SynSetExpan: An Iterative Framework for Joint Entity Set Expansion and Synonym Discovery

Jiaming Shen[1⋆], Wenda Qiu[1⋆], Jingbo Shang[2], Michelle Vanni[3], Xiang Ren[4], Jiawei Han[1]

[1]University of Illinois Urbana-Champaign, IL, USA, [2]University of California San Diego, CA, USA

[3]U.S. Army Research Laboratory, MD, USA, [4]University of Southern California, CA, USA

[1]{js2, qiuwenda, hanj}@illinois.edu [2]jshang@ucsd.edu [4]michelle.t.vanni.civ@mail.mil [4]xiangren@usc.edu

伊利诺伊大学厄巴纳-香槟分校

EMNLP2020

Motivation

Entity set expansion(ESE)

Task description: aims to expand a small set of seed entities into a larger set of entities that belong to the same semantic class

- A core challenge for ESE is to find those infrequent long-tail entities in the target semantic class (e.g., "Lone Star State" in the class US_States) while filtering out false positive entities from other related classes (e.g., "Austin" and "Dallas" in the class City) as they will cause semantic shift to the set.

Entity Synonym discovery(ESD)

- A major challenge here is to combine features with limited supervisions in a way that works for entities from all semantic classes. Another challenge is how to scale a ESD method to a large, extensive vocabulary that contains terms of varied qualities.

Joint Entity Set Expansion and Synonym Discovery

Assumption:

- We hypothesize that these two tasks are tightly coupled because two synonymous entities tend to have similar likelihoods of belonging to various semantic classes.

SynSetExpan:

It uses a synonym discovery model to include popular entities' infrequent synonyms into the set, which boosts the set expansion recall.

Meanwhile, the set expansion model, being able to determine whether an entity belongs to a semantic class, can generate pseudo training data to fine-tune the synonym discovery model towards better accuracy.

Problem Formulation

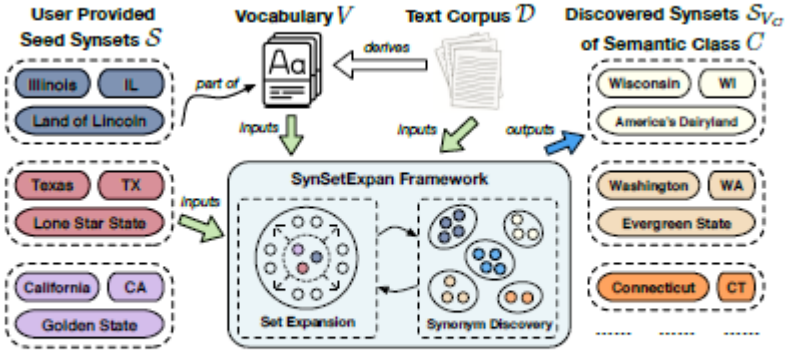


Figure 1: An illustrative example of joint entity set expansion and synonym discovery.

Problem Formulation. Given (1) a text corpus $\mathcal{D}$, (2) a vocabulary $\mathcal{V}$ derived from $\mathcal{D}$, and (3) a seed set of user-provided entity synonym sets $\mathcal{S}_0$ that belong to the same semantic class C, we aim to (1) select a subset of entities $\mathcal{V}_C$ from $\mathcal{V}$ that all belong to C; and (2) clusters all terms in $\mathcal{V}_C$ into entity synsets $\mathcal{S}_{\mathcal{V}_C}$ where the union of all clusters is equal to $\mathcal{V}_C$. In other words, we expand the seed set $\mathcal{S}_0$ into a more complete set of entity synsets $\mathcal{S}_0 \cup \mathcal{S}_{\mathcal{V}_C}$ that belong to the same semantic class C. A concrete example is presented in Fig. 1.

