Bootleg: Chasing the Tail with Self-Supervised Named Entity Disambiguation

Laurel Orr*, Megan Leszczynski*, Neel Guha, Sen Wu, Simran Arora, Xiao Ling, Christopher Ré
{lorr1,mleszczy,nguha,senwu,simran,chrismre}@cs.stanford.edu,xiaoling@apple.com

ABSTRACT

Named Entity Disambiguation (NED) is the task of mapping textual mentions to entities in a database. A key challenge in NED is generalizing to rarely seen entities, termed tail entities. Traditional NED systems use hand-tuned features to improve tail generalization, but these features make the system challenging to deploy and maintain, especially in multiple locales. In 2018, a subset of the authors built a self-supervised NED system at Apple, which improved performance over its hand-tuned predecessor on a suite of downstream products. Motivated to understand the core reasons for this improvement, we introduce Bootleg, a clean-slate, open-source, self-supervised NED system.[1] We first demonstrate that Bootleg matches or exceeds state-of-the-art performance on three NED benchmarks by up to 5.8 F1 points. Importantly, Bootleg improves performance over a BERT-based NED baseline by 41.2 F1 points on tail entities in Wikipedia using a simple transformer-based architecture and a hierarchical regularization scheme. Finally, we observe that embeddings from self-supervised models like Bootleg are increasingly being served to downstream applications, creating an embedding ecosystem. We initiate the study of the data management challenges associated with this ecosystem.

1 INTRODUCTION

Named entity disambiguation (NED), the process of mapping strings to entities in a database, is a core data management problem and a required step in personal assistants, search, and data cleaning. Users submit queries to these applications that span a long tail of entities, and thus success in NED involves capturing subtle reasoning clues to disambiguate rare (i.e. *tail*) entities. For example, in the query *"What role does Harry play in Dunkirk?"* (Figure 1 (left)), "Harry" refers to popular actor/singer Harry Styles—not Harry Collett or Harry Truman—and "Dunkirk" refers to the movie—not Dunkirk, Indiana or the evacuation of Dunkirk. Alternatively, if the query was *"What roles does Harry play in Dunkirk and Dolittle?"*, "Harry" would refer to Harry Collett as only he plays in both movies Dunkirk and Dolittle. The relationships between Harry Collett and the movies he acts in signals the less popular "Harry".

In 2018 at Apple, a subset of the authors built and deployed a first-of-its-kind NED system to use self-supervision — a paradigm which eliminates the need for engineers to hand-label data and manually curate features. Apple's NED system and its output embeddings — which were integrated into downstream tasks — improved product performance, dramatically reduced engineering effort, and could be extended to new languages and locales more easily.

The experience at Apple suggested that a self-supervised approach was a promising direction for designing an NED system. However, as Apple's system was integrated with other proprietary production systems, it was difficult to distill research takeaways. Thus, we decided to start from scratch and initiate a clean-slate, principled study to better understand how to achieve and maintain high quality with self-supervised NED.

The lessons from industry highlighted two distinct challenges in building and maintaining self-supervised NED systems:

(1) *Tackling the Tail:* A natural baseline for a self-supervised NED system uses BERT [7] (or any deep language model) with entity embeddings to memorize textual co-occurrences between entities and words. This baseline performs well on popular entities which are well-represented in self-supervised data but can struggle on tail entities as there is not enough training data to learn co-occurrences. We find that the baseline achieves over 85 F1 points over all entities but less than 28 F1 points on tail entities on a Wikipedia dataset. The tail however is critical for production workloads. Search applications and voice assistants are known to be tail-heavy: an overwhelming number of queries appear infrequently [1]. Many real world entities also lack text data: in Wikidata, only 11% of entities are in Wikipedia, and in MusicBrainz, less than 1% of songs and artists are in Wikidata.
(2) *Fixing Errors in an Embedding Ecosystem:* Within organizations, entity embeddings from NED models are shipped to downstream product teams like question answering and search. These teams then use the embeddings in their product-specific models, which ship to millions of users. We define the training data, embeddings, and the downstream product teams which use the embeddings as forming an *embedding ecosystem*. Inevitably, errors will occur in downstream products—like returning the wrong "Harry"—due to poor embeddings for different entities, especially at the tail. Today, downstream product teams independently fix these errors which can lead to inconsistencies in embeddings and varying user satisfaction across products. A key challenge is understanding how to fix the embeddings from the self-supervised model so that errors across the embedding ecosystem are resolved in a consistent and maintainable way.

We take a first step towards addressing these challenges by building Bootleg, a clean-slate, open-source, self-supervised system for tail NED. Bootleg achieves state-of-the-art performance on KORE50 [13], RSS500 [10], and AIDA CoNLL-YAGO [12] NED benchmarks and improves over a BERT-based NED baseline that uses no structured data by more than 41 F1 points for tail entities on a Wikipedia dataset. We describe two core components of Bootleg:

(1) *Incorporating Structured Data for the Tail:* Structured data about entities in the form of entity *types* and *relations* can be a readily available resource for all entities, including the tail entities. In Wikidata, 75% of entities that are *not* in Wikipedia have type or knowledge graph connectivity information. We hypothesize

*Equal contribution.
[1]Bootleg is open-source at http://hazyresearch.stanford.edu/bootleg.

