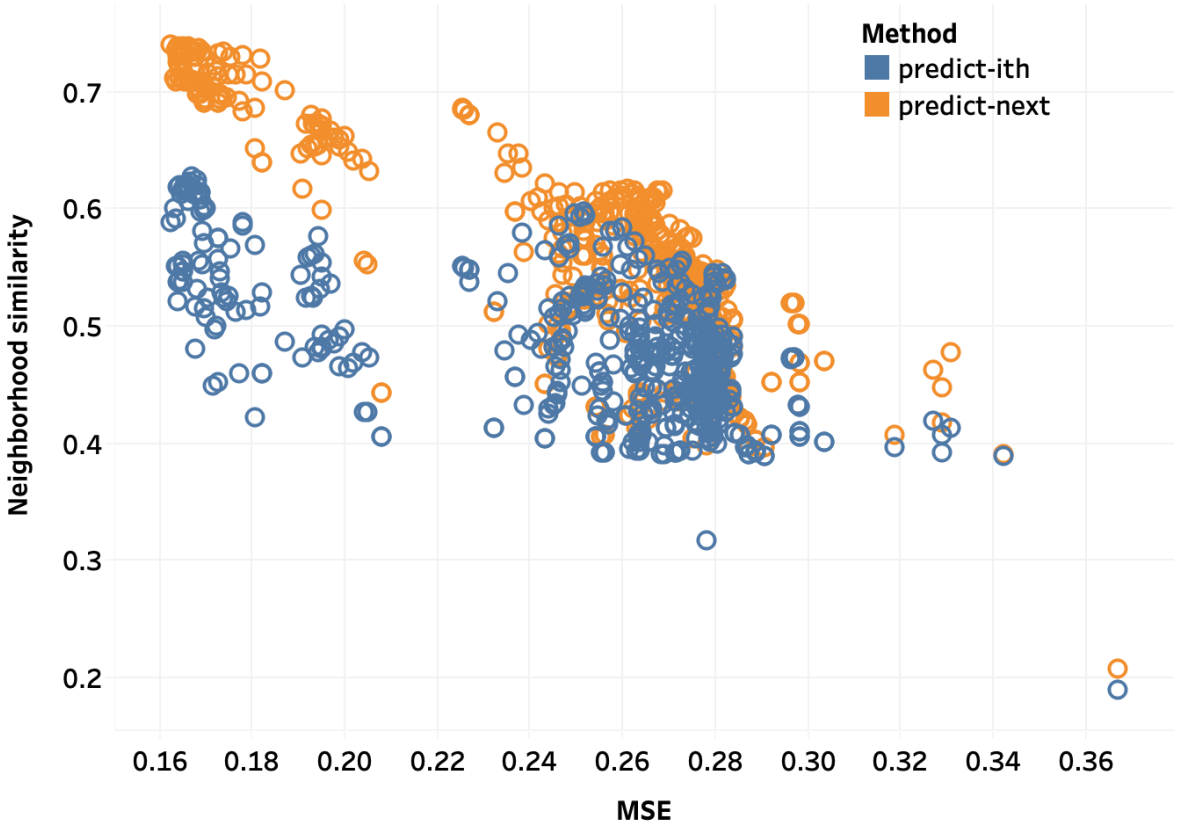


Figure 6.9: Correlation between the mean squared error (MSE) of the predicted embeddings versus the neighborhood similarity between the neighborhoods obtained with the baseline temporal embedding method (Eq. 5.11) and the neighborhoods obtained using the predict-next and predict-ith techniques with different sequential models for the NVD, PubMed and New York Times datasets, using dynamic values of K.

6.4.3 Sensitivity analysis

In this section, we present the effect of performing a sweep of the hyperparameters on the GRU-based sequential model. We performed similar analyses for the other variants, but, for brevity, we only present the results obtained with the best model. The evaluated parameters are (a) the batch size, (b) the fraction of timestamps used for training, (c) the input sequence length, (d) the number of encoder units for the neural network, (e) the optimizer, and (f) the learning rate. We use the neighborhood similarity metric to quantify the performance of each parameter combination.

Figure 6.10 presents the effect of changing the batch size (Fig. 6.10(a) and Fig. 6.10(b)) and the fraction of timestamps used for training (Fig. 6.10(c) and Fig. 6.10(d)) on neighborhood similarity for the **predict-next** (Fig. 6.10(a) and Fig. 6.10(c)) and **predict-*i*th** (Fig. 6.10(b) and Fig. 6.10(d)) methods. Based on the plot, only the fraction of timestamps used for training has a significant effect on the performance of the **predict-*i*th** approach (Fig. 6.10(d)). This is the expected behavior, because for the **predict-next** method the input comes from the original baseline. However, the **predict-*i*th** technique uses previously predicted embeddings as input, so having a smaller number of test timestamps results in a smaller number of predicted embeddings used as input.

Figures 6.11 (a) and (b) present the effect of changing the input sequence length on neighborhood similarity for the **predict-next** (Fig. 6.11(a)) and **predict-*i*th**(Fig. 6.11(b)) methods. Figure 6.11(b) reveals that using an input consisting of the word embeddings for three or more timestamps results in a better performance when using the **predict-*i*th** method. Figures 6.11 (c) and (d) show that changing the number of encoder units for the neural network has a negligible effect on the neighborhood similarity for the **predict-next** and **predict-*i*th** methods.

Figures 6.12 (a) and (b) present the effect of changing the optimizer and learning rate on neighborhood similarity for the **predict-next** (Fig. 6.12(a)) and **predict-*i*th** (Fig. 6.12(b)) methods. It is important to note that for these plots, a learning rate of 0.0 on the x-axis actually represents the neighborhood similarity obtained using the dynamic learning rate

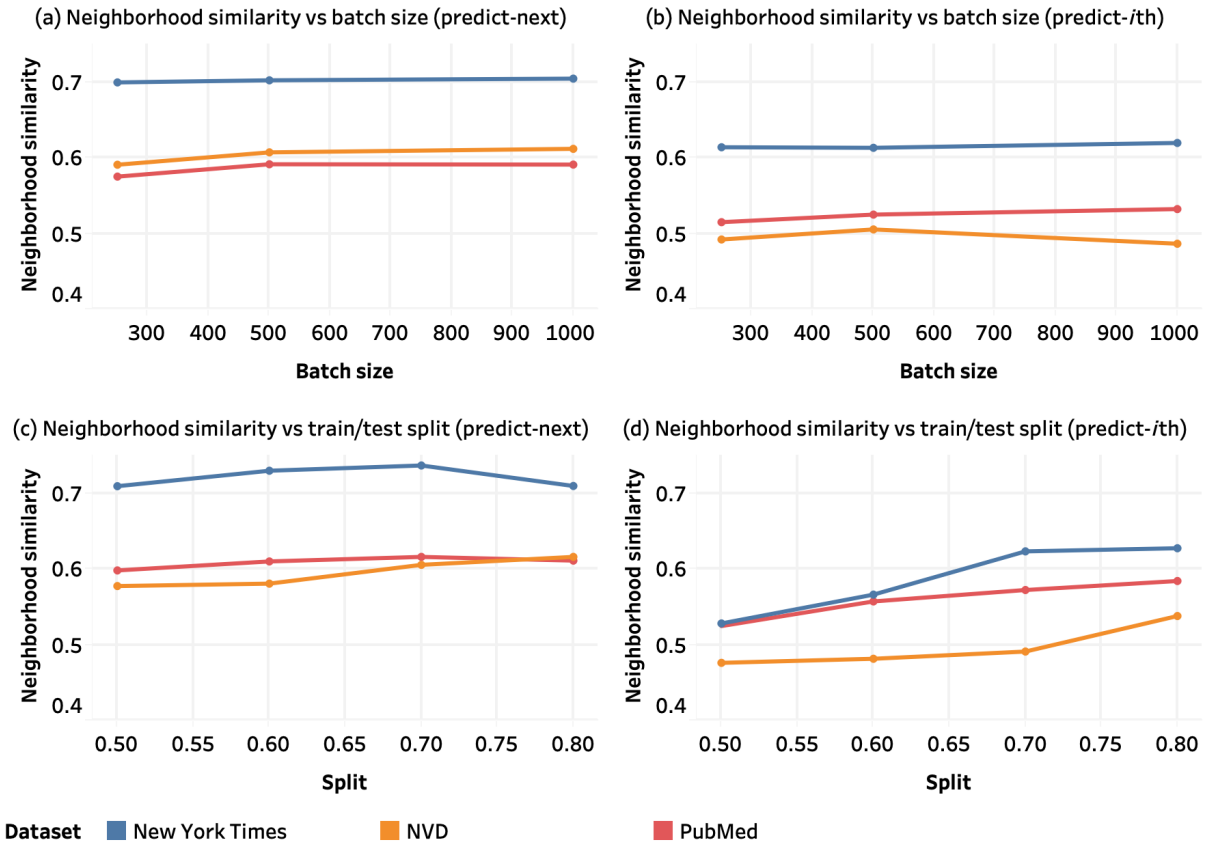


Figure 6.10: Neighborhood similarity between the neighborhoods obtained with the baseline temporal embedding method (Eq. 5.11) and the neighborhoods obtained using the predict-next and predict-ith techniques with different sequential models for the NVD, PubMed and New York Times datasets, using dynamic values of K, while changing (a)(b) the batch size and (c)(d) the fraction of timestamps used for training.

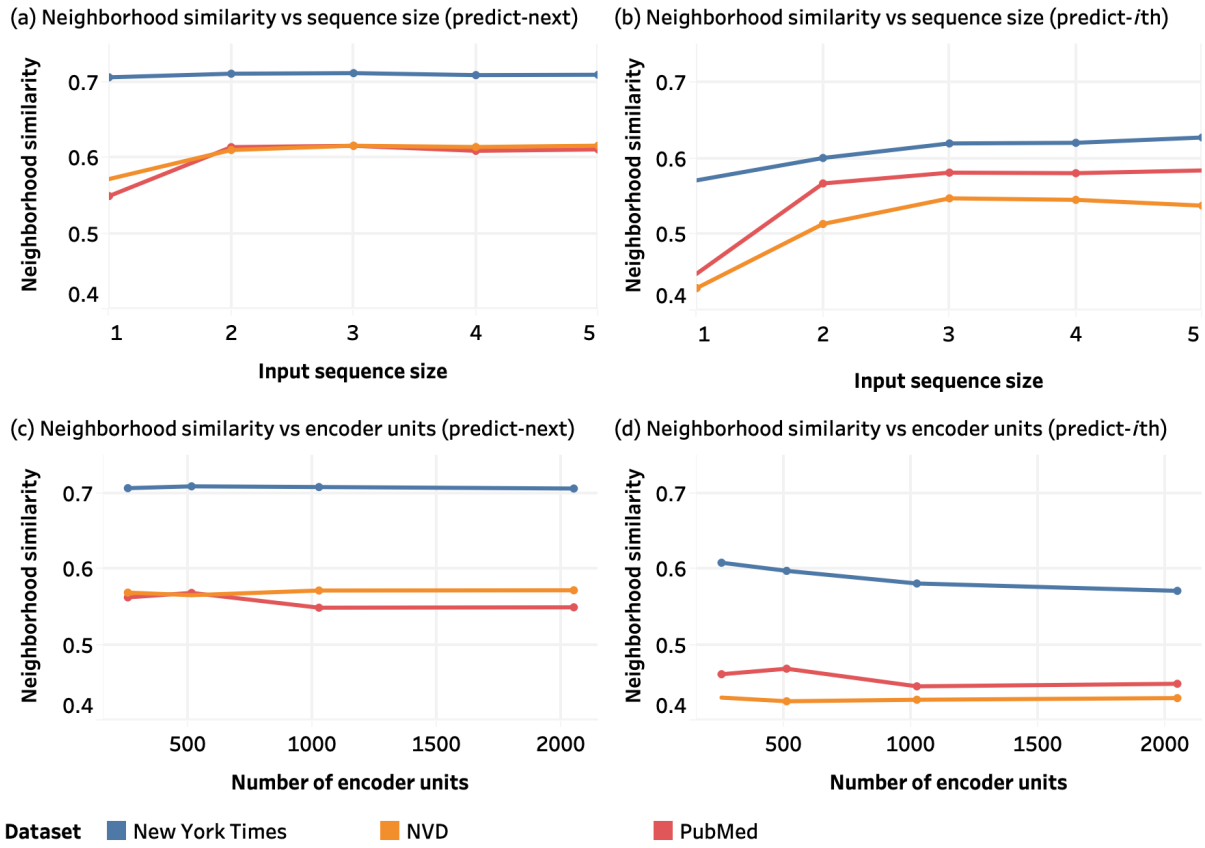


Figure 6.11: Neighborhood similarity between the neighborhoods obtained with the baseline temporal embedding method (Eq. 5.11) and the neighborhoods obtained using the predict-next and predict-ith techniques with different sequential models for the NVD, PubMed and New York Times datasets, using dynamic values of K, while changing (a)(b) the input sequence size and (c)(d) the number of network units.

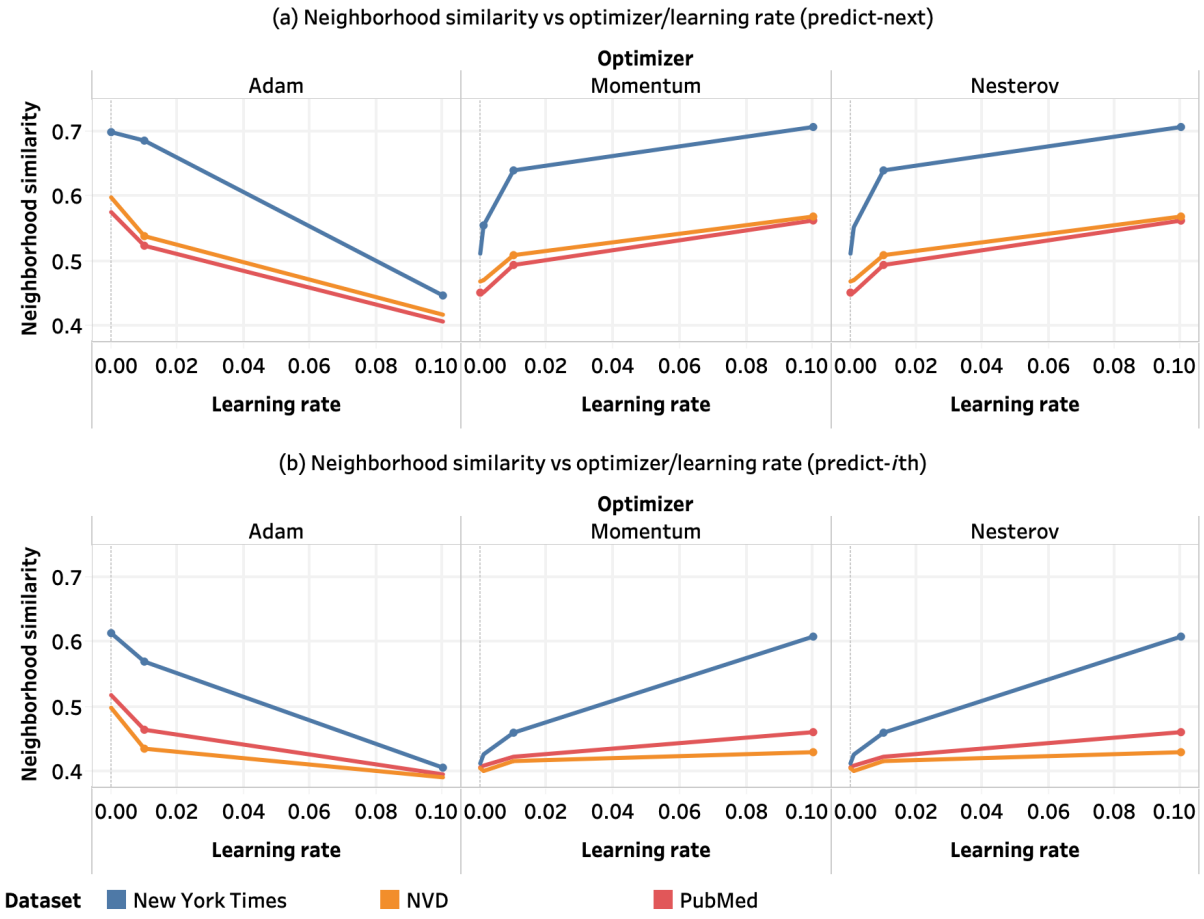


Figure 6.12: Neighborhood similarity between the neighborhoods obtained with the baseline temporal embedding method (Eq. 5.11) and neighborhoods obtained using the (a) predict-next and (b) predict-ith techniques with different sequential models for the NVD, PubMed and New York Times datasets, using dynamic values of K, for different optimizers and learning rates. The learning rate of 0.0 represents the dynamic learning rate presented by Vaswani et al. [132].

presented by Vaswani et al. [132]. The results clearly exhibit the importance of testing different learning rate values to make sure the optimizer does not get stuck in local minima.