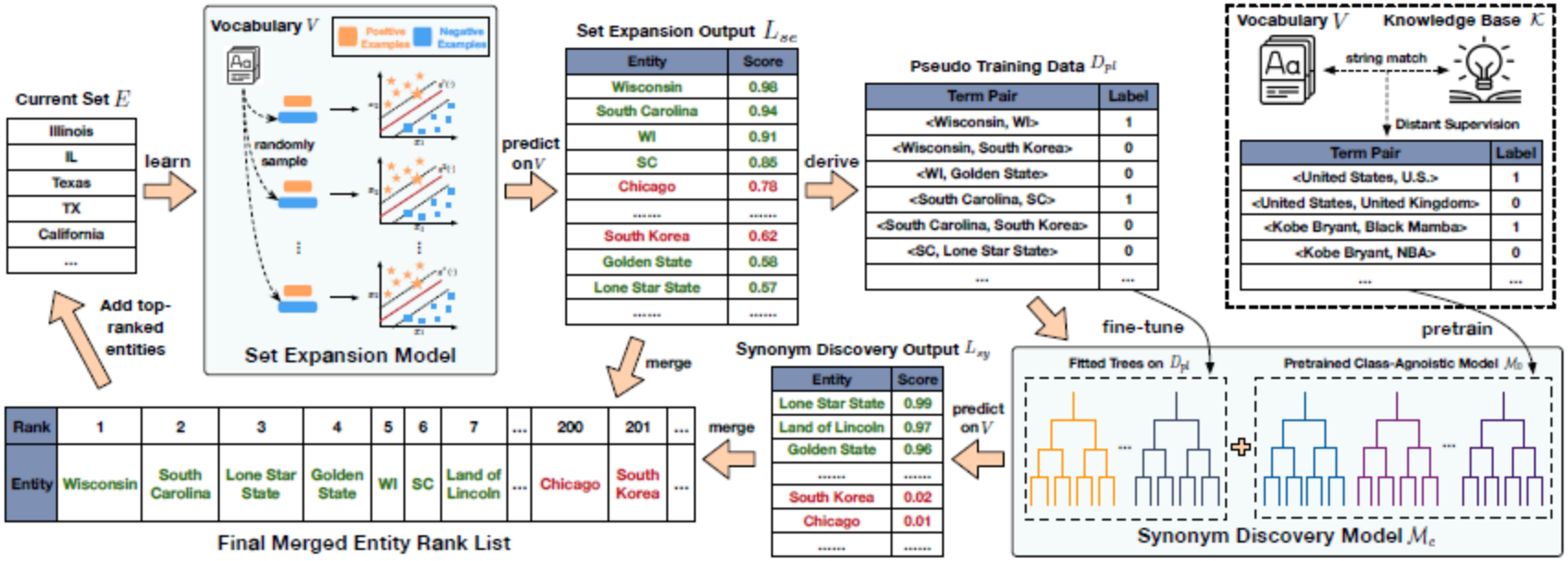


Figure 2: Overview of one iteration in our proposed SynSetExpan framework. Starting from the current set E, we first run a set expansion model to obtain an entity rank list L_{se} based on which we generate pseudo training data D_{pl} to fine-tune a generic synonym discovery model $\mathcal{M}_0$. We then apply this fine-tuned model to get a new rank list L_{sy}; merge it with L_{se} to obtain the final entity rank list, and add top ranked entities into the current set E.

Synonym Discovery Model

Features:

1. lexical features: based on entity surface names, token edit distance
2. semantic features: entity embeddings

As these feature values have different scales, we use a tree-based boosting model XGBoost.

Initializing the synonym discovery model:

- distant supervision data by matching each term in the vocabulary V with the canonical name of one entity (with its unique ID) in a knowledge base (KB). If two terms are matched to the same entity in KB and their embedding similarity is larger than 0.5, we treat them as synonyms.

Negative sampling

Set Expansion Model

Represent each entity using a set of 6 embeddings learned on the given corpus D

- SkipGram, CBOW in word2vec, fastText, SetExpander, JoSE, averaged BERT contextualized embeddings

the bag-of-embedding representation: $[\mathbf{f}^1(e_i), \mathbf{f}^2(e_i), \ldots, \mathbf{f}^B(e_i)]$

$$\mathbf{x}_i = \|_{b=1}^{6} \|_{j=1}^{|E_0|} \left[\sqrt{d_{ij}^b},\ d_{ij}^b,\ (d_{ij}^b)^2\right]$$

$$d_{ij}^b = \cos(\mathbf{f}^b(e_i), \mathbf{f}^b(e_j))$$

SVM classifier g(·) based on the above defined entity features.

Negative sampling

construct a set of $|E_0| \times K$ negative examples by random sampling.

Given a total of $|E_0| \times (K+1)$ examples, we learn a SVM classifier $g(\cdot)$ based on the above defined entity features.

Repeat the above process T times

$$\mathbf{P}(e_i \in C) = \frac{1}{T} \sum_{t=1}^{T} g^t(e_i)$$

Finally, we rank all entities in the vocabulary based on their predicted probabilities

Two Models' Mutual Enhancements

Set Expansion Enhanced Synonym Discovery. In each iteration, we generate a set of *pseudo training* data D_{pl} from the ESE model output L_{se}, to fine-tune the general synonym discovery model $\mathcal{M}_0$. Specifically, we add an entity pair $\langle e_x, e_y \rangle$ into D_{pl} with label 1, if they are among the top 100 entities in L_{se} and $\mathcal{M}_0(e_x, e_y) \geq 0.9$. For each positive pair $\langle e_x, e_y \rangle$, we generate N negative pairs by randomly selecting $\lceil N/2 \rceil$ entities from L_{se} whose set expansion output probabilities are less than 0.5 and pairing them with both e_x and e_y. The intuition is that those randomly selected entities likely come from different semantic classes with entity e_x and e_y, and thus based on our hypothesis, they are unlikely to be synonyms. After obtaining D_{pl}, we fine-tune model $\mathcal{M}_0$ by fitting H additional trees on D_{pl} and incorporate them into the existing bag of trees in $\mathcal{M}_0$. We discuss the detailed choices of N and H in the experiment.

