TransDrift: Modeling Word-Embedding Drift using Transformer

Nishtha Madaan
Indian Institute of Technology Delhi
New Delhi, India
nishthaa.madaan@gmail.com

Prateek Chaudhury
Indian Institute of Technology Delhi
New Delhi, India
prateekchaudhury@gmail.com

Nishant Kumar
Indian Institute of Technology Delhi
New Delhi, India
cs5190586@iitd.ac.in

Srikanta Bedathur
Indian Institute of Technology Delhi
New Delhi, India
srikanta@cse.iitd.ac.in

ABSTRACT

In modern NLP applications, word embeddings are a crucial backbone that can be readily shared across a number of tasks. However, as the text distributions change and word semantics evolve over time, the downstream applications using the embeddings can suffer if the word representations do not conform to the data drift. Thus, maintaining word embeddings to be consistent with the underlying data distribution is a key problem. In this work, we tackle this problem and propose TransDrift[1], a transformer-based prediction model for word embeddings. Leveraging the flexibility of transformer, our model accurately learns the dynamics of the embedding drift and predicts the future embedding. In experiments, we compare with existing methods and show that our model makes significantly more accurate predictions of the word embedding than the baselines. Crucially, by applying the predicted embeddings as a backbone for downstream classification tasks, we show that our embeddings lead to superior performance compared to the previous methods.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**.

KEYWORDS

transformers, word2vec, drift

ACM Reference Format:
Nishtha Madaan, Prateek Chaudhury, Nishant Kumar, and Srikanta Bedathur. 2024. TransDrift: Modeling Word-Embedding Drift using Transformer. In *Companion Proceedings of the ACM Web Conference 2024 (WWW '24 Companion), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3589335.3651894

1 INTRODUCTION

Word embeddings are pivotal in modern NLP tasks, serving as a reusable feature store widely adopted across various industry applications [10, 13, 15, 32, 33]. However, the dynamic nature of data distributions over time poses a challenge. Words evolve in semantics and usage over time. For example, *vacation* may connote *beach-related activities* in summer and *skiing* in winter. This variability poses challenges; for instance, a customer preference model relying on embedding similarity may falter if *vacation* is closer to *beach* than *skiing* in winter. Thus, robust embeddings must align with evolving data distributions to enhance downstream NLP applications.

Temporal drift presents a significant concern, often resulting in inadequate data from the new distribution. For example, transitioning from summer to winter may yield scant winter data for model training. A simple solution of retraining embeddings with new winter data may not be feasible due to data scarcity. Nevertheless, the quest remains for updated embeddings aligned with evolving data distributions.

Prior studies have explored historical data to identify temporal drifts in word embeddings [11, 14, 16, 17, 22, 25], revealing their detrimental impact on downstream tasks. Furthermore, instability in embeddings due to minor data shifts has been underscored [2, 8, 19, 27]. However, these methods lack a consistent approach to update embeddings in response to drift dynamics.

We propose TransDrift, a novel model based on transformer architecture that predicts future embeddings aligned with data drift by integrating past embeddings with drift dynamics. Our experiments validate the utility of predicted embeddings for downstream NLP tasks. The model's simplicity and versatility make it compatible with any word embedding algorithm.

Our main contributions can be summarized as: *1)* We propose a novel model, TransDrift, that leverages transformer to predict the future embeddings. *2)* Our model can predict future embeddings by leveraging some amount of future data, if available. *3)* Our results show that our model is effective in modeling the drift in the embeddings. *4)* Lastly, we also show improvement in the accuracy on downstream NLP tasks when using our predicted embeddings.

