2024

Large Language Models, Prompting, and Synthetic Data Generation for Continual Named Entity Recognition

Charles I. Cutler
Virginia Commonwealth University

LARGE LANGUAGE MODELS, PROMPTING, AND SYNTHETIC DATA GENERATION FOR CONTINUAL NAMED ENTITY RECOGNITION

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

by

CHARLES CUTLER

Bachelor's of Computer Science - August 2020 to May 2023

Director: Dr. Bridget McInnes,

Professor, Department of Computer Science

Virginia Commonwealth University

Richmond, Virginia

May 2024

Acknowledgements

I extend my deepest gratitude to Dr. Bridget McInnes, whose unwavering support, guidance, and passion for machine learning and natural language processing have been the cornerstone of my academic journey. Your mentorship has inspired me to delve deeper into these fields and to never stop asking questions.

To Scott, whose exceptional dedication and remarkable work ethic throughout this research endeavor made it possible to finish on time. Scott, you have consistently gone above and beyond to contribute to our work. Thank you for being the best research partner ever.

To Dr. Christina Tang, Dr. Afroditi Filippas, Dr. Preetam Ghosh, and Dr. Tomasz Arodz, for serving as members of my committee and providing me with invaluable feedback throughout the process.

To my beloved parents for their boundless love, encouragement, and sacrifices. Your belief in me has kept me going and always kept me striving for excellence. I am forever grateful.

To my cherished partner, Sanika, whose patience and understanding have been my rock during moments of frustration and stress. Your steadfast presence and unconditional love have provided me with the strength and comfort needed to overcome challenges and persevere through the toughest of times.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract

LARGE LANGUAGE MODELS, PROMPTING, AND SYNTHETIC DATA GENERATION FOR CONTINUAL NAMED ENTITY RECOGNITION

By Charles Cutler

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

Virginia Commonwealth University, 2024.

Director: Dr. Bridget McInnes,

Professor, Department of Computer Science

With the ever-growing amount of textual data, the task of Named Entity Recognition (NER) is vital to Natural Language Processing (NLP), a field which focuses on enabling computers to understand and manipulate human language. NER enables the extraction of information from unstructured text. Accurate information extraction is crucial for applications ranging from information retrieval to systems for question-answering. To ensure that NER models are robust to changes in data distributions and capable of recognizing new entity types, one may consider expanding the capabilities of an existing model.

Continual learning is a paradigm within machine learning. It studies the objective of learning new information incrementally without forgetting previously learned knowledge. A central concern with continual learning is the phenomenon of catastrophic forgetting, where training a neural network on new information leads to significant degradation in performance on previously learned information. Reannotation of existing data for new information and then training a new model proves costly

and time-consuming, prompting the need for better strategies. Generating and using synthetic data to combat forgetting has been studied in continual learning for vision models and, to a limited extent, with long short-term memory unit (LSTM) generators or inverted models for NER models. One way to achieve this is to use generative large-language models to create synthetic data.

This work focuses on building the foundation for a generative replay approach. We aim to determine the efficacy of using Open AI's GPT-4 model to generate synthetic data to supplement the training of NER systems. We aim to answer the following questions: Can synthetic data be generated to mimic the format of authentic NER training data? Is synthetic data similar to the authentic data? Does the addition of synthetic data improve model performance? Is solely using synthetic data enough to achieve performance on par with a baseline? How do different prompting strategies for generating synthetic data affect model performance?

We conducted experiments using the 2018 TAC SRIE dataset and a DeBERTa-V3-based model with broadcast linear and softmax classification layers. We successfully generated synthetic data using GPT-4 and two different prompting strategies. We found improved performance when supplementing authentic data with synthetic data, even when only supplementing with small amounts.

This work contributes a novel finding concerning NER and synthetic data generation with generative large-language models and lays the foundation for a novel generative-replay approach to continual NER.

CHAPTER 1

INTRODUCTION

With the ever-expanding amount of digital textual data, the task of Named Entity Recognition (NER) is vital to Natural Language Processing (NLP), a field which focuses on enabling computers to understand and manipulate human language. NER enables the extraction of information from unstructured text and involves identifying and classifying named entities, such as names of people, organizations, locations, and dates [1]. Accurately recognizing these named entities facilitates information extraction and plays a crucial role in applications ranging from information retrieval to systems for question-answering [2]. To ensure that NER models are robust to changes in data distributions and capable of recognizing new entity types, one may consider expanding the entity recognition capability of an existing model.

The traditional approach of reannotating an existing dataset for new entities, restructuring, and training a new model for an expanded entity set is expensive and time-consuming [3]. Attempts to comprehensively annotate an entirely new dataset are challenging and time consuming, especially when the number of different entity types grows. They are also prone to human error even when they are conducted by a team of disciplined experts. Self-training, where an older model annotates the dataset for previous entity types, can propagate errors downstream [4]. The conventional method of annotating data solely for new entity types introduces the risk of catastrophic forgetting [5]. Moreover, storage limitations or privacy concerns may hinder access to the original training material [6]. Thus, *continual learning* approaches become practical to train models in recognizing new entity types without forgetting

the types learned previously.

Continual learning is a paradigm within machine learning. It studies the objective of learning new information incrementally without forgetting previously learned knowledge [3]. In continual learning, the tendency of models to forget is called catastrophic forgetting [7], where training a neural network exclusively on new information leads to the severe degradation in performance on previously learned information. In the realm of NER, continual learning refers to the capacity of a model to adjust and enhance its performance when faced with novel entities, languages, and domains. For example, a voice assistant like Siri may benefit from recognizing a new entity type, such as "movie titles," to allow users to order movie tickets. However, training a new version of Siri from scratch to add this feature may not be feasible; therefore, we may want to expand Siri's knowledge incrementally instead [4].

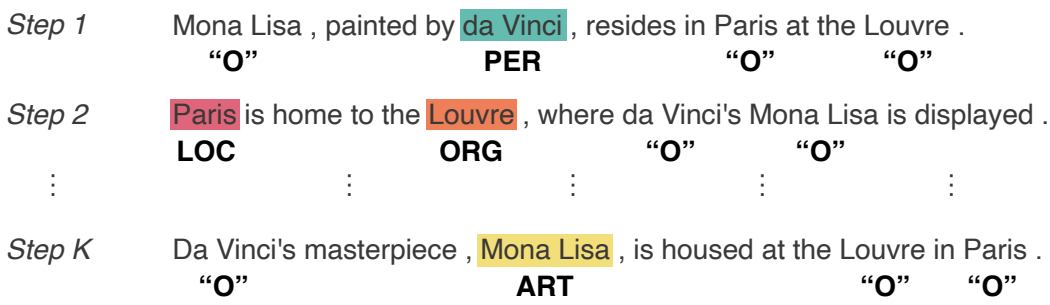

Fig. 1. Each sentence contains the same four entities: "Mona Lisa," "da Vinci," "Paris," and "Louvre." The figure shows an example of how annotators might label these entities differently, demonstrating the "Unlabeled Entity Problem." In each step, data is only annotated for a finite set of entity types. Thus, the non-entity type "O" can contain entities from old and future entity types. (Consider words without an explicit label in the figure as "O".)

Inherent to continual learning for NER is another challenge that exacerbates the issue of catastrophic forgetting. This is called the *Unlabeled Entity Problem* [8]

sometimes referred to as *Semantic Shift* [9]. Researchers typically annotate new training data for a finite set of entity types. Unfortunately, this treatment assigns entities outside this set to the non-entity type "O." For example, in the second step in Figure 1, the sample sentence is only annotated for the **LOC** and **ORG** entity types. Because of this, annotators labeled entities that could have been in the **ART** and **PER** entity types as "O." Unlabeled entities fit into two groups: 1) previously learned entity types labeled in earlier steps or 2) new entity types that someone might label in future steps. In Figure 1, there are three different sentences that each contain the same four entities: "Mona Lisa," "da Vinci," "Paris," and "Louvre." Figure 1 shows how annotators might label these entities differently depending on the chosen entity types in a specific continual learning step, each with a different setup.

This work aims to determine the efficacy of using generative large-language models to create synthetic data to supplement the training of NER systems. This work first provides an overview of current continual learning methods focusing on NER. This includes context from previous work in reviewing the area of continual learning. Evaluation metrics for continual learning are presented and a comparison and evaluation of continual learning methods specific to NER is conducted. The evolution of continual learning is explored, categorizing methods into Replay, Regularization, and Parameter Isolation approaches. The research goals of this work are to to answer the following questions:

1. Can synthetic data be generated to mimic the format of authentic NER training data?

2. Is synthetic data similar to the authentic data?

3. Does the addition of synthetic data improve model performance?

4. Is solely using synthetic data enough to achieve performance on par with a baseline?

5. How do different prompting strategies for generating synthetic data affect model performance?

We conducted experiments using the 2018 TAC SRIE dataset and a DeBERTa-V3-based model with broadcast linear and softmax classification layers. We successfully generated synthetic data using GPT-4 and two different prompting strategies. This work contributes a novel finding concerning NER and synthetic data generation with generative large-language models and lays the foundation for a novel generative-replay approach to continual NER.

CHAPTER 2

LITERATURE REVIEW

2.1 Previous Review Work of Continual Learning

Continual learning methods are often applied to computer vision tasks due to the dynamic and evolving nature of visual data. In computer vision, new images, objects, or scenarios may be encountered over time, requiring models to adapt and learn from new information without forgetting previously acquired knowledge. Continual learning techniques enable computer vision models to incrementally update their knowledge and accommodate changes in data distributions, ensuring robust performance in real-world applications. Surveys of the existing approaches in continual learning often focus on computer vision tasks because of the pressing need for adaptive and flexible vision systems that can handle diverse and evolving visual inputs [3, 10, 11, 12].

De Lange et al. [3] distinguish the three categories of continual learning approaches that we discuss in this paper. Their survey is largely regularization-focused and conducts an empirical study that measures the performance of existing continual learning methods. Lesort et al. [10] focus on robotics applications. The paper provides a four categories of continual learning approaches, contributing an in-depth examination of the combined categories models fall into. Parisi et al. [11] provide an overview of existing continual learning approaches, discussing their connections with lifelong learning in biological systems.

The recent comprehensive continual learning survey from Wang et al. [13] catalogs NLP models based on both the continual learning category and NLP task, touch-

ing on NER models. Biesialska, Biesialska, and Costa-jussà [14] and Ke and Liu [15] focus their surveys entirely on NLP tasks. Biesialska, Biesialska, and Costa-jussà [14] review both task incremental and class incremental continual learning methods and provide an overview of NLP-specific continual learning datasets and benchmarks. Ke and Liu [15] add emphasis on knowledge transfer and the problem of inter-task class separation for task incremental learning methods. To our knowledge, our survey is the only one that focuses on continual learning methods for named entity recognition.

2.2 A Brief History, Categories, and Evaluation of Continual Learning Approaches

The origins of continual learning can be traced back to the seminal work of McCloskey and Cohen [7]. They introduced the concept of catastrophic forgetting, highlighting that learning new tasks may disrupt previously learned ones when sequentially training neural networks. Their analysis revealed that interference often occurs when a network modifies the weights associated with representing a previously learned task. This section aims to offer a selective overview of the history of continual learning, as depicted in Figure 2 , emphasizing key developments and delineating the three primary categories of continual learning approaches.

2.2.1 Continual Learning Approaches

In this section, we describe three categories of continual learning methods: Replay, Regularization, and Parameter Isolation.

2.2.1.1 Replay

Replay approaches involve preserving training data for reuse in subsequent training steps, typically by selecting a small set of representative samples and reintroduc-

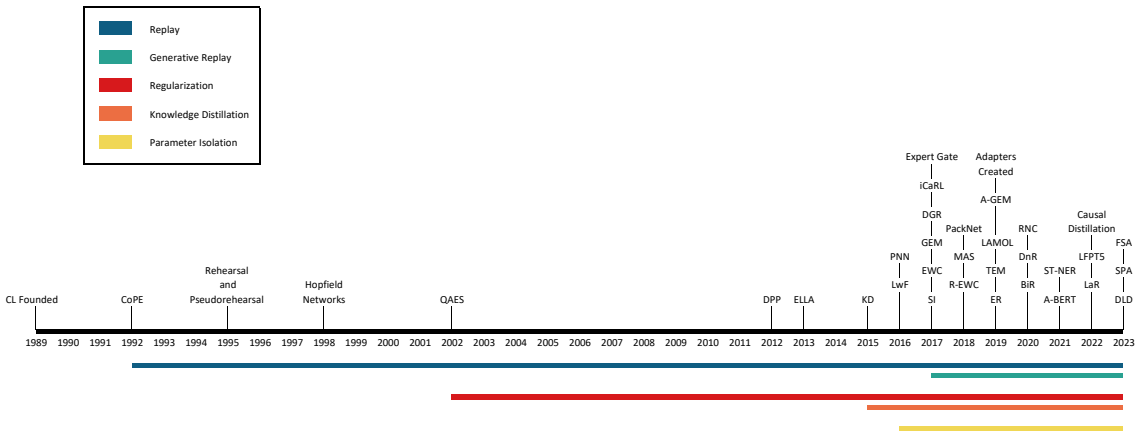

Fig. 2. Timeline of the development of continual learning. Notable publications are shown above the timeline at the date of first publication. Colored bars represent publications for different approach categories starting over time. Generative replay is a sub-category of replay. Knowledge distillation is a sub-category of regularization. (For interpretation of the publication acronyms, see Appendix A.)

ing them during future training iterations. These techniques were among the earliest strategies used in continual learning. For example, seminal works by French [16] and Robins [17] introduced replay to mitigate catastrophic forgetting by leveraging training buffers to reintroduce previously encountered examples during ongoing training. Robins [17] also introduced the concept of pseudo-rehearsal, a form of replay that employs artificially generated pseudoitems instead of actual previously seen data. These pseudoitems are constructed by generating new random input vectors and processing them through the network in a standard manner. Additionally, Robins and McCallum [18] extended the notion of pseudo-rehearsal to Hopfield-type networks, further decreasing catastrophic forgetting. Despite major differences between Hopfield-type networks and multi-layer-perceptrons, they determined that the same general techniques for maintaining previously learned information are applicable, only requiring adaptation of the specific mechanism employed to generate pseudoitems.

Rebuffi et al. [19] developed Incremental Classifier and Representation Learning (iCaRL), a method where a neural network learns classifiers and feature representations by incorporating exemplar selection and prototype rehearsal. Lopez-Paz and Ranzato [20] introduced Gradient Episodic Memory (GEM), which utilizes an episodic memory with a size budget to store a subset of examples from each task encountered during continual training, aiming to prevent increases in loss on previous tasks. Chaudhry et al. [21] improved upon GEM with the creation of Averaged Gradient Episodic Memory (A-GEM), which focuses on preventing increases in the average episodic memory loss over the previous learning tasks. Additionally, Chaudhry et al. [21] introduced Tiny Episodic Memory (TEM) to analyze the effects of a minimal episodic memory setup, where the model encounters previous training samples only once.

Notably, generative replay began with Shin et al. [22] and formed a distinctive sub-category within replay approaches. These methods involve saving or utilizing models capable of generating training data rather than directly saving training data. Shin et al. [22] developed a method where a model creates synthetic data from past input distributions called a "scholar," consisting of a generator and a solver. Rostami, Kolouri, and Pilly [23] created an abstract generative distribution that allows for the creation of data points to represent previous experiences. LAnguage MOdelling for Lifelong Language Learning (LAMOL), developed by Sun, Ho, and Lee [24], learns to generate pseudo-samples and requires no additional memory space. Subsequently, Sun et al. [25] developed Distill and Replay (DnR), which utilizes generative experience replay to imitate previous data distributions. van de Ven, Siegelmann, and Tolias [26] introduced a variant of generative replay, "brain-inspired replay," that uses modifications inspired by the human brain to improve model performance. Modifications include Replay-Through-Feedback, which merges the generator into

the primary model, equipping it with generative backward or feedback connections, and Conditional-Replay, which allows the model to generate samples of only specific classes. Notable replay methods for NER, discussed later in Section 2.3.1 include those proposed by Ma et al. [8] and Xia et al. [27].

Replay approaches in continual learning face notable challenges, particularly related to privacy concerns and data storage issues. For example, storing and reusing large volumes of training data can raise privacy concerns, especially when dealing with sensitive information. It can additionally lead to significant data storage requirements, posing practical challenges, particularly in resource-constrained environments. As a response to these challenges, there has been a significant shift in research toward the development of replay-free approaches or strategies that combine replay methods with additional techniques, such as regularization methods. By incorporating regularization techniques, which modify specific sets of weights during training without nullifying their updates entirely, researchers aim to mitigate the privacy and storage challenges associated with replay approaches. Regularization methods, as described below, offer a complementary approach to replay techniques. By incorporating regularization into the learning process, models can retain previously learned knowledge while adapting to new tasks without relying heavily on storing and replaying large datasets. These methods introduce constraints or penalties to the training process, encouraging the model to maintain stability and prevent catastrophic forgetting without the need for extensive replay. By combining replay-free approaches with regularization methods, researchers aim to strike a balance between preserving privacy, managing data storage requirements, and ensuring continual learning performance. This integrated approach opens avenues for addressing the challenges posed by traditional replay methods, paving the way for more efficient and privacy-preserving continual learning systems.

2.2.1.2 Regularization

Regularization methods modify specific sets of weights during training without nullifying their updates entirely. These methods attempt to mitigate catastrophic forgetting and ensure the retention of previously learned knowledge while adapting to new tasks. This is often achieved by incorporating a term into the objective function or identifying crucial weights and penalizing changes to them.

An early example of regularization in continual learning is presented by French and Chater [28], who introduced a method to mitigate catastrophic forgetting by utilizing noise to approximate the error surface associated with previously learned information. They combined this approximation with the error surface related to new information to retain previously learned tasks. Ruvolo and Eaton [29] proposed an algorithm that maintains a shared basis and enables soft task grouping, facilitating knowledge transfer to new tasks. This approach enhances the model's ability to generalize across tasks and retain knowledge more effectively. Zenke, Poole, and Ganguli [30] introduced intelligent synapses, where each "synapse" in the neural network retains its measure of importance in solving previously learned tasks. This allows the model to prioritize important connections and penalize changes to them when learning new tasks. Lee et al. [31] proposed Incremental Moment Matching, a method that penalizes weight changes during training to prevent catastrophic forgetting. This approach helps stabilize the model's performance across tasks and maintain consistency in learning. Kirkpatrick et al. [32] developed Elastic Weight Consolidation (EWC), which slows down the learning of certain parameter weights based on their importance to previously seen tasks through a soft quadratic constraint. EWC effectively balances the trade-off between learning new tasks and retaining old knowledge. Liu et al. [33] improved EWC by using rotational matrices to achieve a better reparam-

eterization and enhance the diagonal assumption of the Fisher Information matrix. This enhancement further strengthens the model's ability to retain important knowledge while adapting to new tasks. Aljundi et al. [34] proposed a different method to maintain an importance measure for each network parameter, penalizing changes to significant parameters to prevent the overwrite of knowledge essential for previous tasks. This method enhances the model's ability to retain important information and adapt to new tasks while minimizing catastrophic forgetting.

Knowledge distillation approaches, which originated with the work of Hinton, Vinyals, and Dean [35], form a distinctive sub-category within regularization approaches. These approaches aim to transfer knowledge from a larger, more complex model (referred to as the teacher model) to a smaller, simpler model (referred to as the student model) through a process known as distillation. Hinton, Vinyals, and Dean [35] employed a distillation loss to train a student model using the soft label output of a parent model for model consolidation. Unlike traditional self-training with hard labels, distillation provides more information about how the teacher makes predictions, leading to more effective knowledge transfer. Li and Hoiem [6] introduced Learning without Forgetting (LwF), the first method to use Knowledge Distillation specifically for continual learning. LwF addresses the challenge of catastrophic forgetting by leveraging distillation loss to preserve knowledge of previous tasks while learning new ones, thereby improving continual learning performance. Notable examples of regularization approaches for NER include the works of Monaikul et al. [4], Zheng et al. [36], and Zhang et al. [37]. These methods, which incorporate knowledge distillation techniques, are discussed in detail in Section 2.3.2.

2.2.1.3 Parameter Isolation

Parameter Isolation approaches involve dynamically modifying the architecture of a model to prevent catastrophic forgetting by selectively preserving knowledge while learning new tasks. This is achieved through techniques such as freezing weights, adding, cloning, or saving parts of a model to avoid forgetting. The evolution of parameter isolation approaches has seen significant developments in recent years, addressing various challenges in continual learning.

One of the pioneering works in this field is Progressive Neural Networks (PNNs) introduced by Rusu et al. [38], which adds a new neural network for each new task, each with connections to previously learned networks. Aljundi, Chakravarty, and Tuytelaars [39] focus on selecting the most relevant previous task when training a new expert model. By identifying and transferring knowledge from related tasks, this approach leverages past experiences to facilitate learning on new tasks. Pathnet, proposed by Fernando et al. [40], takes a different approach by modifying the forward pass path within the network. It identifies critical paths for each learning task and updates their weights while keeping others unchanged, enabling targeted learning without interference. Mallya and Lazebnik [41] introduce PackNet, which employs iterative pruning and retraining to identify and "free up" parameters for learning new tasks. By efficiently packing multiple tasks into the same neural network, PackNet optimizes parameter utilization and minimizes interference between tasks. Hard Attention to the Task (HAT), developed by Serra et al. [42], trains models by identifying task-relevant weights and then creating masks to prevent changes to these weights during subsequent training. This technique, also known as training with hard attention, ensures that crucial information for each task is preserved throughout the learning process. Collier et al. [43] propose sparse routing networks, which utilize

co-training techniques to activate different paths through a network of experts. By avoiding poorly initialized experts, sparse routing networks enhance the robustness and efficiency of parameter isolation in continual learning scenarios.

In 2019, Houlsby et al. [44] introduced a novel method known as adapter-based tuning, representing a significant shift in parameter isolation approaches. This method offers a modular approach to isolating parameters for continual learning tasks. The primary motivation behind adapter-based tuning was to address the challenges associated with fine-tuning deep neural networks for multiple downstream tasks. Traditional approaches often result in a significant increase in the number of model parameters leading to computational inefficiency. Adapter-based tuning addresses this issue by requiring architectural modifications to pretrained networks. Specifically, a small number of new parameters are added to the model through the introduction of new layers known as adapter layers. These adapter layers are injected into the pretrained network without altering the weights of the original network. Instead, the weights of the adapter layers are randomly initialized and trained. This approach allows for the efficient tuning of pretrained models for specific tasks without the need for extensive parameter modifications.

Several approaches utilize adapter modules or similarly inspired techniques for continual learning. For example, Ke, Xu, and Liu [45] propose BERT-based Continual Learning (B-CL) using a Continual Learning Adapter (CLA) inspired by Adapter-BERT introduced by Houlsby et al. [44]. To mitigate catastrophic forgetting, CLA utilizes task masks to safeguard task-specific knowledge. Additionally, it employs capsules and dynamic routing mechanisms to identify and transfer shared knowledge from previous tasks to the current task. Another approach, developed by Ke et al. [46] is a Continual and contrastive Learning for ASpect SentIment Classification (CLASSIC). CLASSIC employs adapters to transfer and distill knowledge from previous

tasks to the model for the current task, enabling it to perform all past and current tasks effectively. It also learns task masks to isolate task-specific knowledge, thus preventing catastrophic forgetting. Additionally, Ke et al. [47] introduced Capsules and Transfer Routing for continual learning (CTR). CTR utilizes a CL-plugin, similar to an adapter, with a key distinction being that it learns all tasks using only one pair of CL-plugin modules inserted into Bidirectional Encoder Representations from Transformers (BERT) [48]

Panos et al. [49] introduce First Session Adaptation (FSA), a novel approach to continual learning. In this method, a neural network is trained solely during the initial continual learning step. Subsequently, the network is frozen, and Feature-wise Layer Modulation adapters are employed to expand the model's knowledge. Furthermore, notable parameter isolation approaches for NER include the works of Qin and Joty [50] and Zhang et al. [51]. These methods are discussed further in Section 2.3.3, focusing on parameter isolation techniques for NER.

2.2.2 Formal Definition of Continual Learning

Continual learning can be expressed in different forms. This section describes two popular forms: task-incremental learning (TIL) and class-incremental learning (CIL). Wang et al. [13] explore more recent forms, such as domain-incremental learning, and we direct the reader there for additional information.

