Temporal Generalization for Spoken Language Understanding

Judith Gaspers, Anoop Kumar, Greg Ver Steeg, Aram Galstyan
Amazon Alexa AI

Abstract

Spoken Language Understanding (SLU) models in industry applications are usually trained offline on historic data, but have to perform well on incoming user requests after deployment. Since the application data is not available at training time, this is formally similar to the domain generalization problem, where domains correspond to different temporal segments of the data, and the goal is to build a model that performs well on unseen domains, e.g., upcoming data. In this paper, we explore different strategies for achieving good temporal generalization, including instance weighting, temporal fine-tuning, learning temporal features and building a temporally-invariant model. Our results on data of large-scale SLU systems show that temporal information can be leveraged to improve temporal generalization for SLU models.

1 Introduction

Spoken Language Understanding (SLU) models play an important role in voice-controlled devices, such as Alexa or Google Home. Two common SLU tasks are intent classification (IC) and slot filling (SF). Given a user request, IC aims to extract the user's intent, while SF is a sequence labeling task which assigns a slot label to each of the tokens. For example, the user request "play volbeat" should be classified as *PlayMusic* by the IC task, while SF should assign the labels *O* and *Artist* to "play" and "volbeat", respectively. State-of-the-art approaches typically model the two tasks jointly via DNNs (Do and Gaspers, 2019; Chen et al., 2019).

In deployed industry SLU systems, new data continuously flows into the system, and the underlying data distributions keep drifting over time. In this paper, we focus on the setting of temporal *covariate drift*, where the distribution of utterances may change, but the correct label for an utterance or token remains fixed (i.e., no concept drift) (Schölkopf et al., 2012). Such data drifts happen, for example, because customer usage patterns change over time, as new movies are being released or new artists and songs become popular. Another cause of data drifts are seasonal changes or changes related to (re-)occurring events. For instance, the utterance "will it snow tomorrow" is more likely to appear during the winter than the summer season, and the utterance "put on the Christmas lighting" is likely uttered around Christmas.

To accommodate for temporal distributional changes, industry SLU models are typically retrained and redeployed over time; in the following sections, we also refer to this process as *model release*, and we assume that model releases are executed at fixed time intervals, e.g., once per month. We further assume that for each release, new labeled data become available, which were collected since the previous release, yielding data belonging to different time periods.

The common approach to utilize new data is to simply combine them with all previously available data, and subsequently split them into training, validation, and test datasets. We can then build and evaluate a model on these datasets, which we also refer to as *offline* data in this paper. In industry applications, SLU models are subsequently deployed to customers and have to perform well on incoming customer requests, which we also refer to as *online* data. Importantly, aiming to provide the best possible experience for our customers, our main goal is building models which perform well on the online rather than the offline data. Since the online data are not available for model building and evaluation, we need to utilize the offline data to build a model which generalizes well to unseen online data from the upcoming time period.

In this paper, we study this temporal generalization task assuming that data from several consecutive time periods, i.e., months in our case, are available, and we aim to build a SLU model which yields high performance on data from an upcoming

Proceedings of NAACL-HLT 2022: Industry Track Papers, pages 37 - 44

time period. We aim to improve performance over the common approach of simply combining all offline data, which ignores the temporal nature of the data and implicitly assumes that data from all time periods are equally useful for model training. Instead, relating back to the previous examples, we assume that modeling the temporal nature of the data may be beneficial and that data from certain time periods may be particularly valuable. For instance, data from recent time periods may better reflect upcoming trends, and w.r.t. seasonality patterns, data from the same period in previous years may be particularly valuable.

To tackle the task, we explore four directions: i) instance weighting based on our assumptions about the task, ii) temporal finetuning, iii) learning temporal features and iv) building a temporally-invariant model. We present extensive experiments on real-world SLU data of German and Portuguese voice-controlled devices. Our results indicate that temporal information can be leveraged to improve temporal generalization of SLU models. We also show that simple temporal fine-tuning is not very effective and in fact leads to performance drop in certain cases.

2 Related Work

To the best of our knowledge, the temporal generalization scenario studied in this paper has not yet been explored for SLU in the literature, which may be due to the fact that common Academic SLU datasets are rather small and do not have a temporal notation. In general, work addressing the temporal nature of SLU datasets has been limited. Kim et al. (2017) address temporal data drift by adapting from stale to current data with an adversarial domain adaptation approach, treating the stale and current dataset as source and target domain, respectively. Contrasting with our work, they assume the availability of data from the target and they focus on two time periods only. Other work has explored short-term temporal information, e.g., the utterance context provided by a couple of prior utterances to resolve ambiguities (Lin et al., 2021).

