Temporal Attention for Language Models

Guy D. Rosin and **Kira Radinsky**
Technion – Israel Institute of Technology, Haifa, Israel
{guyrosin,kirar}@cs.technion.ac.il

Abstract

Pretrained language models based on the transformer architecture have shown great success in NLP. Textual training data often comes from the web and is thus tagged with time-specific information, but most language models ignore this information. They are trained on the textual data alone, limiting their ability to generalize temporally. In this work, we extend the key component of the transformer architecture, i.e., the self-attention mechanism, and propose temporal attention—a time-aware self-attention mechanism. Temporal attention can be applied to any transformer model and requires the input texts to be accompanied with their relevant time points. It allows the transformer to capture this temporal information and create time-specific contextualized word representations. We leverage these representations for the task of semantic change detection; we apply our proposed mechanism to BERT and experiment on three datasets in different languages (English, German, and Latin) that also vary in time, size, and genre. Our proposed model achieves state-of-the-art results on all the datasets.

1 Introduction

Language models (LMs) are usually pretrained on corpora derived from a snapshot of the web crawled at a specific moment in time (Devlin et al., 2019; Liu et al., 2019). But our language is constantly evolving; new words are created, meanings and word usages change. For instance, the COVID-19 pandemic has caused significant changes to our language; consider the new video-related sense of "Zoom" and the new senses recently associated with the word "vaccine".

The "static" nature of existing LMs makes them unaware of time, and in particular unware of language changes that occur over time. This prevents such models from adapting to time and generalizing temporally (Röttger and Pierrehumbert, 2021; Lazaridou et al., 2021; Hombaiah et al., 2021; Dhingra et al., 2022; Agarwal and Nenkova, 2021; Loureiro et al., 2022), abilities that were shown to be important for many tasks in NLP and Information Retrieval (Kanhabua and Anand, 2016; Rosin et al., 2017; Huang and Paul, 2019; Röttger and Pierrehumbert, 2021; Savov et al., 2021). Recently, to create time-aware models, the NLP community has started to use time as a feature in training and fine-tuning language models (Dhingra et al., 2022; Rosin et al., 2022). These two studies achieve this by concatenating a time token to the text sequence before training the models. The former was concerned with temporal question answering, whereas the latter—with semantic change detection and sentence time prediction. In this work, we introduce a new methodology to create time-aware language models and experiment on the task of semantic change detection.

At the heart of the transformer architecture is the self-attention mechanism (Vaswani et al., 2017). This mechanism allows the transformer to capture the complex relationships between words by relating them to each other multiple times. An attention weight has a clear meaning: how much a particular word will be weighted when computing the next representation for the current word (Clark et al., 2019). This mechanism also enables the above-mentioned temporal models (Dhingra et al., 2022; Rosin et al., 2022) to work; by concatenating time-specific tokens to the text sequences, the self-attention mechanism would compute the relationships between them and the original tokens in the texts, effectively making the output embeddings time-aware (as the output embeddings will depend on the concatenated time tokens).

In this work, instead of changing the text sequences as in prior work, we modify the model itself and specifically the attention mechanism to make it time-aware. We propose a time-aware self-attention mechanism that is an extension of the

self-attention mechanism of the transformer. It considers the time the text sequences (or documents) were written when computing attention scores. As described above, self-attention captures relationships between words. We want to condition these relationships on time. By adding a time matrix as an additional input to the self-attention (along with the standard query, key, and value matrices), we condition the attention weights on the time. In other words, the adapted mechanism also considers the time when calculating the weights of each word. We refer to this adapted attention as *Temporal Attention* (Section 3.2). See Figure 1 for an illustration of our proposed mechanism.

We experiment on the task of semantic change detection — the task of identifying which words undergo semantic changes and to what extent. Semantic change detection methods are used in historical linguistics and digital humanities to study the evolution of word meaning over time and in different domains (Kutuzov et al., 2018). Most existing contextual methods detect changes by first embedding the target words in each time point and then either aggregating them to create a time-specific embedding (Martinc et al., 2020a), or computing a cluster of the embeddings for each time (Giulianelli et al., 2020; Martinc et al., 2020b; Montariol et al., 2021; Laicher et al., 2021). The embeddings or clusters are compared to estimate the degree of change between different times. We experiment with several diverse datasets in terms of time, language, size, and genre. Our empirical results show that our model outperforms state-of-the-art methods (Schlechtweg et al., 2019; Martinc et al., 2020a; Montariol et al., 2021; Rosin et al., 2022).