2.2.2.1 Task-Incremental Learning

With task-incremental learning, the model must incrementally learn to complete an increasing number of distinct tasks. For example, first, perform sentiment analysis, then perform named entity recognition, then word sense disambiguation, and then relationship extraction. Tasks can also be different classification objectives from the

same overall task. Take, for instance, the task of NER. First, the model is asked to classify entities as persons or "O." Then the model is asked to classify images as locations or "O." Naturally, it would seem the model can differentiate between persons, locations, and "O." Instead, the model has learned only two specific tasks: entity recognition of persons vs. "O" **or** entity recognition of locations vs. "O."

Formally, in the TIL form of continual learning, a model is expected to perform only the exact tasks it is trained on. Consider a set of tasks, $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$, where each task $t_i \in \mathcal{T}$ for $i = 1, \ldots, m$, is distinct, the model will be trained to learn. Each continual learning training step is a task $t_i \in T$, and a training sequence is simply an ordering of the set of tasks $\mathcal{T}$, i.e., $[t_1, t_2, \ldots, t_N]$, where t_1 comes before t_2, t_2 comes before t_3, etc. After a given training step $n \leq N$, the model should be able to perform all tasks mastered up to that point, e.g., perform all tasks $[t_1, t_2, \ldots, t_n]$

2.2.2.2 Class-Incremental Learning

With class-incremental learning, we are training the model to complete the same overall task throughout the entire continual learning process, i.e., named entity recognition. Instead, the model must incrementally learn to differentiate between an increasing number of distinct classes with each continual learning training step.

Consider a set of classes we desire the model to learn, $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$, where each class $c_i \in \mathcal{C}$ for $i = 1, \ldots, m$, is distinct. A continual learning training step $\mathcal{S}$ is a set of classes, $\mathcal{S} \subseteq \mathcal{C}$, and a training sequence is simply an ordered set of training steps, $[\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_N]$, where $\mathcal{S}_1$ comes before $\mathcal{S}_2$, $\mathcal{S}_2$ comes before $\mathcal{S}_3$, etc.

The first training step $\mathcal{S}_1$ must contain a discrete set of classes from $\mathcal{C}$. Each subsequent training step $[\mathcal{S}_2, \ldots, \mathcal{S}_N]$ must, at minimum, introduce one class from $\mathcal{C}$ not seen in any of the training steps that come before it. For example, $\mathcal{S}_4$ must introduce at least one class from $\mathcal{C}$ not already in $\{\mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3\}$. Outside of this

restriction, overlapping classes with other steps is acceptable. After a given training step $n \leq N$, the model should be able to recognize all classes seen up to that point, e.g., recognize $\{\mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_n\}$

2.2.3 Evaluation Metrics

In addition to standard task evaluation for continual learning, it is practical to use metrics that track a continual learning model's lost performance on previously learned tasks given the problem of catastrophic forgetting. The most popular metrics are Average Accuracy [20] and Forgetting Measure [52]. Average Accuracy, defined in Equation 2.1, takes the average of the model's accuracy at each step where T is the number of tasks the model has trained on and $R_{T,i}$ is the model accuracy on task i after training on all T tasks.

$$\text{Average Accuracy} = \frac{1}{T}\sum_{i=1}^{T} R_{T,i} \tag{2.1}$$

Forgetting Measure represents the error rate over time. It is a summation of the differences between the final error rates on classes known to the model and the error rates on those classes when they were initially learned by the model. A high forgetting measure indicates the occurrence of catastrophic forgetting. This is defined in Equation 2.2. Where ER is the error rate, $ER_{i,i}$ is the error rate on task i immediately after learning task i, $ER_{T,i}$ is the error rate after training on all T tasks. For each task i, the forgetting measure identifies the subsequent task, which results in the maximum increased error rate, averaging these increases across all tasks.

$$\text{Forgetting Measure} = \frac{1}{T-1}\sum_{i=1}^{T-1} \max_{t \in \{1,\ldots,T-1\}} (ER_{i,i} - ER_{T,i}) \tag{2.2}$$

Other evaluation metrics include Learning Curve Area [21] and the related on-

line codelength [53], average incremental accuracy [19], along with three metrics introduced in Kemker et al. [54]. For further details, we direct readers to the original papers.

The stability-plasticity dilemma is a constraint related to the problem of catastrophic forgetting [55]. Model plasticity refers to a model's ability to incorporate new knowledge, while model stability refers to its ability to retain old knowledge (avoid forgetting). Balancing these properties can improve the performance of continual learning models, and some metrics have been adopted to quantify them.

In addition to the previously mentioned forgetting measure, stability can be measured by backward transfer[20] Backward transfer measures the effect of learning a new task on a model's performance on a previous task, and is typically negative. This is defined in Equation 2.3, where T is the number of tasks the model has trained on, $R_{T,i}$ is the model accuracy on task i after training on all T tasks, and $R_{i,i}$ is the model's classification accuracy on task i immediately after training on task $i-1$ Backward transfer averages over $T-1$ steps and the summation goes only to $T-1$, as Backward transfer cannot be used to measure model stability on the first continual learning step.

$$\text{Backward Transfer} = \frac{1}{T-1}\sum_{i=1}^{T-1}(R_{T,i} - R_{i,i}) \tag{2.3}$$

Model plasticity can be measured by forward transfer [20] and intransigence measure [52]. Forward transfer measures whether the model has useful knowledge for the next continual learning step by comparing its performance to a randomly initialized model. This is defined in Equation 2.4 where $R_{i-1,i}$ is the model's classification accuracy on task i when previously trained on task $i-1$, and $\bar{b}_i$ is the average test accuracy for task i when the model is randomly initialized. Conversely to backward

transfer, forward transfer averages over $T-1$ and begins the summation with $i=2$, as it cannot be used to measure plasticity on the final continual learning step.

$$\text{Forward Transfer} = \frac{1}{T-1}\sum_{i=2}^{T}(R_{i-1,i} - \bar{b}_i) \tag{2.4}$$

Intransigence measure quantifies a model's inability to learn new tasks. It compares the model against a reference model, which is trained using all data from the first task to task i simultaneously. This is defined in Equation 2.5 where R_i^* is the reference model's accuracy on the test set of task i when trained up to task i simultaneously, and $R_{i,i}$ is the continual model's accuracy on task i when trained up to task i incrementally.

$$\text{Intransigence}_i = R_i^* - R_{i,i} \tag{2.5}$$

2.3 Continual Learning Methods for Named Entity Recognition

In this section, we present methods tailored specifically for continual learning in the context of NER, highlighting their unique contributions with the three primary categories introduced in Section 2.2.1: Replay, Regularization, and Parameter Isolation. Table 1 shows a comprehensive overview of the NER-specific methods, categorized according to their respective approaches Notably, some methods employ a combination of continual learning strategies, as indicated by multiple check marks in the table We aim to provide a detailed analysis of each method's strengths and limitations, shedding light on their effectiveness in addressing the challenges of continual learning for NER.

Table 1. NER Methods and their Continual Learning Approaches

References	Regularization	Replay	Parameter Isolation
AddNER and ExtendNER [4]	✓		
Attention Based Sequence-to-Sequence [56]		✓	
Causal Framework for Continual NER [36]	✓		
Continual Learning with NERDA [57]	✓		
CPFD [9]	✓		
Decomposing Logits Distillation [37]	✓		
Few-Shot Class-Incremental Learning [58]		✓	
JCBIE [59]		✓	
Language Model Augmented Learning [60]		✓	
Learn & Review [27]		✓	
Learning "O" helps for Learning More [8]		✓	
LFTP5 [50]	✓	✓	✓
Prototype-based NER [61]	✓	✓	
Relation Distillation and Prototyping [62]	✓		
Single Epoch Recovery [63]		✓	
SKD-NER [64]	✓		
SpanKL [65]	✓		
Sub-networks and Task similarity [66]			✓

2.3.1 Replay

In this section, we describe previous work that has developed replay methods for continual learning for NER.

2.3.1.1 Attention Based Sequence-to-Sequence

Chen and Moschitti [56] present a method for continual learning for NER using a "Progressive Learning" technique. Inspired by Venkatesan and Er [67], the authors

adapted this approach to the sequence labeling task involved in NER. Attention-based Sequence-to-Sequence (Attention-based Seq2Seq) models dynamically focus on different portions of input sequences while generating their output. By assigning weights to input tokens based on their relevance to the current decoding step, attention mechanisms mitigate information loss, allowing for more accurate and contextually relevant sequence generation compared to traditional Seq2Seq models. The authors implemented a Bidirectional Long Short-Term Memory (BiLSTM) model and an attention-based Seq2Seq model to conduct experiments for progressive NER. Both models utilize word and character embeddings to represent input data. Specifically, the attention-based Seq2Seq model employs BiLSTM units in the encoder and a single-layer LSTM in the decoder, ensuring a one-to-one mapping between input words and output labels.

The progressive adaptation process consists of two key steps: first, training a model on source data, and second, transferring the parameters to the target model. Firstly, the model undergoes an initial training step on a source dataset $\mathcal{D}_S$ containing $\mathcal{C}$ NER classes. Then, in a subsequent step the model is further trained on a target dataset $\mathcal{D}_T$, which includes new examples annotated with the NER classes from the initial step as well as additional $\mathcal{E}$ new classes. Finetuning the target model involves modifying the output layer to accommodate new NER classes. To achieve this, the authors extend the output layer with nodes corresponding to each new class, initialized with weights drawn from a distribution based on pre-trained weights. Overall, their experiments never expand beyond two continual learning steps, marking one of the earlier investigations into expanding a NER model's knowledge through progressive or continual learning. This approach is categorized as a replay-based method due to the utilization of data containing all seen entity classes during the subsequent training step.

2.3.1.2 Single Epoch Recovery

Coria et al. [63] propose a straightforward finetuning method for continual learning in NER using a multilingual BERT base model. Their approach involves training a model sequentially on different languages, following a specified order, for the NER tasks. The model employs a two-layer feed-forward classifier with a hidden size of 768 and ReLU activation for sequence labeling. During training, the classifier's input is the last layer word hidden states after applying dropout with a probability of 0.1. This method is introduced within the broader context of continual learning for NER, aiming to address the challenges of adapting NER models to new languages while retaining performance on previously learned languages.

The training process involves training the model on various languages sequentially, with validation performed on the corresponding validation set for each language. The authors explore both forward and backward transfer during training, assessing forward transfer by comparing the average performance on the last language against monolingual and multilingual baselines. Backward transfer is evaluated by comparing the average performance on the first language with monolingual baselines. Additionally, the authors investigate the recovery of performance lost due to forgetting after continual training. They propose using a single additional epoch of training for the model at the end of the training sequence, leveraging data that includes information from all tasks in the training sequence. This approach aims to limit the effects of forgetting while potentially boosting forward transfer.

For continual learning, the authors focus on recovering performance lost due to forgetting. Their approach involves incorporating an additional epoch of training using data containing information from all tasks in the training sequence. This recovery process is crucial for limiting the effects of forgetting while potentially enhancing for-

ward transfer. Notably, this method is categorized as a replay-based approach due to its utilization of data containing all seen entity classes during the recovery process. Essentially, the recovery data acts as a form of replay, enabling the model to retain knowledge of previously learned tasks while adapting to new ones.

2.3.1.3 Learn and Review

Xia et al. [27] introduce the Learn and Review (L&R) method, which incorporates generative replay to the student/teacher implementation of Monaikul et al. [4]. They utilize a one-layer LSTM language model as a generator to address inter-type confusion of previously learned classes in existing student/teacher models. This confusion occurs when the model is unable to discriminate between different entity types during classification. For example, consider a model trained to identify different types of named entities in text, such as "person," "organization," and "location." Inter-type confusion may occur when the model misclassifies the name of a person as the name of an organization, exhibiting inter-type confusion between the "person" and "organization" entity types.

L&R method divides continual learning into two stages: the learning stage and the review stage. The learning stage employs the previous continual learning step's model as a teacher in a standard student/teach configuration. Additionally, for each continual learning step, a separate generator model is trained during this stage. Following the learning stage, a new model M_R is initialized from the student model.

During the review stage, for each previous continual learning step $i \in \{1, 2, \ldots, k-1\}$, the generator G_i creates unlabeled sentences related to the entity type from that step E_i. The sentences are fed to both the student and teacher models. Subsequently, the student model's predictions for the new class are concatenated with the teacher's predictions for the old classes to get a full probability distribution for all classes.

The new model M_R is trained using the Kullback–Leibler (KL) divergence loss between the student and the concatenated predictions, along with the cross-entropy loss between the new class prediction and the current data, which includes the student predictions on the new class as well. This is similar to (Eq. 2.15), except with the student predictions on the new class included in the KL divergence loss.

2.3.1.4 Language Model Augmented Learning

Language Model Augmented Learning (LAMOL) refers to a learning paradigm that incorporates language models, such as GPT (Generative Pre-trained Transformer), into the learning process to enhance performance on various tasks. Payan et al. [60] adapt the LAMOL model [24] for continual learning in NER, leveraging data replay. The model architecture is built on a pre-trained GPT-2 language model base [68], augmented with two layers of bidirectional Long Short-Term Memory (BiLSTM) networks. Each BiLSTM layer consists of 768 dimensions in both the forward and backward sequence processing directions, employing a hyperbolic tangent (tanh) non-linearity and linear transformation. Additionally, the model incorporates a Conditional Random Field (CRF) layer for label prediction. The parameters, excluding those from the GPT-2 base pre-trained on OpenAI's WebText, are randomly initialized, the entire parameter set is either trained or fine-tuned during the training process. In training their model, the authors assume that all entity type classes are known in advance, eliminating the need for label size expansion in subsequent continual learning steps when unseen labels are introduced during the continual learning steps. In the context of data replay, the authors delineate the process for each continual learning step, except the initial one, where a fixed size of replayed examples, set to 20% of the current continual learning step's training set, is selected, with an equal number of examples sampled from each previous step.

2.3.1.5 Joint Continual Learning Neural Network for Biomedical Information Extraction

He et al. [59] introduced the Joint Continual Learning Neural Network for Biomedical Information Extraction (JCBIE), which specializes in continual information extraction, including NER, using data replay. The model's objective is to jointly learn NER and Relation Extraction (RE) in continual learning settings. The NER task is divided into Span Detection (SP) and Entity Typing (ET), where SP utilizes BIOES tagging. JCBIE incorporates two separate BioBERT-based encoders for NER and RE. The authors augment input sentences with special entity markers to emphasize entity positions and types. During training, the NER encoder processes the original sentences, while the RE encoder utilizes sequences with markers based on gold labels. During inference, the model predicts entity spans and types, inserts markers accordingly, and performs RE prediction. Multi-head classifiers are employed for ET and RE, while a single-head classifier is used for SP. SP, ET, and RE are simultaneously trained during the process, and inference relies on SP results for ET predictions and SP and ET results for relation predictions.

The continual learning process in JCBIE is referred to as "Continual Multi-Corpora Learning," where data replay for NER and RE is conducted across datasets or corpora. In each subsequent learning step, data from previous steps, representing previously learned domains or corpora, are utilized. However, the authors do not provide specific guidance on choosing replay data or the amount of replay data in practical settings. Essentially, JCBIE implies an expanding training set as the model continues to learn, suggesting a continuous accumulation of knowledge over time.

2.3.1.6 Few-Shot Class-Incremental Learning

Few-shot class-incremental learning refers to the scenario where a model incrementally learns to recognize new classes with only a few examples available for each class.

Wang et al. [58] propose a framework for few-shot continual NER, where a model incrementally learns from only a few annotations (e.g., 5-10 samples per entity type) without requiring access to previous training data. The model achieves this by generating and training on synthetic samples. The authors argue that the traditional knowledge distillation approach suffers in the few-shot setting. To address this concern, they devised a method to generate more data samples that contain previous entity types.

To construct their synthetic data, the authors invert the teacher model and optimize reconstructed token embeddings such that the predictions from the teacher model match the corresponding randomly sampled label sequences containing previous entity types. Subsequently, distilling from the traditional teacher model with this data will enable the preservation of knowledge about previous entity types. The authors note that the generated input sequences may be unrealistic. To encourage more realistic generated data, they adversarially match hidden features of tokens between synthetic data and real data tokens, excluding the hidden states of entities of previous entity types within the synthetic data being matched.

The authors use BERT with a linear layer and a conditional random field (CRF) as the NER model. During each continual learning step, they initialize the student model from the teacher model to additionally preserve knowledge about previous entity types. When training with real data, the authors assume they have already generated a synthetic dataset with data samples containing all previously seen entity

types. They then use the current continual learning step's real data, the teacher model, and synthetic data to train the student model, treating the real and synthetic data differently during training.

When distilling with real data, the authors approximate the sequence-level distributions from the CRF using the top ten predictions to make the problem tractable. For the "new" entity types appearing only in the current step, they replace the teacher's predictions with the ground truth labels and train the student model with this modified real data using cross entropy to match output distributions with respect to the corrected top ten predictions using Equation 2.6 where $\mathrm{CE}(\cdot|\cdot)$ is cross entropy, $\hat{P}_{M^{t-1}}(Y|X)$ are the teacher's predictions from modified real data, and $\hat{P}_{M^t}(Y|X)$ are the student's predictions from modified real data.

$$L^{real}(D^t) = -\frac{1}{|D^t|} \sum_{X \in D^t} \mathrm{CE}(\hat{P}_{M^{t-1}}(Y|X), \hat{P}_{M^t}(Y|X)) \tag{2.6}$$

For distilling synthetic data, there are no ground truth labels to replace the predictions from the teacher with tokens that might be of an entity type in the current step. Essentially, some tokens predicted as "O" by the teacher could be considered a "new" type by the student. Thus, to compute a loss between the student and teacher, the authors decompose the sequence-level outputs of the CRF into marginal token-level predictions and then match the marginal predictions of the student and the teacher, combining the marginal predictions of new entity types and "O" from the student to match the dimensionality of the teacher using Kullback–Leibler (KL) divergence defined in Equation 2.7 where $\mathrm{KL}(\cdot|\cdot)$ is KL divergence, p_e^{t-1} are the predictions of the teacher, and $\hat{p}_e^t$ are the predictions of the student.

$$L^{syn}(D_r^t) = \mathrm{KL}(\hat{p}_e^t | p_e^{t-1}) \tag{2.7}$$

The final loss to train the student combines losses from distillation using the real data and distillation using the synthetic data as shown in Equation 2.8.

$$L = L^{real}(D^t) + \alpha L^{syn}(D_r^t) \tag{2.8}$$

2.3.1.7 Prototype-based NER

In prototype-based NER, the system learns prototypes or representations of different named entity types during training. These prototypes capture the characteristics or features common to entities of the same type. Kumar et al. [61] present a few-shot approach to continual NER utilizing entity-type prototypes. Their model architecture is built on a transformer-based model with a classification layer, where the last hidden layer serves as a prototype representation to convert the model into a prototypical network during training and inference. During training, if a token is correctly classified, the last hidden layer of the transformer is saved as a prototype for the token's gold entity class. Multiple prototypes are saved per entity class, with the number determined by a hyperparameter. Prototypes are only saved during the final epoch of training. During inference, the model calculates cosine-similarity scores between a token's last hidden layer representation and all saved prototypes. The model predicts a label for the token based on the k-nearest neighbor search on the computed cosine-similarity scores.

For continual NER, the authors first train a model to recognize a baseline set of entity types using multi-class cross-entropy loss. Prototypes for baseline entity types are saved during this process. To perform incremental learning, the classification layer of the baseline model is reset, and "O" entity-type prototypes are removed. Prototypes for old entity types remain unmodified throughout the training. A hybrid loss function is introduced to train the model on new entity types, incorporating both

cosine-similarity loss and cross-entropy loss. When a token belongs to an old entity type, cosine-similarity loss is computed with each prototype of the corresponding old entity type and added to the cross-entropy loss. When a token belongs to a new entity type, only cross-entropy loss is computed This process repeats for each continual learning step.

This approach is categorized as a replay-based method because it utilizes data containing all previously seen entity classes during model training for recognizing new entity types. Although it involves an additional loss term, categorizing it as a regularization method, its reliance on data replay for old entity types aligns it with the replay-based section. During experimentation, the authors' continual learning steps used only 30 additional samples to expand the model's knowledge.

2.3.1.8 Learning "O" helps for Learning More

Ma et al. [8] propose a representation learning method aimed at acquiring discriminative representations for entity classes and the "O" class. The authors introduce an entity-aware contrastive learning approach along with two distance-based relabeling strategies to enhance the learning of old classes. For each class c, a collection of K exemplars $\mathcal{M}_c = x_c^i, y_c^i, \bar{X}_c^i$ is maintained, where x_c^i denotes a token labeled as c, and $\bar{X}$ represents the context of that token labeled as "O." In their experiments, the authors opt for $K = 5$ Their training process spans N epochs for each continual learning step, organized into two stages: the initial M epochs focus on acquiring an entity-oriented feature space, followed by $N - M$ epochs dedicated to further refining the representation of the "O" class.

In the first step of their method, the authors aim to construct an entity-oriented feature space, where the distance between points reflects the semantic similarity of the entities. To achieve this, they utilize cosine-similarity and Supervised Contrastive

Loss [69] to train a nonlinear mapping $F(\cdot)$ on the output hidden state of a pre-trained language model. The Supervised Contrastive Loss (L_{SCL}) is defined in Equation 2.9 where I represents the set of indices of anchor tokens, $A(i)$ is the set of indices of tokens that are not i, and $P(i)$ denotes the set of indices of positive samples distinct from i.

$$L_{SCL} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} log \frac{e^{s(z_i, z_p)/\tau}}{\sum_{a \in A(i)} e^{s(z_i, z_a)/\tau}} \tag{2.9}$$

The hidden state representation after the nonlinear mapping is denoted as $z = F(\mathbf{h})$, and $s(\cdot)$ represents the cosine similarity. The authors exclusively apply contrastive learning on the entity classes and train the model using L_{SCL} during the initial M epochs. To further refine the representation of the "O" class, the authors calculate a threshold to select potential "O" class entity clusters from the feature space. For each entity class, they compute the class similarity, defined as the average cosine similarities between all distinct exemplars as shown in Equation 2.10.

$$S_c = \frac{1}{|\mathcal{M}_c|} \sum_{x_i, x_j \in \mathcal{M}_c, x_i \neq x_j} s(F(h(x_i)), F(h(x_j))) \tag{2.10}$$

The threshold $\mathcal{T}_{ent}$ represents the median class similarity score across all entity classes and is recalculated before each epoch to adapt based on the degree of convergence. Using this threshold, the authors implement entity-aware contrastive learning for the "O" class, employing auto-selected anchors and positive samples as shown in Equation 2.11 where I_O, P_O, and A_O are modified versions of the anchor and positive sample sets tailored for the "O" class, utilizing the threshold $\mathcal{T}_{ent}$.

$$L_{SCL,O} = L_{SCL}(I_O, P_O, A_O) \tag{2.11}$$

In the final $N - M$ epochs, the authors jointly optimize the representations of

entities and the "O" class using the objective function in Equation 2.12.

$$L = L_{SCL,O} + L_{SCL} \tag{2.12}$$

Additionally, to further distinguish old entity classes, the authors propose two distinct distance-based relabeling strategies, leveraging the previous model and the exemplar set. The first strategy involves prototypes, where they compute a prototype for each class using the exemplar representations from the previous model defined in Equation 2.13.

$$\mathbf{p}c = \frac{1}{|M_c|} \sum x \in M_c h_{t-1}(x) \tag{2.13}$$

After computing the prototypes for each class, the authors define a relabeling threshold by determining the lowest similarity score of all exemplars with their respective prototype. A token undergoes relabeling only if its similarity to any given prototype exceeds this threshold. Alternatively, the second relabeling strategy involves nearest neighbors. Here, tokens are relabeled based on their distances to the exemplars of each class, with relabeling occurring only if a token's distance falls within a specified threshold. To leverage the learned entity class representations fully, the authors employ the Nearest Class Mean classifier [19] for classification. For each token, the class prediction is defined in Equation 2.14 where $\mathbf{p}_c$ is the prototype of entity class c calculated with the exemplars as show in Equation 2.13.

$$y^* = \underset{c \in C_t^{all}}{\operatorname{argmax}}\ s(h_t(x), \mathbf{p}_c) \tag{2.14}$$

2.3.2 Regularization

In this section, we describe previous work that has developed regularization methods for continual learning for NER.