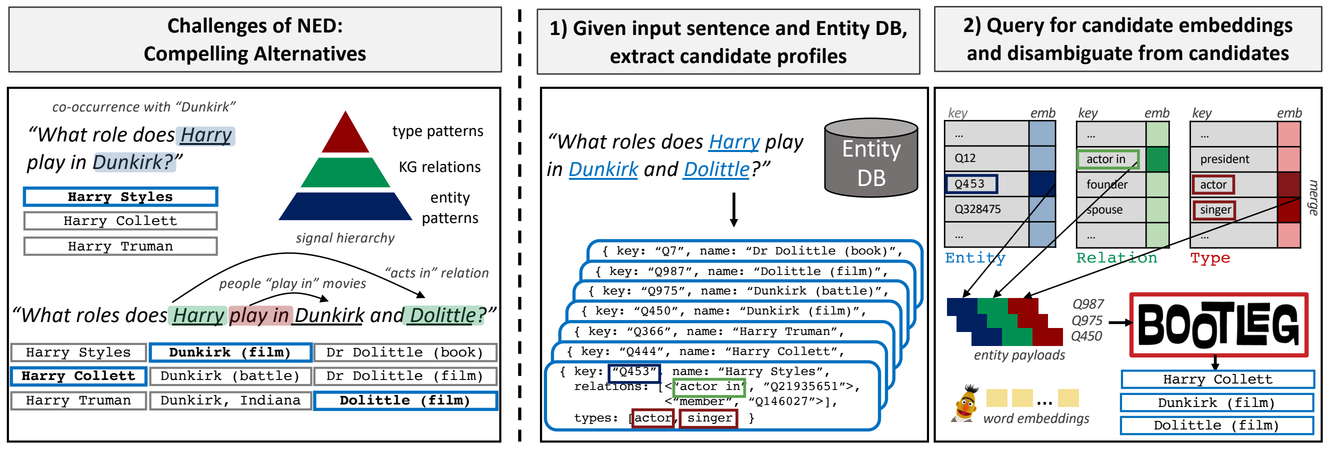
Challenges of NED: Compelling Alternatives
co-occurrence with "Dunkirk"
"What role does Harry play in Dunkirk?"
Harry Styles
Harry Collett
Harry Truman
type patterns
KG relations
entity patterns
signal hierarchy
people "play in" movies
"acts in" relation
"What roles does Harry play in Dunkirk and Dolittle?"
Harry Styles
Harry Collett
Harry Truman
Dunkirk (film)
Dunkirk (battle)
Dunkirk, Indiana
Dr Dolittle (book)
Dr Dolittle (film)
Dolittle (film)
1) Given input sentence and Entity DB, extract candidate profiles
"What roles does Harry play in Dunkirk and Dolittle?"
Entity DB
{ key: "Q7", name: "Dr Dolittle (book)",
{ key: "Q987", name: "Dolittle (film)",
{ key: "Q975", name: "Dunkirk (battle)",
{ key: "Q450", name: "Dunkirk (film)",
{ key: "Q366", name: "Harry Truman",
{ key: "Q444", name: "Harry Collett",
{ key: "Q453", name: "Harry Styles", relations: [<"actor in", "Q21935651">, <"member", "Q146027">], types: [actor, singer] }
2) Query for candidate embeddings and disambiguate from candidates
key
emb
Q12
Q453
Q328475
Entity
actor in
founder
spouse
Relation
president
actor
singer
Type
merge
entity payloads
Q987
Q975
Q450
BOOTLEG
Harry Collett
Dunkirk (film)
Dolittle (film)
word embeddings

Figure 1: (left) shows three reasoning patterns for disambiguation through two example sentences where the correct entity is bolded and (right) shows the dataflow of Bootleg taking an input sentence and outputting disambiguated entities.

that reasoning over this structured data can improve tail generalization. For example, in the question *"How tall is Lincoln?"*, an ideal model can learn that person types have heights to know that "Lincoln" must be a person, rather than a place or brand, and can leverage this type pattern to disambiguate rare people. A key challenge is *how* to incorporate structured data in a self-supervised system. We show how Bootleg incorporates structured data by learning an embedding vector for each type and relation, allowing the model to reason over structured data.

(2) *Managing the Training Set in an Embedding Ecosystem:* As embeddings are simply a function of the training data, improving the training data can help address the root cause of the errors due to the embeddings. In particular, we envision a system where embeddings are fixed upstream at the self-supervised NED model through changes to the source training data. Fixing at the source would obviate the need for each product team to apply idiosyncratic fixes to the embeddings. We introduce the data management challenges associated with this vision, including managing unstructured and structured training data, and present two promising approaches used in Bootleg to fix the errors due to the embeddings.

The rest of the paper is organized as follows: in Section 2, we give an overview of Bootleg. In Section 3, we introduce three reasoning patterns for tail disambiguation and discuss how we encode these patterns using a simple transformer-based architecture with a new regularization technique. In Section 4, we define our vision of the embedding ecosystem, and how embeddings from self-supervised models like Bootleg operate in this ecosystem. In Section 5, we present Bootleg's state-of-the-art results, and demonstrate that entity embeddings produced by Bootleg are useful on downstream tasks, leading to improvements of 8% in a production task at Apple.