Our contributions are threefold: (1) We introduce a time-aware self-attention mechanism as an extension of the original mechanism of the transformer. The proposed mechanism considers the time the text sequences were written. The time is considered during the computation of attention scores, thus allowing to create time-specific contextualized word representations; (2) We conduct evaluations on the task of semantic change detection and reach state-of-the-art performance on three diverse datasets in terms of time, language, size, and genre; (3) We contribute our code and trained models to the community for further research.[1]

[1] https://github.com/guyrosin/temporal_attention

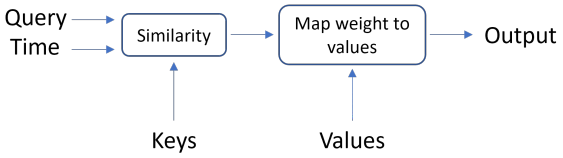

Figure 1: High-level illustration of our proposed temporal attention mechanism.

2 Related Work

2.1 Temporal Language Models

Several recent studies have explored and evaluated the generalization ability of language models to time (Röttger and Pierrehumbert, 2021; Lazaridou et al., 2021; Agarwal and Nenkova, 2021; Hofmann et al., 2021; Loureiro et al., 2022). To better handle continuously evolving web content, Hombaiah et al. (2021) performed incremental training. Dhingra et al. (2022) experimented with temporal language models for question answering. They focused on temporally-scoped facts and showed that conditioning temporal language models on the temporal context of textual data improves memorization of facts. Rosin et al. (2022) similarly concatenated time tokens to text sequences and introduced the concept of time masking (specific masking for the added time tokens). They focused on two temporal tasks: semantic change detection and sentence time prediction. Others focused on document classification by using word-level temporal embeddings (Huang and Paul, 2019) and adapting pretrained BERT models to domain and time (Röttger and Pierrehumbert, 2021). Recently, Hofmann et al. (2021) jointly modeled temporal and social information by changing the architecture of BERT and connecting embeddings of adjacent time points via a latent Gaussian process.

In this work, we create a temporal LM by adapting the transformer's self-attention mechanism to time. The model receives each text sequence along with its writing time and uses both as input to the temporal attention mechanism. As a result, the model creates time-specific contextualized word embeddings.

2.2 Semantic Change Detection

Semantic change detection is the task of identifying words that change meaning over time (Kutuzov et al., 2018; Tahmasebi et al., 2018). This task is often addressed using time-aware word representations that are learned from time-annotated corpora

and then compared between different time points (Jatowt and Duh, 2014; Kim et al., 2014; Kulkarni et al., 2015; Hamilton et al., 2016; Dubossarsky et al., 2019; Del Tredici et al., 2019). Gonen et al. (2020) used a simple nearest-neighbors-based approach to detect semantically-changed words. Others learned time-aware embeddings simultaneously over all time points to resolve the alignment problem, by regularization (Yao et al., 2018), modeling word usage as a function of time (Rosenfeld and Erk, 2018), Bayesian skip-gram (Bamler and Mandt, 2017), or exponential family embeddings (Rudolph and Blei, 2018).

All aforementioned methods limit the representation of each word to a single meaning, ignoring the ambiguity in language and limiting their sensitivity. Recent contextualized models (e.g., BERT (Devlin et al., 2019)) overcome this limitation by taking sentential context into account when inferring word token representations. Such models were applied to diachronic semantic change detection, where most detect changes by creating time-specific embeddings or computing a cluster of the embeddings for each time, and then comparing these embeddings or clusters to estimate the degree of change between different times (Hu et al., 2019; Martinc et al., 2020b,a; Giulianelli et al., 2020; Laicher et al., 2021; Montariol et al., 2021). Recently, Rosin et al. (2022) suggested another approach of detecting semantic change through predicting the writing time of sentences. In our work, we use language models to create time-specific word representations and compare them to detect semantic change. While the above studies used language models as is, we modify their inner workings to make them time-aware by adapting the self-attention mechanism to time.

3 Model

Our model adopts a multi-layer bidirectional transformer (Vaswani et al., 2017). It treats words in the document as input tokens and computes a representation for each token. Formally, given a sequence of n words $w_1, w_2, \ldots, w_n$, the transformer computes D-dimensional word representations $x_1, x_2, \ldots, x_n \in \mathbb{R}^D$.

3.1 Self-Attention

The self-attention mechanism is the foundation of the transformer (Vaswani et al., 2017). It relates tokens to each other based on the attention score between each pair of tokens. In practice, the attention function is computed on a set of tokens simultaneously; our input sequence is packed together into a matrix $X \in \mathbb{R}^{n \times D}$, in which each row i corresponds to a word representation x_i in the input sentence. We denote three trainable weight matrices by $W_Q, W_K, W_V \in \mathbb{R}^{D \times d_k}$. We then create three distinct representations, i.e., query, key, and value: $Q = XW_Q$, $K = XW_K$, $V = XW_V$, respectively, where $Q, K, V \in \mathbb{R}^{n \times d_k}$.

