Learning Dynamic Contextualised Word Embeddings via Template-based Temporal Adaptation

Xiaohang Tang[†] **Yi Zhou**[*] **Danushka Bollegala**[†,‡]
University of Liverpool[†], Cardiff University[*], Amazon[‡]
{sgxtang4, danushka}@liverpool.ac.uk
zhouy131@cardiff.ac.uk

Abstract

Dynamic contextualised word embeddings (DCWEs) represent the temporal semantic variations of words. We propose a method for learning DCWEs by time-adapting a pretrained Masked Language Model (MLM) using time-sensitive templates. Given two snapshots C_1 and C_2 of a corpus taken respectively at two distinct timestamps T_1 and T_2, we first propose an unsupervised method to select (a) *pivot* terms related to both C_1 and C_2, and (b) *anchor* terms that are associated with a specific pivot term in each individual snapshot. We then generate prompts by filling manually compiled templates using the extracted pivot and anchor terms. Moreover, we propose an automatic method to learn time-sensitive templates from C_1 and C_2, without requiring any human supervision. Next, we use the generated prompts to adapt a pretrained MLM to T_2 by fine-tuning using those prompts. Multiple experiments show that our proposed method reduces the perplexity of test sentences in C_2, outperforming the current state-of-the-art.

1 Introduction

Contextualised word embeddings produced by MLMs (Devlin et al., 2019; Liu et al., 2019; Lan et al., 2020; Yang et al., 2019) represent the meaning of a word with respect to the context in which it appears in and have reported substantial performance gains in various NLP tasks. The usage of words change over time and the same word might be associated with different words to mean different concepts over time (Koch, 2016; Baybee, 2015; Rama and Borin, 2015). For example, the word *gay* has gradually changed its meaning from *happy* to *homosexual* over the last five decades (Robinson, 2012; Campbell, 2004). However, MLMs are often trained using a static snapshot of a corpus taken at a specific timestamp, and are *not updated* afterwards. Because of this reason, existing pretrained MLMs do *not* capture the temporal semantic variations of words. For example, Loureiro et al. (2022) showed that neither the original version of BERT (Devlin et al., 2019) nor RoBERTa (Liu et al., 2019) are up-to-date with the information related to the current coronavirus pandemic.

To address the above-mentioned limitations, we propose a Dynamic Contextualised Word Embedding (**DCWE**) method that *adapts* a given pretrained MLM from one timestamp T_1 to another T_2 using two snapshots of a corpus C_1 and C_2, sampled respectively at times T_1 and T_2. We represent a word x by an embedding that depends both on the **context** c of x, as well as on **time** T. Our word embeddings are *dynamic* because they depend on the time, and *contextualised* because they also depend on the context.

We model the problem of adapting a given pretrained MLM to a specific timestamp T_2 as an instance of prompt-based fine-tuning (Liu et al., 2021), which has been successfully used in prior work to adapt MLMs to various tasks such as relation representation (Ushio et al., 2021; Fichtel et al., 2021), domain adaptation (Ben-David et al., 2021), natural language inference (Utama et al., 2021) and question answering (Qin and Eisner, 2021). Compared to fine-tuning MLMs on manually labelled training instances, which might not be readily available or costly to manually annotate in sufficient quantities for a particular task to fine-tune a large-scale MLM, prompt-based methods require only a small number of prompts (Le Scao and Rush, 2021). Luckily, in our case of temporal adaptation of MLMs (Agarwal and Nenkova, 2022), such prompts could be generated from a handful of manually designed templates (§3.1) or automatically extracted from unlabelled data (§3.3). This aspect of our proposed method is particularly attractive compared to prior work (see §2) on DWEs (Rudolph and Blei, 2018; Hofmann et al., 2021; Qiu and Xu, 2022; Loureiro et al., 2022) that require retraining of MLMs from scratch to incorporate the time-

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 9352–9369

sensitive constraints into the embedding spaces.

We first extract *pivot* words, w, that are common to both C_1 and C_2. Second, we extract *anchor* words u and v that are strongly associated with w in respectively C_1 and C_2. We then propose several methods to score tuples (w, u, v) such that the semantic variation of w from T_1 to T_2 is captured by its association with respectively u and v. Finally, we generate a large number of textual prompts using the top-scoring tuples (w, u, v) according to each method to fill the slots in manually written templates such as "$\langle w \rangle$ *is associated with* $\langle u \rangle$ *in* $\langle T_1 \rangle$*, whereas it is associated with* $\langle v \rangle$ *in* $\langle T_2 \rangle$." Here, the slots corresponding to T_1 and T_2 are filled by specific years when respectively C_1 and C_2 were sampled. We differentiate *templates* from *prompts* throughout the paper where the latter is formed by filling one or more slots in the former. We further propose a method to automatically generate templates from sentences selected from C_1 and C_2 using a text-to-text transformation model (Raffel et al., 2020), thereby obviating the need to manually create templates. Finally, the given MLM is adapted to T_2 by fine-tuning it on the generated prompts.

Experimental results conducted on Reddit, Yelp, ArXiv and Ciao datasets show that the proposed prompt-based time-adapting of MLMs consistently outperforms previously proposed DCWEs (Hofmann et al., 2021) and temporal adaptation methods (Rosin et al., 2022) reporting better (lower) perplexity scores on unseen test sentences in C_2. The source code for our proposed method is publicly available.[1]

2 Related Work

Methods that use part-of-speech (Mihalcea and Nastase, 2012), entropy (Tang et al., 2016), latent semantic analysis (Sagi et al., 2011) and temporal semantic indexing (Basile et al., 2014) have been proposed for detecting changes in word meanings. In SemEval-2020 Task 1 (Schlechtweg et al., 2020) two subtasks were proposed for detecting lexical semantic change: a binary classification task (for a given set of target words, decide which words had their meaning altered, and which ones not) and a ranking task (rank a set of target words according to their degree of lexical semantic change between the two corpora). Giulianelli et al. (2020) showed that contextualised embeddings obtained from an MLM can be used to measure the change of word meaning. Rosin and Radinsky (2022) proposed a temporal attention mechanism by extending the self-attention mechanism in transformers, where time stamps of the documents are considered when computing the attention scores. Aida and Bollegala (2023) proposed a method to predict semantic change of words by comparing the distributions of contextualised embeddings of the word between two given corpora, sampled at different points in time. Our goal in this paper extends beyond the detection of a subset of words with a change in lexical semantics, and to adapt MLMs over time.

DWEs (Rudolph and Blei, 2018; Hofmann et al., 2021; Qiu and Xu, 2022; Loureiro et al., 2022) incorporate extralinguistic information such as time, demographic or social aspects of words with linguistic information. Welch et al. (2020) learnt demographic word embeddings, covering attributes such as age, gender, location and religion. Zeng et al. (2017) learnt *socialised* word embeddings considering both a social media user's personal characteristics of language use and that user's social relationships. However, Hofmann et al. (2021) showed that temporal factors have a stronger impact than socio-cultural factors when determining the semantic variations of words. Consequently, in this paper we focus on the temporal adaptation of DCWEs.

Diachronic Language Models that capture the meanings of words at a particular point in time have been trained using historical corpora (Qiu and Xu, 2022; Loureiro et al., 2022). These prior work learn independent word embedding models from different corpora. This is problematic because information related to a word is not shared across different models resulting in inefficient learning, especially when word occurrences within a single snapshot of a corpus are too sparse to learn accurate embeddings.