2.3.2.1 AddNER and ExtendNER

The paper by Monaikul et al. [4] employs knowledge distillation as a technique for continual learning for NER. Initially, a teacher model is trained to recognize a predefined set of classes. Subsequently, the predictions made by the teacher model are integrated into the objective function of a student model. The objective for the student model is to mimic the behavior of the teacher model by imitating its output probability distribution of classes for each token. This imitation process is achieved by computing the KL divergence between the probability distributions produced by the teacher and student models.

The authors introduce two student-teacher methods, namely AddNER and ExtendNER, which utilize knowledge distillation for continual learning. Both methods employ neural network architectures comprising an encoder layer followed by a linear layer for classification. Specifically, the authors use a BERT-based model for the encoder component. They make the assumption that the dataset provides labels in the BIO format, where each token is labeled to indicate whether it marks the beginning of an entity (B-), is inside an entity (I-), or is unrelated to any entity (O), denoting the "other" tag.

In the AddNER approach, the student model initially mirrors the architecture of the teacher model. Subsequently, an additional linear output layer is introduced to enable the student to learn new entity types. Importantly, each linear output layer shares the same encoder layer. During the training process, both the teacher and the

student process a sample. Consequently, both models generate a predicted output probability distribution for each token. The objective of the student is to minimize the weighted sum of two losses for each token: the KL divergence with respect to the teacher's predictions (which penalizes forgetting previous entity types) and the cross-entropy loss with respect to the dataset labels (which penalizes errors related to new entity types). The loss function $\mathcal{L}$ is defined in Equation 2.15 where α and β are hyperparameters that weight the contributions of the respective losses. KL$(\cdot)$ represents the KL divergence between the output probability distributions for each token from the teacher (p^{M_i}) and the student (p^{M_i+1}). CE$(\cdot)$ denotes the cross-entropy between the true class or gold label y and the predictions of the student (p^{M_i+1}).

$$\mathcal{L} = \alpha * KL(p^{M_i}, p^{M_i+1}) + \beta * CE(y, p^{M_i+1}) \tag{2.15}$$

The authors compute the KL divergence between the corresponding output layers in the student and the teacher models. Additionally, they calculate the cross-entropy between the newly added output layer and the gold labels obtained from the dataset. Since there are multiple output layers, conflicts may arise regarding the predicted class of a token. To resolve these conflicts, the authors combine the outputs for each token from different output layers using a set of heuristics.

With ExtendNER, the student model initially replicates the architecture of the teacher model. Subsequently, the authors extend the output layer to accommodate learning the new entity types. The training methodology resembles that of AddNER, where the student model is trained on the weighted sum of two losses. However, in ExtendNER, the model selectively computes either the KL divergence or the cross-entropy loss for each token, based on its corresponding label in the dataset. Specifically, the KL divergence is computed when a token's gold or true label is designated

as "O," while the cross-entropy loss is calculated when the gold label represents one of the new entity types.

2.3.2.2 Causal Framework for Continual Named Entity Recognition

Zheng et al. [36] propose Causal Framework for Continual NER (CFNER), a model designed to learn from historical data without resorting to replay mechanisms. Utilizing a student-teacher setup, their approach leverages the teacher model's capacity to interpret novel tokens within an established feature space. CFNER backpropagates loss for a single token based on a weighted average prediction of many tokens. The target token is referred to as the *anchor token*, while the remaining tokens contributing to the average are referred to as the *matched tokens*. Employing a k-nearest neighbor approach, a preserved model from the preceding continual learning phase identifies tokens closely aligned with the anchor token within the pre-existing feature space.

The model's prediction for the anchor token combines with its predictions for the matched tokens to form the final prediction for loss backpropagation. This process enables the influence of old data on new predictions without resorting to data replay. The model is trained on continual learning sets featuring only one labeled entity type, with the authors adopting a greedy sampling strategy to bias datasets towards the target entity type. Their loss function incorporates cross-entropy on the weighted average prediction of the new model for labeled tokens from the new class (*EffectE* in Equation 2.16), and KL divergence with the teacher's output probability distribution on "other" class tokens (*EffectO* in Equation 2.16):

$$
\begin{aligned}
\mathit{Effect} &= \mathit{EffectE} + \mathit{EffectO} \\
&= -\sum_{i} CE(\overline{Y}_i, Y_i) - \sum_{j} KL(\overline{Y}_j, \widetilde{Y}_j), \\
&\quad s.t. \quad D_i \in D^E, \quad D_j \in D^O.
\end{aligned} \tag{2.16}
$$

where $\overline{Y}$ represents the weighted average prediction of the new model, Y_i is the ground-truth label of the new entity type, $\widetilde{Y}_j$ denotes the output probability distribution of the old model on "other" class anchor tokens, and D_i and D_j signify tokens i and j from the new data corresponding to the new class D^E and the "other" class D^O, respectively.

Their approach to constructing a weighted average prediction ensures that tokens closely aligned with the anchor token exert a stronger influence on the final probability distribution Hu et al. [70].

To address label noise resulting from the retrieval of nearby tokens using the teacher model instead of relying on labeled replay data, the authors implement a confidence threshold-based curriculum learning strategy to filter out noisy "other" class samples defined in Equation 2.17 where M, δ_1, and δ_m are hyperparameters such that $\delta_1 > \delta_m$ and m represents a specific continual learning step.

$$
\delta_i = \begin{cases} \delta_1 + \frac{i-1}{m-1}(\delta_m - \delta_1), & 1 \leq i \leq m \\ \delta_m, & i > m, \end{cases} \tag{2.17}
$$

As the continual learning steps progress, the confidence threshold linearly decreases until reaching δ_m at continual learning step $m + 1$. Additionally, the authors incorporate a self-adaptive weight to enhance the KL divergence loss from the teacher model's soft labels as the number of learned entity types increases.

2.3.2.3 Decomposing Logits Distillation

Zhang et al. [37] propose a model-agnostic method that can enhance existing distillation-loss based models. Typically, these models generate predictions based on the softmax of the dot product of tensors, which aggregates positive and negative features and weights. However, Decomposing Logits Distillation (DLD) introduces a novel approach by decomposing this summation into positive and negative terms. Specifically, it constructs a positive logit from the sum of positive elements derived from element-wise multiplication, representing likelihood, and a negative logit from the sum of negative elements, signifying unlikelihood as as defined in Equation 2.18 where $Z_{i,e}^{t}$ denotes a complete logit, $Z_{i,e}^{t,p}$ represents the sum of positive elements, and $Z_{i,e}^{t,n}$ denotes the sum of negative elements.

$$Z_{i,e}^{t} = Z_{i,e}^{t,p} + Z_{i,e}^{t,n} \tag{2.18}$$

The DLD loss is then defined as the sum of positive and negative losses shown in Equation 2.19 where Θ^t represents the set of trainable parameters of model t.

$$\mathcal{L}_{DLD}\left(\Theta^{t}\right) = \mathcal{L}_{DLD}^{p}\left(\Theta^{t}\right) + \mathcal{L}_{DLD}^{n}\left(\Theta^{t}\right) \tag{2.19}$$

This loss can seamlessly complement models employing traditional KL divergence loss shown in Equation 2.20 where $\mathcal{L}_{LD}\left(\Theta^{t}\right)$ is the traditional KL divergence loss with teacher soft labels, and λ is a loss-balancing hyperparameter. This addition facilitates the computation of KL divergence loss between separate soft distributions, comprising sets of positive and negative terms outputted by both the student and teacher models The authors demonstrate that incorporating DLD enhances the performance of ExtendNER [4] and CFNER[36].

$$\mathcal{L}\left(\Theta^t\right) = \mathcal{L}_{CE}\left(\Theta^t\right) + \lambda\mathcal{L}_{LD}\left(\Theta^t\right) + \mathcal{L}_{DLD}\left(\Theta^t\right) \tag{2.20}$$

2.3.2.4 Continual Learning with Named Entity Recognition for DAnish

Vijay and Priyanshu [57] introduce a variant of Named Entity Recognition for DAnish [1] (NERDA) for continual learning (NERDA-Con). The authors propose incorporating EWC within the training module of NERDA [71] to regularize losses in future training. NERDA is a Python package designed to streamline the fine-tuning process of pretrained transformer models for NER tasks. The authors present two algorithm variants, one for distribution shifts and one for separate tasks. The authors consider the task to have undergone a distribution shift when the task remains the same, but the application domain changes. Consider an example of identifying named entities belonging to some fixed set of preselected classes but changing the domain from news articles into biomedical text. In this case with NERDA-Con, the authors apply EWC to the complete neural network, including the transformer-based and output layers. Alternatively, NERDA-Con allows using EWC only on the shared model parameters. When used this way, NERDA-Con enables the training of NER models across different tasks—for example, training a model first to perform coarse-grain classification and then to perform fine-grain classification.

2.3.2.5 Relation Distillation and Prototypical pseudolabeling

Zhang et al. [62] developed the Relation Distillation and Prototypical pseudo label method (RDP) to perform continual NER. Specifically, the authors define background shift as an effect that aggravates catastrophic forgetting when old and future

[1] According to the authors, NERDA has been expanded for use with other languages, not just Danish

named entity types are classified as the non-entity type, "O," during the current learning step. The RDP method has two components: Task Relation Distillation and Prototypical Pseudo Labeling. The Task Relation Distillation component is designed to combat catastrophic forgetting while providing the model with a balance between stability and plasticity. First, the authors introduce inter-task relation distillation loss to allow for control over stability. They first create a new set of soft-targets $\overline{Y}_t$ to distill against by replacing the dimensions in the "one-hot" ground truth that represent old entity types with the previous model's output probability distribution $\hat{Y}_t$. The inter-task relation distillation loss is defined in Equation 2.21.

$$\mathcal{L}_{cd}(\Theta^t) = -\frac{1}{|X^t|}\sum_{i=1}^{|X^t|}\overline{Y}_t(i)\log\hat{Y}_t(i) \tag{2.21}$$

Second, the authors introduce intra-task self-entropy loss to allow for control over plasticity. Their goal was to increase the model's confidence by minimizing the intra-task self-entropy of the current output probability distribution defined in Equation 2.22.

$$\mathcal{L}_{se}(\Theta^t) = -\frac{1}{|X^t|}\sum_{i=1}^{|X^t|}\hat{Y}_t(i)\log\hat{Y}_t(i) \tag{2.22}$$

These two losses are combined with the standard knowledge distillation loss shown in Equation 2.23 to define the total Task Relation Distillation loss with two hyperparameters to balance the tradeoff between stability and plasticity.

$$\mathcal{L}_{rd}(\Theta^t) = \lambda_1\mathcal{L}_{cd}(\Theta^t) + \lambda_2\mathcal{L}_{se}(\Theta^t) + \mathcal{L}_{kd}(\Theta^t) \tag{2.23}$$

The Prototypical Pseudo Labeling component is designed to combat the background shift effect that can worsen catastrophic forgetting. The authors aimed to clear up any confusion in the non-entity "O" class and generate high-quality pseudo

labels for the model. Their approach compares entity prototypes with the feature representations of tokens. When a token's feature representation is far from an entity prototype, it lowers the prediction probability for that type, allowing for a more refined pseudo-labeling of the token. To generate the prototypes, the authors start with the course pseudo labeling by taking the entity class with the highest probability for each token predicted by the old model. They then calculate the average embedding of an entity type as its prototype. Using the prototypes, they create a pseudo labeling of a sequence, $\widetilde{Y}_t$, and compute the cross entropy with the current output probability distribution defined in Equation 2.24. Overall, the objective function for the model within the RDP is the sum of the two losses defined in Equation 2.25.

$$\mathcal{L}_{ce}(\Theta^t) = -\frac{1}{|X^t|}\sum_{i=1}^{|X^t|}\widetilde{Y}_t(i)\log\hat{Y}_t(i) \tag{2.24}$$

$$\mathcal{L}(\Theta^t) = \mathcal{L}_{ce}(\Theta^t) + \mathcal{L}_{rd}(\Theta^t) \tag{2.25}$$

2.3.2.6 Confidence-based pseudolabeling and Pooled Features Distillation

Zhang et al. [9] developed the Confidence-based pseudo-labeling and Pooled Features Distillation method (CPFD) to perform continual NER. The CPFD method has two components. The first is Pooled Features Distillation designed to strike a balance between stability and plasticity. The second is Confidence-based Pseudo-labeling designed to combat background shift. The Pooled Features Distillation component uses the attention weights of the pretrained language models to retain embedded linguistic knowledge such as coreference and syntax information. The authors create a feature distillation loss to transfer linguistic knowledge from the old model. To mitigate the problem of excess stability, the authors incorporate pooling into their loss. They

argue that a lack of pooling can result in a model that is too stable but that substantial pooling can result in a model that is too plastic. Thus, the authors balance their pooling by aggregating statistics across only one of the head and sequence dimensions as shown in Equation 2.26 where $\mathbf{A}_l^t$ and $\mathbf{A}_l^{t-1}$ correspond to the attention weights of layer l for the old model $\mathcal{M}_{t-1}$ and the current model $\mathcal{M}_t$ respectively. K represents the count of attention heads.

$$\begin{aligned}\mathcal{L}_{PFD} = &\sum_{i=1}^{|X^t|}\sum_{j=1}^{|X^t|}\left\|\sum_{k=1}^{K}\mathbf{A}_{l,k,i,j}^{t} - \sum_{k=1}^{K}\mathbf{A}_{l,k,i,j}^{t-1}\right\|^2 \\ &+ \sum_{k=1}^{K}\sum_{j=1}^{|X^t|}\left\|\sum_{i=1}^{|X^t|}\mathbf{A}_{l,k,i,j}^{t} - \sum_{i=1}^{|X^t|}\mathbf{A}_{l,k,i,j}^{t-1}\right\|^2 \\ &+ \sum_{k=1}^{K}\sum_{i=1}^{|X^t|}\left\|\sum_{j=1}^{|X^t|}\mathbf{A}_{l,k,i,j}^{t} - \sum_{j=1}^{|X^t|}\mathbf{A}_{l,k,i,j}^{t-1}\right\|^2\end{aligned} \tag{2.26}$$

The Confidence-based Pseudo-labeling component uses median entropy as a confidence threshold when generating pseudo labels to reduce error propagation from the old model. The authors combat background shift by defining a new target,$\widetilde{Y}_t$, for the model to learn. To generate $\widetilde{Y}_t$, the authors combine the "one-hot" ground truth with the course pseudo labels from the old model, $\hat{Y}_{t-1}$. Essentially, for all the tokens labeled as "O" in the current ground truth, they use the prediction of the old model as the label. However, the previous model's prediction is only used if the model is confident enough; that is, the uncertainty of the model for its prediction is less than the median entropy confidence threshold. Lastly, because the entities in the current learning step tend to outnumber the old entity classes after pseudo labeling, the authors introduce an adaptive reweighting type balanced strategy where they can weigh classes differently based on their frequency. This is defined in Equation 2.27 where

η_i denotes the weight of the token at the location i in the sequence X^t. Overall, the total loss in CPFD is the sum of the two losses defined in Equation 2.28 with λ as a hyperparameter for balancing losses, and Θ^t being the set of learnable parameters for $\mathcal{M}_t$.

$$\mathcal{L}_{balance-pseudo} = -\frac{1}{|X^t|}\sum_{i=1}^{|X^t|}\eta_i \widetilde{Y}_t(i)\log \hat{Y}_t(i) \tag{2.27}$$

$$\mathcal{L}(\Theta^t) = \mathcal{L}_{balance-pseudo} + \lambda\mathcal{L}_{PFD} \tag{2.28}$$

2.3.2.7 Span-based Knowledge Distillation to preserve memories and multi-label prediction

Zhang and Chen [65] propose a neural SPAN-based model with Knowledge distillation to preserve memories and multi-Label prediction for continual NER (SpanKL). Given a sequence X of tokens, a span is a subsequence of tokens, $x_i, ..., x_j$, that are all part of the same entity. SpanKL aims to turn each span into a representation using the contextualized word embeddings and then perform binary classification for each entity type in the current learning step (i.e., in a given continual learning step with entity types PER and ORG, the model will predict whether a specific span is of the type PER or not and whether the span is of the type ORG or not). The authors develop the span representation using the contextualized word embeddings of the tokens. Given a span, the authors use two distinct one-layer feed-forward networks, one for the starting boundary token and one for the ending boundary token of the span, before performing the scaled dot product. Specifically, they use multi-head dot-product attention. They then organize all of the span representations related to a specific entity type into the upper triangle region of a matrix where the row and col-

umn indicate the beginning and end of that span. To predict the final classification of a span, they pass the predicted logit through a sigmoid activation before performing binary cross-entropy loss where each entity type in the current learning step is binary classified independently. Binary cross entropy is defined in Equation 2.29 where $p(k|s_{ij})$ is the gold label, $\hat{p}(k|s_{ij})$ is the predicted label, k is an entity type, and s_{ij} is a span between tokens x_i and x_j.

$$\begin{aligned}\mathcal{L}_{\mathcal{BCE}} = -\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{L} p(k|s_{ij}) log(\hat{p}(k|s_{ij}) \\ + (1 - p(k|s_{ij})) log(1 - (\hat{p}(k|s_{ij}))\end{aligned} \tag{2.29}$$

The authors specify that they compute the binary cross-entropy loss only between the span matrices of current entity types. For continual learning, the authors incorporate knowledge distillation into SpanKL. The teacher model predicts over the whole dataset and gives the soft label for every span for every old entity type. These are then used to compute the KL divergence loss with the current model shown in Equation 2.30 where $\widetilde{p}(k|s_{ij})$ is the soft label from the teacher model. The authors specify that they compute the KL divergence loss only between the span matrices of old entity types. The final loss for training is the sum of the two losses shown in Equation 2.31. For the model's final output, the authors combine the predicted overlapping entities into a flat one by only keeping the entity type prediction with the highest predicted score.

$$\begin{aligned}\mathcal{L}_{\mathcal{KD}} = -\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{L} \widetilde{p}(k|s_{ij})(log(\widetilde{p}(k|s_{ij}) - log(\hat{p}(k|s_{ij}))) \\ + (1 - \widetilde{p}(k|s_{ij}))(log(1 - \widetilde{p}(k|s_{ij})) - log(1 - \hat{p}(k|s_{ij})))\end{aligned} \tag{2.30}$$

$$\mathcal{L} = \alpha * \mathcal{L}_{\mathcal{BCE}} + \beta * \mathcal{L}_{\mathcal{KD}} \tag{2.31}$$

2.3.2.8 Span-based Knowledge Distillation with Reinforcement Learning

Chen and He [64] propose a model called Span-based Knowledge Distillation for NER (SKD-NER). Their goal was to prevent catastrophic forgetting by minimizing the effect of parameter updates based on poor optimization steps. They use reinforcement learning to adjust the knowledge distillation process by changing the distillation temperature and loss weights to allow for better forward compatibility of the model while addressing the issue of token noise. The SKD-NER model consists of a contextual encoder, span-prediction layer, label-loss layer, and reinforcement learning for knowledge distillation layer. For the contextual encoder, they use BERT. The span prediction layer utilizes a multi-headed attention mechanism that computes the span matrix using the entity's starting position token and ending position token to obtain the predicted entity span. They use Relative Position Encoding to add relative position information during the span prediction process. The label-loss layer incorporates a span-based cross-entropy to encourage the model to learn boundary information better.

The reinforcement learning for knowledge distillation layer starts with the standard knowledge distillation approach. However, during the knowledge distillation process, the authors introduce a reinforcement strategy to optimize the distillation temperature, which acts on the soft labels for each span of each old entity type. Their reinforcement learning strategy keeps a sequence of environment states. Each state is a concatenation of the vector representation of the input instance x_i, the prediction by the teacher model on x_i, and the loss of the student model on the x_i. An agent modifies the distillation temperature and the weight of the distillation loss for the

current teacher model. The policy function defined in Equations 2.32 - 2.34 where $F(s_j)$ is the state vector, θ is the trainable parameters, $a_j \in 0, 1$ is the action value, T is the temperature factor, and $weight$ is the weight of the distillation loss. They use accuracy on a validation set and the student loss as a reward, where a hyperparameter balances the reward between the training and validation sets, and they defer the reward until the end of each batch.

$$\pi_\theta(s_j, a_j) = [temp(s_j, a_j), klweight(s_j, a_j)] \tag{2.32}$$

$$temp(s_j, a_j) = a_j * [T + A * F(s_j)] + (1 - a_j) * T \tag{2.33}$$

$$klweight(s_j, a_j) = min[a_j * (weight + B * F(s_j)) + (1 - a_j) * weight, 0.1] \tag{2.34}$$

2.3.3 Parameter Isolation

In this section, we describe the parameter isolation methods.

2.3.3.1 Task-CL based on Sub-networks and Task similarity

Ke et al. [66] point out that most continual learning methods address only catastrophic forgetting or knowledge transfer and seek to introduce a method that addresses both. They propose **T**ask-CL based on **S**ub-networks and **T**ask similarity (TST). TST uses a frozen $BART_{large}$ as a base model and inserts a randomly initialized, frozen adapter [44] into each transformer layer. A number of binary masks are created with the same parameter size as the inserted adapter. Inspired by network pruning and other sub-network masking methods, TST trains these binary masks following Wortsman et al. [72] to turn off select parameters of the frozen adapter to

change the model output.

TST seeks to perform task-incremental learning using NER as one of its tasks. When a new task is encountered, for each previously trained mask, the model saves an importance score for that mask on the new data. A mask with a high importance score indicates that a previous and current task may be closely related enough to facilitate knowledge transfer by training further on the current task. Aside from outperforming the compared baselines, the authors found that TST did enable knowledge transfer between NER and the related task Aspect Sentiment Classification by randomizing the order of the tasks and finding a negative average forgetting measure when each task was learned before the other.

2.3.3.2 Lifelong Few-Shot Language Learning with Prompt tuning of T5

Qin and Joty [50] explore few-shot task-incremental continual learning and present Lifelong Few-Shot Language Learning with Prompt tuning of T5 (LFPTS). Prompt tuning involves the creation of a matrix of embeddings not restricted to the vocabulary of any model, known as a *soft prompt*, and the training of those embeddings by backpropagating loss to increase the probability of a label being output by a frozen model. The authors point to the likelihood of overfitting during task incremental learning as a motivation to use prompt tuning as an alternative to model fine-tuning. Like adapter-based methods, prompt tuning allows for the training of relatively few parameters when compared to fine-tuning large, generalist models, and both methods are able to retain the generalizability of pre-trained models by freezing their weights.

Following Lester, Al-Rfou, and Constant [73], LFTP5 trains prompts that cause a frozen T5 model to act as both a task solver and a pseudo-sample generator, creating its own replay data to train with NER, text classification, and summarization are learned sequentially in a stream, multiple prompts are used for multiple tasks.

The authors explore the effect of initializing prompt embeddings for a task with those of the previously learned task, and a KL divergence loss is taken between the outputs of previous and current task prompts to combat forgetting by encouraging label agreement.

2.4 Evaluation of Continual Learning NER Systems

Named Entity Recognition methods are evaluated using precision, recall, and F_1 scores. Precision is the ratio between correctly predicted mentions over the total set of predicted mentions for a specific entity, recall is the ratio of correctly predicted mentions over the actual number of mentions, and F_1 is the harmonic mean between precision and recall. The micro- and/or macro- F_1 scores are typically reported as a comparison across systems. The macro F_1 score treats all classes equally. where the micro F_1 score aggregates the contributions of all classes taking into account the inherent class imbalance of the entities in the dataset.

There are four main datasets that the presented NER systems have been evaluated on: CoNLL-03, OntoNotes5, I2B2, and FewNERD. The CoNLL-03 dataset [74] consists of approximately 1,400 sentences from Reuters news articles that have been human-annotated with persons, locations, organizations, and miscellaneous entities [2]. The OntoNotes5 dataset [75] consists of various genres of text such as news, telephone conversations, and talk shows, in English, Chinese, and Arabic. It contains 18 entity types, the six most frequent types being: person, organization, geopolitical entity, date, cardinal number, and Nationalities or religious or political groups (NORP) [3]. The I2B2 dataset [76] consists of 141,000 sentences from 1,304 medical records for 296 diabetic patients pulled from the MIMIC-3 corpus. It contains 16 entity types,

[2]https://www.clips.uantwerpen.be/conll2003/ner/

[3]https://catalog.ldc.upenn.edu/LDC2013T19

including age, doctor, hospital, patient, and medical_record, to name a few [4]. The FewNERD dataset [77] consists of 188,238 sentences from Wikipedia annotated by humans. It contains 8 coarse-grained entity types and 66 fine-grained types. The coarse-grained entity types are: person, location, organization, art, building, product, event, and miscellaneous. For a detailed list of fine-grained entity types we direct you to their publication [5].

