Temporal word embedding with predictive capability

Ahnaf Farhan (afarhan@miners.utep.edu)
The University of Texas at El Paso
Roberto Camacho Barranco
The University of Texas at El Paso
Monika Akbar
The University of Texas at El Paso
M. Shahriar Hossain
The University of Texas at El Paso

Research Article

Keywords: word embedding, dynamic embeddings, temporal embeddings, neural network

Posted Date: December 3rd, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2322214/v1

Version of Record: A version of this preprint was published at Knowledge and Information Systems on July 13th, 2023. See the published version at https://doi.org/10.1007/s10115-023-01920-8.

Temporal word embedding with predictive capability

Ahnaf Farhan[1*†], Roberto Camacho Barranco[1†], Monika Akbar[1] and M. Shahriar Hossain[1*]

[1*]Department of Computer Science, The University of Texas at El Paso, 500 W University Ave, El Paso, 79968, Texas, USA.

*Corresponding author(s). E-mail(s): afarhan@miners.utep.edu; mhossain@utep.edu;
Contributing authors: rcamachobarranco@miners.utep.edu; makbar@utep.edu;
†These authors contributed equally to this work.

Abstract

Semantics in natural language processing is largely dependent on contextual relationships between words and entities in a document collection. The context of a word may evolve. For example, the word "apple" currently has two contexts – a fruit and a technology company. The changes in the context of words or entities in text data such as scientific publications, and news articles can help us understand the evolution of innovation or events of interest. In this work, we present a new diffusion-based temporal word embedding model that can capture short and long-term changes in the semantics of entities in different domains. Our model captures how the context of each entity shifts over time. Existing temporal word embeddings capture semantic evolution at a discrete/granular level, aiming to study how a language developed over a long period. Unlike existing temporal embedding methods, our approach provides temporally smooth embeddings, facilitating prediction and trend analysis better than those of existing models. Extensive evaluations demonstrate that our proposed temporal embedding model performs better in sense-making and predicting relationships between entities in the future compared to other existing models.

Keywords: word embedding, dynamic embeddings, temporal embeddings, neural network

1 Introduction

Text data available over the internet have grown exponentially in the past decade. There are ample techniques to transform an unstructured text collection into a structured representation allowing us to apply conventional data mining and machine learning algorithms for analytical purposes. An issue with the conventional approach to representing unstructured data is that the contextual changes in the meanings of words as the language evolves are not considered in the models. That is, conventional representation models do not consider text publications as evolving streams of data.

The contextual evolution of a word plays a vital role in its contemporary semantics. For example, the context of the word *Cloud* changed over time in the last two decades. The word, *Cloud*, in news articles, reflected its connection with weather-related terms in the beginning. From 2008 to 2012, the word *cloud* started to reflect the context of web-based storage, such as Dropbox and Google drive. Slowly, from 2016 the word *cloud* started to reflect cloud-based computing services, such as Amazon Web Service (AWS), Microsoft Azure, and Google Cloud Platform, as the services became more affordable and popular. Figure 1 demonstrates that the nearest neighbors of the word *cloud* changed over the years for a news dataset. Taking all these changes in the neighborhood of *cloud*, how can we predict its neighborhood in the coming years?

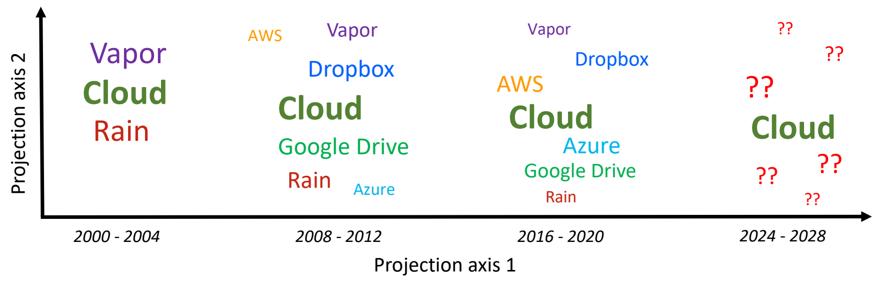


Fig. 1 The nearest neighbors of the word cloud over time.

To serve a larger set of analytic needs – including the prediction of a mathematical space that words may represent in the future – modern unstructured data representation techniques are slowly drifting toward the analysis of temporal aspects of text [1–3]. Nevertheless, the ability to represent unstructured text with a temporal context is still in its infancy. In this paper, we present a temporal embedding model for representing text data in a structured way based on the temporal context.

Our observation from timestamped text collections indicates that new concepts and events do not spike on one day and disappear on the next, rather concepts and events evolve over time. For example, the concept of COVID started to rise at a fast rate in February and March of 2020, but the topic started several months before that. Similarly, the topic of Russia's Ukraine invasion did not occur in one day. In unstructured text representations, discrete timestamps do not help much to reflect the rise and fall of entities well. The appearance of a new entity in one timestamp needs to be diffused earlier than when it appeared first, to reflect a smooth transition to capture evolution. However, not all words and entities evolve, and there can be some words that appear suddenly and then disappear quickly. The scope of this paper is limited to words and entities that evolve over time, with time units of around months or years.

To provide a smooth transition for evolving words in their representations, in one of our previous research efforts, we presented the issue of capturing the evolution of concepts using a diffusion-based time-reflective representation [4]. *Diffusion* of a word, in [4], was reflected by smoothly incorporating its effect across timestamps, before and after it appears. This time-reflective representation enabled the tracking of the meaning of every word in terms of their neighborhood and captured changes in context over time.

One of the challenges of [4] was that each word vector had a length equal to the number of documents in the corpus (See Section 4.1.1), which is not practical for analyzing a corpus containing thousands of documents. Moreover, since the vectors generated for the words were directly dependent on the documents where those words appeared, vectors for the same words in the future cannot be extrapolated (because the future documents are yet to be seen and not features of this representation). That means the model did not generate embeddings with prediction capability. To address these challenges, **in the current paper, we construct a contextual low-dimensional temporal embedding space that mimics this high-dimensional representation while maintaining the essential diffusion information contained in the vectors.** We introduce a neural-network-based framework that generates low dimensional **temporal word embeddings** while optimizing for multiple key objectives.

Word embeddings are low-dimensional vector space models obtained by training a neural network using contextual information from a large text corpus. There are several variants of word embeddings with different features, such as word2vec [5, 6], GloVe [7], and BERT [8]. These embedding techniques do not explicitly address dynamic changes in context. The few existing efforts [3, 9–11] to generate dynamic low-dimensional language representations fail to integrate the concept of temporal diffusion into language models effectively. Moreover, these existing models cannot simultaneously capture both the short-term and long-term drifts in the meaning of words. As a result, sharply trending concepts, such as *COVID-19* (coronavirus disease 2019), cannot be modeled in the embedding space when long-term drifts are considered. On the other hand, long-range effects – such as the change in the meaning of the word *cloud* – are not captured when these algorithms take only short-term drifts into account. Moreover, the existing models only have limited capability to generate dynamic embeddings that can be used for predicting future embeddings.

The diffusion-based temporal word embedding model that we propose in this paper is able to capture both short-term and long-term in the corpus. It incorporates diffusion into the modeling by integrating the time dimension smoothly into the model's objective function. The temporal embeddings are generated for all words in all timestamps in a connected space, and hence the vectors can be used to predict embeddings of words in the future, unlike other existing temporal or dynamic embedding techniques. **In this paper, we use the phrase *temporal word embeddings* and *dynamic word embeddings* interchangeably.**

The experimental results in Section 5 show that the proposed method performs significantly better than the state-of-the-art temporal embedding models [3, 11, 12] in capturing both short-term and long-term changes in word semantics. Additionally, our model provides embeddings to facilitate more meaningful predictions for the future context of a word.

The contributions of this paper are summarized as follows:

- This paper describes a neural network model that generates low-dimensional dynamic embeddings from high-dimensional time-reflective feature vectors without degrading the quality of word vectors.
- Our proposed model creates a homogeneous embedding space for all timestamps in the data so that each word's temporal embedding can be seen as a multi-variate signal, which conventional signal prediction algorithms can leverage to predict embeddings in a future timestamp.
- We introduced a diffusion mechanism in the objective function in order to smooth the embeddings for better prediction capability.
- We compare the new model with regular temporal co-occurrence-based, dynamic-embedding-based, and time-masking transformer-based models on the task of semantic change detection.
- In this paper, we introduce the application of predicting a future embedding space for an existing timestamped document collection.

2 Related Work

This section provides a detailed review of the literature on the various aspects of our research work in order to put our contributions in perspective. The related works presented here are divided into two subsections, describing two different tasks: the generation of temporal embeddings and the prediction of a future embedding space.

2.1 Generation of temporal embeddings

Semantic evolution: Meanings of words in a language change over time depending on their use [13, 14]. Temporal syntactic and semantic shifts are called *diachronic changes* [1]. Several probabilistic approaches tackle the problem of modeling the temporal evolution of a vocabulary by converting a set of timestamped documents into a latent variable model [15–18]. Other approaches model diachronic changes using Parts of Speech features [19] or using graphs where the edges between nodes (that represent words) are stronger based on context information [20]. However, **tracking semantic evolution** is not possible using these techniques because they do not generate language models.

Language Models: The state-of-the-art technique for language modeling is word2vec, introduced by Mikolov et al. [5, 6]. This method generates a *static* language model where every word is represented as a vector (also called *embedding*) by training a neural network to mimic the contextual patterns observed in a text corpus. There are several variants of this method which include probabilistic approaches [21] as well as matrix-factorization-based techniques such as GloVe [7]. A major challenge with *static* representations is that they do not incorporate any temporal information that can be used for tracking semantic evolution. Our work focuses on incorporating the temporal dimension of text data into text embedding models so that evolution of a vector space over time can be studied.

Static to dynamic embeddings: A proposed solution to tracking semantic evolution is to obtain a *static* representation for each timestamp in a corpus and then artificially couple these embeddings over time using regression or similar methods [1, 3, 12, 22]. However, this approach has several drawbacks. First, it requires having a significant number of occurrences for all words at all times, which is usually not the case since words can gain popularity or appear at different times. Second, the artificial coupling of embeddings across timestamps can introduce artifacts in the model that may lead to wrong conclusions. A potential solution to the sparsity problem is introduced by Camacho et al. [4], which leverages diffusion theory [23] to generate a robust temporal representation. The technique uses a temporal tf-idf representation in which the model changes size with the number of documents and as a result, is not extensible.

Joint training of temporal embeddings: The drawbacks of using *static* word embedding models to generate temporal representations have led to the development of new techniques that can train the embeddings for different timestamps jointly. The models use filters or regularization terms to connect the embeddings over time. Yao et al. [10] propose to generate a co-occurrence-based matrix and factorize it to generate temporal embeddings. The embeddings over timestamps are aligned using a regularization term. Rudolph et al. [11] apply Kalman filtering to *exponential family embeddings* to generate temporal representations. Bamler et al. [9] use similar filtering but apply it to embeddings using a probabilistic variant of *word2vec*. According to Bamler et al. [9], using a probabilistic method makes the model less sensitive to noise. All these methods focus primarily on capturing long-term semantic shifts, while our goal is to be able to capture both long and short-term shifts.