We hope the insights from our work will be broadly applicable to the data management challenges inherent to using, managing, and monitoring self-supervised systems. Embeddings are shipped in production at Apple, experienced first-hand by millions of customers. The era of embedding everything both enables exciting new applications and introduces new challenges at the intersection of AI and data management.

2 BOOTLEG OVERVIEW

We now give an overview of Bootleg and motivate our use of structured data for tail disambiguation. Although we focus on Bootleg's process, the inputs and outputs are standard for all NED systems.

Bootleg Dataflow Bootleg takes a sentence as input and outputs the set of entities from an entity database participating in the sentence (dataflow shown in Figure 1). Given the sentence *"What roles does Harry play in Dunkirk and Dolittle?"*, Bootleg queries an *entity database* to find mentions in the sentence and extracts entity candidates for each mention and each candidate's associated entity profile. "Dunkirk", for example, has candidates Dunkirk (film), Dunkirk, Indiana, and Dunkirk (battle). This database also contains structural information about an entity: its set of types (Dunkirk (film) is a movie type), knowledge graph relationships (Dunkirk (film) was directed by Christopher Nolan), possible mentions, and other entity-features. The profile information is encoded as an *entity payload* (described in Section 3) which is input to Bootleg's model. Bootleg selects the most likely candidate (possibly None) from the set of candidates for each mention.

The Motivation for Structured Data To understand the challenges and requirements of tail NED, we start with a natural self-supervised NED baseline, which we refer to as NED-Base. NED-Base incorporates entity embeddings into BERT and is modeled after the system in Févry et al. [9]. BERT gives the baseline the ability to remember phrases in context by learning word co-occurrences and the relative and absolute positions of words. This method excels at disambiguating popular entities as it has seen the entity enough times to memorize distinguishing contextual cues but can struggle on tail entities which are rare in the training data.

Consider disambiguating "Lincoln" in the following: (1) *"How tall is president Lincoln?"* and (2) *"What ingredients are in a Lincoln?"*.

For (1), NED-Base should be able to correctly disambiguate to "Abraham Lincoln" as this entity occurs with the phrase "president" over 3,200 times in the training set. For (2), the baseline is not expected to correctly disambiguate to "Lincoln cocktail" as it has *not* seen the entity "Lincoln cocktail" during training, and as a result, has not learned the association of "in a" or "ingredients" with the entity. As an approximation of how many times NED-Base must see a pattern to memorize it, we measure the performance on a Wikipedia dataset: the baseline achieves 79.3 F1 points on entities it has seen 11-1000 times during training but only 27.8 F1 on entities it has seen 10 or fewer times.

A key insight is that structural resources are more readily available for tail entities than text data, meaning reasoning patterns over structured data can help generalize to the tail. If we extend NED-Base to reason over *type* cues, for example, it could disambiguate the Lincoln cocktail because beverage, not people, types have ingredients. Additionally, these general patterns can be more sample efficient. On a uniform sample of Wikipedia data, there are 2,800 types that each map to more than 100 entities. This means that to disambiguate an entity of a particular type, the NED system can leverage patterns seen for any of the 100+ entities of the corresponding type. To get the same amount of signal without type cues, each of the 280K entities would need to see all of those patterns independently. In BOOTLEG, our goal is to leverage structured data to learn generalizable patterns for the tail.

3 DISAMBIGUATING THE TAIL WITH BOOTLEG

We now describe the implementation of BOOTLEG, a self-supervised NED system designed to succeed on popular (head) *and* tail entities by incorporating structured data, while being easy to maintain. BOOTLEG defines three reasoning patterns leveraging structural signals (Section 3.1) and embeds the signals as inputs to a simple architecture—stacked transformers—to transparently capture the patterns of interest (Section 3.2).

3.1 Incorporating Structured Data

BOOTLEG assembles a three-layer hierarchy (see Figure 1 (left)) of signals that leverages textual information and the structural resources, e.g., types and knowledge graph relations. Our hierarchy represents the tradeoff between discriminativeness and generality of signals with the most discriminative signal at the base.

- **Entity Patterns:** These patterns consist of *entity-level* textual co-occurrences (e.g., the entity "Harry Styles" is associated with the phrase "Dunkirk") and can be learned by the NED-Base model (Section 2). Entity patterns need to be seen *per entity* during training. This may be difficult to scale to applications that encompass millions of entities, like search (e.g., there are 88M entities in Wikidata). As the most discriminative and least general patterns, entity patterns form the base of the hierarchy.
- **Knowledge Graph (KG) Relations:** Relationships between candidates in a sentence can also serve as cues for disambiguation. In the example *"What roles does Harry play in Dunkirk and Dolittle?"*, there are multiple "Harry"'s who act in Dunkirk, but only Harry Collett has a KG relationship "acts in" to Dolittle, the 2020 film, indicated by the phrases "roles" and "play in". This memorization occurs at the *relation-level* so each textual pattern only needs to be seen *per relation* rather than per entity. KG relations are binary and we find that 22% of mentions participated in a relation from a sample of Wikipedia, making them the middle layer of our hierarchy.