Synonym Enhanced Set Expansion.

calculate a new score for each entity $e_i \in \mathcal{V}$

$$\text{sy-score}(e_i) = \max\{\mathcal{M}_c(e_i, e_j) | e_j \in E\}.$$

$$\text{final-score}(e_i) = \sqrt{\mathbf{P}(e_i \in C) \times \text{sy-score}(e_i)}.$$

Algorithm 1: SynSetExpan Framework.

Input: A seed set $\mathcal{S}_0$, a vocabulary $\mathcal{V}$, a knowledge base $\mathcal{K}$, maximum iteration number *max_iter*, maximum size of expanded set Z, and model hyper-parameters $\{K, T, N, H\}$.

Output: A complete set of entity synsets $\mathcal{S}_{\mathcal{V}_C}$.

Learn a general ESD model $\mathcal{M}_0$ using distant supervision in $\mathcal{K}$;
$E \leftarrow$ Union of all synsets in $\mathcal{S}_0$;
for *iter from 1 to max_iter* **do**
    $L_{se} \leftarrow \text{ESEModel}(E, \mathcal{V}, K, T)$;
    Generate pseudo training data D_{pl} from L_{se};
    Construct a class-specific ESD model $\mathcal{M}_c$ by fine-tuning $\mathcal{M}_0$ on D_{pl};
    Apply $\mathcal{M}_c$ on entities in $\mathcal{V}$ and adjust L_{se};
    Add top $\lceil \frac{Z}{\text{max\_iter}} \rceil$ entities in the adjusted rank list into E;
Construct a synonym graph $\mathcal{G}$ based on final set E;
$\mathcal{S}_{\mathcal{V}_C} \leftarrow \text{Louvain}(\mathcal{G})$;
Return $\mathcal{S}_{\mathcal{V}_C}$.

K - negative examples in Set Expansion Model

T- learning T separate classifiers

N - negative examples in Set Expansion Enhanced Synonym Discovery

H-additional trees

The SE2 Dataset

Corpus Size	# Entities	# Classes	# Queries
1.9B Tokens	1.5M	60	1200

Table 1: Our SE2 dataset statistics

1. Corpus and Vocabulary Selection

Wikipedia 20171201
We extract all noun phrases with frequency above 10 as our selected vocabulary

2. Semantic Class Selection. We identify 60 major semantic classes based on the *DBpedia-Entity v2* (Hasibi et al., 2017) and *WikiTable* (Bhagavatula et al., 2015) entities found in our corpus. These 60 classes cover 6 different entity types (e.g., Person, Location, Organization). As such generated classes may miss some correct entities, we enlarge each class via crowdsourcing in the following step.

3. Query Generation and Class Enrichment.

We first generate 20 queries for each semantic class.
Then, we aggregate the top 100 results from all baseline methods and obtain 17,400 <class, entity> pairs.

4. Synonym Set Curation

To construct synsets in each class, we first run all baseline methods to generate a candidate pool of possible synonymous term pairs. Then, we treat those pairs with both terms mapped to the same entity in WikiData as positive pairs and ask two human annotators to label the remaining 7,625 pairs.

Set expansion difficulty of each class and Synonym discovery difficulty of each class

$$\text{Set-Expansion-Difficulty}(C) = \frac{1}{|C|} \sum_{e \in C} \frac{|C - \text{Top}k(e)|}{|C|}$$

Lexical difficulty defined as the average Jaro-Winkler distance between the surface names of two synonyms.

Semantic difficulty defined as the average cosine distance between two synonymous entities' embeddings.

Class Type	ESE	ESD (Lexical)	ESE (Semantic)
Location	0.3789	0.2132	0.6599
Person	0.2322	0.2874	0.5526
Product	0.0848	0.3922	0.4811
Facility	0.0744	0.2345	0.4466
Organization	0.1555	0.2566	0.4935
Misc	0.4282	0.2743	0.5715

Table 2: Difficulty of each semantic class for entity set expansion (ESE) and entity synonym discovery (ESD).

