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

Laurel Orr[†], Megan Leszczynski[†], Simran Arora[†], Sen Wu[†], Neel Guha[†], Xiao Ling[‡], and Christopher Ré[†]

[†]Stanford University
[‡]Apple

{lorr1,mleszczy,simran,senwu,nguha,chrismre}@cs.stanford.edu,
xiaoling@apple.com

Abstract

A challenge for named entity disambiguation (NED), the task of mapping textual mentions to entities in a knowledge base, is how to disambiguate entities that appear rarely in the training data, termed *tail* entities. Humans use subtle reasoning patterns based on knowledge of entity facts, relations, and types to disambiguate unfamiliar entities. Inspired by these patterns, we introduce Bootleg, a self-supervised NED system that is explicitly grounded in reasoning patterns for disambiguation. We define core reasoning patterns for disambiguation, create a learning procedure to encourage the self-supervised model to learn the patterns, and show how to use weak supervision to enhance the signals in the training data. Encoding the reasoning patterns in a simple Transformer architecture, Bootleg meets or exceeds state-of-the-art on three NED benchmarks. We further show that the learned representations from Bootleg successfully transfer to other non-disambiguation tasks that require entity-based knowledge: we set a new state-of-the-art in the popular TACRED relation extraction task by 1.0 F1 points and demonstrate up to 8% performance lift in highly optimized production search and assistant tasks at a major technology company.

1 Introduction

Knowledge-aware deep learning models have recently led to significant progress in fields ranging from natural language understanding [38, 41] to computer vision [56]. Incorporating explicit knowledge allows for models to better recall factual information about specific entities [38]. Despite these successes, a persistent challenge that recent works continue to identify is how to leverage knowledge for low-resource regimes, such as *tail* examples that appear rarely (if at all) in the training data [16].

In this work, we study knowledge incorporation in the context of named entity disambiguation (NED) to better disambiguate the long tail of entities that occur infrequently during training.[1] Humans disambiguate by leveraging subtle reasoning over entity-based knowledge to map strings to entities in a knowledge base. For example, in the sentence *"Where is Lincoln in Logan County?"*, resolving the mention "Lincoln" to "Lincoln, IL" requires reasoning about *relations* because "Lincoln, IL"—not "Lincoln, NE" or "Abraham Lincoln"—is the capital of Logan County. Previous NED systems disambiguate by memorizing co-occurrences between entities and textual context in a self-supervised manner [16, 51]. The self-supervision is critical to building a model that is easy to maintain and does not require expensive hand-curated features. However, these approaches struggle to handle tail entities: a baseline SotA model from [16] achieves less than 28 F1 points over the tail, compared to 86 F1 points over all entities.

Despite their rarity in training data, many real-world entities are tail entities: 89% of entities in the Wikidata knowledge base do not have Wikipedia pages to serve as a source of textual training data. However, to achieve 60 F1 points on disambiguation, we find that the prior SotA baseline model should see an entity

[1] In this work, we define tail entities as those occurring 10 or fewer times in the training data.

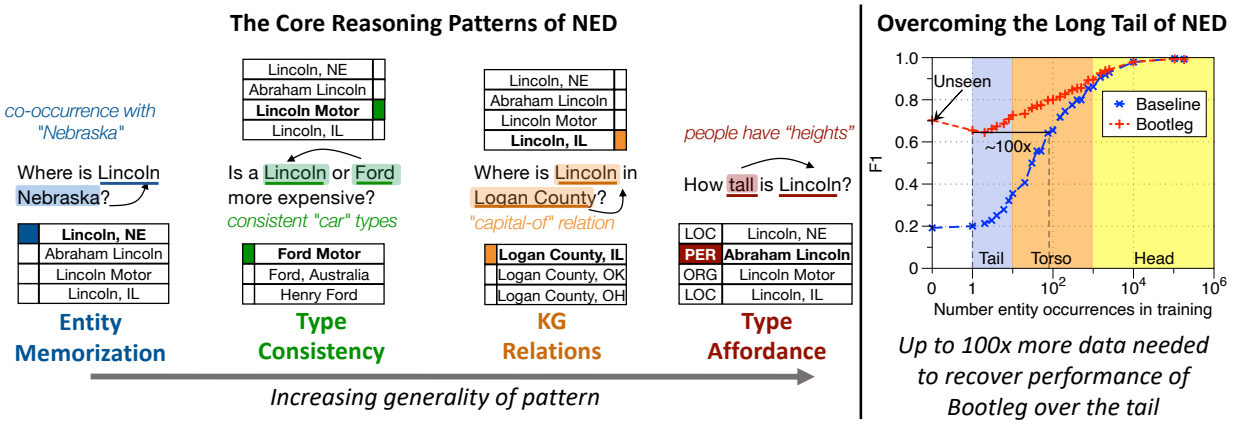

Figure 1: (Left) shows the four reasoning patterns for disambiguation. The correct entity is bolded. (Right) shows F1 versus number of times an entity was seen in training data for a baseline NED model compared to BOOTLEG across the head, torso, tail, and unseen.

on-the-order-of 100 times during training (Figure 1 (right)). This presents a scalability challenge as there are 15x more entities in Wikidata than in Wikipedia, the majority of which are tail entities. For the model to observe each of these tail entities 100x, the training data would need to be scaled by 1,500x the size of Wikipedia. Prior approaches struggle with the tail, yet industry applications such as search and voice assistants are known to be tail-heavy [4, 20]. Given the requirement for high quality tail disambiguation, major technology companies continue to press on this challenge [29, 39].

Instead of scaling the training data until co-occurrences between tail entities and text can be memorized, we define a principled set of reasoning patterns for entity disambiguation across the head and tail. When humans disambiguate entities, they leverage signals from context as well as from entity *relations* and *types*. For example, resolving "Lincoln" in the text *"How tall is Lincoln?"* to "Abraham Lincoln" requires reasoning that people, not locations or car companies, have heights—a type affordance pattern. These core patterns apply to both head and tail examples with high coverage and involve reasoning over entity facts, relations, and types, information which is available for both head and tail in structured data sources. [2] Thus, we hypothesize that these patterns assembled from the structured resources can be learned over training data and generalize to the tail.

In this work, we introduce BOOTLEG, an open-source, self-supervised NED system designed to succeed on head *and* tail entities. [3] BOOTLEG encodes the entity, relation, and type signals as embedding inputs to a simple stacked Transformer architecture. The key challenges we face are understanding how to use knowledge for NED, designing a model that learns those patterns, and fully extracting the useful knowledge signals from the training data:

- **Tail Reasoning:** Humans use subtle reasoning patterns to disambiguate different entities, especially unfamiliar tail entities. The first challenge is characterizing these reasoning patterns and understanding their coverage over the tail.
- **Poor Tail Generalization:** We find that a model trained using standard regularization and a combination of entity, type and relation information performs 10 F1 points *worse* on disambiguating unseen entities compared to the two models which respectively use only type and only relation information. We find this performance drop is due to the model's over-reliance on discriminative textual and entity features compared to more general type and relation features.
- **Underutilized Data:** Self-supervised models improve with more training data [7]. However, only a

[2] We find that type affordance patterns apply to over 84% of all examples, including tail examples, while KG relation patterns apply to over 27% of all examples and type consistency applies to over 8% of all examples. In Wikidata, 75% of entities that are *not* in Wikipedia have type or knowledge graph connectivity signals, and among tail entities, 88% are in non-tail type categories and 90% are in non-tail relation categories.

[3] BOOTLEG is open-source at `http://hazyresearch.stanford.edu/bootleg`

limited portion of the standard NED training dataset, Wikipedia, is useful: Wikipedia lacks labels [19] and we find that an estimated 68% of entities in the dataset are not labeled.[4]

Bootleg addresses these challenges through three contributions:

- **Reasoning Patterns for Disambiguation:** We contribute a principled set of core disambiguation patterns for NED (Figure 1 (left))—entity memorization, type consistency, KG relation, and type affordance—and show that on slices of Wikipedia examples exemplifying each pattern, Bootleg provides a lift over the baseline SotA model on tail examples by 18 F1, 56 F1, 62 F1, and 45 F1 points respectively. Overall, using these patterns, Bootleg meets or exceeds state-of-the-art performance on three NED benchmarks and outperforms the prior SotA by more than 40 F1 points on the tail of Wikipedia.
- **Generalizing Learning to the Tail:** Our key insight is that there are distinct entity-, type-, and relation- tails. Over tail entities (based on entity count in the training data), 88% have non-tail types and 90% have non-tail relations. The model should balance these signals differently depending on the particular entity being disambiguated. We thus contribute a new 2D regularization scheme to combine the entity, tail, and relation signals and achieve a lift of 13.6 F1 points on unseen entities compared to the model using standard regularization techniques. We conduct extensive ablation studies to verify the effectiveness of our approach.
- **Weak Labelling of Data:** Our insight is that because Wikipedia is highly structured—most sentences on an entity's Wikipedia page refer to that entity via pronouns or alternative names—we can weakly label our training data to label mentions. Through weak labeling, we increase the number of labeled mentions in the training data by 1.7x, and find this provides a 2.6 F1 point lift on unseen entities.

With these three contributions, Bootleg achieves SotA on three NED benchmarks. We further show that embeddings from Bootleg are useful for downstream applications that require the knowledge of entities. We show the reasoning patterns learned in Bootleg transfer to tasks beyond NED by extracting Bootleg's learned embeddings and using them to set a new SotA by 1.0 F1 points on the TACRED relation extraction task [2, 53], where the prior SotA model also uses entity-based knowledge [38]. Bootleg representations further provide an 8% performance lift on highly optimized industrial search and assistant tasks at a major technology company. For Bootleg's embeddings to be viable for production, it is critical that these models are space-efficient: the models using only Bootleg relation and type embeddings each achieve 3.3x the performance of the prior SotA baseline over unseen entities using 1% of the space.

2 NED Overview and Reasoning Patterns

We now define the task of named entity disambiguation (NED), the four core reasoning patterns, and the structural resources required for learning the patterns.

