Towards Realistic Single-Task Continuous Learning Research for NER

Justin Payan[1]*, **Yuval Merhav**[2], **He Xie**[2], **Satyapriya Krishna**[2],
Anil Ramakrishna[2], **Mukund Sridhar**[2], **Rahul Gupta**[2]
[1]University of Massachusetts Amherst, MA, USA
[2]Amazon Alexa AI, MA, USA
[1]jpayan@umass.edu
[2]{merhavy, hexie, satyapk, aniramak, harakere, gupra}@amazon.com

Abstract

There is an increasing interest in continuous learning (CL), as data privacy is becoming a priority for real-world machine learning applications. Meanwhile, there is still a lack of academic NLP benchmarks that are applicable for realistic CL settings, which is a major challenge for the advancement of the field. In this paper we discuss some of the unrealistic data characteristics of public datasets, study the challenges of realistic single-task continuous learning as well as the effectiveness of data rehearsal as a way to mitigate accuracy loss. We construct a CL NER dataset from an existing publicly available dataset and release it along with the code to the research community[1].

1 Introduction

Data privacy is a hot topic in ML, gaining attention in both industry and academia (Papernot et al., 2016; Perera et al., 2015). One of the topics of interest is data retention, which can be improved by training models incrementally (Wu et al., 2019). An ideal training regime would involve continuously updating a model on newly acquired data, then deleting the data. Benchmarking CL strategies today is still highly nonstandard in academic research (Maltoni and Lomonaco, 2019).

One key difference between real-world and academic datasets is the dynamic nature of the former. Academic datasets are often static and contain data that is annotated all at once based on fixed annotation guidelines. When building real-world applications, such data collection and annotation workflow is often not realistic. Rather, an initial dataset is created and then is evolved over time based on usage pattern changes and business needs. For example, new labels are added periodically, data distribution changes significantly due to seasonality or other factors, annotation guidelines are updated, etc. While such datasets exist in industry, they are often confidential or proprietary and cannot be shared with the research community.

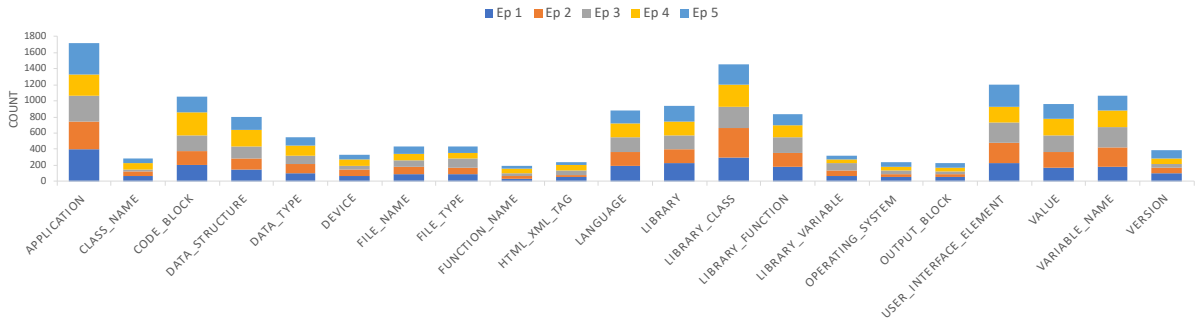
Consequently, the academic CL research focus has been mainly on the multi-task learning scenario, where the same model is required to learn a number of isolated tasks incrementally without forgetting how to solve the previous ones. In this work we tackle the single-task scenario using the Named Entity Recognition (NER) task. There is only one task, but it evolves over time due to data distribution shift, introduction of new labels, or other factors. Single-task is often considered to be more difficult than multi-task (Kemker et al., 2018; Kemker and Kanan, 2018; Maltoni and Lomonaco, 2019) and is also a common real-world scenario.

To the best of our knowledge, there are no public NLP benchmarks specifically designed for single-task CL. In order to study this problem we pick the recent StackOverflowNER dataset (Tabassum et al., 2020). The dataset authors' motivation was studying named entity recognition in the social computer programming domain, not continuous learning. However, the characteristics of the dataset are ideal for a study in CL. It spans roughly 10 years (from September 2008 to March 2018) of question-answer threads that are manually annotated with close to 30 types of entities. The dataset is also very diverse and has a large sample size – other public NER datasets are too small or contain only a few entity types. Finally, the manual annotation process resembles that of industrial use cases, where the labeling process might be subject to noise and human error.