Methods	SE2			Wiki			APR		
	MAP@10	MAP@20	MAP@50	MAP@10	MAP@20	MAP@50	MAP@10	MAP@20	MAP@50
Egoset (Rong et al., 2016)	0.583	0.533	0.433	0.904	0.877	0.745	0.758	0.710	0.570
SetExpan (Shen et al., 2017)	0.473	0.418	0.341	0.944	0.921	0.720	0.789	0.763	0.639
SetExpander (Mamou et al., 2018b)	0.520	0.475	0.397	0.499	0.439	0.321	0.287	0.208	0.120
MCTS (Yan et al., 2019)	—	—	—	0.980	0.930	0.790	0.960	0.900	0.810
CaSE (Yu et al., 2019c)	0.534	0.497	0.420	0.897	0.806	0.588	0.619	0.494	0.330
SetCoExpan (Huang et al., 2020)	—	—	—	0.976	0.964	0.905	0.933	0.915	0.830
CGExpan (Zhang et al., 2020)	0.601	0.543	0.438	**0.995**	**0.978**	0.902	**0.992**	**0.990**	0.955
SynSetExpan-NoSYN	0.612	0.567	0.484	0.991	**0.978**	**0.904**	0.985	**0.990**	**0.960**
SynSetExpan	**0.628***	**0.584***	**0.502***	—	—	—	—	—	—

Table 3: Set expansion results on three datasets. MCTS and SetCoExpan do not scale to the SE2 dataset. SynSetExpan-Full is inapplicable for Wiki and APR datasets because they contain no synonym information. The superscript * indicates the improvement is statistically significant compared to SynSetExpan-NoSYN.

Method	SE2			PubMed		
	AP	AUC	F1	AP	AUC	F1
SVM	0.1870	0.8547	0.3300	0.2250	0.8206	0.4121
XGB-S (Chen and Guestrin, 2016)	0.7654	0.9696	0.6389	0.5012	0.8625	0.4968
XGB-E (Chen and Guestrin, 2016)	0.4762	0.8750	0.4810	0.4906	0.9190	0.5388
DPE (Qu et al., 2017)	0.7972	0.9792	0.6392	0.6338	0.8979	0.6038
SynSetMine (Shen et al., 2019)	0.7562	0.9782	0.6347	0.6757	0.9453	0.6287
SynSetExpan-NoFT	0.8197	0.9844	0.7159	0.6615	0.9445	0.6204
SynSetExpan	**0.8736**	**0.9953**	**0.7592**	**0.7152**	**0.9695**	**0.6388**

Table 6: Synonym discovery results on both SE2 dataset and PubMed dataset.

Class Type	MAP@10	MAP@20	MAP@50
Person	86.7%	80.0%	93.3%
Organization	83.3%	83.3%	100%
Location	69.2%	65.4%	80.8%
Facility	85.7%	71.4%	100%
Product	100%	66.7%	100%
Misc	66.7%	66.7%	100%
Overall	**78.3%**	**71.7%**	**90.0%**

Table 4: Ratio of semantic classes on which SynSetExpan outperforms SynSetExpan-NoSYN.

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

Motivation

To better disambiguate the long tail of entities

1. Previous NED systems disambiguate by memorizing co-occurrences between entities and textual context in a self-supervised manner. The self-supervision is critical to building a model that is easy to maintain and does not require expensive hand-curated features.
2. However, these approaches struggle to handle tail entities: a baseline SotA model from 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 on-the-order-of 100 times during training.

- **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 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]

Reasoning Patterns for Disambiguation

Generalizing Learning to the Tail: contribute a new 2D regularization scheme to combine the entity, tail, and relation signal

Weak Labelling of Data: most sentences on an entity's Wikipedia page refer to that entity via pronouns or alternative name