Table 2 reports micro- and macro- F_1 scores across these four aforementioned popular datasets for each of the reviewed methods if possible. ExtendNER [4], CFNER [36], CPFD [9], Decomposing Logits Distillation (DLD) [37], and RDP [62] are directly comparable on CoNLL-03, OntoNotes5, and I2B2. Their CoNLL-03 setup contains two classes in the initial step and one class in each following continual learning step. For OntoNotes and I2B2, their setup contains eight classes in the initial step and two classes in each following continual learning step. SKD-NER [64], SpanKL [65], and NERDA-CON [57] can be compared on FewNERD, each using only the 8-coarse grained entity types from this dataset. SKD-NER [64], SpanKL [65], and Learn & Review [27] can be compared on OntoNotes5, each used only the six most frequent entity classes. Attention Based Seq2Seq [56] reports results on CoNLL-03 using three classes in the initial step and one class in the subsequent continual learning step. Few-Shot Class-Incremental NER [58] report results on CoNLL-03 using one class in the initial step and one class in each of the following continual learning steps. They report results on OntoNotes5 for all 18 entity classes but use their own method when grouping them into continual learning steps. Learning "O" [8] reports results on FewNERD by using six steps of 11 entities and on OntoNotes5 by using

[4]https://portal.dbmi.hms.harvard.edu/projects/n2c2-nlp/

[5]https://ningding97.github.io/fewnerd/

three steps of 6 entities.

Note that JCBIE [59], LAMOL-NER [60], Single Epoch Recovery [63], and Proto-NER [61] conduct experiments that do not contain any overlapping datasets and are subsequently not included in the table. TST [66] and LFTP5 [50] are task-incremental continual learning methods and the experiments conducted in the original papers do not produce results comparable to the other methods reviewed in this paper. Subsequently they are also not included in the table. The OntoNotes5 results for Few-Shot Class-Incremental NER [58] were extrapolated from a graph and represent an approximate measure. All of the results for ExtendNER [4] were included from [36].

Table 2. NER Methods and their F_1 Results

Approach	**Method**	**CoNLL-03**		**OntoNotes5**		**I2B2**		**FewNERD**	
Category		**Micro**	**Macro**	**Micro**	**Macro**	**Micro**	**Macro**	**Micro**	**Macro**
Regularization	ExtendNER [4]	76.66±0.66	66.36±0.64	76.85±0.77	54.37±0.57	52.25±5.36	30.93±2.77		
	CFNER [36]	80.83±0.36	75.20±0.32	80.68±0.25	60.52±0.84	69.07±0.89	51.09±1.05		
	CPFD [9]	85.70±0.19	83.49±0.16	83.38±0.18	66.27±0.75	81.05±0.87	65.04±1.13		
	DLD [37]	83.40±0.87	79.54±0.71	82.26±0.55	63.07±1.13	70.88±0.70	53.21±0.92		
	RDP [62]	85.82±0.36	83.59±0.37	83.30±0.30	66.92±1.26	80.08±0.40	63.72±0.71		
	SKD-NER [64]				88.17				67.14
	SpanKL [65]				88.98				62.15
	NERDA-CON [57]	77.61						78.40	
Replay	Seq2Seq [56]		88.20						
	Few-Shot CINER [58]		64.18		33*				
	Learn & Review [27]		85.74±0.44		83.02±0.63				
	Learning "O" [8]			83.18				51.16	

CHAPTER 3

METHODOLOGY

3.1 Research Goals and Experimental Design

Through our review, we identified a critical gap in the literature. Generating and using synthetic data to combat forgetting has been studied in continual learning for computer-vision models, notable examples being [22, 23, 26]. However, for NER models, using synthetic data to combat forgetting has only been studied to a limited extent with LSTM generators [27] or inverted models [58]. Thus, a critical gap, namely the absence of generative large-language models to create synthetic data to combat forgetting, presented a timely and promising avenue for exploration.

This work focuses on building the foundation for a generative replay approach for continual NER. We aim to determine the efficacy of using Open AI's GPT-4 model to generate synthetic data to supplement the training of NER systems. To that effect, we aim to answer the following questions:

Research Question #1: Can synthetic data be generated to mimic the format of authentic NER training data? To answer this question, we plan to repeatedly prompt Meta's Llama-2-70b and OpenAi's GPT-4-0125-preview and visually inspect the output. The visual inspection will be implemented with the assistance of a domain expert in information extraction to determine whether we can use the generated synthetic data to train a NER model. We will consider generated synthetic data usable if it resembles authentic NER training data in the CoNLL-03 format, i.e., sequences of tokens, each with corresponding entity labels. Figure 3 shows an example sentence in both BRAT and CoNLL-03 formats. During

this process, we will construct and refine an ad-hoc prompting strategy to generate and improve the outputs of the large-language models. We will refine our prompting strategy until the large-language model, if possible, generates consistent outputs in the correct format.

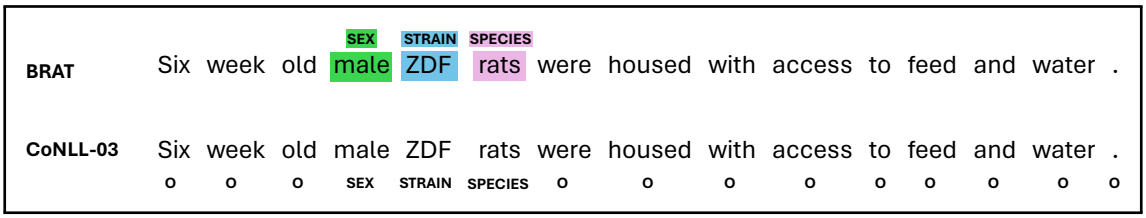

Fig. 3. A side-by-side comparison of a sentence in the BRAT and CoNLL-03 formats. In BRAT, the sentence is highlighted for different entity types only. BRAT is easier for annotating data however, NER models require data in CoNLL-03 format for training. A model's parameters are updated based on the cross entropy between the true and predicted label for each word, which requires each word to be assigned a true label. BRAT does not provide such information but, in CoNLL-03, an equal length list of labels, including the non-entity type, is created for the sentence assigning one label to each token.

Research Question #2: Is synthetic data similar to the authentic data? To answer this question, we plan to compare the contextualized embeddings for entities of the same entity type across authentic and synthetic data. We will use a pretrained but not finetuned DeBERTa-based model to generate these contextualized embeddings. We can determine the similarity between the two groups by clustering the embeddings and comparing the two clusters. Overall, the closer the synthetic entity embeddings are to the authentic ones, the more similar the synthetic data is to the authentic data.

Research Question #3: Does the addition of synthetic data improve model performance? To answer this question, we first plan to select three target entities to study from a dataset using a baseline model trained in the traditional fash-

ion for NER. We intend to select a well-represented and high-baseline-performing entity, a well-represented and low-baseline-performing entity, and a poorly-represented and low-baseline-performing entity. The task of NER is one that inherently deals with class imbalance. Additionally, even when trained in the traditional sense, not all entities are equally easy to recognize. Thus, our motivation for these three target choices is to differentiate and measure the effect of our approach on different naturally occurring scenarios for entities.

Next, we plan to filter the chosen dataset into three variants, one for each targeted entity. The filtered variants will be labeled for only the "O" and targeted entity types. Filtering the original dataset in this fashion will convert the standard multi-class classification problem for NER into a binary classification problem. We will train a model for each filtered variant using only authentic data as the baseline for comparison with our approach.

The first experimental model we propose is "combined_gen." For this model, we will generate a fixed amount of synthetic data. Then, we will augment the baseline model's training set with all of this generated synthetic data and train the model. For data generation, we intend to use a 5-shot example prompting, which is to include five examples of real data in each prompt. The motivation behind using 5-shot was to restrict the amount of human-annotated data needed when using our method. This restriction significantly reduces annotation labor and time costs, especially when fully synthetic data could replace real training data. Additionally, we propose a second experimental model, "combined_gen_small." For this model, we will only augment the baseline model's training set with 10% of the generated synthetic data and train the model.

Research Question #4: Is solely using synthetic data enough to achieve performance on par with a baseline? To answer this question, we propose

a third experimental model, "replaced_gen." For this model, we will not use any authentic training data and solely train a model with the generated synthetic data. Figure 4 visualizes the differences between the training set for each of the described experimental models.

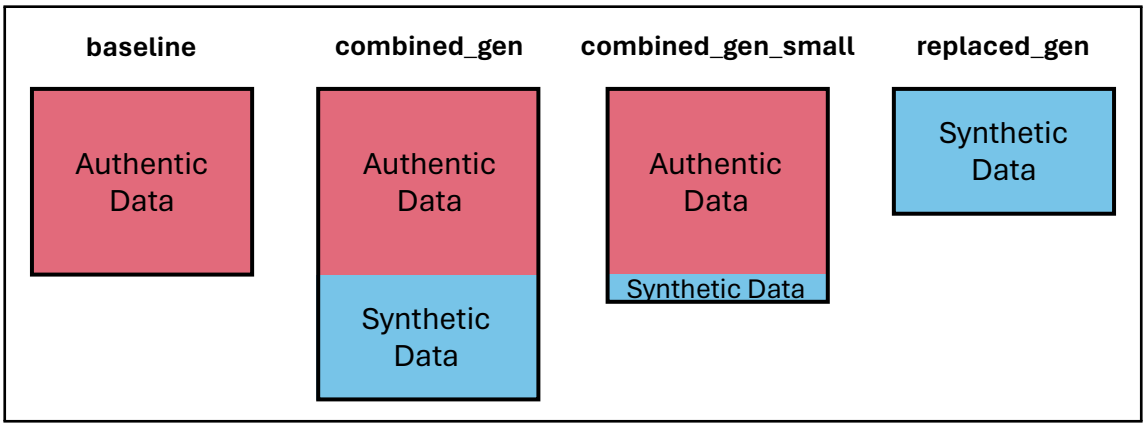

Fig. 4. Visualization of each proposed variant of the training set.

Research Question #5: How do different prompting strategies for generating synthetic data affect model performance? We plan to explore a second prompting strategy to answer this question. With this additional strategy, we will generate an equivalent amount of synthetic data as with the ad-hoc strategy. We then plan to repeat the same experimental models, "combined_gen," "combined_gen_small," and "replaced_gen." We will then compare results to determine the effect, if any, of prompting on the efficacy of synthetic data for NER.

3.2 System Design

This section provides a detailed description of the structural framework used for the NER system, including data preprocessing, the model architecture, the training methodology, and the inference procedure. We developed the model using PyTorch

Lightning [1], a lightweight PyTorch wrapper that aims to streamline the training process for PyTorch models. PyTorch Lightning provides a high-level interface for organizing code modularly, reducing development time for complex machine learning projects. It abstracts the initialization of training loops, logging, and distributed training in favor of model development.

3.2.1 Data Preprocessing

3.2.1.1 Dataset loading

We first use the brat rapid annotation tool (BRAT) brat2conll converter to convert the TAC dataset from BRAT format to CoNLL-03 format for NER. We then read each document into a Pandas data frame. Each sentence and corresponding sequence of entity labels are collected into lists respectively and grouped to represent instances. We instantiated a PyTorch Lightning Data Module [2] for the TAC dataset, allowing for better reusability in the future for different experiments and models without duplicating code. Another benefit of designing our model to use Data Modules is the ease of using different datasets. We simply pass a different module to our model's trainer.

3.2.1.2 Removal of IOB Tags

The entities were initially labeled in the IOB format. The IOB (Inside, Outside, Beginning) format in NER serves as a methodology for annotating and categorizing named entities. This format uses a hierarchical tagging scheme, wherein each token within a given text is labeled to denote its positional context within a potential

[1] https://lightning.ai/docs/pytorch/stable/

[2] https://lightning.ai/docs/pytorch/stable/data/datamodule.html

entity. A "B-" proceeding an entity label, e.g., "B-PER," indicates the beginning of that entity. An "I-" proceeding an entity label, e.g., "I-PER," indicates a token inside an entity. We removed the IOB tags for our experiments and replaced them with the generic entity label (e.g., I-PER becomes simply PER). Additionally, we remove non-contiguous entity types from the dataset by replacing them with the "O" non-entity type label. Figure 5 shows an example sentence labeled in both the IOB format and non-IOB format.

IOB	John	Smith	works	at	Virginia	Commonwealth	University	.
	B-PER	I-PER	O	O	B-ORG	I-ORG	I-ORG	O
No-IOB	John	Smith	works	at	Virginia	Commonwealth	University	.
	PER	PER	O	O	ORG	ORG	ORG	O

Fig. 5. An example sentence labeled with both IOB and non-IOB labels.

3.2.1.3 Filtering on TAC-variants

The full TAC dataset was filtered three times, once for each targeted entity type we selected. The filtering process involved relabeling as "O" all tokens not already labeled as "O" or the targeted entity type.

3.2.2 Architecture

3.2.2.1 Underlying Model

The underlying model used within our framework is based on DeBERTa. He et al. [78] developed Decoding Enhanced BERT with disentangled attention (DeBERTa) and introduce two major changes to the standard BERT architecture: disentangled attention and the enhanced mask decoder. Disentangled attention aims to reduce the interference between different attention heads. Within an encoder block, DeBERTa

represents tokens using two vectors, one for content and one for relative position. By decomposing content and position, this mechanism allows the model to better focus on different aspects of the input text independently. Additionally, incorporating absolute positional information before the final classification layer enhances the decoding of masked tokens during the masked-language-modeling pre-training task. He, Gao, and Chen [79] develop DeBERTaV3, which improves the original DeBERTa model by replacing the masked-language-modeling task with replaced token detection (RTD). The RTD task uses a generator to create corruptions to token sequences. The authors then treat DeBERTaV3 as a discriminator to predict whether a token in the corrupted input is either original or replaced by the generator, similar to the idea behind Generative Adversarial Networks (GAN). We first use the DeBERTa tokenizer before passing the subtokens through pretrained DeBERTa-v3-large. We then pass the outputs through a broadcast linear layer with a softmax. The final output for each subtoken is its predicted entity class. When trained with a mix of authentic and synthetic data, the files are read into memory and joined into one large training set. Figure 6 shows the model architecture of our DeBERTa based NER model.

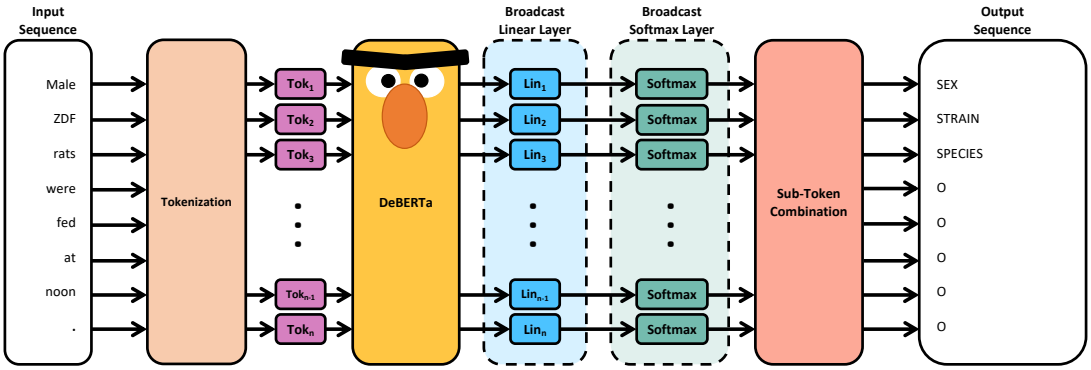

Fig. 6. The model architecture.

3.2.2.2 Regularization

Regularization techniques can prevent overfitting and improve the generalization ability of machine learning models. We use several techniques with our model. We use *logit scaling*, which involves rescaling the logits or raw predictions of a model before applying a softmax function to help calibrate and stabilize the model's predictions. We use a *learning rate scheduler* to adjust the learning rate during training to help the model converge more effectively. We use a scheduler that raises (warm-up) and then lowers the learning rate on an epoch schedule, with the highest learning rate occurring 30% into the training sequence. We use batching to reduce the number of samples the model processes before updating its parameters; we set the batch size to 32. We used *gradient clipping* to scale the gradients during training and prevent them from becoming too large, helping to prevent/mitigate the exploding gradient problem. We also used *gradient flooding*, which adds a constant value to the gradients to prevent them from becoming too small, helping to prevent/mitigate the vanishing gradient problem. We used *label smoothing* that helps with class imbalance by assigning some extra probability of other classes to the model's predictions. We also use *dropout* to drop a certain proportion of neurons during training randomly and encourage the model to learn multiple paths for information to traverse through the network. Within DeBERTa, we also use *attention dropout*, which refers to applying dropout specifically to attention mechanisms in models such as transformers, to improve their generalization performance.

3.2.3 Training

We feed the tokenized sentences into the DeBERTa model. The model is tasked with predicting the sequence labels for each token in the input. Based on the difference

between predicted and true labels, a loss value is calculated and back-propagated through the DeBERTa model to adjust its weights. We use the AdamW [80] optimizer, which decouples weight decay from the optimization steps. We train each model for ten epochs. We use checkpointing to take the model that performs the best on the validation set during training. Figure 7 shows how generated data is incorporated into the training procedure.

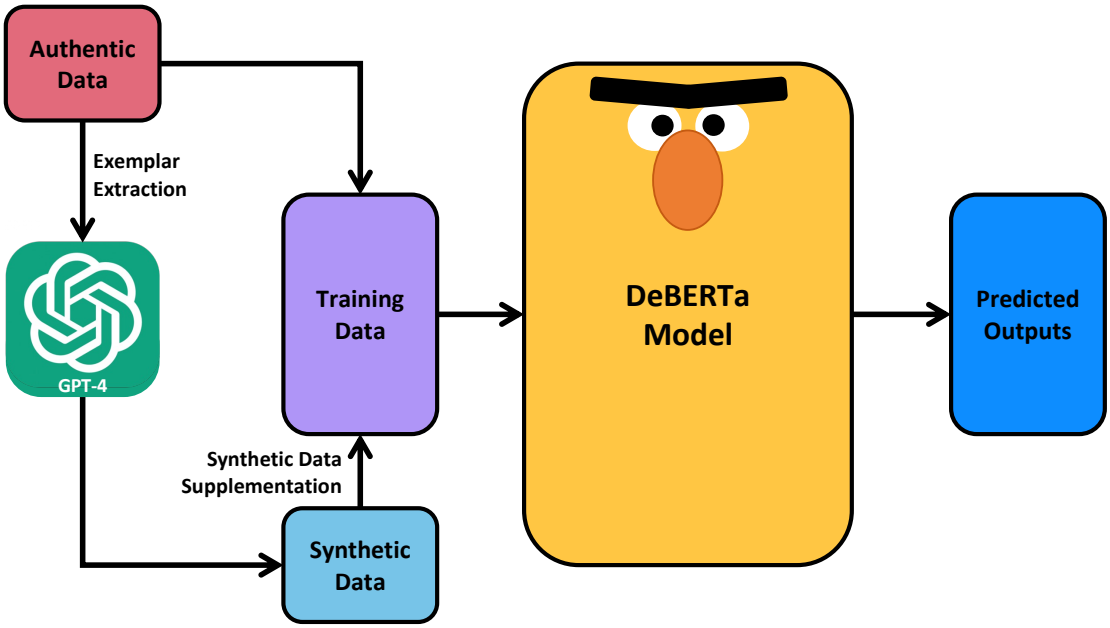

Fig. 7. Incorporation of Synthetic Data into Training.

We trained our models using the resources provided by the VCU High Performance Research Computing (HPRC) core. [3] The VCU HPRC core facility occupies approximately 2,000 square feet of space on the third floor of Harris Hall on the Monroe Park campus. The mission of the HPRC is to provide high performance computing services for the VCU research community. To accomplish this goal, the HPRC maintains two major supercomputing clusters, each specialized for different

[3] https://hprc.vcu.edu

computing environments: Athena and Fenn. We used the Athena cluster.

Athena is the primary cluster for large scale parallel computing. Athena consists of 109 compute nodes with a total of 7628 CPU cores, 38 TB of RAM, 28 GPUs, 3 PB of lustre filesystem, and a high-speed InfiniBand architecture of up to 100 Gb/s. Each node comprises between 28 to 128 cores, and 128 GB to 1 TB of RAM. GPU nodes comprise the NVIDIA V100, A100, and H100 GPUs. We trained our models exclusively using NVIDIA A100 GPUs. The clusters provide 70+ software packages including R, SAS, Matlab, MPI, Bioperl, CellRanger, Gaussian16, VASP, Intel compilers, MCNP, nekRS, OpenFOAM, Salmon, etc. To facilitate the usage of the software and lower the barrier to non-traditional users, a web portal based on Open OnDemand offers a graphical user interface to interactive applications, including Jupyter lab, RStudio, VS Code, Matlab, SAS, etc. To support this infrastructure, the HPRC employs 6 FTE staff positions: Alberto Cano, faculty director, Mike Davis, technical director; three systems analysts; and an applications analyst. In addition to maintaining the hardware, the HPRC works collaboratively with VCU researchers to maintain and optimize a large number of applications, scientific, statistical and development software.

3.3 Data

The National Toxicology Program (NTP), an interagency program headquartered at the National Institute of Environmental Health Sciences (NIEHS, part of the National Institutes of Health), and the Environmental Protection Agency (EPA) created the 2018 TAC SRIE dataset for the Text Analysis Conference Systematic Review Information Extraction (SRIE) task. They were interested in automated systems for information extraction from systematic reviews of environmental chemicals to reduce human labor costs while maintaining quality. The following link,

"https://tac.nist.gov/2018/SRIE/", provides more information about the task. The 2018 TAC SRIE dataset contains experimental design factors for the categories of Animal, Dose, Exposure, and Endpoint found in the "Material and Methods" section of journal articles describing experiments related to toxicity and health effects on animals of environmental agents. Extracting these entities is difficult due to the diversity of writing and presentation styles in published literature. Many publications contain multiple experiments with various exposures and doses evaluated at multiple endpoints. Authors may report experimental details with different units and often use many different names to refer to the same chemical. In addition, this information may be presented in the text, a table, a figure caption, or a figure itself. Thus, creating information extraction models for this domain is a non-trivial task.

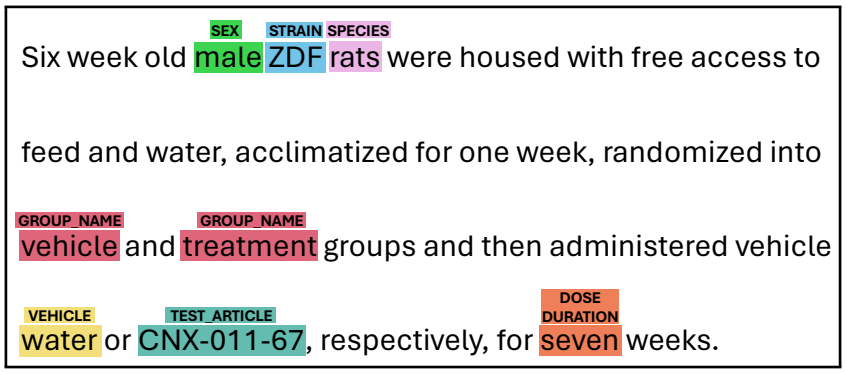

Fig. 8. Method section excerpt with entities highlighted

The TAC dataset contains 7,417 sentences annotated for 24 different entities. After preprocessing as defined in the previous section, the number of entity types is reduced from 24 to 21. Due to processing errors within the brat2conll conversion scripts from BRAT, five pairs of files did not get included in the experiments. We summarize information about the entities from the correctly processed files in Table 3.

We trained and evaluated a baseline model on the full TAC dataset and collected

Table 3. Entities from the TAC SRIE dataset

Category	**Entity**	**# of instances**	**# of entities**	**# of labels**
Non-entity	O	7,402	N/A	171,249
Animal	CellLine	38	38	133
	GroupName	462	734	1,970
	GroupSize	250	335	736
	SampleSize	40	41	79
	Sex	353	481	538
	Species	1,133	1,363	1,427
	Strain	233	323	821
Dose	Dose	301	585	1,000
	DoseDuration	142	182	211
	DoseDurationUnits	137	164	198
	DoseFrequency	70	81	185
	DoseRoute	394	466	786
	DoseUnits	292	438	1272
	TimeAtDose	51	86	139
	TimeAtFirstDose	30	32	54
	TimeAtLastDose	9	11	13
	TimeUnits	363	540	605
Exposure	TestArticle	1,056	1,444	2,730
	TestArticlePurity	16	23	91
	TestArticleVerification	3	3	78
	Vehicle	277	324	723

results as presented in Table 4. Then we selected three entities based on the criteria outlined in Research Question #3. In particular, for our experiments, we were interested in three entities from the Animal group: "CellLine," "GroupName," and "Species."