In NLP, previous research has explored the significance of temporal drift for several tasks, such as headline generation (Søgaard et al., 2021), sentiment analysis (Lukes and Søgaard, 2018) and named entity recognition (NER) (Rijhwani and Preotiuc-Pietro, 2020), providing evidence that model performance drops when training data age increases compared to the test data. However, despite this evidence, the vast majority of NLP research does not take the temporal nature of data into account for evaluation (Lazaridou et al., 2021).

Lazaridou et al. (2021) show that (downstream task) performance of pre-trained language models suffers when performance is measured on future data. Since (re-)training of larger language models is costly, the authors propose to update model parameters by executing few steps of gradient decent on new data. Other approaches to mitigate temporal drift include predictive feature selection for sentiment analysis (Lukes and Søgaard, 2018), and selecting data based on frequent n-grams for NER (Chen et al., 2021). The most similar to our work is the study conducted by Rijhwani and Preotiuc-Pietro (2020) who tackle temporal drift for a small-scale NER task, i.e., including only 3 named entities. They consider data from several consecutive years and aim to build a model which performs well on data of the following year, focusing – in line with the previously described work – on the effects of data receny. The best performance is achieved by using instance weighting of recent data and temporal finetuning for a Bi-LSTM-CRF with Flair and GloVe embeddings, respectively. By contrast, we study temporal generalization in the context of a large-scale SLU production system covering a large numbers of labels. We focus on smaller time periods, i.e., spanning one month instead of a year, and we consider cyclic/seasonal changes in addition to data recency. For this purpose, we include methods which have not yet been explored in temporal generalization tasks, such as building models that leverage temporally-invariant representations.

3 Method

Given labeled SLU data from several consecutive time periods, our goal is to build a model which generalizes well to unseen data from an upcoming time period. In the following, we first provide a formal definition for our tasks and subsequently present the modeling approaches.

3.1 Learning scenario

We assume that labeled data are available, which span N consecutive time periods, i.e., $D = [D_1, \ldots, D_N]$ with $D_i \in D = \{(x_{i,j}, y_{i,j})\}_{j=1}^{|D_i|}$, where $x_{i,j_1}, \ldots, x_{i,j_n}$ is an utterance with n tokens which was observed during time period D_i. For the SF task, a slot label is available for each token in

$x_{i,j}$, and $y_{i,j}$ is a sentence-level intent label for the IC task. In this work, each $D_i \in D$ comprises data of one month.

The goal is to build a model using $[D_1, \ldots, D_{N-1}]$ which generalizes well to the unseen data D_N. $[D_1, \ldots, D_{N-1}]$ corresponds to the offline data in a release scenario, while D_N corresponds to the online data.

Note that this learning scenario corresponds to the task of domain generalization (DG) (Wang et al., 2021) when considering the time periods as individual domains. In DG, one aims to build a model given several different but related domains (datasets) which works well on data of a new (unseen) domain during testing. However, in work addressing DG, typically the domains under consideration are more distant than the datasets of different time periods in our scenario, which are in fact drawn from the same overall source (domain), and contrasting with our scenario, the domain datasets in DG usually do not have a natural order.

Note further that DG differs from domain adaptation (DA) in that DA assumes the availability of some (unlabeled) data of the target domain, which can be utilized for adaptation.

3.2 Basic SLU model

Our basic SLU model is a common state-of-the-art SLU architecture for joint intent classification and slot filling. It is comprised of a BERT encoder, an intent decoder and a slot decoder. The BERT encoder's outputs at sentence and token level are used as inputs for the intent and slot decoders, respectively. The intent decoder is a standard feed-forward network including two standard dense layers and a softmax layer on top. The slot decoder uses a CRF layer on top of two dense layers to leverage the sequential information of slot labels. As loss we use a weighted sum of the loss of IC L_i and the loss of SF L_s, i.e.

$$L = \lambda_i L_i + \lambda_s L_s, \quad (1)$$

where λ_i and λ_s are weights. We use cross-entropy and CRF loss for IC and SF, respectively.

3.3 Instance weighting

Instance weighting assigns a weight to each training data instance. In a DA task, a weight may be selected such that it reflects the instance's similarity to the target (Jiang and Zhai, 2007). However, since we do not assume the availability of data from the target time period, we cannot compute such similarity scores. Instead, we simply weight instances based on our assumptions about the task.