Rudolph and Blei (2018) proposed a dynamic Bernoulli embedding method based on exponential family embeddings, where each word is represented by a one-hot vector with dimensionality set to the vocabulary size. This model is extended to the temporal case by considering different time-slices where only the word embedding vector is time-specific and the context vectors are shared across the corpus and over time-slices. Because the joint distribution over time and context is intractable, they maximise the pseudo log-likelihood

[1] https://github.com/LivNLP/TimeAdapted-DCWE

of the conditional distribution for learning the parameters of their DWE model. Ben-David et al. (2021) proposed a domain adaptation method based on automatically learnt prompts. Given a test example, they generate a unique prompt and conditioned on it, then predict labels for test examples. Although their method uses prompts to adapt a model, they do *not* consider temporal adaptation of MLMs, which is our focus. Moreover, we do not require any labelled examples in our proposal.

Amba Hombaiah et al. (2021) proposed a model updating method using vocabulary composition and data sampling to adapt language models to continuously evolving web content. However, their work is specific to one dataset and two classification tasks, and focuses on incremental training. Jang et al. (2022) introduced a benchmark for ever-evolving language models, utilising the difference between consecutive snapshots of datasets, to track language models' ability to retain existing knowledge while incorporating new knowledge at each time point. Jin et al. (2022) studied the lifelong language model pretraining problem, where the goal is to continually update pretrained language models using emerging data. Dhingra et al. (2022) introduced a diagnostic dataset to investigate language models for factual knowledge that changes over time and proposed an approach to jointly model texts with their timestamps. They also demonstrated that models trained with temporal context can be adapted to new data without retraining from scratch. Rosin et al. (2022) proposed TempoBERT, where they insert a special time-related token to each sentence and fine-tune BERT using a customised time masking. TempoBERT reports superior results in SemEval 2020 Task 1 semantic variation detection benchmark. As shown later in §4.1, our proposed method outperforms TempoBERT.

Hofmann et al. (2021) proposed DCWEs, which are computed in two stages. First, words are mapped to dynamic type-level representations considering temporal and social information. The type-level representation of a word is formed by combining a non-dynamic embedding of a word and a dynamic offset that is specific to the social and temporal aspects of the word. Second, these dynamic embeddings are converted to context-dependent token-level representations. To the best of our knowledge, this is the only word embedding method that produces both dynamic as well as contextualised representations, thus mostly relates to us. As shown in §4, our proposed method outperforms their DCWEs on four datasets.

3 Prompt-based Time Adaptation

Given two snapshots C_1 and C_2 of a corpus taken respectively at timestamps T_1 and $T_2 (> T_1)$, we consider the problem of adapting a pretrained MLM M from T_1 to T_2. We refer to a word w that occurs in both C_1 and C_2 but has its meaning altered between the two snapshots as a **pivot**. We propose three methods for selecting tuples (w, u, v), where u is closely associated with the meaning of w in C_1, whereas v is closely associated with the meaning of w in C_2. We name u and v collectively as the **anchors** of w, representing its meaning at T_1 and T_2. If the meaning of w has changed from T_1 to T_2, it will be associated with different sets of anchors, otherwise by similar sets of anchors. We then fine-tune M on prompts generated by substituting (w, u, v) in templates created either manually (§3.1) or automatically (§3.3).

3.1 Prompts from Manual Templates

In order to capture temporal semantic variations of words, we create the template $\langle w \rangle$ *is associated with* $\langle u \rangle$ *in* $\langle T_1 \rangle$, *whereas it is associated with* $\langle v \rangle$ *in* $\langle T_2 \rangle$.[2] We generate multiple prompts from this template by substituting tuples (w, u, v) extracted using three methods as described in § 3.2. For example, given a tuple (*mask*, *hide*, *vaccine*) and $T_1 = 2010$ and $T_2 = 2020$, the previous template produces the prompt: *mask is associated with hide in 2010, whereas it is associated with vaccine in 2020*. These prompts are used in §3.5 to fine-tune an MLM to adapt it to T_2 for obtaining DCWEs.

3.2 Tuple Selection Methods

Given a template with slots corresponding to w, u and v, we propose three different criteria for selecting tuples to fill those slots.

3.2.1 Frequency-based Tuple Selection

Prior work on domain adaptation has shown that words highly co-occurring in both source and target domains are ideal candidates for adapting a model trained on the source domain to the target domain. Following prior work on cross-domain representation learning, we call such words as pivots (Bollegala et al., 2015). Specifically, we measure the

[2]We experimented with multiple manual templates as shown in the Supplementary but did not observe any clear improvements over this template.

suitability of a word w, $\text{score}(w)$ as a pivot by (1).

$$\text{score}(w) = \min(f(w, C_1), f(w, C_2)) \quad (1)$$

Here, $f(w, C_1)$ and $f(w, C_2)$ denote the frequency of w respectively in C_1 and C_2, measured by the number of sentences in which w occurs in each corpus. We sort words in the descending order of the scores given by (1) and select the top k-ranked words as pivots.

Next, for each pivot w, we select its anchors x by the Pointwise Mutual Information, $\text{PMI}(w, x; C)$, computed from the snapshot C as in (2).

$$\text{PMI}(w, x; C) = \log\left(\frac{p(w, x)}{p(w)p(x)}\right) \quad (2)$$

Here, $p(x)$ is the marginal probability of x in C, estimated as $f(x, C)/N_C$, where N_C is the total number of sentences in C. Moreover, $p(w, x)$ is the joint probability between w and x, estimated as $\text{cooc}(w, x)/N_C$, where cooc is the total number of co-occurrences between w and x in C, considering sentences as the contextual window for the co-occurrences.

We select the set of words $\mathcal{U}(w)$ with high $\text{PMI}(w, u; C_1)$ values as the *anchors* of w in C_1. Likewise, the set of words $\mathcal{V}(w)$ with the top-$\text{PMI}(w, v; C_2)$ are selected as the anchors of w in C_2. By construction, anchors are the words that are strongly associated with a pivot in each snapshot of the corpus, thus can be regarded as representing the meaning carried by the pivot in a snapshot according to the distributional hypothesis (Firth, 1957). Finally, for each w, we obtain a set of tuples, $\mathcal{S}_{\text{freq}} = \{(w, u, v) | u \in \mathcal{U}(w), v \in \mathcal{V}(w)\}$, by considering all pairwise combinations of anchors with a pivot for the purpose of filling the templates to generate prompts.

3.2.2 Diversity-based Tuple Selection

Recall that our objective is to select anchors u and v, respectively in C_1 and C_2 such that the change of meaning of a pivot w is captured by the tuple (w, u, v). Frequency-based tuple selection method described in § 3.2.1 finds u and v, which are strongly associated with w in the two snapshots of the corpus. However, if $\mathcal{U}(w)$ and $\mathcal{V}(w)$ are highly similar, it could mean that the meaning of w might not have changed from T_1 to T_2. To address this issue, we define *diversity* of w as the dissimilarity between its sets of anchors as in (3).

$$\text{diversity}(w) = 1 - \frac{|\mathcal{U}(w) \cap \mathcal{V}(w)|}{|\mathcal{U}(w) \cup \mathcal{V}(w)|} \quad (3)$$

Here, $|\mathcal{X}|$ denotes the cardinality of the set $\mathcal{X}$, and the term subtracted from 1 can be identified as the Jaccard coefficient between $\mathcal{U}(w)$ and $\mathcal{V}(w)$. We select the top scoring w according to (1) and re-rank them by (3) to select top-k pivots. Finally, we generate a set of tuples, $\mathcal{S}_{\text{div}}(w) = \{(w, u, v) | u \in \mathcal{U}(w), v \in \mathcal{V}(w)\}$, by pairing each selected pivot w with its anchors in C_1 and C_2 for the purpose of filling the templates to generate prompts.

3.2.3 Context-based Tuple Selection

The anchor words used in both frequency- and diversity-based tuple selection methods use PMI to measure the association between a pivot and an anchor. This approach has two important drawbacks.