Task Definition Given a knowledge base of entities $\mathcal{E}$ and an input sentence, the goal of named entity disambiguation is to determine the entities $e \in \mathcal{E}$ referenced in each sentence. Specifically, the input is a sequence of N tokens $\mathcal{W} = \{w_1, \ldots, w_N\}$ and a set of M non-overlapping spans in the sequence $\mathcal{W}$, termed *mentions*, to be disambiguated $\mathcal{M} = \{m_1, \ldots, m_M\}$. The output is the most likely entity for each mention.

The Tail of NED We define the tail, torso, and head of NED as entities occurring less than 11 times, between 11 and 1,000, and more than 1,000 times in training, respectively. Following Figure 1 (right), the head represents those entities a simple language-based baseline model can easily resolve, as shown by a baseline SotA model from [16] achieving 86 F1 over all entities. These entities were seen enough times during training to memorize distinguishing contextual cues. The tail represents the entities these models struggle to resolve due to their rarity in training data, as shown by the same baseline model achieving less than 28 F1 on the tail.

[4] We computed this statistic by computing the number of proper nouns and the number of pronouns/known aliases for an entity on that entity's page that were not already linked.

2.1 Four Reasoning Patterns

When humans disambiguate entities in text, they conceptually leverage signals over entities, relationships, and types. Our empirical analysis reveals a set of desirable reasoning patterns for NED. The patterns operate at different levels of granularity (see Figure 1 (left))—from patterns which are highly specific to an entity, to patterns which apply to categories of entities—and are defined as follows.

- **Entity Memorization:** We define entity memorization as the factual knowledge associated with a specific entity. Disambiguating "Lincoln" in the text *"Where is Lincoln, Nebraska?"* requires memorizing that "Lincoln, Nebraska", not "Abraham Lincoln" frequently occurs with the text "Nebraska" (Figure 1 (left)). This pattern is easily learned by now-standard Transformer-based language models. As this pattern is at the *entity-level*, it is the least general pattern.
- **Type Consistency:** Type consistency is the pattern that certain textual signals in text indicate that the types of entities in a collection are likely similar. For example, when disambiguating "Lincoln" in the text *"Is a Lincoln or Ford more expensive?"*, the keyword "or" indicates that the entities in the pair (or sequence) are likely of the same Wikidata type, "car company". Type consistency is a more general pattern than entity memorization, covering 12% of the tail examples in a sample of Wikipedia.[5]
- **KG Relations:** We define the knowledge graph (KG) relation pattern as when two candidates have a known KG relationship and textual signals indicate that the relation is discussed in the sentence. For example, when disambiguating "Lincoln" in the sentence *"Where is Lincoln in Logan County?"*, "Lincoln, IL" has the KG relationship "capital of" with Logan County while Lincoln, NE does not. The keyword "in" is associated with the relation "capital of" between two location entities, indicating that "Lincoln, IL" is correct, despite being the less popular candidate entity associated with "Lincoln". As patterns over pairs of entities with KG relations cover 23% of the tail examples, this is a more general reasoning pattern than consistency.
- **Type Affordance:** We define type affordance as the textual signals associated with a specific entity-type in natural language. For example, "Manhattan" is likely resolved to the cocktail rather than the burrough in the sentence *"He ordered a Manhattan."* due to the affordance that drinks, not locations, are "ordered". As affordance signals cover 76% of the tail examples, it is the most general reasoning pattern.

Required Structural Resources An NED system requires entity, relation, and type knowledge signals to learn these reasoning patterns. Entity knowledge is captured in unstructured text, while relation signals and type signals are readily available in structured knowledge bases such as Wikidata: from a sample of Wikipedia, 27% of all mentions and 23% of tail mentions participate in a relation, and 97% of all mentions and 92% of tail mentions are assigned some type in Wikidata. As these structural resources are readily available for all entities, they are useful for generalizing to the tail. A rare entity with a particular type or relation can leverage textual patterns learned from every other entity with that type or relation.

Given the input signals and reasoning patterns, the next key challenge is ensuring that the model combines the discriminative entity and more general relation and type signals that are useful for disambiguation.

3 Bootleg Architecture for Tail Disambiguation

We now describe our approach to leverage the reasoning patterns based on entity, relation, and type signals. We then present our new regularization scheme to inject inductive bias of when to use general versus discriminative reasoning patterns and our weak labeling technique to extract more signal from the self-supervision training data.

[5]Coverage numbers are calculated from representative slices of Wikidata that require each reasoning pattern. Additional details in Section 5.

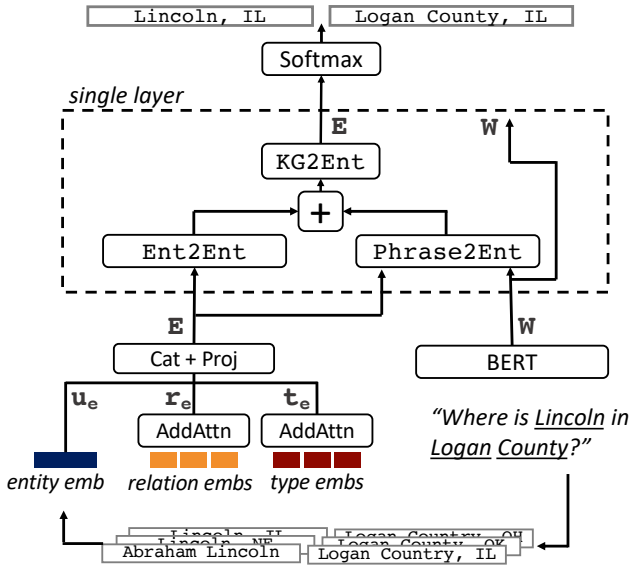

Figure 2: BOOTLEG's neural model. The entity, type, and relation embeddings are generated for each candidate and concatenated to form our entity representation matrix $\mathbf{E}$. This, together with our word embedding matrix $\mathbf{W}$, are inputs to BOOTLEG's `Ent2Ent`, `Phrase2Ent`, and `KG2Ent` modules which aim to encode the four reasoning patterns. The most likely candidate for each mention is returned.

3.1 Encoding the Signals

We first encode the structural signals—entities, KG relations and types—by mapping each to a set of embeddings.

- *Entity Embedding:* Each entity e is represented by a unique embedding $\mathbf{u}_e$.
- *Type Embedding:* Let $\mathcal{T}$ be the set of possible entity types. Given a known mapping from an entity e to its set $\{t_{e,1}, \ldots, t_{e,T} | t_{e,i} \in \mathcal{T}\}$ of T possible types, BOOTLEG assigns an embedding $\mathbf{t}_{e,i}$ to each type. Because an entity can have multiple types, we use an additive attention [3], AddAttn, to create a single type embedding $\mathbf{t}_e = \text{AddAttn}([\mathbf{t}_{e,1}, \ldots, \mathbf{t}_{e,T}])$. We further allow the model to leverage coarse named entity recognition types through a mention-type prediction module (see Appendix A for details). This coarse predicted type is concatenated with the assigned type to form $\mathbf{t}_e$.
- *Relation Embedding:* Let $\mathcal{R}$ represent the set of possible relationships any entity can participate in. Similar to types, given a mapping from an entity e to its set $\{r_{e,1}, \ldots, r_{e,R} | r_{e,i} \in \mathcal{R}\}$ of R relationships, BOOTLEG assigns an embedding $\mathbf{r}_{e,i}$ to each relation. Because an entity can participate in multiple relations, we use the additive attention to compute $\mathbf{r}_e = \text{AddAttn}([\mathbf{r}_{e,1}, \ldots, \mathbf{r}_{e,R}])$.

As in existing work [16, 40], given the input sentence of length N and set of M mentions, BOOTLEG generates for each mention m_i a set $\Gamma(m_i) = \{e_i^1, \ldots, e_i^K\}$ of K possible entity candidates that could be referred to by m_i. For each candidate and its associated types and relations, BOOTLEG uses a multi-layer perceptron $\mathbf{e} = \text{MLP}([\mathbf{u}_e, \mathbf{t}_e, \mathbf{r}_e])$ to generate a vector representation for each candidate entity, for each mention. We denote this entity matrix as $\mathbf{E} \in \mathbb{R}^{M \times K \times H}$, where H is the hidden dimension. We use BERT to generate contextual embeddings for each token in the input sentence. We denote this sentence embedding as $\mathbf{W} \in \mathbb{R}^{N \times H}$. $\mathbf{W}$ and $\mathbf{E}$ are passed to BOOTLEG's model architecture, described next.

3.2 BOOTLEG Model Architecture

The design goal of BOOTLEG is to capture the reasoning patterns by modeling textual signals associated with entities (for entity memorization), co-occurrences between entity types (for type consistency), textual signals associated with relations along with which entities are explicitly linked in the KG (for KG relations),

and textual signals associated with types (for type affordance). We design three modules to capture these design goals: a phrase memorization module, a co-occurrence memorization module, and a knowledge graph connection module. The model architecture is shown in Figure 2. We describe each module next.

Phrase Memorization Module We design the phrase memorization module, `Phrase2Ent`, to encode the dependencies between the input text and the entity, relation, and type embeddings. The purpose of this module is to learn textual cues for the entity memorization and type affordance patterns. It should also learn relation context for the KG relation pattern. It will, for example, allow the person type embedding to encode the association with the keyword "height". The module accepts as input $\mathbf{E}$ and $\mathbf{W}$ and outputs $\mathbf{E}_p = \text{MHA}(\mathbf{E}, \mathbf{W})$, where MHA is the standard multi-headed attention with a feed-forward layer and skip connections [48].

Co-occurrence Memorization Module We design the co-occurrence memorization module, `Ent2Ent`, to encode the dependencies between entities. The purpose of the `Ent2Ent` module is to learn textual cues for the type consistency pattern. The module accepts $\mathbf{E}$ and computes $\mathbf{E}_c = \text{MHA}(\mathbf{E})$ using self-attention.