"CellLine" denotes a population of cells derived from a single cell and cultured under specific laboratory conditions. These cell lines are tools for studying cellular processes, disease mechanisms, drug responses, and various biological phenomena. Each

Table 4. TAC Baseline Results

Category	**Entity**	**Test** F_1
Non-entity	O	0.98
Animal	CellLine	0.73
	GroupName	0.73
	GroupSize	0.84
	SampleSize	0.43
	Sex	0.92
	Species	0.91
	Strain	0.92
Dose	Dose	0.77
	DoseDuration	0.79
	DoseDurationUnits	0.83
	DoseFrequency	0.52
	DoseRoute	0.84
	DoseUnits	0.81
	TimeAtDose	0.58
	TimeAtFirstDose	0.55
	TimeAtLastDose	0.0
	TimeUnits	0.71
Exposure	TestArticle	0.67
	TestArticlePurity	0.57
	TestArticleVerification	0.0
	Vehicle	0.68
Overall	Micro-F_1	0.97
	Macro-F_1	0.67

cell line typically possesses unique characteristics and behaviors, reflecting its tissue of origin and any genetic modifications introduced during culturing. Researchers utilize cell lines from diverse sources, including human, animal, and microbial origins, to investigate a wide range of scientific questions and to develop therapeutic interventions. "GroupName" refers to distinct categories or levels within a dataset that are compared to assess differences or effects. These groupings typically represent experimental conditions, treatment regimens, or other factors of interest being investigated. Assigning observations to appropriate group names allows researchers to

conduct comparative evaluations and infer relationships or associations between variables. Group names play a crucial role in hypothesis testing, where researchers aim to determine whether observed differences between groups are statistically significant or merely due to chance. "Species" refers to a fundamental unit of classification in the taxonomic hierarchy. It represents a group of organisms capable of interbreeding and producing fertile offspring under natural conditions. Each species typically exhibits distinct morphological, physiological, and behavioral characteristics, contributing to its unique identity within an ecosystem. Within scientific research involving animal subjects, "Species" denotes the specific type or category of organisms under investigation, encompassing a variety of taxa such as mammals, birds, reptiles, amphibians, fish, and invertebrates.

We created the three variants of the TAC dataset by filtering it. Each variant was filtered for a specific entity label, marking the remaining labels as "O". We then partitioned each variant into a training, validation, and test set with a stratified split of 70/15/15 percent of the full TAC dataset, respectively. Table 5 displays a summary of the entity statistics for each of the filtered and partitioned versions.

Table 5. Entities from the filtered TAC SRIE dataset

Filter	**Entity**	**# of instances**	**# of labels in Train**	**# of labels in Val**	**# of labels in Test**
CellLine	CellLine	38	99	15	19
	O	7,417	130,003	28,004	26,898
GroupName	GroupName	462	1,460	266	244
	O	7,417	128,705	27,042	27,321
Species	Species	1,133	988	207	232
	O	7,412	128,986	27,029	27,596

3.4 Generating Synthetic Data

The synthetic data for our experiments was generated using 5-shot prompts with OpenAi's GPT-4-0125-preview model. This section describes the specific procedure

used when generating the data.

We first extracted example instances, called exemplars, to provide context within the prompts. These exemplars were extracted from the unfiltered version of the TAC dataset. We considered an instance suitable if and only if it met our extraction criteria. Specifically, the unique entity labels present in the instance contained exclusively the specific entity label we wished to generate for and the "O" label. We carefully partitioned the filtered versions of the dataset into the training, validation, and testing sets. We ensured that all the extracted exemplars were placed in the training set. This was necessary to prevent reporting results on an instance used to generate synthetic data. For the "CellLine" label, we extracted 9 exemplars, and for "GroupName" and "Species", we extracted 50 exemplars. The choice in the number of extracted exemplars depended on the availability of the data; however, we set an upper bound of 50.

For constructing the prompts, we used two different strategies. The first strategy was ad-hoc, where we collectively designed a prompt using our domain knowledge. The second strategy, which we refer to as Knowledge Entity Extraction (KEE), was inspired by Shao et al. [81], who were inspired by Mishra et al. [82]. Our KEE style prompt was created using Shao et al. [81] as guide. The prompts are comprised of different mixtures of five generic components: *Task Descriptions*, *Entity Definitions*, *Task Emphasis*, *Task Examples*, and a *Prompt*. We use the *Task Descriptions* to ask the large-language model to take on a role, for example "You are an experienced biologist, capable of easily recognizing named entities." We also define the overall task to complete and output format within the *Task Descriptions*. The *Entity Definitions* provide the large-language model with specific definition of the entity that we would like to be generated. The *Task Emphasis* is to provide the large-language model with additional rules to follow when generating its output, such as "Ensure the samples

are of adequate length." The *Task Examples* are the 5-shot exemplars. The *Prompt* is a component we added to formally request the large-language model generate 5 instances of synthetic data for NER. Figure 9 compares the generic structure of the two different prompting strategies.

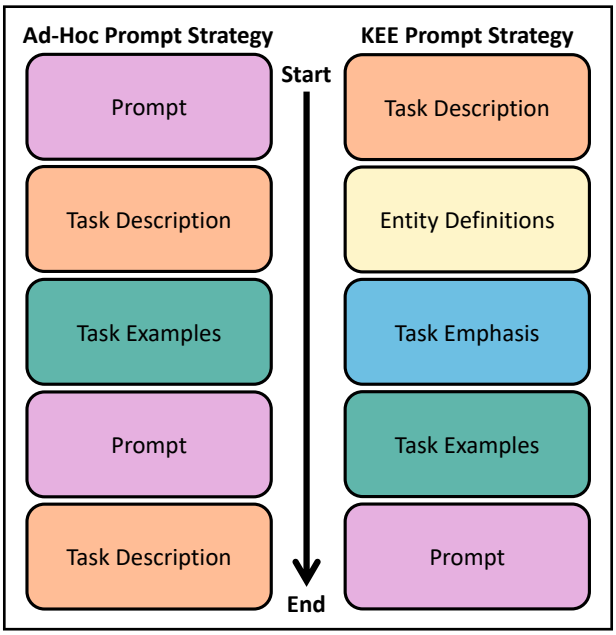

Fig. 9. Comparison of prompting strategies

We utilized five exemplars (5-shots) within each prompt. For "Species" and "GroupName", we created 10 prompts under each prompting strategy. For "CellLine", we created 2 prompts under each prompting strategy. The exemplars used were the same across prompting strategies. That is, ad-hoc Prompt 1 and KEE Prompt 1 contained the same five exemplars, ad-hoc Prompt 2 and KEE Prompt 2 contained the same five exemplars, and so on. There is no overlap of exemplars across prompts within a prompting strategy except with "CellLine." There were only 9 samples that met the criteria for exemplar extraction, thus we used one exemplar twice to to allow us to create two 5-shot prompts.

To generate the synthetic data, we initially experimented with using Meta's Llama-2-70b with limited success. The model struggled to produce outputs that were correctly formatted and did not produce outputs that passed visual inspection. Additionally, the model often shifted into unrelated tasks or domains, occasionally even producing outputs that were not human-readable. As a result, we transitioned to utilizing OpenAi's GPT-4-0125-preview model. GPT-4-0125-preview was successful in outputting data that met the requirements to be considered usable for NER.

When generating synthetic data, each prompt was used to generate a maximum of 100 instances. The prompt was used to initially query the large language model (LLM), requesting 5 synthetic data instances. The subsequent response was appended to the context history of the conversation. Following the initial query and response, the phrase "Create five more." was used to query the model for additional instances until 50 instances were created, each time appending the LLM's response to the context history. The process was repeated to reach a total of 100 instances before moving to the next prompt. The decision to generate instances using a prompt in two batches of 50 instead of a single batch of 100 stemmed from several observations discovered when attempting to generate 100 in a single batch. We observed that in more extended conversations, the LLM's responses drifted further from the desired domain and suffered more from repetitive patterns. We hypothesize that the increased distance from the initial prompt in more extended conversations affects the quality of subsequent responses. However, we leave a detailed investigation of this effect for future work.

Overall, we generated 1,000 synthetic instances for the "GroupName" and "Species" entity types and 200 for "CellLine". We combined the complete set of synthetic instances for each type (1,000 or 200) with their corresponding training set of authentic data to create the "combined_gen" set for experimental model #1. For the

"combined_gen_small" for experimental model #2, a randomly selected 10% of the synthetic instances were used. For experimental model #3, the complete set of synthetic instances on their own represent the "replaced_gen" training set. This was done independently for each prompting strategy. Despite being mostly structured as Python dictionaries, some very minor syntax errors required the raw LLM outputs to be further preprocessed before they were directly useable. The average cost to generate 50 samples using GPT-4-0125-preview was $0.87, representing a significant decrease in cost compared to manual collection and annotation of data.

```
Please generate 5 synthetic data samples to augment an NER dataset.

Each sample should include a list of sentence tokens and their corresponding class labels. It is
imperative that there is the exact same number of class labels as tokens in the sentence. Every
generated sentence token must have a corresponding label.

The complete set of generated samples should cover all requested entity types, but individual
samples may vary in the number of entity types included. For each sample, include at least one
token for each requested entity type. You can repeat entities within a sentence or across sentences
to ensure diversity in the data. Ensure that each label corresponds to its corresponding entity
type (e.g., PER for names of people, ORG for organizations, LOC for locations, DATE for dates). The
output should be a Python dictionary.


Here are five examples:

    {
        "sample1": {
            "tokens": [`The', `following', ..., `bedding', `.']
            "labels": [`O', `O', ..., `O', `O']
        },
        "sample2": {
            "tokens": [`Once', `mice', ..., `recovery', `.']
            "labels": [`O', `Species', ..., `O', `O']
        },
        "sample3": {
            "tokens": [`The', `labeled', ..., `)', `.']
            "labels": [`O', `O', ..., `O', `O']
        },
        "sample4": {
            "tokens": [`Plates', `were', ..., `)', `.']
            "labels": [`O', `O', ..., `O', `O']
        },
        "sample5": {
            "tokens": [`At', `the',..., `asphyxiation', `.']
            "labels": [`O', `O', ..., `O', `O']
        }
    }

Please generate 5 synthetic data samples to augment a NER dataset consisting of the method section
of scientific articles describing animal studies.

Each sample should include a list of sentence tokens and their corresponding class labels. It is
imperative that there is the exact same number of class labels as tokens in the sentence. Every
generated sentence token must have a corresponding label.

The complete set of generated samples should cover all requested entity types, but individual
samples may vary in the number of entity types included. For each sample, include at least one
token for each requested entity type. You can repeat entities within a sentence or across sentences
to ensure diversity in the data. Ensure the samples are of adequate length. Ensure that each label
corresponds to its corresponding entity type. The entities are: Species. The output should be a
Python dictionary.
```

Fig. 10. Example of a Ad-Hoc style prompt for the “Species” entity type

```
<Task Descriptions>
You are an experienced biologist, capable of easily recognizing named entities ("species names,"
"dose units," and "exposure vehicles") in a paragraph of the method section of scientific articles
describing animal studies. Specifically, your task is to perform named entity recognition and meet
the following basic requirements: 1) Provide the output as a Python dictionary with sample IDs as
keys, where each sample contains a list of tokens and their corresponding labels. Python dictionary
example:

    {
        "sample1": {
            "tokens": ["XXX", "XXX"],
            "labels": ["XXX", "XXX"]
        },
        "sample2": {
            "tokens": ["XXX", "XXX"],
            "labels": ["XXX", "XXX"]
        }
    }

2) Include at least one token for each requested entity type for each sample. 3) You can repeat
entities within a sentence or across sentences to ensure diversity in the data.

<Entity Definitions>
In biological and ecological studies, "Species" refers to a fundamental unit of classification in
the taxonomic hierarchy. It represents a group of organisms capable of interbreeding and producing
fertile offspring under natural conditions. Each species typically exhibits distinct morphological,
physiological, and behavioral characteristics, contributing to its unique identity within an
ecosystem. Within the context of scientific research involving animal subjects, "Species" denotes
the specific type or category of organisms under investigation, encompassing a variety of taxa such
as mammals, birds, reptiles, amphibians, fish, and invertebrates.

<Task Emphasis>
1) Each sample should include a list of sentence tokens and their corresponding class labels. It is
imperative that the exact same number of class labels are used as tokens in the sentence. 2) Ensure
that each label corresponds to its corresponding entity type. 3) Ensure the samples are of adequate
length. 4) The complete set of generated samples should cover all requested entity types, but
individual samples may vary in the number of entity types included.

<Task Examples>
Here are five examples:

    {
        "sample1": {
            "tokens": [`The', `following', ..., `bedding', `.']
            "labels": [`O', `O', ..., `O', `O']
        },
        "sample2": {
            "tokens": [`Once', `mice', ..., `recovery', `.']
            "labels": [`O', `Species', ..., `O', `O']
        },
        "sample3": {
            "tokens": [`The', `labeled', ..., `)', `.']
            "labels": [`O', `O', ..., `O', `O']
        },
        "sample4": {
            "tokens": [`Plates', `were', ..., `)', `.']
            "labels": [`O', `O', ..., `O', `O']
        },
        "sample5": {
            "tokens": [`At', `the',..., `asphyxiation', `.']
            "labels": [`O', `O', ..., `O', `O']
        }
    }

<Prompt>
Generate 5 synthetic data samples to augment a NER dataset comprising the method section of
scientific articles describing animal studies. Start your sample number with "sample1". The entity
to recognize is "Species".
```

Fig. 11. Example of a KEE style prompt for the “Species” entity type

CHAPTER 4

RESULTS

4.1 Results

This section presents the results of our experiments.

4.1.1 Authentic and Synthetic Entity Similarity

We used pretrained deberta-v3-large to produce contextualized embeddings for each entity in the TAC dataset. For multi-word entities, the embeddings of each word were averaged together to generate the overall entity embedding. Each embedding was a 1,024-dimensional vector; thus, it was necessary to perform dimensionality reduction to plot the embeddings. We used principal component analysis (PCA) from the Sci-Kit Learn (sklearn) Python library to reduce the 1,024-dimensional vectors into two dimensions. PCA reduces dimensionality by transforming the vectors into a new coordinate system, where the axes are the principal components. These principal components are linear combinations of the original features and are orthogonal to each other. Figure 12 plots the embedding representations of all the entities in the TAC dataset and all the synthetic entities we generated. Figure 12 presents the overall area of interest, which the subsequent sections will explore in more detail.

4.1.1.1 Species Entity Similarity

For the "Species" entity type, Figures 13, 14, 15, 16 (Appendix B) showcase the similarity between the authentic and synthetic "Species" entities. Figure 13 plots only the authentic "Species" entities from TAC, the entities marked with a black

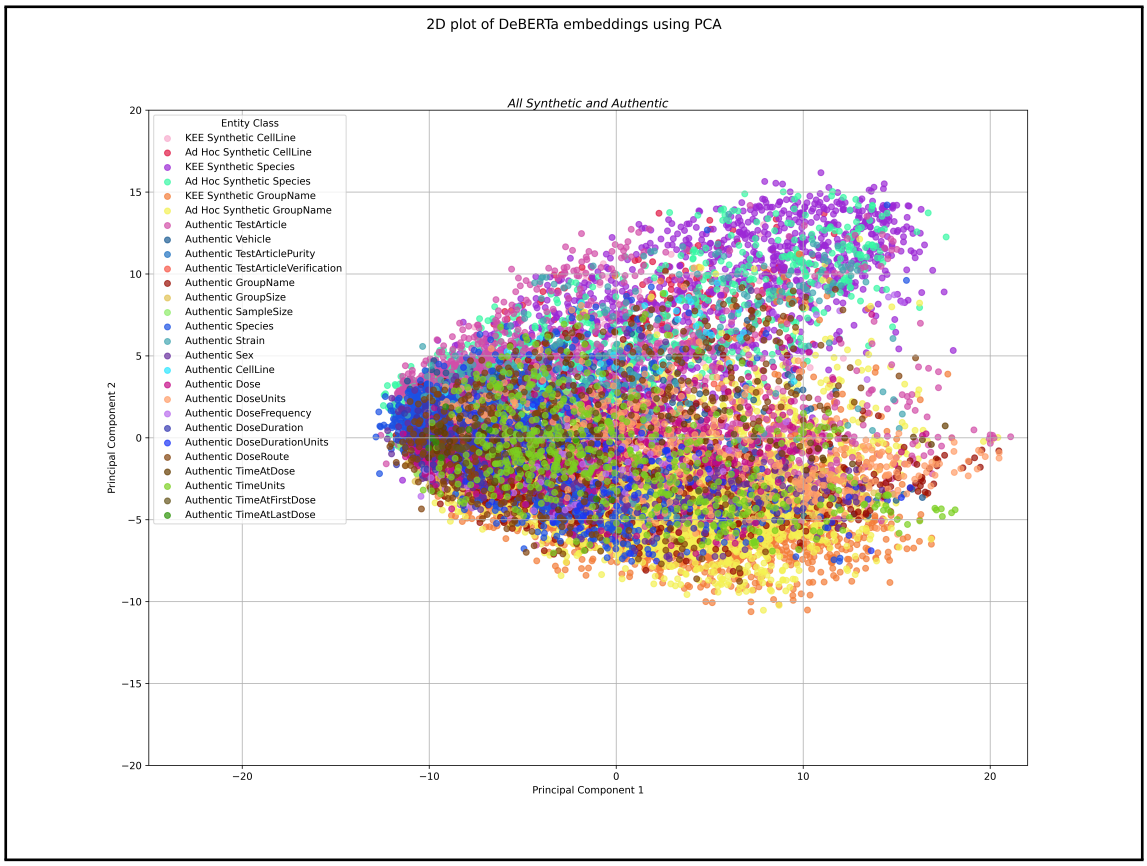

Fig. 12. Plot of the embedding representations for all authentic entities in TAC and all synthetic entities.

"X" denote those which appeared in the randomly selected exemplars. Figure 14 adds synthetic entities from the ad-hoc prompted generated data. Figure 15 adds the synthetic entities from the KEE prompted generated data. Lastly, Figure 16 adds both styles of synthetic data. With "Species", across both prompt styles, synthetic entities appear to be slightly grouped amongst the authentic entities in the left portion of the authentic data in the semantic space. Unfortunately, the synthetic data appears to stray from the general trend of the authentic entities across the entire space. Distinctly, synthetic data seems to partially overlap with the authentic data before diverging up and to the right. For the KEE synthetic data, the divergence appears

more prevalent.

4.1.1.2 CellLine Entity Similarity

For the "CellLine" entity type, Figures 17, 18, 19, 20 (Appendix B) showcase the similarity between the authentic and synthetic "CellLine" entities. Figure 17 plots only the authentic "CellLine" entities from TAC; the entities marked with a black "X" denote those that appeared in the randomly selected exemplars. Figure 18 adds the synthetic entities from the ad-hoc prompted generated data. Figure 19 adds the synthetic entities from the KEE prompted generated data. Lastly, Figure 20 adds both styles of synthetic data. Across both prompt styles, synthetic entities appear to be closely grouped amongst the authentic entities within the semantic space. Additionally, the synthetic data appears to match the general trend of the authentic entities in the entire space. Noticeably, however, along the top of the cluster, there appears to be a non-negligible number of entities that stray from the authentic data.

4.1.1.3 GroupName Entity Similarity

For the "GroupName" entity type, Figures 21, 22, 23, 24 (Appendix B) showcase the similarity between the authentic and synthetic "GroupName" entities. Figure 21 plots only the authentic "GroupName" entities from TAC, the entities marked with a black "X" denote those which appeared in the randomly selected exemplars. Figure 22 adds the ad-hoc prompted synthetic entities. Figure 23 adds the synthetic entities from the KEE prompted generated data. Lastly, Figure 24 adds both styles of synthetic data. Again, with "GroupName", synthetic entities appear to be closely grouped amongst the authentic entities across both prompt styles. The synthetic data appears to mostly match the general trend of the authentic entities in through the entire semantic space. Interestingly, the synthetic data seems to "fill in" the

middle area of the arch-shaped trend of the authentic data. Noticeably, however, in the upper right the synthetic data seems to fan out away from the authentic data.

4.1.2 NER Results on TAC

The overall results of our experiments are found in Table 6.

Table 6. Test Set Results on the filtered TAC SRIE dataset

TAC Variant	**Model**	**Class 0 ("O")**			**Class 1 ("Entity")**			**Micro F_1**	**Macro F_1**
		P	R	F_1	P	R	F_1		
Species	baseline	1.00	1.00	1.00	0.95	0.88	0.91	1.00	**0.96**
	ad_hoc_combined_gen	1.00	1.00	1.00	0.94	0.89	0.91	1.00	**0.96**
	kee_combined_gen	1.00	1.00	1.00	0.93	0.89	0.91	1.00	**0.96**
	ad_hoc_combined_gen_small	1.00	1.00	1.00	0.95	0.88	0.91	1.00	**0.96**
	kee_combined_gen_small	1.00	1.00	1.00	0.95	0.88	0.91	1.00	**0.96**
	ad_hoc_replaced_gen	1.00	0.99	0.99	0.26	0.67	0.38	0.98	0.69
	kee_replaced_gen	1.00	0.98	0.99	0.19	0.62	0.29	0.98	0.64
CellLine	baseline	1.00	1.00	1.00	1.00	0.30	0.47	1.00	0.73
	ad_hoc_combined_gen	1.00	1.00	1.00	0.42	0.43	0.43	1.00	0.71
	kee_combined_gen	1.00	1.00	1.00	0.89	0.35	0.50	1.00	0.75
	ad_hoc_combined_gen_small	1.00	1.00	1.00	1.00	0.30	0.47	1.00	0.73
	kee_combined_gen_small	1.00	1.00	1.00	1.00	0.43	0.61	1.00	**0.80**
	ad_hoc_replaced_gen	1.00	1.00	1.00	0.16	0.65	0.26	1.00	0.63
	kee_replaced_gen	1.00	0.99	0.99	0.04	0.87	0.08	0.99	0.54
GroupName	baseline	0.99	1.00	1.00	0.32	0.03	0.06	0.99	0.53
	ad_hoc_combined_gen	1.00	1.00	1.00	0.78	0.80	0.79	1.00	**0.90**
	kee_combined_gen	1.00	1.00	1.00	0.73	0.77	0.75	1.00	0.87
	ad_hoc_combined_gen_small	1.00	1.00	1.00	0.79	0.46	0.58	0.99	0.79
	kee_combined_gen_small	1.00	1.00	1.00	0.73	0.82	0.77	1.00	0.80
	ad_hoc_replaced_gen	0.99	0.99	0.99	0.06	0.09	0.07	0.98	0.53
	kee_replaced_gen	0.99	0.98	0.99	0.09	0.19	0.12	0.98	0.56

Recall that NER methods are evaluated using precision, recall, and F_1 scores. Precision is the ratio between correctly predicted mentions over the total set of predicted mentions for a specific entity. Recall is the ratio of correctly predicted mentions over the actual number of mentions. F_1 is the harmonic mean between precision and recall. The micro- and/or macro- F_1 scores are typically reported as a comparison across systems. The macro F_1 score treats all classes equally, whereas the micro

F_1 score aggregates the contributions of all classes, taking into account the inherent class imbalance of the entities in the dataset. Also, notice that the task of NER is highly imbalanced. This imbalance stems from the fact that most words or phrases are not an entity we are interested in recognizing and are subsequently classified as the non-entity type "O." As a result of the "O" classes prevalence, models tend to favor predicting "O" as opposed to other classes and generally perform exceptionally well on the "O" class.