We assume that recent data, i.e., data from the period prior to the target period, may be particularly valuable, because this period may better reflect recent and upcoming trends, Moreover, instance weighting of data from a recent year has already been shown to improve performance on data of a future year for a small-scale NER task (Rijhwani and Preotiuc-Pietro, 2020). On the other hand, with respect to seasonal changes, data from the same period in previous years could also be of particular value. Thus, taken together, we explore three instance weighting strategies based on recency, seasonality, and combination of both:

1. Reweight each data instance in the period prior to the target period by a weight $w > 1$ (i.e., reweight D_{N-1}),
2. Reweight each data instance from the same calendar month as the target data by $w > 1$,
3. Reweight all instances from either the same calendar month as the target data or from the period prior to the target data by $w > 1$.

Given an utterance and a corresponding weight, we weight the losses of both IC and SF.

The described instance weighting techniques do not require any temporal information during the application phase and no architectural changes.

3.4 Temporal finetuning

We explore temporal finetuning, as it has been previously shown to improve performance on other temporal tasks. In particular, Rijhwani and Preotiuc-Pietro (2020) i) trained Bi-LSTM-CRF models for a small-scale NER task on data from several years, and ii) fine-tuned these models on data of the most recent year, yielding improved performance over the initial models when evaluated on data of a future year. Similarly, we first train a model on all offline data, i.e., on $D_1 \cup \ldots \cup D_{N-1}$, and subsequently we finetune it on data of the most recent offline time period, i.e., on D_{N-1}.[1]

[1] We do not explore sequential temporal finetuning, i.e., first train on D_1, then on D_2, etc. This approach degraded performance on the small-scale NER task explored by Rijhwani and Preotiuc-Pietro (2020), and we expect it to not perform well for our task either, as it corresponds to a life-long learning scenario in which catastrophic forgetting of previously acquired knowledge is a known issue.

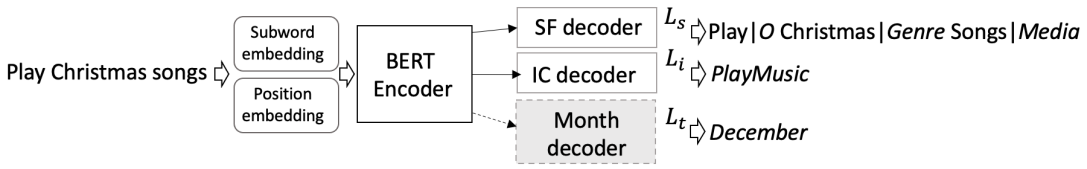

Figure 1: SLU architecture for joint IC and SF with an auxiliary task for predicting the month of an input utterance. While the model is trained by alternating across tasks, during inference only the SF and IC decoders are used.

This technique does not require any temporal information during the application phase and no architectural changes.

3.5 Learning temporal features

We hypothesize that learning temporal features could be beneficial for our SLU task and therefore aim to build a temporally-aware model. For this purpose, we explore two established techniques for injecting auxiliary information, i.e., i) via additional input features, and ii) via an auxiliary task. The training of an auxiliary prediction task to improve embeddings is sometimes called self-supervised learning, and has also been shown to improve generalization in vision tasks (Albuquerque et al., 2020).

3.5.1 Using additional input features

We define a special token for each month, e.g., "[JAN]", . . . "[DEC]". For each utterance, information about the month in which it was observed is added before model training and inference using the corresponding special token. E.g., "[DEC] play Christmas songs" indicates that the utterance "play Christmas songs" was observed in December.

While this technique does not require any architectural changes, temporal information is needed during the application phase, which, however, should be easily accessible in most cases.

3.5.2 Using an auxiliary task

We extend the basic SLU model described in section 3.2 by an auxiliary task which predicts the month given an utterance. Specifically, we apply a multi-task model which is comprised of a joint IC and SF task and an additional classification task; the overall architecture is illustrated in Fig. 1. The auxiliary task decoder is a standard feed-forward network comprising two standard dense layers and a softmax layer on top. We optimize the model by alternating across the two subtasks (joint SF and IC vs month prediction), and we use a combined loss

$$L = \lambda_i L_i + \lambda_s L_s + \lambda_t L_t \quad (2)$$

where L_i, L_s and L_t are the losses of the IC, SF and month prediction tasks, respectively, and λ_i, λ_s and λ_i are weights. We use cross-entropy loss for the IC and month prediction tasks and CRF loss for the SF task.