Knowledge Graph (KG) Connection Module We design the KG module, `KG2Ent`, to collectively resolve entities based on pairwise connectivity features. Let $\mathbf{K}$ represent the adjacency matrix of a (possibly weighted) graph where the nodes are entities and an edge between e_i and e_j signifies that the two entities share some pairwise feature. Given $\mathbf{E}$, `KG2Ent` computes $\mathbf{E}_k = \text{softmax}(\mathbf{K} + w\mathbf{I})\mathbf{E} + \mathbf{E}$ where $\mathbf{I}$ is the identity and w is a learned scalar weight that allows Bootleg to learn to balance the original entity and its connections. This module allows for representation transfer between two related entities, meaning entities with a high-scoring representation will boost the score of related entities. The second computation acts as a skip connection between the input and output. In Bootleg, we allow the user to specify multiple `KG2Ent` modules: one for each adjacency matrix. The purpose of `KG2Ent` along with `Phrase2Ent` is to learn the KG relation pattern.

End-to-End The computations for one layer of Bootleg includes:

$$\begin{aligned}\mathbf{E}' &= \text{MHA}(\mathbf{E}, \mathbf{W}) + \text{MHA}(\mathbf{E}) \\ \mathbf{E}_k &= \text{softmax}(\mathbf{K} + w\mathbf{I})\mathbf{E}' + \mathbf{E}'\end{aligned}$$

where $\mathbf{E}_k$ is passed as the entity matrix to the next layer. After the final layer, Bootleg scores each entity by computing $\mathbf{S}_{dis} = \max(\mathbf{E}_k\mathbf{v}^T, \mathbf{E}'\mathbf{v}^T)$ with $\mathbf{S}_{dis} \in \mathbb{R}^{M \times K}$ and learned scoring vector $\mathbf{v} \in \mathbb{R}^H$. Bootleg then outputs the highest scoring candidate for each mention. This scoring treats $\mathbf{E}_k$ and $\mathbf{E}'$ as two separate predictions in an ensemble method, allowing the model to use collective reasoning from $\mathbf{E}_k$ when it achieves the highest scoring representation. If there are multiple `KG2Ent` modules, we use the average of their outputs as input to the next layer and, for scoring, take the maximum score across all outputs. For training, we use the cross-entropy loss of $\mathbf{S}$ to compute the disambiguation loss $\mathcal{L}_{dis}$.

3.3 Improving Tail Generalization

Regularization is the standard technique to encourage models to generalize, as models will naturally fit to discriminative features. However, we demonstrate that standard regularization is not effective when we want to leverage a combination of general and discriminative signals. We then present two techniques, regularization and weak labeling, to encourage Bootleg to incorporate general structural signals and learn general reasoning patterns.

3.3.1 Regularization

We hypothesize that Bootleg will over-rely on the more discriminative entity features compared to the more general type and relation features to lower training loss. However, tail disambiguation requires Bootleg to leverage the general features. Using standard regularization techniques, we evaluate three models which respectively use only type embeddings, only relation embeddings, and a combination of type, relation, and

entity embeddings. BOOTLEG's performance on unseen entities is 10 F1 points worse on the latter than each of the former two, suggesting that standard regularization is not sufficient when the signals operate at different granularities (details Table 9 in Appendix B).

We can improve tail performance if BOOTLEG leverages memorized discriminative features for popular entities and general features for rare entities. We achieve this by designing a new regularization scheme for the entity-specific embedding $\mathbf{u}$, which has two key properties: it is 2-dimensional and more popular entities are regularized less than less popular ones.

- *2-dimensional*: In contrast to 1-dimensional dropout, 2-dimensional regularization involves masking the full embedding. With probability $p(e)$, we set $\mathbf{u} = \mathbf{0}$ *before* the MLP layer; i.e., $\mathbf{e} = \text{MLP}([\mathbf{0}, \mathbf{t}_e, \mathbf{r}_e])$. Entirely masking the entity embedding in these cases, the model learns to disambiguate using the type and relation patterns, without entity knowledge.
- *Inverse Popularity*: We find in ablations (Appendix B) that setting $p(e)$ proportional to the power of the inverse of the entity e's popularity in the training data (i.e., the more popular the less regularized), gives us the best performance and improves by 13.6 F1 on unseen entities over standard regularization. In contrast, fixing $p(e)$ at 80% improves performance by over 11.3 F1 over standard regularization, and regularizing proportional to the power of popularity only improves performance by 3.8 F1 (details in Section 4).

The regularization scheme encourages BOOTLEG to use entity-specific knowledge when the entity is seen enough times to memorize entity patterns and encourages the use of generalizable patterns over the rare, highly masked, entities.

3.3.2 Weakly Supervised Data Labeling

We use Wikipedia to train BOOTLEG: we define a self-supervision task in which the internal links in Wikipedia are the gold entity labels for mentions during training. Although this dataset is large and widely used, it is often incomplete with an estimated 68% of named entities being unlabeled. Given the scale and the requirement that BOOTLEG be self-supervised, it is not feasible to hand-label the data. Our insight is that because Wikipedia is highly structured—most sentences on an entity's Wikipedia page refer to that entity via pronouns or alternative names—we can *weakly label* our training data [44] to label mentions. We use two heuristics for weak labeling: the first labels pronouns that match the gender of a person's Wikipedia page as references to that person, and the second labels known alternative names for an entity if the alternative name appears in sentences on the entity's Wikipedia page. Through weak labeling, we increase the number of labeled mentions in the training data by 1.7x across Wikipedia, and find this provides a 2.6 F1 lift on unseen entities (full results in Appendix B Table 11).

4 Experiments

We demonstrate that BOOTLEG (1) nearly matches or exceeds state-of-the-art performance on three standard NED benchmarks and (2) outperforms a BERT-based NED baseline on the tail. As NED is critical for downstream tasks that require the knowledge of entities, we (3) verify BOOTLEG's learned reasoning patterns can transfer by using them for a downstream task: using BOOTLEG's learned representations, we achieve a new SotA on the TACRED relation extraction task and improve performance on a production task at a major technology company by 8%. Finally, we (4) demonstrate that BOOTLEG can be sample-efficient by using only a fraction of its learned entity embeddings without sacrificing performance. We (5) ablate BOOTLEG to understand the impact of the structural signals and the regularization scheme on improved tail performance.

4.1 Experimental Setup

Wikipedia Data We define our knowledge base as the set of entities with mentions in Wikipedia (for a total of 5.3M entities). We allow each mention to have up to $K = 30$ possible candidates. As BOOTLEG is a sentence disambiguation system, we train on individual sentences from Wikipedia, where the anchor links and our weak labeling (Section 3.3) serve as mention labels.

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 for each metric on each benchmark.

Benchmark	Model	Precision	Recall	F1
KORE50	Hu et al. [24][7]	80.0	79.8	79.9
	BOOTLEG	**86.0**	**85.4**	**85.7**
RSS500	Phan et al. [40]	82.3	82.3	82.3
	BOOTLEG	**82.5**	**82.5**	**82.5**
AIDA	Févry et al. [16]	-	**96.7**	-
	BOOTLEG	96.9	**96.7**	96.8

Our candidate lists Γ are mined from Wikipedia anchor links and the "also known as" field in Wikidata. For each person, we further add their first and last name as aliases linking to that person. We use the mention boundaries provided in the Wikipedia data and generate candidates by performing a direct lookup in Γ.

We use Wikidata and YAGO knowledge graphs and Wikipedia to extract structural data about entity types and relations as input for BOOTLEG. Further details about data are in Appendix B.

Metrics We report micro-average F1 scores for all metrics over true anchor links in Wikipedia (not weak labels). We measure the torso and tail sets based on the number of times that an entity is the gold entity across Wikipedia anchors and weak labels, as this represents the number of times an entity is seen by BOOTLEG. For benchmarks, we also report precision and recall using the number of mentions extracted by BOOTLEG and the number of mentions defined in the data as denominators, respectively. The numerator is the number of correctly disambiguated mentions. For Wikipedia data experiments, we filter mentions such that (a) the gold entity is in the candidate set and (b) they have more than one possible candidate. The former is to decouple candidate generation from model performance for ablations.[6] The latter is to not inflate a model's performance, as all models are trivially correct when there is a single candidate.

Training For our main BOOTLEG model, we train for two epochs on Wikipedia sentences with a maximum sentence length of 100. For our benchmark model, we train for one epoch and additionally add a title embedding feature, a sentence co-occurrence KG matrix as another KG module, and a Wikipedia page co-occurrence statistical feature. Additional details about the models and training procedure are in Appendix B.

4.2 BOOTLEG Performance

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). Benchmark details are in Appendix B.

For AIDA, we first convert each document into a set of sentences where a sentence is the document title, a BERT SEP token, and the sentence. We find this is sufficient to encode document context into BOOTLEG. We fine-tune the pretrained BOOTLEG model on the AIDA training set with learning rate of 0.00007, 2 epochs, batch size of 16, and evaluating every 25 steps. We choose the test score associated with the best validation score.[8] In Table 1, we show that BOOTLEG achieves up to 5.8 F1 points higher than prior reported numbers on benchmarks.

Tail Performance To validate that BOOTLEG improves tail disambiguation, we compare against a baseline model from Févry et al. [16], which we refer to as NED-Base.[9] NED-Base learns entity embeddings by

[6] We drop only 1% of mentions from this filter.

[8] We use the standard candidate list from Pershina et al. [36] when comparing to existing systems for fine-tuning and inference for AIDA CoNLL-YAGO.

[9] As code for the model from Févry et al. [16] is not publicly available, we re-implemented the model. We used our candidate

Table 2: (top) We compare Bootleg to a BERT-based NED baseline (NED-Base) on validation sets of a Wikipedia dataset. We report micro-average F1 scores. All torso, tail, and unseen validation sets are filtered by the number of entity occurrences in the training data and such that the mention has more than one candidate.