6.4.4 Trend analysis

In this section, we perform a qualitative analysis of the performance of the predicted temporal word embeddings on the task of tracking the semantic evolution, when compared to the original embeddings introduced in Chapter 5. First, we use the **predict-next** and **predict-*i*th** methods presented in this chapter to generate predicted word embeddings for every timestamp in the test dataset, and we extrapolate the embeddings for one timestamp using **predict-next** and three timestamps using **predict-*i*th**. Next, we obtain the neighborhood for every word at every timestamp by sorting the neighbors based on the cosine distance between the base word vector and the rest of the vectors. For this experiment, we only analyze the temporal evolution of the top-10 words such that their regular tf-idf-based vectors have the highest cosine similarity at any point in time with the tf-idf vector for the word of interest.

Figure 6.13 presents the evolution of the neighborhood of the term *chronic myeloid leukemia* at different points in time, for (b) the **predict-next** approach, (b) the baseline temporal embedding method (Eq. 5.11), and (c) the **predict-*i*th** method. The height of each stream represents the cosine similarity (i.e., 1.0-cosine distance) between the word vector for *treatment* and the vector of the stream label. The three dotted lines in the plot represent, from left to right, (1) where the test data starts for the **predict-next** method, (2) where the test data starts for the **predict-*i*th** approach, and (3) where the test data ends. Thus, the relevant part of the plots is after the first dotted line for Fig. 6.13(a) when compared to Fig. 6.13(b), and after the second dotted line for Fig. 6.13(c) when compared to Fig. 6.13(b).

The results show that the **predict-next** approach performs very well on this task, which we expected because the input data uses the baseline data, so the word vectors are artificially *realigned* at every timestamp, even though we are actually predicting the

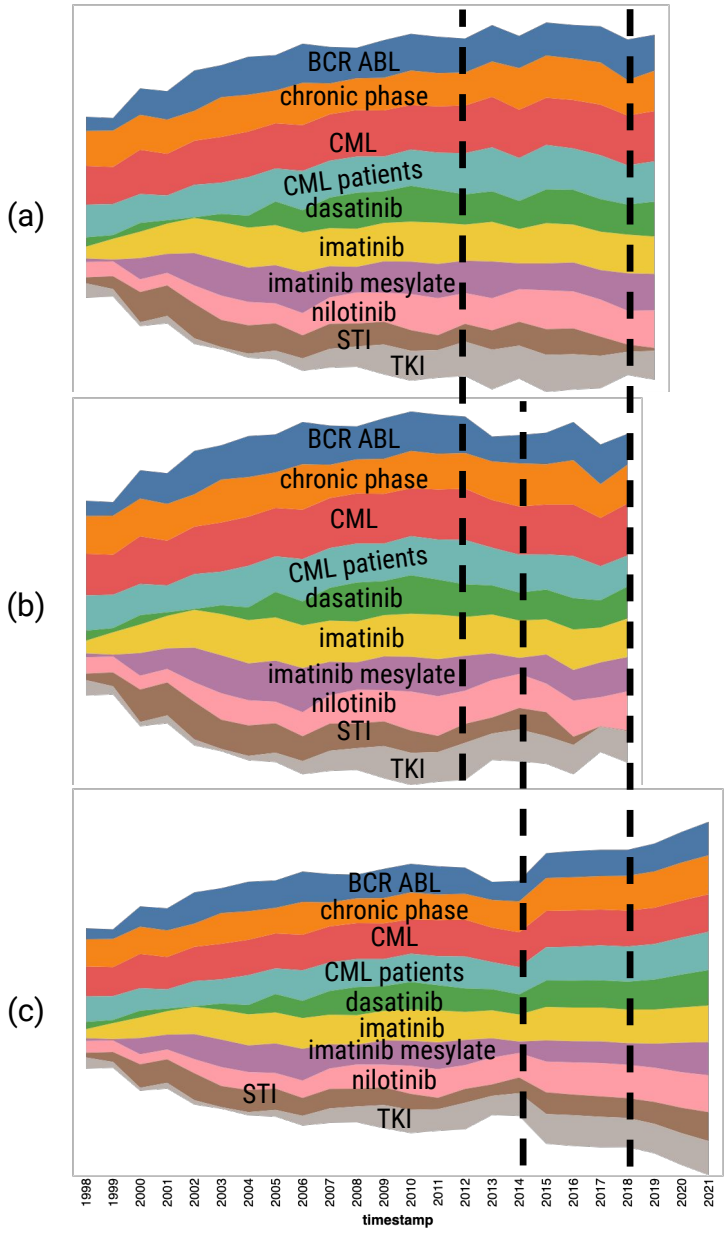


Figure 6.13: Evolution of the neighborhood of the term *chronic myeloid leukemia* in the Pubmed dataset using: (a) the predict-next method with test timestamps from 2013 to 2019 (b) the temporal embedding method presented in Chapter 5, and (c) the predict-ith method with test timestamps from 2015 to 2021.

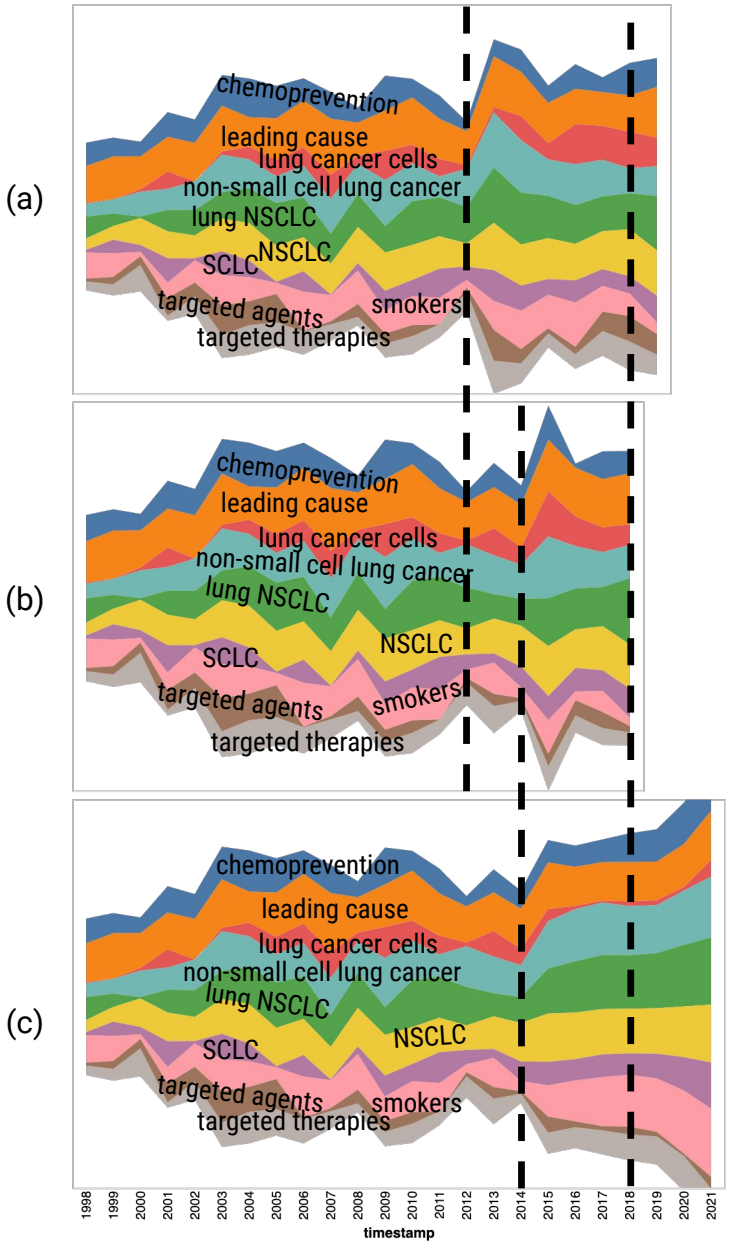


Figure 6.14: Evolution of the neighborhood of the term *lung cancer* in the Pubmed dataset using: (a) the predict-next method with test timestamps from 2013 to 2019 (b) the temporal embedding method presented in Chapter 5, and (c) the predict-ith method with test timestamps from 2015 to 2021.

embeddings for the next timestamp. The best example of this is the term *TKI*, which in Fig. 6.13(b) shrinks at around 2017, while in Fig. 6.13(a) we observe this reduction in cosine similarity until the next timestamp. Examining Fig. 6.13(c), we observe that the the cosine similarity between *chronic myeloid leukemia* and *TKI* increases instead of decreasing. Based on this observation, we can infer that the **predict-*i*th** method is not able to handle trends that are not present in the previous timestamps. We expected this behavior because there is no way for the generated model to *guess* that there will be a change without having sufficient information, which is the case of the **predict-next** approach.

Figure 6.14 shows the evolution of the neighborhood of the term *lung cancer* at different points in time, for (b) the **predict-next** approach, (b) the baseline temporal embedding method (Eq. 5.11), and (c) the **predict-*i*th** method, following the same format as Fig. 6.13. The trends of Fig. 6.14(a) again show a remarkable similarity with those on Fig. 6.14(b), with the exception of the term *smokers* which appears to have a higher similarity in the predicted vectors. The results from Fig. 6.14(c) do a good job at capturing the general trends observed in the baseline model, with a few specific exceptions such as *lung cancer cells* and *smokers*.

Overall, the qualitative analysis presented in this section offers promising results in terms of trend analytics, assuming that there are no extreme changes from one timestamp to the other if the training data does not contain this type of change. Appendix D contains more examples that show that our models can track most of the semantic changes successfully.

6.4.5 Case study

In this experiment, we analyze the semantic changes of the term *president* from the New York Times dataset using the baseline temporal embeddings (Table 6.1), the **predict-next** method (Table 6.2) and the **predict-*i*th** method (Table 6.3). Tables 6.1, 6.2, and 6.3 show the top-5 nearest neighbors of the word *president* obtained using each of the three

Table 6.1: Nearest neighbors for the test timestamps of the word *president* obtained using the temporal embeddings method.

2014	**2015**	**2016**	**2017**	**2018**
Obama	Trump	Trump	Trump	Trump
chairman	Obama	Obama	White House	White House
office	Clinton	White House	Washington	Washington
senate	Washington	office	United States	office
Washington	country	Clinton	office	United States

Table 6.2: Nearest neighbors for the test timestamps of the word *president* obtained using the predict-next method.

2014	**2015**	**2016**	**2017**	**2018**	**2019**
Obama	Obama	Trump	Trump	Trump	Trump
Clinton	White House	Obama	Obama	White House	White House
campaign	Washington	Washington	White House	United States	United States
White House	office	Clinton	country	Washington	Washington
United States	chairman	state	office	office	Congress

Table 6.3: Nearest neighbors for the test timestamps of the word *president* obtained using the predict-ith method.

2014	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
Obama	Obama	Obama	Obama	Obama	Obama	VP	VP
White House	White House	VP	campaign	campaign	VP	Obama	views
Clinton	VP	chairman	director	agency	Kerry	staff	strategy
Washington	Biden	campaign	agency	director	leadership	Oval Office	staff
campaign	Washington	director	agencies	agencies	staff	Kerry	vice chairman

methods. The resulting neighborhoods illustrate that both the temporal word embeddings (Table 6.1) and the **predict-next** (Table 6.2) approach are able to capture significantly well the changes in the context of the word *president*, with the presidential candidates *Trump* and *Clinton* appearing in the neighborhood around 2015-2016, with the presidential election taking place during 2016. After the election year, the term *Clinton* disappears from the context while the term *Trump* is still the nearest neighbor, as expected.

The **predict-*i*th** method (Table 6.3), on the other hand, tells a completely different story, with terms such as *Kerry* and *Vice President (VP)* appearing in the timestamps where we would expect the appearance of terms such as *Trump*. As mentioned in the last section, this is mainly because the term *Trump* only gets closer to the word *president* in the training data, so the **predict-*i*th** method is lacking information that could identify a potential trend that included *Trump* in the same context as *president*.

Even though the term *Trump* does not appear in the top-5 nearest neighbors, we seek to investigate the trend of the cosine similarity between *Trump* and *president*. Figure 6.15 presents the evolution of the neighborhood of the term *president* at different points in

time, for (b) the **predict-next** approach, (b) the baseline temporal embedding method (Eq. 5.11), and (c) the **predict-*i*th** method, including the term *Trump*. As can be seen from Fig. 6.15(c), the trend is that the word *Trump* will get closer to the term *president* over time, which is the actual trend, even though the change was much faster in reality.

Finally, we wanted to explore our conjecture from Section 6.4.2 that the weak relationship between MSE and neighborhood similarity has more to do with large differences between the predicted and the baseline word vectors, than with a weak performance in terms of neighborhood similarity, which depends more on the relative distances between vectors. Figure 6.16 presents how the L2 norm of the vectors for the word *president* changes over time for the three different methods. The plot shows that the word vectors generated using the **predict-*i*th** approach change significantly, while the **predict-next**-based vectors show a more attenuated shift. However, our qualitative analysis has shown that both methods perform well in capturing the trends the neighborhoods follow over time, which is our main goal. Thus, we believe that our sequential models are actually shrinking the whole embedding space, but because for our application the absolute value of the embeddings is irrelevant, we leave this issue for a future work.

6.4.6 Hypothesis generation

In this section, we propose a simple technique to automatically generate interesting hypotheses based on the embeddings generated using the **predict-next** or **predict-*i*th** method. We perform a qualitative evaluation of several of the obtained hypotheses. The closed-hypothesis generation procedure consists of the following steps:

1. The user defines the word of interest, or *base word*, for which related words must be found.

2. Use the **predict-next** or **predict-*i*th** method to generate predicted word embeddings for i timestamps, where i is a user-defined parameter. When using the **predict-next** method i is always equal to 1.

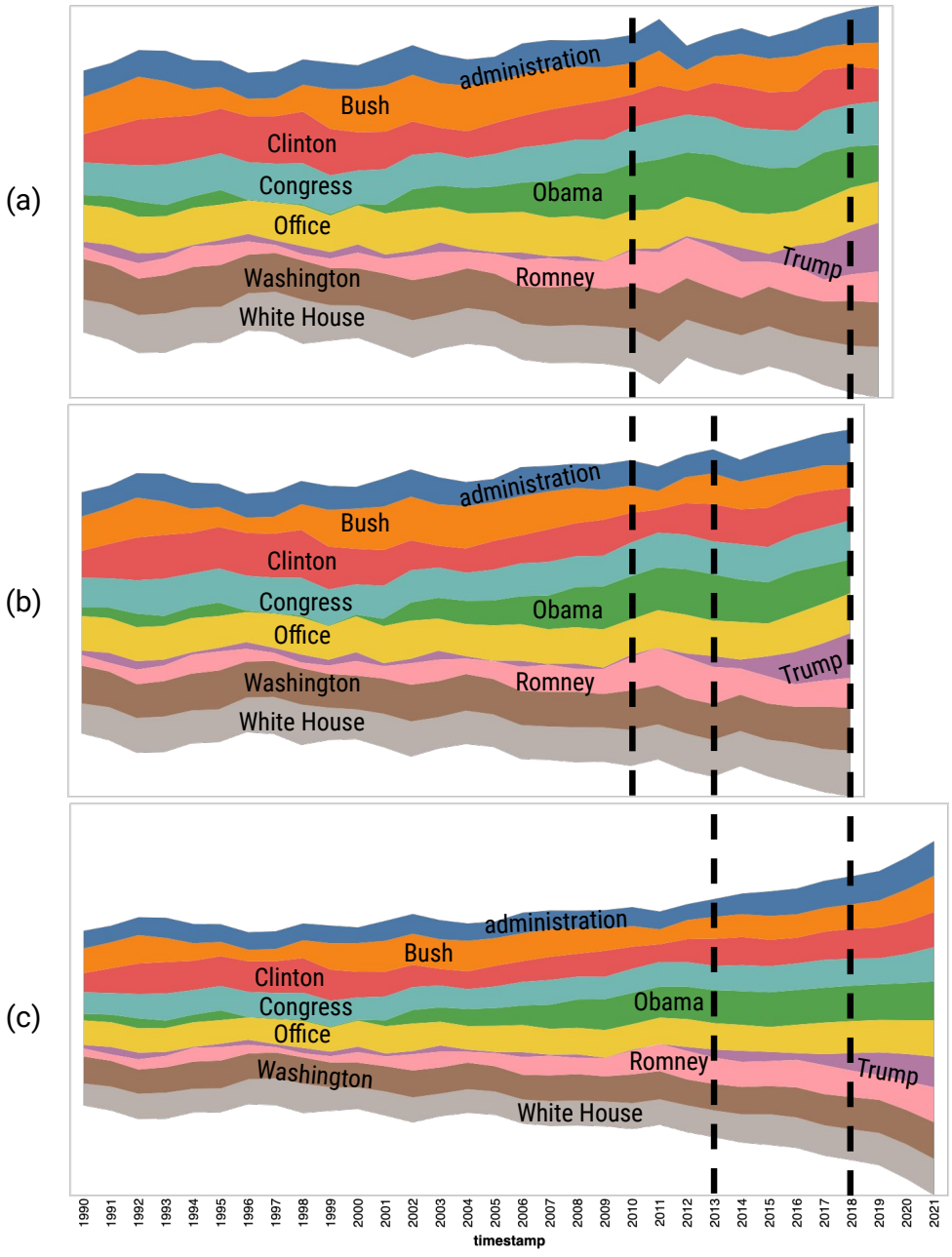


Figure 6.15: Evolution of the neighborhood of the term *president* in the NYT dataset using: (a) the predict-next embeddings with test timestamps from 2011 to 2019 (b) the temporal embedding method presented in Chapter 5, and (c) the predict-ith embeddings with test timestamps from 2014 to 2021.