2.2 Prediction of a future embedding space

Prediction using linear transformation: Several researchers have focused on studying multi-dimensional time-series prediction [24–27] using methods such as linear regression and accounting for temporal effects such as seasonality [25].

[28] discussed linear, homogenous, and heterogenous transformation of embeddings with timestamps for estimating embeddings for future timestamps. Linear transformation learns a mapping between embeddings of two consecutive timestamps. Homogenous transformation learns a mapping between stacked embeddings of timestamp 1 to $(T-1)$ and stacked embeddings of timestamp 2 to (T). In heterogeneous transformation, instead of stacking the embeddings from all the timestamps, it learns weight matrices for mapping embeddings from all pairs of consecutive timestamps and uses various smoothing functions to combine the weight matrices. However, the embedding vectors for each timestamp are learned separately, which means the embedding vectors at each timestamp are static. Some other studies [29], [30] also leverage linear transformations of embedding vectors in order to predict embeddings in the future timestamps [29] first train a dynamic embedding model then create a time-context vector to predict estimated embeddings for the next timestamp through the linear transformation of embeddings of the present timestamp.

However, these models are not well-suited for the prediction of high-dimensional non-linear phenomena, which is the case of semantic evolution.

Prediction using non-linear modeling: Our generated temporal embedding is a nonlinear sequence of signals. Therefore, our prediction of a future embedding space from existing time series data requires sequence modeling that can handle nonlinearity.

To model non-linear phenomena, several neural-network-based sequence modeling techniques have been introduced recently [31–34]. In this paper, we leverage state-of-the-art techniques in terms of sequence modeling, such as (1) the Long Short Term Memory (LSTM) [31] and (2) Gated Recurrent Units (GRU) [32] recurrent neural networks (RNNs). We also explore the effect of adding an attention mechanism [33] 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 [34] which, in contrast to RNNs, can be trained in parallel.

3 Problem Description

In this paper, we focus on timestamped text corpora, such as collections of scientific publications, or news articles, that have publication dates. Let $\mathcal{D} = \{d_1, d_2, \ldots, d_{|\mathcal{D}|}\}$ be a corpus of $|\mathcal{D}|$ documents and $\mathcal{W} = \{w_1, w_2, \ldots, w_{|\mathcal{W}|}\}$ be the set of $|\mathcal{W}|$ entities (names, places, scientific terms) extracted from the text corpus $\mathcal{D}$. We consider each of the 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 paper is two-fold.

- **Task 1: Constructing a low-dimensional temporal embedding space, with predictive capability:** The first task is to obtain a temporal word embedding model $\mathcal{U}$ from corpus $\mathcal{D}$. 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 constant that indicates the size of each vector. 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 all the timestamps.

- **Task 2: Predicting a future embedding space:** The second task is associated with predicting a future embedding space. We aim 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 timestamps and predicts as output embedding vectors for every word in the vocabulary for timestamp t_{b+1}. The output of $\mathcal{P}$ can be used to forecast the future contexts of the words in the vocabulary.

4 Methodology

This section is divided into two subsections describing the two tasks: temporal embedding generation and prediction of a future embedding space, as outlined in the problem description.

4.1 Construction of a temporal embedding space

To construct a temporal embedding space that can be used for the prediction of future embedding space, we design an objective function that satisfies several crucial aspects in terms of similarity between words, weights reflecting relevance between contextual words, temporal diffusion of amplitudes of words, and connecting embedding spaces of different timestamps. We use a shallow neural network to accommodate the embedding generation with a complex objective function that models a temporally-driven training dataset. The training data generation and each component of the objective function are explained below.

4.1.1 Training data for generating low-dimensional temporal embeddings

To construct the training dataset for generating low-dimensional temporal embeddings, we use the diffusion-based time reflecting representation from our previous research efforts [4]. In this subsection, we first summarize the steps of generating a diffusion-based time-reflecting representation, constructed over a high-dimensional space and not having predictive capacity. Afterward, we discuss the process of creating training data.

As part of our previous research [4], we compute the frequency of words appearing in the documents over time to track the semantic evolution of the words. In our earlier research, we observed that the distribution of the frequency of a word over time was severely irregular, in particular for words or noun phrases that suddenly appeared at a particular timestamp or that are used sporadically. A frequency distribution that is inconsistent or uneven with time does not help capture trends because an evolving trend is considered a slower process compared to a sporadic one. Our idea of evolving trends is based on a social science concept known as the diffusion theory [23].

Based on *diffusion theory*[23], which refers to the change of the distributional patterns of a phenomenon over time, the meaning of a word and, consequently, its vector representation, diffuses over time. In [4], to smooth the word vectors over time based on the diffusion theory, we assumed 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 used a Gaussian filter to diffuse the contribution of the document smoothly before and after the publication date of the document. The Gaussian filter used a sliding window, going from the first to the last timestamp. The contribution of a document d increased the closer its timestamp t_d was to the current timestamp t. The tf-idf weight of a word was modified at each timestamp with Equation (1) to generate a time-reflective representation, which is referred to as temporal tf-idf [4].

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

Here, $\hat{w}$ is the weighted tf-idf value at timestamp t for the word $w \in \mathcal{W}$ in document $d \in \mathcal{D}$, which was published at timestamp t_d. The term $f_{w,d}$ represents the term frequency of word w in document d, λ_w is the number of documents that contain word w, and $\mathcal{W}_d$ is the set of words that appear in document d. The standard deviation of the Gaussian distribution function is represented by ς, and is set by the user.

In order to construct a training dataset for the current paper, using the temporal tf-idf representation of Eq. 1, we compute the cosine dissimilarity (1.0–cosine similarity) between every pair of words and store these as a distance/dissimilarity matrix Δ, where each element can be addressed as $\delta_{ijt} \in \Delta$. This distance/dissimilarity matrix Δ becomes the training data for the expected distance/dissimilarity between a particular pair of words $(w_i, w_j) \in \mathcal{W}$ at time $t \in \mathcal{T}$. We use the notation δ_{ij} to represent a vector of size $|\mathcal{T}|$ with the temporal tf-idf-based cosine dissimilarity between $(w_i, w_j) \in \mathcal{W}$ for all the timestamps. The cosine dissimilarities are later used in the output layer of our proposed neural network.

4.1.2 Simplistic embedding model optimizing for similarity only

One of our objectives is to obtain a low-dimensional word embedding model $\mathcal{U}$ such that computing the cosine dissimilarity between the word vectors results in a distance matrix that closely resembles Δ. Equation (2) formulates this objective as ϑ. In this case, we are optimizing the vectors in $\mathcal{U}$ to minimize the difference between the cosine dissimilarity of each pair of word vectors for every timestamp and the cosine dissimilarity from temporal tf-idf model in

Δ (Eq. (1)). The minimization of the difference will ensure that our model captures the same similarity as the temporal tf-idf model, but ours will provide low-dimensional contextual vectors.

In this paper, the term $dist(A, B)$ refers to the cosine dissimilarity between vectors A and B. The cosine dissimilarity between a pair of word vectors is bounded between $[0, 1]$. A cosine dissimilarity of 0 between two word vectors means that both words share the same context, while a cosine dissimilarity of 1 means that the vectors are completely orthogonal, thus they do not share contextual similarities. 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, incorporating only the similarity aspects, is as follows.

$$\vartheta_1(\mathcal{U}) = \sum_{i=1}^{|\mathcal{W}|}\sum_{j=1}^{|\mathcal{W}|}\sum_{t=1}^{|\mathcal{T}|}\left(\alpha \cdot dist(u_{it}, u_{jt}) - \alpha \cdot \delta_{ijt}\right)^2 \quad (2)$$

4.1.3 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. We empirically discovered 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 temporal tf-idf-based cosine dissimilarity δ_{ijt} is small, ensuring that our word embedding model captures the *relevant* context accurately.

$$\vartheta_2(\mathcal{U}) = \sum_{i=1}^{|\mathcal{W}|}\sum_{j=1}^{|\mathcal{W}|}\sum_{t=1}^{|\mathcal{T}|}\left(\alpha \cdot dist(u_{it}, u_{jt}) - \alpha \cdot \delta_{ijt}\right)^2 \cdot e^{-\beta\delta_{ijt}} \quad (3)$$

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 Eq. (3) imposes a higher penalty to examples with smaller baseline distances. The penalty is less when the dissimilarity from the temporal tf-idf model is large (that is a lesser penalty for contextually similar words). Equation (3) 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 temporal tf-idf model.

4.1.4 Temporal diffusion filter

In connection with the diffusion theory [23] (introduced earlier with frequency-based training data generation in Section 4.1.1), we assume that the meaning of a word, and consequently its vector representation, diffuses (or drifts) over time. Thus, our model should generate word embeddings that evolve smoothly over time. To introduce this concept in our objective function, we model the effect of every word vector in all timestamps to some degree.

We use a Gaussian filter (Eq. (4)) to *diffuse* the contribution of each vector smoothly before and after the timestamp of 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 for capturing short-term changes in meaning.

$$\gamma(t,\sigma) = \left\langle \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t_i - t)^2}{2\sigma^2}} \right) \text{with } t_i = 1, \ldots, |\mathcal{T}| \right\rangle \tag{4}$$

Equation (5) presents the updated objective ϑ_3 which includes the temporal diffusion of the word embeddings.

$$\vartheta_3(\mathcal{U}) = \sum_{i=1}^{|\mathcal{W}|} \sum_{j=1}^{|\mathcal{W}|} \sum_{t=1}^{|\mathcal{T}|} \left(\alpha \cdot dist(\gamma(t,\sigma)U_i, \gamma(t,\sigma)U_j) - \alpha \cdot \delta_{ijt} \right)^2 \cdot e^{-\beta\delta_{ijt}} \tag{5}$$

4.1.5 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 (6) captures the expected behavior by penalizing significant spatial changes.

$$\varepsilon_{1a}(\mathcal{U}) = \sum_{i=1}^{|\mathcal{W}|} \sum_{t=1}^{|\mathcal{T}|-1} ||u_{it+1}, u_{it}||_2 \tag{6}$$

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 Eq. (7).

$$\mathcal{F}_a(\mathcal{U}) = \vartheta_3(\mathcal{U})^{\omega_\vartheta} \varepsilon_1(\mathcal{U})^{\omega_\varepsilon} \tag{7}$$

An alternative form would be:

$$\mathcal{F}_b(\mathcal{U}) = \omega_\vartheta \log \vartheta_3(\mathcal{U}) + \omega_\varepsilon \log \varepsilon_1(\mathcal{U}) \tag{8}$$

or

$$\mathcal{F}_c(\mathcal{U}) = \omega_\vartheta \vartheta_3(\mathcal{U}) + \omega_\varepsilon \varepsilon_1(\mathcal{U}) \tag{9}$$

4.1.6 Implementation

We implemented a neural network-based model using Tensorflow to generate our **low-dimensional temporal word embeddings**. An overall view of the architecture of our neural network is shown in figure 2. The goal of the neural network is to minimize Eq. (7). The embeddings for all words in all timestamps are generated in the hidden layer. We initialize the weights in the hidden layer in the range [0, 1]. The data used for training the model contains three inputs (one-hot encoding of a pair of words for which the cosine dissimilarity is known, and the timestamp) and one target value (cosine dissimilarity). The inputs are the indices for two random words w_{it} and w_{jt}, at timestamp t. The target value is the expected cosine dissimilarity between w_{it} and w_{jt}, obtained using the temporal tf-idf representations of Eq. (1).