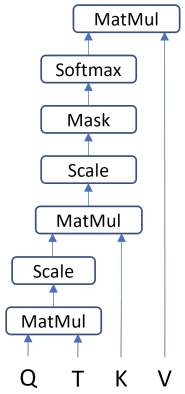

Figure 2: Illustration of our proposed temporal attention mechanism.

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and outputs are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is determined by the dot product of the query with all the keys:

$$\textit{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\intercal}{\sqrt{d_k}}\right)V \quad (1)$$

3.2 Temporal Attention

We now describe the temporal attention mechanism. In the temporal setting, similarly to the vocabulary of the model, our model has a vocabulary of time points. Theoretically, each token in an input sequence could have its own time point, but we simplify and assume the most common case where text sequences always refer to a single time point t.[2] Given a sequence of n words $w_1, w_2, \ldots, w_n$ and its corresponding time point t, our model computes D-dimensional time-specific word representations $x_1^t, x_2^t, \ldots, x_n^t$, where $x_i^t \in \mathbb{R}^D$. As a by-product,

[2]Our mechanism also supports the setting where different tokens in a sequence are associated with different time points.

we also compute D-dimensional time representations for the time points. Now, similarly to the input matrix X (Section 3.1), we define an embedding matrix $X^t \in \mathbb{R}^{n \times D}$ where each row i contains the embedding vector of x_i's time point.[3]

To incorporate time in the attention mechanism, we use an additional trainable weight matrix $W_T \in \mathbb{R}^{D \times d_k}$ and create its corresponding representation matrix $T = X^t W_T$. Note $T \in \mathbb{R}^{n \times d_k}$, i.e., its dimensions are the same as the key, query, and value matrices.

To calculate the attention scores, we multiply the query matrix by the time matrix and then multiply by its transposed matrix, to keep the dimensions intact. We then divide by the time matrix's norm, to avoid getting too large values. Formally, we define temporal attention by:

$$\textit{TemporalAttention}(Q, K, V, T) = \\ \text{softmax}\left(\frac{Q\frac{T^\intercal T}{\|T\|}K^\intercal}{\sqrt{d_k}}\right)V \tag{2}$$

Intuitively, by multiplying the query by the time, the attention weights are now conditioned on the time, i.e., they are time-dependent.

Temporal attention can be used together with other, existing temporal language models, such as (Rosin et al., 2022; Dhingra et al., 2022). In these two models, a time-specific token is prepended to each sentence. In comparison to those methods, our approach does not require changing the input text, as it only modifies the attention mechanism of the language model. We further discuss and compare the two methods in Section 3.3.

The temporal attention mechanism requires each input text to be accompanied with a time point. There are no constraints on these time points, i.e., the mechanism is agnostic to the time granularity and the number of time points in the model.

3.3 Theoretical Analysis

We now theoretically analyze the temporal attention mechanism more deeply and compare it to existing time concatenation methods (Dhingra et al., 2022; Rosin et al., 2022). We omit the scaling factor $\sqrt{d_k}$ for readability.

We denote the row vectors of the matrices Q, K, V, and T by q_i, k_i, v_i, and t_i, respectively. The attention head computes attention weights α between all pairs of words as softmax-normalized dot products between the query and key vectors:

$$\alpha_{ij} = \text{softmax}\left(q_i k_j^\intercal\right) \tag{3}$$

where $i, j \in \{1, \ldots, n\}$.

The output y_i of the attention head is a weighted sum of the value vectors:

$$y_i = \sum_{j=1}^{n} \alpha_{ij} v_j = \sum_{j=1}^{n} \text{softmax}\left(q_i k_j^\intercal\right) v_j \tag{4}$$

Baseline models, such as Rosin et al. (2022) and Dhingra et al. (2022), prepend the text sequence with a time token at index 0, resulting in:

$$y_i = \sum_{j=0}^{n} \alpha_{ij} v_j = \sum_{j=1}^{n} \alpha_{ij} v_j + \text{softmax}\left(q_0 k_0^\intercal\right) v_0 \tag{5}$$

As we can see, by concatenating the time token, we add query, key, and value vectors for that token, i.e., a time component is added to the weighted sum.

In contrast, by using temporal attention, the attention weights become:

$$\alpha_{ij} = \text{softmax}\left(q_i \frac{t_i t_j^\intercal}{\|T\|} k_j^\intercal\right) \tag{6}$$

The i-th output vector y_i is computed as:

$$y_i = \sum_{j=1}^{n} \alpha_{ij} v_j = \sum_{j=1}^{n} \text{softmax}\left(q_i \frac{t_i t_j^\intercal}{\|T\|} k_j^\intercal\right) v_j \tag{7}$$

Intuitively, we multiply by the vectors of time to scale the attention weight α_{ij} by time. We observe two main differences between our proposed mechanism and prior work:

1. The time component is more tightly integrated in temporal attention: instead of just adding a time component to the weighted sum, in temporal attention the time component is multiplied by every component in the sum.