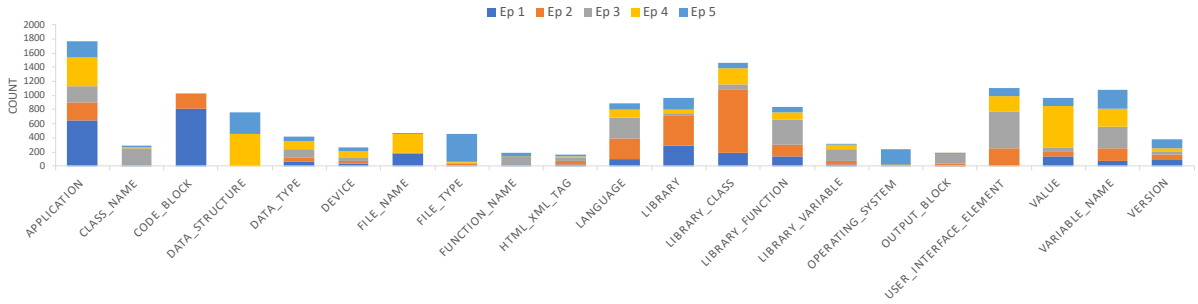
In order to simulate CL we split the data into time-based episodes and train an NER model incrementally over 5 episodes. Our results show no regression and limited forgetting. To present a more realistic challenge, we propose a configurable

*Work completed while first author was an intern at Amazon Alexa AI.

[1]https://github.com/justinpayan/StackOverflowNER-NS



(a) Temporal



(b) Skewed

Figure 1: Entity type distribution across episodes comparing the temporal and skewed episodes. Each vertical bar has the frequency for each of the 5 episodes. For readability we removed types with low counts in each episode.

distribution-based sampling of data inspired by our experiences with a confidential industrial dataset. We show that our sampled episodes can be used to study the effectiveness of different single-task CL strategies in the context of NER. The resulting dataset is the main contribution of this work.

2 Continuous Learning

Strategies. The main focus in training deep learning models in CL fashion is prevention of catastrophic forgetting (Kirkpatrick et al., 2017). Neural networks trained on new data tend to do poorly on old data and to mitigate catastrophic forgetting different strategies have been proposed, such as specific architectures for CL (Lomonaco and Maltoni, 2017; Rusu et al., 2016), regularization techniques (Kirkpatrick et al., 2017; Li and Hoiem, 2017), and data rehearsal/replay where small subsets of old data (real or generated) is periodically supplied to the model during training on new data (Sun et al., 2019; Shin et al., 2017). The latter is considered a strong CL baseline (Maltoni and Lomonaco, 2019) and thus we use this approach in this study. We also compare against a variation of the replay-based GDumb baseline (Prabhu et al., 2020). GDumb collects examples into a memory buffer with a limited budget size k, balancing the distribution over labels by greedily sampling under-represented label types and ejecting over-sampled label types. The model trains on the buffer after all tasks are seen.

Our CL model. Our model design is inspired by LAMOL (Sun et al., 2019) and adapted for NER. We employ a pre-trained GPT-2 language model base (Radford et al., 2019), then 2 layers of bi-LSTM with 768 dimensions in each direction, a $\tanh$ non-linearity and linear transformation (1536 by number of labels), and a CRF layer to predict labels. All parameters besides the GPT-2 base (pre-trained on OpenAI's WebText) are randomly initialized, and we train or finetune all parameters during training. Training on all 5 episodes takes less than 12 hours on an NVIDIA Tesla M40 GPU for all experimental settings. We assume that all entity types are known in advance so we do not need to expand the label size in a later episode if a new label is introduced. In our experiments, our baseline is a model fine-tuned on all training data. We compare the baseline to GDumb and two CL strategies: training with and without data replay.

Data replay. For each episode (barring the first), we set the size of replayed examples to be sampled from previous episodes to 20% of the size of the current episode's training set. An equal number of

replayed examples are sampled from each previous episode. To apply GDumb to NER, we add examples containing under-represented entity types to the buffer, and we eject examples which have the maximum value for their least well-represented entity type.

3 Experimental Setup

3.1 Time-Based Episodic Setup

Our first motivation is to investigate continual learning over time. We construct our continual learning datasets from StackOverflowNER, a dataset of questions and answers on StackOverflow annotated with 28 entity types (Tabassum et al., 2020). We combine StackOverflowNER's training and development sets to construct a pool for sampling training episodes, and we use the test set as a pool for sampling test episodes. All data splits and code are available at https://github.com/justinpayan/StackOverflowNER-NS.