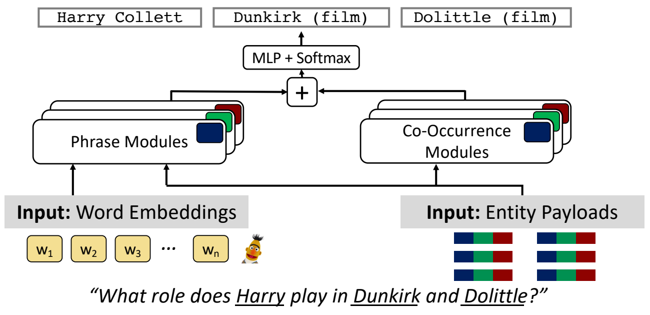

Figure 2: BOOTLEG's architecture.

- **Type Patterns:** Semantic or linguistic cues may indicate that a mention must be of a certain type. In *"What role does Harry play in Dunkirk?"*, "play" suggests that "Dunkirk" refers to the movie, not the WWII evacuation. Type memorization occurs at the *type-level* so an entity of a particular type can leverage textual patterns learned from every other mention of that type. As 96% of mentions have some type, they are the most general signals and form the top of our hierarchy.

This hierarchy captures a key point of tension in self-supervised models: models will more easily learn the more discriminative, less generalizable features. However, to improve disambiguation of tail entities, we need to incorporate structural resources and encourage the model to learn the more general patterns. We now discuss how we leverage embeddings to encode these patterns.

Embedding the Hierarchy We encode the signals by taking an "embedding-centric" view and represent each layer of the hierarchy as a set of embeddings. All embeddings are jointly learned during training. Each entity, type, and KG relation is assigned its own embedding vector, allowing each property to store relevant disambiguation cues. For example, if the "movie" *type* learns cues such as the words "stars in" or "viewed", every entity with the movie type will receive the same cues in the movie embedding vector. Given an input sentence and the set of candidate entity profiles (described in Section 2), BOOTLEG queries the relevant embedding matrix for each candidate's entity, type, and relation embeddings to form the entity payload. We use BERT to encode the sentence into contextual word embeddings. The word embeddings and the entity payloads are passed to BOOTLEG's backbone model, described next.

3.2 Learning Reasoning Patterns

Given the word embeddings and the entity payloads, BOOTLEG uses two modules of the standard transformer architecture [27] to learn each reasoning pattern. One module is for phrase memorization (left of Figure 2) and the other is for co-occurrence memorization (right of Figure 2), which encompasses entity, type, and KG co-occurrence patterns; e.g., when entities are of similar types, such as entities in a list. This allows for the embeddings for each level of the hierarchy to learn dependencies between related phrases as well as between embeddings. For each entity candidate, BOOTLEG adds the outputs of the two modules and scores the result via an MLP softmax layer. The most likely candidate for each mention is selected as the correct entity.

Hierarchical Regularization Without interference, Bootleg will overly-leverage more discriminative, entity-specific features over more general ones. To improve tail generalization, we discourage the model from relying on entity-specific textual cues to disambiguate more rare entities by regularizing ("zeroing out") the *entire* entity embedding with frequency proportional to the inverse of the entity's popularity (e.g., the more popular the less regularized). We find this boosts performance by up to 2.8 F1 points over entities occurring 10 or fewer times and 13.6 F1 points over entities unseen during training compared to not masking the entity embedding. This can be repeated at each level of the hierarchy.

4 MANAGING THE TRAINING SET IN AN EMBEDDING ECOSYSTEM

We now describe our vision of how embeddings are becoming a new medium for data storage and sharing, presenting exciting data management challenges.

4.1 The Embedding Ecosystem Vision

Embeddings from NED models can improve performance in downstream products that leverage knowledge of entities (e.g., question answering, news, search). For instance, using Bootleg embeddings at Apple leads to up to 8% improvements in a downstream task used to answer factoid questions. With the recent advances in self-supervised models such as BERT [7] and GPT-3 [2], embeddings are increasingly used in products and becoming a powerful and complex data storage tool in industry pipelines [18, 25, 29]. This trend is occurring across products and companies, and we believe it is part of a larger shift in the future of data management.

We refer to the training data, embeddings, and the downstream products that consume them as forming an *embedding ecosystem*. As shown in Figure 3, the ecosystem is instantiated by training self-supervised models like Bootleg on structured and unstructured data sources. These models then generate embeddings which are stored in an embedding server—similar to models being stored in a model store [11]. This server then shares the stored embeddings among downstream products. The embedding, a complicated function of the input training data, is therefore becoming an efficient medium to store and share data.

4.2 Data Management Challenges in the Embedding Ecosystem

An ecosystem where embeddings are the central data format presents numerous new data management challenges. We will focus our discussion on the challenge of managing embedding quality in downstream products, specifically in the context of entity embeddings. We highlight other future directions below.