2. Temporal attention requires learning an additional weight matrix $W_T \in \mathbb{R}^{D \times d_k}$. In prior work, each input sequence is prepended with a time token, i.e., its length n is increased by 1. As a result, the temporal attention mechanism consumes more memory (as it has additional $D \cdot d_k$ trainable parameters[4]), whereas prior

[3] Most tokens share the same time point, as noted above, except for special tokens such as padding and masking tokens, to which we associate unique time points.

[4] When using the standard BERT-base architecture: $D \cdot d_k = 768 \cdot 64 = 49{,}152$

work requires more time to train (as its sequences are longer). From our experiments, the overhead of both methods is negligible compared to the memory consumption and training time of BERT (see analysis in Section 6.3).

4 Semantic Change Detection

In this section, we employ our proposed temporal attention mechanism (Section 3.2) for the task of semantic change detection (Kutuzov et al., 2018; Tahmasebi et al., 2018). The ability to detect and quantify semantic changes is important to lexicography, linguistics, and is a basic component in many NLP tasks. For example, search in temporal corpora, historical sentiment analysis, and understanding historical documents. The objective of this task is to rank a set of target words according to their degree of semantic change between two time points t_1 and t_2. In this work, we follow the practice of (Martinc et al., 2020a; Rosin et al., 2022) to estimate the semantic change a word underwent and rank the target words based on these estimates.

Given a target word w, we generate time-specific representations of it and compare them to detect semantic changes. Algorithm 1 formally describes the method. We begin by sampling n sentences containing w from each time point $t \in \{t_1, t_2\}$ (line 3). For each sentence $sent$, we create a sequence embedding by running it through the temporal attention model (note the model receives as input both $sent$ and t) and extracting the model's hidden layers that correspond to w (lines 5–6). We then choose the last h hidden layers and average them to get a single vector (line 7). This is the contextual word embedding of w, denoted by v. Following, the resulting embeddings are aggregated at the token level and averaged (line 10), in order to create a non-contextual time-specific representation for w for each time t, denoted by x_t. Finally, we estimate the semantic change of w by measuring the cosine distance (*cos_dist*) between two time-specific representations of the same token (line 12).

Algorithm 1: Semantic change estimation

Input: w (target word)
Input: t_1 (first time point)
Input: t_2 (last time point)
Input: C (diachronic corpus)
Input: n (# of sentences to sample)
Input: h (# of last hidden layers to extract)

1 **for** $t \in \{t_1, t_2\}$ **do**
2 $\quad L_t \leftarrow \{\}$
3 $\quad S_w \leftarrow n$ sentences sampled from $C(t, w)$
4 $\quad$ **for** $sent \in S_w$ **do**
5 $\quad\quad H \leftarrow \textit{TempAttModel}(sent, t)$
6 $\quad\quad H_w \leftarrow H[w]$
7 $\quad\quad v \leftarrow \textit{AvgHiddenLayers}(H_w, h)$
8 $\quad\quad L_t.\textit{insert}(v)$
9 $\quad$ **end**
10 $\quad x_t \leftarrow \textit{avg}(L_t)$
11 **end**
12 $score = \textit{cos\_dist}(x_{t_1}, x_{t_2})$
13 **return** $score$

5 Experimental Setup

5.1 Data

To train and evaluate our models, we use data from the SemEval-2020 Task 1 on Unsupervised Detection of Lexical Semantic Change (Schlechtweg et al., 2020). We use corpora provided by this task for English, German, and Latin, covering a variety of genres, times, languages, and sizes. They are all long-term: the English and German corpora span two centuries each, and the Latin corpus spans more than 2000 years. The German corpus is much larger than the other two (7x – 10x). Each corpus is genre-balanced, and split into two time points; see Table 1 for their statistics.

Each corpus is accompanied with labeled data for semantic change evaluation. We use the data from Subtask 2 of this task, where the objective is to rank a set of target words according to their degree of semantic change between t_1 and t_2. The provided data is a set of target words that are either words that changed their meaning(s) (lost or gained a sense) between the two time points, or stable words that did not change their meaning during that time. The target words are balanced for part of speech (POS) and frequency. Each target word was assigned a graded label (between 0 and 1) according to their degree of semantic change (0 means no change, 1 means total change). For the English dataset, we follow (Montariol et al., 2021) and remove POS tags from both the corpus and the evaluation set.