4.1.2.1 Species Results

The "Species" results of our experiments are found in Table 6, in the upper third. With the "Species" entity type experiments, we achieved a baseline performance of 0.96 Macro-F_1. We achieved similar results (0.96 Macro-F_1) when including all synthetically generated data (*..._combined_gen*) independently for each prompting strategy with minor fluctuations in precision and recall with the "Species" entity type. We again achieved similar results (0.96 Macro-F_1) when including a random 10% of synthetically generated data (*..._combined_gen_small*) independently for each prompting strategy with no fluctuations in precision and recall, compared to the baseline, with the "Species" entity type. When we excluded all authentic data (*..._replaced_gen*), we saw a drop in performance; however, we achieved some ability to recognize species. For the ad-hoc synthetic data, a 0.69 Macro-F_1 was achieved, with 0.38 F_1 for the "Species" entity type. For the KEE synthetic data, a 0.64 Macro-F_1 was achieved, with 0.29 F_1 for the "Species" entity type. We speculate that the difference in performance between the different prompt styles is related to the quality of samples generated. The KEE "Species" entity embeddings tended to stray further from the authentic "Species" entity than the ad-hoc did, thus reducing the performance.

4.1.2.2 CellLine Results

The "CellLine" results of our experiments are found in Table 6, in the middle third. With the "CellLine" entity type experiments, we achieved a baseline performance of 0.73 Macro-F_1. When including all synthetically generated data (*..._combined_gen*) independently for each prompting strategy, we achieved 0.71 Macro-F_1 with ad-hoc generated data and 0.75 Macro-F_1 with KEE generated data. With the (*..._combined_gen*) experiments, ad-hoc achieved a 0.43 F_1 compared to 0.50 F_1 for KEE for the "CellLine" entity type. When including a random 10% of synthetically generated data (*..._combined_gen_small*) independently for each prompting strategy, we achieved 0.73 Macro-F_1 with ad-hoc generated data and 0.80 Macro-F_1 with KEE generated data. KEE_combined_gen_small (0.61 F_1) outperformed ad_hoc_combined_gen_small (0.47 F_1) on the "CellLine" entity type. Additionally, kee_combined_gen_small performed best overall across all of the CellLine experiments. We speculate that the strong performance by KEE in both (*..._combined_gen*) and (*..._combined_gen_small*) is related to the observation that KEE "CellLine" entities were clustered more closely amongst the authentic entities. Additionally, that implies that in (*..._combined_gen_small*), we were more likely to randomly select higher quality data for the 10% partition with KEE than with ad-hoc. When we excluded all authentic data (*..._replaced_gen*), we again saw a drop in performance; however, we achieved some minimal ability to recognize cell lines. For the ad-hoc synthetic data, a 0.63 Macro-F_1 was achieved, with 0.26 F_1 for the "CellLine" entity type. For the KEE synthetic data, a 0.54 Macro-F_1 was achieved, with 0.08 F_1 for the "CellLine" entity type.

4.1.2.3 GroupName Results

The "GroupName" results of our experiments are found in Table 6, in the lower third. With the "GroupName" entity type experiments, we achieved a baseline performance of 0.53 Macro-F_1. When including all synthetically generated data (*..._combined_gen*) independently for each prompting strategy, we achieved 0.90 Macro-F_1 with ad-hoc generated data and 0.87 Macro-F_1 with KEE generated data. Ad-hoc achieved a 0.79 F_1 compared to 0.75 F_1 for KEE for the "GroupName" entity type and ad_hoc_combined_gen performed the best overall for "GroupName." When including a random 10% of synthetically generated data (*..._combined_gen_small*) independently for each prompting strategy, we achieved 0.79 Macro-F_1 with ad-hoc generated data and 0.80 Macro-F_1 with KEE generated data. KEE_combined_g58_small (0.77 F_1) outperformed ad_hoc_combined_gen_small (0.47 F_1) on the "GroupName" entity type. We speculate that the synthetic data, regardless of the prompting strategy, performed well due to the observation that synthetic entities seemed to "fill in" the gaps in the entity feature space. Areas in the space more sparsely populated with authentic entities were supplemented well with synthetic data across both prompting strategies. When we excluded all authentic data (*..._replaced_gen*), we again saw a drop in performance; however, we achieved a comparable ability to recognize group names as the baseline. For the ad-hoc synthetic data, a 0.53 Macro-F_1 was achieved, with 0.007 F_1 for the "GroupName" entity type. For the KEE synthetic data, a 0.56 Macro-F_1 was achieved, with 0.12 F_1 for the "GroupName" entity type.

4.2 Conclusions

In conclusion, we can answer our five research questions.

Can synthetic data be generated to mimic the format of authentic NER training

data? We prompted multiple generative large-language models and inspected their outputs. With OpenAI's GPT-4, we found that we can generate synthetic data that mimics the format of authentic NER training data. Additionally, we found that we could generate synthetic NER samples cheaper than annotating authentic samples. We averaged $0.87 per 50 synthetic sentences.

Is synthetic data similar to the authentic data? We used a pretrained version of DeBERTa-V3-large to produce entity embeddings for all the authentic and synthetic entities. Then, after performing dimensionality reduction using PCA, we clustered and compared the authentic and synthetic representations for the three entities we investigated. We found that GPT-4 can produce synthetic data that falls within the cluster of its corresponding authentic data in semantic space.

Does the addition of synthetic data improve model performance? We studied two experimental combinations of synthetic with authentic data for each of the three entities. "Combined_gen" added all of the synthetic data to the authentic data. "Combined_gen_small" added a random 10% of the synthetic data to the authentic data. Each combination was done independently for the two different prompting strategies. We found that the models maintained or improved on the baseline model's performance across the three entities and two prompting strategies we investigated.

Is solely using synthetic data enough to achieve performance on par with a baseline? We studied an additional experimental setup of solely data for each of the three entities. "Replaced_gen" used only the synthetic data to train the model independently for the two prompting strategies. We found that the models could not achieve performance on par with the baseline models.

How do different prompting strategies for generating synthetic data affect model performance? The two strategies were compared nine times across our experiments (three experimental setups times three variants). For the seven non-baseline models in

which the two prompting strategies produced different F_1 scores, Ad-Hoc performed best three out of seven times, and KEE performed best the other four times. For the other two times, they tied. Overall, we found no single best prompting strategy.

4.3 Contribution to the field

- Determined synthetic data can be generated by GPT-4 in the proper format allowing the automated creation of labeled training data for NER.
- Evaluated Ad-Hoc and KEE Prompting.
- Evaluated the distribution of synthetic entities vs authentic entities.
- Initially evaluated the ratio of synthetic to authentic data.

4.4 Future Work

This section presents future research directions to extend the findings and contributions of this thesis, addressing both challenges and opportunities.

4.4.1 Statistical Significance Validation

We intend to validate our experimental results for statistical significance. We will repeat our experiments to collect mean, median, and standard deviation for each set of results and conduct significance testing, ensuring that the observed effects are not merely due to random chance.

4.4.2 Best 10% Selection Experiments for Generated Data

We intend to devise a sample selection method for generated synthetic data based on the entity representation clustering experiments. We propose selecting the samples containing entities most semantically similar to the entities in the authentic

data. Samples will be selected by empirically determining a distance or an amount of similarity from the centroid of the exemplars or alternatively all available authentic data. Then, we propose exploring the effectiveness of selecting the best 10% of synthetic data for model training as opposed to a random 10%. By focusing only on the highest-quality synthetic samples, we hope to further improve model performance and generalization. We plan to conduct experiments to compare different data selection strategies and evaluate their impact on model learning.

4.4.3 Alternative Oversampling Techniques

We intend to compare synthetic data generation to oversampling techniques for NER to better determine the extent to which the observed performance gains from our experiments should be attributed to simply having more data versus the novel synthetic data supplementation.

4.4.4 Ablation Study on Quantity of Synthetic Data

We intend to conduct an ablation study on the amount of synthetic data required.

4.4.5 Decreasing Semantic Drift and Increasing Diversity

We intend to determine if prompting in even smaller batches further reduces semantic drift. We also intend to adjust the generative model's temperature to increase the diversity of generated samples.

4.4.6 Implementation of Continual Learning

Finally, we intend to implement a continual learning system for NER that incorporates the novel synthetic data as a generative replay-based approach.

Appendix A

ABBREVIATIONS

A-BERT	Adapter BERT	[44]
A-GEM	Averaged Gradient Episodic Memory	[21]
Adapters Created	Creation of adapter models in NLP	[44]
AddNER	Additive variant of student teacher NER	[4]
B-CL	BERT-based Continual Learning	[45]
BERT	Bidirectional Encoder Representations from Transformers	[48]
BIO	Inside, Outside, Beginning labeling framework for NER	
BiR	Brain Inspired Replay	[26]
BiLSTM	Bidirectional Long Short Term Memory Unit	
BRAT	Web-based tool for text annotation	
CFNER	Causal Framework for Continual NER	[36]
CIL	Class-incremental learning	
CL Founded	The idea of catastrophic forgetting is first presented	[7]
CLA	Continual Learning Adapter	[44]
CLASSIC	Continual and contrastive Learning for ASpect SentIment Classification	[46]
CoNLL-03	2003 Conference on Natural Language Learning	
CoPE	Continual Prototype Evolution	[28]
CPFD	Confidence-based pseudo-labeling and Pooled Features Distillation	[9]
CRF	Conditional Random Field	
CTR	Capsules and Transfer Routing	[47]
DAnish	North Germanic language of Danish	
DGR	Deep Generative Replay	[22]

DLD	Decomposing Logits Distillation	[37]
DnR	Distill and Replay	[25]
ELLA	Efficient Lifelong Learning Algorithm	[29]
ER	Experience Replay	[23]
ET	Entity Typing	
EWC	Elastic Weight Consolidation	[32]
ExtendNER	Extension variant of student teacher NER	[4]
FSA	First Session Adaptation	[49]
GAN	Generative Adversarial Network	
GEM	Gradient Episodic Memory	[20]
GPT	Generative Pretrained Transformer	
HPRC	VCU High Performance Research Computing	
iCaRL	Incremental Classifier and Representation Learning	[19]
IOB	Inside, Outside, Beginning labeling framework for NER	
JCBIE	Joint Continual Learning for Biomedical Information Extraction	[59]
KD	Knowledge Distillation	[35]
KEE	Knowledge Entity Extraction	[81]
KL	Kullback Leibler Divergence	
LAMOL	LAnguage MOdelling for Lifelong Language Learning	[24]
LaR	Learn and Review	[27]
LFPT5	Lifelong Few-shot Learning with Prompt Tuning of T5	[50]
LLM	Large Language Model	
LOC	Location entity type	

LoLM	Learning O Helps For Learning More	[8]
LSTM	Long-Short Term Memory Unit	
LwF	Learning Without Forgetting	[6]
MAS	Memory Aware Synapses	[34]
NER	Named Entity Recognition	
NERDA	Named Entity Recognition for DAnish	[71]
NIEHS	National Institute of Environmental Health Sciences	
NLP	Natural Language Processing	
NTP	National Toxicology Program	
ORG	Organization entity type	
PER	Person entity type	
PNN	Progressive Neural Networks	[38]
QAES	Quadratic Approximation to the Energy Surface	[28]
R-EWC	Rotated Elastic Weight Consolidation	[33]
RDP	Relation Distillation and Prototypical pseudo label	[62]
RNC	Routing Networks for Continual Language Learning	[43]
RTD	Replaced token detection	
SI	Synaptic Intelligence	[30]
SKD-NER	Span-based Knowledge Distillation for NER	[64]
SP	Span Detection	
SPA	Selective Parameter Updates	[51]
SpanKL	Span-based Knowledge Distillation	[65]
SRIE	Systematic Review Information Extraction	
ST-NER	Student-Teacher NER	[4]

TAC	Text Analysis Conference	
Task-CL	Task incremental Continual Learning	
TEM	Tiny Episodic Memory	[83]
TIL	Task incremental Continual Learning	
TST	**T**ask-CL based on **S**ub-networks and **T**ask similarity	[66]
VCU	Virginia Commonwealth University	

Appendix B

ENTITY EMBEDDING PLOTS

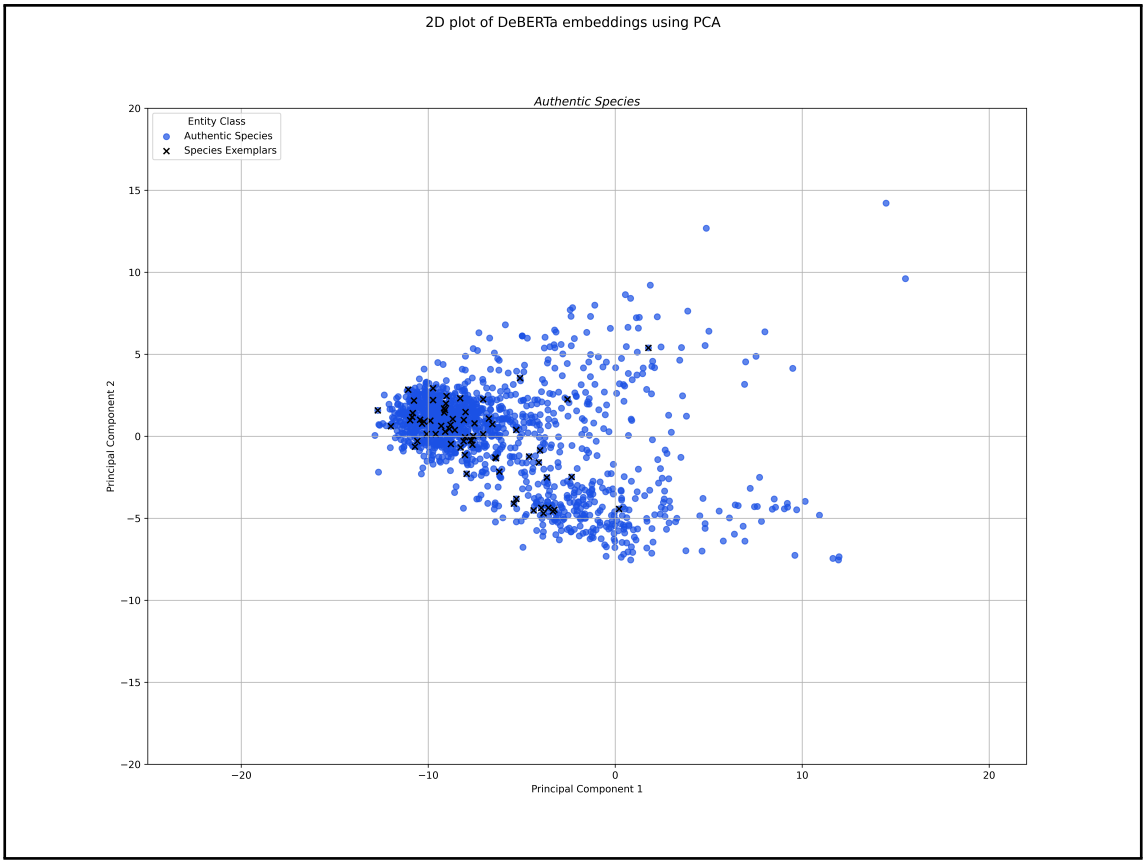

Fig. 13. Plot of the embedding representations for all authentic Species entities in TAC.

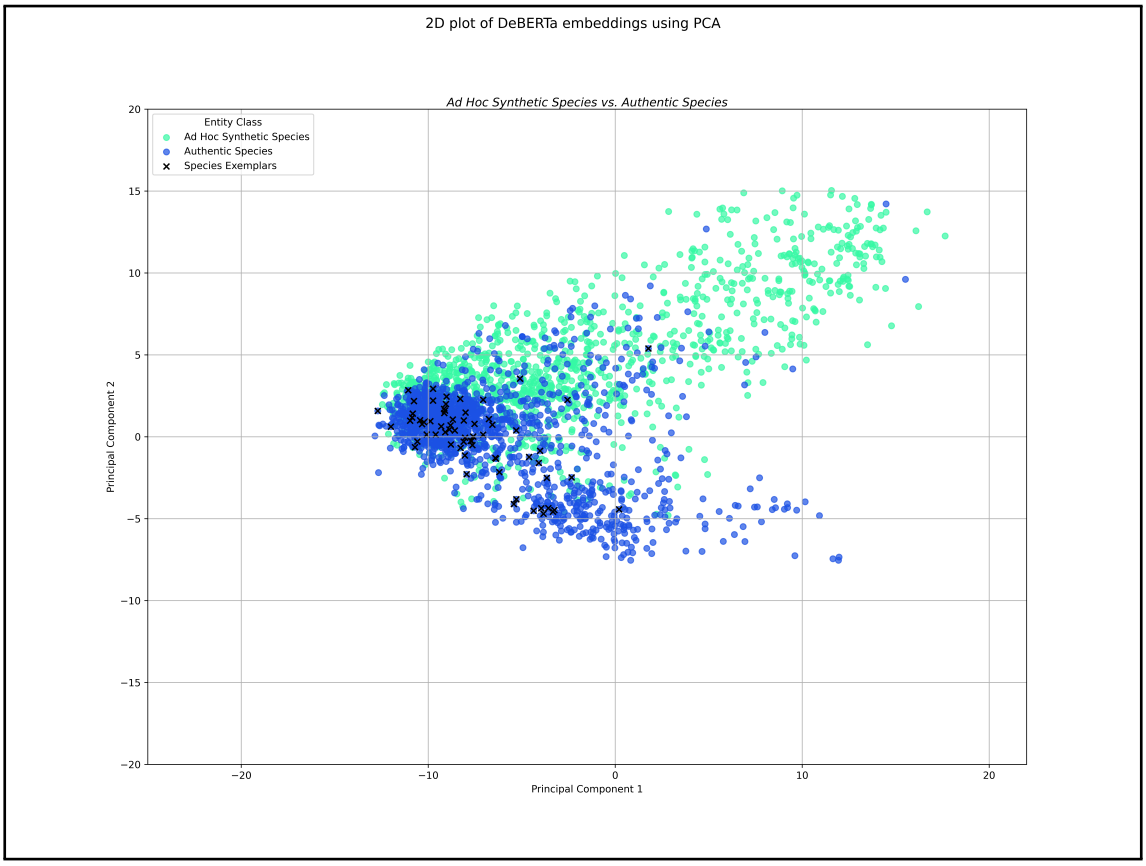

Fig. 14. Plot of the embedding representations for all authentic Species entities in TAC and ad-hoc Species synthetic entities.

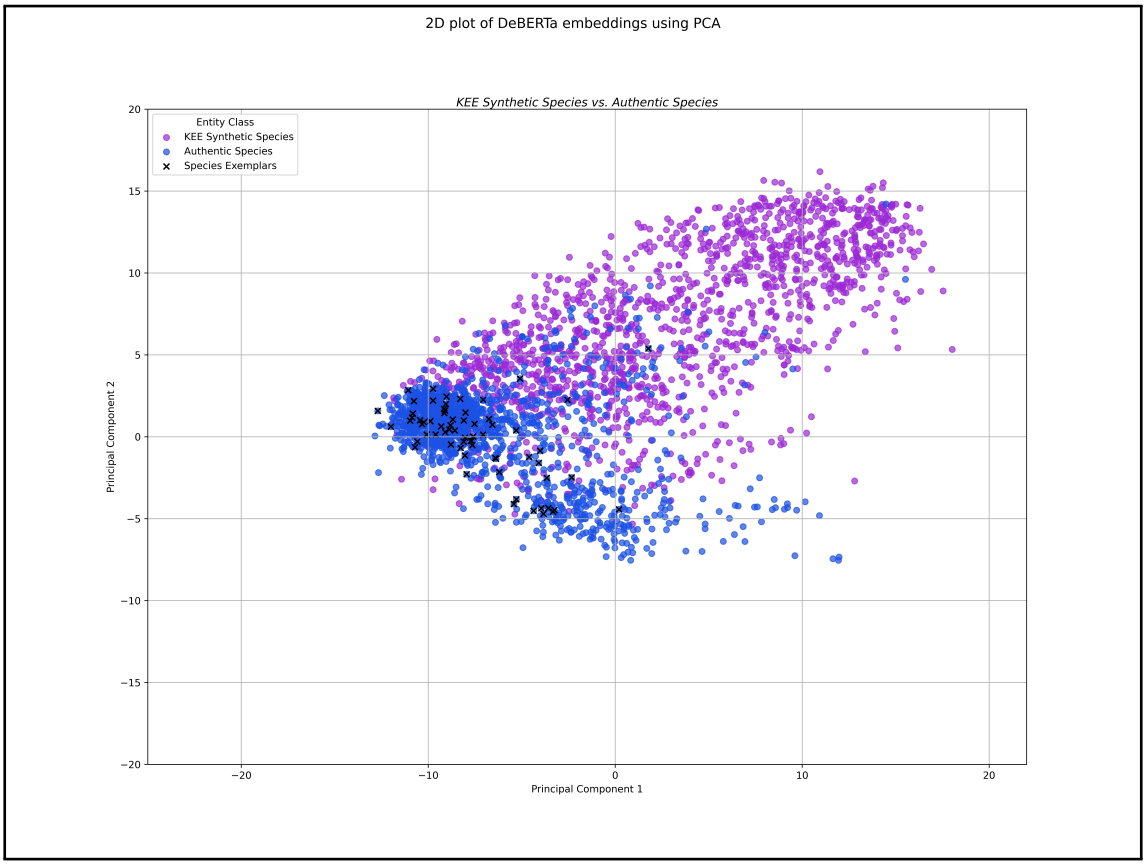

Fig. 15. Plot of the embedding representations for all authentic Species entities in TAC and KEE Species synthetic entities.

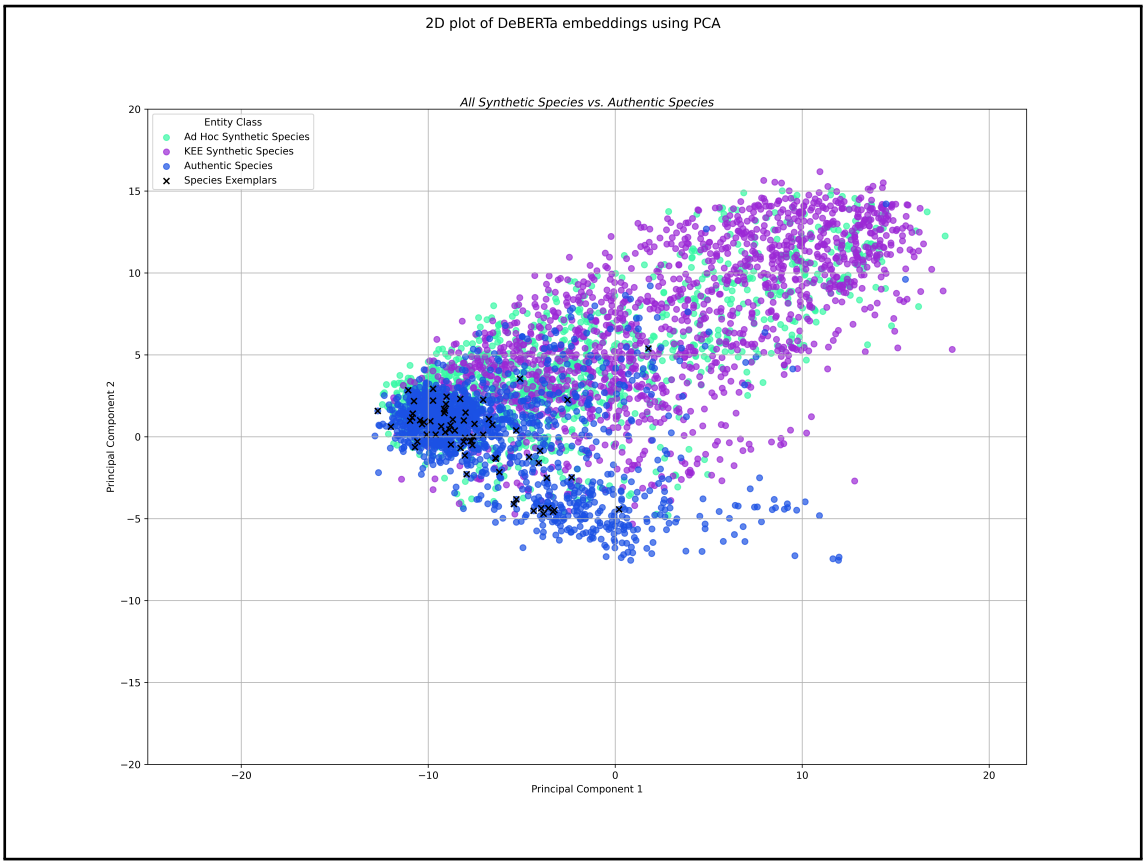

Fig. 16. Plot of the embedding representations for all authentic Species entities in TAC and all Species synthetic entities.