2 BACKGROUND

2.1 Word Embeddings

Methods like *word2vec* commonly generate application-agnostic word embeddings [4, 29, 30]. These methods take a text corpus $\mathcal{D}$ and generate embeddings $\mathbf{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_N\}$ for N words, where each $\mathbf{e}_n$ is a d-dimensional vector representing the n-th word. The aim is to embed words in a feature space while capturing their semantic structure, clustering similar words like *apple* and *orange*

[1]Codebase: https://github.com/data-iitd/transdrift

WWW '24 Companion, May 13–17, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-0172-6/24/05…$15.00
https://doi.org/10.1145/3589335.3651894

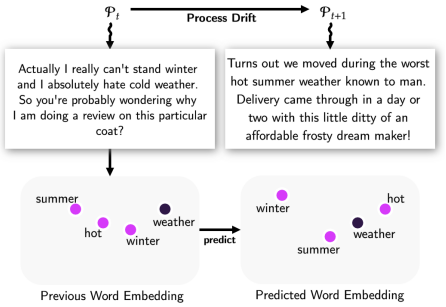

Figure 1: Overview of TransDrift model. We show an illustration of product review data showing that the data generating process undergoes drift between winter and summer. In these reviews, winters are characterized by the mentions of cold weather while summers are characterized by mentions of hot weather. Our model takes the past word embeddings as input to predict the embeddings for the drifted data distribution.

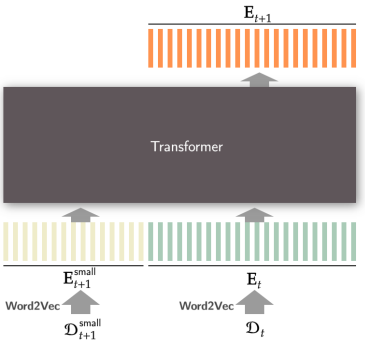

Figure 2: TransDrift model. Our model takes the word embedding of previous time-step as input optionally along with a few word embeddings of the current time-step trained with a small dataset.

together. This is achieved by predicting neighboring words for each word in the corpus $\mathcal{D}$. Backpropagating gradients from this prediction objective to the input word's embedding enables the learning of word embeddings. Consequently, these embeddings encode information about the common usage context of the word.

2.2 Transformer Architectures

The Transformer architecture processes a set of vectors, allowing each vector to interact flexibly with all others [26, 35]. In this model, a *transformer layer* takes N input vectors and maps them to N output vectors, facilitating interaction through self-attention mechanisms [35]. Following self-attention, each vector undergoes transformation via a MLP, enhancing the model's expressiveness. Residual connections are incorporated into both self-attention and MLP steps to improve gradient flow. To boost modeling capacity, multiple transformer blocks are stacked. Given their effectiveness in capturing complex interactions, we aim to leverage the Transformer architecture to monitor drift in word embeddings over time in this study.

3 METHOD

In this section, we propose a method to model the drift in word embeddings over time. We begin with the text distribution at each time-step $\mathcal{P}_t$, which provides us with a data sample $\mathcal{D}_t \sim \mathcal{P}_t$. Crucially, this distribution evolves over time: $\mathcal{P}_t \to \mathcal{P}_{t+1}$ as shown in Figure 1. As a result, the semantics and usage of words in the sampled datasets, $\mathcal{D}_t$ and $\mathcal{D}_{t+1}$, change over time. We aim for these evolving semantics to be reflected in word embeddings $\mathbf{E}_t$ at each time-step t, making them useful for downstream tasks.

While data at time t is extensive, $\mathcal{D}_{t+1}$ at time $t+1$ is often much smaller and may even be empty. Thus, while word embedding $\mathbf{E}_t$ can be accurately learned from $\mathcal{D}_t$ using standard methods like *word2vec*, however, directly learning $\mathbf{E}_{t+1}$ from $\mathcal{D}_{t+1}$ is usually ineffective or impossible if $\mathcal{D}_{t+1}$ is empty. Therefore, during training, we aim to learn drift dynamics that can be used at test time to predict $\mathbf{E}_{t+1}$ directly from $\mathbf{E}_t$, even when $\mathcal{D}_{t+1}$ is small or empty.

For this, we first train the embedding at time t using the large data set at time t and then use a Transformer to map the word embeddings at time-step t to the embeddings of the next time-step $t+1$. Formally,

$$\mathbf{E}_t = \text{TrainWordEmbeddings}(\mathcal{D}_t),$$
$$\mathbf{E}_{t+1} = \text{Transformer}_\phi(\mathbf{E}_t).$$

Using a Transformer model enables each word embedding prediction to consider the embeddings of all other words through attention mechanisms. This capability allows our model to learn complex embedding drift dynamics, enhancing prediction accuracy.