Model	All Entities	Torso Entities	Tail Entities	Unseen Entities
NED-Base	85.9	79.3	27.8	18.5
Bootleg	**91.3**	**87.3**	**69.0**	**68.5**
Bootleg (Ent-only)	85.8	79.0	37.9	14.9
Bootleg (Type-only)	88.0	81.6	62.9	61.6
Bootleg (KG-only)	87.1	79.4	64.0	64.7
# Mentions	4,065,778	1,911,590	162,761	9,626

maximizing the dot product between the entity candidates and fine-tuned BERT-contextual representations of the mention.

NED-Base is successful overall on the validation achieving 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 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.

4.3 Downstream Evaluation

Relation Extraction Using the learned representations from Bootleg, we achieve the new state-of-the-art on TACRED, a standard relation extraction benchmark. TACRED involves identifying the relationship between a specified subject and object in an example sentence as one of 41 relation types (e.g., spouse) or no relation. Relation extraction is a well-suited for evaluating Bootleg because the substrings in the text can refer to many different entities, and the disambiguated entities impact the set of likely relations.

Given an example, we run inference with the Bootleg model to disambiguate named entities and generate the contextual Bootleg entity embedding matrix, which we feed to a simple Transformer architecture that uses SpanBERT [27] (details in Appendix C). We achieve a micro-average test F1 score of 80.3, which improves upon the prior state of the art—KnowBERT [38], which also uses entity-based knowledge—by 1.0 F1 points and the baseline SpanBERT model by 2.3 F1 points on TACRED-Revisited data (Table 3) ([53], Alt et al. [2]). We find that the Bootleg downstream model corrects errors made by the SpanBERT baseline, for example by leveraging entity, type, and relation information or recognizing that different textual aliases refer to the same entity (see Table 4).

generators and fine-tuned a pretrained BERT encoder rather than training a BERT encoder from scratch, as is done in Févry et al. [16]. We trained NED-Base on the same weak labelled data as Bootleg for 2 epochs.

Table 3: Test micro-average F1 score on revised TACRED dataset.

Validation Set	F1
Bootleg Model	**80.3**
KnowBERT	79.3
SpanBERT	78.0

Table 4: The following are examples of how the contextual entity representation from BOOTLEG, generated from entity, relation, and type signals, can help our downstream model. We provide the TACRED example, signals provided by BOOTLEG, as well our model and the baseline SpanBERT models' predictions.

TACRED Example	BOOTLEG Signals	Our Prediction	SpanBERT Prediction
Vincent Astor, like Marshall (subj), died unexpectedly of a heart attack (obj) in 1959 ... **Gold relation: Cause of Death**	Disambiguates "Marshall" to Thomas Riley Marshall and "heart attack" to myocardial infarction, which have the Wikidata relation "cause of death"	Cause of Death	No Relation
The International Water Management (obj) Institute or IWMI (subj) study said both .... **Gold relation: Alternate Names**	Disambiguates alias "International Water Management Institute" and its acronym, the alias "IWMI", to the same Wikidata entity	Alternate Names	No Relation

In studying the slices for which the BOOTLEG downstream model improves upon the baseline SpanBERT model, we rank TACRED examples in three ways: by the proportion of words where BOOTLEG disambiguates it as an entity, leverages Wikidata relations for the embedding, and leverages Wikidata types for the embedding. For each of these three, we report the gap between the SpanBERT model and Bootleg model's error rates on the examples with above-median proportion (more BOOTLEG signal) relative to the below-median proportion (less BOOTLEG signal). We find that the relative gap between the baseline and Bootleg error rates is larger on the slice above (with more BOOTLEG information) than below the median by 1.10x, 4.67x, and 1.35x respectively: with more BOOTLEG information, the improvement our SotA model provides over SpanBERT increases (more details in Appendix C).

Industry Use Case We additionally demonstrate how the learned entity embeddings from BOOTLEG provide useful information to a system at a large technology company that answers factoid queries such as *"How tall is the president of the United States?"*. We use BOOTLEG's embeddings in the Overton [45] system 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 include unseen entities. Per company policy, we report relative to the baseline rather than raw F1 score; for example, if the baseline F1 score is 80.0 and the subject F1 is 88.0, the relative quality is $88.0/80.0 = 1.1$. Table 5 shows that the use of BOOTLEG's embeddings consistently results in a positive relative quality, even over Spanish, French, and German, where improvements are most visible over tail entities.

4.4 Memory Usage

We explore the memory usage of BOOTLEG during inference and demonstrate that by only using the entity embeddings for the top 5% of entities, ranked by popularity in the training data, BOOTLEG reduces its

Table 5: Relative F1 quality of an Overton[45] 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

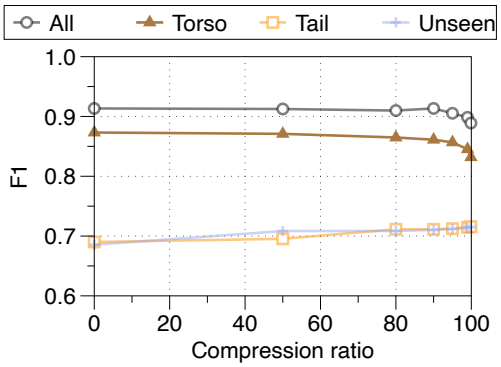

Figure 3: We show the error across all entities, torso entities, tail entities, and unseen entities as we decrease the number embeddings we use during inference, assigning the non-popular entities to a fixed unseen entity embedding. For example, a compression ratio of 80 means only the top 20% of entity embeddings are used, ranked by entity popularity.

embedding memory consumption by 95%, while sacrificing only 0.8 F1 points over all entities. We find that the 5.3M entity embeddings used in BOOTLEG consume the most memory, taking 5.2 GB of space while the attention network only consumes 39 MB (1.37B updated model parameters in total, 1.36B from embeddings). As BOOTLEG's representations must be used in a variety of downstream tasks, the representations must be memory-efficient: we thus study the effect of reducing BOOTLEG's memory footprint by only using the most popular entity embeddings.

Specifically, for the top k% of entities ranked by the number of occurrences in training data, we keep the learned entity embedding intact. For the remaining entities, we choose a random entity embedding for an unseen entity to use instead. Instead of storing 5.3M entity embeddings, we thus store $((100-k)/100)*5.3M$, which gives a compression ratio of $(100-k)$. Figure 3 shows performance for k of 100, 50, 20, 10, 5, 1, and 0.1. We see that when just the top 5% of entity embeddings are used, we only sacrifice 0.8 F1 points overall and in fact score 2 F1 points *higher* over the tail. We hypothesize that the increase in tail performance is due to the fact that the majority of mention candidates all have the same learned embedding, decreasing the amount of conflict among candidates from textual patterns.

4.5 Ablation Study

BOOTLEG To better understand the performance gains of BOOTLEG, we perform an ablation study over a subset of Wikipedia (data details explained in Appendix B). We train BOOTLEG with: (1) only learned entity embeddings (Ent-only), (2) only type information from type embeddings (Type-only), and (3) only knowledge graph information from relation embeddings and knowledge graph connections (KG-only). All model sizes are reported in Appendix B Table 10. In Table 2, we see that just using type or knowledge graph information leads to improvements on the tail of over 25 F1 points and on the unseen entities of over 46 F1 points compared to the Ent-only model. However, neither the Type-only nor KG-only model performs as well on any of the validation sets as the full BOOTLEG model. An interesting comparison is between Ent-only and NED-Base. NED-Base overall outperforms Ent-only due to the fine-tuning of BERT word embeddings. We attribute the high performance of Ent-only on the tail compared to NED-Base to our `Ent2Ent` module which allows for memorizing co-occurrence patterns over entities.

Regularization To understand the impact of our entity regularization function $p(e)$ on overall performance, we perform an ablation study on a sample of Wikipedia (explained in Appendix B). We apply (1) a fixed regularization set to a constant percent of 0, 20, 50 and 80, (2) a regularization function proportional to the power of the inverse popularity, and (3) the inverse of (2). Table 6 shows results over unseen entities (full results and details in Appendix B). We see that the fixed regularization of 80% achieves the highest F1 over the fixed regularizations of (1). The method that regularizes by inverse popularity achieves the highest

Table 6: We show the micro F1 score over unseen entities for a Wikipedia sample as we vary the entity regularization scheme $p(e)$. A scalar percent means a fixed regularization. InvPop (inverse poularity scheme) applies less regularization for more popular entities and Pop applies more regularization for more popular entities.

$p(e)$	0%	20%	50%	80%	Pop	InvPop
Unseen Entities	48.6	52.5	57.7	59.9	52.4	**62.2**

overall F1. We further see that the scheme where popular entities are more regularized sees a drop of 9.8 F1 points in performance compared to the inverse popularity scheme.

5 Analysis

We have shown that Bootleg excels on benchmark tasks and that Bootleg's learned patterns can transfer to non-NED tasks. We now verify whether the defined entity, type consistency, KG relation, and affordance reasoning patterns are responsible for these results. We evaluate each over a representative slice of the Wikipedia validation set that exemplifies one of the reasoning patterns and present the results from each ablated model (Table 7).

- **Entity** To evaluate whether Bootleg captures factual knowledge about entities in the form of textual entity cues, we consider the slice of 28K overall, 5K tail examples where the gold entity has no relation or type signals available.
- **Type Consistency** To evaluate whether Bootleg captures consistency patterns, we consider the slice of 312K overall, 19K tail examples that contain a list of three or more sequential distinct gold entities, where all items in the list share at least one type.
- **KG Relation** To evaluate whether Bootleg captures KG relation patterns, we consider the slice of 1.1M overall, 37K tail examples for which the gold entities are connected by a known relation in the Wikidata knowledge graph.
- **Type Affordance** To evaluate whether Bootleg captures affordance patterns, we consider a slice where the sentence contains keywords that are afforded by the type of the gold entity. We mine the keywords afforded by a type by taking the 15 keywords that receive the highest TF-IDF scores over training examples with that type. This slice has 3.4M overall, 124K tail examples.