For inference, we apply only the joint SF and IC task, and temporal information is not required during the application phase. The intuition is that via the joint training, temporal information is acquired by the model which can then influence the SF and IC predictions during inference.

3.6 Building a temporally-invariant model

Relating back to section 3.1, our learning scenario corresponds to the task of DG when considering the time periods as individual domains. We selected a popular direction explored in DG (Wang et al., 2021) and DA (Ramponi and Plank, 2020), i.e., invariant representation learning. The intuition is that by removing information which is specific to individual domains, the model should generalize better to an unseen target domain. Thus, contrasting with the approaches described in the previous section which aim to learn temporal features, in this approach we aim to build a temporally-invariant model by removing features which are specific to certain time periods.

Note that both approaches may be reasonable, as there may be different kinds of temporal features and artifacts related to our data, out of which we may want to leverage some, but abstract away from others. For instance, it may be beneficial if the models learn a notation of seasonality and/or recency, but we may want to abstract away from artifacts related to out-dated trends, annotation inconsistencies across time, etc.

One established approach to domain-invariant representation learning is adding an auxiliary domain classifier to a main task predictor, and then

optimizing for an accurate task predictor while applying an adversarial training strategy to confuse the auxiliary domain classifier by making the features from source and target domain indistinguishable, thus yielding domain-invariant features. A gradient-reversal layer can be applied for this purpose (Ganin and Lempitsky, 2015; Ganin et al., 2016). We adapt the approach from Ganin and Lempitsky (2015) to build a temporally-invariant model, i.e., we apply it with our SLU model as the main task predictor and using an auxiliary month classifier instead of the domain classifier (and BERT as the feature extractor).

As in case of learning temporal features using an auxiliary task, for inference we apply only the main task, and thus temporal information or architectural changes are not required during the application phase.

4 Experiments

4.1 Data

We use data from large-scale industry SLU systems comprising user requests to voice-controlled devices; all requests were de-identified, annotated with intent and slot labels, and marked with a time stamp. We collected data for two languages, i.e. German and Portuguese, and three domains, i.e. *Music*, *Video* and *Shopping*. The data range from May 2019 to December 2020, and we split them into one dataset per month based on timestamps, resulting in 20 datasets $D_1, \ldots, D_{20}$ for each domain and language. Thus, one dataset is available per month, domain and language. For each domain and language, D_{20} is used for testing, and for $D_1, \ldots, D_{19}$ we create training and validation datasets. For German, for each domain we have more than 100,000 data instances available, while for Portuguese for each domain the data amounts are on the order of tens of thousands.

Qualitative data analyses indicate that both gradual and seasonal drifts are indeed present in the data, but there are domain-specific differences. Due to confidentiality reasons, a detailed data analysis is beyond the scope of this paper.

4.2 Experimental setup

For each domain and language, we use the $D_1, \ldots, D_{19}$ training datasets for model training, and D_{20} for testing. Since we do not have access to target period data, we study two options for creating an offline validation dataset:

1. val_a comprises the offline validation data from $D_1, \ldots, D_{19}$. This corresponds to the common approach.
2. val_r comprises only recent offline validation data, i.e., the validation data of D_{19}.

We train and evaluate our modeling approaches on the described setup. As baseline, we train a model, i.e., the basic SLU model described in section 3.2, following the common approach of simply training on the combined offline data (without leveraging any temporal information). In the following, we refer to this approach as *concat*.

We measure performance using a semantic error rate, which measures intent classification and slot filling jointly and is defined as follow:

$$\textit{SemER} = \frac{\#(\text{slot+intent errors})}{\#\text{slots in reference + 1}} \quad (3)$$

4.3 Settings

We used pre-trained multilingual BERT (Devlin et al., 2019) (size 768, 110M parameters)[2], and max-pooling for sentence representations. Each of our decoders has 2 dense layers of size 768 with gelu activation. The dropout values used in IC, SF and month decoders are 0.5, 0.2 and 0.5, respectively. We used equal weights for λ_s and λ_i (1.0:1.0) and Adam optimizer with a learning rate of 0.1 and a Noam learning rate scheduler. We trained our models for 20 and 25 epochs for German and Portuguese, respectively, with a batch size of 32. These hyper-parameters were used for all models (where they apply). The best models were selected on offline validation data (val_{all} or val_{rec}). We tried $w \in [2, 5]$ and we varied λ_t from 0.2 to 0.6. Each model was trained on a single GPU.