During inference, if a small dataset $\mathcal{D}^{\text{small}}_{t+1}$ is available at timestep $t+1$, we can use it to train embeddings for a subset of words. These resulting embeddings, $\mathbf{E}^{\text{small}}_{t+1}$, can serve as additional context for our model during prediction. Alongside embeddings from the previous timestep, our model predicts all embeddings for timestep $t+1$ as shown in Figure 2. This can be summarized as follows:

$$\mathbf{E}^{\text{small}}_{t+1} = \text{TrainWordEmbeddings}(\mathcal{D}^{\text{small}}_{t+1}),$$
$$\mathbf{E}_{t+1} = \text{Transformer}_\phi(\mathbf{E}_t, \mathbf{E}^{\text{small}}_{t+1}).$$

In our experiments, we shall show that providing such additional context can lead to moderate improvements in the prediction accuracy. For downstream applications, providing such additional embeddings can therefore be beneficial.

Training. For training, we assume that our historical data provides large datasets for both time-steps t and $t+1$ which we denote as $\mathcal{D}_t$ and $\mathcal{D}_{t+1}$. Taking these two datasets, we train the word embeddings as follows:

$$\mathbf{E}_t = \text{TrainWordEmbeddings}(\mathcal{D}_t),$$
$$\mathbf{E}_{t+1} = \text{TrainWordEmbeddings}(\mathcal{D}_{t+1}).$$

To train the Transformer, we minimize the following cosine embedding loss $\mathcal{L}^{\text{predict}}(\phi)$ for predicting the embedding at time $t+1$:

$$1 - \cos(\mathbf{E}_{t+1}, \text{Transformer}_\phi(\mathbf{E}_t, \mathbf{E}^{\text{small}}_{t+1})),$$

where $\cos(\cdot,\cdot)$ denotes cosine similarity.

Downstream Task. As our end goal of modeling the embedding drift is to help downstream task, we now describe how we

utilize our predicted word embedding to achieve this. For our purpose, we perform classification tasks for input review text (x) and ground truth (y). We train a downstream task neural network on the predicted word embeddings as follows:

$$f_\theta(\mathbf{x}; \mathbf{E}_{t+1}).$$

Given embedding $\mathbf{E}_{t+1}$, input $\mathbf{x}$ and target label $\mathbf{y}$, we learn the task-specific neural network at time-step $t+1$ as:

$$\mathcal{L}^{\text{task}}(\theta) = \text{CrossEntropy}(\mathbf{y}, f_\theta(\mathbf{x}; \mathbf{E}_{t+1})).$$

4 IMPLEMENTATION DETAILS

Before training word vectors, datasets undergo preprocessing: all corpus sentences are converted to lowercase, non-alphabetic characters are removed using a regular expression. NLTK's sentence and word tokenizer generate tokens from the processed data. Subsequently, stop-words are removed from the tokens. Finally, the tokenized data is employed to train a word2vec language model, resulting in 50-dimensional word embeddings.

We utilized the Word2Vec language model to train embeddings $\mathbf{E}_t$, $\mathbf{E}_{t+1}$, and $\mathbf{E}_{t+1}^{\text{small}}$ for datasets $\mathcal{D}_t$, $\mathcal{D}_{t+1}$, and $\mathcal{D}_{t+1}^{\text{small}}$, respectively, following incremental word embedding training as proposed by Kim et al. [24]. This method initializes word embeddings $\mathbf{E}_{t+1}^{\text{small}}$ and $\mathbf{E}_{t+1}$ for the drifted timestamps with $\mathbf{E}_t$ from the preceding timestamp. The Word2Vec model was initialized using $\mathbf{E}_t$ embeddings and subsequently retrained on $\mathcal{D}_{t+1}$, following the same preprocessing steps, to generate $\mathbf{E}_{t+1}$. A similar approach is used for $\mathbf{E}_{t+1}^{\text{small}}$. This guarantees that the embeddings are aligned.