Pattern Analysis For the slice representing each reasoning pattern, we find that Bootleg provides a lift over the Entity-only and NED-Base models, especially over the tail. We find that Bootleg generally combines the entity, relation, and type signals effectively, performing better than the individual Entity-only, KG-only, and Type-only models, although the KG-only model performs well on the KG relation slice. The lift from Bootleg across slices indicates the model's ability to capture the reasoning required for the slice. We provide additional details in Appendix D.

Error Analysis We next study the errors made by Bootleg and find four key error buckets.

- **Granularity** Bootleg struggles with *granularity*, predicting an entity that is too general or too specific compared to the gold entity (example in Table 8). Considering the set of examples where the predicted entity is a Wikidata *subclass* of the gold entity or vice versa, Bootleg predicts a too general or specific entity in 12% of overall and 7% of tail errors.
- **Numerical** Bootleg struggles with entities containing numerical tokens, which may be due to the fact that the BERT model represents some numbers with sub-word tokens and is known to not perform as well for numbers as other language models [49] (example in Table 8). To evaluate examples requiring

Table 7: We report the *Overall/Tail* F1 scores across each ablation model for a slice of data that exemplifies a reasoning pattern. Each slice is representative but may not cover every example that contains the reasoning pattern.

Model	Entity	Type Consistency	KG Relation	Type Affordance
NED-Base	59/29	84/29	91/30	87/28
Bootleg	**66/47**	**95/85**	**98**/92	**93/73**
Bootleg (Ent-only)	59/31	87/45	90/42	87/39
Bootleg (Type-only)	53/44	93/80	93/69	90/66
Bootleg (KG-only)	40/29	92/79	97/**93**	89/68
% Coverage	0.7%/3.3%	8%/12%	27%/23%	84%/76%

reasoning over numbers, we consider the slice of data where the entity title contains a year, as this is the most common numerical feature in a title. This slice covers 14% of overall and 25% of tail errors.

- **Multi-Hop** There is room for improvement in multi-hop reasoning. In the example shown Table 8, none of the present gold entities—Stillwater Santa Fe Depot, Citizens Bank Building (Stillwater, Oklahoma), Hoke Building (Stillwater, Oklahoma), or Walker Building (Stillwater, Oklahoma)—are directly connected in Wikidata; however, they share connections to the entity "Oklahoma". This indicates that the correct disambiguation is Citizens Bank Building (Stillwater, Oklahoma), not Citizens Bank Building (Burnsville, North Carolina). To evaluate examples requiring 2-hop reasoning, we consider examples where none of the present entities are directly linked in the KG, but a present pair connects to a different entity that is not present in the sentence. We find this occurs in 6% of overall and 7% of tail errors. This type of error represents a fundamental limitation of Bootleg as we do not encode any form of multi-hop reasoning over a KG in Bootleg. Our KG information only encodes single-hop patterns (i.e., direct connections).
- **Exact Match** Bootleg struggles on several examples in which the exact entity title is present in the text. Considering examples where the BERT Baseline is correct but Bootleg is incorrect, in 28% of the examples, the textual mention is an exact match of the entity title. Further, 32% of the examples contain a keyword from the entity title that Bootleg misses (example in Table 8). We attribute this decrease in performance to Bootleg's regularization. This mention-to-entity similarity would need to be encoded in Bootleg's entity embedding, but the regularization encourages Bootleg to not use entity-level information.

6 Related Work

We discuss related work in terms of both NED and the broader picture of self-supervised models and tail data. Standard, pre-deep-learning approaches to NED have been rule-based [1] or leverage statistical techniques and manual feature engineering to filter and rank candidates [50]. For example, link counts and similarity scores between entity titles and mention are two such features [12]. These systems tend to be hard to maintain over time, with the work of Petasis et al. [37] 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 the new standard (see Mudgal et al. [32] for a high-level overview of deep learning approaches to entity disambiguation and entity matching problems). The most recent state-of-the-art models generally rely on deep contextual word embeddings with entity embeddings [16, 46, 51]. As we showed in Table 2, these models perform well over popular entities, but struggle to resolve the tail. Jin et al. [26] and Hoffart et al. [23] study disambiguation at the tail, and both rely on phrase-based language models for feature extraction. Unlike our work, they do not fuse type or knowledge graph information for disambiguation.

Table 8: We identify four key error buckets for BOOTLEG: granularity, numerical errors, multi-hop reasoning, and missed exact matches. We provide a Wikipedia example, the gold entity, and BOOTLEG's predicted entity for each example.

Error	Wikipedia Example	BOOTLEG Prediction	Gold Entity
Granularity	*Posey is the recipient of a Golden Globe Award nomination, a **Satellite Award** nomination and two Independent Spirit Award nominations.*	Satellite Awards	Satellite Award for Best Actress – Motion Picture
Numerical	*He competed in the individual road race and team time trial events at the **1976 Summer Olympics**.*	Cycling at the 1960 Summer Olympics – 1960 Men's Road Race	Cycling at the 1976 Summer Olympics – 1976 Men's Road Race
Multi-hop	*Other nearby historic buildings include the Santa Fe Depot, the **Citizens Bank Building**, the Hoke Building, the Walker Building, and the Courthouse*	Citizens Bank Building (Burnsville, North Carolina)	Citizens Bank Building (Stillwater, Oklahoma)
Exact Match	*According to the **Nielsen Media Research**, the episode was watched by 469 million viewers...*	Nielsen ratings	Nielsen Media Research

Disambiguation with Types Similar to our work, recent approaches have found that type information can be useful for entity disambiguation [9, 14, 21, 31, 43, 55]. Dredze et al. [14] use predicted coarse-grained types as entity features into a SVM classifier. Chen et al. [9] models type information as local context and integrates a BERT contextual embedding into the model from [17]. Raiman and Raiman [43] learns its own type systems and performs disambiguation through type prediction alone (essentially capturing the type affordance pattern). Ling et al. [31] demonstrate that the 112-type FIGER type ontology could improve entity disambiguation, and the LATTE framework [55] uses multi-task learning to jointly perform type classification and entity disambiguation on biomedical data. Gupta et al. [21] adds both an entity-level and mention-level type objective using type embeddings embeddings. We build on these works using fine and coarse-grained entity-level type embeddings and a mention-level type prediction task.

Disambiguation with Knowledge Graphs Several recent works have also incorporated (knowledge) graph information through graph embeddings [35], co-occurrences in the Wikipedia hyperlink graph [42], and the incorporation of latent relation variables [30] to aid disambiguation. Cetoli et al. [8] and Mulang et al. [33] incorporate Wikidata triples as context into entity disambiguation by encoding triples as textual phrases (e.g., "<subject> <predict> <object>") to use as additional inputs, along with the original text to disambiguate, into a language model. In BOOTLEG, the Wikidata connections through the `KG2Ent` module allow for collective resolution and are not just additional features.

Entity Knowledge in Downstream Tasks The works of Broscheit [6], Peters et al. [38], Poerner et al. [41], Zhang et al. [54] all try to add entity knowledge into a deep language model to improve downstream natural language task performance. Peters et al. [38], Poerner et al. [41], Zhang et al. [54] incorporate pretrained entity embeddings and finetune either on a the standard masked sequence-to-sequence prediction task or combined with an entity disambiguation/linking task.[10] On the other hand, Broscheit [6] trains its own entity embeddings. Most works, like BOOTLEG, see lift from incorporating entity representations in the downstream tasks.

[10]Entity disambiguation refers to when the mentions are pre-detected in text. Entity linking includes the mention detection phase. In BOOTLEG, we focus on the entity disambiguation task.

Wikipedia Weak Labelling Although uncommon, Broscheit [6], De Cao et al. [13], Ghaddar and Langlais [19], Nothman et al. [34] all apply some heuristic weak labelling techniques to increase link coverage in Wikipedia for either entity disambiguation or named entity recognition. All methods generally rely on finding known surface forms for entities and labelling those in the text. Bootleg is the first to investigate the lift from incorporating weakly labelled Wikipedia data over the tail.

Self-Supervision and the Tail The works of Tata et al. [47], Chung et al. [11], Ilievski et al. [25], and Chung et al. [10] 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. In particular, Ilievski et al. [25] studies the tail in NED and encourages the use of separate head and tail subsets of data. From a broader perspective of natural language systems and generalizability, Ettinger et al. [15] highlights that many NLP systems are brittle in the face of tail linguistic patterns. Bootleg builds off this work, investigating the tail with respect to NED and demonstrating the generalizable reasoning patterns over structural resources can aid tail disambiguation.

7 Conclusion

We present Bootleg, a state-of-the-art NED system that is explicitly grounded in a principled set of reasoning patterns for disambiguation, defined over entities, types, and knowledge graph relations. The contributions of this work include the characterization and evaluation of core reasoning patterns for disambiguation, a new learning procedure to encourage the model to learn the patterns, and a weak supervision technique to increase utilization of the training data. We find that Bootleg improves over the baseline SotA model by over 40 F1 points on the tail of Wikipedia. Using Bootleg's entity embeddings for a downstream relation extraction task improves performance by 1.0 F1 points, and Bootleg's representations lead to an 8% lift on highly optimized production tasks at a major technology company. We hope this work inspires future research on improving tail performance by incorporating outside knowledge in deep models.

Acknowledgements: We thank Jared Dunnmon, Dan Fu, Karan Goel, Sarah Hooper, Monica Lam, Fred Sala, Nimit Sohoni, and Silei Xu for their valuable feedback and Pallavi Gudipati for help with experiments. We gratefully acknowledge the support of DARPA under Nos. FA86501827865 (SDH) and FA86501827882 (ASED); NIH under No. U54EB020405 (Mobilize), NSF under Nos. CCF1763315 (Beyond Sparsity), CCF1563078 (Volume to Velocity), and 1937301 (RTML); ONR under No. N000141712266 (Unifying Weak Supervision); the Moore Foundation, NXP, Xilinx, LETI-CEA, Intel, IBM, Microsoft, NEC, Toshiba, TSMC, ARM, Hitachi, BASF, Accenture, Ericsson, Qualcomm, Analog Devices, the Okawa Foundation, American Family Insurance, Google Cloud, Swiss Re, the HAI-AWS Cloud Credits for Research program, and members of the Stanford DAWN project: Teradata, Facebook, Google, Ant Financial, NEC, VMWare, and Infosys. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of DARPA, NIH, ONR, or the U.S. Government.