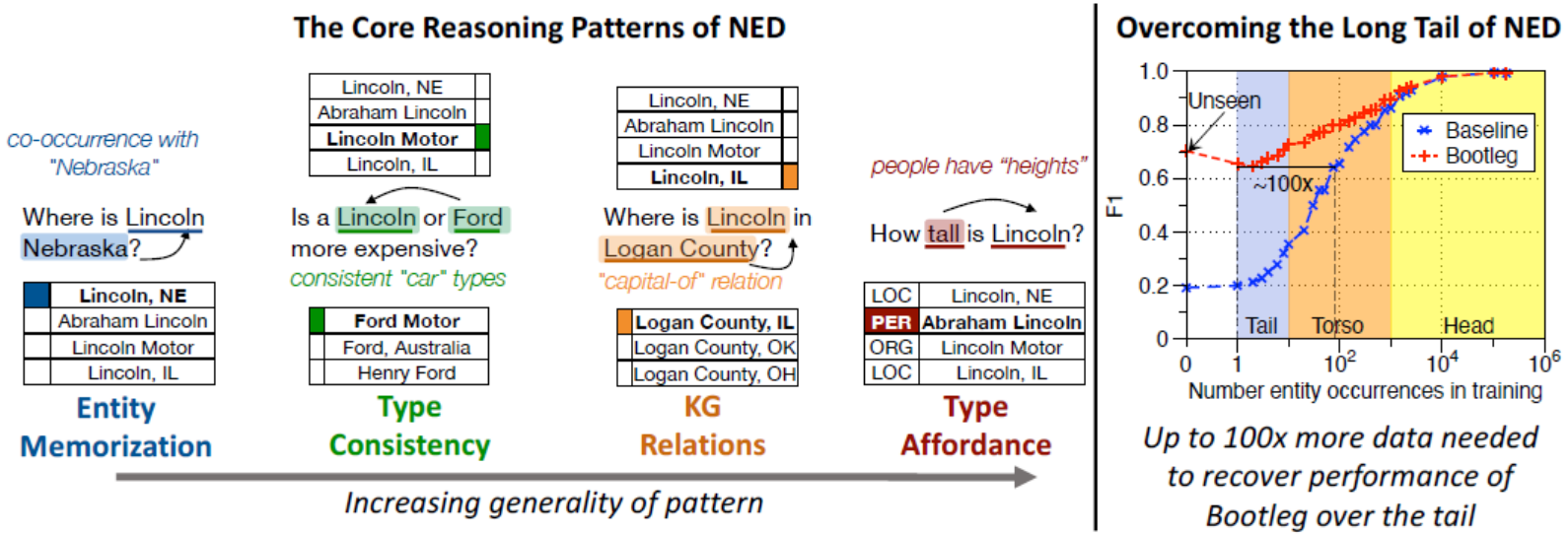


Figure 1: (Left) shows the four reasoning patterns for disambiguation. The correct entity is bolded. (Right) shows F1 versus number of times an entity was seen in training data for a baseline NED model compared to BOOTLEG across the head, torso, tail, and unseen.

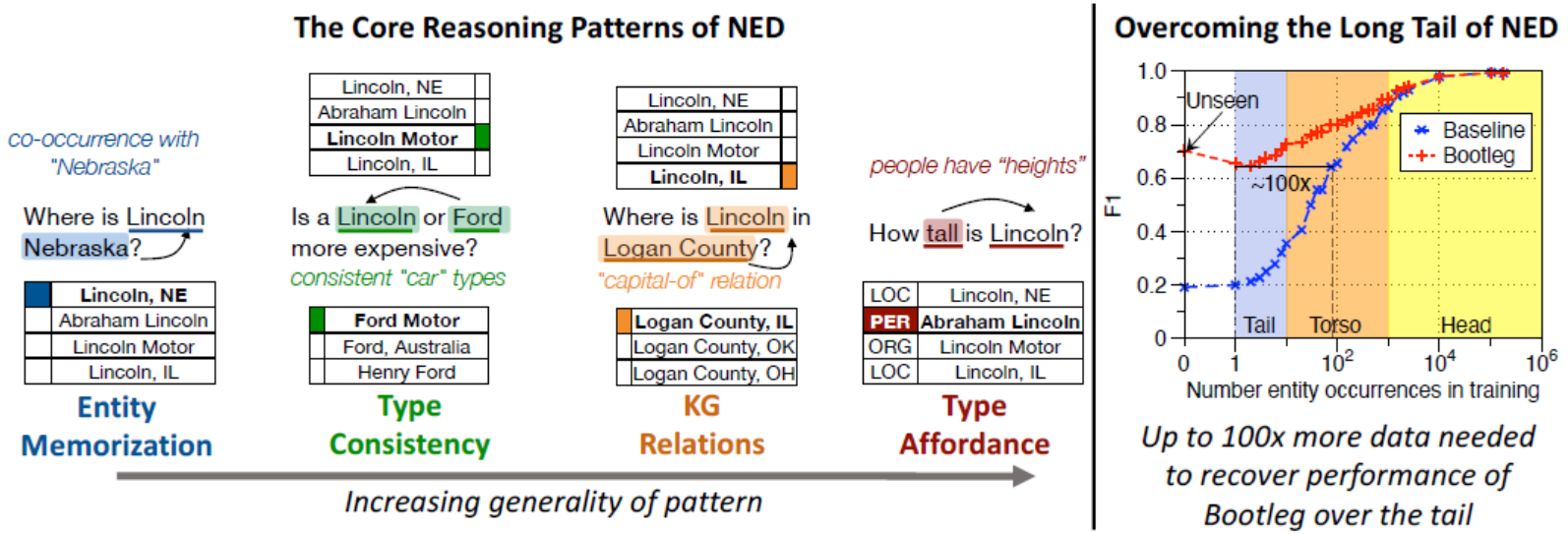


Figure 1: (Left) shows the four reasoning patterns for disambiguation. The correct entity is bolded. (Right) shows F1 versus number of times an entity was seen in training data for a baseline NED model compared to BOOTLEG across the head, torso, tail, and unseen.

Entity Memorization: Disambiguating “Lincoln” in the text “Where is Lincoln, Nebraska?” requires memorizing that “Lincoln, Nebraska”, not “Abraham Lincoln” frequently occurs with the text “Nebraska”. 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.