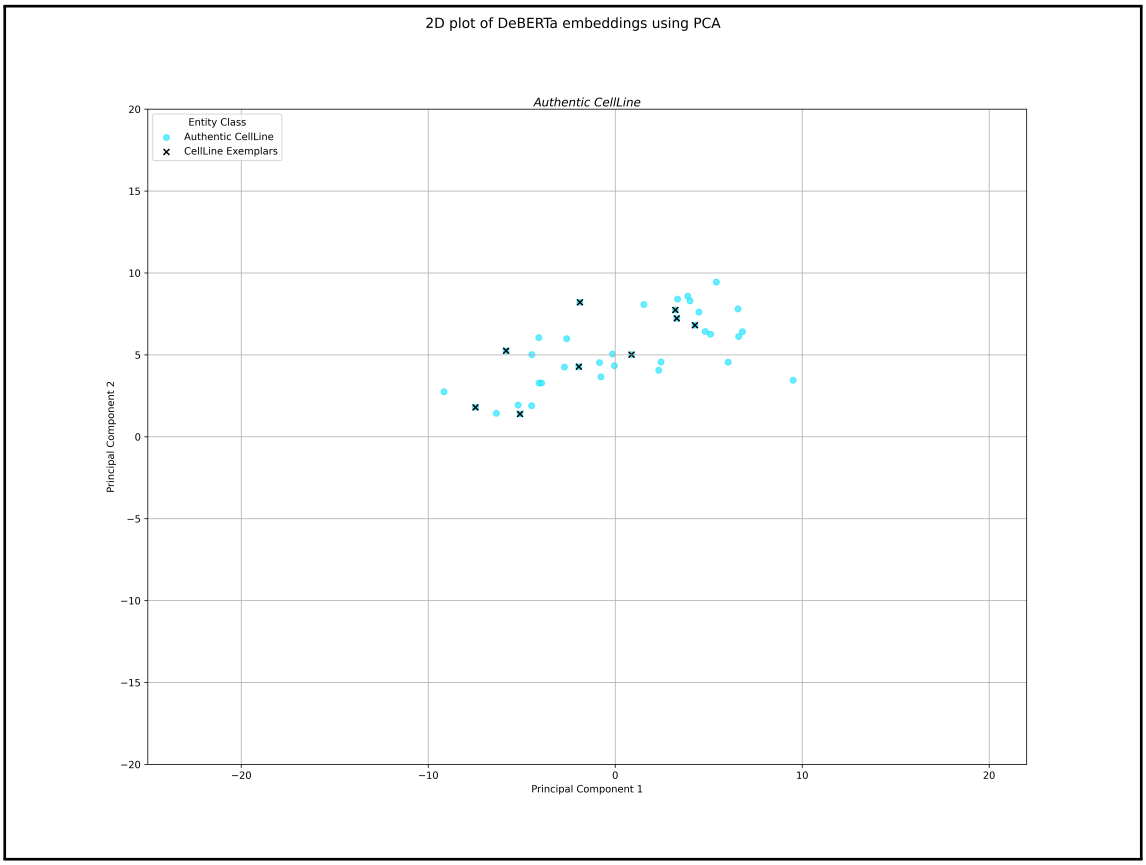

Fig. 17. Plot of the embedding representations for all authentic CellLine entities in TAC.

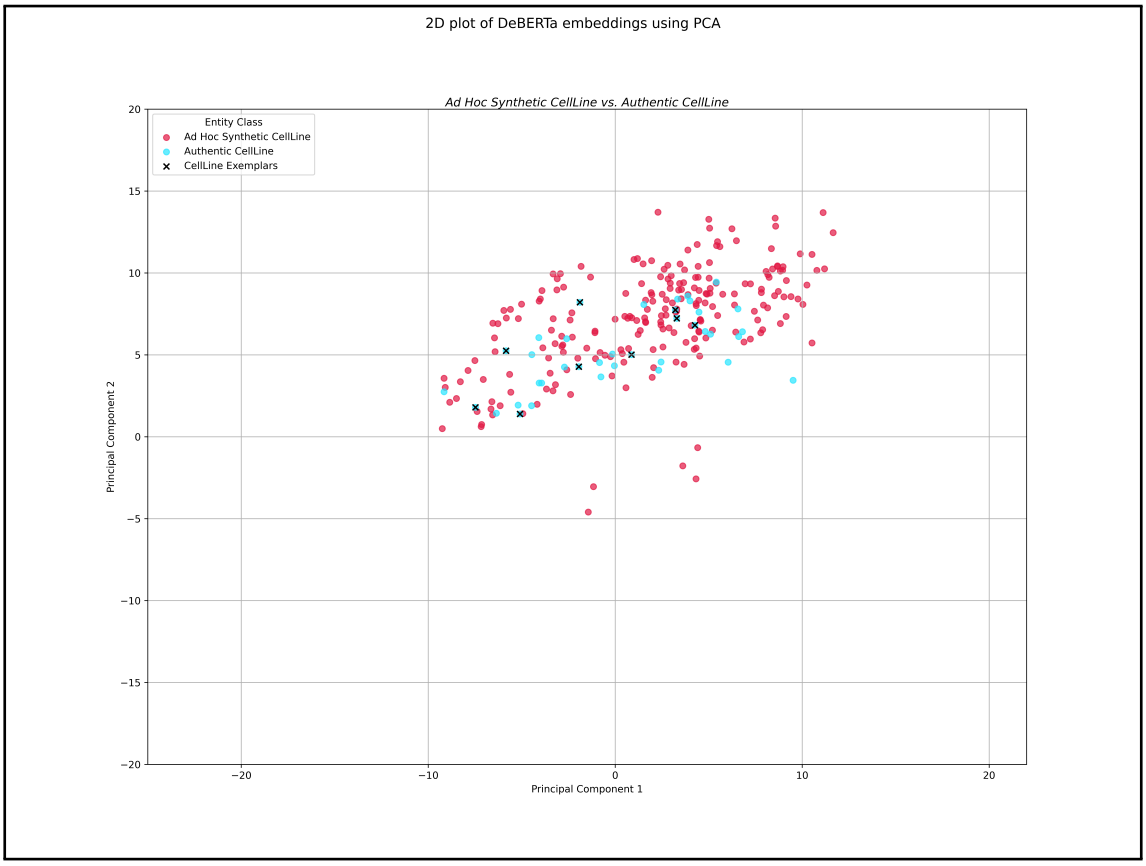

Fig. 18. Plot of the embedding representations for all authentic CellLine entities in TAC and ad-hoc CellLine synthetic entities.

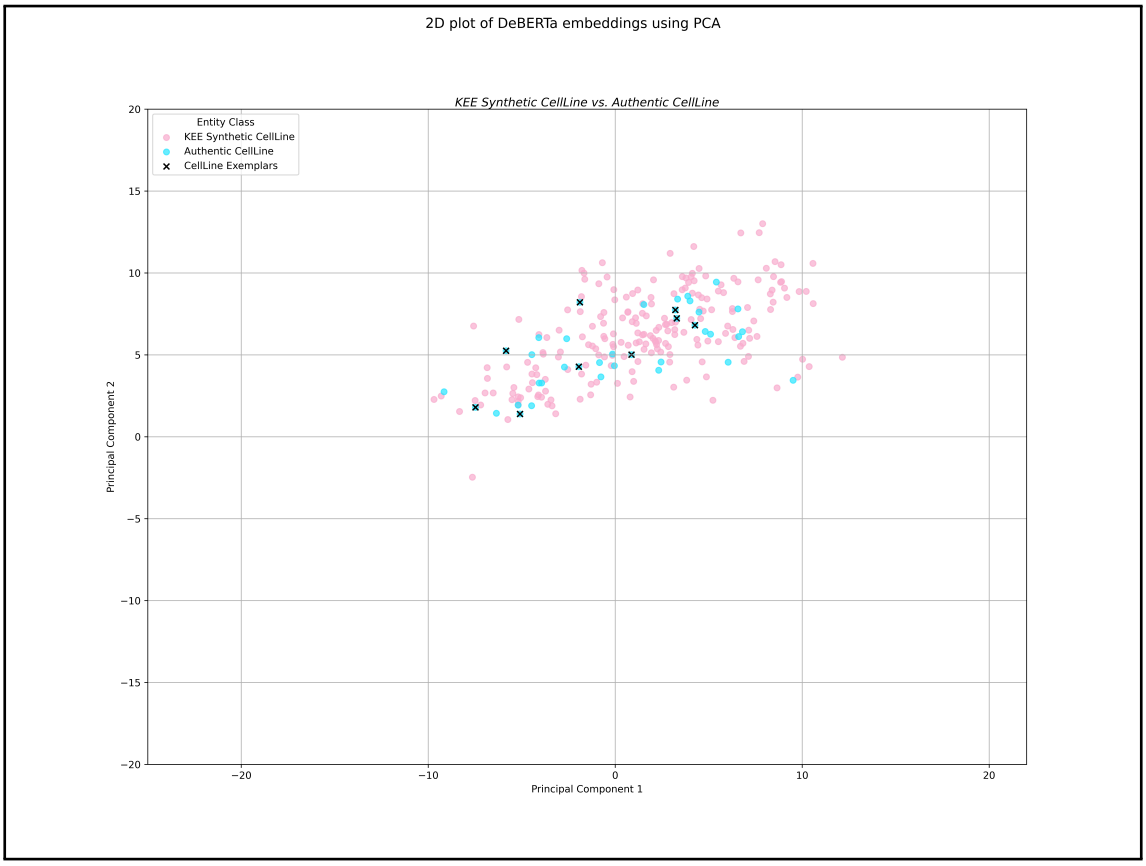

Fig. 19. Plot of the embedding representations for all authentic CellLine entities in TAC and KEE CellLine synthetic entities.

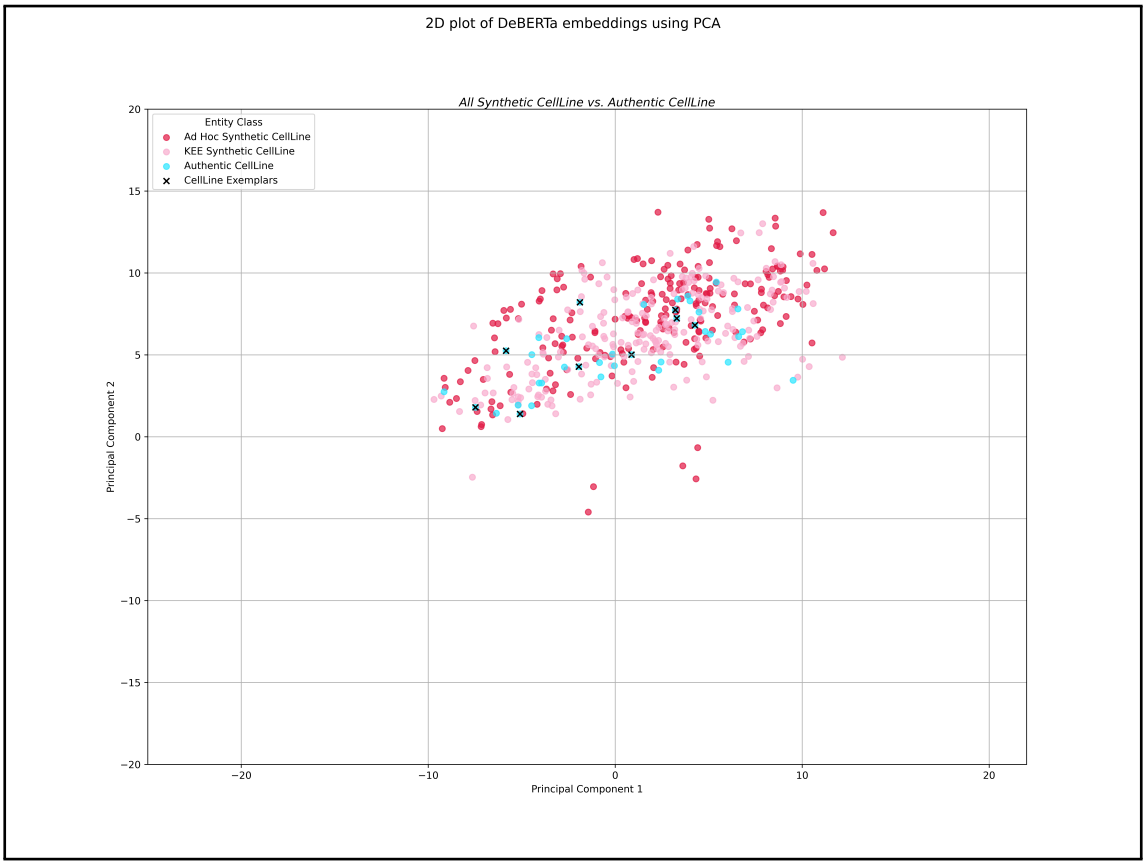

Fig. 20. Plot of the embedding representations for all authentic CellLine entities in TAC and all synthetic CellLine entities.

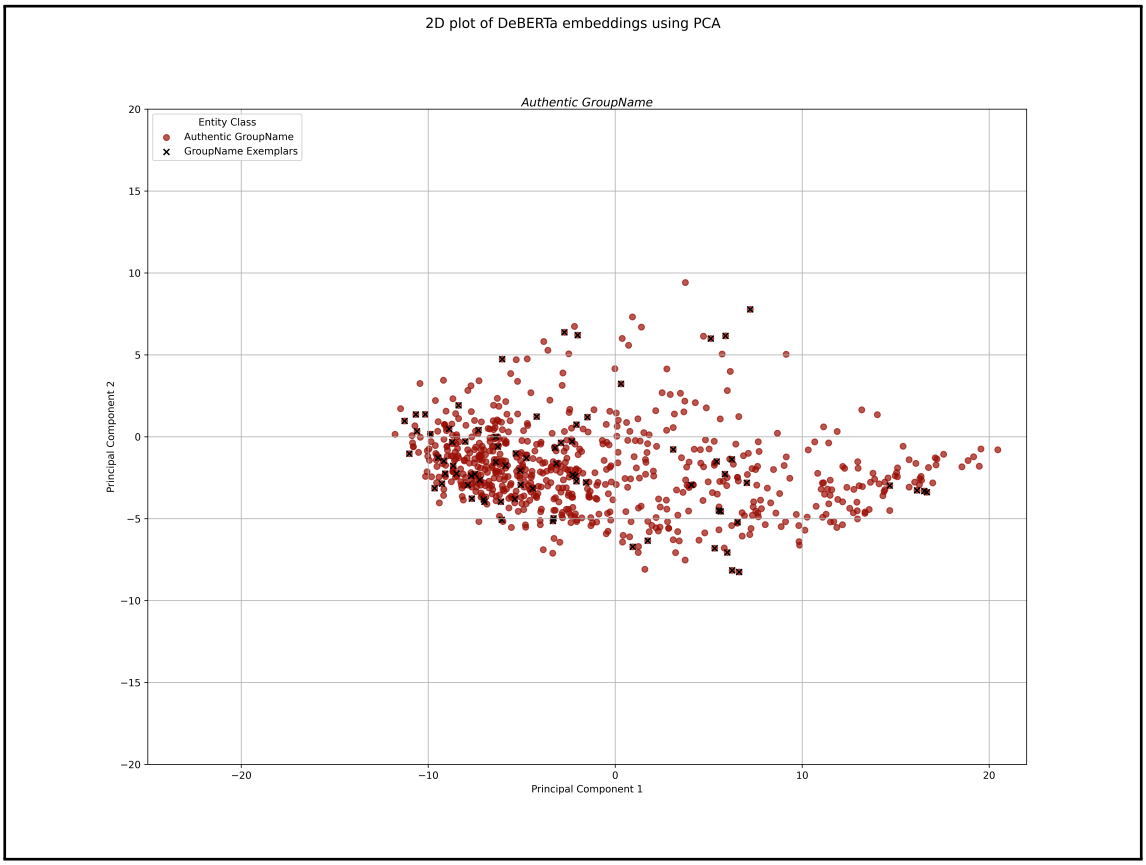

Fig. 21. Plot of the embedding representations for all authentic GroupName entities in TAC.

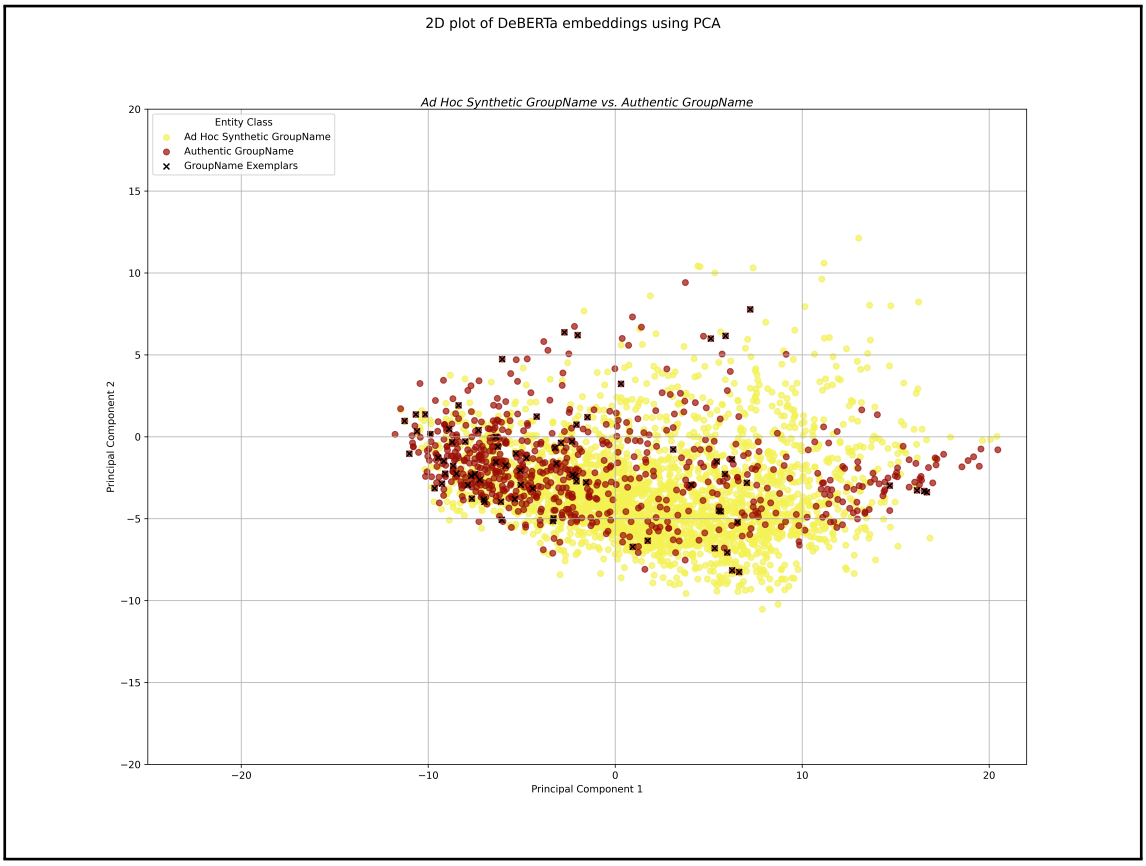

Fig. 22. Plot of the embedding representations for all authentic GroupName entities in TAC and ad-hoc GroupName synthetic entities.

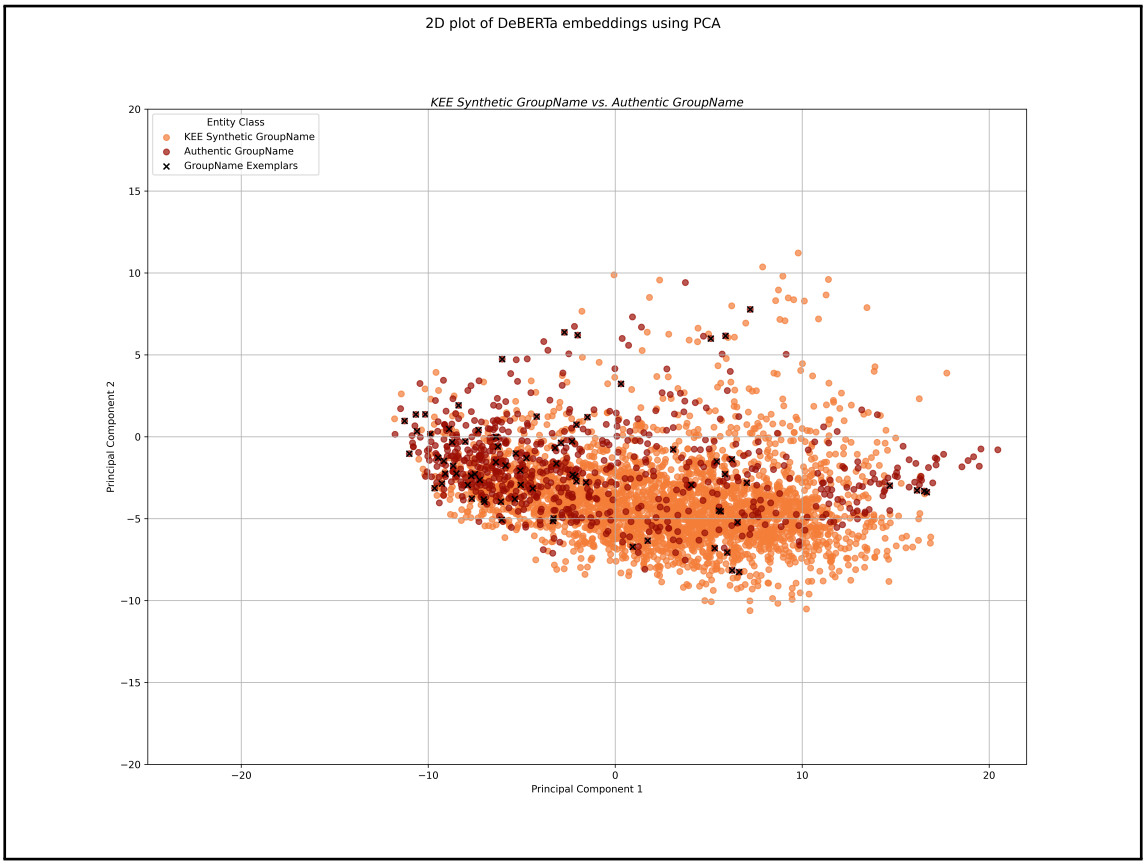

Fig. 23. Plot of the embedding representations for all authentic GroupName entities in TAC and KEE GroupName synthetic entities.

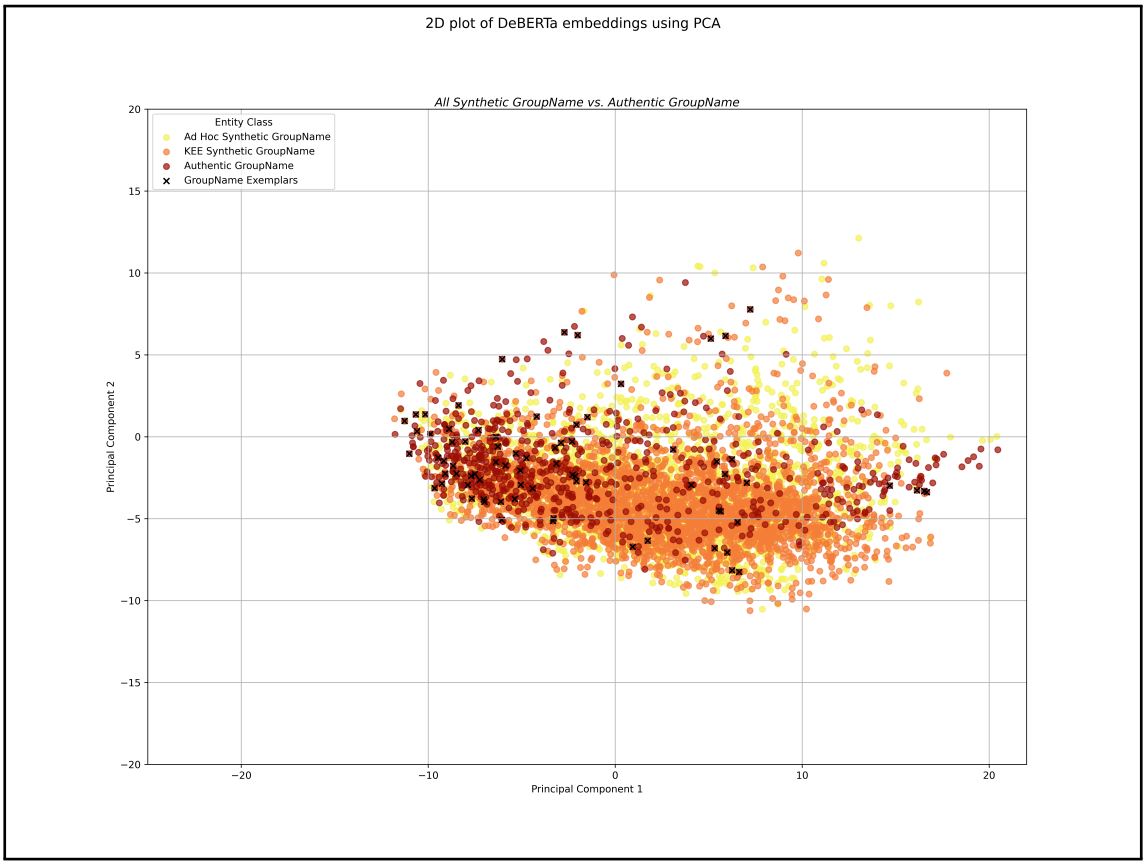

Fig. 24. Plot of the embedding representations for all authentic GroupName entities in TAC and all GroupName synthetic entities.