References

[1] John Aberdeen, John D Burger, David Day, Lynette Hirschman, David D Palmer, Patricia Robinson, and Marc Vilain. Mitre: Description of the alembic system as used in met. In *TIPSTER TEXT PROGRAM PHASE II: Proceedings of a Workshop held at Vienna, Virginia, May 6-8, 1996*, pages 461–462, 1996.

[2] Christoph Alt, Aleksandra Gabryszak, and Leonhard Hennig. Tacred revisited: A thorough evaluation of the tacred relation extraction task. In *ACL*, 2020.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Michael S Bernstein, Jaime Teevan, Susan Dumais, Daniel Liebling, and Eric Horvitz. Direct answers for search queries in the long tail. In *SIGCHI*, 2012.

[5] Terra Blevins and Luke Zettlemoyer. Moving down the long tail of word sense disambiguation with gloss-informed biencoders. *arXiv preprint arXiv:2005.02590*, 2020.

[6] Samuel Broscheit. Investigating entity knowledge in bert with simple neural end-to-end entity linking. *arXiv preprint arXiv:2003.05473*, 2020.

[7] 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.

[8] Alberto Cetoli, Mohammad Akbari, Stefano Bragaglia, Andrew D O'Harney, and Marc Sloan. Named entity disambiguation using deep learning on graphs. *arXiv preprint arXiv:1810.09164*, 2018.

[9] Shuang Chen, Jinpeng Wang, Feng Jiang, and Chin-Yew Lin. Improving entity linking by modeling latent entity type information. *arXiv preprint arXiv:2001.01447*, 2020.

[10] Yeounoh Chung, Peter J Haas, Eli Upfal, and Tim Kraska. Unknown examples & machine learning model generalization. *arXiv preprint arXiv:1808.08294*, 2018.

[11] 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.

[12] Silviu Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, 2007.

[13] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904*, 2020.

[14] Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 277–285, 2010.

[15] Allyson Ettinger, Sudha Rao, Hal Daumé III, and Emily M Bender. Towards linguistically generalizable nlp systems: A workshop and shared task. *arXiv preprint arXiv:1711.01505*, 2017.

[16] Thibault Févry, Nicholas FitzGerald, Livio Baldini Soares, and Tom Kwiatkowski. Empirical evaluation of pretraining strategies for supervised entity linking. In *AKBC*, 2020.

[17] Octavian-Eugen Ganea and Thomas Hofmann. Deep joint entity disambiguation with local neural attention. *arXiv preprint arXiv:1704.04920*, 2017.

[18] 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.

[19] Abbas Ghaddar and Philippe Langlais. Winer: A wikipedia annotated corpus for named entity recognition. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 413–422, 2017.

[20] Ben Gomes. Our latest quality improvements for search. `https://blog.google/products/search/our-latest-quality-improvements-search/`, 2017.

[21] Nitish Gupta, Sameer Singh, and Dan Roth. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2681–2690, 2017.

[22] 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 *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics, 2011.

[23] Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. Kore: keyphrase overlap relatedness for entity disambiguation. In *CIKM*, 2012.

[24] 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.

[25] Filip Ilievski, Piek Vossen, and Stefan Schlobach. Systematic study of long tail phenomena in entity linking. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 664–674, 2018.

[26] Yuzhe Jin, Emre Kiciman, Kuansan Wang, and Ricky Loynd. Entity linking at the tail: Sparse signals, unknown entities and phrase models. In *WSDM '14 Proceedings of the 7th ACM international conference on Web search and data mining*, pages 453–462. ACM, February 2014.

[27] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. SpanBERT: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529*, 2019.

[28] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[29] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[30] Phong Le and Ivan Titov. Improving entity linking by modeling latent relations between mentions. *arXiv preprint arXiv:1804.10637*, 2018.

[31] Xiao Ling, Sameer Singh, and Daniel S Weld. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics*, 3:315–328, 2015.

[32] 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.

[33] Isaiah Onando Mulang, Kuldeep Singh, Chaitali Prabhu, Abhishek Nadgeri, Johannes Hoffart, and Jens Lehmann. Evaluating the impact of knowledge graph contexton entity disambiguation models. *arXiv preprint arXiv:2008.05190*, 2020.

[34] Joel Nothman, James R Curran, and Tara Murphy. Transforming wikipedia into named entity training data. In *Proceedings of the Australasian Language Technology Association Workshop 2008*, pages 124–132, 2008.

[35] Alberto Parravicini, Rhicheek Patra, Davide B Bartolini, and Marco D Santambrogio. Fast and accurate entity linking via graph embedding. In *Proceedings of the 2nd Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, pages 1–9, 2019.

[36] Maria Pershina, Yifan He, and Ralph Grishman. Personalized page rank for named entity disambiguation. In *NAACL*, 2015.

[37] 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.

[38] Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1005.

[39] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vassilis Plachouras, Tim Rocktäschel, et al. Kilt: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252*, 2020.

[40] 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.

[41] N Poerner, U Waltinger, and H Schütze. E-bert: Efficient-yet-effective entity embeddings for bert. *arXiv preprint arXiv:1911.03681*, 2019.

[42] Priya Radhakrishnan, Partha Talukdar, and Vasudeva Varma. Elden: Improved entity linking using densified knowledge graphs. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1844–1853, 2018.

[43] Jonathan Raphael Raiman and Olivier Michel Raiman. Deeptype: multilingual entity linking by neural type system evolution. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[44] 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.

[45] Christopher Ré, Feng Niu, Pallavi Gudipati, and Charles Srisuwananukorn. Overton: A data system for monitoring and improving machine-learned products. In *CIDR*, 2020.

[46] Hamed Shahbazi, Xiaoli Z Fern, Reza Ghaeini, Rasha Obeidat, and Prasad Tadepalli. Entity-aware elmo: Learning contextual entity representation for entity disambiguation. *arXiv preprint arXiv:1908.05762*, 2019.

[47] Sandeep Tata, Vlad Panait, Suming Jeremiah Chen, and Mike Colagrosso. Itemsuggest: A data management platform for machine learned ranking services. In *CIDR*, 2019.

[48] 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.

[49] 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.

[50] Vikas Yadav and Steven Bethard. A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv:1910.11470*, 2019.

[51] Ikuya Yamada and Hiroyuki Shindo. Pre-training of deep contextualized embeddings of words and entities for named entity disambiguation. *arXiv preprint arXiv:1909.00426*, 2019.

[52] Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. HYENA: Hierarchical type classification for entity names. In *Proceedings of COLING 2012: Posters*, pages 1361–1370, Mumbai, India, December 2012. The COLING 2012 Organizing Committee.

[53] Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 35–45, 2017.

[54] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. Ernie: Enhanced language representation with informative entities. *arXiv preprint arXiv:1905.07129*, 2019.

[55] Ming Zhu, Busra Celikkaya, Parminder Bhatia, and Chandan K. Reddy. Latte: Latent type modeling for biomedical entity linking. In *AAAI*, 2020.

[56] Yuke Zhu, Joseph J Lim, and Li Fei-Fei. Knowledge acquisition for visual question answering via iterative querying. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1154–1163, 2017.

A Extended Model Details

We now provide additional details about the model introduced in Section 3. We first describe our type prediction module and then describe the added entity positional encoding.

Type Prediction To allow the model to further infer the correct types for an entity, especially when the entity does not have a preassigned type, we add a coarse *mention* type prediction task given the mention embedding. Given a mention m and a coarse type embedding matrix $\mathbf{T}$, the task is to assign a coarse type embedding for the mention m; i.e., determine $\mathbf{t}_m$. We do so by adding the first and last token of the mention from $\mathbf{W}$ to generate a contextualized mention embedding $\mathbf{m}$. We predict the coarse type of the mention $\hat{\mathbf{t}}_m$ by computing

$$\mathbf{S}_{type} = \text{softmax}(\text{MLP}(\mathbf{m}))$$
$$\hat{\mathbf{t}}_m = \mathbf{S}_{type}\mathbf{T}$$

where $\mathbf{S}_{type}$ generates a distribution over coarse types. For each entity candidate of m, $\hat{\mathbf{t}}_\mathbf{m}$ gets concatenated to the other type embedding $\mathbf{t}_e$ before the MLP. This is supervised by minimizing the cross entropy between $\mathbf{S}_{type}$ and the true coarse type for the gold entity, generating a type prediction loss $\mathcal{L}_{type}$. When performing type prediction, our overall loss is $\mathcal{L}_{dis} + \mathcal{L}_{type}$.

Position Encoding We need Bootleg to be able to reason over absolute and relative positioning of the words in the sentence and the mentions. For example, in the sentence *"Where is America in Indiana?"*, "America" refers to the city in Indiana, not the United States. In the sentence *"Where is Indiana in America?"*, "America" refers to the United States. The relative position of "Indiana", "in", and "America" signals the correct answer.

To achieve this signaling, we add the sin positional encoding from Vaswani et al. [48] to $\mathbf{E}$ before it is passed to our neural model. Specifically, for mention m, we concatenate of the positional encoding of the first and last token of m, project the concatenation to dimension H, and add it to each of m's K candidates in $\mathbf{E}$. As we use BERT word embeddings for W, the positional encoding is already added to words in the sentence.

B Extended Results

We now give the details of our experimental setup and training. We then give extended results over the regularization scheme and model ablations. Lastly, we extend our error analysis to validate Bootleg's ability to reason over the four patterns.