Episode	Date Range	Train / Test Size
1	8/4/2008 – 6/26/2012	2551 / 775
2	6/27/2012 – 3/13/2014	2444 / 665
3	3/14/2014 – 6/27/2015	2243 / 521
4	6/28/2015 – 10/1/2016	2450 / 496
5	10/2/2016 – 3/27/2018	2386 / 632

Table 1: Date boundaries for each episode.

We first split the StackOverflowNER data into 5 time-based episodes. The StackOverflowNER dataset does not have timestamps, so we align their annotated examples with posts in the StackOverflow data dump. We select date boundaries for each episode to obtain roughly equal-sized training and test sets. Table 1 lists the dates.

3.2 Results

Figure 1a shows the distribution of each entity type across the 5 episodes. While some entity types are more common than others, the frequency distribution is consistent across episodes. The percentage of examples tagged with a particular entity type does not change much across episodes and there are no deletions or additions of new entity types over time. Such data characteristics are not realistic for a real-world application evolving over 10 years.

We train our model incrementally on the 5 episodes with and without data replay and compare it to a baseline model that is trained on all data at once in a non-CL fashion. Table 2 shows the averaged F1 score over the 5 episodes' test data (comprehensive results can be found in Appendix A). Not surprisingly, training incrementally performs on-par with training on all data at once, meaning that if there is any catastrophic forgetting, it does not impact the test performance of the model. As such, applying data replay that is supposed to mitigate catastrophic forgetting has no benefit and even results in a mild performance degradation. Preliminary manual analysis suggests that degredation stems from memorization of infrequent patterns sampled in the relatively small replay set. Given these data characteristics and results, it is clear that the dataset, in this format, is not proper for a comparison of CL strategies.

3.3 Skewed Class Distribution Setup

Motivated by our findings, we create an updated version of the episodic dataset based on more realistic assumptions. The first assumption is of data distribution shift and variance. Data distribution shift is expected due to various factors such as seasonality. A second factor is annotation cost. When a model is doing well on specific types of data/labels, there is no need to continue annotating similar examples and labels. We modify the StackOverflow dataset by sampling the distribution over entity types from a Dirichlet distribution for each episode. To simplify, we assume independence between entity types, although entity types often co-occur.

We first compute the distribution over entity types in the training pool, and denote that with α. We then sample distributions for the 5 training episodes, $\{\mathbf{X}_i^{tr}\}_{i=1}^5 \sim Dir(c\alpha)$ and the 5 test episodes $\{\mathbf{X}_i^{te}\}_{i=1}^5 \sim Dir(\mathbf{X}_i^{tr})$. We set $c = 5$ but the parameter can be changed to increase or decrease variance. To sample the train (test) episodes, we cycle through the episodes, each time selecting an entity type from the episode's distribution and then selecting an example containing that entity type from the train (test) pool without replacement.

In addition to modeling distribution shift, we also introduce class incrementality. We select 3 entity types that are relatively frequent: CODE_BLOCK, DATA_STRUCTURE, and USER_INTERFACE_ELEMENT. We simulate the data shift by removing the CODE_BLOCK entity in episode 3 and onward, adding the DATA_STRUCTURE entity only in episodes 4 and 5,

		Overall	CodeBlock	DataStruct.
Temporal	Baseline (non-CL)	51.36	25.67	75.27
	CL w/o Replay	51.52	28.59	73.76
	CL w/ Real Replay	51.12	26.41	72.82
Skewed	Baseline (non-CL)	52.24	12.51	32.03
	CL w/o Replay	42.61	0.00	32.60
	CL w/ Replay	49.82	7.74	33.82
	GDumb (500)	24.28 ± 0.98	6.81 ± 0.49	7.80 ± 4.25
	GDumb (1000)	35.41 ± 0.90	8.10 ± 0.60	24.09 ± 1.38
	GDumb (1500)	40.19 ± 0.67	8.82 ± 0.54	27.46 ± 1.52

Table 2: Overall and selected entity type F1 scores after training incrementally over all 5 episodes vs on all training data at once. All scores are averaged over all 5 episodes' test sets. We also compare against the GDumb baseline, with memory budgets of 500, 1000, or 1500 examples. We run GDumb over 10 random orderings within each episode, and report means and standard deviations.

and removing the USER_INTERFACE_ELEMENT entity from episode 1. To achieve this, each time we sample one of these entity types in a disallowed episode, we put that sample back into the pool.