5.2 Baseline Methods

We use the following baseline methods:

Corpus	C1 Source	C1 Time	C1 Tokens	C2 Source	C2 Time	C2 Tokens	Target Words
SemEval-English	CCOHA	1810–1860	6.5M	CCOHA	1960–2010	6.7M	37
SemEval-Latin	LatinISE	-200–0	1.7M	LatinISE	0–2000	9.4M	40
SemEval-German	DTA	1800–1899	70.2M	BZ, ND	1946–1990	72.4M	48

Table 1: Corpora for semantic change detection. Each corpus is split into two time points, denoted by C1 and C2.

1. Schlechtweg et al. (2019) train Skip-gram with Negative Sampling (SGNS) on two time points independently and align the resulting embeddings using Orthogonal Procrustes. They compute the semantic change scores using cosine distance.

2. Gonen et al. (2020) use SGNS embeddings as well. They represent a word in a time point by its top nearest neighbors according to cosine distance. Then, they measure semantic change as the size of intersection between the nearest neighbors lists in the two time points.

3. Martinc et al. (2020a) were one of the first to use BERT for semantic change detection. They create time-specific embeddings of words by averaging token embeddings over sentences in each time point, and then compare them by calculating cosine distance.

4. Montariol et al. (2021) use BERT to create a set of contextual embeddings for each word. They cluster these embeddings and then compare the cluster distributions across time slices using various distance measures. We use their best-performing method for each dataset as reported in the paper, which uses affinity propagation for clustering word embeddings and either Wasserstein or Jensen-Shannon distance as a distance measure between clusters.

5. Rosin et al. (2022) create a time-aware BERT model by preprocessing input texts to concatenate time-specific tokens to them, and then masking these tokens while training. They introduce two methods to measure semantic change, namely temporal-cosine and time-diff. We use their best-performing method as reported in the paper, which is temporal-cosine.

6. "Scaled attention": We present several baselines which are simplified versions of our temporal attention mechanism. Intuitively, our mechanism differentiates between different time points by learning a scaling factor per each pair of time points (based on the multiplication of learned time vectors; see Section 3.2). In these baselines, we use a constant scaling factor per time point and calculate attention weights using the following formula:

$$\alpha_{ij} = \text{softmax}\left(q_i s_{ij} k_j^\intercal\right)$$

where s_{ij} is the scaling factor. This scaling method can be seen as a combination of (Martinc et al., 2020a) and our temporal attention method. We present three options for s_{ij}: (1) Linear scaling. We hypothesize that recent texts should be given more weight, and define $s_{ij} = index(t_i)$, where $index(t_i)$ is the index of the time point t_i out of all time points $t_1, \ldots, t_{n_t}$. (2) Exponential scaling: similarly to linear scaling, but using an exponent: $s_{ij} = 2^{index(t_i)}$. (3) Proportional to the number of documents: here we hypothesize that larger corpora should be given more weight, and define $s_{ij} = \frac{doc\_count(t_i)}{\sum_{k=1}^{n_t} doc\_count(t_k)}$, where $doc\_count(t_i)$ is the number of documents in t_i.

5.3 Our Method

To train our models, for each language we use a pretrained BERT (Devlin et al., 2019) model (bert-base-uncased, with 12 layers, 768 hidden dimensions, and 110M parameters) and post-pretrain it on the temporal corpus using our proposed temporal attention, as described in Section 3.2. For semantic change detection, we use the method described in Section 4. We use the Hugging Face's Transformers library[5] for our implementation.

Before training, we add any missing target words to the model's vocabulary. Since a pretrained model's vocabulary may not contain all the target words in our evaluation dataset, this is necessary to avoid the tokenizer splitting any occurrences of the target words into subwords (which we found out to reduce performance). The added words are randomly initialized.

[5] https://github.com/huggingface/transformers

5.4 Metrics

We measure semantic change detection performance by the correlation between the semantic shift index (i.e., the ground truth) and the model's semantic shift assessment for each word in the evaluation set. We follow prior work (Rosin et al., 2022) and use both Pearson's correlation coefficient r and Spearman's rank correlation coefficient ρ. The difference between them is that Spearman's ρ considers only the ranking order, while Pearson's r considers the actual predicted values. In our evaluation, we make an effort to evaluate our methods and the baselines using both correlation coefficients, to make the evaluation as comprehensive as possible. There were some cases where we could not reproduce the original authors' results; in such cases, we opted to report only the original result.

5.5 Implementation Details

Due to limited computational resources, we follow Rosin et al. (2022) and train our models with a maximum input sequence length of 128 tokens. We perform all experiments on a single NVIDIA Quadro RTX 6000 GPU. We tune the following hyperparameters for each language: for training: learning rate in {1e-8, 1e-7, 1e-6, 1e-5, 1e-4} and number of epochs in $\{1, 2, 3, 4\}$. For inference: number of last hidden layers to use for embedding extraction $h \in \{1, 2, 4, 12\}$.