B.1 Evaluation Data

Wikipedia Datasets We use two main datasets to evaluate Bootleg.

- **Wikipedia**: we use the November 2019 dump of Wikipedia to train Bootleg. We use the set of entities that are linked to in Wikipedia for a total of 3.3M entities. After weak labelling, we have a total of 5.7M sentences.
- **Wikipedia Subset**: we use a subset of Wikipedia for our micro ablation experiments over regularization parameters. We generate this subset by taking all sentences where at least one mention is a mention from the KORE50 disambiguation benchmark. Our set of entities is all entities and entity candidates referred to by mentions in this subset of sentences. We have a total of 370,000 entities and 520,000 sentences.

For our Wikipedia experiments, we use a 80/10/10 train/test/dev split by Wikipedia pages, meaning all sentences for a single Wikipedia page get placed into one of the splits. For our benchmark model, we use a 96/2/2 train/test/dev split over sentences to allow our model to learn as much as possible from Wikipedia for our benchmark tasks.

Benchmark Datasets We use three benchmark NED datasets. Following standard procedure [17], we only consider mentions whose linked entities appear in Wikipedia. The datasets are summarized as follows:

- **KORE50**: KORE50 [23] represents difficult-to-disambiguate sentences and contains 144 mentions to disambiguate. Note, as of the Nov 2019 Wikipedia dump, one mention in the 144 does not have a Wikipedia page. Although it is standard to remove mentions that do not link to an entity in Wikipedia, to be comparable to other methods, we measure with 144 mentions, not 143.
- **RSS500**: RSS500 [18] is a dataset of news sentences and contains 520 mentions (4 of the mentions did not have entities in $\mathcal{E}$).
- **AIDA CoNLL-YAGO**: AIDA CoNLL-YAGO [22] is a document-based news dataset containing $4,485$ test mentions, $4,791$ validation set mentions, and $18,541$ training mentions. As Bootleg is a sentence-level NED system, we create sentences from documents following the technique from Févry et al. [16] where we concatenate the title of the document to the beginning of each sentence.

To improve the quality of annotated mention boundaries in the benchmarks, we follow the technique of Phan et al. [40] and allow for mention boundary expansion using a standard off-the-shelf NER tagger.[11] For candidate generation, as aliases may not exist in Γ, we gather possible candidates by looking at n-grams in descending order of length and determine the top 30 by measuring the similarity of the proper nouns in the example sentence to each candidate's Wikipedia page text.

Structural Resources The last source of input data to Bootleg is the structural resources of types and knowledge graph relations. We extract relations from Wikidata knowledge graph triples. For our pairwise KG adjacency matrix used in `KG2Ent`, we require the subject and object to be in $\mathcal{E}$. For our relation embeddings, we only require the subject be in $\mathcal{E}$ as our goal is to extract all relations an entity participates in independent of the other entities in the sentence. We have a total of 1,197 relations.

We use two different type sources to assign types to entities—Wikidata types and HYENA types [52]—and use coarse HYENA types for type prediction. The Wikidata types are generated from Wikidata's "instance of", "subclass of", and "occupation" relationships. The "occupation" types are used to improve disambiguation of people, which otherwise only receive "human" types in Wikidata. We filter the set of Wikidata types to be only those occurring 100 or more times in Wikipedia, leaving 27K Wikidata types in total. The HYENA type hierarchy has 505 types derived from the YAGO type hierarchy. We also use the coarsest HYENA type for an entity as the gold type for type prediction. There are 5 coarse HYENA types of person, location, organization, artifact, event, and miscellaneous.

B.2 Training Details

Model Parameters We run three separate models of Bootleg: two on our full Wikipedia data (one for the ablation and one for the benchmarks) and one on our micro data. For all models we use 30 candidates for each mention and incorporate the structural resources discussed above. We set $T = 3$ and $R = 50$ for the number of types and relations assigned to each entity.

For the models trained on our full Wikipedia data, we set the hidden dimension to 512, the dimension of $\mathbf{u}$ to 256, and the dimension of all other type and relation embeddings to 128. For our models trained on our micro dataset, we set the hidden dimension to 256, the dimension of $\mathbf{u}$ to 256, and the dimension of all other type and relation embeddings to 128.

The final differences to discuss are between the benchmark model and ablation model over all of Wikipedia. To make the best performing model for benchmarks, we add two additional additional features we found improved performance:

- We use an additional `KG2Ent` module in addition to an adjacency matrix indicating if two entities are connected in Wikidata. We add a matrix containing the log of the number of times two entities occur in a sentence together in Wikipedia. If they co-occur less than 10 times, the weight is 0. We found this helped the model better learn entity co-occurrences from Wikipedia.
- We allow our model to use additional entity-based features to be concatenated into our final $\mathbf{E}$ matrix. We add two features. The first is the average BERT WordPiece embeddings of the title of an entity. This is

[11] We use the spaCy NER tagger from `https://spacy.io/`

Table 9: (top) We compare Bootleg to a BERT-based NED baseline (NED-Base) on validation sets of our micro Wikipedia dataset and ablate Bootleg by only training with entity, type, or knowledge graph data. We further ablate (bottom 8 rows) the regularization schemes for the entity embeddings for Bootleg.

Model	All Entities	Torso Entities	Tail Entities	Unseen Entities
NED-Base	90.2	91.6	50.5	21.5
Bootleg (Ent-only)	89.1	89.0	48.3	15.5
Bootleg (Type-only)	91.6	90.4	65.9	56.8
Bootleg (KG-only)	91.8	90.8	65.3	58.6
Bootleg ($p(e) = 0\%$)	92.5	92.3	67.7	48.6
Bootleg ($p(e) = 20\%$)	92.8	92.5	68.9	52.5
Bootleg ($p(e) = 50\%$)	**92.9**	**92.7**	70.1	57.7
Bootleg ($p(e) = 80\%$)	92.8	92.2	69.5	59.9
Bootleg (InvPopLog)	92.7	91.9	69.7	61.1
Bootleg (InvPopPow)	92.8	92.3	**70.5**	**62.2**
Bootleg (InvPopLin)	92.6	91.8	69.7	61.0
Bootleg (PopPow)	**92.9**	92.5	68.9	52.4
# Mentions	96,237	37,077	11,087	2,810

similar to improving tail generalization by embedding a word definition in word sense disambiguation [5]. This allows the model to better capture textual cues indicating the correct entity. We also append a 1-dimensional feature of how many other entities in the sentence appear on the entity's Wikipedia page. This increases the likelihood of an entity that has more connection to other candidates in the sentence. We empirically find that using the page co-occurrences as an entity feature rather than as a `KG2Ent` module performs similarly and reduces the runtime.

Further, our benchmark model uses a fixed regularization scheme of 80% which did not hurt benchmark performance and training was marginally faster than the inverse popularity scheme. We did not use these features for ablations as we wanted a clean study of the model components as described in Section 3 with respect to the reasoning patterns.

Training We initialize all entity embeddings to the same vector to reduce the impact of noise from unseen entities receiving different random embeddings. We use the Adam optimizer [28] with a learning rate of 0.0001 and a dropout of 0.1 in all feedforward layers, 16 heads in our attention modules, and we freeze the BERT encoder stack. Note for the NED-Base model, we do not freeze the encoder stack to be consistent with Févry et al. [16].

For the models trained on all of Wikipedia, we use a batch size of 512 and train for 1 epoch for the benchmark model and 2 epochs for the ablation models on 8 NVIDIA V100 GPUs. For our micro data model, we use a batch size of 96 and train for 8 epochs on a NVIDIA P100 GPU.

B.3 Extended Ablation Results

Ablation Model Size Table 10 reports the model sizes of each of the five ablation models from Table 2. As we finetuned the BERT language model in NED-Base (to be consistent with Févry et al. [16]) but do not do so in Bootleg, we do not count the BERT parameters in our reported sizes to be comparable.

Regularization We now present the extended results of our regularization and weak labelling ablations over our representative micro dataset. Table 9 gives full ablations over a variety of regularization techniques. As in Table 2, we include results from models using only the entity, type, or relation information, in addition to the BERT and Bootleg models.

Table 10: We report the model sizes in MB of each of the five ablation models: NED-Base, Bootleg, Bootleg (Ent-Only), Bootleg (KG-Only), and Bootleg (Type-Only).

Model	NED-Base	Bootleg	Ent-Only	Type-Only	KG-Only
Embedding Size (MB)	5,186	5,201	5,186	13	1
Network Size (MB)	4	39	35	38	34
Total Size (MB)	5,190	5,240	5,221	51	35

Table 11: We report Bootleg trained with versus without weak labelling on our micro Wikipedia dataset. The slices defined by gold anchor counts (pre-weak labelling). We use the InvPopPow regularization for both.

Model	All Entities	Torso Entities	Tail Entities	Unseen Entities
Bootleg	92.8	92.6	**70.5**	**63.3**
Bootleg (No WL)	**93.3**	**93.1**	70.2	60.7
# Mentions	96,237	36,904	11,541	3,146

We report the results of inverse popularity regularization based on three different functions that map the the curve of entity counts in training to the regularization value. For each function, we fix that entities with a frequency 1 receive a regularization value of 0.95 while entities with a frequency of 10,000 receive a value of 0.05 and assign intermediate values to generate a linear, logarithmic, and power curve that applies less regularization for more popular entities. The regularization reported in Table 6 uses a power law function ($f(x) = 0.95(x^{-0.32})$). We also report in Table 9 a linear function ($f(x) = -0.00009x + 0.9501$) and a logarithmic function ($f(x) = -0.097\log(x) + 0.96$). Each regularization function is set to a range of 0.05 to 0.95. We leave it as future work to explore other varied regularization functions.

Table 9 shows similar trends as reported in Section 4 that Bootleg with all sources of information and the power law inverse regularization performs best over the tail and the unseen entities. We do see that the model trained with a fixed regularization of 0.5 performs marginally better on the torso and over all entities, likely because this involves less regularization over those entity embeddings, allowing it to better leverage the memorized entity patterns, while also leveraging some type and relational information (as shown by its improved performance over a lower fixed regularization). This model, however, performs 4.5 F1 points worse over unseen entities than the best model.