First, the number of sentences in a snapshot of a corpus taken at a specific time point could be small. Therefore, the co-occurrences (measured at sentence level) between a pivot and a candidate anchor could be small, leading to data sparseness issues. PMI is known to overestimate the association between rare words.[3] Second, PMI considers only the two words (i.e pivot and the anchor) and *not* the other words in their contexts.

We address the above-mentioned limitations of PMI by using contextualised word embeddings, $M(x, d)$ obtained from an MLM M representing a word x in a context d. We use sentences as the contexts of words and represent a word x by an embedding $\boldsymbol{x}$, computed as the average of $M(x, d)$ over $\mathcal{D}(x)$, given by (4).

$$\boldsymbol{x} = \frac{1}{|\mathcal{D}(x)|} \sum_{d \in \mathcal{D}(x)} M(x, d) \quad (4)$$

Using (4), for each word x we compute two embeddings $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ respectively in C_1 and C_2. If the word x is split into multiple subtokens, we use the average of those subtoken embeddings as $\boldsymbol{x}$. If x does not exist in a particular snapshot, it will be represented by a zero vector in that snapshot.

Specifically, given $w \in C_1 \cap C_2$, $u \in C_1$ and $v \in C_2$, we score a tuple (w, u, v) as in (5).

$$g(\boldsymbol{w}_1, \boldsymbol{u}_1) + g(\boldsymbol{w}_2, \boldsymbol{v}_2) - g(\boldsymbol{w}_2, \boldsymbol{u}_2) - g(\boldsymbol{w}_1, \boldsymbol{v}_1) \quad (5)$$

Here, $g(\boldsymbol{x}, \boldsymbol{y})$ is the cosine similarity between the embeddings $\boldsymbol{x}$ and $\boldsymbol{y}$. Note that (5) assigns higher scores to tuples (w, u, v) where w and u are highly

[3] For example, if $p(w, x) \approx p(w)$. Then, (2) reduces to $-\log p(x)$, which becomes larger for rare x (i.e. when $p(x) \to 0$).

related in C_1 and w and v in C_2, whereas it discourages the associations of w and u in C_2 and w and v in C_1. This enforces the diversity requirement discussed in § 3.2.2 and makes the tuple scoring method asymmetric between C_1 and C_2, which is desirable. Finally, we rank tuples by the scores computed using (5) and select the set, $\mathcal{S}_{\text{cont}}$, of top-$k$ ranked tuples to fill the templates to generate prompts.

This embedding-based tuple selection method overcomes the limitations of PMI discussed at the beginning of this section as follows. We can use contextualised embeddings from an MLM that is trained on a much larger corpus than two snapshots to obtain $M(x, d)$, thereby computing non-zero cosine similarities even when a pivot and an anchor *never* co-occurs in any sentence in a snapshot. Moreover, contextualised embeddings are known to encode semantic information that is useful to determine the word senses (Zhou and Bollegala, 2021) and semantic variations (Giulianelli et al., 2020), thus enabling us to better estimate the suitability of tuples.

3.3 Prompts from Automatic Templates

Given two snapshots C_1, C_2 of a timestamped corpus and a set $\mathcal{S}$ of tuples (w, u, v) extracted from any one of the three methods described in § 3.2, we propose a method to automatically learn a diverse set of templates. For this purpose, we can use any of the sets of tuples $\mathcal{S}_{\text{freq}}$, $\mathcal{S}_{\text{div}}$ or $\mathcal{S}_{\text{cont}}$ extracted as $\mathcal{S}$. We model template generation as an instance of text-to-text transformation. For example, given the context "**mask** *is associated with* **hide** *in* 2010 *and associated with* **vaccine** *in* 2020", containing the tuple (*mask*, *hide*,*vaccine*), we would like to generate the sequences shown in red italics as a template. Given a tuple (w, u, v), we extract two sentences $S_1 \in C_1$ and $S_2 \in C_2$ containing the two anchors respectively u and v, and use a pretrained T5 (Raffel et al., 2020) model to generate the slots Z_1, Z_2, Z_3, Z_4 for the conversion rule $\mathcal{T}_g(u, v, T_1, T_2)$ shown in (6).

$$S_1, S_2 \rightarrow S_1\ \langle Z_1\rangle\ u\ \langle Z_2\rangle\ T_1\ \langle Z_3\rangle\ v\ \langle Z_4\rangle\ T_2\ S_2 \quad (6)$$

The length of each slot to be generated is not required to be predefined, and we generate one token at a time until we encounter the next non-slot token (i.e. u, T_1, v, T_2).

The templates we generate must cover *all* tuples in $\mathcal{S}$. Therefore, when decoding we prefer templates that have high log-likelihood values according to (7).

$$\sum_{i=1}^{|\mathcal{T}|} \sum_{(w,u,v)\in\mathcal{S}} \log P_{T5}(t_i|t_1, \ldots, t_{i-1}; \mathcal{T}_g(u, v, T_1, T_2)) \quad (7)$$

where $t_1, \ldots, t_{|\mathcal{T}|}$ are the template tokens belonging to the slots Z_1, Z_2, Z_3 and Z_4.[4]

Following Gao et al. (2021), we use beam search with a wide beam width (e.g. 100) to obtain a large set of diverse templates. We then select the templates with the highest log-likelihood scores according to (7) as *auto* templates. By substituting the tuples in $\mathcal{S}$ in auto templates, we generate a set of auto prompts.

3.4 Examples of Prompts

Table 1 shows the manually-written templates and the automatically learnt templates We see that prompts describing diverse linguistic patterns expressing how a word's usage could have changed from one time stamp to another are learnt by the proposed method. Moreover, from Table 1, we see that automatically learnt templates tend to be shorter than the manually-written templates. Recall that the automatic template generation method prefers sequences with high likelihoods. On the other hand, longer sequences tend to be rare and have low likelihoods. Moreover, we require automatic templates to cover many tuples that are selected by a particular tuple selection method, thus producing more generalisable prompts. We believe the preference to generate shorter templates by the proposed method is due to those reasons.

3.5 Time-adaptation by Fine-Tuning

Given a set of prompts obtained by using the tuples in $\mathcal{S}_{\text{freq}}, \mathcal{S}_{\text{div}}$, or $\mathcal{S}_{\text{cont}}$ to fill the slots in either manually-written or automatically generated templates, we fine-tune a pretrained MLM M on those prompts such that M captures the semantic variation of a word w from T_1 to T_2. For this purpose, we add a language modelling head on top of M, randomly mask out one token at a time from each prompt, and require that M correctly predicts those masked out tokens from the remainder of the tokens in the context. We also experimented with a variant where we masked out only the anchor words from a prompt, but did not observe a notable difference in performance over random masking of all tokens.

[4] Each slot can contain zero or more template tokens.

Template	Type
$\langle w \rangle$ *is associated with* $\langle u \rangle$ *in* $\langle T_1 \rangle$*, whereas it is associated with* $\langle v \rangle$ *in* $\langle T_2 \rangle$.	Manual
Unlike in $\langle T1 \rangle$*, where* $\langle u \rangle$ *was associated with* $\langle w \rangle$*, in* $\langle T2 \rangle$ $\langle v \rangle$ *is associated with* $\langle w \rangle$.	Manual
The meaning of $\langle w \rangle$ *changed from* $\langle T_1 \rangle$ *to* $\langle T_2 \rangle$ *respectively from* $\langle u \rangle$ *to* $\langle v \rangle$.	Manual
$\langle u \rangle$ *in* $\langle T_1 \rangle$ $\langle v \rangle$ *in* $\langle T_2 \rangle$	Automatic
$\langle u \rangle$ *in* $\langle T_1 \rangle$ *and* $\langle v \rangle$ *in* $\langle T_2 \rangle$	Automatic
The $\langle u \rangle$ *in* $\langle T_1 \rangle$ *and* $\langle v \rangle$ *in* $\langle T_2 \rangle$	Automatic

Table 1: Experimented templates. "Manual" denotes that the template is manually-written, whereas "Automatic" denotes that the template is automatically-generated.