We prepared 1,000 sets of $\mathbf{E}_t$, $\mathbf{E}_{t+1}$, and $\mathbf{E}_{t+1}^{\text{small}}$ from each dataset (Amazon, Yelp, and Synthetic). Each set served as a training example for the transformer, with input(X) consisting of word embedding vectors from $\mathbf{E}_t$ and $\mathbf{E}_{t+1}^{\text{small}}$, and target(Y) containing vectors from $\mathbf{E}_{t+1}$. Only common words across all 1,000 instances of $\mathbf{E}_t$ and $\mathbf{E}_{t+1}$ were selected to maintain coherence between training samples. Words common across the 1,000 instances of $\mathbf{E}_{t+1}^{\text{small}}$ were also included with the $\mathbf{E}_t$ embedding to provide context for predicting $\mathbf{E}_{t+1}$. This comprehensive approach ensures proper training of our transformer-based model.

The TransDrift model utilizes the transformer architecture, featuring an encoder and decoder, each consisting of 4 layers. Within each layer, there are 2 sub-layers: the first is a multi-headed attention layer, and the second includes a RELU layer sandwiched between 2 linear layers. Each sub-layer is followed by *layernorm*, which normalises the sum of the input and output and passes it to the next layer.

We historically possess datasets $\mathcal{D}_t$ and $\mathcal{D}_{t+1}$ for specific timestamps, corresponding to $\mathbf{E}_t$ and $\mathbf{E}_{t+1}$ embeddings. These datasets train our model, allowing it to capture drift patterns. In future timestamps where $\mathcal{D}_{t+1}$ is limited or absent, our model predicts $\mathbf{E}_{t+1}$ embeddings using learned drift patterns and available context from limited amount of $\mathcal{D}_{t+1}(\mathcal{D}_{t+1}^{\text{small}})$. As the model undergoes periodic training, the captured drift pattern evolves.

Downstream task is performed on reviews taken from both timestamps to check the robustness of our predicted embeddings.

5 EXPERIMENTS

The goal of our experiments is to show how well our model can accurately predict the drifted word embeddings relying on little or no data from the drifted text distribution. Furthermore, we also show the benefits of our predicted word embedding in improving the performance of downstream classification tasks. As an instance to test our idea, we intentionally choose simplest and widely used embedding method word2vec so that our results can be interpreted more generally.

Table 1: Comparison of word embedding prediction between our model and the baselines. We report the cosine similarity of the predicted embedding with the ground truth embedding trained using large amount of data from the drifted distribution. The predicted embeddings do not use any data from the drifted distribution. We note that our model, TransDrift, is significantly more accurate with respect to the baseline models.

Dataset	Drift Model		
	No-Drift	Additive	TransDrift
Synthetic	0.3200	0.3300	**0.7724**
Yelp	0.1900	0.7956	**0.8910**
Amazon	-0.0040	-0.0002	**0.8170**

5.1 Experiment Setup

5.1.1 Datasets. We evaluate our models on a synthetic dataset, Yelp Academic dataset [9] and Amazon Customer Review dataset [28]. For each dataset, we consider drift instances with each instance consisting of $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_2^{\text{small}}$ during training. Here, the subscript 1 denotes the source time-step $t = 1$ and subscript 2 denotes the next time-step (i.e. $t = 2$) in which the underlying text distribution has undergone a shift.

Synthetic Dataset. The dataset comprises multiple instances, each featuring a randomly generated sparse graph where nodes represent tokens from the vocabulary. Edge weights in the graph signify token co-occurrence patterns, and a random walk from a random node yields encountered tokens as text. Transition probabilities during the walk depend on edge weights, resulting in dataset $\mathcal{D}_1$. Edge weights are randomly modified to create a drifted data process. From this drifted graph, we perform another random walk to sample both a small dataset $\mathcal{D}_2^{\text{small}}$ and a large dataset $\mathcal{D}_2$.

Yelp Academic Dataset. For Yelp Academic Dataset, we use the businesses, reviews, and user data. For this, we divide the dataset into two parts by timestamp – reviews before the year 2016 and reviews after the year 2016. We denote these two parts as: $\mathcal{D}_1$ and $\mathcal{D}_2$. We take smaller subsets of $\mathcal{D}_2$ to obtain $\mathcal{D}_2^{\text{small}}$.