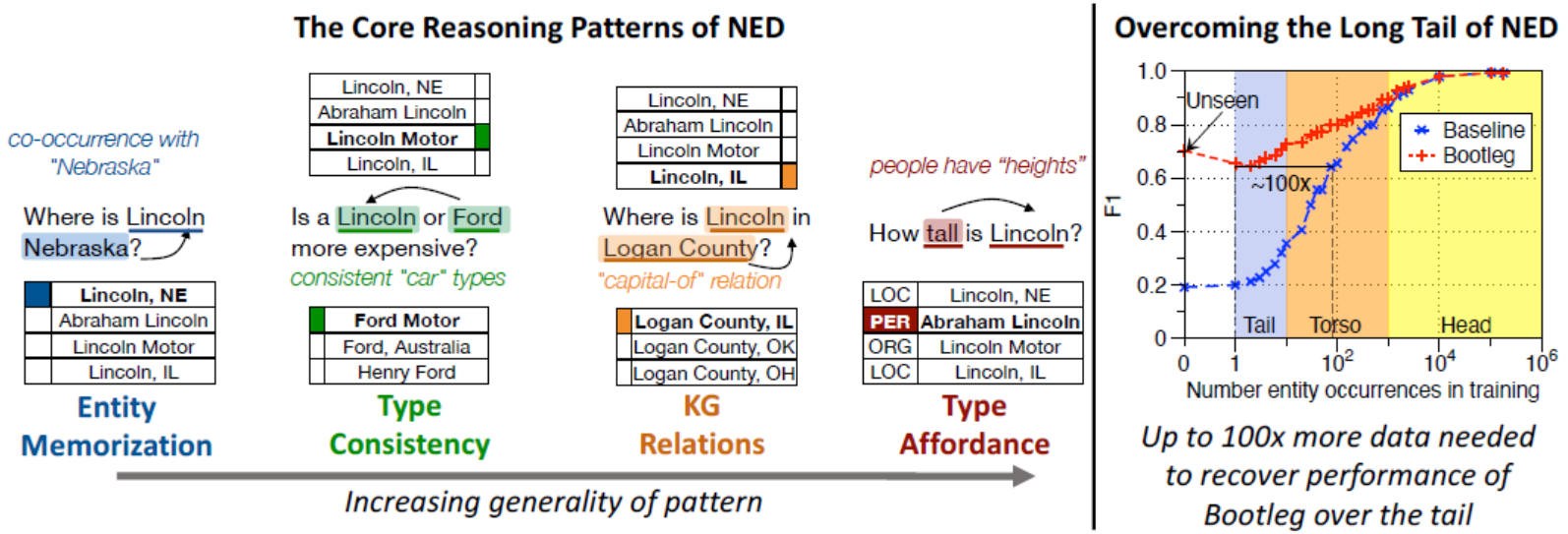


Figure 1: (Left) shows the four reasoning patterns for disambiguation. The correct entity is bolded. (Right) shows F1 versus number of times an entity was seen in training data for a baseline NED model compared to BOOTLEG across the head, torso, tail, and unseen.

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”.