4 Experiments and Results

Datasets: We use the following four datasets that were collected and used by Hofmann et al. (2021) for evaluating DCWEs: **Yelp**, **Reddit**, **ArXiv**, and **Ciao** (Tang et al., 2012). Details of these datasets and all pre-processing steps are detailed in Appendix C. We remove duplicates as well as texts with less than 10 words in each dataset. We then randomly split each snapshot of a dataset into training, development, and test sets, containing respectively 70%, 10% and 20% of the original dataset.

Evaluation Metric: If an MLM is correctly adapted to a timestamp T_2, it should be able to assign higher probabilities to the masked out tokens in unseen texts in C_2, sampled at T_2. We follow prior work on DCWE (Hofmann et al., 2021) and use the masked language modelling perplexity as our evaluation metric on test texts in C_2. If an MLM is well-adapted to T_2, it will have a lower perplexity for test texts in C_2.

Baselines: To put our results into perspective, we consider the following baselines:

Original BERT: We use pretrained `BERT-base-uncased`[5] as the MLM without any fine-tuning to be consistent with (Hofmann et al., 2021). Further evaluations on RoBERTa (Liu et al., 2019) are given in Appendix B.

BERT(T_1): We fine-tune the Original BERT model on the training data sampled at T_1.

BERT(T_2): We fine-tune the Original BERT model on the training data sampled at T_2. Note that this is the same training data that was used for selecting tuples in §3.2

FT: The BERT models fine-tuned by the proposed method. We use the notation FT(`model`, `template`) to denote the model obtained by fine-tuning a given MLM using a `template`, which is either manually-written (*manual*) or automatically-generated (*auto*) as described in §3.3.

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

MLM	Yelp	Reddit	ArXiv	Ciao
Original BERT	15.125	25.277	11.142	12.669
FT(BERT, Manual)	14.562	24.109	**10.849**	**12.371**
FT(BERT, Auto)	**14.458**	**23.382**	10.903	12.394
BERT(T_1)	5.543	**9.287**	5.854	7.423
FT(BERT(T_1), Manual)	**5.534**	9.327	**5.817**	**7.334**
FT(BERT(T_1), Auto)	5.541	9.303	5.818	7.347
BERT(T_2)	4.718	8.927	3.500	5.840
FT(BERT(T_2), Manual)	4.714	**8.906**†	3.499	**5.813**†
FT(BERT(T_2), Auto)	**4.708**†	8.917	**3.499**†	5.827

Table 2: Masked language modelling perplexities (lower the better) on test sentences in C_2 in YELP, Reddit, ArXiv, and Ciao datasets are shown for different MLMs. Best results in each block (methods using the same baseline MLM) are shown in bold, while overall best results are indicated by †.

Hyperparameters: We use the held-out development data to tune all hyperparameters. We follow the recommendations of Mosbach et al. (2021) for fine-tuning BERT on small datasets and use weight decay (0.01) with bias correction in Adam optimiser (Kingma and Ba, 2015). We use a batch size of 4, learning rate of 3×10^{-8}, the number of tuples used for prompt-based fine-tuning (k) is selected from $\in \{500, 1000, 2000, 5000, 10000\}$, and the number of epochs is set to 20. (further details on hyperparameters are given in Appendix D).

We used a single NVIDIA RTX A6000 and 64 GB RAM in our experiments. It takes approximately 6 hours to fine-tune, validate and test all methods reported in the paper for the four datasets. The run time varies depending on the number of tuples used in the proposed method. Tuple selection takes, on average, 0.5 hours.

4.1 Results

In Table 2, we compare the effect of fine-tuning BERT MLMs using the prompts generated by filling the selected tuples in either the manually-written (manual) or automatically learnt (auto) templates. We use the optimal tuples selection method

MLM	Yelp	Reddit	ArXiv	Ciao
FT(BERT(T_2), Manual)	4.714	**8.906**†	3.499	**5.813**†
FT(BERT(T_2), Auto)	**4.708**†	8.917	**3.499**†	5.827
TempoBERT	5.516	12.561	3.709	6.126
CWE	4.723	**9.555**	3.530	5.910
DCWE [temp. only]	4.723	9.631	3.515	**5.899**
DCWE [temp.+social]	**4.720**	9.596	**3.513**	5.902

Table 3: MLM perplexities (lower the better) are shown for the proposed method, previously proposed TempoBert (Rosin et al., 2022), and DCWE variants (Hofmann et al., 2021). Best results in each block (methods using the same baseline MLM) are shown in bold, while overall best results are indicated by †.

and the number of tuples for prompt-based fine-tuning, decided using the validation data in each datasets.

From Table 2 we see that the **Original BERT** model has the highest perplexity scores in all four datasets. This shows that the **Original BERT** model is not accurately capturing the meanings of words as used in C_2. Although fine-tuning **Original BERT** using manual or auto prompts improves the perplexity scores, they still remain very high. **BERT**(T_1), obtained by fine-tuning the **Original BERT** model on C_1, immediately reduces the perplexity scores on both datasets. Recall that C_1 is sampled from the same domain but at T_1, which is different from T_2. This indicates the importance of using in-domain data for fine-tuning MLMs even-though it might be from a different timestamp.

On the other hand, **BERT**(T_2), obtained by fine-tuning **Original BERT** on the training data from T_2, further reduces perplexity over **BERT**(T_2). Importantly, fine-tuning using both manual and auto prompts further reduces perplexity over **BERT**(T_2). Overall, the best performances on Yelp and ArXiv are reported by fine-tuning **BERT**(T_2) using auto prompts (i.e. **FT(BERT**(T_2)**, Auto)**), whereas the same on Reddit and Ciao are reported by manual prompts (i.e. **FT(BERT**(T_2)**, Manual)**). Applying auto or manual prompts for fine-tuning **BERT**(T_1) improves perplexity in Yelp, ArXiv, and Ciao, but not on Reddit. This shows the importance of first fine-tuning BERT on C_2 (in-domain and contemporary) before using prompts to further fine-tune models, because prompts are designed or learnt for the purpose for adapting to T_2 and not to T_1, reflecting the direction of the time adaptation ($T_1 \rightarrow T_2$).

Although there is prior work on non-contextualised dynamic embeddings, we cannot perform language modelling with those as the probability of predicting a word will be a constant independent of its context. Moreover, none of those work evaluate on the benchmarks proposed by Hofmann et al. (2021), which we also use. Therefore, we consider the current SoTA for DCWEs proposed by Hofmann et al. (2021) and the SoTA for time-adaptation, TempoBERT (Rosin et al., 2022), as our main comparison points in Table 3.

The DCWE method proposed by Hofmann et al. (2021) uses BERT fine-tuned on training data from T_2 as the baseline MLM (i.e. **CWE**). Moreover, their method is available in two flavours: a version that uses both social and temporal information (denoted as **DCWE [social + temp.]**) and a version where social information is ablated (denoted as **DCWE [temp.]**). Given that we do not use social information in our proposed method, the direct comparison point for us would be **DCWE [temp.]**. We see that our proposed method with both manual and auto prompts consistently outperforms both flavours of the SoTA DCWEs proposed by Hofmann et al. (2021) in all datasets.

TempoBert inserts a special token indicating the time period at the beginning of each sentence in a training corpus, and fine-tunes BERT on the corpora available for each time period. We trained TempoBert on our datasets for the same number of epochs as and with an initial learning rate of 3e-6 and measured perplexity on the same test splits. As seen from Table 3, the proposed method using both manual and automatic templates outperforms TempoBert in all datasets.