Amazon Customer Review Dataset. For Amazon Customer Review dataset, we separately consider the categories: *Books*, *Electronics*, *DVD*, and *Kitchen*. For this, we divide the dataset into two parts by timestamp – summer reviews and winter reviews. We call these two parts as: $\mathcal{D}_1$ and $\mathcal{D}_2$. We take smaller subsets of $\mathcal{D}_2$ to be $\mathcal{D}_2^{\text{small}}$.

5.1.2 Metrics. We evaluate prediction accuracy by computing cosine similarity between $\mathbf{E}_2$ learned from the full dataset $\mathcal{D}_2$ and our predicted embeddings, derived from previous embeddings $\mathbf{E}_1$ and $\mathbf{E}_2^{\text{small}}$. Additionally, downstream model accuracy reflects the benefits of our predicted embeddings on task performance.

Table 2: Comparison of the word embedding prediction performance under varying percentages of D_2^{small} data used. We report the average cosine similarity.

	Size of $\mathcal{D}_2^{\text{small}}$ as % of $\mathcal{D}_2$		
Dataset	30%	20%	0%
Synthetic	**0.8067**	0.7913	0.7724
Yelp	**0.9119**	0.9075	0.8910
Amazon	**0.8829**	0.8076	0.8170

Table 3: Qualitative analysis of nearest neighbors of the predicted word embeddings on Yelp dataset. For each prediction model, we find 30 nearest neighbors for each word shown in the first column. We then count the number of these nearest neighbors that are also the nearest neighbor in the target word embeddings. Thus, the higher number of nearest neighbors of our model TransDrift shows that our predicted embeddings agree significantly more with the target embeddings.

	# Common Neighbors		
Word	No-Drift	Additive	TransDrift
well	7	4	**7**
place	8	8	**15**
great	10	11	**13**
time	7	7	**12**
nice	11	9	**15**
customer	8	9	**10**
happy	3	1	**7**
people	9	8	**12**

Table 4: Downstream Prediction Results on Amazon Review (AR) and Yelp Datasets. Using embeddings from the evaluated methods, we train a downstream sentiment classifier and report its test accuracy. We note that the No-Drift model which re-uses the outdated embedding from the previous time-step suffers compared to TransDrift. TransDrift is significantly more accurate than the baseline models.

	Accuracy (in %)		
Dataset	No-Drift	Additive	TransDrift
AR-Electro	60.50%	60.56%	**69.60%**
AR-Kitchen	63.60%	63.52%	**75.70%**
AR-DVD	59.00%	59.03%	**63.50%**
Yelp	58.00%	60.00%	**65.00%**

5.1.3 Baselines. As no previous work directly tackles our problem setting, we develop the following baselines to show the efficacy of our model.

No-Drift Model. In this baseline for predicting the future embeddings, the modeling assumption is that the word embeddings do not undergo drift. That is, the model assumes that the embeddings learned at time-step 1 using $\mathcal{D}_1$ can be naively re-used at time-step 2 even though the underlying data distribution has drifted between timesteps 1 and 2. The goal of this comparison is to justify the need for predicting the word embedding instead of simply re-using the previous outdated embeddings.

Additive-Drift Model. In this baseline for modeling the embedding drift, we assume that the drift can be modeled by adding a constant embedding vector to all the words in vocabulary as proposed by [34]. That is, this model learns a vector Δ such that the embedding at time-step 2 can be predicted as $\mathbf{E}_2 = \mathbf{E}_1 + \Delta$. The goal of this comparison is to show that it is not enough to simply model the drift as a constant additive vector and it is required to model complex interaction and non-linear drift dynamics to predict the future embedding accurately.

5.2 Word-Embedding Prediction

We now evaluate the performance of word embedding prediction by the models.

5.2.1 Quantitative Evaluation. We perform a quantitative evaluation by reporting the average cosine similarity under two prediction regimes: with and without the available data from the drifted distribution.