The chosen pretrained model and hyperparameters, along with the steps number and training time per language are as follows:

- For English: bert-base-uncased,[6] with 1e-9 learning rate for 2 epochs (6.3K steps, took 70 minutes); all (12) hidden layers for inference.
- For Latin: latin-bert,[7] with 1e-5 learning rate for 1 epoch (3.5K steps, took 25 minutes); last hidden layer for inference.
- For German: bert-base-german-cased,[8] with 1e-6 learning rate for 1 epoch (38.1K steps, took 10 hours); last hidden layer for inference.

[6] https://huggingface.co/bert-base-uncased

[7] https://github.com/dbamman/latin-bert

[8] https://huggingface.co/bert-base-german-cased

6 Results

In this section, we outline the results of our empirical evaluation. In all tables throughout the section, the best result in each column is boldfaced; performance is measured using Pearson's r and Spearman's ρ correlation coefficients.

6.1 Main Result

Table 2 shows the results for semantic change detection on the SemEval datasets. Our temporal attention model outperforms all the baselines for all datasets and metrics with significant correlations ($p < 0.0005$) and large margins (7%–36%). We observe moderate to strong correlations (around 0.52–0.76) for all datasets. Even for the German dataset, on which recent BERT-based methods got relatively lower results (and were outperformed by word2vec-based methods such as Schlechtweg et al. (2019)), our model achieves strong correlations and state-of-the-art performance. In Section 6.2 and Section 6.3, we experiment with variations of our method and achieve even stronger performance on the English dataset.

Finally, looking at the three scaled attention baselines, they all perform similarly and are positioned between Martinc et al. (2020a) and our temporal attention model, as expected.

6.2 Temporal Attention with Temporal Prepend

Until now, we used temporal attention on BERT (Devlin et al., 2019) to create our model. In this section, in addition to using temporal attention, we also prepend a time token to the input sequences, as done in Rosin et al. (2022). That is, we experiment with applying temporal attention on top of their model.

Table 3 shows the results of this combined model compared to each of its components. First, prepending time tokens is inferior to the other models. When comparing our proposed temporal attention and the combined model, we observe mixed results: temporal attention alone works better for the Latin and German datasets, but for the English dataset the combination of temporal attention and prepending time tokens performs better.

6.3 Impact of BERT Model Size on Temporal Attention

Our model is based on the most commonly used pretrained BERT model, called BERT-base, which

Method	SemEval-Eng		SemEval-Lat		SemEval-Ger	
	r	ρ	r	ρ	r	ρ
Schlechtweg et al. (2019)	0.512	0.321	0.458	0.372	–	0.712
Gonen et al. (2020)	0.504	0.277	0.417	0.273	–	0.627
Martinc et al. (2020a)	–	0.315	–	0.496	–	0.565
Montariol et al. (2021)	0.566	0.456	–	0.488	0.618	0.583
Rosin et al. (2022)	0.538	0.467	0.485	0.512	0.592	0.582
Scaled Linear Attention	0.517	0.506	0.524	0.478	0.580	0.550
Scaled Exp. Attention	0.491	0.487	0.633	0.528	0.569	0.526
Scaled by Doc Attention	0.532	0.478	0.657	0.505	0.595	0.567
Temporal Attention	**0.620**	**0.520**	**0.661**	**0.565**	**0.767**	**0.763**

Table 2: Semantic change detection results on SemEval-English, SemEval-Latin, and SemEval-German, measured using Pearson's r and Spearman's ρ correlation coefficients.

Method	SE-Eng		SE-Lat		SE-Ger	
	r	ρ	r	ρ	r	ρ
Temp. Prep.	0.538	0.467	0.485	0.512	0.592	0.582
Temp. Att.	0.620	0.520	**0.556**	**0.556**	**0.767**	**0.763**
Both	**0.655**	**0.548**	0.541	0.508	0.645	0.682

Table 3: Semantic change detection results on the English, Latin, and German datasets, comparing time token prepending (Rosin et al., 2022) with our proposed temporal attention, and a combination of both.

Method	Params	r	ρ
Rosin et al. (2022) base	109.52M	0.538	0.467
Rosin et al. (2022) tiny	4.42M	0.534	0.427
Temp. Att. base	116.61M	0.620	0.520
Temp. Att. small	29.85M	0.660	0.584
Temp. Att. tiny	4.45M	**0.703**	**0.627**

Table 4: Results for semantic change detection for models of different sizes on SemEval-English.

contains 12 transformer layers and a hidden dimension size of 768. In this section, we train and evaluate models of different sizes, namely 'small' and 'tiny', that are based on much smaller pretrained variants of BERT: BERT-small[9] has 26% of the parameters of BERT-base, containing only 4 transformer layers while its hidden dimension is 512; BERT-tiny[10] has just 4% of the parameters of BERT-base, with 2 transformer layers and a hidden dimension of 128. We perform this evaluation only for the SemEval-English dataset, as smaller pretrained BERT models are currently publicly available only for the English language.