The number of tuples selected (i.e. k) to generate prompts with manually-written or automatically generated templates determines the number of prompts used to fine-tune a given MLM. To study the effect of k, we use a particular tuple selection method and select the top-ranked k tuples according to that method with either a manually-written (**Manual**) or automatically learnt (**Auto**) template to generate prompts. This results in six different types of prompts. Next, we fine-tune a given MLM using the generated prompts and repeat this process for increasing k values. Figure 1 shows the results of fine-tuning **BERT**(T_2) to T_2. (Results for **BERT**(T_1) are shown in Appendix A)

Overall, from Figure 1 we see that when the number of tuples used for fine-tuning increases, almost all methods reach some minimum perplexity score. However, we see that for each method, its

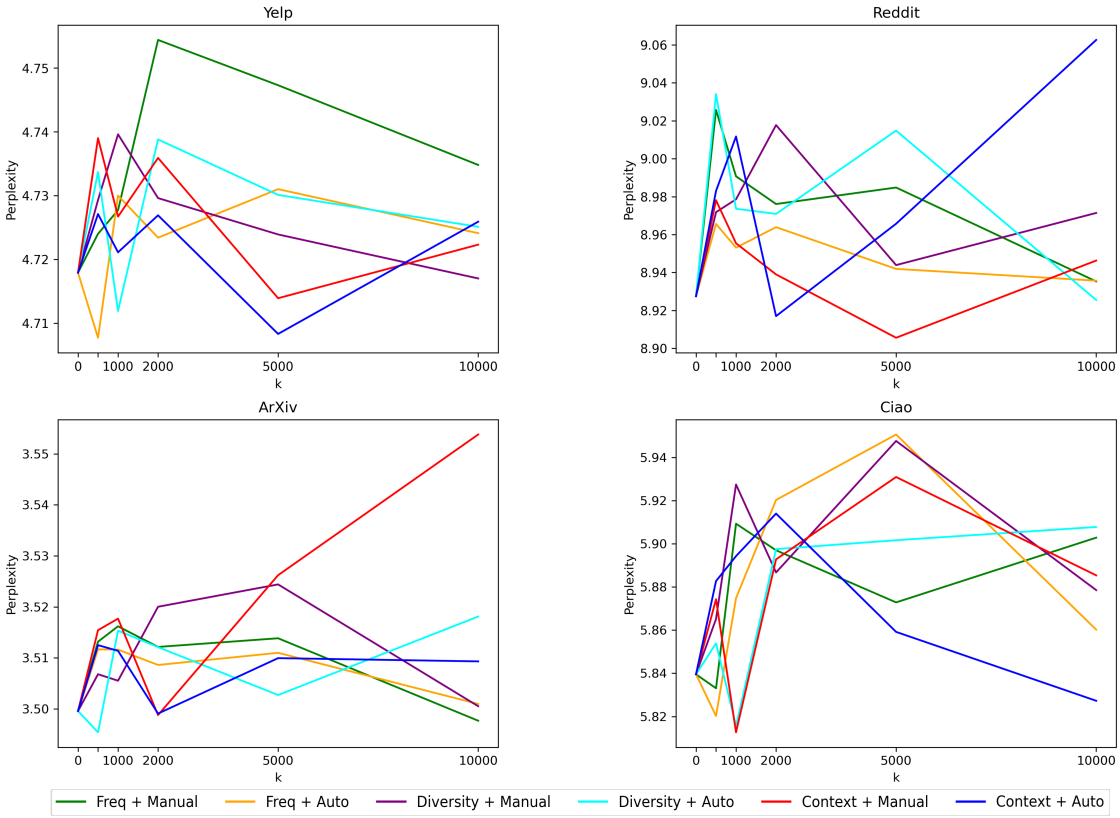

Figure 1: Adapting BERT(T_2) to T_2 on Yelp (top left), Reddit (top right), ArXiv (bottom left), and Ciao (bottom right) datasets using different tuple selection methods (**Freq**uency, **Diversity**, **Context**) and templates (**Auto**, **Manual**). Perplexity scores are shown against the the number of tuples (k) used in prompt-based fine-tuning.

minimum perplexity scores on different datasets is obtained by using different k tuples. Recall that each tuple selection method ranks tuples by some goodness score. Therefore, when we increase k we are using less reliable noisy tuples to generate prompts, thus leading to reduced performance. Interestingly, we see that the best performances can be obtained with relatively a smaller number of tuples (< 5000) in all datasets.

A closer look reveals that on Yelp, all **Freq+Auto** and **Context+Auto** obtain similar best performances. However, **Freq+Auto** reaches its optimal point with 500 tuples, whereas **Context+Auto** requires 5000 tuples. Among the three tuple selection methods Context-based tuple selection is the best. Frequency-based tuple selection method works well with auto templates but not so much with manual ones. This shows that auto templates can be used to extract good performance from a simple tuple selection method such as Frequency-based tuple selection.

On Reddit, the overall best performance is obtained by **Context+Manual** with 5000 tuples, and its performance drops with higher k values, due to the increasing noise in tuples as explained above. Likewise in Yelp, context-based tuple selection emerges as the overall best method in Reddit as well with 5000 tuples. However, context-based tuple selection has to be used with manual templates to obtain good performance on Reddit, whereas in Yelp using it with the auto templates was the best.

On ArXiv, both **Freq+Manual** and **Diversity+Auto** reach similar best performances. While **Freq+Manual** requires 500 tuples, it takes **Diversity+Auto** 10000 tuples to reach its best performance. Unlike Yelp and Reddit, the best tuple selection in ArXiv is Diversity-based tuple selection. The Frequency-based tuple selection also similar performance, but requires more tuples. For Context-based tuple selection, although it improves the perplexity scores over the baseline MLM, the improvements are smaller than other methods.

On Ciao, **Context+Manual** and **Diversity + Auto** obtain similar best performances, both with 1000 tuples. Similarly as Yelp and Reddit, the overall best tuple selection is Context-based tuple selection, which obtains the best perplexity scores. Diversity-based tuple selection also has good per-

Pivot (w)	Anchors (u, v)
place	(burgerville, takeaway), (burgerville, dominos), (joes, dominos)
service	(doorman, staffs), (clerks, personnel), (clerks, administration)
phone	(nokia, iphone), (nokia, ipod), (nokia, blackberry)

Table 4: Top-ranked pivots w and their associated anchors u and v selected according to the contextualised tuple selection method from Yelp (Row 1 and 2) and Ciao (Row 3).

formance, although it only occurs when it is used with auto templates.

Table 4 shows some examples of the top scoring pivots and their anchors retrieved by the context-based tuple selection method from Yelp and Ciao. From Yelp, in year 2010 (T_1), we see that dine-in restaurants such as *burgerville*[6] and *joes*[7] are associated with *place*, whereas in 2020 *takeaway* and *dominos*[8] are associated with *place* probably due to the COVID-19 imposed lockdowns restricting eating out. Moreover, we observe a shift in office-related job titles between these time periods where *service* is closely associated with *doorman* (T1: 108, T2: 48) and *clerks* (T1: 115, T2: 105), which are rarely used in 2020 and are replaced with more gender-neutral titles such as *staff* (T1: 28618, T2: 60421), *personnel* (T1: 85, T2: 319) and *administration* (T1: 37, T2: 109). From Ciao, in year 2001 (T_1), we see that phone brands like *nokia*[9] are closely connected with *phone*, while in year 2011 (T_2), as companies such as apple[10] and *blackberry*[11] took a large part of the mobile phone market, *iphone*, *ipod*, and *blackberry* become more related with *phone*.