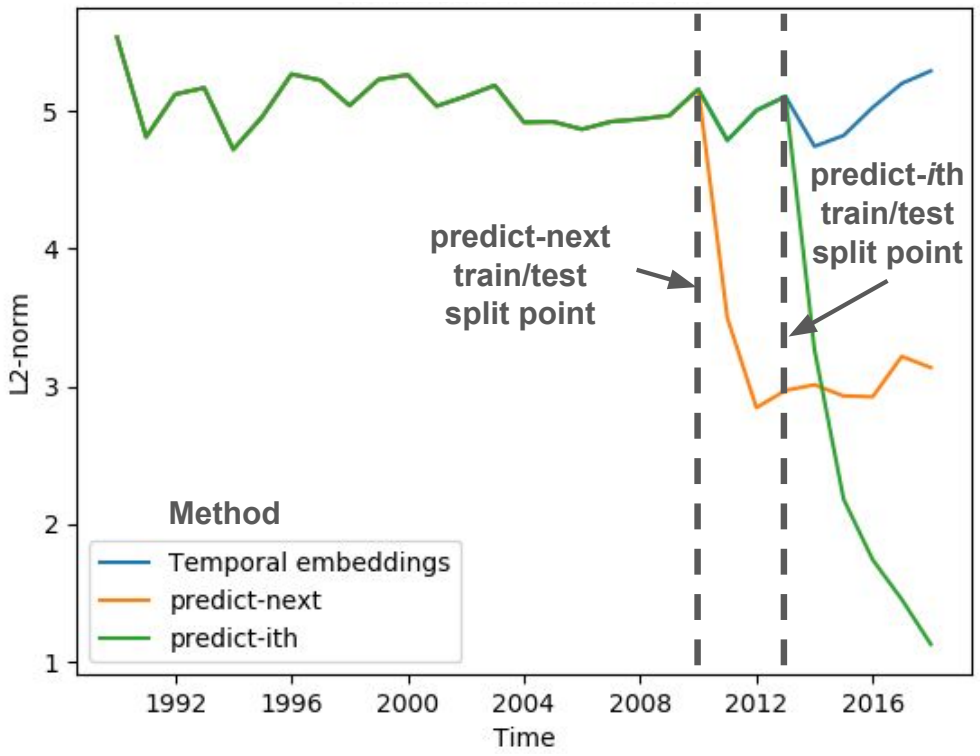


Figure 6.16: Evolution of the L2-norm of the vectors for the term *president* for the temporal embeddings, the predict-next embeddings and the predict-ith embeddings. The plot shows the split between train and test data for the predicted embeddings.

3. Obtain the top-n nearest neighbors of the *base word* for every timestamp, where the neighborhood size of interest n is a user-defined parameter.

4. Assuming the neighborhood for each timestamp is a set, compute the set difference between the neighborhood of the last timestamp with known data and the neighborhood of the last predicted timestamp.

The set of words resulting from the previous procedure includes terms that do not appear in the top-n neighborhood of the real embeddings. However, our model predicts, based on the current trends, that these words will appear in the top-n neighborhood i timestamps away from the present time. The user can then hypothesize that there potentially exists a previously-unknown relationship between the *base word* and the obtained words.

Tables 6.4, 6.5, and 6.6 present the neighborhoods of the word *nausea* from the PubMed dataset obtained using the original temporal embeddings (Table 6.4), as well as the embeddings generated using the **predict-next** (Table 6.5) and **predict-*i*th** (Table 6.6) methods. The procedure described above takes the neighborhood obtained using the temporal embeddings for 2018, and compares it with the neighborhood of 2019 for the **predict-next** approach and with the neighborhood of 2021 for the **predict-*i*th** method. The hypothesis generated using this procedure is that the term *nausea* might be related to *anemia*. According to the National Heart, Lung, and Blood Institute, *nausea* is a common symptom of a condition called *pernicious anemia*[101]. However, the most common type of *anemia*, caused by iron-deficiency, does not have *nausea* as its symptom [100], which explains why it is not part of the original neighborhood.

Tables 6.7, 6.8, and 6.9 present the neighborhoods of the term *maximo asset management* from the NVD dataset obtained using the original temporal embeddings (Table 6.7), **predict-next**-based embeddings (Table 6.8) and **predict-*i*th**-based embeddings (Table 6.9). The hypotheses generated using our algorithm are that the term *maximo asset management* might be related to *Java SE*, *album*, or *CWE improper restriction*. In this case there is no way to verify these relationships, but with this information we could guess

Table 6.4: Nearest neighbors for the test timestamps of the word *nausea* obtained using the temporal embeddings method on the PubMed dataset.

2015	**2016**	**2017**	**2018**
vomiting	vomiting	vomiting	vomiting
fatigue	fatigue	fatigue	fatigue
diarrhea	diarrhea	diarrhea	diarrhea
anorexia	anorexia	anorexia	neutropenia
grade	diarrhoea	diarrhoea	anorexia

Table 6.5: Nearest neighbors for the test timestamps of the word *nausea* obtained using the predict-next method on the PubMed dataset.

2015	**2016**	**2017**	**2018**	**2019**
vomiting	vomiting	vomiting	vomiting	vomiting
anorexia	fatigue	anorexia	anorexia	fatigue
fatigue	diarrhea	fatigue	fatigue	diarrhea
diarrhea	anorexia	diarrhoea	diarrhea	anorexia
neutropenia	diarrhoea	diarrhea	diarrhoea	neutropenia

Table 6.6: Nearest neighbors for the test timestamps of the word *nausea* obtained using the predict-ith method on the PubMed dataset.

2015	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
vomiting	vomiting	vomiting	vomiting	vomiting	vomiting	vomiting
anorexia	fatigue	fatigue	anorexia	fatigue	fatigue	anorexia
fatigue	anorexia	anorexia	fatigue	anorexia	anorexia	fatigue
diarrhea	diarrhea	diarrhea	diarrhea	anemia	anemia	anemia
neutropenia	anemia	anemia	diarrhoea	diarrhea	diarrhea	diarrhea

Table 6.7: Nearest neighbors for the test timestamps of the word *maximo asset management* obtained using the temporal embeddings method on the NVD dataset.

2015	**2016**	**2017**	**2018**
smartcloud control desk	tivoli asset management	smartcloud control desk	smartcloud control desk
tivoli asset management	smartcloud control desk	tivoli asset management	tivoli asset management
ifix	ifix	ifix	ibm maximo asset management
ibm maximo asset management	configuration management database ccmdb	ibm maximo asset management	ifix
configuration management database ccmdb	ibm maximo asset management	configuration management database ccmdb	double extension

that there might exist a vulnerability on IBM's *Maximo Asset Management* solution that can only be exploited on the *Java SE* platform, or that *Maximo Asset Management* has a vulnerability related to the common weakness *CWE-307: Improper Restriction of Excessive Authentication Attempts* [21].

Tables 6.10, 6.11, and 6.12 present the neighborhoods of the term *protests* from the New York Times dataset obtained using temporal embeddings (Table 6.10), **predict-next** (Table 6.11) and **predict-*i*th** (Table 6.12) methods. The hypotheses generated using our algorithm are that the term *protests* have a relationship with words such as *security forces*, *streets*, and *revolution*. Even though the relationships between these terms are well-known, we find it particularly interesting that our model projects that the term *revolution* will gain traction in 2021, based on the current trends.

Appendices E, F and G present more results in which our procedure identified the following interesting relationships:

Table 6.8: Nearest neighbors for the test timestamps of the word *maximo asset management* obtained using the predict-next method on the NVD dataset.

2015	**2016**	**2017**	**2018**	**2019**
smartcloud control desk	smartcloud control desk	smartcloud control desk	smartcloud control desk	smartcloud control desk
tivoli asset management	tivoli asset management	tivoli asset management	tivoli asset management	tivoli asset management
ibm maximo asset management	ibm maximo asset management	ifix	ifix	ifix
ifix	ifix	ibm maximo asset management	configuration management database ccmdb	ibm maximo asset management
configuration management database ccmdb	change	configuration management database ccmdb	ibm maximo asset management	album

Table 6.9: Nearest neighbors for the test timestamps of the word *maximo asset management* obtained using the predict-*i*th method on the NVD dataset.

2015	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
smartcloud control desk	smartcloud control desk	smartcloud control desk	smartcloud control desk	smartcloud control desk	smartcloud control desk	smartcloud control desk
tivoli asset management	tivoli asset management	tivoli asset management	tivoli asset management	tivoli asset management	tivoli asset management	tivoli asset management
ibm maximo asset management	ibm maximo asset management	ibm maximo asset management	ibm maximo asset management	ibm maximo asset management	ibm maximo asset management	ibm maximo asset management
ifix	ifix	ifix	change	change	java se	java se
change	change	change	ifix	java se	change	cwe improper restriction

Table 6.10: Nearest neighbors for the test timestamps of the word *protests* obtained using the temporal embeddings method on the New York Times dataset.

2014	**2015**	**2016**	**2017**	**2018**
protesters	protesters	demonstrations	demonstrations	demonstrations
demonstrations	demonstrators	demonstrators	demonstrators	protesters
protest	protest	protest	protesters	protest
demonstrators	demonstrations	protesters	protest	demonstrators
streets	streets	police officers	demonstration	organizers

Table 6.11: Nearest neighbors for the test timestamps of the word *protests* obtained using the predict-next method on the New York Times dataset.

2014	**2015**	**2016**	**2017**	**2018**	**2019**
protesters	protesters	protesters	demonst-rations	demonst-rations	protesters
demonst-rations	protest	demonst-rators	demonst-rators	demonst-rators	demonst-rations
demonst-rators	demonst-rations	demonst-rations	protesters	protesters	protest
protest	demonst-rators	protest	protest	protest	demonst-rators
streets	bangkok	streets	police offi-cers	demonst-ration	streets

Table 6.12: Nearest neighbors for the test timestamps of the word *protests* obtained using the predict-ith method on the New York Times dataset.

2014	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
protesters	protesters	protesters	protesters	protesters	protesters	protesters	protesters
demonst-rations	demonst-rations	demonst-rations	demonst-rations	demonst-rations	demonst-rations	demonst-rations	demonst-rations
demonst-rators	protest	protest	protest	demonst-rators	demonst-rators	police of-ficers	security forces
protest	demonst-rators	demonst-rators	demonst-rators	protest	police of-ficers	demonst-rators	demonst-rators
streets	demonst-ration	demonst-ration	streets	streets	protest	protest	revolution

- *imatinib & BCR-ABL*
- *vomiting & dyspnea*
- *catalase & ErbB*
- *cat gene & tamofixen treatment*
- *ranibizumab & mitomycin-C*
- *spoof servers & long cmd args*
- *Wireshark & proxy server*
- *South Korea & Philippines*
- *treatment & diabetes*

6.5 Conclusions

In this chapter, we leverage different types of sequential models to generate predicted future embeddings that can capture the existing semantic trends. The results indicate that out of five different techniques, the most simple model (GRU) provides the best performance, which can be explained by the low number of timestamps on the training data. We introduced two different embedding generation procedures, **predict-next** and **predict-*i*th**, where the former outperforms the latter significantly, but the **predict-next** approach is limited to predicting only the embeddings of the next timestamp.

With both models, it is impossible to replicate sudden semantic changes that are only observed in the test data since the generated model accounts only for the trends in the training data. Additionally, the further away in time the predicted embeddings are from the current timestamp, the lower the expected accuracy of the neighborhoods. Finally, we introduce a simple technique for automatic hypothesis generation that can identify pairs of words that might have a latent relationship, which is not evident in the original text corpus. Through multiple examples, we have shown that this approach has a very high potential for use in applications such as Literature-Based Discovery [52] and intelligence analysis.

Chapter 7

Concluding Remarks

7.1 Significance of the Results

The current state-of-the-art in automatic extraction of significant, relevant, and useful trends from large temporal text corpora is still in its early stages. Identifying such trends can aid, for example, (1) intelligence analysis, identifying potential threats against national security, or (2) scientific research, by finding latent relations between previously unrelated terms that can form hypotheses which can then be verified empirically.

In this work, we introduce a full framework that automatically generates a vector-space representation of a text corpus that can capture the semantic evolution of the vocabulary over time. The generated model is robust against the sparsity, uncertainty, and noise present in real-world data. We improve a preliminary high-dimensional temporal representation by using a neural-network-based approach to reduce its dimensionality and improve the continuity of the spatial representation over time.

Finally, we use sequential modeling techniques to predict the future semantic evolution of the vocabulary based on current trends. The predicted word embeddings can then be used to identify pairs of words that are potentially related, i.e., a hypothesis. Our framework achieves promising results in both the semantic evolution tracking and hypotheses generation tasks.

We designed our framework in such a way that every part of the process is standalone, which means that the user can employ our techniques for many other potential applications. For example, media providers could automatically generate a temporal vector representation about the preferences of the user over time. Then, they could apply the work presented

in Chapters 5 and 6 to automatically predict if the user preferences will change in the future and adjust their recommendations according to the predicted behavior.

7.2 Future Work

As future work, we would first address the embedding space shrinkage observed in Chapter 6, since for other applications, it might be useful to be able to replicate the actual values of the embeddings in the predicted vectors. A new interesting direction would be to test the current framework with a larger variety of time-based datasets, such as online reviews, Facebook posts, and public forums, as well as for other applications such as recommender systems.

In its current state, the framework consists of four main stages that are sequentially dependent: (1) preprocessing, (2) time-reflective vector representation, (3) temporal embeddings, and (4) sequential model. A potential improvement to the framework as a whole would be to implement a single network that performs steps 2 through 4, taking as input the original tf-idf representation and returning a sequential model. However, one potential issue with this idea is that the temporal word embeddings are actually the weights of a neural network, so the backpropagation step in a general network might be overly complicated.

We could also explore using other methods to model the semantic trends in the corpus, such as adversarial networks and vector auto-regressive models. Further, we believe that we can use density-based clustering to divide the vocabulary into groups with similar semantic trends and generate forecasting models for each cluster. The main issue with this approach is that we could lose information about the latent relationships that might exist across clusters.

References

[1] Harshavardhan Achrekar, Avinash Gandhe, Ross Lazarus, Ssu-Hsin Yu, and Benyuan Liu. Predicting flu trends using twitter data. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 702–707. IEEE, 2011.

[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*, WWW '11, pages 267–276, New York, NY, USA, 2011. ACM.

[3] Jean Aitchison. *Language change: progress or decay?* Cambridge approaches to linguistics. Cambridge University Press, Oxford ; New York, 4th ed edition, 2013.

[4] Alias-i. LingPipe 4.1.0, 2008.

[5] American Cancer Society. Chemotherapy for Stomach Cancer, 2017.

[6] American Cancer Society. Treatment Choices by Type and Stage of Stomach Cancer, 2017.

[7] J Angulo, C Pederneiras, W Ebner, E Kimura, and P Megale. Concepts of diffusion theory and a graphic approach to the description of the epidemic flow of contagious disease. *Public Health Rep*, 95(5):478–485, 1980.

[8] Seung Han Baek, Dahee Lee, Minjoo Kim, Jong Ho Lee, and Min Song. Enriching plausible new hypothesis generation in PubMed. *PLoS ONE*, 12(7), July 2017.

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]*, September 2014. arXiv: 1409.0473.

[10] Kimberly A. Baltzell, Hua Min Shen, Savitri Krishnamurthy, Jennette D. Sison, Gerard J. Nuovo, and Gertrude C. Buehring. Bovine leukemia virus linked to breast cancer but not coinfection with human papillomavirus: Case-control study of women in Texas. *Cancer*, 124(7):1342–1349, 2018.

[11] Robert Bamler and Stephan Mandt. Dynamic Word Embeddings. In *Int. Conf. on Machine Learning*, pages 380–389, July 2017.

[12] Oren Barkan. Bayesian Neural Word Embedding. In *AAAI*, pages 3135–3143, 2017.

[13] R. C. Barranco, L. M. Rodriguez, R. Urbina, and M. S. Hossain. Enhancing Yelp Data with Deep Learning and Information Reuse. In *2017 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 452–461, August 2017.

[14] Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. Analysing word meaning over time by exploiting temporal random indexing. In *Italian Conf. on Computational Linguistics CLiC-it*, 2014.