5 Results and discussion

The results on the "online" test data for using either all offline validation data val_{all} or recent validation data val_{rec} to select the best model are shown in Table 1. For confidentiality reasons, we report the relative change in SemER compared to the *concat* baseline using val_{all}. In the following, we discuss the results w.r.t. different research questions.

[2]The model is taken from https://github.com/google-research/bert/blob/master/multilingual.md (Apache 2.0) and was used for experiments only, not for production cases.

	German						Portuguese					
Method	**Music**		**Video**		**Shopping**		**Music**		**Video**		**Shopping**	
	val_a	val_r	val_a	val_r	val_a	val_r	val_a	val_r	val_a	val_r	val_a	val_r
Concat	0	-1.48	0	-3.0	0	-4.59	0	-1.67	0	**-11.0**	0	-0.54
Weight prev. period	-1.48	-0.67	**-2.96**	**-4.47**	**-8.68**	-7.06	-1.89	-0.33	-4.85	-4.86	-1.99	-5.87
Weight same month	-1.75	-0.47	0.84	-1.48	1.49	-5.09	0.39	-0.78	-4.56	-8.73	-0.09	-3.07
Weight both	1.34	-0.2	-1.96	-2.96	-4.47	**-9.31**	-1.83	-0.39	**-7.15**	-5.14	-2.8	-5.78
Temporal finetuning	-0.2	-0.2	-1.76	-3.04	4.84	-2.73	0.06	2.33	-3.05	0.11	3.43	0
Month feature	-0.94	-1.28	1.045	1.52	-1.61	-0.99	-1.1	1.0	-3.38	-1.69	-2.35	3.61
Auxiliary task	-1.75	**-1.75**	-1.72	-2.32	-2.98	-7.94	**-3.06**	**-2.22**	-5.96	-2.37	**-8.57**	-6.23
Temp.-invariant	**-3.63**	-0.74	-0.92	1.76	-4.84	-5.58	-2.44	-0.67	-6.65	-1.19	-3.88	**-9.2**

Table 1: Results on the "online" test data for using either all offline validation data (val_a) or recent offline validation data (val_r). We report the relative change in SemER compared to the *concat* baseline using val_a.

RQ 1: Can temporal information be leveraged to improve temporal generalization for SLU? Across all domains and languages, improvements in SemER on future data can be achieved by taking the temporal nature of data into account. The best methods differ across domains and languages, which is expected, given that there are domain and language specific differences w.r.t. seasonal and gradual shifts. However, two of the methods yield consistent gains across all domains and languages, i.e., instance weighting of the previous time period and using an auxiliary task yield improved performance compared to the baseline for all considered conditions. Using an auxiliary task achieves the best performance most often.

RQ 2: What is the impact of seasonality and re-occurring events vs recency effects? Previous work in NLP on temporal adaptation and generalization has focused on larger time periods and the effects of data recency, showing strong performance for instance weighting of recent data and temporal finetuning on a small-scale NER task (Rijhwani and Preotiuc-Pietro, 2020). By contrast, our domain datasets cover smaller time periods, with different seasonal effects and re-occruring patterns.

On our task, temporal finetuning gives mixed results, with decreasing performance in several cases. We assume that the models may overfit to the recent data, and some previously acquired knowledge related to older time periods might have been forgotten. However, unlike in the NER task which included only the three named entities *PER*, *LOC*, and *ORG* and in which there might be mostly gradual temporal trends, in our task seasonal drifts exist, potentially making certain older knowledge more relevant. By finetuning on recent data, the models may lose too much relevant seasonal knowledge, harming performance for domains with changes related to seasons or re-occurring events. Instance weighting of recent data gives consistent improvements, which is in line with previous findings, while instance weighting of the same time period gives mixed results, i.e., it helps in some cases, but decreases performance in others. To some extent this may be due to domain-specific differences. However, an issue might also be that there can be conflicting seasonal and gradual drifts. In particular, weighting is performed at the dataset level and a dataset from a year ago might include relevant seasonal data instances, but also less useful data instances such as data related to older (already outdated) trends. The negative effects can be mitigated to some extent by selecting the best model on recent validation data, which yields consistent gains in performance across all domains and languages. Future work may explore how to disentangle these effects, and in temporal DA scenarios one may select utterances based on the similarity to the target. However, in our scenario which does not assume the availability of target period data, modeling seasonal and re-occurring patterns indirectly via an auxiliary month prediction task appears to be a better choice in most cases, yielding consistent – and in most cases higher – gains.