In today's pipelines, downstream products may make errors (often over the tail) due to failure modes and poor quality of the embeddings. When these errors occur, downstream product teams often patch the embedding errors independently. This can lead to inconsistencies between embedding versions, which can cause

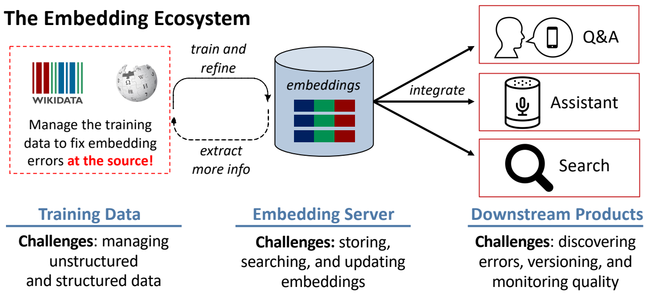
The Embedding Ecosystem
WIKIDATA
Manage the training data to fix embedding errors at the source!
train and refine
extract more info
embeddings
integrate
Q&A
Assistant
Search
Training Data
Challenges: managing unstructured and structured data
Embedding Server
Challenges: storing, searching, and updating embeddings
Downstream Products
Challenges: discovering errors, versioning, and monitoring quality

Figure 3: Embeddings are trained, shipped to an embedding server, and integrated into downstream products.

products to disambiguate entities differently from each other and users to experience varying quality across products. Moreover, the idiosyncratic fixes by each team do not address the root cause of the embedding error: the embedding failed to capture some signal in the training data.

Our envisioned solution to this problem is for embeddings to be fixed *upstream* at the source task (e.g., the NED system) by treating this as a *data management problem*. Specifically, we imagine fixing the self-supervised NED models through managing the training data, not the model architecture or training scheme.

During training, NED models can leverage two types of data: *unstructured* data, such as the Wikipedia page text, and *structured* data, such as an entity's types and relationships from a knowledge graph. We now describe the data management challenges for each below and our proposed solutions.

4.2.1 Managing Unstructured Data. For self-supervised NED models trained on unstructured data (e.g., textual data from Wikipedia), errors arise because the data is often incomplete or naturally sparse over the tail.[2] We now introduce two *data management* techniques that Bootleg uses to improve the unstructured training data: (1) training set refinement to *globally* improve the data quality, and (2) model guiding through data to address *targeted* errors. Finally, we describe open challenges for managing unstructured data in the embedding pipeline.

Training Set Refinement: To globally improve data quality, we apply weak supervision [22], a technique to noisily, but efficiently, label data, by using heuristic *labeling functions* to label unlinked mentions. For each entity's Wikipedia page, we label pronouns and known entity mentions as links to the entity. Our initial work shows that this refinement technique improves performance by up to 2.6 F1 points over entities unseen during training.

Model Guiding through Training Data: A targeted error is one where a model consistently mispredicts a specific textual, type, or relation cue. We will use the running example of a model predicting national sports teams as countries (e.g., predicting England (country) rather than the correct England (soccer team) in the sentence "England played Spain in soccer". To correct targeted errors, we apply three techniques: data augmentation [30], data slicing [3], and weak supervision [22].

- *Textual augmentation* involves applying transformations to original training examples to boost the size of the training

[2]This parallels traditional DBMS problems of KB and relational data cleaning.

dataset and provide more diverse signals to the model. In Bootleg, we take examples where a positive signal is present (e.g., examples with a national sports team rather than a country) and augment these examples by swapping mentions of similar types and KG relations.

- *Data slicing* is a technique for training an inexpensive mixture-of-experts model where an expert specializes on a specific subset (e.g., a subset where a national sports team is the correct answer).
- *Weak supervision*, as described above, can also be modified to increase coverage for specific missed signals. For example, we can add examples with national sports teams by labeling sports news data or using sport-specific heuristic labeling functions to label Wikipedia.

We find that applying these techniques on targeted errors increases model performance by 2-12 F1 points.[3]

While our initial results on refinement and guiding are promising, there remain open challenges in understanding how these techniques can be combined, how to evaluate the trade-offs between them, and how to automate and optimize the process.

4.2.2 Managing Structured Data. Structured data, like unstructured data, can also be incomplete and sparse on tail entities. While well-developed knowledge base and relational database cleaning techniques [16, 24] can address some challenges around missing and incomplete data, there remain open challenges of correctly managing signals of different generalities and creating structured data for resource-poor languages or topical domains such as medicine or music. We discuss each in turn.

First, the signals in structured sources face trade-offs between generality and discriminativeness. For example, in Bootleg, types are more general but less discriminative than KG relations, which are less discriminative than entities (Section 3.1). In Bootleg, we find that regularizing our entity signals by inverse popularity in training data greatly improves model performance (Section 3.2), but it remains an open challenge to help developers automatically manage and regularize these different signals. We need new methods to evaluate signals' generality and discriminativeness and new techniques for automating the application of signal regularization.

Second, rare languages and locals, such as Greenlandic and Tahitian, have limited representation in knowledge bases such as Wikidata. Extracting candidate entities and labeling these entities with Wikidata types and relations is manually intensive. Certain languages or topical domains such as medicine may also introduce types and relations that differ from standard English Wikidata. It remains an open challenge to develop self-supervised approaches for creating knowledge graphs and extracting relevant relations to support these new domains.