[15] Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 95–104. ACM, 2007.

[16] (MD) Bethesda. PubMed, 1946.

[17] David M. Blei and John D. Lafferty. Dynamic Topic Models. In *Proc. Int. Conf. on Machine Learning*, ICML '06, pages 113–120, New York, NY, USA, 2006. ACM.

[18] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[19] Johan Bollen, Huina Mao, and Alberto Pepe. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. *Icwsm*, 11:450–453, 2011.

[20] Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of computational science*, 2(1):1–8, 2011.

[21] Harold Booth, Doug Rike, and Gregory A. Witte. The National Vulnerability Database (NVD): Overview. *ITL Bulletin* -, December 2013.

[22] Roberto Camacho, Arnold P. Boedihardjo, and M. Shahriar Hossain. Analyzing evolving stories in news articles. *International Journal of Data Science and Analytics*, December 2017.

[23] Roberto Camacho, 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, Seattle, WA, USA, December 2018. IEEE.

[24] Sung-Hyuk Cha. Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. *International Journal of Mathematical models and Methods in Applied Sciences*, 1(4):300–307, 2007.

[25] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078 [cs, stat]*, June 2014. arXiv: 1406.1078.

[26] Jorge Cortes, Elias Jabbour, Hagop Kantarjian, C. Cameron Yin, Jianqin Shan, Susan O'Brien, Guillermo Garcia-Manero, Francis Giles, Megan Breeden, Nubia Reeves, William G. Wierda, and Dan Jones. Dynamics of BCR-ABL kinase domain mutations in chronic myeloid leukemia after sequential treatment with multiple tyrosine kinase inhibitors. *Blood*, 110(12):4005–4011, December 2007.

[27] Scott Coyne, Praveen Madiraju, and Joseph Coelho. Forecasting Stock Prices Using Social Media Analysis. In *Dependable, Autonomic and Secure Comput-*

ing, 15th Intl Conf on Pervasive Intelligence & Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), 2017 IEEE 15th Intl, pages 1031–1038. IEEE, 2017.

[28] Lucia Martinez Cuesta, Pamela Anahi Lendez, Maria Victoria Nieto Farias, Guillermina Laura Dolcini, and Maria Carolina Ceriani. Can Bovine Leukemia Virus Be Related to Human Breast Cancer? A Review of the Evidence. *J Mammary Gland Biol Neoplasia*, pages 1–7, May 2018.

[29] Ramzi Dagher, Martin Cohen, Gene Williams, Mark Rothmann, Jogarao Gobburu, Gabriel Robbie, Atiqur Rahman, Gang Chen, Ann Staten, Donna Griebel, and Richard Pazdur. Approval Summary: Imatinib Mesylate in the Treatment of Metastatic and/or Unresectable Malignant Gastrointestinal Stromal Tumors. *Clinical Cancer Research*, 8(10):3034–3038, October 2002.

[30] Valerio Di Carlo, Federico Bianchi, and Matteo Palmonari. Training Temporal Word Embeddings with a Compass. *arXiv:1906.02376 [cs]*, June 2019. arXiv: 1906.02376.

[31] Peter Sheridan Dodds, Kameron Decker Harris, Isabel M. Kloumann, Catherine A. Bliss, and Christopher M. Danforth. Temporal patterns of happiness and information in a global social network: Hedonometrics and Twitter. *PloS one*, 6(12):e26752, 2011.

[32] Brian J. Druker, Franois Guilhot, Stephen G. O'Brien, Insa Gathmann, Hagop Kantarjian, Norbert Gattermann, Michael W.N. Deininger, Richard T. Silver, John M. Goldman, Richard M. Stone, Francisco Cervantes, Andreas Hochhaus, Bayard L. Powell, Janice L. Gabrilove, Philippe Rousselot, Josy Reiffers, Jan J. Cornelissen, Timothy Hughes, Hermine Agis, Thomas Fischer, Gregor Verhoef, John Shepherd, Giuseppe Saglio, Alois Gratwohl, Johan L. Nielsen, Jerald P. Radich, Bengt Simonsson, Kerry Taylor, Michele Baccarani, Charlene So, Laurie Letvak, and Richard A.

Larson. Five-Year Follow-up of Patients Receiving Imatinib for Chronic Myeloid Leukemia. *N Engl J Med*, 355(23):2408–2417, December 2006.

[33] Haim Dubossarsky, Daphna Weinshall, and Eitan Grossman. Outta Control: Laws of Semantic Change and Inherent Biases in Word Representation Models. In *Proc. of Conf. on EMNLP*, pages 1136–1145. ACL, September 2017.

[34] Lauri Eronen and Hannu Toivonen. Biomine: predicting links between biological entities using network models of heterogeneous databases. *BMC bioinformatics*, 13(1):119, 2012.

[35] 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*, KDD '04, pages 118–127, New York, NY, USA, 2004. ACM.

[36] Lujun Fang, Anish Das Sarma, Cong Yu, and Philip Bohannon. REX: Explaining Relationships Between Entity Pairs. *Proc. VLDB Endow.*, 5(3):241–252, November 2011.

[37] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[38] Kirsten Fischer, Paula Cramer, Raymonde Busch, Sebastian Bttcher, Jasmin Bahlo, Jerg Schubert, Karl H. Pflger, Silke Schott, Valentin Goede, Susanne Isfort, Julia von Tresckow, Anna-Maria Fink, Andreas Bhler, Dirk Winkler, Karl-Anton Kreuzer, Peter Staib, Matthias Ritgen, Michael Kneba, Hartmut Dhner, Barbara F. Eichhorst, Michael Hallek, Stephan Stilgenbauer, and Clemens-Martin Wendtner. Bendamustine in Combination With Rituximab for Previously Untreated Patients With Chronic

Lymphocytic Leukemia: A Multicenter Phase II Trial of the German Chronic Lymphocytic Leukemia Study Group. *J Clin Oncol*, 30(26):3209–3216, September 2012.

[39] Lea Frermann and Mirella Lapata. A Bayesian Model of Diachronic Meaning Change. *Transactions of the Association for Computational Linguistics*, 4:31–45, 2016.

[40] 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*, EMNLP-CoNLL '12, pages 710–720, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[41] Anna Gladkova and Aleksandr Drozd. Intrinsic Evaluations of Word Embeddings: What Can We Do Better? In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 36–42, Berlin, Germany, August 2016. Association for Computational Linguistics.

[42] Akanksha Goel, Divanshu Khandelwal, Jayant Mundhra, and Ritu Tiwari. Intelligent and Integrated Book Recommendation and Best Price Identifier System Using Machine Learning. In *Intelligent Engineering Informatics*, pages 397–412. Springer, 2018.

[43] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, May 2013.

[44] Alex Graves and Jrgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

[45] Wanrong Gu, Shoubin Dong, and Mingquan Chen. Personalized news recommendation based on articles chain building. *Neural Computing and Applications*, 27(5):1263–1272, July 2016.

[46] David Hall, Daniel Jurafsky, and Christopher D. Manning. Studying the History of Ideas Using Topic Models. In *Proc. of Conf. on EMNLP*, pages 363–371. ACL, 2008.

[47] William L. Hamilton, Jure Leskovec, and Dan Jurafsky. Cultural Shift or Linguistic Drift? Comparing Two Computational Measures of Semantic Change. In *Proc. of Conf. on EMNLP*, pages 2116–2121, November 2016.

[48] William L. Hamilton, Jure Leskovec, and Dan Jurafsky. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. In *Proc. of ACL*, volume 1, pages 1489–1501, 2016.

[49] Sonya Haslam. Dasatinib: the emerging evidence of its potential in the treatment of chronic myeloid leukemia. *Core Evidence*, 1(1):1–12, 2005.

[50] K. Heath, N. Gelfand, M. Ovsjanikov, M. Aanjaneya, and L. J. Guibas. Image webs: Computing and exploiting connectivity in image collections. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3432–3439, June 2010.

[51] Johannes Hellrich and Udo Hahn. Bad Company-Neighborhoods in Neural Embedding Spaces Considered Harmful. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2785–2796, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

[52] Sam Henry and Bridget T. McInnes. Literature Based Discovery: Models, methods, and trends. *Journal of Biomedical Informatics*, 74:20–32, October 2017.

[53] Gerhard Heyer, Florian Wood, and Sven Teresniak. Change of Topics over Time and Tracking Topics by Their Change of Meaning. In Ana LN Fred, editor, *Proc. of Int. Conf. on KDIR*. INSTICC Press, October 2009.

[54] Sepp Hochreiter and Jrgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[55] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 2017.

[56] M. Shahriar Hossain, Christopher Andrews, Naren Ramakrishnan, and Chris North. Helping Intelligence Analysts Make Connections. In *Proceedings of the 17th AAAI Conference on Scalable Integration of Analytics and Visualization*, AAAIWS'11-17, pages 22–31, Menlo Park, California, 2011. AAAI Press.

[57] M. Shahriar Hossain, Patrick Butler, Arnold P. Boedihardjo, and Naren Ramakrishnan. Storytelling in Entity Networks to Support Intelligence Analysts. In *Proc. of ACM SIGKDD*, pages 1375–1383, New York, NY, USA, 2012. ACM.

[58] 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, January 2012.

[59] Renfen Hu, Shen Li, and Shichen Liang. Diachronic Sense Modeling with Deep Contextualized Word Embeddings: An Ecological View. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3899–3908, Florence, Italy, July 2019. Association for Computational Linguistics.

[60] Xiaolei Huang and Michael J. Paul. Neural Temporality Adaptation for Document Classification: Diachronic Word Embeddings and Domain Adaptation Models. In

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4113–4123, Florence, Italy, July 2019. Association for Computational Linguistics.

[61] A. Jatowt and K. Duh. A framework for analyzing semantic change of words across time. In *IEEE/ACM Joint Conference on Digital Libraries*, pages 229–238, September 2014.

[62] Brendan J. Jenkins, Virginie Deswaerte, Paul M. Nguyen, Alison West, Alison F. Browning, Liang Yu, Saleela Ruwanpura, Jesse Balic, Thaleia Livis, Charlotte Girard, Adele Preaudet, Hiroko Oshima, Ka Yee Fung, Hazel Tye, Meri Najdovska, Matthias Ernst, Masanobu Oshima, Cem Gabay, and Tracy L. Putoczki. Inflammasome adaptor ASC suppresses apoptosis of gastric cancer cells by an IL-18 mediated inflammation-independent mechanism. *Cancer Research*, page canres.1887.2017, January 2017.

[63] Chunyi Jiang, Jinhong Zhu, Pengcheng Zhou, Huijun Zhu, Wei Wang, Qin Jin, and Peng Li. Overexpression of FIBCD1 Is Predictive of Poor Prognosis in Gastric Cancer. *American Journal of Clinical Pathology*, 149(6):474–483, April 2018.

[64] Yookyung Jo, John E. Hopcroft, and Carl Lagoze. The Web of Topics: Discovering the Topology of Topic Evolution in a Corpus. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 257–266, New York, NY, USA, 2011. ACM.

[65] Mitsuro Kanda and Yasuhiro Kodera. Molecular mechanisms of peritoneal dissemination in gastric cancer. *World Journal of Gastroenterology*, 22(30):6829–6840, August 2016.

[66] Hagop Kantarjian, Charles Sawyers, Andreas Hochhaus, Francois Guilhot, Charles Schiffer, Carlo Gambacorti-Passerini, Dietger Niederwieser, Debra Resta, Renaud

Capdeville, Ulrike Zoellner, Moshe Talpaz, and Brian Druker. Hematologic and Cytogenetic Responses to Imatinib Mesylate in Chronic Myelogenous Leukemia. *N Engl J Med*, 346(9):645–652, February 2002.

[67] Tom Kenter, Melvin Wevers, Pim Huijnen, and Maarten de Rijke. Ad Hoc Monitoring of Vocabulary Shifts over Time. In *Proc. of ACM CIKM*, pages 1191–1200, New York, NY, USA, 2015. ACM.

[68] Leyla Kilic, Cetin Ordu, Ibrahim Yildiz, Fatma Sen, Serkan Keskin, Rumeysa Ciftci, and Kezban Nur Pilanci. Current adjuvant treatment modalities for gastric cancer: From history to the future. *World J Gastrointest Oncol*, 8(5):439–449, May 2016.

[69] Dongwoo Kim and Alice Oh. Topic Chains for Understanding a News Corpus. In *Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part II*, CICLing'11, pages 163–176, Berlin, Heidelberg, 2011. Springer-Verlag.

[70] Yoon Kim, Yi-I. Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. Temporal Analysis of Language through Neural Language Models. *Proc. of ACL*, pages 61–65, 2014.

[71] Alexander Kinsora, Kate Barron, Qiaozhu Mei, and VG Vinod Vydiswaran. Creating a Labeled Dataset for Medical Misinformation in Health Forums. In *Healthcare Informatics (ICHI), 2017 IEEE International Conference on*, pages 456–461. IEEE, 2017.

[72] Jon Kleinberg. Bursty and Hierarchical Structure in Streams. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 91–101, New York, NY, USA, 2002. ACM.

[73] W Koizumi, S Tanabe, K Saigenji, A Ohtsu, N Boku, F Nagashima, K Shirao, Y Matsumura, and M Gotoh. Phase I/II study of S-1 combined with cisplatin in

patients with advanced gastric cancer. *British Journal of Cancer*, 89(12):2207–2212, December 2003.

[74] Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Statistically Significant Detection of Linguistic Change. In *Proc. of Int. Conf. on WWW*, pages 625–635, 2015.

[75] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[76] D. Kumar, N. Ramakrishnan, R. F. Helm, and M. Potts. Algorithms for Storytelling. *IEEE Transactions on Knowledge and Data Engineering*, 20(6):736–751, June 2008.

[77] Erdal Kuzey, Jilles Vreeken, and Gerhard Weikum. A Fresh Look on Knowledge Bases: Distilling Named Events from News. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 1689–1698, New York, NY, USA, 2014. ACM.

[78] Xiaojiang Lei, Xueming Qian, and Guoshuai Zhao. Rating prediction based on social sentiment from textual reviews. *IEEE Transactions on Multimedia*, 18(9):1910–1921, 2016.

[79] Jure Leskovec and Rok Sosi. SNAP: A General-Purpose Network Analysis and Graph-Mining Library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1, 2016.

[80] Ying-Yang Liao, Ning-Fu Peng, Di Long, Peng-Cheng Yu, Sen Zhang, Jian-Hong Zhong, and Le-Qun Li. Hepatectomy for liver metastases from gastric cancer: a systematic review. *BMC Surg*, 17, February 2017.

[81] Xiangfeng Luo, Junyu Xuan, Jie Lu, and Guangquan Zhang. Measuring the Semantic Uncertainty of News Events for Evolution Potential Estimation. *ACM Trans. Inf. Syst.*, 34(4):24:1–24:25, June 2016.

[82] Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. AutoPlait: Automatic Mining of Co-evolving Time Sequences. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 193–204, New York, NY, USA, 2014. ACM. event-place: Snowbird, Utah, USA.

[83] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM, 2013.

[84] Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2015.

[85] Amy McGovern, Derek H Rosendahl, Rodger A Brown, and Kelvin K Droegemeier. Identifying predictive multi-dimensional time series motifs: an application to severe weather prediction. *Data Mining and Knowledge Discovery*, 22(1-2):232–258, 2011.

[86] Md Abdul Kader, Sheikh Motahar Naim, Arnold P. Boedihardjo, and M. Shahriar Hossain. Connecting the Dots Using Contextual Information Hidden in Text and Images. *AAAI Conference on Artificial Intelligence; Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[87] Rada Mihalcea and Vivi Nastase. Word Epoch Disambiguation: Finding How Words Change over Time. In *Proc. of ACL*, pages 259–263, 2012.

[88] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*, January 2013. arXiv: 1301.3781.

[89] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. *Adv Neural Inf Process Syst*, pages 3111–3119, 2013.

[90] C.-K. Min, K.-S. Eom, S. Lee, D. W. Kim, J. W. Lee, WS Min, and C. C. Kim. Effect of induced GVHD in leukemia patients relapsing after allogeneic bone marrow transplantation: single-center experience of 33 adult patients. *Bone Marrow Transplantation*, 27(9):999–1005, May 2001.

[91] 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, November 2015.

[92] E Mochiki, T Ohno, Y Kamiyama, R Aihara, N Haga, H Ojima, J Nakamura, H Ohsawa, T Nakabayashi, K Takeuchi, T Asao, and H Kuwano. Phase I/II study of S-1 combined with paclitaxel in patients with unresectable and/or recurrent advanced gastric cancer. *British Journal of Cancer*, 95(12):1642–1647, December 2006.

[93] Samaneh Moghaddam and Martin Ester. ILDA: interdependent LDA model for learning latent aspects and their ratings from online product reviews. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 665–674. ACM, 2011.

[94] Samaneh Moghaddam and Martin Ester. On the design of LDA models for aspect-based opinion mining. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 803–812. ACM, 2012.