5 Conclusion

We propose an unsupervised method to learn DCWEs by time-adapting a pretrained MLM using prompts from manual and automatic templates. Experimental results on multiple datasets demonstrate that our proposed method can obtain better perplexity scores on unseen target texts compared to prior work. In the future, we plan to extend the proposed method to adapt multilingual word embeddings.

[6] www.burgerville.com
[7] www.joespizzanyc.com
[8] www.dominos.com
[9] www.nokia.com
[10] www.apple.com
[11] www.blackberry.com

6 Limitations

This paper takes into account the temporal semantic variations of words and proposes a method to learn dynamic contextualised word embeddings by time-adapting an MLM using prompt-based fine-tuning methods. In this section, we highlight some of the important limitations of this work. We hope this will be useful when extending our work in the future by addressing these limitations.

The learned dynamic contextualised word embeddings are limited to the English language, which is a morphologically limited language. Therefore, the findings reported in this work might not generalise to other languages. However, there are already numerous multilingual MLMs such as mBERT (Devlin et al., 2019), XLM (CONNEAU and Lample, 2019) and XLM-R (Conneau et al., 2020), to name a few. Extending our work to multilingual dynamic contextualised word embeddings will be a natural line of future work.

Dynamic contextualised word embeddings represent words as a function of extralinguistic context (Hofmann et al., 2021), which consider both time and social aspects of words. However, in this paper we focused solely on the temporal aspect and ignored the social aspect. Extending our work to take into account the social semantic variations of a word is an important future direction.

Due to the costs involved when fine-tuning large-scale MLMs, we keep the number of manually-written and automatically learnt templates to a manageable small number as shown in Table 1 in §3.4. However, it remains to be evaluated the impact of increasing the number of templates on the performance of the proposed method.

7 Ethical Considerations

In this paper, we considered the problem of capturing temporal semantic variation of words by learning dynamic contextualised word embeddings. For this purpose, we proposed a method to adapt a masked language model from to a given time stamp. We did not collect, annotate or release new datasets during this process. However, we used pretrained MLMs and four datasets from the internet (Yelp, Reddit, Arxiv, and Ciao). It is known that pretrained MLMs contain unfair social biases (May et al., 2019; Nadeem et al., 2021; Kaneko and Bollegala, 2021; Kaneko et al., 2022). Such biases can be amplified by fine-tuning methods, especially when the fine-tuning prompts are extracted from

social data such as customer reviews in Yelp or discussions in Reddit. Therefore, we consider it is important to further evaluate (Nangia et al., 2020; Nadeem et al., 2021; Kaneko and Bollegala, 2022) the adapted MLMs for social biases, and if they do exist, then apply appropriate debiasing methods (Kaneko and Bollegala, 2021; Lauscher et al., 2021) before the MLMs are deployed in downstream NLP applications that are used by billions of users world-wide.

Acknowledgements

Danushka Bollegala holds concurrent appointments as a Professor at University of Liverpool and as an Amazon Scholar. This paper describes work performed at the University of Liverpool and is not associated with Amazon.

References

Oshin Agarwal and Ani Nenkova. 2022. Temporal effects on pre-trained models for language processing tasks. *Transactions of the Association for Computational Linguistics*, 10:904–921.

Taichi Aida and Danushka Bollegala. 2023. Unsupervised semantic variation prediction using the distribution of sibling embeddings. In *Proc. of the Findings of 61st Annual Meeting of the 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 *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 2514–2524, New York, NY, USA. Association for Computing Machinery.

Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. 2014. Analysing word meaning over time by exploiting temporal random indexing.

Joan Baybee. 2015. *Language Change*. Cambridge University Press.

Eyal Ben-David, Nadav Oved, and Roi Reichart. 2021. PADA: Example-based Prompt Learning for on-the-fly Adaptation to Unseen Domains. *Transactions of Association for Computational Linguistics*.

Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. Unsupervised cross-domain word representation learning. In *Proc. of ACL*, pages 730 – 740.

Lyle Campbell. 2004. *Historic Linguistics*. Edinburgh University Press.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *ACL*.

Alexis CONNEAU and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

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.

Leandra Fichtel, Jan-Christoph Kalo, and Wolf-Tilo Balke. 2021. Prompt tuning or fine-tuning - investigating relational knowledge in pre-trained language models. In *3rd Conference on Automated Knowledge Base Construction*.

John R. Firth. 1957. A synopsis of linguistic theory 1930-55. *Studies in Linguistic Analysis*, pages 1 – 32.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. 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 3816–3830, Online. 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.

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.

Joel Jang, Seonghyeon Ye, Changho Lee, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, and Minjoon Seo. 2022. Temporalwiki: A lifelong benchmark for training and evaluating ever-evolving language models. *arXiv preprint arXiv:2204.14211*.

Xisen Jin, Dejiao Zhang, Henghui Zhu, Wei Xiao, Shang-Wen Li, Xiaokai Wei, Andrew Arnold, and Xiang Ren. 2022. Lifelong pretraining: Continually adapting language models to emerging corpora. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 1–16, virtual+Dublin. Association for Computational Linguistics.

Masahiro Kaneko and Danushka Bollegala. 2021. Debiasing pre-trained contextualised embeddings. In *Proc. of 16th conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Masahiro Kaneko and Danushka Bollegala. 2022. Unmasking the mask – evaluating social biases in masked language models. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, page (to appear), Vancouver, BC, Canada.

Masahiro Kaneko, Aizhan Imankulova, Danushka Bollegala, and Naoaki Okazaki. 2022. Gender bias in masked language models for multiple languages. In *Proc. of NAACL-HLT*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Peter Koch. 2016. *Meaning change and semantic shifts.*, pages 21–66. De Gruyter.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *Proc. of ICLR*.

Anne Lauscher, Tobias Lueken, and Goran Glavaš. 2021. Sustainable modular debiasing of language models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4782–4797, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Teven Le Scao and Alexander Rush. 2021. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636, Online. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing.

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.

Daniel Loureiro, Francesco Barbieri, Leonardo Neves, Luis Espinosa Anke, and Jose Camacho-Collados. 2022. TimeLMs: Diachronic Language Models from Twitter.

Chandler May, Alex Wang, Shikha Bordia, Samuel R. Bowman, and Rachel Rudinger. 2019. On measuring social biases in sentence encoders. 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 622–628, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning BERT: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*.

Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. 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 5356–5371, Online. Association for Computational Linguistics.

Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R. Bowman. 2020. CrowS-pairs: A challenge dataset for measuring social biases in masked language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1953–1967, Online. Association for Computational Linguistics.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

Wenjun Qiu and Yang Xu. 2022. HistBERT: A Pretrained Language Model for Diachronic Lexical Semantic Analysis.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Taraka Rama and Lars Borin. 2015. Comparative evaluation of string similarity measures for automatic language classification. In *Sequences in Language and Text*, pages 171–200. DE GRUYTER.

Justyna A. Robinson. 2012. A gay paper: why should sociolinguistics bother with semantics?: Can sociolinguistic methods shed light on semantic variation and change in reference to the adjective gay? *English Today*, 28(4):38–54.

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*, WSDM '22, page 833–841, New York, NY, USA. Association for Computing Machinery.

Guy D. Rosin and Kira Radinsky. 2022. Temporal attention for language models. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1498–1508, Seattle, United States. Association for Computational Linguistics.

Maja Rudolph and David Blei. 2018. Dynamic embeddings for language evolution. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 1003–1011, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Eyal Sagi, Stefan Kaufmann, and Brady Clark. 2011. Tracing semantic change with latent semantic analysis. In *Current Methods in Historical Semantics*, pages 161–183. DE GRUYTER.

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.

Jiliang Tang, Huiji Gao, and Huan Liu. 2012. mtrust: Discerning multi-faceted trust in a connected world. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 93–102.