Prediction with No Data. In Table 1, we present the average cosine similarity between the predicted word embeddings and the target embeddings. The target embeddings, denoted as $\mathbf{E}_2$, are obtained by applying *word2vec* to a large dataset collected from the drifted distribution. Notably, we generate embedding predictions without using any data from the drifted distribution. Our TransDrift model consistently outperforms all baselines across various datasets. Particularly, the No-Drift baseline performs poorly, indicating that relying on outdated embeddings is insufficient. Even when assuming drift as a constant vector added to all words (i.e., the Additive-Drift model), performance is better than the No-Drift model but significantly inferior to our model. This underscores the importance of capturing complex interactions and non-linear drift behavior, which our transformer-based predictor accomplishes effectively.

Prediction with Available Drifted Data. In Table 2, we show the effect of using increasingly larger amount of data $\mathcal{D}_2^{\text{small}}$ from time-step 2 to inform the word embedding prediction in our model. We note that with increasing the size of this data, we see an increase in prediction accuracy across all datasets. In deployment settings, this property may be useful to continually improve the embeddings as increasingly more data is gradually collected. Interestingly, we note that even with no data from the time-step 2, our prediction accuracy already surpasses all our baselines reported in Table 1 across all datasets.

5.2.2 Qualitative Evaluation. To analyze prediction results qualitatively, we select eight words from Yelp dataset: *well*, *place*, *great*, *time*, *nice*, *customer*, *happy* and *people*, computing their nearest neighbors using predicted embeddings from all models. The target embeddings are based on word embeddings trained using a large amount of data from time-step 2. A higher number of common nearest neighbors between predicted and target embeddings indicates model effectiveness. We report the number of common nearest neighbors for each word in Table 3. Notably, assuming No-Drift

Table 5: Text samples from AR dataset that were misclassified when using No-Drift model compared to our TransDrift model.

Review Text	Ground Truth	TransDrift
First off, the ipod jiggles no matter what you do, secondly, it doesn't stay straight on the power plug, it constantly tilts(the whole thing)...not worth $10	Negative	Negative
I bought this amazing product and now it is easy to have high quality music. Just plug the iPod to your music equipment and you are done.	Positive	Positive
I love these dishes! The proportions, color vibrance, surface wearability and chip-resistance can't be beat for the price! I suggest buying a few colors to mix and match, these dishes come in so many great colors! If you're considering a lesser-priced set (I was), spend a little bit more, the extra quality and classic style are definitely worth it	Positive	Positive

results in fewer nearest neighbors, indicating changing word usage over time due to underlying data drift. However, using TransDrift for embedding prediction yields the highest number of nearest neighbors compared to No-Drift and Additive Drift baselines.

5.3 Downstream Tasks

We assess how effectively our predicted embeddings enhance downstream task performance, especially under data drift conditions at time-step 2. If so then which prediction approach should be preferred. We compare different approaches for obtaining word embeddings under drift: *i) No-Drift*, *ii) Additive-Drift* and *iii) TransDrift*.

We train a downstream classification model using embeddings from each method and report test accuracy in Table 4. We train a binary classifier for Amazon Review dataset and multi-label classifier with 5 labels for the Yelp dataset. Results show that the *No-Drift model*, which re-uses outdated embeddings, and *Additive-Drift model* lag behind our *TransDrift model*, indicating the usefulness of embedding prediction by our approach. We further analyze downstream performance by presenting qualitative examples of text inputs from the drifted distribution misclassified by the *No-Drift model* but correctly classified by *TransDrift*, as shown in Table 5.

5.4 Ablation Study

To better justify our choice of architecture for TransDrift, we perform additional experiments that we describe here. In terms of architectural components, our model can be seen as Self-Attention + Feed Forward Network, while our baseline MLP can be seen as Feed Forward Network. We analyze the effect of this choice in our experiments. We compute the cosine similarity of the predicted embedding with the ground truth embedding trained using large amount of data from the drifted distribution. Our model, Transdrift, achieves *6.2%* higher similarity than the MLP model. Using embeddings from the evaluated methods, we train a downstream sentiment classifier and observe that TransDrift is *10.8%* more accurate than the MLP model. Details of the ablation study are available in [1].