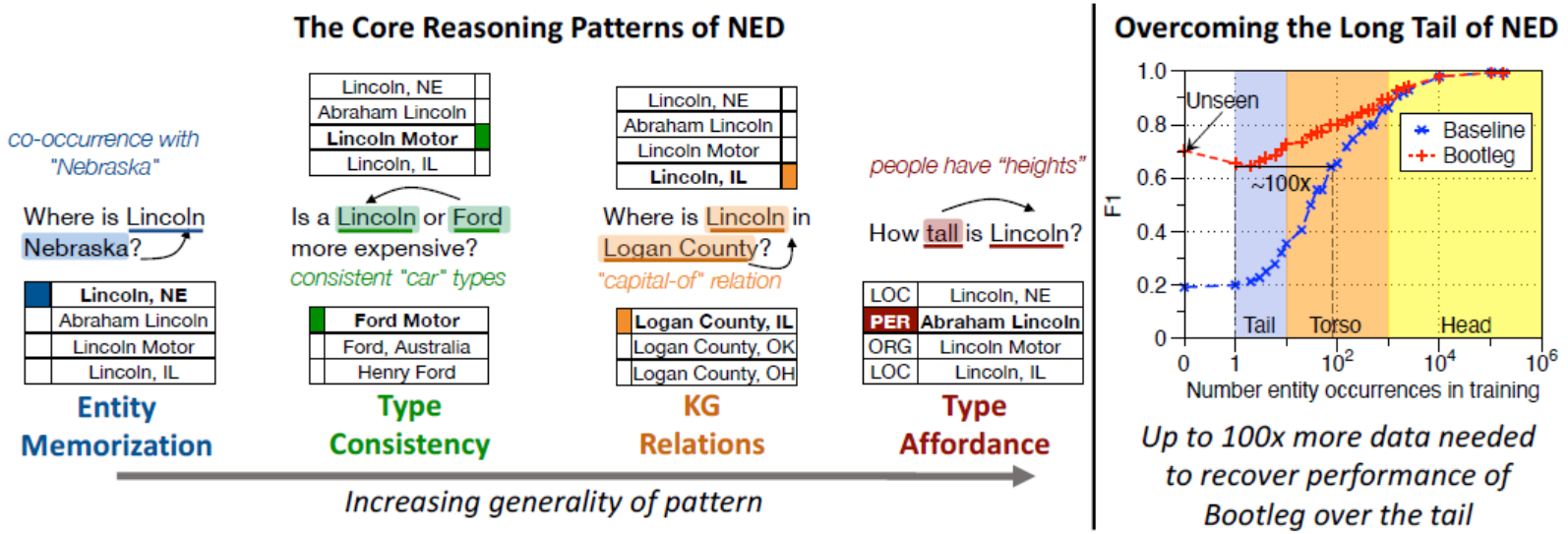


Figure 1: (Left) shows the four reasoning patterns for disambiguation. The correct entity is bolded. (Right) shows F1 versus number of times an entity was seen in training data for a baseline NED model compared to BOOTLEG across the head, torso, tail, and unseen.

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.

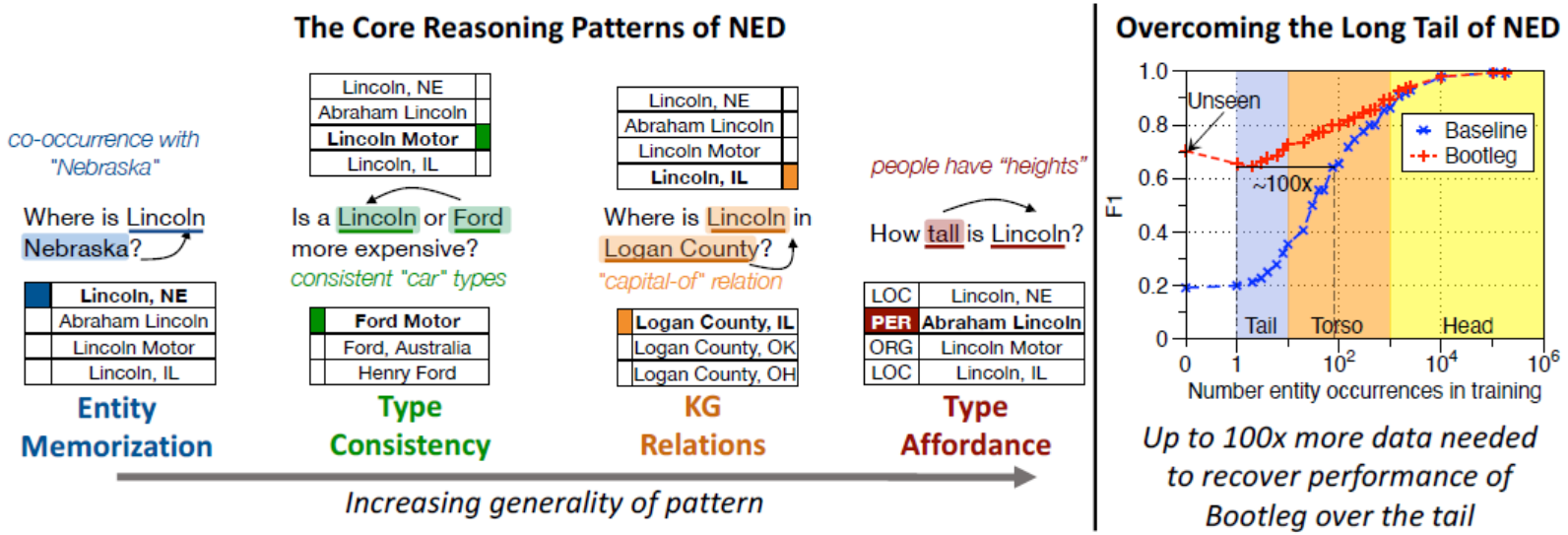
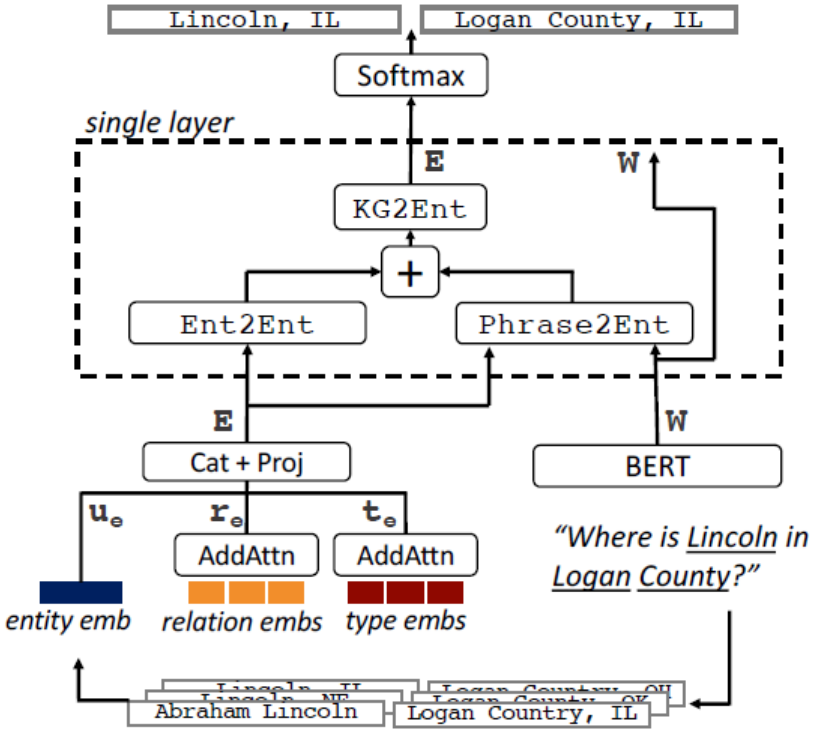