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

Asahi Ushio, Jose Camacho-Collados, and Steven Schockaert. 2021. Distilling relation embeddings from pretrained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9044–9062, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Prasetya Utama, Nafise Sadat Moosavi, Victor Sanh, and Iryna Gurevych. 2021. Avoiding inference heuristics in few-shot prompt-based finetuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9063–9074, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Charles Welch, Jonathan K. Kummerfeld, Verónica Pérez-Rosas, and Rada Mihalcea. 2020. Compositional demographic word embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4076–4089, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Ziqian Zeng, Yichun Yin, Yangqiu Song, and Ming Zhang. 2017. Socialized word embeddings. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, California. International Joint Conferences on Artificial Intelligence Organization.

Yi Zhou and Danushka Bollegala. 2021. Learning sense-specific static embeddings using contextualised word embeddings as a proxy. In *Proceedings of the 35th Pacific Asia Conference on Language, Information and Computation*, pages 493–502, Shanghai, China. Association for Computational Linguistics.

Appendix

A Fine-tuning results on C_1

Figure 2 shows the effect of the number of tuples (i.e. k) selected using different tuple selection methods, and the perplexity scores for the **BERT**(T_1) models, fine-tuned using the prompts generated by filling the slots with those tuples in either manually-written (**Manual**) or automatically-generated (**Auto**) templates. Note that we have three methods to select tuples (i.e. **Freq**uency-based tuple selection, **Diversity**-based tuple selection, and **Context**-based tuple selection). Combined with the two methods for obtaining tuples, we have six comparisons in Figure 2 on Yelp, Reddit, ArXiv, and Ciao datasets. Because a only a single template is used in each setting, the number of tuples (k) is equal to the number of prompts used to fine-tune an MLM in this experiment.

On Yelp, we see that **Freq+Auto** and **Diversity+Auto** both obtain the lowest (best) perplexity scores. In particular, we see that **Freq+Auto** reaches this optimal performance point with as less as 500 prompts, whereas **Diversity+Auto** requires 1000 prompts. However, when we increase the number of prompts beyond the optimal performance points for each method, we see that the perplexity increases due to the added noise when

using low-scoring tuples for generating prompts. Although for both of those methods the perplexity scores drop again when a large number of prompts are being used (i.e. more than 5000 prompts) only **Diversity+Auto** recovers to the best performance level it obtained with 1000 prompts. Therefore, we note that there is a trade-off here between the quality vs. quantity of using noisy prompts for fine-tuning. However, from a computational point of view it is desirable to use a small number of prompts if that suffice to obtain good performance. Therefore, we recommend using **Freq+Auto** in this case because it obtained good performance with only 500 prompts.

On Reddit we see that the perplexity increases with the number of prompts in almost all methods from the start. However, they reach a peak and then start decreasing again. However, among all methods we see that only Diversity+Auto recovers to its initial levels. In fact, with 10000 prompts it is able to report perplexity scores lower than that of its initial values, thus reporting the best performance on Reddit by any fine-tuning method. However, recall that auto templates were specifically learnt to obtain good performance when adapting from T_1 to T_2, and the perplexity scores obtained by fine-tuning **BERT**(T_2) are much better than those obtained by fine-tuning **BERT**(T_1) (which are shown in Figure 2) as explained in the main body of the paper.

On ArXiv we see that **Freq+Auto** obtain the best perplexity score. In almost all methods, the perplexity scores drop first and then increase. However, the increases are followed by drops and then increases. The trend of perplexity scores regarding the tuple numbers seems a wave. Unlike other mehtods, **Context+Auto** almost continues to improve its performances as the number of tuples increases. **Freq+Auto** is the overall best method as it reaches the best perplexity score with 2000 tuples. In addition, we see that the potential performances of **Context+Auto** would be high since its performances increase with the number of tuples.

On Ciao we see that **Diversity+Auto** obtains the best perplexity score and it is much better than other methods. Unlike other datasets, all methods reach their best perplexity scores with small numbers of tuples (< 1000). The trend of perplexity score changing regarding the numbers of tuples is almost the same in all methods: drop, increase, and drop.

B Experiment on RoBERTa

To explore the proposed method's potential on other MLMs than BERT, we conduct a small-scale experiment on RoBERTa. The baselines and evaluation metric setting are similar to the experiment in the main body except that the MLM is changed to `RoBERTa-base`[12] and we only use the Reddit datasets.

In Table 5 we compare the effect of fine-tuning RoBERTa MLMs using the prompts from both automatic and manual templates. From Table 5 we see that the **Original RoBERTa** has the highest perplexity score in Reddit dataset, and fine-tuning **Original RoBERTa** with manual or auto prompts improves the perplexity. While applying manual prompts does not improve the perplexity score over **RoBERTa**(T_1), fine-tuning with auto prompts makes some improvements. Likewise the results of the main experiment on BERT, fine-tuning using both manual and auto prompts further reduces perplexity over **RoBERTa**(T_2).

Figure 3 shows the results of fine-tuning **RoBERTa**(T_1) and **RoBERTa**(T_2) **to** T_2.

For **RoBERTa**(T_1), **Context+Auto** has the best perplexity score with 1000 tuples. However, the context-based tuple selection method only improve the perplexity score over the baseline when it is used with auto templates. Moreover, **Context+Auto** is the only method that improves the perplexity against the baseline MLM.

For **RoBERTa**(T_2), similar as **RoBERTa**(T_1), **Context+Auto** obtain the lowest (best) perplexity score with 1000 tuples. **Freq+Auto** also reaches a similar perplexity score with 2000 tuples. As tuple numbers increase, almost all methods first reach optimal points, and then their perplexity scores increase as the tuple numbers increase. **Context+Auto** is the overall best method because its best performance and the smallest tuple number.

C Datasets

Yelp: Yelp is a platform which provides crowdsourced reviews on businesses. We select publicly available reviews[13] covering the years 2010 (=T_1) and 2020 (=T_2).

Reddit: Reddit is a social media platform covering a wide range of topics arranged into communities called *subreddits*. Following Hofmann et al.