3.4 Results

Figure 1b shows the distribution of each entity type across the 5 skewed episodes. In comparison to Figure 1a, one can see the increased variance of the distribution across episodes. Appendix B shows further comparisons between the skewed and temporal settings. We find the degree of variance to be similar to that of our confidential industrial NER dataset. Following the previous model training procedure, we train our model incrementally on the 5 skewed episodes with and without data replay and compare it to a baseline model that is trained on all data at once in a non-CL fashion. Table 2 shows the averaged F1 score over the 5 episodes' test data. Contrary to the previous setup, we see that the non-CL baseline heavily outperforms CL without replay. Data replay helps, but there is still a gap in performance. Even with a buffer size of 1500, GDumb greatly underperforms even the continual learning setup without replay. As GDumb is a strong baseline, this suggests the setting is quite difficult.

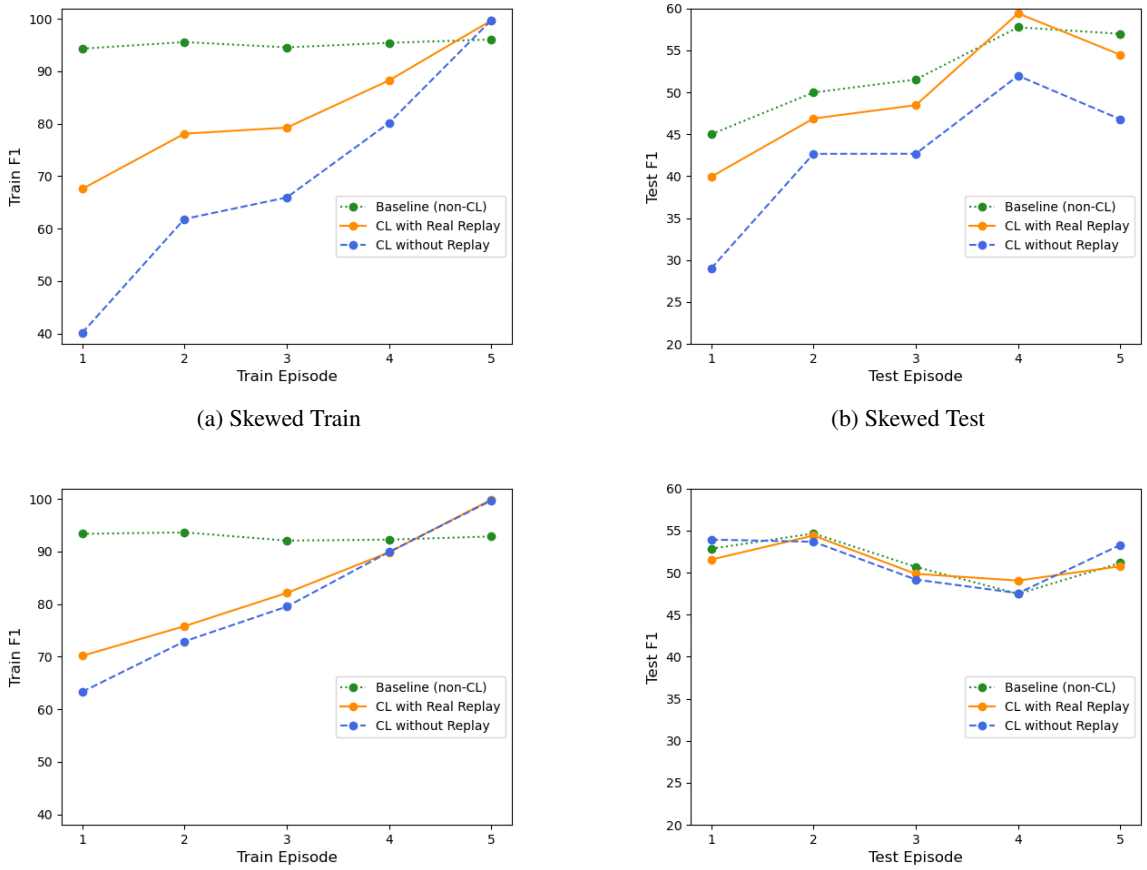
We can also see the impact of excluding CODE_BLOCK from episode 3 onward. The model completely stops predicting it in the no replay case. The CL models also struggle with DATA_STRUCTURE, perhaps because the final model learns a grossly inflated probability for that tag while the baseline sees the training examples in a consistently balanced fashion.