[95] Mohamed M. Mostafa. More than words: Social networks text mining for consumer brand sentiments. *Expert Systems with Applications*, 40(10):4241–4251, 2013.

[96] S. M. Naim, A. P. Boedihardjo, and M. S. Hossain. A scalable model for tracking topical evolution in large document collections. In *IEEE Big Data*, pages 726–735, December 2017.

[97] Ramesh Nallapati, Ao Feng, Fuchun Peng, and James Allan. Event Threading Within News Topics. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pages 446–453, New York, NY, USA, 2004. ACM.

[98] Arman Khadjeh Nassirtoussi, Saeed Aghabozorgi, Teh Ying Wah, and David Chek Ling Ngo. Text mining for market prediction: A systematic review. *Expert Systems with Applications*, 41(16):7653–7670, 2014.

[99] National Cancer Institute. Adult Acute Myeloid Leukemia Treatment.

[100] and Blood Institute National Heart, Lung. Iron-Deficiency Anemia.

[101] National Heart, Lung, and Blood Institute. Pernicious Anemia.

[102] 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*, ASONAM '15, pages 1–8, New York, NY, USA, 2015. ACM.

[103] A. NithyaKalyani, S. Ushasukhanya, T. Y. J. Nagamalleswari, and S. Girija. Rating prediction using textual reviews. In *Journal of Physics: Conference Series*, volume 1000, page 012044. IOP Publishing, 2018.

[104] Stephen G. O'Brien, Franois Guilhot, Richard A. Larson, Insa Gathmann, Michele Baccarani, Francisco Cervantes, Jan J. Cornelissen, Thomas Fischer, Andreas Hochhaus, Timothy Hughes, Klaus Lechner, Johan L. Nielsen, Philippe Rousselot, Josy Reiffers, Giuseppe Saglio, John Shepherd, Bengt Simonsson, Alois Gratwohl,

John M. Goldman, Hagop Kantarjian, Kerry Taylor, Gregor Verhoef, Ann E. Bolton, Renaud Capdeville, and Brian J. Druker. Imatinib Compared with Interferon and Low-Dose Cytarabine for Newly Diagnosed Chronic-Phase Chronic Myeloid Leukemia. *N Engl J Med*, 348(11):994–1004, March 2003.

[105] Sungrae Park, Wonsung Lee, and Il-Chul Moon. Supervised dynamic topic models for associative topic extraction with a numerical time series. In *Proceedings of the 2015 Workshop on Topic Models: Post-Processing and Applications*, pages 49–54. ACM, 2015.

[106] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global Vectors for Word Representation. In *Proc. of Conf. on EMNLP*, pages 1532–1543. ACL, October 2014.

[107] Jerald P. Radich and Vivian Oehler. Monitoring chronic myelogenous leukemia in the age of tyrosine kinase inhibitors. *Journal of the National Comprehensive Cancer Network: JNCCN*, 5(5):497–504, May 2007.

[108] Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. Learning Causality for News Events Prediction. In *Proc. of Int. Conf. on WWW*, pages 909–918. ACM, 2012.

[109] Guy D. Rosin, Eytan Adar, and Kira Radinsky. Learning Word Relatedness over Time. July 2017.

[110] Marco Rospocher, Marieke van Erp, Piek Vossen, Antske Fokkens, Itziar Aldabe, German Rigau, Aitor Soroa, Thomas Ploeger, and Tessel Bogaard. Building Event-centric Knowledge Graphs from News. *Web Semant.*, 37(C):132–151, March 2016.

[111] Maja Rudolph and David Blei. Dynamic Embeddings for Language Evolution. pages 1003–1011. ACM Press, 2018.

[112] David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group, editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. MIT Press, Cambridge, MA, USA, 1986.

[113] Adam Sadilek, Henry A. Kautz, and Vincent Silenzio. Predicting disease transmission from geo-tagged micro-blog data. In *AAAI*, pages 136–142, 2012.

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

[115] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, January 1988.

[116] Shengtian Sang, Zhihao Yang, Zongyao Li, and Hongfei Lin. Supervised Learning Based Hypothesis Generation from Biomedical Literature, 2015.

[117] Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. Evaluation methods for unsupervised word embeddings. In *Proc. of Conf. on EMNLP*, pages 298–307. ACL, September 2015.

[118] Robert P. Schumaker and Hsinchun Chen. Textual analysis of stock market prediction using breaking financial news: The AZFin text system. *ACM Transactions on Information Systems*, 27(2):12, 2009.

[119] 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*, KDD '10, pages 623–632, New York, NY, USA, 2010. ACM.

[120] Dafna Shahaf, Carlos Guestrin, Eric Horvitz, and Jure Leskovec. Information Cartography. *Commun. ACM*, 58(11):62–73, October 2015.

[121] Amit Singhal, Gerard Salton, Mandar Mitra, and Chris Buckley. Document length normalization. *Information Processing & Management*, 32(5):619–633, September 1996.

[122] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, January 1972.

[123] 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*, WWW '13, pages 1237–1248, New York, NY, USA, 2013. ACM.

[124] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.

[125] Nina Tahmasebi, Lars Borin, and Adam Jatowt. Survey of Computational Approaches to Lexical Semantic Change. *arXiv:1811.06278 [cs]*, November 2018. arXiv: 1811.06278.

[126] Xuning Tang, Christopher C. Yang, and Mi Zhang. Who will be participating next?: predicting the participation of Dark Web community. In *Proceedings of the ACM SIGKDD Workshop on Intelligence and Security Informatics*, page 1. ACM, 2012.

[127] Xuri Tang. A State-of-the-Art of Semantic Change Computation. *Natural Language Engineering*, pages 1–28, June 2018. arXiv: 1801.09872.

[128] Xuri Tang, Weiguang Qu, and Xiaohe Chen. Semantic change computation: A successive approach. *World Wide Web*, 19(3):375–415, May 2016.

[129] P. D. Turney and P. Pantel. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37:141–188, February 2010.

[130] U.S. Food and Drug Administration. The Drug Development Process - Step 3: Clinical Research.

[131] U.S. Food and Drug Administration. FDA approves new combination treatment for acute myeloid leukemia, 2018.

[132] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv:1706.03762 [cs]*, June 2017. arXiv: 1706.03762.

[133] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and Tell: A Neural Image Caption Generator. pages 3156–3164, 2015.

[134] Chong Wang, David Blei, and David Heckerman. Continuous Time Dynamic Topic Models. In *Proc. of Uncertain Artif Intell*, pages 579–586, 2008.

[135] Wei Jie Wang, Di Chen, Ming Zuo Jiang, Bing Xu, Xiao Wei Li, Yi Chu, Yu Jie Zhang, Ren Mao, Jie Liang, and Dai Ming Fan. Downregulation of gasdermin D promotes gastric cancer proliferation by regulating cell cycle-related proteins. *Journal of Digestive Diseases*, 19(2):74–83, February 2018.

[136] Xiaolong Wang, Chengxiang Zhai, and Dan Roth. Understanding Evolution of Research Themes: A Probabilistic Generative Model for Citations. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 1115–1123, New York, NY, USA, 2013. ACM.

[137] Xuerui Wang and Andrew McCallum. Topics over Time: A non-Markov Continuous-time Model of Topical Trends. In *Proc. of ACM SIGKDD*, pages 424–433. ACM, 2006.

[138] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based LSTM for Aspect-level Sentiment Classification. In *Proceedings of the 2016 Conference on*

Empirical Methods in Natural Language Processing, pages 606–615, Austin, Texas, November 2016. Association for Computational Linguistics.

[139] Marc Weeber, Rein Vos, Henny Klein, Lolkje TW de Jong-van den Berg, Alan R. Aronson, and Grietje Molema. Generating hypotheses by discovering implicit associations in the literature: a case report of a search for new potential therapeutic uses for thalidomide. *Journal of the American Medical Informatics Association*, 10(3):252–259, 2003.

[140] Derry Tanti Wijaya and Reyyan Yeniterzi. Understanding Semantic Change of Words over Centuries. In *Proc. of DETECT*, pages 35–40. ACM, 2011.

[141] Chunzi Wu, Bin Wu, and Bai Wang. Event Evolution Model Based on Random Walk Model with Hot Topic Extraction. In *Advanced Data Mining and Applications*, pages 591–603. Springer, Cham, December 2016.

[142] T.-p Xu, X.-x Liu, R. Xia, L. Yin, R. Kong, W.-m Chen, M.-d Huang, and Y.-q Shu. SP1-induced upregulation of the long noncoding RNA *TINCR* regulates cell proliferation and apoptosis by affecting *KLF2* mRNA stability in gastric cancer. *Oncogene*, 34(45):5648–5661, November 2015.

[143] Guangxu Xun, Kishlay Jha, Vishrawas Gopalakrishnan, Yaliang Li, and Aidong Zhang. Generating Medical Hypotheses Based on Evolutionary Medical Concepts. In *Data Mining (ICDM), 2017 IEEE International Conference on*, pages 535–544. IEEE, 2017.

[144] K Yamaguchi, T Shimamura, I Hyodo, W Koizumi, T Doi, H Narahara, Y Komatsu, T Kato, S Saitoh, T Akiya, M Munakata, Y Miyata, Y Maeda, H Takiuchi, S Nakano, T Esaki, F Kinjo, and Y Sakata. Phase I/II study of docetaxel and S-1 in patients with advanced gastric cancer. *British Journal of Cancer*, 94(12):1803–1808, June 2006.

[145] Masamitsu Yanada, Jin Takeuchi, Isamu Sugiura, Hideki Akiyama, Noriko Usui, Fumiharu Yagasaki, Tohru Kobayashi, Yasunori Ueda, Makoto Takeuchi, Shuichi Miyawaki, Atsuo Maruta, Nobuhiko Emi, Yasushi Miyazaki, Shigeki Ohtake, Itsuro Jinnai, Keitaro Matsuo, Tomoki Naoe, and Ryuzo Ohno. High Complete Remission Rate and Promising Outcome by Combination of Imatinib and Chemotherapy for Newly Diagnosed BCR-ABLPositive Acute Lymphoblastic Leukemia: A Phase II Study by the Japan Adult Leukemia Study Group. *J Clin Oncol*, 24(3):460–466, January 2006.

[146] Y. Yang, J. G. Carbonell, R. D. Brown, T. Pierce, B. T. Archibald, and X. Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems and their Applications*, 14(4):32–43, July 1999.

[147] Yiming Yang, Tom Ault, Thomas Pierce, and Charles W. Lattimer. Improving Text Categorization Methods for Event Tracking. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, pages 65–72, New York, NY, USA, 2000. ACM.

[148] Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. Discovery of Evolving Semantics through Dynamic Word Embedding Learning. *arXiv preprint arXiv:1703.00607*, 2017.

[149] Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. Dynamic Word Embeddings for Evolving Semantic Discovery. In *Proc. of ACM WSDM*, pages 673–681, 2018.

[150] Dani Yogatama, Chong Wang, Bryan R. Routledge, Noah A. Smith, and Eric P. Xing. Dynamic Language Models for Streaming Text. *Assoc Comput Linguist*, 2(0):181–192, April 2014.

[151] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S. Dhillon. High-dimensional Time Series Prediction with Missing Values. *arXiv:1509.08333 [cs, stat]*, September 2015. arXiv: 1509.08333.

[152] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. Temporal Regularized Matrix Factorization for High-dimensional Time Series Prediction. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 847–855. Curran Associates, Inc., 2016.

[153] Shengkang Yu, Xi Li, Xueyi Zhao, Zhongfei Zhang, and Fei Wu. Tracking News Article Evolution by Dense Subgraph Learning. *Neurocomput.*, 168(C):1076–1084, November 2015.

[154] George Yule. *The study of language*. Cambridge University Press, Cambridge, UK, 6th edition, 2017.

[155] J. Zhang, D. D. Tong, M. Xue, Q. Y. Jiang, X. F. Wang, P. B. Yang, L. Ni, L. Y. Zhao, and C. Huang. FAM196b acts as oncogene and promotes proliferation of gastric cancer cells through AKT signaling pathway. *Cellular and Molecular Biology*, 63(9):18–23, September 2017.

[156] Xiaojing Zhang, Wei Wang, Peng Li, Xudong Wang, and Kan Ni. High TREM2 expression correlates with poor prognosis in gastric cancer. *Human Pathology*, 72:91–99, February 2018.

[157] Xue Zhang, Hauke Fuehres, and Peter A. Gloor. Predicting stock market indicators through twitter I hope it is not as bad as I fear. *Procedia-Social and Behavioral Sciences*, 26:55–62, 2011.

[158] Y. Zhang, A. Jatowt, S. S. Bhowmick, and K. Tanaka. The Past is Not a Foreign Country: Detecting Semantically Similar Terms across Time. *IEEE Transactions on Knowledge and Data Engineering*, 28(10):2793–2807, October 2016.

[159] Yin Zhang, Min Chen, Dijiang Huang, Di Wu, and Yong Li. iDoctor: Personalized and professionalized medical recommendations based on hybrid matrix factorization. *Future Generation Computer Systems*, 66:30–35, 2017.

[160] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-scale Bound-constrained Optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, December 1997.

[161] Huanhuan Zhu, Hangyong Zhao, Linjie Zhang, Jianmin Xu, Chunhua Zhu, Hui Zhao, and Guoqiang Lv. Dandelion root extract suppressed gastric cancer cells proliferation and migration through targeting lncRNA-CCAT1. *Biomedicine & Pharmacotherapy*, 93:1010–1017, September 2017.

[162] Xianshu Zhu and Tim Oates. Finding Story Chains in Newswire Articles Using Random Walks. *Information Systems Frontiers*, 16(5):753–769, November 2014.

Appendix A

Sample Neighborhoods of PubMed Using Temporal Word Embeddings

Table A.1: Nearest neighbors of the term *c release*.

1998	**2001**	**2004**	**2007**	**2010**	**2013**	**2016**
cytochrome	cytochrome	cytochrome	cytochrome	cytochrome	cytochrome	cytochrome

Table A.2: Nearest neighbors of the term *retrospective review*.

1998	**2001**	**2004**	**2007**	**2010**	**2013**	**2016**
prevalence	medical records	institution	records	consecutive patients	january	medical records

Table A.3: Nearest neighbors of the term *c mice*.

1998	**2001**	**2004**	**2007**	**2010**	**2013**	**2016**
balb	balb	balb	balb	balb	balb	balb

Table A.4: Nearest neighbors of the term *signal transducer*.

1998	**2001**	**2004**	**2007**	**2010**	**2013**	**2016**
activator	activator	transcription stat	activator	activator	activator	activator

Table A.5: Nearest neighbors of the term *catenin*.

1998	**2001**	**2004**	**2007**	**2010**	**2013**	**2016**
p expression	e cadherin	e cadherin	clinical studies			
e cadherin	mcf	p expression	combined use			
high levels	high levels	mcf	preoperative chemotherapy			
mcf	exposure	exposure	monitoring			
exposure	p expression	chemosensitivity	ccl			
mda mb cells	maspin	indomethacin	plasma levels			
breast cancer cell lines	mda mb cells	nurses	costs			
			normalization			
			su			
			gst			
			immunostaining			
			medications			

Table A.6: Nearest neighbors of the term *cohorts*.

1998	**2001**	**2004**	**2007**	**2010**	**2013**	**2016**
doses	recommended dose	mtd	dose levels	dose levels	dlt	cohort

Table A.7: Nearest neighbors of the term *dasatinib*.

1998	**2001**	**2004**	**2007**	**2010**	**2013**	**2016**
src	molecular mechanism	prostate cancer cells				
treatment strategies	treatment strategies	antiproliferative effect				
signaling pathways	invasion	molecular mechanism				
invasion	src	signaling pathways				
molecular mechanism	potential	invasion				
cancers	nude mice	src				
growth	cancers	nilotinib				
nude mice	signaling pathways	treatment strategies				
prostate cancer cells	mda mb	disease control				
potential	prostate cancer cells	sunitinib				
specimens	inflammatory bowel disease	option				
formulation	growth	cancers				
nb cells	inhibition	hgf				

Table A.8: Nearest neighbors of the term *objective response*.

1998	**2001**	**2004**	**2007**	**2010**	**2013**	**2016**
overall response rate	stable disease	stable disease	response rate	stable disease	primary end point	disease control rate

Table A.9: Nearest neighbors of the term *gender*.

1998	**2001**	**2004**	**2007**	**2010**	**2013**	**2016**
age	significant differences	age	age	age	age	sex

Table A.10: Nearest neighbors of the term *median age*.

1998	**2001**	**2004**	**2007**	**2010**	**2013**	**2016**
cycles	males	cycles	seven patients	cycles	march	courses

Appendix B

Sample Neighborhoods of NVD Using Temporal Word Embeddings

Table B.1: Nearest neighbors of the term *isc bind*.

1999	**2002**	**2005**	**2008**	**2011**	**2014**	**2017**
devices	glibc	different responses	sys	multiple integer	eml	themes
parent	libc			cache	pid parameter	
	long line					

Table B.2: Nearest neighbors of the term *squid*.