REFERENCES

[1] David Nadeau and Satoshi Sekine. "A Survey of Named Entity Recognition and Classification". In: *Lingvisticæ Investigationes* 30.1 (Jan. 2007), pp. 3–26. ISSN: 0378-4169, 1569-9927. DOI: 10.1075/li.30.1.03nad. (Visited on 12/08/2023).

[2] Jing Li et al. "A Survey on Deep Learning for Named Entity Recognition". In: *IEEE Transactions on Knowledge and Data Engineering* 34.1 (Jan. 2022), pp. 50–70. ISSN: 1558-2191. DOI: 10.1109/TKDE.2020.2981314. (Visited on 12/08/2023).

[3] Matthias De Lange et al. "A Continual Learning Survey: Defying Forgetting in Classification Tasks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.7 (July 2022), pp. 3366–3385. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2021.3057446.

[4] Natawut Monaikul et al. "Continual Learning for Named Entity Recognition". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. Palo Alto, CA, USA: AAAI Press, May 2021, pp. 13570–13577. DOI: 10.1609/aaai.v35i15.17600. (Visited on 08/26/2023).

[5] Marc Masana et al. "Class-Incremental Learning: Survey and Performance Evaluation on Image Classification". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.5 (May 2023), pp. 5513–5533. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2022.3213473. (Visited on 12/08/2023).

[6] Zhizhong Li and Derek Hoiem. "Learning Without Forgetting". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Lecture Notes in Com-

puter Science. Amsterdam, The Netherlands: Springer International Publishing, 2016, pp. 614–629. ISBN: 978-3-319-46493-0. DOI: 10.1007/978-3-319-46493-0_37.

[7] Michael McCloskey and Neal J. Cohen. "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem". In: *Psychology of Learning and Motivation* 24 (Jan. 1989). Ed. by Gordon H. Bower, pp. 109–165. DOI: 10.1016/S0079-7421(08)60536-8. (Visited on 10/10/2023).

[8] Ruotian Ma et al. *Learning "O" Helps for Learning More: Handling the Concealed Entity Problem for Class-incremental NER*. July 2023. DOI: 10.48550/arXiv.2210.04676. arXiv: 2210.04676 [cs]. (Visited on 12/20/2023).

[9] Duzhen Zhang et al. "Continual Named Entity Recognition without Catastrophic Forgetting". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 8186–8197. DOI: 10.18653/v1/2023.emnlp-main.509. (Visited on 03/06/2024).

[10] Timothée Lesort et al. "Continual Learning for Robotics: Definition, Framework, Learning Strategies, Opportunities and Challenges". In: *Information Fusion* 58 (June 2020), pp. 52–68. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2019.12.004. (Visited on 11/12/2023).

[11] German I. Parisi et al. "Continual Lifelong Learning with Neural Networks: A Review". In: *Neural Networks* 113 (May 2019), pp. 54–71. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2019.01.012. (Visited on 11/13/2023).

[12] Andrea Madotto et al. "Continual Learning in Task-Oriented Dialogue Systems". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural*

Language Processing. Ed. by Marie-Francine Moens et al. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 7452–7467. DOI: 10.18653/v1/2021.emnlp-main.590. URL: https://aclanthology.org/2021.emnlp-main.590 (visited on 12/06/2023).

[13] Liyuan Wang et al. *A Comprehensive Survey of Continual Learning: Theory, Method and Application*. June 2023. DOI: 10.48550/arXiv.2302.00487. arXiv: 2302.00487 [cs]. (Visited on 09/21/2023).

[14] Magdalena Biesialska, Katarzyna Biesialska, and Marta R. Costa-jussà. "Continual Lifelong Learning in Natural Language Processing: A Survey". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 6523–6541. DOI: 10.18653/v1/2020.coling-main.574. (Visited on 09/08/2023).

[15] Zixuan Ke and Bing Liu. *Continual Learning of Natural Language Processing Tasks: A Survey*. May 2023. DOI: 10.48550/arXiv.2211.12701. arXiv: 2211.12701 [cs]. (Visited on 12/20/2023).

[16] Robert M. French. "Semi-Distributed Representations and Catastrophic Forgetting in Connectionist Networks". In: *Connection Science* 4.3-4 (Jan. 1992), pp. 365–377. ISSN: 0954-0091. DOI: 10.1080/09540099208946624. (Visited on 12/06/2023).

[17] Anthony Robins. "Catastrophic Forgetting, Rehearsal and Pseudorehearsal". In: *Connection Science* 7.2 (June 1995), pp. 123–146. ISSN: 0954-0091. DOI: 10.1080/09540099550039318. (Visited on 12/06/2023).

[18] Anthony Robins and Simon McCallum. "Catastrophic Forgetting and the Pseudorehearsal Solution in Hopfield-type Networks". In: *Connection Science* 10.2

(June 1998), pp. 121–135. ISSN: 0954-0091. DOI: 10.1080/095400998116530. (Visited on 12/06/2023).

[19] Sylvestre-Alvise Rebuffi et al. "iCaRL: Incremental Classifier and Representation Learning". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA: IEEE Computer Society, July 2017, pp. 5533–5542. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.587. (Visited on 03/05/2024).

[20] David Lopez-Paz and Marc' Aurelio Ranzato. "Gradient Episodic Memory for Continual Learning". In: *Advances in Neural Information Processing Systems*. Vol. 30. Long Beach, CA, USA: Curran Associates, Inc., 2017, pp. 6467–6476. (Visited on 12/06/2023).

[21] Arslan Chaudhry et al. "Efficient Lifelong Learning with A-GEM". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. New Orleans, LA, USA: OpenReview.net, 2019.

[22] Hanul Shin et al. "Continual Learning with Deep Generative Replay". In: *Advances in Neural Information Processing Systems*. Vol. 30. Long Beach, CA, USA: Curran Associates, Inc., Dec. 2017, pp. 2991–3000. (Visited on 10/09/2023).

[23] Mohammad Rostami, Soheil Kolouri, and Praveen K. Pilly. "Complementary Learning for Overcoming Catastrophic Forgetting Using Experience Replay". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. Macao, China: International Joint Conferences on Artificial Intelligence Organization, Aug. 2019, pp. 3339–3345. ISBN: 978-0-9992411-4-1. DOI: 10.24963/ijcai.2019/463. (Visited on 12/06/2023).

[24] Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. *LAMOL: LAnguage MOdeling for Lifelong Language Learning*. Dec. 2019. DOI: 10.48550/arXiv.1909.03329. arXiv: 1909.03329 [cs]. (Visited on 12/04/2023).

[25] Jingyuan Sun et al. "Distill and Replay for Continual Language Learning". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Ed. by Donia Scott, Nuria Bel, and Chengqing Zong. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 3569–3579. DOI: 10.18653/v1/2020.coling-main.318. (Visited on 12/06/2023).

[26] Gido M. van de Ven, Hava T. Siegelmann, and Andreas S. Tolias. "Brain-Inspired Replay for Continual Learning with Artificial Neural Networks". In: *Nature Communications* 11.1 (Aug. 2020), p. 4069. ISSN: 2041-1723. DOI: 10.1038/s41467-020-17866-2. (Visited on 08/25/2023).

[27] Yu Xia et al. "Learn and Review: Enhancing Continual Named Entity Recognition via Reviewing Synthetic Samples". In: *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, 2022, pp. 2291–2300. DOI: 10.18653/v1/2022.findings-acl.179. (Visited on 07/26/2023).

[28] Robert M. French and Nick Chater. "Using Noise to Compute Error Surfaces in Connectionist Networks: A Novel Means of Reducing Catastrophic Forgetting". In: *Neural Computation* 14.7 (July 2002), pp. 1755–1769. ISSN: 0899-7667. DOI: 10.1162/08997660260028700.

[29] Paul Ruvolo and Eric Eaton. "ELLA: An Efficient Lifelong Learning Algorithm". In: *Proceedings of the 30th International Conference on Machine Learning*. Atlanta, GA, USA: PMLR, Feb. 2013, pp. 507–515. (Visited on 12/06/2023).

[30] Friedemann Zenke, Ben Poole, and Surya Ganguli. "Continual Learning Through Synaptic Intelligence". In: *Proceedings of the 34th International Conference on Machine Learning*. Sydney, Austrailia: PMLR, July 2017, pp. 3987–3995. (Visited on 10/09/2023).

[31] Sang-Woo Lee et al. "Overcoming Catastrophic Forgetting by Incremental Moment Matching". In: *Advances in Neural Information Processing Systems*. Vol. 30. Long Beach, CA, USA: Curran Associates, Inc., 2017, pp. 4652–4662. (Visited on 10/11/2023).

[32] James Kirkpatrick et al. "Overcoming Catastrophic Forgetting in Neural Networks". In: *Proceedings of the National Academy of Sciences of the United States of America* 114.13 (Mar. 2017), pp. 3521–3526. ISSN: 1091-6490. DOI: 10.1073/pnas.1611835114.

[33] Xialei Liu et al. "Rotate Your Networks: Better Weight Consolidation and Less Catastrophic Forgetting". In: *2018 24th International Conference on Pattern Recognition (ICPR)*. Los Alamitos, CA, USA: IEEE Computer Society, Aug. 2018, pp. 2262–2268. DOI: 10.1109/ICPR.2018.8545895. (Visited on 12/06/2023).

[34] Rahaf Aljundi et al. "Memory Aware Synapses: Learning What (Not) to Forget". In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari et al. Vol. 3. Munich, Germany: Springer International Publishing, 2018, pp. 144–161. ISBN: 978-3-030-01219-9. DOI: 10.1007/978-3-030-01219-9_9.

[35] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. Mar. 2015. DOI: 10.48550/arXiv.1503.02531. arXiv: 1503.02531 [cs, stat]. (Visited on 08/29/2023).

[36] Junhao Zheng et al. “Distilling Causal Effect from Miscellaneous Other-Class for Continual Named Entity Recognition”. In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 3602–3615. DOI: 10.18653/v1/2022.emnlp-main.236. (Visited on 09/08/2023).

[37] Duzhen Zhang et al. “Decomposing Logits Distillation for Incremental Named Entity Recognition”. In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Taipei Taiwan: ACM, July 2023, pp. 1919–1923. ISBN: 978-1-4503-9408-6. DOI: 10.1145/3539618.3591970. (Visited on 09/21/2023).

[38] Andrei A. Rusu et al. *Progressive Neural Networks*. June 2016. DOI: 10.48550/arXiv.1606.04671. arXiv: 1606.04671 [cs]. (Visited on 12/08/2023).

[39] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. “Expert Gate: Lifelong Learning with a Network of Experts”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA: IEEE Computer Society, July 2017, pp. 7120–7129. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.753. (Visited on 12/06/2023).

[40] Chrisantha Fernando et al. *PathNet: Evolution Channels Gradient Descent in Super Neural Networks*. Jan. 2017. DOI: 10.48550/arXiv.1701.08734. arXiv: 1701.08734 [cs]. (Visited on 12/04/2023).

[41] Arun Mallya and Svetlana Lazebnik. “PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Salt Lake City, UT: IEEE

Computer Society, June 2018, pp. 7765–7773. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00810. (Visited on 12/06/2023).

[42] Joan Serra et al. "Overcoming Catastrophic Forgetting with Hard Attention to the Task". In: *Proceedings of the 35th International Conference on Machine Learning*. Stockholm, Sweden: PMLR, July 2018, pp. 4548–4557. (Visited on 12/04/2023).

[43] Mark Collier et al. *Routing Networks with Co-training for Continual Learning*. Sept. 2020. DOI: 10.48550/arXiv.2009.04381. arXiv: 2009.04381 [cs, stat]. (Visited on 02/01/2024).

[44] Neil Houlsby et al. "Parameter-Efficient Transfer Learning for NLP". In: *Proceedings of the 36th International Conference on Machine Learning*. Long Beach, CA, USA: PMLR, May 2019, pp. 2790–2799. (Visited on 09/21/2023).

[45] Zixuan Ke, Hu Xu, and Bing Liu. "Adapting BERT for Continual Learning of a Sequence of Aspect Sentiment Classification Tasks". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Ed. by Kristina Toutanova et al. Online: Association for Computational Linguistics, June 2021, pp. 4746–4755. DOI: 10.18653/v1/2021.naacl-main.378. (Visited on 12/20/2023).

[46] Zixuan Ke et al. "CLASSIC: Continual and Contrastive Learning of Aspect Sentiment Classification Tasks". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 6871–6883. DOI: 10.18653/v1/2021.emnlp-main.550. (Visited on 09/21/2023).

[47] Zixuan Ke et al. “Achieving Forgetting Prevention and Knowledge Transfer in Continual Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Online: Curran Associates, Inc., 2021, pp. 22443–22456.

[48] Jacob Devlin et al. “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)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, MN, USA: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. (Visited on 12/04/2023).

[49] Aristeidis Panos et al. “First Session Adaptation: A Strong Replay-Free Baseline for Class-Incremental Learning”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. Paris, France: IEEE Computer Society, 2023, pp. 18820–18830. (Visited on 03/06/2024).

[50] Chengwei Qin and Shafiq Joty. *LFPT5: A Unified Framework for Lifelong Few-shot Language Learning Based on Prompt Tuning of T5*. Mar. 2022. DOI: 10.48550/arXiv.2110.07298. arXiv: 2110.07298 [cs]. (Visited on 11/29/2023).

[51] Wenxuan Zhang et al. *Overcoming General Knowledge Loss with Selective Parameter Update*. Oct. 2023. DOI: 10.48550/arXiv.2308.12462. arXiv: 2308.12462 [cs]. (Visited on 10/15/2023).

[52] Arslan Chaudhry et al. “Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence”. In: *Computer Vision – ECCV 2018*. Ed. by Martial Hebert et al. Vol. 11. Munich, Germany: Springer International

Publishing, 2018, pp. 556–572. ISBN: 978-3-030-01252-6. DOI: 10.1007/978-3-030-01252-6_33.

[53] Dani Yogatama et al. *Learning and Evaluating General Linguistic Intelligence*. Jan. 2019. DOI: 10.48550/arXiv.1901.11373. arXiv: 1901.11373 [cs, stat]. (Visited on 02/17/2024).

[54] Ronald Kemker et al. "Measuring Catastrophic Forgetting in Neural Networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. New Orleans, LA, USA: AAAI Press, Apr. 2018, pp. 3390–3398. DOI: 10.1609/aaai.v32i1.11651.

[55] Martial Mermillod, Aurélia Bugaiska, and Patrick BONIN. "The Stability-Plasticity Dilemma: Investigating the Continuum from Catastrophic Forgetting to Age-Limited Learning Effects". In: *Frontiers in Psychology* 4 (2013). ISSN: 1664-1078. DOI: 10.3389/fpsyg.2013.00504.

[56] Lingzhen Chen and Alessandro Moschitti. "Learning to Progressively Recognize New Named Entities with Sequence to Sequence Models". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, NM, USA: Association for Computational Linguistics, Aug. 2018, pp. 2181–2191. (Visited on 08/26/2023).

[57] Supriti Vijay and Aman Priyanshu. *NERDA-Con: Extending NER Models for Continual Learning – Integrating Distinct Tasks and Updating Distribution Shifts*. June 2022. DOI: 10.48550/arXiv.2206.14607. arXiv: 2206.14607 [cs]. (Visited on 09/08/2023).

[58] Rui Wang et al. "Few-Shot Class-Incremental Learning for Named Entity Recognition". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Asso-

ciation for Computational Linguistics, May 2022, pp. 571–582. DOI: 10.18653/v1/2022.acl-long.43. (Visited on 09/08/2023).

[59] Kai He et al. "JCBIE: A Joint Continual Learning Neural Network for Biomedical Information Extraction". In: *BMC Bioinformatics* 23.1 (Dec. 2022), p. 549. ISSN: 1471-2105. DOI: 10.1186/s12859-022-05096-w. (Visited on 11/10/2023).

[60] Justin Payan et al. "Towards Realistic Single-Task Continuous Learning Research for NER". In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3773–3783. DOI: 10.18653/v1/2021.findings-emnlp.319. (Visited on 09/08/2023).

[61] Ritesh Kumar et al. *ProtoNER: Few Shot Incremental Learning for Named Entity Recognition Using Prototypical Networks*. Oct. 2023. DOI: 10.48550/arXiv.2310.02372. arXiv: 2310.02372 [cs]. (Visited on 12/06/2023).

[62] Duzhen Zhang et al. "Task Relation Distillation and Prototypical Pseudo Label for Incremental Named Entity Recognition". In: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. CIKM '23. New York, NY, USA: Association for Computing Machinery, Oct. 2023, pp. 3319–3329. ISBN: 9798400701245. DOI: 10.1145/3583780.3615075. (Visited on 12/06/2023).

[63] Juan Manuel Coria et al. "Analyzing BERT Cross-lingual Transfer Capabilities in Continual Sequence Labeling". In: *Proceedings of the First Workshop on Performance and Interpretability Evaluations of Multimodal, Multipurpose, Massive-Scale Models*. Virtual: International Conference on Computational Linguistics, Oct. 2022, pp. 15–25. (Visited on 09/08/2023).

[64] Yi Chen and Liang He. "SKD-NER: Continual Named Entity Recognition via Span-based Knowledge Distillation with Reinforcement Learning". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 6689–6700. DOI: 10.18653/v1/2023.emnlp-main.413. (Visited on 01/25/2024).

[65] Yunan Zhang and Qingcai Chen. *A Neural Span-Based Continual Named Entity Recognition Model*. June 2023. DOI: 10.1609/aaai.v37i11.26638. arXiv: 2302.12200 [cs]. (Visited on 11/10/2023).

[66] Zixuan Ke et al. *Continual Learning Based on Sub-Networks and Task Similarity*. Sept. 2022. (Visited on 11/28/2023).

[67] Rajasekar Venkatesan and Meng Joo Er. "A Novel Progressive Learning Technique for Multi-Class Classification". In: *Neurocomputing* 207 (Sept. 2016), pp. 310–321. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2016.05.006. (Visited on 12/04/2023).

[68] Alec Radford et al. "Language Models Are Unsupervised Multitask Learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[69] Prannay Khosla et al. "Supervised Contrastive Learning". In: *Advances in Neural Information Processing Systems*. Vol. 33. Online and Vancouver, BC, Canada: Curran Associates, Inc., 2020, pp. 18661–18673. (Visited on 02/03/2024).

[70] Xinting Hu et al. "Distilling Causal Effect of Data in Class-Incremental Learning". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Online: IEEE Computer Society, June 2021, pp. 3956–3965. DOI: 10.1109/CVPR46437.2021.00395. (Visited on 03/05/2024).

[71] Lars Kjeldgaard and Lukas Nielsen. *NERDA*. Technical University of Denmark, University of Copenhagen, and Copenhagen Business School, 2021. URL: https://github.com/ebanalyse/NERDA (visited on 12/04/2023).

[72] Mitchell Wortsman et al. "Supermasks in Superposition". In: *Advances in Neural Information Processing Systems*. Vol. 33. Online and Vancouver, BC, Canada: Curran Associates, Inc., 2020, pp. 15173–15184. (Visited on 12/31/2023).

[73] Brian Lester, Rami Al-Rfou, and Noah Constant. "The Power of Scale for Parameter-Efficient Prompt Tuning". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Ed. by Marie-Francine Moens et al. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3045–3059. DOI: 10.18653/v1/2021.emnlp-main.243. (Visited on 03/06/2024).

[74] Erik F Tjong Kim Sang and Fien De Meulder. "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. Edmonton, Canada: Association for Computational Linguistics, 2003, pp. 142–147. URL: https://aclanthology.org/W03-0419.

[75] Sameer Pradhan et al. "Towards Robust Linguistic Analysis using OntoNotes". In: *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Ed. by Julia Hockenmaier and Sebastian Riedel. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 143–152. URL: https://aclanthology.org/W13-3516 (visited on 04/10/2024).

[76] Amber Stubbs, Christopher Kotfila, and Özlem Uzuner. "Automated systems for the de-identification of longitudinal clinical narratives: Overview of 2014

i2b2/UTHealth shared task Track 1". In: *Journal of biomedical informatics* 58 (2015), S11–S19.

[77] Ning Ding et al. "Few-NERD: A Few-shot Named Entity Recognition Dataset". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by Chengqing Zong et al. Online: Association for Computational Linguistics, Aug. 2021, pp. 3198–3213. DOI: 10.18653/v1/2021.acl-long.248. URL: https://aclanthology.org/2021.acl-long.248 (visited on 04/10/2024).

[78] Pengcheng He et al. *DeBERTa: Decoding-enhanced BERT with Disentangled Attention*. arXiv:2006.03654 [cs]. Oct. 2021. DOI: 10.48550/arXiv.2006.03654. URL: http://arxiv.org/abs/2006.03654 (visited on 04/14/2024).

[79] Pengcheng He, Jianfeng Gao, and Weizhu Chen. *DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing*. arXiv:2111.09543 [cs]. Mar. 2023. DOI: 10.48550/arXiv.2111.09543. URL: http://arxiv.org/abs/2111.09543 (visited on 04/14/2024).

[80] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. arXiv:1711.05101 [cs, math]. Jan. 2019. URL: http://arxiv.org/abs/1711.05101 (visited on 04/22/2024).

[81] Wujun Shao et al. *Astronomical Knowledge Entity Extraction in Astrophysics Journal Articles via Large Language Models*. arXiv:2310.17892 [astro-ph]. Jan. 2024. DOI: 10.48550/arXiv.2310.17892. URL: http://arxiv.org/abs/2310.17892 (visited on 02/16/2024).

[82] Swaroop Mishra et al. *Cross-Task Generalization via Natural Language Crowdsourcing Instructions*. arXiv:2104.08773 [cs]. Mar. 2022. DOI: 10.48550/arXiv.2104.08773. URL: http://arxiv.org/abs/2104.08773 (visited on 04/09/2024).

[83] Arslan Chaudhry et al. *On Tiny Episodic Memories in Continual Learning*. June 2019. DOI: 10.48550/arXiv.1902.10486. arXiv: 1902.10486 [cs, stat]. (Visited on 12/06/2023).

VITA

Charles Ian Cutler was born in Portsmouth, Virginia, on February 19, 2002, and grew up in the small town of Smithfield, Virginia. He graduated from Smithfield High School, Smithfield, Virginia in 2020. Charles embarked on his academic journey at Virginia Commonwealth University (VCU) in Fall 2020, transferring nearly 40 credits from high school, which accelerated his undergraduate trajectory. Spring 2022 marked a pivotal moment when Charles met Dr. McInnes during his first Natural Language Processing class. Also around this time, he was accepted into the Accelerated bachelor's to master's program in Computer Science, enabling him to complete his bachelor's and master's in Computer Science within a cumulative four years. Charles received his Bachelor of Science in Computer Science from Virginia Commonwealth University in 2023.

Notable achievements included his senior capstone team winning the Sternheimer Award, a rare honor in the College of Engineering, and clinching 1st place overall at the Engineering Capstone Expo, showcasing his ingenuity and leadership. Spring 2023 saw Charles taking on the maximum workload of graduate courses while completing his undergraduate degree, laying the groundwork for his transition into full-time graduate studies. Currently, he has one research paper almost out the door to publication and is working on two more (one of which is his thesis.)

As Charles embarks on the next chapter of his career, he remains steadfast in his pursuit of excellence in the field of Computer Science. Additionally, he intends to continue playing Jazz. (Charles is a baritone saxophonist.)