We find that the CL models suffer from subtler distribution shift errors too. For example, we see forgetting of common named entities. Episode 1 includes many instances with the APPLICATION "Android Studio," while Episode 5 only references the OPERATING_SYSTEM "Android." Thus the final CL models classify "Android" as OPERATING_SYSTEM and "Studio" as APPLICATION. More sophisticated replay techniques could address such issues by reducing distribution shift or replaying representatives for common entities/phrases.

3.5 Forgetting Over Time

Figure 2a shows how the **final** model (trained on all data) in each experiment performs on each of the train episodes with the skewed distribution. The figure shows that the CL approaches suffer from catastrophic forgetting compared to the non-CL baseline, with no replay performing worse, as expected. While the performance of the baseline model is consistent over the train episodes, the CL models' performance degrades on the earlier training episodes. While data replay helps, the gap is still large which leaves room for future work. The same plot for the temporal data splits is shown in Figure 2c. Forgetting still occurs in this case, but at a lower rate.

We also demonstrate the forgetting on the test sets in Figures 2b and 2d, where we see little impact of forgetting for the temporal setting compared to the skewed setting. The baseline's lower performance on skewed episodes 1, 2, and 3 stems from the removal of USER_INTERFACE_ELEMENT



(a) Skewed Train

(b) Skewed Test

(c) Temporal Train

(d) Temporal Test

Figure 2: Overall F1 score evaluated on each of the 5 episodes' train or test sets, for both skewed and temporal settings. All models evaluated here are trained on data from all episodes, where the CL models are trained incrementally, starting with episode 1 and finishing with episode 5.

from test episode 1 and DATA_STRUCTURE from test episodes 1, 2, and 3. The baseline can predict these entity types with relatively high accuracy, and they are fairly common. When they are removed, the baseline model loses the boost in overall F1 these types provide. Overall, we see higher forgetting when evaluating the CL approaches on train than on test, which can be explained by overfitting to the most recent episodes during training.

In the future we would like to explore hyperparameter tuning which could further reduce forgetting, and apply privacy preserving techniques such as generative replay (Sun et al., 2019). Establishing more advanced benchmarks using recent CL techniques or creating similar episodic splits for other NLP tasks would also be of interest.

4 Conclusions

We demonstrate that even in an academic dataset spanning a decade, some important characteristics of applied single-task continual learning settings, such as data shift and label imbalance, are missing. We modify and release a dataset that contains some of these realistic challenges, and we establish a data replay baseline. Although the ability to access and publish statistics for real industrial datasets is limited due to privacy and business concerns, we find that our dataset exhibits many important similarities to such datasets. Our method for producing the dataset is configurable and can be used to build different degrees of data variance to support different use cases. Although our dataset is a useful first step towards more realistic single-task continual learning, this work highlights the need for a public benchmark with truly continuous annotation.

Acknowledgements

We are grateful to Emre Barut for helpful feedback on drafts of this paper.

References

Ronald Kemker and Christopher Kanan. 2018. Fearnet: Brain-inspired model for incremental learning. In *International Conference on Learning Representations*.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. 2018. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

Vincenzo Lomonaco and Davide Maltoni. 2017. Core50: a new dataset and benchmark for continuous object recognition. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 17–26. PMLR.

Davide Maltoni and Vincenzo Lomonaco. 2019. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73.

Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. 2016. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814*.

Charith Perera, Rajiv Ranjan, Lizhe Wang, Samee U Khan, and Albert Y Zomaya. 2015. Big data privacy in the internet of things era. *IT Professional*, 17(3):32–39.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. 2020. Gdumb: A simple approach that questions our progress in continual learning. In *European conference on computer vision*, pages 524–540. Springer.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. In *Advances in neural information processing systems*, pages 2990–2999.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2019. Lamol: Language modeling for lifelong language learning. In *International Conference on Learning Representations*.

Jeniya Tabassum, Mounica Maddela, Wei Xu, and Alan Ritter. 2020. Code and named entity recognition in StackOverflow. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4913–4926, Online. Association for Computational Linguistics.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. 2019. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382.