6 RELATED WORK

Word Vectors. In the past decade, there has been significant interest in learning word representations [3–7, 12, 21, 29]. Mikolov et al. [29] propose CBOW and skip-gram architectures as the most common approach for learning high-quality word vectors from large text datasets. CBOW predicts the current word based on context words, while skip-gram predicts nearby context words. Bojanowski et al. [4] introduce a novel method by incorporating character n-grams to the skip-gram model, considering sub-word information to enhance embedding quality and predict embeddings for unseen words.

Data Drift in Text. Recent years have witnessed significant interest in analyzing text drift, yet efforts to model it are still in their infancy. Researchs such as [22], [23] and [38] highlight the adverse effects of drift on downstream performance when training and test sets diverge due to drift. [8, 27] define task instability with respect to word embeddings, identifying trade-offs between stability, precision, and model dimension. Instability in word neighbors, as observed in *word2vec* and *fasttext* embeddings, is discussed in [19] and [2]. [31, 36] define stability as percent overlap among neighbors, crucially serving as a task independent definition. Various factors affecting word stability and their impacts on downstream tasks are analyzed. Approaches like down-sampling by [20], drift reversal by [34], and evolutionary approach by [18] aim to stabilize word embeddings, yet they assume downstream tasks remain unaffected by data drift. In contrast, our work integrates drift and changing word semantics to enhance embedding and task performance. Xu et al. [37] propose a meta-learning approach for adapting word embeddings from source to target domains. However, unlike our method, it requires direct access to all previously seen corpora, making it unsuitable for domains lacking target data.

Contextual Embeddings. Contextual embeddings have also seen a rise alongside word2vec. However, word2vec is widely used in a lot of industrial applications [10, 13, 15, 32, 33], the scope of our work is to deal with drift in regular word2vec embeddings.

7 CONCLUSION

In this paper, we proposed TransDrift, a framework to track embeddings under data drift. We showed that using a transformer model perform this task effectively with no data. Optionally, our model can also leverage small amount of data from drifted distribution to further improve its prediction. Finally, by performing downstream tasks using the predicted embeddings, we show a significant performance improvement compared to other options. One of the future work can be to study multi-step word embedding prediction.

REFERENCES

[1] anonymized due to double blind review. [n. d.]. *Anonymized.* Technical Report. [available on request].

[2] Maria Antoniak and David Mimno. 2018. Evaluating the Stability of Embedding-based Word Similarities. *Transactions of the Association for Computational Linguistics* 6 (2018), 107–119. https://doi.org/10.1162/tacl_a_00008

[3] Oren Barkan, Avi Caciularu, Idan Rejwan, Ori Katz, Jonathan Weill, Itzik Malkiel, and Noam Koenigstein. 2021. Representation Learning via Variational Bayesian Networks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 78–88.

[4] Piotr Bojanowski*, Edouard Grave*, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146. https://arxiv.org/abs/1607.04606v2

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the association for computational linguistics* 5 (2017), 135–146.

[6] Danushka Bollegala, Mohammed Alsuhaibani, Takanori Maehara, and Ken-ichi Kawarabayashi. 2016. Joint word representation learning using a corpus and a semantic lexicon. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[7] Avi Caciularu, Ido Dagan, and Jacob Goldberger. 2021. Denoising word embeddings by averaging in a shared space. *arXiv preprint arXiv:2106.02954* (2021).

[8] Mansi Chugh, Peter A. Whigham, and Grant Dick. 2018. Stability of Word Embeddings Using Word2Vec. In *Australasian Conference on Artificial Intelligence*.

[9] Chris Crawford. 2018. Yelp Academic Data. https://www.kaggle.com/yelp-dataset/yelp-dataset.

[10] Leon Derczynski, Isabelle Augenstein, and Kalina Bontcheva. 2015. Usfd: Twitter ner with drift compensation and linked data. *arXiv preprint arXiv:1511.03088* (2015).

[11] Haim Dubossarsky, Daphna Weinshall, and Eitan Grossman. 2017. Outta control: Laws of semantic change and inherent biases in word representation models. In *Proceedings of the 2017 conference on empirical methods in natural language processing*. 1136–1145.

[12] Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting Word Vectors to Semantic Lexicons, NAACL.