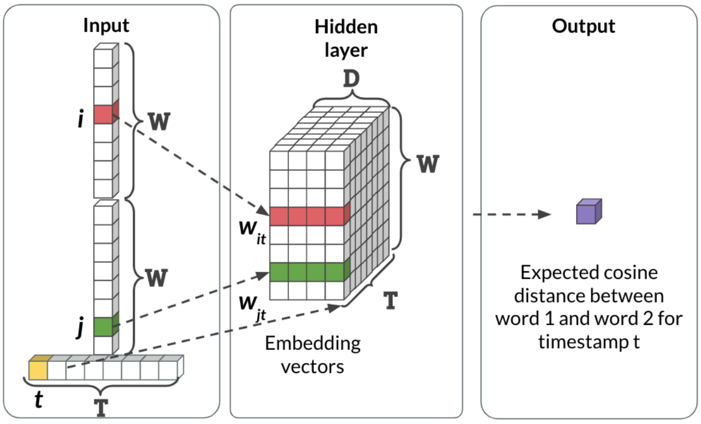

Input
i
W
W
j
t
T
Hidden layer
D
W
w_it
w_jt
T
Embedding vectors
Output
Expected cosine distance between word 1 and word 2 for timestamp t

Fig. 2 The proposed neural network architecture for temporal embedding generation in the hidden layer.

4.2 Sequential model predicting a future embedding space:

In this subsection, we explain how to extrapolate the generated temporal embeddings to build an embedding space for a future timestamp.

For predicting a future embedding space, we leverage state-of-the-art techniques in terms of sequence modeling, such as (1) the Long Short Term Memory (LSTM) [31] and (2) Gated Recurrent Units (GRU) [32] recurrent neural networks (RNNs). We also explore the effect of adding an attention mechanism [33] 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 [34] which, in contrast to RNNs, can be trained in parallel. The following subsections provide a detailed description of the four sequential models we used for predicting future embeddings.

4.2.1 LSTM model structure

Most of the progress in RNN-based sequence modeling has been oriented towards machine translation, which uses an encoder-decoder architecture [32, 33, 35]. 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. Each element of the sequence is a word embedding for timestamp t, or, more generally, a fixed-size vector.

Figure 3 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.

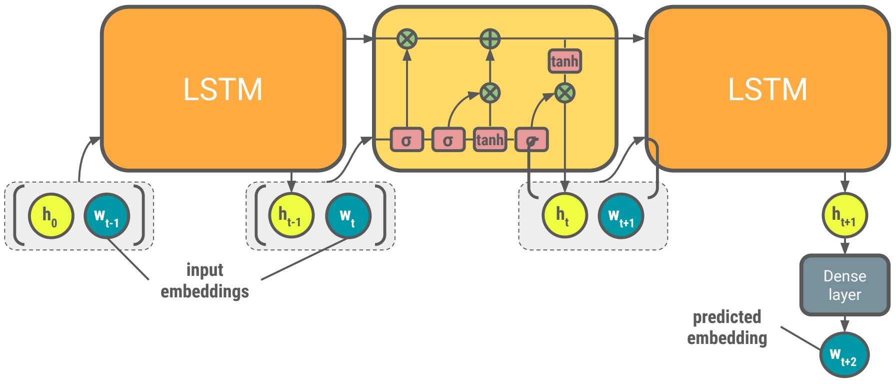


Fig. 3 Structure of the LSTM-based network used for word embedding prediction.

4.2.2 GRU model structure

Figure 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.

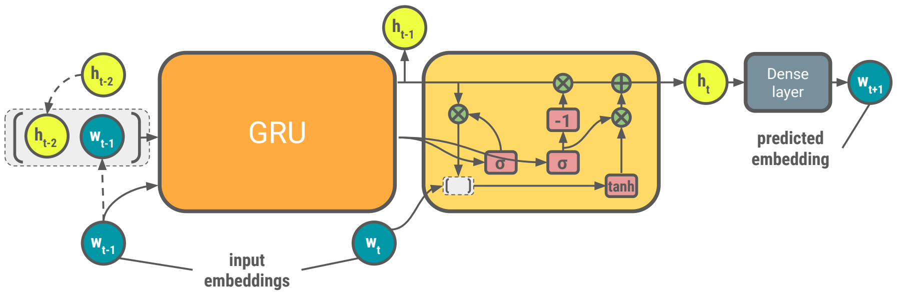


Fig. 4 Structure of the GRU-based network used for word embedding prediction.

4.2.3 Attention-based model structure

Figure 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.

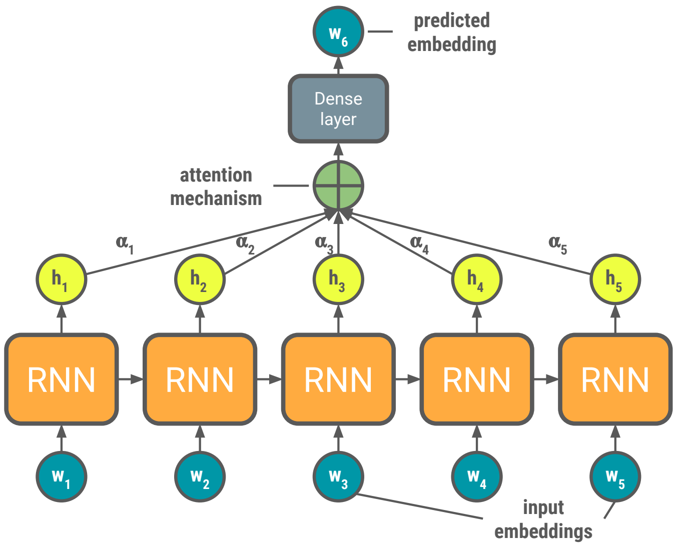


Fig. 5 Structure of the attention-based networks [33] used for word embedding prediction, with LSTM and GRU cells instead of vanilla RNNs.

4.2.4 Transformer-based approach

The sequential nature of the different versions of RNNs prohibits parallel training. The *Transformer* model [34] 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 [34].

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 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.
- 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

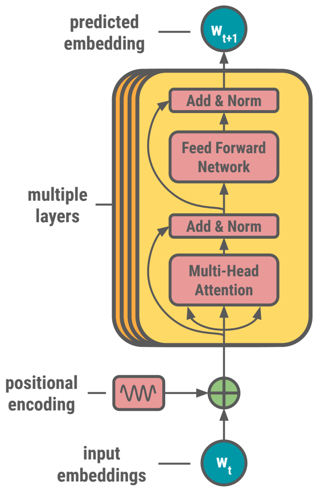


Fig. 6 Structure of the network used for word embedding prediction using only the encoder layer from the Transformer architecture [34].

al. [34], which consists of a vector of sine-cosine pairs at each position that rotate at different frequencies.

5 Experimental Results

We performed experiments using multiple datasets: a synthetic dataset, PubMed Pandemic dataset, PubMed COVID dataset, NyTimes news dataset, and the National Vulnerability dataset. Experimental analysis is conducted using these datasets to evaluate the stages of problems described in the problem description, namely (1) the generation of temporal embeddings and (2) the prediction of future embedding space. Based on these two stages, we present our experimental results in subsections 5.1 and 5.2.

The datasets that we used in this section are outlined in table 1.

Table 1 Datasets used for our experiments.

Data	Documents	Span	# Timestamps
Synthetic Data	10,000	-	10
PubMed(Pandemic) [2]	328,908	2000-2020	21
PubMed(COVID) [1]	374,883	Jan 2020-May 2022	29
PubMed(CANCER) [2]	613,949	1998-2018	21
NyTimes [2]	812,857	1990-2018	29
NyTimes (President) [2]	769,214	1990-2022	32
NyTimes (Russia-Ukraine) [2]	4,957	Jan 2020-Dec 2021	24
NVD [2]	165,552	1999-2021	23

Note: The PubMed(Pandemic), PubMed(Cancer), NyTimes, and NVD dataset files are available in Kaggle link[2] below.

[1]www.kaggle.com/datasets/allen-institute-for-ai/CORD-19-research-challenge[36]

[2]www.kaggle.com/datasets/ahnaffarhan/temporal-word-embedding

We generated **the synthetic dataset** consisting of 10,000 words and ten timestamps. For this dataset, we already know the 10-nearest neighbors of each word in every timestamp. Neighborhoods of larger sizes will contain random words starting at the 11th nearest neighbor.

The **PubMed pandemic dataset**, contains **328,908** abstracts of *pandemic and epidemic*-related biomedical publications. The abstracts were published between years 2000 to 2020. **The PubMed COVID** dataset contains **374,883** abstracts of biomedical papers related to COVID-19, published between 2020-2022. The corpus was collected from Kaggle COVID19 Open Research Dataset Challenge [36]. The **PubMed CANCER** 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. We collected **Russia-Ukraine** related news from **NYTimes** that contains **50,000** news articles published between years 2020 to 2021. The NVD dataset includes **165,552** bulletins published in the last 20 years.

The named entities from the **NYTimes** and **NVD** dataset are extracted using *spaCy*'s Named Entity Recognition (NER) model [37], and We extracted the biomedical entities for the **PubMed** abstracts using *scispaCy*'s Biomedical Named Entity Recognition [38].

The analysis of the PubMed and NVD datasets was limited to the top-3,000 entities 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.

5.1 Experiments on the generated temporal embeddings

We evaluate our temporal word embedding method by comparing its performance with that of a regular tf-idf model, the temporal tf-idf model [4], dynamic Bernoulli embeddings [11], temporal word embeddings with a compass (TWEC) [12], and tempoBert [3]. In all our experiments, we used an **embedding size of 64**.

We seek to answer the following questions through experiments and case studies.

1. What is the effect of introducing different penalty terms in our objective function? (Section 5.1.1)
2. How well do the models perform in terms of capturing the neighborhood of entities over time, compared to the temporal tf-idf? (Section 5.1.2)
3. How well do the models perform in terms of capturing changes in the neighborhood over time in the respective embedding spaces? (Section 5.1.3)
4. How well does our algorithm track the quick evolution of a specific entity, such as *COVID*, compared to other methods? (Section 5.1.4)
5. How well does our algorithm capture the semantic evolution of a general term, such as *pandemic*, compared to other methods? (Section 5.1.5)

5.1.1 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 this study correspond to ϑ_1 (2), ϑ_2 (3), ϑ_3 (5), $\mathcal{F}_a$ (7), $\mathcal{F}_b$ (8), and $\mathcal{F}_c$ (9). We quantify the quality of 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 temporal tf-idf model and each of the different versions of our objective function. The goal of the similarity evaluation is to quantify how well our model mimics the 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 temporal tf-idf model representation does not take into account latent contextual relationships between words.

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.

Figure 7 shows the results for the similarity evaluation with the synthetic dataset described at the beginning of Section 5. The objective function labeled as $\mathcal{F}_a$ on the figure 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 nearest neighbors are known and set as *accurate* in the synthetic data and the rest of the neighbors are random, having an average of 8 intersections means that our model can correctly capture the semantic evolution of the 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 in the embedding space. However, a small maximum MSE value would mean that the word embeddings are not following the trends observed in the temporal tf-idf model-based training. Thus, the best model is one that has high similarity with the temporal tf-idf model while maintaining a low MSE value.