4.2.3 New Opportunities in an Embedding Ecosystem. Although we detail the challenges of managing unstructured and structured data in the new embedding ecosystem to fix model errors, there are many other exciting data management challenges and opportunities. Some relevant open questions are how to search to find relevant embeddings efficiently, how to monitor and detect out-of-date embeddings, how to measure the quality of embeddings, and how to efficiently store and version embeddings.

As industry organizations maintain ever-expanding data lakes, another significant challenge is inspecting the data to discover sources of error. We imagine that the embeddings generated from the data could be used to extract *more* structured information from the data lake (i.e., using standard *information extraction* techniques). The structured signals could then be used in a feedback loop to refine the original embeddings (shown in Figure 3).

Table 1: We compare Bootleg to the best published numbers on three NED benchmarks. "-" indicates that the metric was not reported. Bolded numbers indicate the best value.

Benchmark	Model	Precision	Recall	F1
KORE50	Hu et al. [14][4]	80.0	79.8	79.9
	Bootleg	**86.0**	**85.4**	**85.7**
RSS500	Phan et al. [21]	82.3	82.3	82.3
	Bootleg	**82.5**	**82.5**	**82.5**
AIDA	Févry et al. [9]	-	**96.7**	-
	Bootleg	96.9	**96.7**	96.8

5 EVALUATION

We demonstrate that Bootleg (1) achieves state-of-the-art performance on three NED benchmarks, (2) outperforms a BERT-based baseline on the tail, (3) maintains performance under inference-time memory optimizations, and (4) improves performance in downstream tasks in a production system at Apple (Section 5.1). We further investigate the limitations of Bootleg through examining buckets of Bootleg errors (Section 5.2).

Experimental Setup We define our database as the set of entities in Wikipedia (for a total of 5.3M entities), and use Wikipedia, YAGO, and Wikidata to mine for the type and relation information needed for our entity profiles. We allow each mention to have up to 30 possible candidates and train on Wikipedia anchor links from the Nov 2019 dump with training set refinement. The entire Wikipedia dataset has a total of 57M sentences.

We train all models over Wikipedia sentences with a maximum word token length of 100 using 8 NVIDIA V100 GPUs for two epochs. For our benchmark model (only used in Table 1), we train for one epoch and add a title embedding, sentence co-occurrence, and Wikipedia page co-occurrence feature.

5.1 Bootleg Performance

We discuss Bootleg's benchmark and tail performance, performance under memory constraints, and performance on a downstream task in industry.

Benchmark Performance To understand the overall performance of Bootleg, we compare against reported state-of-the-art numbers of two standard sentence benchmarks (KORE50, RSS500) and the standard document benchmark (AIDA CoNLL-YAGO). For AIDA,

[3]We used the three techniques to correct two under-performing subsets where the model, trained on a 400K sentence subset of Wikipedia, was predicting countries instead of airports and locations instead of football teams.

[4]Although Hu et al. [14] does end-to-end entity linking, their reported KORE50 result is the current SotA, beating the result of 78 from Phan et al. [21].

Table 2: We compare Bootleg to a BERT-based NED baseline (NED-Base) on validation sets of a Wikipedia dataset. We report micro F1 scores.

Validation Set	NED-Base	Bootleg	# Mentions
All Entities	85.9	**91.3**	4,066K
Torso Entities	79.3	**87.3**	1,912K
Tail Entities	27.8	**69.0**	163K
Unseen Entities	18.5	**68.5**	10K

we follow standard procedure and fine-tune the Bootleg model on the training set, choosing the test score associated with the best dev score[5]. In Table 1, we show that Bootleg achieves up to 5.8 F1 points higher than prior reported numbers.

Tail Performance To validate that Bootleg improves tail disambiguation, we compare against NED-Base (see Section 2).[6] NED-Base learns entity embeddings by maximizing the dot product between the gold entity candidates and the BERT-contextualized representations of the mention. Both NED-Base and Bootleg take approximately 14 hours per epoch.

NED-Base is successful overall and achieves 85.9 F1 points, which is within 5.4 F1 points of Bootleg (Table 2). However, when we examine performance over the torso and tail, we see that Bootleg outperforms NED-Base by 8.0 and 41.2 F1 points, respectively. Finally, on unseen entities, Bootleg outperforms NED-Base by 50 F1 points. Note that NED-Base only has access to textual data, indicating that text is often sufficient for popular entities, but not for rare entities.

Memory Optimized Inference As 99% of the model size is taken up by the entity embeddings, we explore how to optimize inference by only using a subset of entity embeddings. This reduces the overall memory requirement and allows for larger batch sizes. Specifically, we reduce the memory consumption by 95% by keeping the entity embedding for the top 5% of entities ranked by popularity. For the remaining entities, we use the same entity embedding, chosen randomly from the embeddings for unseen entities. We find that this memory reduction only sacrifices 0.8 F1 points overall and, in fact, improves tail performance by 2 F1 points. We hypothesize that this improvement is due to the fact that the majority of entity candidates all have the same learned embedding, decreasing the amount of conflict among candidates from textual patterns.