[13] Hege Fromreide, Dirk Hovy, and Anders Søgaard. 2014. Crowdsourcing and annotating NER for Twitter# drift.. In *LREC*. 2544–2547.

[14] Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2018. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences* 115, 16 (2018), E3635–E3644.

[15] Josh Gordon. 2018. Introducing tensorflow hub: A library for reusable machine learning modules in tensorflow. https://medium.com/tensorflow/introducing-tensorflow-hub-a-library-for-reusable-machine-learning-modules-in-tensorflow-cdee41fa18f9.

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

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

[18] Yu He, Jianxin Li, Yangqiu Song, Mutian He, Hao Peng, et al. 2018. Time-evolving Text Classification with Deep Neural Networks.. In *IJCAI*, Vol. 18. 2241–2247.

[19] Johannes Hellrich and Udo Hahn. 2016. Bad company—neighborhoods in neural embedding spaces considered harmful. In *Proceedings of coling 2016, the 26th international conference on computational linguistics: Technical papers*. 2785–2796.

[20] Johannes Hellrich, Bernd Kampe, and Udo Hahn. 2018. The influence of down-sampling strategies on SVD word embedding stability. *arXiv preprint arXiv:1808.06810* (2018).

[21] Ziniu Hu, Ting Chen, Kai-Wei Chang, and Yizhou Sun. 2019. Few-shot representation learning for out-of-vocabulary words. *arXiv preprint arXiv:1907.00505* (2019).

[22] Xiaolei Huang and Michael J Paul. 2018. Examining temporality in document classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vol. 2.

[23] Zhenhao Huang and Chenxu Wang. 2020. Measuring the Semantic Stability of Word Embedding. In *CCF International Conference on Natural Language Processing and Chinese Computing*. Springer, 378–390.

[24] Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal Analysis of Language through Neural Language Models. arXiv:1405.3515 [cs.CL]

[25] Andrey Kutuzov, Lilja Øvrelid, Terrence Szymanski, and Erik Velldal. 2018. Diachronic word embeddings and semantic shifts: a survey. *arXiv preprint arXiv:1806.03537* (2018).

[26] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*. PMLR, 3744–3753.

[27] Megan Leszczynski, Avner May, Jian Zhang, Sen Wu, Christopher Aberger, and Christopher Ré. 2020. Understanding the downstream instability of word embeddings. *Proceedings of Machine Learning and Systems* 2 (2020), 262–290.

[28] Julian McAuley. 2018. Amazon Product Data. https://jmcauley.ucsd.edu/data/amazon/.

[29] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *International Conference on Learning Representations* (2013). https://arxiv.org/pdf/1301.3781.pdf

[30] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[31] Bénédicte Pierrejean and Ludovic Tanguy. 2018. Predicting word embeddings variability. In *The seventh Joint Conference on Lexical and Computational Semantics*. 154–159.

[32] Tim Sell and Willem Pienaar. 2018. Introducing Feast: an open source feature store for machine learning. https://cloud.google.com/blog/products/ai-machine-learning/introducing-feast-an-open-source-feature-store-for-machine-learning.

[33] Dan Shiebler, Chris Green, Luca Belli, and Abhishek Tayal. 2018. Embeddings@Twitter. https://blog.twitter.com/engineering/en_us/topics/insights/2018/embeddingsattwitter.

[34] Kevin Stowe and Iryna Gurevych. 2021. Combating Temporal Drift in Crisis with Adapted Embeddings. *arXiv preprint arXiv:2104.08535* (2021).

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 5998–6008. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[36] Laura Wendlandt, Jonathan K Kummerfeld, and Rada Mihalcea. 2018. Factors influencing the surprising instability of word embeddings. *arXiv preprint arXiv:1804.09692* (2018).

[37] Hu Xu, Bing Liu, Lei Shu, and Philip S Yu. 2018. Lifelong domain word embedding via meta-learning. *arXiv preprint arXiv:1805.09991* (2018).

[38] Yang Yang, Da-Wei Zhou, De-Chuan Zhan, Hui Xiong, and Yuan Jiang. 2019. Adaptive deep models for incremental learning: Considering capacity scalability and sustainability. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 74–82.