Figure 8 shows the results for the continuity evaluation. In this case, F_c has a continuity of 0.0, 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. (7)), and we confirm that the smoothness penalty (Eq. (6)) has a positive effect both on the similarity and continuity results.

5.1.2 Capability to capture content neighborhood

A major purpose of any temporal or dynamic word representation modeling is to capture content similarity over time. We compare three models – TWEC, Dynamic Bernoulli embeddings, and our temporal word embedding – with Temporal tf-idf [4] in Figure 9, using PubMed (pandemic) dataset. We use

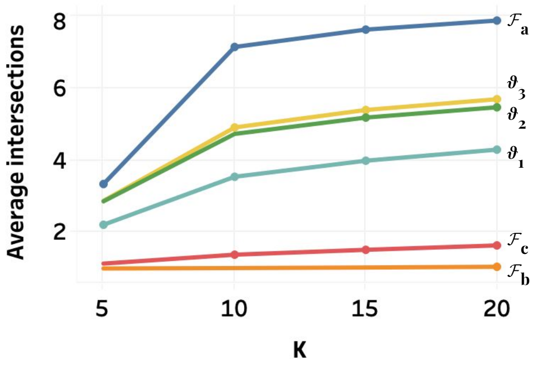


Fig. 7 Average number of intersections per timestamp for different neighborhood sizes (k) between the neighborhoods obtained with the baseline method and those obtained using the different versions of our objective function (embedding size = 64).

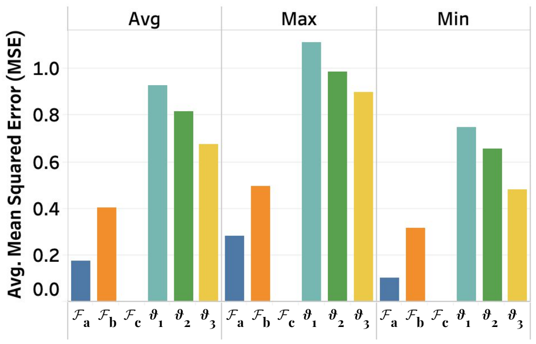


Fig. 8 Average mean squared error (MSE) for different versions of our objective function. The average MSE is computed from obtaining the squared difference between vectors for the same word for every pair of consecutive timestamps (embedding size = 64).

temporal tf-idf [4] for this comparison because it models content smoothly over time. Each line in the figure represents average set-based Jaccard similarity between the 10-nearest neighbors of 1000 randomly selected entities using temporal tf-idf and the 10-nearest neighbors of the same entities using one of the three models. Figure 9 demonstrates that our embedding model and TWEC have closer similarity with temporal tf-idf than Bernoulli embeddings. Additionally, our model has greater similarity with the neighborhood of temporal tf-idf in the earlier timestamps, compared to both TWEC and Bernoulli embeddings. Bernoulli embeddings over different timestamps do not change much to capture the evolution of words. This resulted in an almost horizontal line for Bernoulli embeddings in figure 9.

Our model smoothly spreads word influence using diffusion over the years. As a result, our embedding model performs significantly better than other methods, even when the vocabulary is smaller in the earlier timestamps.

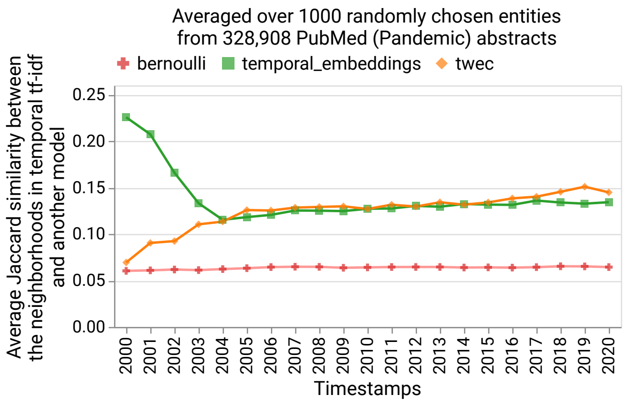


Fig. 9 Jaccard similarity in the neighborhoods between temporal tf-df [4] and each of the three models – TWEC, Bernoulli embeddings, and our temporal word embedding model. (PubMed (pandemic) dataset. Embedding size = 64.)

5.1.3 Capability to detect changes in neighborhood

An objective of a temporal embedding technique is to capture changes in the neighborhood of each word over time. The ability to capture changes allows us to study the evolution of concepts. This subsection provides an experiment to investigate how much change occurs from one year to another in the neighborhood using different models. We quantify the change in terms of set-based Jaccard dissimilarity (1.0-Jaccard similarity) between the neighborhood of a word in the current year and the neighborhood of the same word in the previous year. Average Jaccard dissimilarity over many words in a certain year for a model gives an overall idea of how much the model can detect changes in the neighborhood.

Figure 10 demonstrates average Jaccard dissimilarity (change) at each year for five different models – our temporal embedding model, Bernoulli embeddings, TWEC, and vanilla tf-idf computed independently at each year, and temporal tf-idf using 1000 randomly selected entities from the PubMed (pandemic) dataset. The plot shows that our temporal embedding model detects more changes in terms of average Jaccard dissimilarity compared to other models.

The Bernoulli embeddings capture the least amount of changes. Based on further investigation (not covered in this paper), we noticed that Bernoulli embeddings rarely capture any changes. These embeddings capture only a few long-term changes, whereas our temporal embedding model significantly captures both long-term and short-term changes. TWEC captures more changes than Bernoulli and temporal tf-idf, but lesser changes than the vanilla tf-idf. Our temporal word embedding performs even better than the vanilla tf-idf. Contextual changes are best-captured using our temporal embedding because the objective function of our model spreads the effect of each word smoothly from the current year to other years. As a result, our model captures changes, in

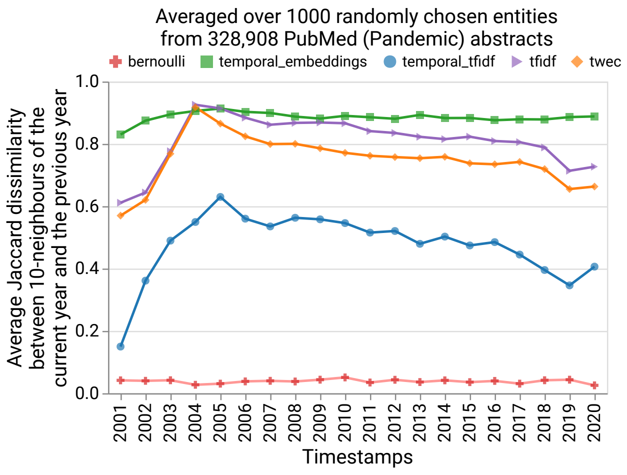


Fig. 10 Comparison of average Jaccard dissimilarity (change) between 10-neighbors of the current year and the previous year.

terms of average Jaccard dissimilarity, better than regular tf-idf and temporal tf-idf models.

Our model is clearly superior in terms of the ability to capture changes. In subsection 5.1.4, we explain how the superiority in the detection of changes in the neighborhood helps in analyzing evolving concepts, such as *COVID-19*.

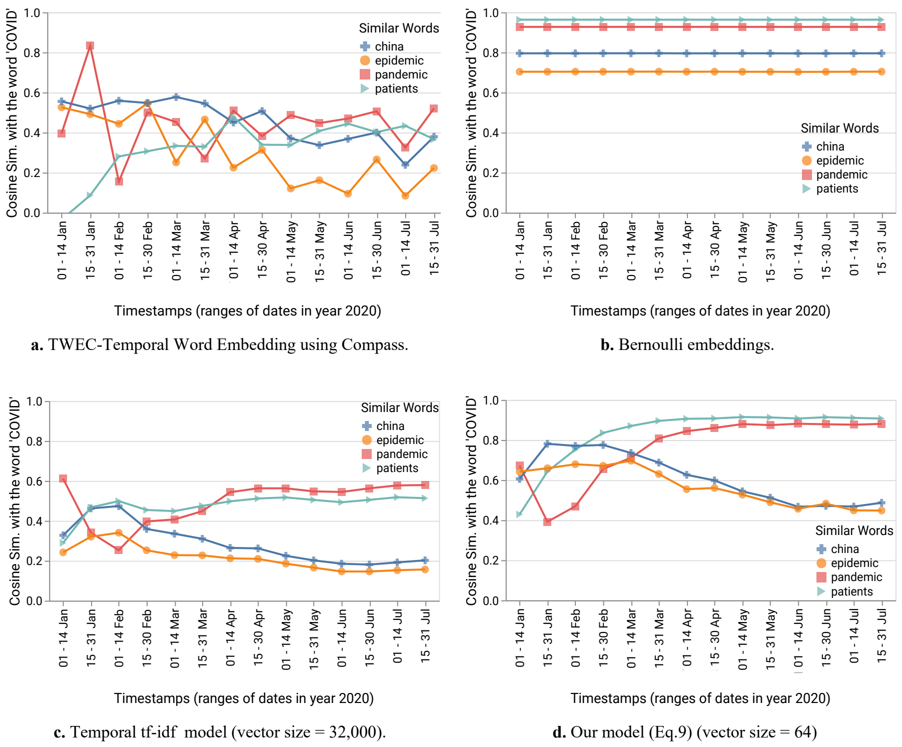
5.1.4 Analyzing the neighborhood of COVID-19

In this experiment, we analyze the changes in the neighborhood of the word *COVID* in the PubMed (COVID) dataset. Figure 11 presents how the similarities between the entity *COVID* and some of its nearest neighbors–*China*, *epidemic*, *pandemic*, and *patients*– change over time using (a) TWEC model, (b) Bernoulli embeddings, (c) temporal tf-idf, and (d) our temporal embedding model. The data contains ranges of two-weeks from January to July of 2020. From August 2020, COVID-19 was considered a pandemic – which is a global outbreak rather than a local epidemic [39]. In figure 11, we observe that (figure 11 (c)) temporal tf-idf and (figure 11 (d)) our temporal embedding can detect the rising trends of *pandemic* and falling trends of the word *epidemic*. This observation matches our known knowledge regarding COVID-19. TWEC (figure 11 (a)) is able to track this to some degree but with zigzag-patterns in the trends. Bernoulli embeddings (figure 11 (b)) give higher similarity for *pandemic* than *epidemic* with the word *COVID*, which is correct in July but the timeline does not demonstrate any rising and falling trends of the words *pandemic* and *epidemic*, as they should based on our knowledge about COVID-19.

Our temporal embedding model (figure 11 (d)) demonstrates that the word *China* had high similarity with *COVID* in the beginning. The similarity started to fall by the end of March. According to our model, starting at the end of March, the word *epidemic* started to exhibit lesser similarity with *COVID* and the word *pandemic* started to show higher similarity. The temporal tf-idf model (figure 11 (c)) demonstrates a similar trend. The trends resemble our common knowledge regarding the COVID-19 pandemic. Also, TWEC (figure 11 (a)) has an overall downward trend for the word *China*, but with zigzag movements over the timeline. Bernoulli embeddings (figure 11 (b)) do not demonstrate any change and capture a static similarity for the entire timeline. We noticed that the underlying vectors in Bernoulli embeddings change, but the neighbors of a word do not change much. That indicates that the changes in the vectors generated by Bernoulli embeddings might be the result of some scaling effect rather than changes due to the reformation of the neighborhood.