1999	**2002**	**2005**	**2008**	**2011**	**2014**	**2017**
april	stable	stable	relationship	setuid root	office web apps server sp	operating systems
		suse linux				root

Table B.3: Nearest neighbors of the term *snitz forums*.

1999	**2002**	**2005**	**2008**	**2011**	**2014**	**2017**
cross site scripting vulnerability	microsoft excel sp	simple	sun java system web server	post.php	sql injection vulnerability	sql injection vulnerability

Table B.4: Nearest neighbors of the term *aka zdi*.

1999	**2002**	**2005**	**2008**	**2011**	**2014**	**2017**
pointer	videolan vlc media player	cscug	cisco nx os	impact	crafted regular expression	long arguments
proxy server	arbitrary files				crafted bmp image	

Table B.5: Nearest neighbors of the term *siemens simatic*.

1999	**2002**	**2005**	**2008**	**2011**	**2014**	**2017**
long line	privileges	non root account	privileges	info		simatic
trojan horse shared library	web site	october	kernel mode	unspecified directory		android
						sp

Table B.6: Nearest neighbors of the term *vbscript*.

1999	**2002**	**2005**	**2008**	**2011**	**2014**	**2017**
temporary file	modules	loop			microsoft internet explorer	microsoft edge
write access	op	op			crafted web site	internet explorer
multiple interpre-tation error	events	service deadlock				
		arbitrary javascript				
		core dump				
		send				
		data				
		attachment				
		javascript				
		workcentre pro				
		aka bug id cscuq				

Table B.7: Nearest neighbors of the term *location header*.

1999	**2002**	**2005**	**2008**	**2011**	**2014**	**2017**
arbitrary headers		format string specifiers	data text html uri	data text html uri	data text html uri	data text html uri
crlf sequences		data text html uri	refresh header	web site	crafted length value	router
allocation				refresh header		
data text html uri				site		
arbitrary scripts						

Table B.8: Nearest neighbors of the term *service abend*.

1999	**2002**	**2005**	**2008**	**2011**	**2014**	**2017**
cacti	sgi irix	long password	crafted pdf file	udp	memory allocation	mediawiki

Table B.9: Nearest neighbors of the term *nexus x*.

1999	**2002**	**2005**	**2008**	**2011**	**2014**	**2017**
qualcomm sound driver	qualcomm sound driver	qualcomm sound driver	qualcomm sound driver	qualcomm sound driver	android devices	nexus

Table B.10: Nearest neighbors of the term *detail.asp*.

1999	**2002**	**2005**	**2008**	**2011**	**2014**	**2017**
sensitive information memory contents	fiyo cms	help	ios	forum	sql injection vulnerability	sql injection vulnerability
					environments	arbitrary sql commands
					postnuke	news.php

Table B.11: Nearest neighbors of the term *oracle flexcube enterprise limits*.

1999	**2002**	**2005**	**2008**	**2011**	**2014**	**2017**
java advanced management console	java advanced management console	successful attacks	java advanced management console	java advanced management console	successful attacks	oracle hospitality reporting
successful attacks	oracle hospitality reporting	unauthorized read access	oracle webcenter sites	oracle hospitality reporting	oracle hospitality reporting	oracle webcenter sites
						java advanced management console
						successful attacks

Table B.12: Nearest neighbors of the term *sql*.

1999	**2002**	**2005**	**2008**	**2011**	**2014**	**2017**
command injection	command injection	original disclosure	elements	proftpd	php code	
articles	sql server		inc	heap based buffer	sap	
original disclosure	articles		successful exploitation	udp port	hp storage data protector	
microsoft windows	original disclosure		action		stack based buffer	
			cms		win k.sys	
			register globals		trojan horse executable file	
			magic quotes gpc		management interface	
					xml external entity reference xxe	
					phpinfo function	

Table B.13: Nearest neighbors of the term *registry*.

1999	**2002**	**2005**	**2008**	**2011**	**2014**	**2017**
	file system			length	traffic	products
	cleartext			strings	middle attackers	weak permissions
	conversion			current working directory	man	ptrace
	reply			arbitrary scripts	proxy server	
	sip implementation				known fixed releases	
					client	
					connection	
					kerberos	
					open redirect vulnerability	

Appendix C

Sample Neighborhoods of New York Times Using Temporal Word Embeddings

Table C.1: Nearest neighbors of the term *annual rate*.

1990	**1994**	**1998**	**2002**	**2006**	**2010**	**2014**	**2018**
commerce department	commerce department	gross domestic product	commerce department	commerce department	consumer spending	commerce department	inflation

Table C.2: Nearest neighbors of the term *political science*.

1990	**1994**	**1998**	**2002**	**2006**	**2010**	**2014**	**2018**
professor	professor	professor	professor	professor		professor	
economics		dean	psychology				

Table C.3: Nearest neighbors of the term *fort lauderdale*.

1990	**1994**	**1998**	**2002**	**2006**	**2010**	**2014**	**2018**
fla.	fla.	fla.	jamaica	fla.	fla.	fla.	orlando

Table C.4: Nearest neighbors of the term *federal bureau*.

1990	**1994**	**1998**	**2002**	**2006**	**2010**	**2014**	**2018**
investigation	investigation	investigation	f.b.i.	f.b.i.	f.b.i.	f.b.i.	f.b.i.

Table C.5: Nearest neighbors of the term *mit*.

1990	**1994**	**1998**	**2002**	**2006**	**2010**	**2014**	**2018**
science	economics	physics	columbia university	physics	technology	economics	political science

Table C.6: Nearest neighbors of the term *chase*.

1990	**1994**	**1998**	**2002**	**2006**	**2010**	**2014**	**2018**
pricing	mom	walker	dozens				
walker	walker	pricing	fears				
american military	pricing	institutional investors	investment				
investment	adoption	youtube	opener				
			spill				
			aspect				
			austria				
			demonstrations				

Table C.7: Nearest neighbors of the term *maliki*.

1990	**1994**	**1998**	**2002**	**2006**	**2010**	**2014**	**2018**
faith	numbers	faith	faith				coalition
numbers	faith	numbers	numbers				sunnis
expenses	photo	world trade center	muslims				defense minister
referees	difference	muslims	members				falluja
							rubin
							comeback
							rejection
							general
							liability

Table C.8: Nearest neighbors of the term *interpublic group*.

1990	**1994**	**1998**	**2002**	**2006**	**2010**	**2014**	**2018**
johnson johnson	billings	billings	omnicom group	omnicom group	omnicom group	omnicom group	statistics
							foreign invest-ment
							half
							mid s.

Table C.9: Nearest neighbors of the term *berkeley*.

1990	**1994**	**1998**	**2002**	**2006**	**2010**	**2014**	**2018**
university	university	dean	expert	university		university	university
professor	institute	university	university	stanford		mit	california
						professor	

Table C.10: Nearest neighbors of the term *senator john mccain*.

1990	**1994**	**1998**	**2002**	**2006**	**2010**	**2014**	**2018**
senate floor	biden	mccain	mccain	mccain		mccain	south carolina
confirmation	majority leader	republican	judiciary committee			iraq war	

Appendix D

Predicted Evolution of Sample PubMed Terms

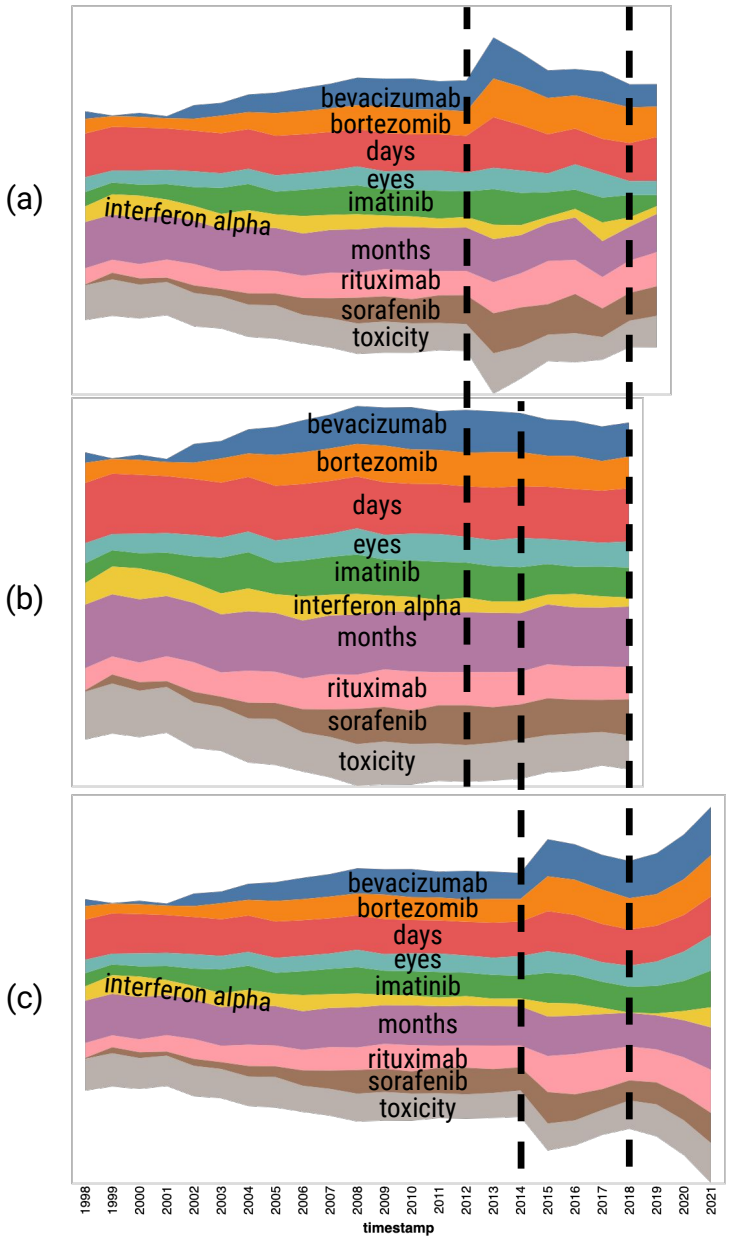


Figure D.1: Evolution of the neighborhood of the term *treatment* in the Pubmed dataset using: (a) the predict-next method with test timestamps from 2013 to 2019 (b) the temporal embedding method presented in Chapter 5, and (c) the predict-ith method with test timestamps from 2015 to 2021.

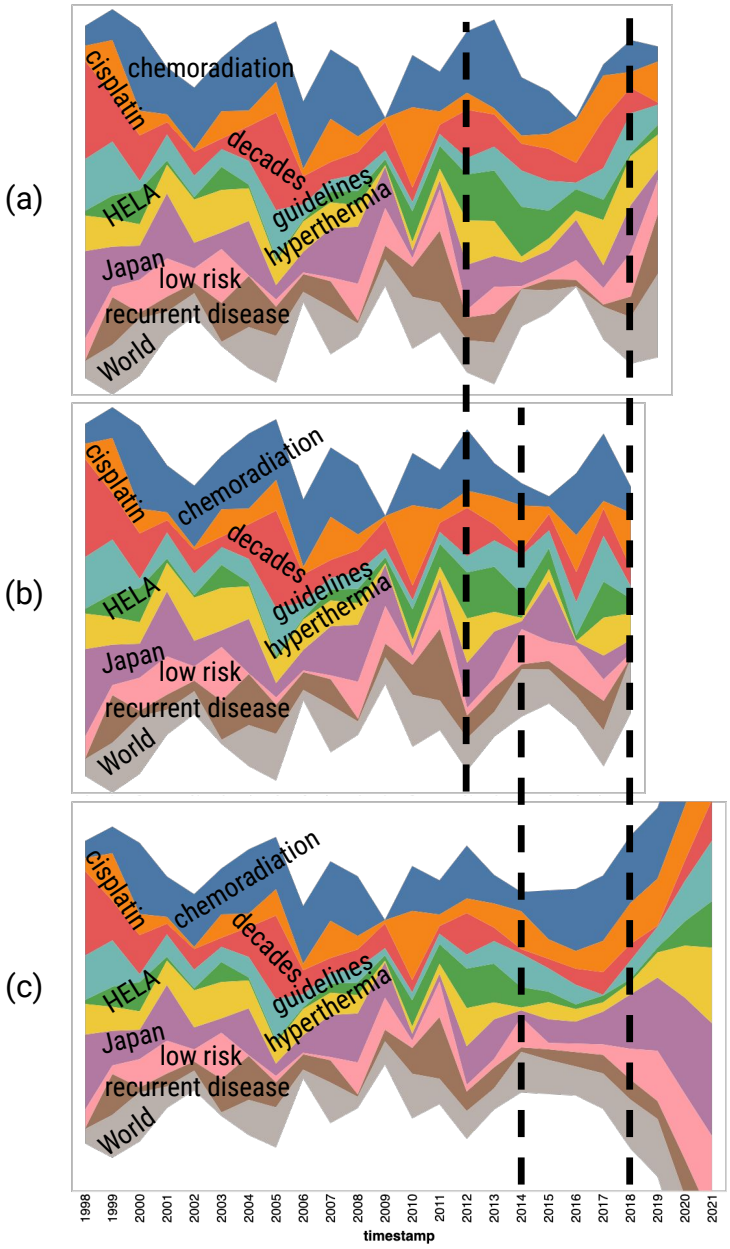


Figure D.2: Evolution of the neighborhood of the term *cervical cancer* in the Pubmed dataset using: (a) the predict-next method with test timestamps from 2013 to 2019 (b) the temporal embedding method presented in Chapter 5, and (c) the predict-ith method with test timestamps from 2015 to 2021.

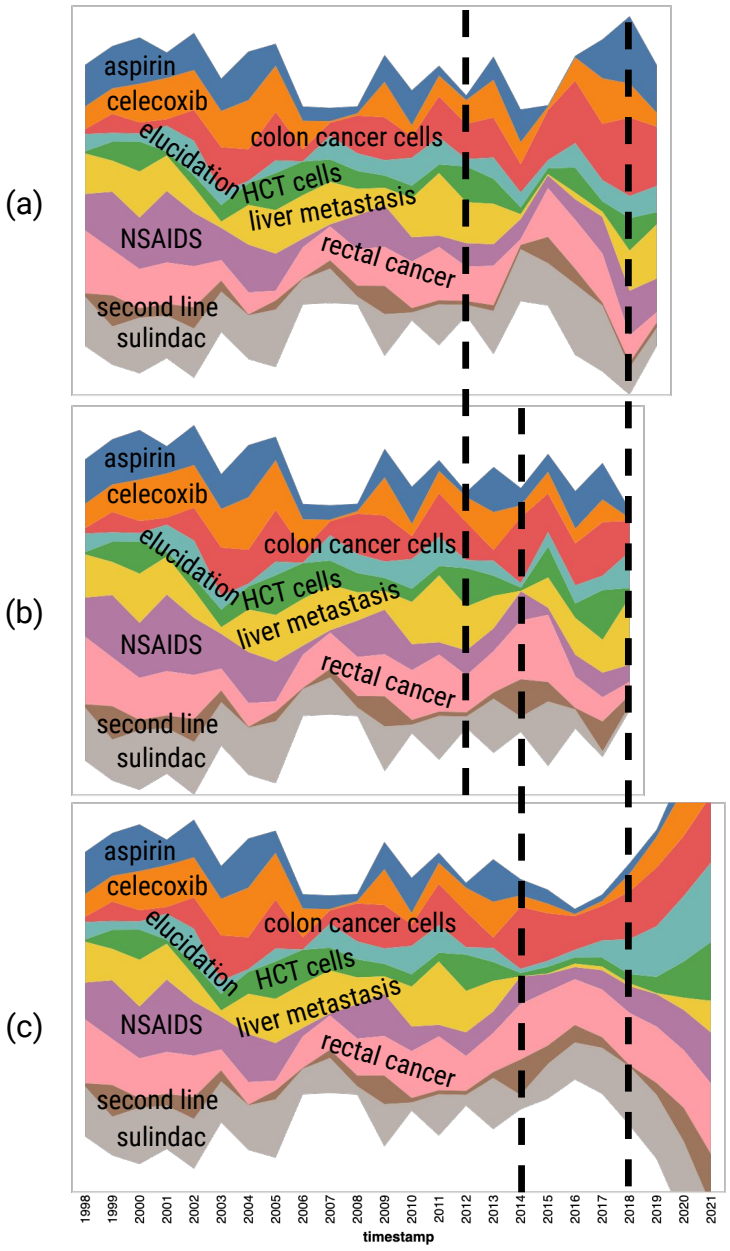


Figure D.3: Evolution of the neighborhood of the term *colon cancer* in the Pubmed dataset using: (a) the predict-next method with test timestamps from 2013 to 2019 (b) the temporal embedding method presented in Chapter 5, and (c) the predict-ith method with test timestamps from 2015 to 2021.

Appendix E

Predicted Neighborhoods of Sample PubMed Terms

Table E.1: Nearest neighbors for the test timestamps of the word *results* obtained using the temporal embeddings method.