[12] https://huggingface.co/roberta-base
[13] https://www.yelp.com/dataset

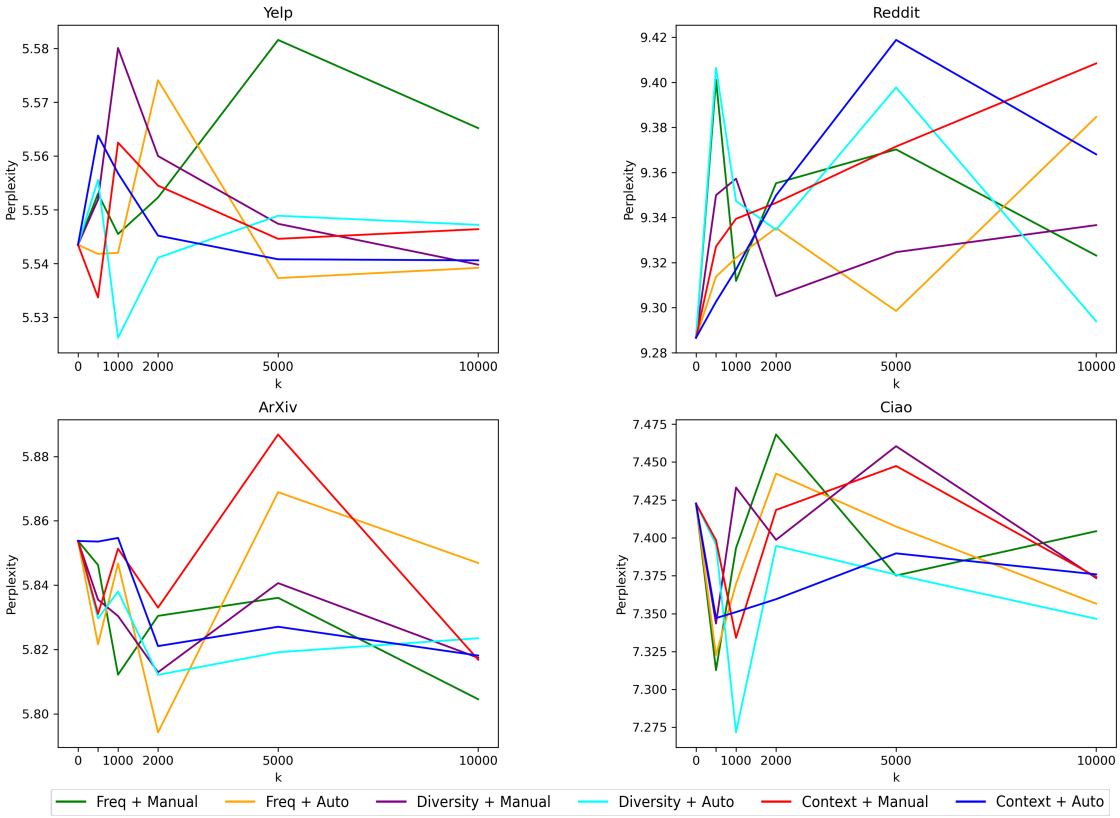

Figure 2: Adapting BERT(T_1) to T_2 on YELP (top left), Reddit (top right), ArXiv (bottom left) and Ciao (bottom right) datasets using different tuple selection methods (**Freq**uency, **Diversity**, **Context**) and templates (**Auto**, **Manual**). Perplexity scores are shown against the the number of tuples (k) used in prompt-based fine-tuning.

(2021), from the publicly released Reddit posts,[14] we take all comments from September 2019 (=T_1) and April 2020 (=T_2), which reflect the effects of the COVID-19 pandemic. We remove subreddits with fewer than 60 comments and randomly sample 60 comments per subreddit.

ArXiv: ArXiv is an open-access repository of scientific articles. We obtain abstracts of papers published at years 2001 (=T_1) and 2020 (=T_2) on ArXiv from a publicly available dataset[15]. Following Hofmann et al. (2021), we drop those data under ArXiv's subjects (e.g., CS.CL) that has less than 100 publications between 2001 and 2020.

Ciao: Ciao is a product review site. We select reviews from years 2000 (=T_1) and 2011 (=T_2) from a publicly released dataset (Tang et al., 2012)[16].

D Hyperparameters

Table 6 shows the hyperparameter values for fine-tuning **BERT**(T_2) and **RoBERTa**(T_2) using prompts on T_2. We used `T5-base`[17] to generate automatic prompts. The batch size of generating process is 32, and the width of the beam search is set to 100.

To examine the influence of the random seeds, we firstly perform **FT(BERT**(T_2)**,Auto)** on ArXiv with different numbers of tuples and three random seeds. Then we calculated the mean values and the standard deviations of perplexities with different tuple numbers regarding the random seeds. As we average the mean values and the standard deviation, we see that the average standard deviation (i.e. 0.0066) is much smaller than the average mean (i.e. 3.5079), which is nearly 1/1000. Thus, we only use 123 as the random seed for the formal experiments.

[14] https://files.pushshift.io/reddit/comments/
[15] https://www.kaggle.com/datasets/Cornell-University/arxiv
[16] https://www.cse.msu.edu/~tangjili/trust.html
[17] https://huggingface.co/t5-base

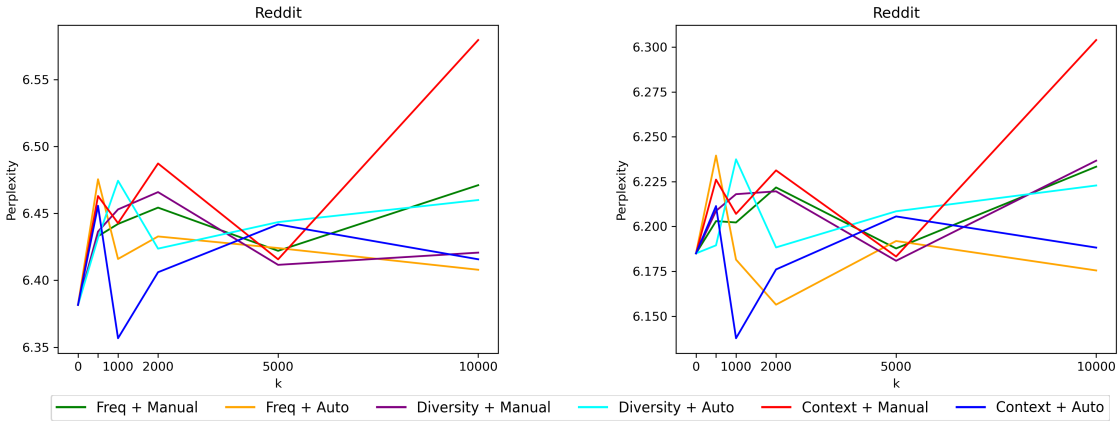

Figure 3: Adapting RoBERTa(T_1) to T_2 (left) and RoBERTa(T_2) to T_2 (right) on Reddit dataset using different tuple selection methods (**Freq**uency, **Diversity**, **Context**) and templates (**Auto**, **Manual**). Perplexity scores are shown against the the number of tuples (k) used in prompt-based fine-tuning.

MLM Reddit	
Original RoBERTa	13.997
FT(RoBERTa, Manual)	13.818
FT(RoBERTa, Auto)	**13.323**
RoBERTa(T_1)	6.382
FT(RoBERTa(T_1), Manual)	6.443
FT(RoBERTa(T_1), Auto)	**6.357**
RoBERTa(T_2)	6.185
FT(RoBERTa(T_2), Manual)	6.183
FT(RoBERTa(T_2), Auto)	**6.138**†

Table 5: Masked language modelling perplexities (lower the better) on test sentences in C_2 in Reddit datasets are shown for different MLMs. Best results in each block (methods using the same baseline MLM) are shown in bold, while overall best results are indicated by †.

MLM	Yelp		Reddit		ArXiv		Ciao	
	l	s	l	s	l	s	l	s
FT(BERT(T_2), Manual)	3e-8	–	1e-8	–	5e-7	warm up*	6e-8	warm up
FT(BERT(T_2), Auto)	3e-8	–	2e-7	warm up*	3e-8	–	6e-7	warm up
FT(RoBERTa(T_2), Manual)	–	–	3e-8	–	–	–	–	–
FT(RoBERTa(T_2), Auto)	–	–	3e-8	–	–	–	–	–

Table 6: Hyperparameters setting for adapting BERT(T_2) and RoBERTa(T_2) to T_2 on Yelp, Reddit, ArXiv, and Ciao datasets. Here, "warm up" denotes that the learning rate is linearly increased for the first $p\%$ of the steps, where $p = 10$ for {Reddit,Arxiv} and $p = 65$ for {Ciao} if applicable. * indicates that the learning rate is linearly decayed until zero after the "warm up".

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
section 6

☑ A2. Did you discuss any potential risks of your work?
section 7

☑ A3. Do the abstract and introduction summarize the paper's main claims?
abstract and section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

sections 3 and 4

☑ B1. Did you cite the creators of artifacts you used?
sections 3 and 4

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
sections 3 and 4

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
section 4

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
sections 3, 4 and Appendix

C ☑ Did you run computational experiments?

sections 3, 4, and Appendix

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
section 4 and Appendix

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
section 3, section 4 and Appendix

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
section 4 and Appendix

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
section 4 and Appendix

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.