We know that the number of COVID-infected *patients* increased over the months of 2020. Our temporal embedding model (as well as the temporal tf-idf) captures the rising-similarity of the word *patients* in the context of *COVID* quite smoothly (figure 11 (d)). TWEC also has an upward trend which is less smooth. However, the Bernoulli embeddings do not demonstrate any changes in the similarity between the words *patients* and *COVID*.

This experiment demonstrates that our temporal embedding model captures the short-term changes in content (as shown by temporal tf-idf) while also



a. TWEC-Temporal Word Embedding using Compass.

b. Bernoulli embeddings.

c. Temporal tf-idf model (vector size = 32,000).

d. Our model (Eq.9) (vector size = 64)

Fig. 11 Evolution of the word *COVID* in PubMed COVID-19-related abstracts published in 2020 using four different models – TWEC, Bernoulli embeddings, temporal tf-idf, and our temporal embedding model. *Cosine similarity* is used to compute the similarity between the vectors of the word *COVID* and any other word.

capturing the context that we can track smoothly to study the evolution of a concept, such as *COVID*. In contrast, Bernoulli embeddings construct a context that is intractable in terms of similarity. TWEC provides noisy patterns that are difficult to interpret.

5.1.5 Analysis of the the word *Pandemic*

With the rise of the COVID-19 pandemic, it has become essential to study how biomedical scientists have dealt with a pandemic in the past years. Such an analysis requires a model that can capture long term changes. In this experiment, we attempt to track the closest term to the word *pandemic* in each year of the PubMed (pandemic) dataset, which spans biomedical abstracts from 2000 to 2020.

Each line of figure 12 plots the similarity of the top nearest-neighbor of the word *pandemic* in each year. The five lines represent similarities using five different models – Bernoulli embeddings, our temporal embedding model, temporal tf-idf, vanilla tf-idf, and TWEC. Notice that our temporal embedding model demonstrates peak similarities in 2009/2010 and in 2020, when H1N1 influenza (swine flu) and COVID-19, respectively became prominent. This signal from our temporal embedding model reflects the fact that the worst pandemics in the last 20 years are the H1N1 influenza in 2009 [40] and COVID-19 in 2020 [41]. Note that other words like *concerns* in 2004 and *public* in 2015 are detected as the top nearest neighbors, which are not highly similar to the word *pandemic*. This indicates that no entities appeared too close to the word *pandemic* in those years.

TWEC captures *influenza* and *H1N1* in the middle of the timeline but fails to capture *COVID*-related keywords in 2020 as the closest entity to *pandemic*. In figure 12, the Bernoulli model can pick up *coronavirus* as the nearest neighbor of *pandemic* but it was not able to pick up influenza in its trend. Moreover, *coronovirus* appears in all the years as the top nearest neighbor of *pandemic* which is not correct because the fact is that the coronavirus spread started in 2019 and became a pandemic in 2020 [41]. Temporal tf-idf and vanilla tf-idf were able to pick up *coronavirus*/*COVID*. Temporal tf-idf and vanilla tf-idf were also able to pick up influenza subtype *H1N1* (swine flu) but the respective similarities were not high.

Based on the experiment presented in this subsection, our temporal embedding model has the ability to separate highly contextual words (such as H1N1 and COVID) of a concept (such as *pandemic*) via similarity-peaks. Our model

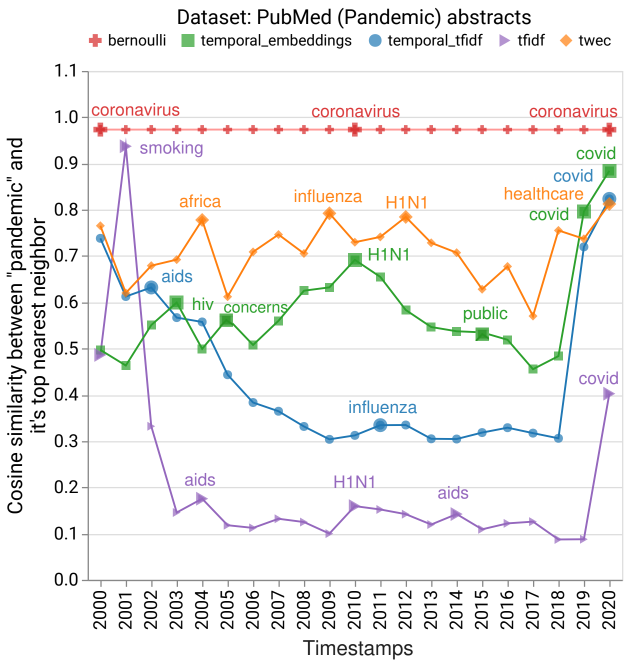


Fig. 12 Cosine similarity of the top nearest neighbor of *pandemic* in each year using all methods. Nearest neighbors are written with selected peaks. Our temporal embedding method provides better context for *pandemic*. Embedding size = 64.

helps in determining prominent neighbors of a concept in the past. Our vectors are able to construct a peak for a prominent nearest neighbor because our method models diffusion. That is, a concept that appears today affects the past and the future to some extent, regardless of whether the concept directly appears in the contents or not.

5.1.6 Comparison with BERT temporal embedding

Based on recent literature, there is a surge in applications using Bidirectional Encoder Representations from Transformers (BERT). BERT provides vectors for each appearance of words. The vectors of the same word appearing in the same timestamp can be used to create a word embedding vector for that word in that timestamp, leading to temporal word embeddings. TempoBert model [3] is such a mechanism. In the TempoBert model, the timestamps are added at the start and end of each sentence as a means of training text data for different timestamps. The BERT-generated embeddings are clustered for semantic evolution. In our experiment here, we use a variant of TempoBert, referred to as temporal BERT, which does not cluster vectors but rather calculates the arithmetic mean of embedding vectors of a word at each timestamp to generate one vector for one word at each timestamp.

A drawback of the BERT-based model is its excessive computation time compared to other embedding models due to the complex structure of the deep learning model. Therefore, for this experimental analysis, we downsample the corpus to 10% of articles per month to ease the computation for BERT. The corpus for this experiment includes 10% of Pubmed COVID-related articles (between Jan 2020 – May 2022) and 10% of NYTimes Russia-Ukraine-related articles (over 24 months between Jan 2020 - Dec 2021). In this section, we examine, how the neighborhood of the words COVID and UKRAINE changes over time using the word embeddings generated by temporal BERT and our temporal embedding model.

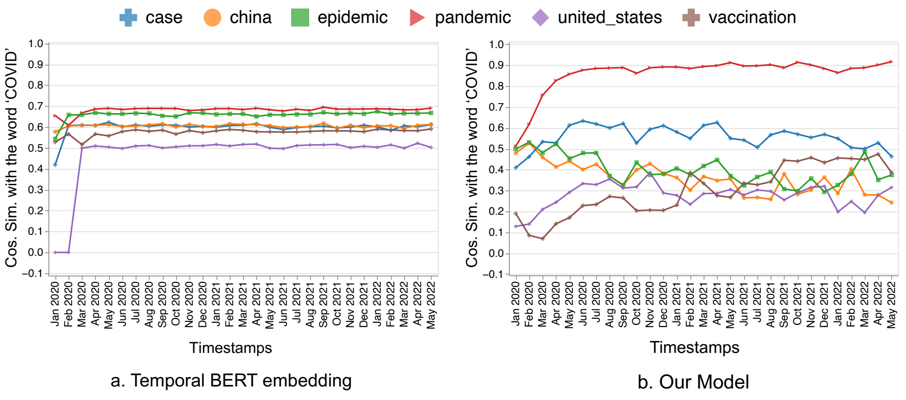


Fig. 13 Cosine similarity of the embedding of "COVID" with neighbor words at the different timestamp. [Temporal BERT vs Our Model]

Figure 13 presents the cosine similarity of embeddings of the word "COVID" with its neighboring words at different timestamps. The plot on the left (a) is generated by using embeddings from temporal BERT, and the plot on the right (b) is generated by using our temporal embedding model. In figure 13(b), we observe that at the beginning of the month of January 2020, the words "case", "china", "epidemic", and "pandemic" are the closest neighbors of the word "COVID". Then, over time, the similarities of the words "china" and "epidemic" with "COVID" decreases while the similarities of the words "pandemic", "united states", and "vaccination", with COVID increase. The changing neighborhood of "COVID" reflects how "COVID" spreads throughout the world over time, starting from china and eventually becoming the world's most widespread pandemic. Further, vaccination programs were initiated at the beginning of 2021. Our model captures these trends well, as reflected in the plot of figure 13(b). The plot on the left (figure 13(a)), on the other hand, provides mostly straight lines and fails to capture such changes in the neighborhood of "COVID".

Another experimental result in figure 14 presents the cosine similarity of embeddings of the word "UKRAINE" with its neighboring words at different timestamps. Based on our known knowledge from the news, Russia invaded Ukrainian territory in February 2022 [42]. As part of our experiment, we trained both the temporal BERT and our model with news data collected prior to the invasion. The embeddings generated by our model (Figure 14 (b)) shows that the similarities between each of the words "cold war", "invasion", and "Russia" with the word "UKRAINE" increased at the end of the year 2021, which is prior to the actual invasion. The temporal BERT, however, failed to detect any changes in the neighborhood of the word "Ukraine". This analysis indicates that the embeddings generated by our model have better prediction capability compared to the temporal BERT embedding model.

Temporal BERT does not perform well due to the fact that complex deep neural network models require extensive training on large datasets. In order to train a BERT model from scratch, it is recommended to use billions of sentences,

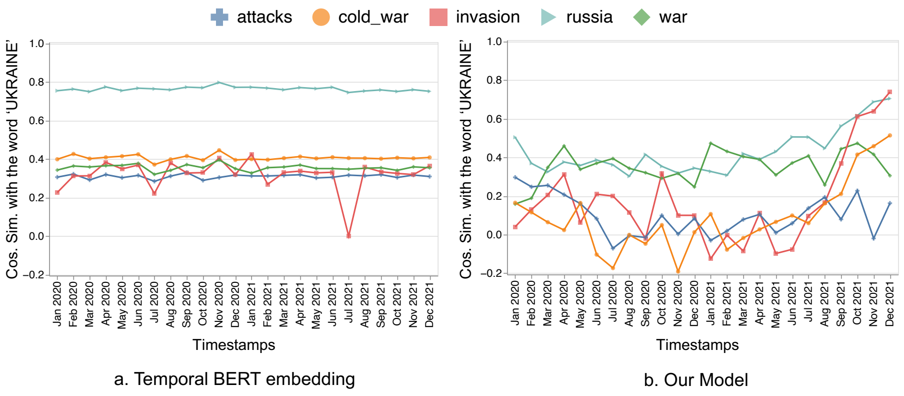


Fig. 14 Cosine similarity of the embedding of "UKRAINE" with neighbor words at the different timestamp. [Temporal BERT vs Our Model]

which is sometimes not available for a specific domain. In our experiment related to COVID (figure 13), we used a pre-trained BERT model (clinical-bert [43] and fine-tuned it with our data. For the Russia-Ukraine-related experiment (figure 14), we used bert-small [44, 45]), which is later fine-tuned by the smaller dataset. In these experiments, we find that despite fine-tuning the BERT model, it fails to capture the temporal evolution of words. Our model performs well even with small datasets, requiring lesser training samples.