Industry Use Case We demonstrate how the learned entity embeddings from a Bootleg model provide useful information to a system at Apple that answers factoid queries such as *"How tall is the president of the United States?"*. We use Bootleg's embeddings in the Overton system [23] and compare to the same system without Bootleg embeddings as the baseline. We measure the overall test quality (F1) on an in-house entity disambiguation task as well as the quality over the tail slices which includes unseen entities. The relative quality is reported as a percentage of the baseline; e.g., if the baseline F1 is 80.0 and the subject F1 is 88.0, the relative quality is 88.0/80.0 = 1.1. Table 3 shows Bootleg's embeddings consistently result in a positive relative quality, even over Spanish, French, and German, where improvements are most visible in the tail entities.

Table 3: Relative quality of an Overton model with Bootleg embeddings over one without in four languages.

Validation Set	English	Spanish	French	German
All Entities	1.08	1.03	1.02	1.00
Tail Entities	1.08	1.17	1.05	1.03

5.2 Error Analysis

We have shown that Bootleg succeeds on NED benchmarks, tail NED, and downstream tasks. To understand Bootleg's limitations, we identify four key buckets of errors made by Bootleg.

- **Granularity** Bootleg struggles with *granularity*, predicting an entity that is too general or specific (e.g., *Academy Award* vs. *Academy Award for Best Actress*). The set of all examples where Bootleg's predicted entity is a Wikidata sub-type (too specific) or super-type (too general) of the gold entity covers 12% of overall and 7% of tail errors.
- **Numerical** Entities containing numerical tokens are challenging for Bootleg to disambiguate. This may be because the BERT model represents certain numbers with sub-word tokens and is known to under-perform other language models on numbers [28]. The slice of data where the entity title contains a numerical year (e.g., *1976 Summer Olympics*) covers 14% of overall and 25% of tail errors.
- **Multi-Hop** Multi-hop reasoning involves utilizing information that is not present in the sentence but is linked to the present entities through hops over edges in the KG. For example, two city entities in a sentence are not directly connected in the KG, but both are connected to the same state entity. The set of examples requiring two-hop reasoning covers 6% of overall and 7% of tail errors. Bootleg only encodes single-hop patterns (direct KG connections), so multi-hop reasoning is a fundamental limitation of the current model.
- **Exact Match** Bootleg struggles on examples where the exact entity title appears in the text. In 28% of the examples where NED-Base is correct but Bootleg is incorrect, the textual mention is an exact match of the gold entity title. We attribute the performance drop to Bootleg's regularization scheme: mention-to-entity similarity would be encoded in Bootleg's entity embedding, but the regularization encourages Bootleg to not rely on entity-level information.

These error buckets are key opportunities for future work alongside extending Bootleg to new languages and new domains, which may lack representation in resources such as Wikidata and Wikipedia.

6 RELATED WORK

We discuss related work in terms of both NED and entity resolution systems, and the broader landscape of self-supervised models and tail data. NED and entity resolution, also called record linkage, are both *entity matching* problems that aim to find occurrences of entities. NED involves matching free text to a record in a knowledge graph, while entity resolution involves matching the same structured record in different datasets. Standard approaches to entity

[5]We use the standard candidate list from Pershina et al. [19] for fine-tuning and inference for AIDA CoNLL-YAGO. For the other benchmarks, we follow Phan et al. [21] and allow for mention boundary expansion and generate candidates by choosing the top scoring candidates based on their title and Wikipedia page context.
[6]Code is not publicly available for baseline models reported in Table 1.

matching have leveraged statistical techniques (e.g., link counts and feature similarity), but these systems tend to be hard to maintain over time, with the work of Petasis et al. [20] building a model to detect when a rule-based NED system needs to be retrained and updated. In recent years, deep learning systems have become standard (see Mudgal et al. [17] for a high-level overview of deep learning approaches to entity matching problems). Jin et al. [15] and Hoffart et al. [13] study disambiguation at the tail, and both rely on phrase-based language models for feature extraction. Unlike our work, they do not fuse various type or knowledge-graph information for disambiguation.

Recent work in Ré et al. [23] demonstrates the importance of fine-grained performance metrics (data slices), support for weakly supervised data, and pretrained embeddings. Bootleg is another such system showing the benefit of these methods.

AutoML focuses on automatic search of model architecture and hyperparameters [8]. In Bootleg, we take the position that the data is more important than the architecture. The work of Chepurko et al. [4] adopts a similar position and focuses on automating the data discovery and curation process for ML.

The work of Tata et al. [26], Chung et al. [6], and Chung et al. [5] all focus on the importance of the tail during inference and the challenges of capturing it during training. They all highlight the data management challenges of monitoring the tail (and other missed slices of data) and improving generalizability. The augmentation and weak supervision techniques in Bootleg are similar to Chung et al. [5]'s synthetically generated training examples.

7 CONCLUSION

In this work, we present Bootleg, a state-of-the-art self-supervised system for NED. Bootleg leverages structured and unstructured data to learn reasoning patterns that generalize to rarely seen *tail* entities, while maintaining high performance on popular entities. Bootleg improves performance over a BERT-based NED baseline by over 41 F1 points on tail entities on a Wikipedia dataset, and Bootleg embeddings improve performance on a downstream production task at Apple by 8%. More broadly, embeddings from self-supervised systems like Bootleg are used in suites of downstream products and are becoming a core medium for storing and sharing data. We introduce and discuss the data management challenges associated with this new embedding ecosystem. We hope this work inspires future research towards realizing the vision of an embedding ecosystem surrounding self-supervised models.