How to create a validation dataset without having access to target data? For half of the domain-language pairings, performance is improved by using a recent offline validation dataset. The choice of the best validation dataset may be both method-specific and domain-specific, as drifts differ across domains.

Interestingly, performance of the concat baseline model, which ignores the temporal nature of the data, is consistently improved across domains and languages when using recent validation data. This shows that model performance can also be improved by taking the temporal nature of the data into account for creating a validation dataset in-

stead of model building.

Should we aim for a temporally aware or invariant model? In this paper, we have explored both building temporally-aware and temporally-invariant models, since there may be different kinds of temporal features and artifacts related to our data, out of which we may want to leverage some, but abstract away from others. Our results show consistent gains for temporally-aware models using an auxiliary month classifier as well as gains in all but one case for temporally-invariant models, with temporally-aware models giving better performance in most, but not all cases. Thus, both directions appear to be generally promising.

Future work may explore different approaches to learning temporally aware or invariant models, for instance, by exploring others DG approaches in the latter case. One potentially interesting direction is to learn disentangled representations that separate temporally-invariant and seasonal components.

6 Conclusion

We studied a temporal generalization task in which we used offline data of time periods spanning one month each to build a model that performs well on future online data. We explored four directions to leverage temporal information which are rather easy to apply in production, i.e., i) instance weighting based on our assumptions about the task, ii) temporal finetuning, iii) learning temporal features and iv) building a temporally-invariant model. Our results on real-world SLU data covering two languages and three domains each show that temporal information can be leveraged to improve temporal generalization for SLU. While several of the explored methods provide consistent gains across all domain-language pairings, the best methods differ, as different domain datasets have different gradual and seasonal drifts. Moreover, our results indicate that methods, such as temporal finetuning, which have been previously shown to provide strong performance on small-scale academic tasks with longer time periods and mostly gradual temporal drifts, do not necessarily yield the best performance in our large-scale SLU task including seasonality patterns.

Acknowledgements

We would like to thank Yannick Versley and the NAACL reviewers for helpful feedback for revising the paper.

References

Isabela Albuquerque, Nikhil Naik, Junnan Li, Nitish Shirish Keskar, and Richard Socher. 2020. Improving out-of-distribution generalization via multi-task self-supervised pretraining. *CoRR*, abs/2003.13525.

Q. Chen, Z. Zhuo, and W. Wang. 2019. Bert for joint intent classification and slot filling. *arXiv:1902.10909*.

Shuguang Chen, Leonardo Neves, and Thamar Solorio. 2021. Mitigating temporal-drift: A simple approach to keep ner models crisp. In *SOCIALNLP*.

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.

Quynh Ngoc Thi Do and Judith Gaspers. 2019. Cross-lingual transfer learning for spoken language understanding. *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271, Prague, Czech Republic. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017. Adversarial adaptation of synthetic or stale data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1297–1307, Vancouver, Canada. Association for Computational Linguistics.

Angeliki Lazaridou, Adhiguna Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Sebastian Ruder, Dani Yogatama, Kris Cao, Tomás Kociský, Susannah Young, and Phil Blunsom. 2021. Mind the gap: Assessing temporal generalization in neural language models. *Proceedings of NeurIPS*, abs/2102.01951.

Tzu-Hsiang Lin, Yipeng Shi, Chentao Ye, Yang Fan, Weitong Ruan, Emre Barut, Wael Hamza, and

Chengwei Su. 2021. Contextual domain classification with temporal representations. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers*, pages 41–48, Online. Association for Computational Linguistics.

Jan Lukes and Anders Søgaard. 2018. Sentiment analysis under temporal shift. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 65–71, Brussels, Belgium. Association for Computational Linguistics.

Alan Ramponi and Barbara Plank. 2020. Neural unsupervised domain adaptation in NLP—A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6838–6855, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Shruti Rijhwani and Daniel Preotiuc-Pietro. 2020. Temporally-informed analysis of named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7605–7617, Online. Association for Computational Linguistics.

Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. 2012. On causal and anticausal learning. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 459–466.

Anders Søgaard, Sebastian Ebert, Jasmijn Bastings, and Katja Filippova. 2021. We need to talk about random splits. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1823–1832, Online. Association for Computational Linguistics.

Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, and Tao Qin. 2021. Generalizing to unseen domains: A survey on domain generalization. In *IJCAI*.