A Comprehensive Results

We include full results for all entity types, for both the temporal data split and the skewed data split. The full results for the temporal data split are included in Table 3, and the full results for the skewed data split are included in Table 4.

B Diachronicity of Temporal and Skewed

We include some additional demonstrations of the differences between the temporal and skewed settings. In Table 5, we show the top five entity types for all episodes' train and test for both settings. Although there is some variation across episodes for the temporal setting, the variation is stronger for the skewed setting.

We demonstrate a few examples of the CODE_BLOCK, DATA_STRUCTURE, and USER_INTERFACE_ELEMENT types in Table 6. Recall that in the skewed data, we remove the CODE_BLOCK entity in episode 3 and onward, add the DATA_STRUCTURE entity only in episodes 4 and 5, and remove the USER_INTERFACE_ELEMENT entity from episode 1. This behavior impacts the top five entities, as Table 5 makes apparent.

Entity Type	Baseline (non-CL)	CL w/o Replay	CL w/ Real Replay	Avg. Count
Overall	51.36	51.52	51.12	777.60
Algorithm	24.00	19.64	21.82	0.00
Application	57.94	57.76	58.36	2.80
ClassName	25.38	18.89	18.33	80.00
CodeBlock	25.67	28.59	26.41	25.80
DataStructure	75.27	73.76	72.82	59.80
DataType	67.52	70.24	70.81	48.00
Device	59.38	60.24	58.99	21.60
ErrorName	3.64	3.64	14.16	10.60
FileName	62.31	64.06	60.01	3.60
FileType	69.82	66.28	77.19	32.60
FunctionName	12.25	4.86	9.71	21.60
HTMLXMLTag	42.32	41.51	40.96	9.20
KeyboardIP	1.74	9.78	1.67	10.40
Language	75.41	74.09	70.75	7.00
Library	53.97	53.28	47.46	35.40
LibraryClass	47.55	48.00	47.06	50.20
LibraryFunction	44.69	48.43	47.20	72.80
LibraryVariable	18.58	10.79	21.57	43.00
License	0.00	0.00	0.00	21.80
OperatingSystem	82.46	79.15	82.85	0.00
Organization	10.00	53.33	43.33	12.20
OutputBlock	75.20	68.71	67.14	1.80
UserInterfaceElement	56.43	56.96	56.67	10.80
UserName	35.83	35.69	32.21	69.40
Value	45.68	44.88	34.68	4.60
VariableName	28.44	28.81	27.53	43.00
Version	72.05	72.26	72.74	53.00
Website	25.99	22.29	28.00	21.20

Table 3: F1 scores by type after training incrementally over all 5 temporal episodes vs on all training data at once. Scores are averaged over all 5 episodes' test sets. We also denote the average count of each entity type in all 5 test episodes.

Entity Type	Baseline (non-CL)	CL w/o Replay	CL w/ Real Replay	GDumb (1500)	Avg. Count
Overall	52.24	42.61	49.82	40.19 ± 0.67	750.40
Algorithm	10.00	14.44	28.33	32.43 ± 4.17	0.00
Application	55.93	53.01	55.68	47.23 ± 1.33	3.20
ClassName	19.84	5.24	8.21	21.17 ± 1.94	75.40
CodeBlock	12.51	0.00	7.74	8.82 ± 0.54	25.60
DataStructure	32.03	32.60	33.82	27.46 ± 1.52	47.20
DataType	72.45	67.24	68.77	63.53 ± 3.89	45.60
Device	53.32	47.39	47.21	45.28 ± 5.18	21.20
ErrorName	0.00	0.00	10.00	4.39 ± 2.61	10.60
FileName	54.79	6.17	46.81	39.09 ± 3.88	3.60
FileType	55.95	38.51	55.10	43.37 ± 5.23	32.60
FunctionName	26.16	5.34	8.58	11.23 ± 2.03	25.80
HTMLXMLTag	40.25	27.98	41.61	33.55 ± 3.70	9.20
KeyboardIP	8.00	5.71	13.33	8.31 ± 3.39	10.40
Language	69.11	67.59	67.83	57.26 ± 1.21	7.00
Library	55.35	48.26	55.43	40.70 ± 2.57	35.60
LibraryClass	48.52	42.13	45.70	36.97 ± 2.20	51.40
LibraryFunction	44.95	34.87	44.49	32.91 ± 3.64	75.40
LibraryVariable	23.33	7.99	3.12	6.21 ± 1.99	41.40
License	0.00	0.00	0.00	0.00 ± 0.00	22.40
OperatingSystem	79.74	60.45	65.81	64.81 ± 3.93	0.00
Organization	13.33	20.00	20.00	21.76 ± 4.62	13.20
OutputBlock	63.07	0.00	63.78	59.16 ± 6.16	2.00
UserInterfaceElement	44.94	40.81	43.25	34.68 ± 2.02	10.60
UserName	30.73	39.63	36.00	28.48 ± 4.81	54.00
Value	56.27	45.75	46.19	38.92 ± 2.61	4.60
VariableName	25.87	26.05	26.35	18.83 ± 3.00	42.80
Version	77.89	70.54	77.41	73.98 ± 3.18	51.60
Website	36.66	27.81	49.53	35.05 ± 4.63	22.20