Acknowledgments We thank Jared Dunnmon, Dan Fu, Karan Goel, Monica Lam, Fred Sala, and Silei Xu for their valuable feedback and Pallavi Gudipati for help with experiments.

REFERENCES

[1] Michael S Bernstein, Jaime Teevan, Susan Dumais, Daniel Liebling, and Eric Horvitz. Direct answers for search queries in the long tail. In *SIGCHI*, 2012.
[2] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.
[3] Vincent Chen, Sen Wu, Alexander J Ratner, Jen Weng, and Christopher Ré. Slice-based learning: A programming model for residual learning in critical data slices. In *Neurips*, 2019.
[4] Nadiia Chepurko, Ryan Marcus, Emanuel Zgraggen, Raul Castro Fernandez, Tim Kraska, and David Karger. Arda: Automatic relational data augmentation for machine learning. In *VLDB*, 2020.
[5] Yeounoh Chung, Peter J Haas, Eli Upfal, and Tim Kraska. Unknown examples & machine learning model generalization. *arXiv preprint arXiv:1808.08294*, 2018.
[6] Yeounoh Chung, Neoklis Polyzotis, Kihyun Tae, Steven Euijong Whang, et al. Automated data slicing for model validation: A big data-ai integration approach. *TKDE*, 2019.
[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.
[8] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Neurips*, 2015.
[9] Thibault Févry, Nicholas FitzGerald, Livio Baldini Soares, and Tom Kwiatkowski. Empirical evaluation of pretraining strategies for supervised entity linking. In *AKBC*, 2020.
[10] Daniel Gerber, Sebastian Hellmann, Lorenz Bühmann, Tommaso Soru, Ricardo Usbeck, and Axel-Cyrille Ngonga Ngomo. Real-time rdf extraction from unstructured data streams. In *ISWC*, 2013.
[11] Josh Gordon. Introducing tensorflow hub: A library for reusable machine learning modules in tensorflow. https://medium.com/tensorflow/introducing-tensorflow-hub-a-library-for-reusable-machine-learning-modules-in-tensorflow-cdee41fa18f9, 2018.
[12] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *EMNLP*, 2011.
[13] Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. Kore: keyphrase overlap relatedness for entity disambiguation. In *CIKM*, 2012.
[14] Shengze Hu, Zhen Tan, Weixin Zeng, Bin Ge, and Weidong Xiao. Entity linking via symmetrical attention-based neural network and entity structural features. *Symmetry*, 2019.
[15] Yuzhe Jin, Emre Kıcıman, Kuansan Wang, and Ricky Loynd. Entity linking at the tail: sparse signals, unknown entities, and phrase models. In *WSDM*, 2014.
[16] Xueling Lin, Haoyang Li, Hao Xin, Zijian Li, and Lei Chen. Kbpearl: a knowledge base population system supported by joint entity and relation linking. In *VLDB*, 2020.
[17] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. Deep learning for entity matching: A design space exploration. In *SIGMOD*, 2018.
[18] Pandu Nayak. Understanding searches better than ever before. https://www.blog.google/products/search/search-language-understanding-bert/, 2019.
[19] Maria Pershina, Yifan He, and Ralph Grishman. Personalized page rank for named entity disambiguation. In *NAACL*, 2015.
[20] Georgios Petasis, Frantz Vichot, Francis Wolinski, Georgios Paliouras, Vangelis Karkaletsis, and Constantine D Spyropoulos. Using machine learning to maintain rule-based named-entity recognition and classification systems. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 426–433, 2001.
[21] Minh C. Phan, Aixin Sun, Yi Tay, Jialong Han, and Chenliang Li. Pair-linking for collective entity disambiguation: Two could be better than all. *TKDE*, 2019.
[22] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In *VLDB*, 2017.
[23] Christopher Ré, Feng Niu, Pallavi Gudipati, and Charles Srisuwananukorn. Overton: A data system for monitoring and improving machine-learned products. In *CIDR*, 2020.
[24] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. HoloClean: Holistic data repairs with probabilistic inference. In *VLDB*, 2017.
[25] Dan Shiebler, Chris Green, Luca Belli, and Abhishek Tayal. Embeddings@Twitter. https://blog.twitter.com/engineering/en_us/topics/insights/2018/embeddingsattwitter.html, 2018.
[26] Sandeep Tata, Vlad Panait, Suming Jeremiah Chen, and Mike Colagrosso. Itemsuggest: A data management platform for machine learned ranking services. In *CIDR*, 2019.
[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neurips*, 2017.
[28] Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. Do nlp models know numbers? probing numeracy in embeddings. *arXiv preprint arXiv:1909.07940*, 2019.
[29] Kyle Wiggers. Amazon researchers use nlp data set to improve alexa's answers. https://venturebeat.com/2019/11/25/amazon-researchers-use-nlp-data-set-to-improve-alexas-answers/, 2019.
[30] Sen Wu, Hongyang Zhang, Gregory Valiant, and Christopher Ré. On the generalization effects of linear transformations in data augmentation. In *ICML*, 2020.