2015	**2016**	**2017**	**2018**
study	study	study	study
addition	addition	addition	addition
effect	effect	effect	effect
treatment	treatment	treatment	treatment
effects	effects	effects	effects

Table E.2: Nearest neighbors for the test timestamps of the word *results* obtained using the predict-next method.

2015	**2016**	**2017**	**2018**	**2019**
effect	addition	study	study	study
study	study	addition	addition	addition
addition	effect	effect	effect	effect
cells	cells	expression	data	data
effects	expression	cells	cells	treatment

Table E.3: Nearest neighbors for the test timestamps of the word *results* obtained using the predict-*i*th method.

2015	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
effect	effect	effect	effect	effect	effect	effect
addition	addition	effects	effects	effects	effects	effects
study	study	study	study	study	study	study
effects	effects	addition	addition	addition	addition	cells
cells	cells	cells	treatment	treatment	vivo	vivo

Table E.4: Nearest neighbors for the test timestamps of the word *imatinib* obtained using the temporal embeddings method.

2015	**2016**	**2017**	**2018**
nilotinib	cml	nilotinib	chronic myeloid leukemia
cml patients	cml patients	cml	nilotinib
cml	dasatinib	chronic phase	cml
dasatinib	chronic myeloid leukemia	dasatinib	dasatinib
imatinib mesylate	nilotinib	tki	tki

Table E.5: Nearest neighbors for the test timestamps of the word *imatinib* obtained using the predict-next method.

2015	**2016**	**2017**	**2018**	**2019**
nilotinib	nilotinib	cml	cml	nilotinib
cml	cml	bcr abl	nilotinib	cml
cml patients	dasatinib	dasatinib	chronic myeloid leukemia	chronic myeloid leukemia
dasatinib	cml patients	nilotinib	bcr abl	dasatinib
bcr abl	tki	tki	chronic phase	cml patients

Table E.6: Nearest neighbors for the test timestamps of the word *imatinib* obtained using the predict-*i*th method.

2015	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
cml	nilotinib	nilotinib	nilotinib	nilotinib	nilotinib	nilotinib
nilotinib	cml	cml	cml	cml	cml	dasatinib
chronic myeloid leukemia	dasatinib	dasatinib	dasatinib	dasatinib	dasatinib	impairment
dasatinib	chronic myeloid leukemia	chronic myeloid leukemia	cml patients	cml patients	cml patients	cml patients
chronic myelogenous leukemia	cml patients	cml patients	chronic myeloid leukemia	chronic myeloid leukemia	tki	bcr abl

Table E.7: Nearest neighbors for the test timestamps of the word *vomiting* obtained using the temporal embeddings method.

2015	**2016**	**2017**	**2018**
nausea	nausea	nausea	nausea
fatigue	fatigue	fatigue	fatigue
diarrhea	anorexia	diarrhea	diarrhea
anorexia	diarrhea	diarrhoea	diarrhoea
grade	diarrhoea	anorexia	stomatitis

Table E.8: Nearest neighbors for the test timestamps of the word *vomiting* obtained using the predict-next method.

2015	**2016**	**2017**	**2018**	**2019**
nausea	nausea	nausea	nausea	nausea
fatigue	fatigue	anorexia	diarrhea	diarrhoea
anorexia	anorexia	fatigue	fatigue	anorexia
diarrhea	diarrhea	diarrhoea	anorexia	fatigue
abdominal pain	diarrhoea	diarrhea	headache	cinv

Table E.9: Nearest neighbors for the test timestamps of the word *vomiting* obtained using the predict-ith method.

2015	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
nausea	nausea	nausea	nausea	nausea	nausea	nausea
fatigue	fatigue	fatigue	fatigue	fatigue	fatigue	fatigue
anorexia	anorexia	anorexia	anorexia	anorexia	headache	dyspnea
diarrhea	diarrhea	diarrhea	diarrhea	headache	dyspnea	headache
neutropenia	anemia	headache	headache	diarrhea	anorexia	abdominal pain

Table E.10: Nearest neighbors for the test timestamps of the word *isolates* obtained using the temporal embeddings method.

2015	**2016**	**2017**	**2018**
ciprofloxacin	strains	ciprofloxacin	ciprofloxacin
strains	mic	mic	strains
mic	ciprofloxacin	strains	mic
escherichia coli	e coli	mics	levofloxacin
enrofloxacin	mics	e coli	mics

Table E.11: Nearest neighbors for the test timestamps of the word *isolates* obtained using the predict-next method.

2015	**2016**	**2017**	**2018**	**2019**
ciprofloxacin	ciprofloxacin	strains	mic	ciprofloxacin
strains	strains	ciprofloxacin	strains	mic
mic	mic	mic	ciprofloxacin	strains
mics	pseudomonas aeruginosa	fluoroquinolones	bacteria	bacteria
pseudomonas aeruginosa	enrofloxacin	bacteria	e coli	antibiotics

Table E.12: Nearest neighbors for the test timestamps of the word *isolates* obtained using the predict-ith method.

2015	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
ciprofloxacin	ciprofloxacin	ciprofloxacin	ciprofloxacin	ciprofloxacin	ciprofloxacin	ciprofloxacin
strains	strains	strains	mic	mic	mic	mic
mic	mic	mic	strains	e coli	e coli	e coli
mics	mics	mics	e coli	strains	quinolones	quinolones
antibiotics	antibiotics	antibiotics	mics	mics	gatifloxacin	sti

Table E.13: Nearest neighbors for the test timestamps of the word *catalase* obtained using the temporal embeddings method.

2015	**2016**	**2017**	**2018**
superoxide dismutase	superoxide dismutase	superoxide dismutase	superoxide dismutase
sod	sod	cat	sod
cat	cat	sod	glutathione s
glutathione s	lipid peroxidation	glutathione s	cat
glutathione	glutathione	activities	gst

Table E.14: Nearest neighbors for the test timestamps of the word *catalase* obtained using the predict-next method.

2015	**2016**	**2017**	**2018**	**2019**
superoxide dismutase	superoxide dismutase	superoxide dismutase	superoxide dismutase	superoxide dismutase
cat	sod	sod	sod	sod
sod	cat	glutathione	cat	activities
alt	glutathione	cat	glutathione	cat
glutathione	activities	mda	activities	glutathione s

Table E.15: Nearest neighbors for the test timestamps of the word *catalase* obtained using the predict-ith method.

2015	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
superoxide dismutase	superoxide dismutase	superoxide dismutase	superoxide dismutase	superoxide dismutase	superoxide dismutase	superoxide dismutase
cat	cat	cat	cat	cat	sod	sod
sod	sod	sod	sod	sod	cat	self
lipid peroxidation	activities	restoration	restoration	placebo group	erbb	erbb
activities	gsh	mda	mda	restoration	improved survival	cat

Table E.16: Nearest neighbors for the test timestamps of the word *cat* obtained using the temporal embeddings method.

2015	**2016**	**2017**	**2018**
sod	sod	sod	sod
catalase	catalase	catalase	superoxide dismutase
glutathione s	superoxide dismutase	superoxide dismutase	catalase
superoxide dismutase	mda	glutathione s	gst
gst	activities	gst	mda

Table E.17: Nearest neighbors for the test timestamps of the word *cat* obtained using the predict-next method.

2015	**2016**	**2017**	**2018**	**2019**
sod	sod	sod	sod	sod
catalase	superoxide dismutase	catalase	catalase	catalase
mda	catalase	superoxide dismutase	superoxide dismutase	superoxide dismutase
superoxide dismutase	gst	mda	gst	gst
gsh	glutathione s	gst	glutathione	gsh

Table E.18: Nearest neighbors for the test timestamps of the word *cat* obtained using the predict-ith method.

2015	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
sod	sod	sod	sod	sod	sod	sod
catalase	mda	mda	mda	mda	mda	mda
superoxide dismutase	catalase	superoxide dismutase	catalase	measures	measures	measures
mda	superoxide dismutase	catalase	superoxide dismutase	catalase	tamoxifen treatment	tamoxifen treatment
glutathione s	gsh	increases	measures	extent	gsh	gsh

Table E.19: Nearest neighbors for the test timestamps of the word *ranibizumab* obtained using the temporal embeddings method.

2015	**2016**	**2017**	**2018**
eyes	visual acuity	eyes	visual acuity
visual acuity	eyes	visual acuity	eyes
bcva	iop	letters	intravitreal injection
intravitreal injection	intravitreal injection	intravitreal injection	cnv
injections	macular edema	macular edema	macular edema

Table E.20: Nearest neighbors for the test timestamps of the word *ranibizumab* obtained using the predict-next method.

2015	**2016**	**2017**	**2018**	**2019**
intravitreal injection	visual acuity	visual acuity	visual acuity	intravitreal injection
eyes	eyes	eyes	eyes	visual acuity
visual acuity	intravitreal injection	macular edema	letters	eyes
macular edema	bcva	intravitreal injection	intravitreal injection	cnv
intravitreal bevacizumab	macular edema	letters	macular edema	intravitreal bevacizumab

Table E.21: Nearest neighbors for the test timestamps of the word *ranibizumab* obtained using the predict-ith method.

2015	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
visual acuity	intravitreal injection	intravitreal injection	intravitreal injection	intravitreal injection	intravitreal injection	macular edema
eyes	visual acuity	macular edema	macular edema	macular edema	macular edema	visual acuity
intravitreal injection	macular edema	visual acuity	visual acuity	visual acuity	visual acuity	intravitreal injection
macular edema	eyes	eyes	eyes	eyes	medications	medications
iop	intravitreal bevacizumab	trabeculectomy	eye	iop	iop	mitomycin c mmc

Appendix F

Sample Predicted Neighborhoods of NVD Sample Terms

Table F.1: Nearest neighbors for the test timestamps of the word *spoof servers* obtained using the temporal embeddings method.

2015	**2016**	**2017**	**2018**
middle attackers	middle attackers	middle attackers	middle attackers
crafted certificate	crafted certificate	crafted certificate	crafted certificate
man	man	man	man
arbitrary valid certificate	arbitrary ssl servers	ssl servers	ssl servers
ssl servers	ssl servers	arbitrary valid certificate	arbitrary ssl servers

Table F.2: Nearest neighbors for the test timestamps of the word *spoof servers* obtained using the predict-next method.

2015	**2016**	**2017**	**2018**	**2019**
crafted certificate	middle attackers	middle attackers	middle attackers	middle attackers
middle attackers	man	crafted certificate	man	crafted certificate
man	crafted certificate	man	crafted certificate	man
ssl servers	ssl servers	arbitrary ssl servers	ssl servers	ssl servers
arbitrary ssl servers	arbitrary ssl servers	ssl servers	arbitrary valid certificate	arbitrary ssl servers

Table F.3: Nearest neighbors for the test timestamps of the word *spoof servers* obtained using the predict-ith method.

2015	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
crafted certificate	crafted certificate	crafted certificate	crafted certificate	crafted certificate	crafted certificate	long command line arguments
middle attackers	middle attackers	man	man	man	long command line arguments	crafted certificate
man	man	middle attackers	arbitrary valid certificate	long command line arguments	different responses	different responses
arbitrary ssl servers	ssl servers	ssl servers	middle attackers	different responses	cscvc	message
ssl servers	arbitrary ssl servers	arbitrary valid certificate	ssl servers	cscvc	wraparound	eml

Table F.4: Nearest neighbors for the test timestamps of the word *wireshark* obtained using the temporal embeddings method.

2015	**2016**	**2017**	**2018**
epan dissectors	epan dissectors	epan dissectors	epan dissectors
dissect	malformed capture file	malformed capture file	malformed capture file
malformed capture file	dissect	dissector	dissector
ieee	ieee	infinite loop	dissect
dissector	dissector	dissect	local remote attackers

Table F.5: Nearest neighbors for the test timestamps of the word *wireshark* obtained using the predict-next method.

2015	**2016**	**2017**	**2018**	**2019**
epan dissectors	epan dissectors	epan dissectors	epan dissectors	epan dissectors
dissect	dissect	malformed capture file	malformed capture file	malformed capture file
dissector	malformed capture file	dissect	dissector	dissector
ieee	ieee	dissector	dissect	dissect
local remote attackers	dissector	ieee	infinite loop	local remote attackers

Table F.6: Nearest neighbors for the test timestamps of the word *wireshark* obtained using the predict-ith method.

2015	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
epan dissectors	epan dissectors	epan dissectors	epan dissectors	epan dissectors	epan dissectors	epan dissectors
dissector	dissector	malformed capture file	malformed capture file	malformed capture file	malformed capture file	malformed capture file
dissect	malformed capture file	dissector	dissector	dissector	dissector	additional privileges
ieee	dissect	ieee	proxy server	proxy server	proxy server	dissector
local remote attackers	ieee	dissect	dissect	double free vulnerability	additional privileges	proxy server

Table F.7: Nearest neighbors for the test timestamps of the word *adobe acrobat reader* obtained using the temporal embeddings method.

2015	**2016**	**2017**	**2018**
successful exploitation	image conversion engine	exploitable memory corruption vulnerability	exploitable memory corruption vulnerability
exploitable memory corruption vulnerability	exploitable memory corruption vulnerability	image conversion engine	image conversion engine
image conversion engine	successful exploitation	exploitable heap overflow vulnerability	successful exploitation
exploitable heap overflow vulnerability	exploitable heap overflow vulnerability	successful exploitation	exploitable use
arbitrary code execution	exploitable use	exploitable use	arbitrary code execution

Table F.8: Nearest neighbors for the test timestamps of the word *adobe acrobat reader* obtained using the predict-next method.

2015	**2016**	**2017**	**2018**	**2019**
image conversion engine	image conversion engine	image conversion engine	exploitable memory corruption vulnerability	image conversion engine
exploitable heap overflow vulnerability	exploitable memory corruption vulnerability	exploitable memory corruption vulnerability	exploitable heap overflow vulnerability	exploitable memory corruption vulnerability
successful exploitation	exploitable heap overflow vulnerability	successful exploitation	image conversion engine	successful exploitation
exploitable memory corruption vulnerability	successful exploitation	exploitable heap overflow vulnerability	successful exploitation	exploitable use
exploitable use	arbitrary code execution	arbitrary code execution	adobe flash player versions	exploitable heap overflow vulnerability

Table F.9: Nearest neighbors for the test timestamps of the word *adobe acrobat reader* obtained using the predict-*i*th method.

2015	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
image conversion engine	image conversion engine	image conversion engine	image conversion engine	image conversion engine	image conversion engine	image conversion engine
exploitable heap overflow vulnerability	successful exploitation	successful exploitation	successful exploitation	successful exploitation	php file	php file
successful exploitation	exploitable heap overflow vulnerability	exploitable memory corruption vulnerability	exploitable memory corruption vulnerability	exploitable memory corruption vulnerability	directory traversal sequences	malicious file
exploitable memory corruption vulnerability	exploitable memory corruption vulnerability	exploitable heap overflow vulnerability	exploitable heap overflow vulnerability	php file	malicious file	multiple stack based buffer
exploitable use	adobe flash player versions	arbitrary code execution	adobe flash player versions	executable file	executable file	directory traversal sequences

Table F.10: Nearest neighbors for the test timestamps of the word *image conversion engine* obtained using the temporal embeddings method.

2015	**2016**	**2017**	**2018**
exploitable memory corruption vulnerability	adobe acrobat reader	adobe acrobat reader	exploitable memory corruption vulnerability
successful exploitation	exploitable memory corruption vulnerability	exploitable memory corruption vulnerability	adobe acrobat reader
exploitable heap overflow vulnerability	successful exploitation	successful exploitation	successful exploitation
adobe acrobat reader	exploitable heap overflow vulnerability	arbitrary code execution	arbitrary code execution
arbitrary code execution	arbitrary code execution	exploitable heap overflow vulnerability	exploitable heap overflow vulnerability

Table F.11: Nearest neighbors for the test timestamps of the word *image conversion engine* obtained using the predict-next method.

2015	**2016**	**2017**	**2018**	**2019**
adobe acrobat reader	exploitable memory corruption vulnerability	exploitable memory corruption vulnerability	exploitable memory corruption vulnerability	adobe acrobat reader
exploitable memory corruption vulnerability	adobe acrobat reader	adobe acrobat reader	adobe acrobat reader	exploitable memory corruption vulnerability
exploitable heap overflow vulnerability	successful exploitation	successful exploitation	successful exploitation	successful exploitation
successful exploitation	exploitable heap overflow vulnerability	exploitable heap overflow vulnerability	exploitable heap overflow vulnerability	exploitable heap overflow vulnerability
arbitrary code execution	arbitrary code execution	arbitrary code execution	arbitrary code execution	arbitrary code execution

Table F.12: Nearest neighbors for the test timestamps of the word *image conversion engine* obtained using the predict-ith method.