5.1.7 Stacking cosine similarities

A streamgraph is a stacked area chart widely used in concept visualization [46]. In this subsection, we provide an analysis of the entity *president* using a streamgraph using the New York Times dataset. Cosine similarities of the nearest neighbors of the entity *president* are stacked in the streamgraph of figure 15.

We observe how the entities *Obama* and *Trump* started to get closer to *president* only a few years before their presidency. *Biden* started to get closer to *president* in the year 2008 when he became the vice president. The cosine similarity between the word *Biden* and the word *president* increased in 2020, which matches with the actual event that Biden won the presidential election in 2020. We also observe the particular cases of political families, such as the *Bush*, and *Clinton* families, that have been relevant to the presidential elections for the last four decades. General entities – such as *White House*, and *United States* – relevant to the entity *president* do not change much in similarity (the width

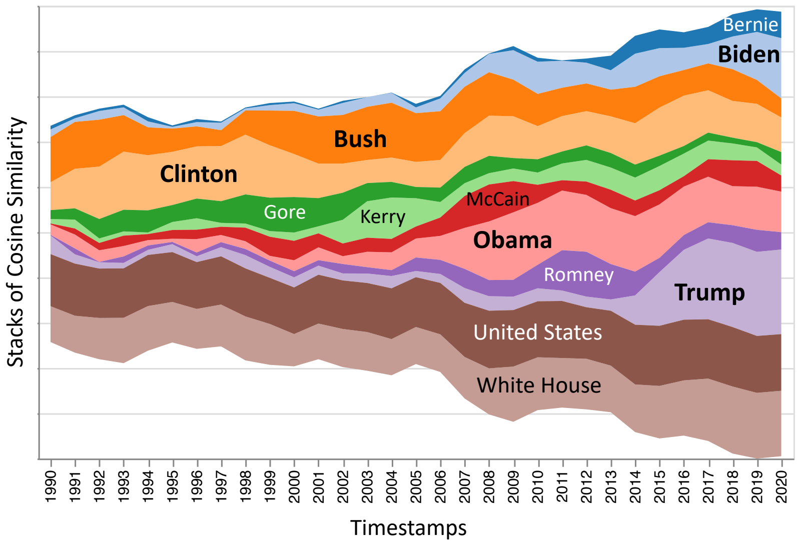


Fig. 15 Streamgraph of the word ***President*** using the New York Times corpus. The width of each of the bands represent cosine similarity of each word with the word ***President***. The word vectors at different timestamps are generated using our temporal word embedding model (embedding size = 64).

of each of the bands) but remain nearest neighbors at all times. This indicates the consistency of the words *White House*, and *United States* in relation to the word *president*. Other entities, on the other hand, appear and disappear over time based on their relevance to the word president. For example, *Gore* and *Kerry* were presidential candidates in the years 2000 and 2004, respectively. Their cosine similarity with the word *president* increased during those years. The results of this experiment demonstrate that our temporal embedding model effectively captures the trends of a historical concept. Additionally, the changes in the cosine similarity of the words are smooth, indicating that the embeddings are changing smoothly over time and that they are well suited for extrapolating into future timestamps for predicting the future state of the embeddings. Subsection 5.2.3 discusses experiments related to predicted embeddings of the word *president*.

5.1.8 Sensitivity analysis for hyperparameters

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 temporal tf-idf method as our accuracy metric.

Figure 16 presents the effect of changing the parameters of interest on the accuracy of our model. The results show that the embedding size, the scale

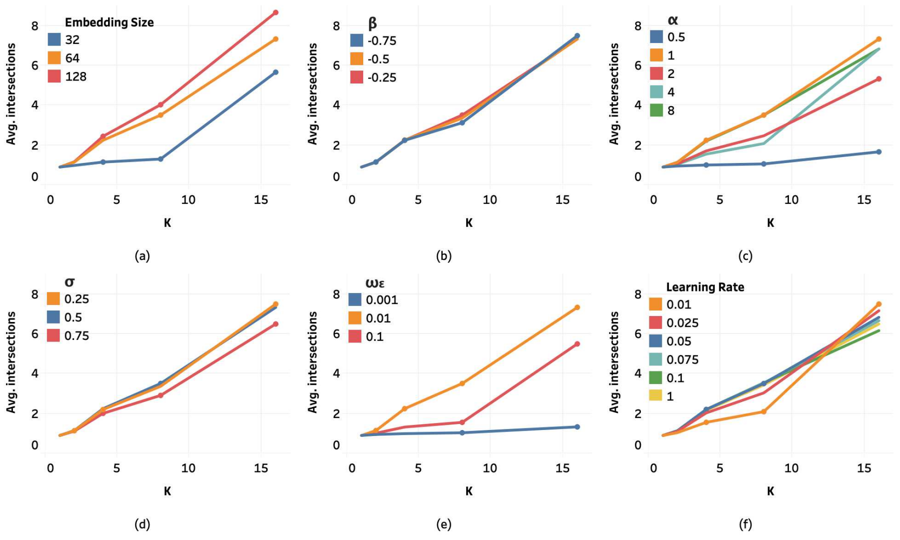


Fig. 16 Average number of intersections per timestamp between the neighborhoods obtained with temporal tf-idf [4] and our temporal embedding model for the NVD data with changing (a)embedding size, (b)β, (c)α, (d)σ, (e)ω_ϵ, and (f) learning rate.

factor (α), and the smoothness penalty factor (ω_ϵ) have significant effects on the accuracy of neighborhood-detection. Figure 16 only presents the results obtained using the NVD dataset. We performed similar analyses for the other datasets, which resulted in similar findings.

Figure 16 (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 training time and computational/storage complexity. Training time increased from $\sim$ 18 hours to $\sim$ 50 hours per experiment in a GPU-enabled cluster when we increase the embedding size from 64 to 128. Therefore, we decided to keep an embedding size of 64. The selection of all the parameters in a neural network with a complex cost function like ours is a “big data” challenge. After many different iterations, we discovered that the size of 64 provides reasonably meaningful results in reasonable run times with the available resources.

5.2 Experiments on prediction capability

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 [47], (2) New York Times articles, and (3) National Vulnerability Database (NVD) bulletins [48].

The temporal word embeddings used as baseline data were generated using the method presented in section 4. 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.

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 5.2.1)
2. How sensitive is the selected time-series modeling technique to changes in the hyperparameters? (Section 5.2.2)
3. How well does our algorithm predict the evolution of a specific term? (Section 5.2.3)

5.2.1 Model selection for prediction

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 5.2.2 describes this sensitivity analysis in more detail.

For each model, we generate predicted word embeddings for every timestamp of the test dataset. We use the neighborhood similarity metric to measure the performance of each model.

Figures 17 and 18 present a comparison, for each dataset, between the best versions of each sequence modeling technique. Figure 17 presents the results in terms of the neighborhood similarity, while Figure 18 shows the effect of changing the neighborhood size K on the average number of intersections between the baseline and the predicted embeddings.

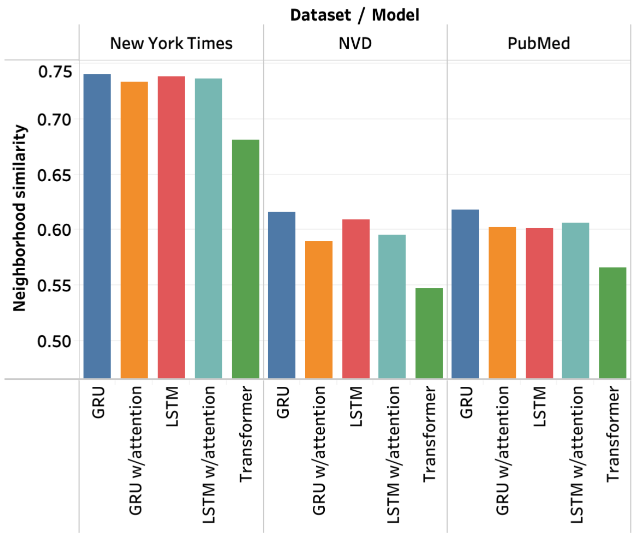


Fig. 17 Neighborhood similarity between the neighborhoods obtained with the baseline temporal embedding method (Eq. 7) and neighborhoods obtained using the predict-next technique with different sequential models for the NVD, PubMed and New York Times datasets, using dynamic values of K.

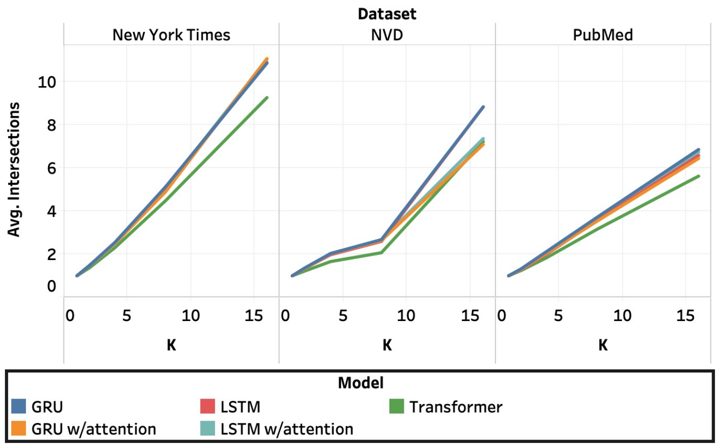


Fig. 18 Average number of intersections for different neighborhood size k per timestamp between the neighborhoods obtained with the baseline temporal embedding method (Eq. 7) and neighborhoods obtained using the predict-next technique with different sequential models for the NVD, PubMed and New York Times datasets (embedding size = 64), using dynamic values of K.

The results show that the GRU and LSTM-based networks outperform the more complex sequential models in all datasets. For both GRU and LSTM, the inclusion of the attention mechanism did not result in better performance. This is because the attention mechanism is explicitly designed to model long sequences [34], but our text-based embeddings have a limited number of timestamps, resulting in a small temporal sequence. There fore GRU and LSTM without the attention mechanism are more appropriate for predicting future embeddings.

5.2.2 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 19 presents the effect of changing the batch size (Figure 19(a)) and the fraction of timestamps used for the training/test split (Figure 19(b)) on neighborhood similarity. Based on the plots, it appears that changing the batch size does not have a significant effect on neighborhood similarity. Meanwhile, changing the train-test size has little effect on neighborhood similarity.

Figure 20 (a) presents the effect of changing the input sequence length on neighborhood similarity. This plot reveals that using an input consisting of the word embeddings for two or more timestamps results in a slightly better performance. Figure 20 (b) shows that changing the number of encoder units for the neural network has a negligible effect on the neighborhood similarity.

The plots of figures 19 and 20 indicate that the generation of neighborhoods of words from predicted embeddings is not sensitive to batch size, training/test split, input sequence size, and the number of encoder units, in general.