Table 4 shows the comparison results, where we compare the three variants of our temporal attention model, along with the two variants of Rosin et al. (2022). We also denote the number of trainable parameters for each model (see the theoretical analysis in Section 3.3). We observe a clear negative correlation between model size and performance (measured by both Pearson's r and Spearman's ρ); the smaller the model, the better the performance. While this finding may sound counterintuitive, it is in line with Rosin et al. (2022), who hypothesized that to understand time there is no need to use extremely large models, and reported higher-than-expected performance for the tiny model. In their study, that model achieved a slightly lower performance compared to their standard (base) model, but still outperformed most baselines. Overall, this is an encouraging finding; smaller models mean faster training and inference times, as well as smaller memory footprints. This lowers the bar to enter the field.

6.4 Qualitative Analysis

Figure 3 shows the Spearman correlation between the ground truth ranks and our model's ranks for the SemEval-English dataset. The correlation is moderate (0.520), and we observe a similar number of false-positive words (top-left corner) and false negatives (bottom-right corner). Interestingly, we can see that the model performs better on the more changed words (right half, rank above 19, e.g., "plane", "tip", and "head"), while there are more errors on the static words (left half, e.g., "chair-

[9] https://huggingface.co/prajjwal1/bert-small

[10] https://huggingface.co/prajjwal1/bert-tiny

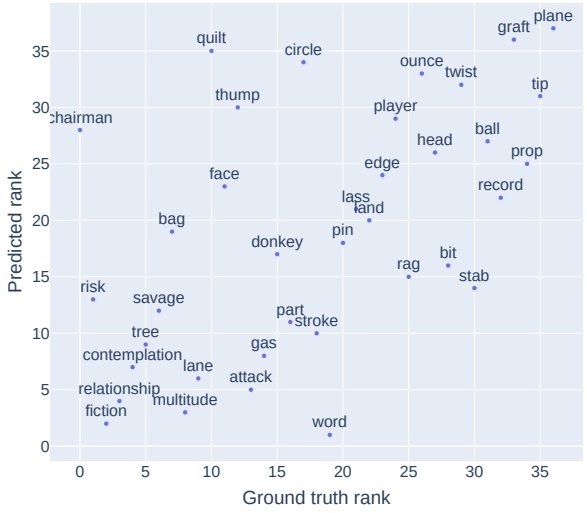

Figure 3: Semantic change detection on the SemEval-English dataset: ground truth ranks vs. our model's ranks (Spearman's $\rho = 0.520$).

man", "risk", and "quilt"). Most of the false positives seem to be either slang words or concerning word usages that are less likely to appear in our corpora which is mainly composed of newsletters and books (Section 5.1). For example, the verb "stab", which traditionally means to push a knife into someone, has a newer meaning of attempting to do something. The noun "word" can be used to express agreement.

7 Conclusion

In this paper, we presented a time-aware self-attention mechanism as an extension of the original mechanism of the transformer. The proposed mechanism considers the time the text sequences were written when computing attention scores, thus allowing creating time-specific contextualized word representations. We conducted evaluations on the task of semantic change detection and reached state-of-the-art performance on three diverse datasets in terms of time, language, size, and genre. In addition, we experimented with small-sized pretrained models and found they outperform larger models on this task. We conduct an experiment evaluating the marginal addition of time token prepending along with temporal attention and conclude that on all but the English dataset it hurts performance. We wish to study how to best combine the two approaches in future work. Additionally, for future work, we plan to extend this work by applying temporal attention to other tasks, such as web search and sentence time prediction, as well as experimenting with more time points and different granularities.

References

Oshin Agarwal and Ani Nenkova. 2021. Temporal effects on pre-trained models for language processing tasks. *arXiv preprint arXiv:2111.12790*.

Robert Bamler and Stephan Mandt. 2017. Dynamic word embeddings. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 380–389. PMLR.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Marco Del Tredici, Raquel Fernández, and Gemma Boleda. 2019. Short-term meaning shift: A distributional exploration. 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)*, pages 2069–2075, Minneapolis, Minnesota. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Bhuwan Dhingra, Jeremy R Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W Cohen. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273.

Haim Dubossarsky, Simon Hengchen, Nina Tahmasebi, and Dominik Schlechtweg. 2019. Time-out: Temporal referencing for robust modeling of lexical semantic change. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 457–470, Florence, Italy. Association for Computational Linguistics.