Weak Labeling Table 11 shows Bootleg's results with the inverse power law regularization with and without weak labelling. For this ablation, we define our set of torso, tail, and unseen entities by counting entity occurrence *before* weak labelling to have a better understanding as to the lift from adding weak labelling (rather than the drop without it).

We see that weak labelling provides a lift of 2.6 F1 points over unseen entities and 0.3 F1 points over tail entities. Surprisingly, without weak labeling, Bootleg performs 0.5 F1 points better on torso entities. We hypothesize this occurs because the noisy weakly labels increase the amount available signals for Bootleg to learn consistency patterns for rarer entities—noisy signals are better than no signals—however, popular entities have enough less-noisy gold labels in the training data, so the noise from weak labels may create conflicting signals that hurt performance.

To validate this hypothesis, we see that overall, counting both true and weakly labelled mentions, 4% of mentions without weak labeling share the same types as at least one other mention in the sentence while 14% of mentions with weak labelling do. Our model predicts a consistent answer only 4% of the time without weak labeling compared to 13% of the time with weak labeling. Note this is a slightly higher coverage numbers than reported in Section 5 as we are using a weaker form of consistency—two mentions in the sentence share the same type independent of position and ordering—and are including weakly labelled mentions. This

indicates consistency is a significantly more useful pattern with weak labelling, and our model predicts more consistent answers with weak labelling than without. Over the torso with weak labelling, we find that 14% of the errors across all mentions (weak labelled and anchor) are when Bootleg uses consistency reasoning, but the correct answer does not follow the consistency pattern. Without weak labelling, only 5% of the errors are due to consistency.

C Extended Downstream Details

We now provide additional details of our SotA TACRED model, which uses Bootleg embeddings.

Input We use the revisited TACRED dataset [2]: each example includes text and subject and object positions in the text. The task involves extracting the relation between the subject and object. There are 41 potential relations as well as a "no relation" option. The other features we use as inputs are NER, POS tags, and contextual Bootleg embeddings for entities that Bootleg disambiguates in the sentence.

Bootleg Model As TACRED does not come with existing mention boundaries, we perform mention extraction by searching over n-grams, from longest to shortest, in the sentence and extract those that are known mentions in Bootleg's candidate maps. We use the same Bootleg model from our ablations with entity, KG, and type information except with the addition of fine-tuned BERT word embeddings. For efficiency, we train on a subset of Wikipedia training data relevant to TACRED. To obtain the relevant subset, we take Wikipedia sentences containing entities extracted during candidate generation from a uniform sample of TACRED data; i.e., entities in the candidate lists of detected mentions from a uniform TACRED sample. The contextualized entity embeddings from Bootleg over all TACRED are fed to the downstream model.

Downstream Model We first use standard SpanBERT-Large embeddings to encode the input text, concatenate the contextual Bootleg embeddings with the SpanBERT embeddings, and then pass this through four transformer layers. We then calculate the cross-entropy loss and apply a softmax for scoring. We freeze the Bootleg embeddings and fine-tune the SpanBERT embeddings. We use the following hyperparameters: the learning rate is 0.00002, batch size is 8, gradient accummulation is 6, number of epochs is 10, L2 regularization is 0.008, warm-up percentage is 0.1, and maximum sequence length is 128. We train with a NVIDIA Tesla V100 GPU.

Extended Results We study the model performance as a function of the signals provided by Bootleg. In Table 12, we show that on slices with above-median numbers of Bootleg entity, relation, and type signal counts identified in the TACRED example, the relative gap between BERT and Bootleg errors is larger on the slice above, than below, the median by 1.10x, 4.67x, and 1.35x respectively. In Table 13 we show the relative error rates from the Bootleg and baseline SpanBERT models for the slices where Bootleg provides an entity, relation, or type signal for the TACRED example's subject or object. On the slice of these signals are respectively present, the baseline model performs 1.20x, 1.18x, and 1.20x worse than the Bootleg TACRED model. These results indicate that the knowledge representations from Bootleg successfully transfer useful information to the downstream model.

D Extended Error Analysis

D.1 Reasoning Patterns

Here we provide additional details about the distributions of types and relations in the data.

Distinct Tails Like entities, types and relations also have tail distributions. For example, types such as "country" and "film" appear 2.7M and 800k times respectively, while types such as "quasiregular polyhedron" and "hospital-acquired infection" appear once each in our Wikipedia training data. Meanwhile, relations such as "occupation" and "educated at" appear 35M and 16M times respectively, while relations such as

Table 12: We rank TACRED examples by the proportion of words that receive BOOTLEG embedding features where: BOOTLEG disambiguates an entity, leverages Wikidata relations for the embedding, and leverages Wikidata types for the embedding. We take examples where the proportion is greater than 0. For each of these three slices, we report the gap between the SpanBERT model and Bootleg model's error rates on the examples with above-median proportion (more BOOTLEG signal) relative to the below-median proportion (less BOOTLEG signal). With more BOOTLEG information, we see the improvement our SotA model provides over SpanBERT increases.

BOOTLEG Signal	# Examples with the Signal	Gap Above/Below Median
Entity	15323	1.10
Relation	5400	4.67
Type	15509	1.35

Table 13: We compute the error rate of SpanBERT relative to our BOOTLEG downstream model for three slices of TACRED data where respectively BOOTLEG disambiguates the subject and/or object, BOOTLEG leverages Wikidata relations for the embedding of the subject and object pair, and BOOTLEG leverages Wikidata types for the embedding of the subject and/or object in the example.

Subject-Object Signal	# Examples	BERT/BOOTLEG Error Rate
Entity	12621	1.20
Relation	542	1.18
Obj Type	12044	1.20

"positive diagnostic predictor" and "author of afterword" appear 7 times respectively in the Wikipedia training data. However we find that the entity-, relation-, and type-tails are distinct: 88% of the tail-entities by entity-count have Wikidata types that are non-tail types and 90% of the tail-entities by entity-count have non-tail relations.[12] For example, the head Wikidata type "country" contains rare entities "Palaú" and "Belgium–France border".

We observe that BOOTLEG significantly improves tail performance over each of the tails. We rank the Wikidata types and relations by the number of occurrences in the training data and study the lift from BOOTLEG as a function of the number of times the signal appears during training. BOOTLEG performs an 9.4 F1 and 20.3 F1 points better than the NED-Base baseline for examples with gold types appearing more and less than the median number of times during training respectively. BOOTLEG provides a a 7.8 F1 points and 13.7 F1 points better than the baseline for examples with gold relations appearing more and less than the median number of times during training respectively. These results indicate that BOOTLEG excels on the tails of types and relations as well.

Next, ranking the Wikidata types and relations by the proportion of comprised rare (tail and unseen) entities, we further find that BOOTLEG provides the lowest error rates across types and relations, regardless of the proportion of rare entities, while the baseline and Entity-Only models give relatively larger error rates as the proportion of rare entities increases (Figure 4). The trend for types is flat as the proportion of rare entities increases, while the trend for relations is upwards sloping. These results indicate that BOOTLEG is better able to transfer the patterns learned from one entity to other entities that share its same types and relations. The improvement from BOOTLEG over the baseline increases as the rare-proportion increases, indicating that BOOTLEG is able to efficiently transfer knowledge even when the type or relation category contains none or few popular entities.

Type Affordance For the type affordance pattern, we find that the TF-IDF keywords provide high coverage over the examples containing the gold type: 88% of examples where the gold entity has a particular type contain an affordance keyword for that type. An example of a type with full coverage by the affordance

[12]Similar to tail-entities, tail-types and tail-relations are defined as those appearing 1-10 times in the training data.

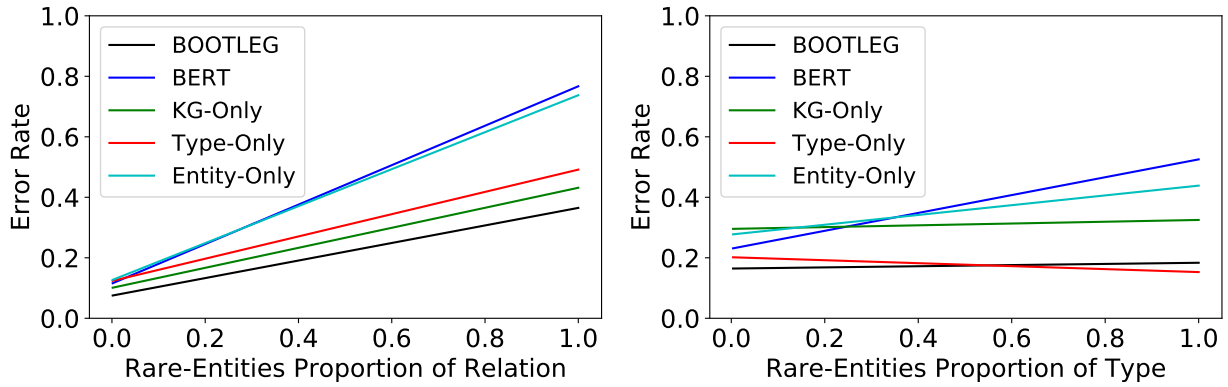

Figure 4: For all the entities of a particular type or relation, we calculate the percentage of rare entities (tails and toes entities). We show the error rate on the Wikipedia validation set as a function of the rare-proportion of entities of a given (Left) relation or (Right) type appearing in the validation set.

keywords is "café", with keywords such as "coffee", "Starbucks", and "Internet"; in each of the 77 times an entity of the type "cafe" appears in the validation set, an affordance keyword is present. Types with low coverage in the validation set by affordance keywords tend to be the rare types: for the types with coverage less than 50%, such as "dietician" or "chess official", the median number of occurrences in the validation set is 1. This supports the need for knowledge signals with distinct tails, which can be assembled to together address the rare examples.