Figure 21 presents the effect of changing the optimizer and learning rate on neighborhood similarity. 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 presented by Vaswani et al. [34]. The results clearly exhibit downward or upward trends with increasing learning rates. This is an indication that the neighborhood similarity is sensitive to the

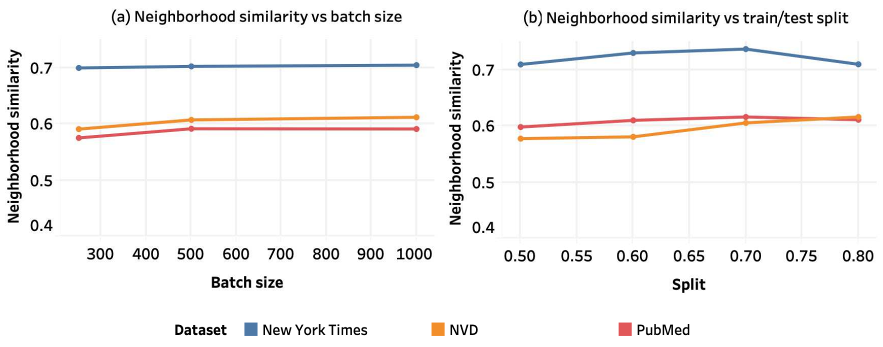


Fig. 19 Neighborhood similarity between the neighborhoods obtained with the baseline temporal embedding method (Eq. 7) and the neighborhoods obtained using the predict-next techniques with different sequential models for the NVD, PubMed and New York Times datasets, using dynamic values of K, while changing (a) the batch size and (b) the fraction of timestamps used for training.

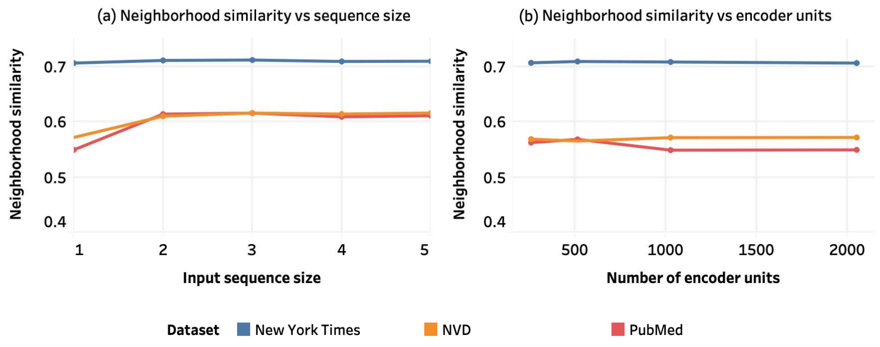


Fig. 20 Neighborhood similarity between the neighborhoods obtained with the baseline temporal embedding method (Eq. 7) and the neighborhoods obtained using the predicted embeddings with different sequential models for the NVD, PubMed and New York Times datasets, using dynamic values of K, while changing (a) the input sequence size and (b) the number of encoder units.

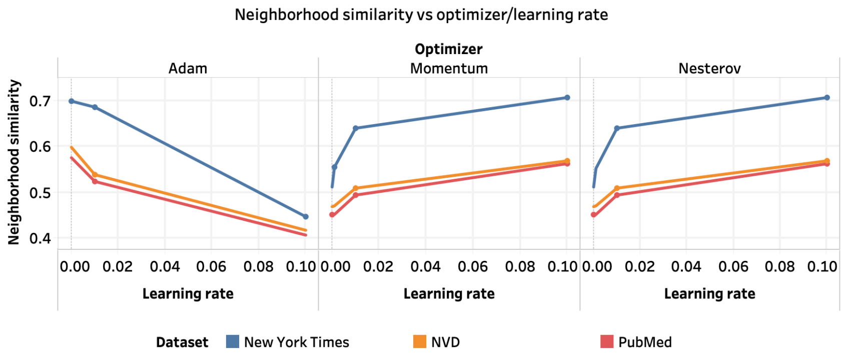


Fig. 21 Neighborhood similarity between the neighborhoods obtained with the baseline temporal embedding method (Eq. 7) and neighborhoods obtained using the predicted embeddings with different sequential models for the NVD, PubMed and New York Times datasets, with different optimizers and learning rates. The learning rate of 0.0 represents the dynamic learning rate presented by Vaswani et al. [34].

learning rate and hence the model requires tuning with different learning rate values to make sure the optimizer does not get stuck in local minima.

5.2.3 Case study and trend analysis

In this section, we perform a qualitative analysis of the performance of the predicted temporal word embeddings on the task of tracking semantic evolution. In this subsection, we report prediction results using LSTM. Similar results are observed using GRU. We used New York Time data for two case studies – how well do the predicted embeddings (1) relevant to the word ***war*** represent our knowledge about contemporary political tension between different countries, and (2) capture entities in the US political domain while studying the word ***president***.

Figure 22(a) contains cosine similarity of the nearest neighbors of the word ***war***. Relevant nearest neighbors – "Russia", "Ukraine", "ISIS", "Syria", and "Taliban" – were selected from the predicted embedding for the year 2022. Embeddings from previous years were used for training the LSTM. Figure 22(a) shows that the embeddings of the words "Syria" and "ISIS" were two words most similar to the embedding of ***war*** in the beginning. Their similarity with ***war*** gradually declined. The prediction for the year 2022 demonstrates the continuous decline of "Syria" and "ISIS" from the word ***war***. The word "Taliban" became more similar to the word ***war*** between 2017 and 2020. The U.S. and the Taliban peace deal occurred in 2020 [49]. The Taliban took over the Afghan government in 2021 [50]. The prediction for 2022 reflects the end of a long-lasting "war" by showing that the similarity of the word "Taliban" and ***war*** will be lesser.

The similarities of the words "Russia" and "Ukraine" with ***war*** were declining through 2017-2020 but started to rise between 2020 and 2021. The predicted embeddings in the year 2022 show that "Russia" becomes the topmost

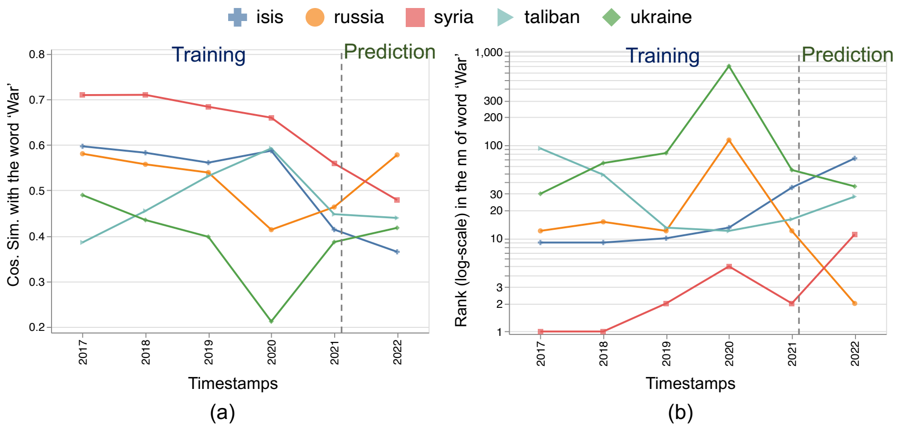


Fig. 22 Evolution of the neighborhood of the term ***war*** in the NYTimes dataset where the embeddings of words at timestamp 2022 is extrapolated using LSTM (a) cosine similarity between the embedding of the word ***war*** and relevant word embeddings at different timestamps. (b) Rank (position) of relevant words in the nearest neighborhood of the word ***war*** at different timestamps.

nearest neighbor of ***war***. Simultaneously, the similarity between "Ukraine" and ***war*** is predicted to increase in 2022.

Figure 22(b) shows the ranks of the same nearest neighbors of the word ***war***. The smaller the rank value more similar a neighbor is to the word ***war***. The vertical axis is in logarithmic scale for better visualization. Similar trends as figure 22(a) are observed in figure 22(b). For example, 'Russia" and "Ukraine" both are coming closer in rank to the word ***war***, which is reflected as a falling pattern in figure 22(b). Also, it is noticeable that the prediction of 'Syria" for 2022 has an upward direction indicating that 'Syria" is shifting away from the word ***war***. On the other hand, the downward direction of "Russia" in the 2022 prediction supports our known knowledge that "Russia" moved closer to ***war***.

For the second case study, our word of interest is ***President***. We selected some relevant words – "Obama", "Trump", "Biden", "Bernie", "Democrats", and "Republicans"– which have been closely associated with the word *President* based on our knowledge of US politics of the past decade. We used embeddings from 2011 to 2020 for training and extrapolated the embeddings of 2021.

In figure 23(a), we observe that the embedding of the word "Obama" is the most similar to the embedding of *President* in the year 2011. In the year 2016, the embedding of the word "Trump" gains more similarity to the word *President* as "Trump" was elected as the new president of the United States. On the other hand, the similarity of the words "Biden" and "Bernie" started to increase in the year 2018 as they were competing for the presidential candidate for the election in the year 2020. The predicted embeddings in the year 2021 show that the similarity of the embedding of the word "Biden" increases substantially, and it becomes the second top closest neighbor between all the relevant words in this study. Figure 23(b) shows that the position of "Biden" in the nearest

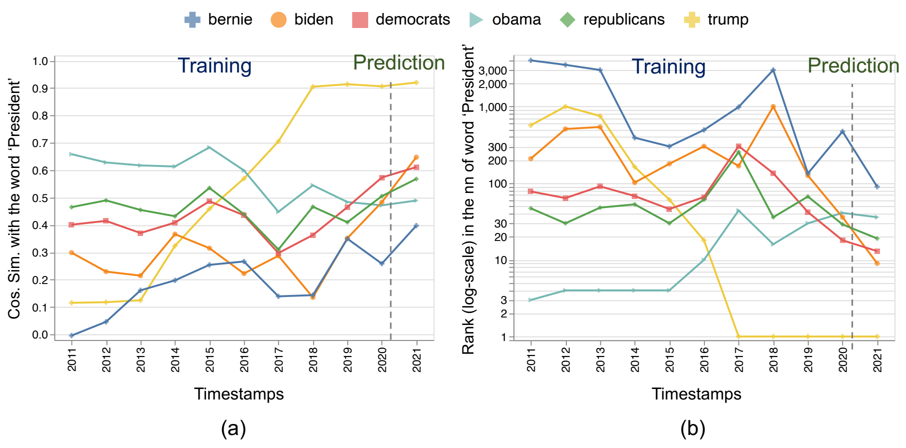


Fig. 23 Evolution of the neighborhood of the term ***President*** in the NYTimes dataset where the embeddings of words at timestamp 2021 is extrapolated using LSTM (a) cosine similarity between the embedding of the word ***President*** and relevant word embeddings at different timestamps. (b) Rank (position) of relevant words in the nearest neighborhood of the word ***President*** at different timestamps.

neighbors of *President* dropped significantly from 1000 to 9 in the last 3 years, indicating that "Biden" quickly moved closer to the word *President*.

These case studies demonstrate that the predicted embeddings using an LSTM-based prediction model can capture the trend in the training data and provide well-explainable relationships between entities in a predicted embedding space of a future timestamp. The performance of forecasting the embeddings depends heavily on the ability of temporal embeddings to capture trends. In section 5.1, we demonstrate that the state-of-the-art models cannot capture changes in the context of words and cannot produce smooth transitions of word similarity over time. While our temporal embedding model can capture the change in the context and produce a smooth transition of embeddings, thus performs well when extrapolated into the future embedding space.