Mario Giulianelli, Marco Del Tredici, and Raquel Fernández. 2020. Analysing lexical semantic change with contextualised word representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3960–3973, Online. Association for Computational Linguistics.

Hila Gonen, Ganesh Jawahar, Djamé Seddah, and Yoav Goldberg. 2020. Simple, interpretable and stable method for detecting words with usage change across corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 538–555, Online. Association for Computational Linguistics.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1489–1501, Berlin, Germany. Association for Computational Linguistics.

Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. 2021. Dynamic contextualized word embeddings. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6970–6984, Online. Association for Computational Linguistics.

Spurthi Amba Hombaiah, Tao Chen, Mingyang Zhang, Michael Bendersky, and Marc Najork. 2021. Dynamic language models for continuously evolving content. In *KDD 2021*.

Renfen Hu, Shen Li, and Shichen Liang. 2019. Diachronic sense modeling with deep contextualized word embeddings: An ecological view. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3899–3908, Florence, Italy. Association for Computational Linguistics.

Xiaolei Huang and Michael J. Paul. 2019. Neural temporality adaptation for document classification: Diachronic word embeddings and domain adaptation models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4113–4123, Florence, Italy. Association for Computational Linguistics.

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

Nattiya Kanhabua and Avishek Anand. 2016. Temporal information retrieval. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 1235–1238. ACM.

Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 61–65, Baltimore, MD, USA. Association for Computational Linguistics.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 625–635. ACM.

Andrey Kutuzov, Lilja Øvrelid, Terrence Szymanski, and Erik Velldal. 2018. Diachronic word embeddings and semantic shifts: a survey. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1384–1397, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Severin Laicher, Sinan Kurtyigit, Dominik Schlechtweg, Jonas Kuhn, and Sabine Schulte im Walde. 2021. Explaining and improving BERT performance on lexical semantic change detection. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 192–202, Online. Association for Computational Linguistics.

Angeliki Lazaridou, Adhi Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, et al. 2021. Mind the gap: Assessing temporal generalization in neural language models. *Advances in Neural Information Processing Systems*, 34.

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

Daniel Loureiro, Francesco Barbieri, Leonardo Neves, Luis Espinosa Anke, and Jose Camacho-Collados. 2022. Timelms: Diachronic language models from twitter. *arXiv preprint arXiv:2202.03829*.

Matej Martinc, Petra Kralj Novak, and Senja Pollak. 2020a. Leveraging contextual embeddings for detecting diachronic semantic shift. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4811–4819, Marseille, France. European Language Resources Association.

Matej Martinc, Syrielle Montariol, Elaine Zosa, and Lidia Pivovarova. 2020b. Capturing evolution in word usage: Just add more clusters? In *Companion Proceedings of the Web Conference 2020*, pages 343–349.

Syrielle Montariol, Matej Martinc, and Lidia Pivovarova. 2021. Scalable and interpretable semantic change detection. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4642–4652, Online. Association for Computational Linguistics.

Alex Rosenfeld and Katrin Erk. 2018. Deep neural models of semantic shift. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 474–484, New Orleans, Louisiana. Association for Computational Linguistics.

Guy D. Rosin, Eytan Adar, and Kira Radinsky. 2017. Learning word relatedness over time. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1168–1178, Copenhagen, Denmark. Association for Computational Linguistics.

Guy D. Rosin, Ido Guy, and Kira Radinsky. 2022. Time masking for temporal language models. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 833–841.

Paul Röttger and Janet Pierrehumbert. 2021. Temporal adaptation of BERT and performance on downstream document classification: Insights from social media. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2400–2412, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Maja R. Rudolph and David M. Blei. 2018. Dynamic embeddings for language evolution. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 1003–1011. ACM.

Pavel Savov, Adam Jatowt, and Radoslaw Nielek. 2021. Predicting the age of scientific papers. In *International Conference on Computational Science*, pages 728–735. Springer.

Dominik Schlechtweg, Anna Hätty, Marco Del Tredici, and Sabine Schulte im Walde. 2019. A wind of change: Detecting and evaluating lexical semantic change across times and domains. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 732–746, Florence, Italy. Association for Computational Linguistics.

Dominik Schlechtweg, Barbara McGillivray, Simon Hengchen, Haim Dubossarsky, and Nina Tahmasebi. 2020. SemEval-2020 task 1: Unsupervised lexical semantic change detection. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1–23, Barcelona (online). International Committee for Computational Linguistics.

Nina Tahmasebi, Lars Borin, and Adam Jatowt. 2018. Survey of computational approaches to lexical semantic change. *arXiv preprint arXiv:1811.06278*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. 2018. Dynamic word embeddings for evolving semantic discovery. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, pages 673–681. ACM.