2015	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
adobe acrobat reader	adobe acrobat reader	adobe acrobat reader	adobe acrobat reader	adobe acrobat reader	adobe acrobat reader	adobe acrobat reader
successful exploitation	successful exploitation	exploitable memory corruption vulnerability	exploitable memory corruption vulnerability	exploitable memory corruption vulnerability	exploitable memory corruption vulnerability	dot dot sequences
exploitable memory corruption vulnerability	exploitable memory corruption vulnerability	successful exploitation	successful exploitation	successful exploitation	executable file	cpu consumption
exploitable heap overflow vulnerability	exploitable heap overflow vulnerability	exploitable heap overflow vulnerability	exploitable heap overflow vulnerability	exploitable heap overflow vulnerability	successful exploitation	executable file
arbitrary code execution	arbitrary code execution	arbitrary code execution	adobe flash player versions	adobe flash player versions	dot dot sequences	cisco ios xr

Table F.13: Nearest neighbors for the test timestamps of the word *khobe attack* obtained using the temporal embeddings method.

2015	**2016**	**2017**	**2018**
aka argument switch attack	aka argument switch attack	aka argument switch attack	aka argument switch attack
hook handler execution	hook handler execution	hook handler execution	hook handler execution
certain user space memory changes	certain user space memory changes	certain user space memory changes	certain user space memory changes
parties	signature based malware detection	signature based malware detection	signature based malware detection
signature based malware detection	parties	parties	handler

Table F.14: Nearest neighbors for the test timestamps of the word *khobe attack* obtained using the predict-next method.

2015	**2016**	**2017**	**2018**	**2019**
aka argument switch attack	aka argument switch attack	aka argument switch attack	aka argument switch attack	aka argument switch attack
hook handler execution	hook handler execution	hook handler execution	hook handler execution	hook handler execution
certain user space memory changes	certain user space memory changes	certain user space memory changes	certain user space memory changes	certain user space memory changes
signature based malware detection	parties	parties	parties	signature based malware detection
parties	signature based malware detection	operations	exposure	handler

Table F.15: Nearest neighbors for the test timestamps of the word *khobe attack* obtained using the predict-ith method.

2015	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
aka argument switch attack	aka argument switch attack	aka argument switch attack	aka argument switch attack	aka argument switch attack	aka argument switch attack	aka argument switch attack
hook handler execution	hook handler execution	hook handler execution	hook handler execution	hook handler execution	certain user space memory changes	certain user space memory changes
certain user space memory changes	certain user space memory changes	certain user space memory changes	certain user space memory changes	certain user space memory changes	hook handler execution	start parameter
signature based malware detection	signature based malware detection	bind	cisco	cisco	cisco	drivers
parties	parties	cisco	unauthorized actions	unauthorized actions	start parameter	filesystem

Table F.16: Nearest neighbors for the test timestamps of the word *acrobat pro* obtained using the temporal embeddings method.

2015	**2016**	**2017**	**2018**
extended users	product jsp product platform windows	extended users	extended users
product jsp product platform windows	extended users	product jsp product platform windows	product jsp product platform windows
appropriate update	appropriate update	appropriate update	appropriate update
www adobe com support	www adobe com support	www adobe com support	www adobe com support
macintosh	macintosh	acrobat pro users	acrobat pro users

Table F.17: Nearest neighbors for the test timestamps of the word *acrobat pro* obtained using the predict-next method.

2015	**2016**	**2017**	**2018**	**2019**
extended users	extended users	extended users	extended users	extended users
product jsp product platform windows	product jsp product platform windows	product jsp product platform windows	product jsp product platform windows	product jsp product platform windows
appropriate update	appropriate update	appropriate update	appropriate update	appropriate update
www adobe com support	www adobe com support	www adobe com support	www adobe com support	www adobe com support
macintosh	acrobat pro users	macintosh	product jsp product platform windows acrobat d users	macintosh

Table F.18: Nearest neighbors for the test timestamps of the word *acrobat pro* obtained using the predict-*i*th method.

2015	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
extended users	extended users	extended users	product jsp product platform windows	product jsp product platform windows	product jsp product platform windows	product jsp product platform windows
product jsp product platform windows	product jsp product platform windows	product jsp product platform windows	extended users	extended users	extended users	product jsp product platform macintosh
appropriate update	appropriate update	macintosh	macintosh	appropriate update	product jsp product platform macintosh	imap service
macintosh	macintosh	appropriate update	appropriate update	macintosh	appropriate update	silverstripe
www adobe com support	www adobe com support	www adobe com support	www adobe com support	product jsp product platform macintosh	silverstripe	extended users

Table F.19: Nearest neighbors for the test timestamps of the word *signature based malware detection* obtained using the temporal embeddings method.

2015	**2016**	**2017**	**2018**
certain user space memory changes	certain user space memory changes	certain user space memory changes	certain user space memory changes
hook handler execution	hook handler execution	hook handler execution	hook handler execution
dangerous code	handler	dangerous code	dangerous code
handler	dangerous code	aka argument switch attack	aka argument switch attack
aka argument switch attack	aka argument switch attack	handler	handler

Table F.20: Nearest neighbors for the test timestamps of the word *signature based malware detection* obtained using the predict-next method.

2015	**2016**	**2017**	**2018**	**2019**
certain user space memory changes	certain user space memory changes	certain user space memory changes	certain user space memory changes	certain user space memory changes
hook handler execution	hook handler execution	handler	dangerous code	hook handler execution
handler	handler	hook handler execution	hook handler execution	dangerous code
aka argument switch attack	dangerous code	dangerous code	handler	aka argument switch attack
dangerous code	aka argument switch attack	aka argument switch attack	kernel mode hook handlers	handler

Table F.21: Nearest neighbors for the test timestamps of the word *signature based malware detection* obtained using the predict-ith method.

2015	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
certain user space memory changes	certain user space memory changes	certain user space memory changes	certain user space memory changes	certain user space memory changes	certain user space memory changes	certain user space memory changes
hook handler execution	hook handler execution	handler	handler	sun solaris	attribute	attribute
handler	handler	hook handler execution	hook handler execution	handler	sun solaris	sun solaris
aka argument switch attack	dangerous code	aka argument switch attack	aka argument switch attack	attribute	mishandles	image php
dangerous code	aka argument switch attack	dangerous code	sun solaris	mishandles	sensitive information memory contents	ibm websphere mq

Appendix G

Predicted Neighborhoods of NYT Sample Terms

Table G.1: Nearest neighbors for the test timestamps of the word *europe* obtained using the temporal embeddings method.

2014	**2015**	**2016**	**2017**	**2018**
germany	germany	germany	germany	germany
france	european union	european union	france	france
european union	france	britain	european union	european union
britain	britain	france	britain	britain
countries	italy	italy	countries	countries

Table G.2: Nearest neighbors for the test timestamps of the word *europe* obtained using the predict-next method.

2014	**2015**	**2016**	**2017**	**2018**	**2019**
germany	germany	germany	germany	germany	european union
european union	european union	european union	european union	european union	france
france	france	france	france	france	germany
countries	britain	britain	britain	britain	britain
britain	countries	countries	greece	countries	countries

Table G.3: Nearest neighbors for the test timestamps of the word *europe* obtained using the predict-ith method.

2014	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
germany	germany	germany	european union	european union	european union	recession	recession
european union	european union	european union	spain	russia	russia	growth	growth
france	france	spain	germany	britain	recession	european union	european union
countries	countries	france	france	germany	germany	change	continent
britain	ireland	paris	britain	ukraine	growth	germany	employment

Table G.4: Nearest neighbors for the test timestamps of the word *treatment* obtained using the temporal embeddings method.

2014	**2015**	**2016**	**2017**	**2018**
doctors	doctors	doctors	doctors	patients
patient	patients	patients	patients	doctors
patients	patient	patient	patient	patient
hospital	h i v	chemotherapy	cancer	cancer
infection	drugs	h i v	medication	treatments

Table G.5: Nearest neighbors for the test timestamps of the word *treatment* obtained using the predict-next method.

2014	**2015**	**2016**	**2017**	**2018**	**2019**
doctors	patient	doctors	doctors	patients	patients
patients	patients	patients	patients	doctors	patient
patient	doctors	h i v	patient	patient	doctors
disease	hospital	patient	chemotherapy	cancer	cancer
infection	drug	drugs	drug	drug	treatments

Table G.6: Nearest neighbors for the test timestamps of the word *treatment* obtained using the predict-ith method.

2014	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
doctors	patients	patients	patients	patients	patient	patient	patient
patients	doctors	doctors	doctors	doctors	patients	patients	patients
patient	patient	patient	patient	patient	doctors	doctors	doctors
infection	h i v	h i v	drug	drug	diabetes	diabetes	infection
disease	disease	diabetes	h i v	diabetes	drug	infection	diabetes

Table G.7: Nearest neighbors for the test timestamps of the word *loans* obtained using the temporal embeddings method.

2014	**2015**	**2016**	**2017**	**2018**
borrowers	lenders	lenders	borrowers	borrowers
banks	banks	debt	banks	banks
debt	debt	banks	mortgages	lenders
lenders	borrowers	borrowers	debt	mortgages
mortgages	funds	mortgages	payments	dollars

Table G.8: Nearest neighbors for the test timestamps of the word *loans* obtained using the predict-next method.

2014	**2015**	**2016**	**2017**	**2018**	**2019**
banks	borrowers	lenders	lenders	borrowers	borrowers
lenders	banks	banks	banks	banks	lenders
mortgages	debt	borrowers	borrowers	mortgages	banks
debt	lenders	debt	debt	payments	payments
borrowers	loan	funds	mortgages	debt	dollars

Table G.9: Nearest neighbors for the test timestamps of the word *loans* obtained using the predict-ith method.

2014	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
lenders	lenders	lenders	lenders	lenders	lenders	lenders	lenders
banks	banks	mortgages	mortgages	mortgages	mortgages	mortgages	mortgages
mortgages	mortgages	banks	borrowers	borrowers	borrowers	borrowers	borrowers
borrowers	borrowers	borrowers	banks	banks	banks	banks	freddie mac
debt	bank	assets	debt	debt	homeowner	freddie mac	banks

Table G.10: Nearest neighbors for the test timestamps of the word *south korea* obtained using the temporal embeddings method.

2014	**2015**	**2016**	**2017**	**2018**
japan	japan	japan	japan	japan
seoul	seoul	north korea	seoul	seoul
north korea	north korea	seoul	north korea	north korea
korea	korea	pyongyang	pyongyang	kim
kim	china	nuclear weapons	kim	pyongyang

Table G.11: Nearest neighbors for the test timestamps of the word *south korea* obtained using the predict-next method.

2014	**2015**	**2016**	**2017**	**2018**	**2019**
north korea	japan	japan	japan	japan	japan
seoul	seoul	seoul	north korea	seoul	north korea
japan	north korea	china	seoul	north korea	seoul
kim	china	north korea	china	kim	kim
korea	pyongyang	korea	pyongyang	pyongyang	north

Table G.12: Nearest neighbors for the test timestamps of the word *south korea* obtained using the predict-ith method.

2014	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
north korea	north korea	seoul	north korea	north korea	north korea	japan	japan
japan	seoul	north korea	seoul	japan	japan	north korea	seoul
seoul	japan	china	pyongyang	pyongyang	pyongyang	seoul	pyongyang
chinese	china	japan	japan	seoul	chinese	pyongyang	north korea
china	kim	pyongyang	china	china	philippines	philippines	philippines

Table G.13: Nearest neighbors for the test timestamps of the word *airline* obtained using the temporal embeddings method.

2014	**2015**	**2016**	**2017**	**2018**
airlines	flights	flights	flights	airlines
flights	airlines	american airlines	airlines	passengers
passengers	passengers	airlines	american airlines	flights
flight	flight	passengers	passengers	flight
united airlines	american airlines	delta	flight	delta

Table G.14: Nearest neighbors for the test timestamps of the word *airline* obtained using the predict-next method.

2014	**2015**	**2016**	**2017**	**2018**	**2019**
airlines	airlines	airlines	flights	airlines	airlines
flights	flights	flights	airlines	flights	passengers
passengers	passengers	passengers	passengers	american airlines	flights
flight	flight	flight	american airlines	passengers	flight
airways	united airlines	american airlines	flight	united	delta

Table G.15: Nearest neighbors for the test timestamps of the word *airline* obtained using the predict-ith method.

2014	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
airlines	airlines	airlines	airlines	airlines	airlines	airlines	airlines
flights	flights	flights	flights	flights	passengers	passengers	flights
passengers	passengers	passengers	passengers	passengers	flights	flights	passengers
delta	delta	delta	carriers	airport	airport	airport	airport
flight	flight	travelers	british airways	british airways	british airways	british airways	british airways

Table G.16: Nearest neighbors for the test timestamps of the word *central bank* obtained using the temporal embeddings method.

2014	**2015**	**2016**	**2017**	**2018**
european central bank	european central bank	european central bank	interest rates	federal reserve
interest rates	interest rates	monetary policy	fed	monetary policy
fed	monetary policy	fed	monetary policy	interest rates
monetary policy	fed	interest rates	european central bank	fed
lending	inflation	federal reserve	federal reserve	inflation

Table G.17: Nearest neighbors for the test timestamps of the word *central bank* obtained using the predict-next method.

2014	**2015**	**2016**	**2017**	**2018**	**2019**
fed	european central bank	european central bank	european central bank	fed	monetary policy
monetary policy	interest rates	monetary policy	interest rates	interest rates	interest rates
european central bank	monetary policy	interest rates	monetary policy	monetary policy	federal reserve
interest rates	fed	fed	fed	european central bank	fed
inflation	lending	inflation	federal reserve	federal reserve	inflation

Table G.18: Nearest neighbors for the test timestamps of the word *central bank* obtained using the predict-ith method.

2014	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
fed	european central bank	european central bank	fed	fed	fed	fed	fed
monetary policy	fed	interest rates	interest rates	monetary policy	monetary policy	monetary policy	interest rates
european central bank	interest rates	fed	financial markets	federal reserve	federal reserve	bernanke	monetary policy
interest rates	monetary policy	monetary policy	monetary policy	interest rates	interest rates	interest rates	federal reserve
federal reserve	federal reserve	financial markets	federal reserve	financial markets	bernanke	federal reserve	bernanke

Table G.19: Nearest neighbors for the test timestamps of the word *global warming* obtained using the temporal embeddings method.

2014	**2015**	**2016**	**2017**	**2018**
climate change	climate change	climate change	climate change	climate change
emissions	carbon dioxide	carbon dioxide	carbon dioxide	carbon dioxide
carbon dioxide	emissions	emissions	emissions	emissions
climate	climate	planet	planet	climate
forests	pollution	temperatures	environmentalists	pollution

Table G.20: Nearest neighbors for the test timestamps of the word *global warming* obtained using the predict-next method.

2014	**2015**	**2016**	**2017**	**2018**	**2019**
climate change	climate change	carbon dioxide	climate change	climate change	climate change
emissions	emissions	climate change	carbon dioxide	carbon dioxide	carbon dioxide
carbon dioxide	carbon dioxide	emissions	emissions	emissions	emissions
climate	climate	climate	temperatures	planet	climate
atmosphere	forests	environment	planet	environmental groups	planet

Table G.21: Nearest neighbors for the test timestamps of the word *global warming* obtained using the predict-ith method.

2014	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
climate change	climate change	climate change	climate change	climate change	climate change	climate change	climate change
emissions	emissions	climate	climate	carbon dioxide	carbon dioxide	climate	climate
carbon dioxide	climate	emissions	carbon dioxide	climate	climate	emissions	emissions
climate	carbon dioxide	carbon dioxide	emissions	emissions	emissions	e p a	e p a
atmosphere	planet	e p a	e p a	atmosphere	atmosphere	carbon dioxide	atmosphere

Curriculum Vitae

Roberto Camacho Barranco earned his Bachelor of Engineering degree in Mechatronics Engineering from the Tecnológico de Monterrey Campus Juárez in 2009. In 2015, he received his Master of Science degree in Computation Science and Engineering from the Technische Universität München. He joined UTEP's doctoral program in Computer Science in 2015. Dr. Camacho Barranco was the recipient of the Good Neighbor Scholarship and a UTEP Roy and Keith Chapman Endowed Presidential Scholarship. He was also a recipient of a Generation Google Scholarship.

Dr. Camacho Barranco has presented his research at several meetings, including the 2018 IEEE BigData 4th Special Session on Intelligent Data Mining, the 2017 International Workshop on Big Data, Streams and Heterogeneous Source Mining, and the 2017 IEEE International Conference on Information Reuse and Integration. His work has appeared in the proceedings of these conferences, as well as The International Journal of Data Science and Analytics.

While pursuing his degree, Dr. Camacho Barranco worked as a Research Assistant for the High-Performance Systems and Discovery Analytics laboratories. In 2017 and 2018, he was as an intern at Google, where he will start working after graduation as a full-time software engineer. Dr. Camacho Barranco's dissertation, "Time-Reflective Text Representations for Semantic Evolution Tracking and Trend Analytics", was supervised by Dr. M. Shahriar Hossain.

Permanent address: Privada Jardin 1635

Ciudad Juarez, Chihuahua, Mexico 32460