6 Conclusions

This paper introduces a new technique to generate low-dimensional temporal word embeddings for timestamped documents and predict a future embedding space. We compare our temporal word embedding technique with other state-of-the-art techniques. Our temporal embeddings reflect a representation that: (1) can track changes observed within a short period, (2) provides a smooth evolution of the word vectors over a continuous temporal vector space, (3) uses the concept of *diffusion* to capture trends better than the existing models, (4) is low-dimensional, and (5) performs well in capturing future neighborhoods of words. Unlike previous dynamic embedding models, our proposed model creates a homogeneous space over every timestamp of the embeddings. As a result, the generated vectors of timestamps perform well in the prediction of a future embedding space using conventional predictive models. The future direction of our research is to automate hyperparameter tuning and study temporal embedding models for images to learn text-image joint temporal embeddings.

Statements and Declarations

- Financial & Non-financial interests: The authors have no relevant financial or non-financial interests to disclose. All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript. The authors have no financial or proprietary interests in any material discussed in this article.
- Conflicts of interest: The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

[1] Hamilton, W.L., Leskovec, J., Jurafsky, D.: Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. In: Proceedings of ACL, vol. 1, pp. 1489–1501. Association for Computational Linguistics, Berlin, Germany (2016). https://doi.org/10.18653/v1/P16-1141

[2] Tang, X.: A State-of-the-Art of Semantic Change Computation. Natural Language Engineering, 1–28 (2018). https://doi.org/10.1017/S1351324918000220

[3] Rosin, G.D., Guy, I., Radinsky, K.: Time masking for temporal language models. In: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, pp. 833–841. Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3488560.3498529

[4] Barranco, R.C., Dos Santos, R.F., Hossain, M.S., Akbar, M.: Tracking the evolution of words with time-reflective text representations. In: 2018 IEEE International Conference on Big Data (Big Data), pp. 2088–2097. IEEE, Seattle, WA, USA (2018). https://doi.org/10.1109/BigData.2018.8621902

[5] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, pp. 3111–3119. Curran Associates Inc., Red Hook, NY, USA (2013)

[6] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR **abs/1301.3781** (2013). https://doi.org/10.48550/arXiv.1301.3781

[7] Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014). https://doi.org/10.3115/v1/D14-1162

[8] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (2019). https://doi.org/10.18653/v1/N19-1423

[9] Bamler, R., Mandt, S.: Dynamic word embeddings. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 380–389 (2017)

[10] Yao, Z., Sun, Y., Ding, W., Rao, N., Xiong, H.: Dynamic word embeddings for evolving semantic discovery. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp. 673–681. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3159652.3159703

[11] Rudolph, M., Blei, D.: Dynamic Embeddings for Language Evolution. In: Proceedings of the 2018 World Wide Web Conference, pp. 1003–1011. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2018). https://doi.org/10.1145/3178876.3185999

[12] Di Carlo, V., Bianchi, F., Palmonari, M.: Training temporal word embeddings with a compass. Proceedings of the AAAI Conference on Artificial Intelligence **33**(01), 6326–6334 (2019). https://doi.org/10.1609/aaai.v33i01.33016326

[13] Aitchison, J.: Language Change: Progress or Decay?, 4th edn. Cambridge University Press, Cambridge (1981)

[14] Yule, G.: The Study of Language, 6th edn. Cambridge University Press, Cambridge (2016). https://doi.org/10.1017/CBO9781316594131

[15] Radinsky, K., Davidovich, S., Markovitch, S.: Learning causality for news events prediction. In: Proceedings of the 21st International Conference on World Wide Web, pp. 909–918. Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2187836.2187958

[16] Yogatama, D., Wang, C., Routledge, B.R., Smith, N.A., Xing, E.: Dynamic language models for streaming text. Transactions of the Association for

Computational Linguistics **2**, 181–192 (2014). https://doi.org/10.1162/tacl_a_00175

[17] Tang, X., Qu, W., Chen, X.: Semantic change computation: A successive approach. In: Behavior and Social Computing, pp. 68–81. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-04048-6_7

[18] Naim, S.M., Boedihardjo, A.P., Hossain, M.S.: A scalable model for tracking topical evolution in large document collections. In: IEEE BigData, pp. 726–735 (2017). https://doi.org/10.1109/BigData.2017.8257988

[19] Mihalcea, R., Nastase, V.: Word epoch disambiguation: Finding how words change over time. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 259–263. Association for Computational Linguistics, Jeju Island, Korea (2012). https://aclanthology.org/P12-2051

[20] Mitra, S., Mitra, R., Maity, S., Riedl, M., Biemann, C., Goyal, P., Mukherjee, A.: An automatic approach to identify word sense changes in text media across timescales. Natural Language Engineering **21**, 773–798 (2015)

[21] Barkan, O.: Bayesian neural word embedding. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, pp. 3135–3143 (2017)

[22] Rosin, G.D., Adar, E., Radinsky, K.: Learning Word Relatedness over Time. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 1168–1178. Association for Computational Linguistics, Copenhagen, Denmark (2017). https://doi.org/10.18653/v1/D17-1121

[23] Angulo, J., Pederneiras, C., Ebner, W., Kimura, E., Megale, P.: 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)

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

[25] Matsubara, Y., Sakurai, Y., Faloutsos, C.: AutoPlait: Automatic Mining of Co-evolving Time Sequences. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, pp. 193–204. ACM, New York, NY, USA (2014). https://doi.org/10.1145/2588555.2588556

[26] Yu, H.-F., Rao, N., Dhillon, I.S.: High-dimensional Time Series Prediction with Missing Values. arXiv (2015). https://doi.org/10.48550/ARXIV.1509.

08333

[27] Yu, H.-F., Rao, N., Dhillon, I.S.: Temporal regularized matrix factorization for high-dimensional time series prediction. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. NIPS'16, pp. 847–855. Curran Associates Inc., Red Hook, NY, USA (2016)

[28] Saha, T.K., Williams, T., Hasan, M.A., Joty, S., Varberg, N.K.: Models for Capturing Temporal Smoothness in Evolving Networks for Learning Latent Representation of Nodes. arXiv (2018). https://doi.org/10.48550/ARXIV.1804.05816

[29] Kumar, S., Zhang, X., Leskovec, J.: Predicting dynamic embedding trajectory in temporal interaction networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1269–1278. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3292500.3330895

[30] Kutuzov, A.: Distributional word embeddings in modeling diachronic semantic change. PhD thesis, University of Oslo (2020). http://urn.nb.no/URN:NBN:no-84130

[31] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735

[32] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation, 1724–1734 (2014). https://doi.org/10.3115/v1/D14-1179

[33] Bahdanau, D., Cho, K., Bengio, Y.: Neural Machine Translation by Jointly Learning to Align and Translate. arXiv:1409.0473 [cs, stat] (2014). https://doi.org/10.48550/ARXIV.1409.0473

[34] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 6000–6010. Curran Associates Inc., Red Hook, NY, USA (2017)

[35] Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks, pp. 3104–3112. MIT Press, Cambridge, MA, USA (2014)

[36] Wang, L.L., Lo, K., Chandrasekhar, Y., Reas, R., Yang, J., Burdick, D., Eide, D., Funk, K., Katsis, Y., Kinney, R.M., Li, Y., Liu, Z., Merrill, W., Mooney, P., Murdick, D.A., Rishi, D., Sheehan, J., Shen, Z., Stilson, B., Wade, A.D., Wang, K., Wang, N.X.R., Wilhelm, C., Xie, B., Raymond,

D.M., Weld, D.S., Etzioni, O., Kohlmeier, S.: CORD-19: The COVID-19 Open Research Dataset. Association for Computational Linguistics, Online (2020). https://aclanthology.org/2020.nlpcovid19-acl.1

[37] Montani, I., Honnibal, M., Honnibal, M., Landeghem, S.V., Boyd, A., Peters, H., McCann, P.O., jim geovedi, O'Regan, J., Samsonov, M., Altinok, D., Orosz, G., de Kok, D., Kristiansen, S.L., Bournhonesque, R., Kannan, M., Miranda, L., Baumgartner, P., Edward, Bot, E., Hudson, R., Roman, Fiedler, L., Mitsch, R., Daniels, R., Howard, G., Phatthiyaphaibun, W., Tamura, Y., Bozek, S.: explosion/spaCy: v3.4.3: Extended Typer support and bug fixes. Zenodo (2022). https://doi.org/10.5281/zenodo.7310816

[38] Neumann, M., King, D., Beltagy, I., Ammar, W.: ScispaCy: Fast and robust models for biomedical natural language processing. In: Proceedings of the 18th BioNLP Workshop and Shared Task, pp. 319–327. Association for Computational Linguistics, Florence, Italy (2019). https://doi.org/10.18653/v1/W19-5034

[39] Steffens, I.: A hundred days into the coronavirus disease (COVID-19) pandemic. Euro Surveill. **25**(14) (2020)

[40] Sullivan, S.J., Jacobson, R.M., Dowdle, W.R., Poland, G.A.: 2009 H1N1 influenza. Mayo Clin. Proc. **85**(1), 64–76 (2010)

[41] Cucinotta, D., Vanelli, M.: Who declares covid-19 a pandemic. Acta biomedica : Atenei Parmensis **91**(1), 157–160 (2020). https://doi.org/10.23750/abm.v91i1.9397

[42] Tosun, O.K., Eshraghi, A.: Corporate decisions in times of war: Evidence from the russia-ukraine conflict. Finance Research Letters **48**, 102920 (2022). https://doi.org/10.1016/j.frl.2022.102920

[43] Alsentzer, E., Murphy, J., Boag, W., Weng, W.-H., Jindi, D., Naumann, T., McDermott, M.: Publicly available clinical BERT embeddings. In: Proceedings of the 2nd Clinical Natural Language Processing Workshop, pp. 72–78. Association for Computational Linguistics, Minneapolis, Minnesota, USA (2019). https://doi.org/10.18653/v1/W19-1909

[44] Bhargava, P., Drozd, A., Rogers, A.: Generalization in NLI: Ways (Not) To Go Beyond Simple Heuristics. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic (2021). https://doi.org/10.18653/v1/2021.insights-1.18

[45] Turc, I., Chang, M., Lee, K., Toutanova, K.: Well-read students learn better: The impact of student initialization on knowledge distillation. CoRR **abs/1908.08962** (2019). https://doi.org/10.48550/arXiv.1908.08962

[46] Byron, L., Wattenberg, M.: Stacked graphs – geometry & aesthetics. IEEE Transactions on Visualization and Computer Graphics **14**(6), 1245–1252 (2008). https://doi.org/10.1109/TVCG.2008.166

[47] PubMed. U.S. National Library of Medicine, Bethesda, Maryland. https://pubmed.ncbi.nlm.nih.gov/

[48] Booth, H., Rike, D., Witte, G.A.: The National Vulnerability Database (NVD): Overview. NIST Pubs, National Institute of Standards and Technology (December 2013). https://www.nist.gov/publications/national-vulnerability-database-nvd-overview

[49] Verma, R.: US–Taliban peace deal and regional powers as potential spoilers: Iran as a case study. International Politics **59**(2), 260–279 (2022). https://doi.org/10.1057/s41311-021-00302-7

[50] Boni, F.: Afghanistan 2021: Us withdrawal, the taliban return and regional geopolitics. Asia Maior **XXXII**, 375–391 (2022)