Table 4: F1 scores by type after training incrementally over all 5 skewed episodes vs on all training data at once. Scores are averaged over all 5 episodes' test sets. We also include results for GDumb with memory budget of 1500 examples, averaged over 10 random initializations. We also denote the average count of each entity type in all 5 test episodes.

	Ep. 1	Ep. 2	Ep. 3	Ep. 4	Ep. 5
Temporal Train	Application	LibraryClass	Application	CodeBlock	Application
	LibraryClass	Application	LibraryClass	LibraryClass	UserInterfaceElem.
	UserInterfaceElem.	UserInterfaceElem.	UserInterfaceElem.	Application	LibraryClass
	Library	VariableName	VariableName	VariableName	Library
	CodeBlock	Value	Value	Value	CodeBlock
Temporal Test	UserInterfaceElem.	UserInterfaceElem.	LibraryClass	LibraryClass	Application
	Application	LibraryClass	Value	Application	CodeBlock
	LibraryClass	Application	CodeBlock	Library	VariableName
	VariableName	LibraryFunction	VariableName	CodeBlock	LibraryFunction
	Library	LibraryVariable	DataStructure	UserInterfaceElem.	Library
Skewed Train	CodeBlock	LibraryClass	UserInterfaceElem.	Value	FileType
	Application	Library	LibraryFunction	DataStructure	DataStructure
	Library	Language	Language	Application	VariableName
	LibraryClass	Application	VariableName	FileName	Application
	FileName	UserInterfaceElem.	ClassName	VariableName	OperatingSystem
Skewed Test	CodeBlock	UserInterfaceElem.	VariableName	DataStructure	LibraryClass
	Value	Language	UserInterfaceElem.	Application	DataStructure
	Application	CodeBlock	Application	LibraryClass	VariableName
	Library	Application	ClassName	LibraryFunction	Library
	LibraryClass	FileType	LibraryClass	UserInterfaceElem.	FileName

Table 5: Top five entity types (in order) for each episode of temporal/skewed train/test splits.

CODE_BLOCK	Instead, start a **command prompt (Application)** and " **cd (Code_Block)** " to where your **jar (File_Type)** file is. Add **rm -r (Code_Block)** to remove the file hierarchy rooted in each file argument. **rm /path/to/directory/ * (Code_Block)**
DATA_STRUCTURE	Allocate an **array (Data_Structure)** of **pointers (Data_Type)** to **chars (Data_Type)** where **keywords (Variable_Name)** is the **list (Data_Structure)** of **strings (Data_Type)** so we can parse and find the correct item, and **session (Variable_Name)** is the a new **session (Library_Class)** from the **requests (Library)** module. I need to get the 14 days average **Col 1 (Variable_Name)** and update **Col 2 (Variable_Name)** of the same **table (Data_Structure)**.
USER_INTERFACE_ELEMENT	There will be a class method, which opens a new **tab (User_Interface_Element)**, renders some **HTML (Language)**, and returns the **PDF (File_Type)** data, and closes the **tab (User_Interface_Element)**. I'm trying to create a responsive effect, where I hide a **column (User_Interface_Element)** when my **screen (User_Interface_Element)** is **960 (Value)** or lower. But in **iOS (Operating_System) 10 (Version)**, **photos (User_Interface_Element)** not appearing until I tap on **cell (User_Interface_Element)** that holds **collection view (Library_Class)**.

Table 6: Examples containing the CODE_BLOCK, DATA_STRUCTURE, and USER_INTERFACE_ELEMENT types. We remove all examples with these types in different episodes to simulate class incrementality in the skewed dataset. All entities are bolded with the entity type in parentheses following the entity.