Figure 1: (Left) shows the four reasoning patterns for disambiguation. The correct entity is bolded. (Right) shows F1 versus number of times an entity was seen in training data for a baseline NED model compared to BOOTLEG across the head, torso, tail, and unseen.

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”.



Entity embedding

Each entity e is represented by a unique embedding

Type Embedding:

Let $\mathcal{T}$ be the set of possible entity types.

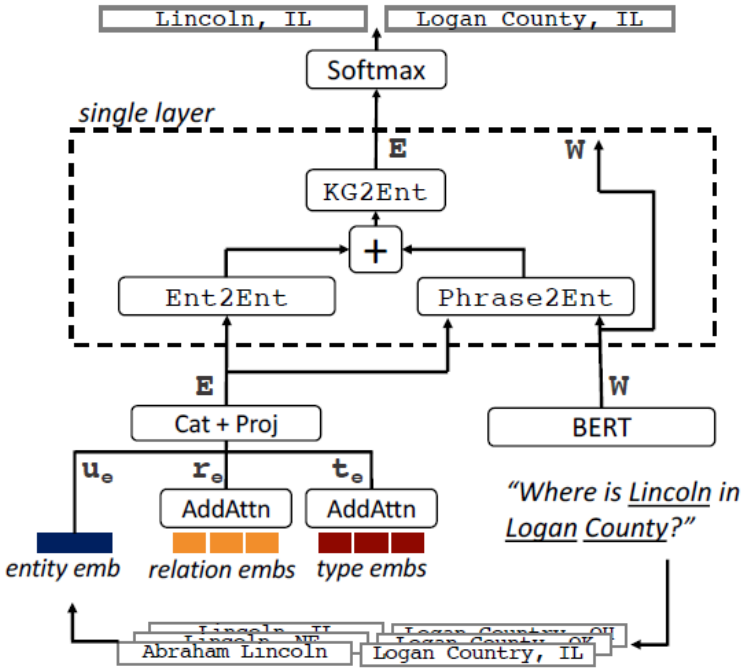
$\{t_{e,1}, \ldots, t_{e,T} | t_{e,i} \in \mathcal{T}\}$ of T possible types.

BOOTLEG assigns an embedding $\mathrm{t}_{e,i}$ to each type.

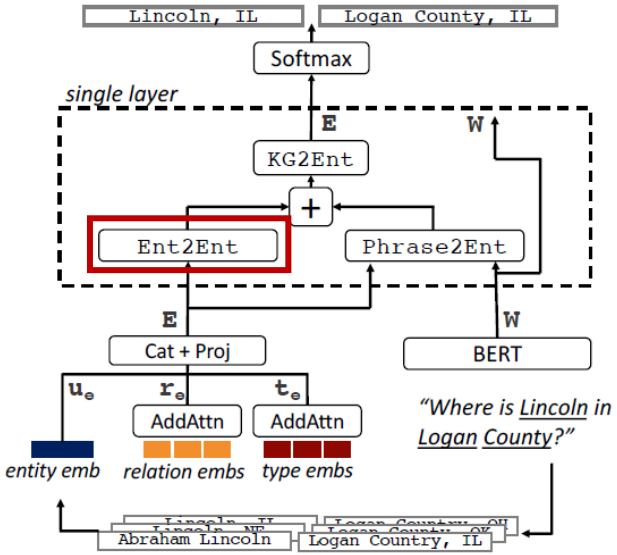
Because an entity can have multiple types, we use an additive attention, AddAttn

$\mathrm{t}_e = \mathrm{AddAttn}([\mathrm{t}_{e,1}, \ldots, \mathrm{t}_{e,T}])$

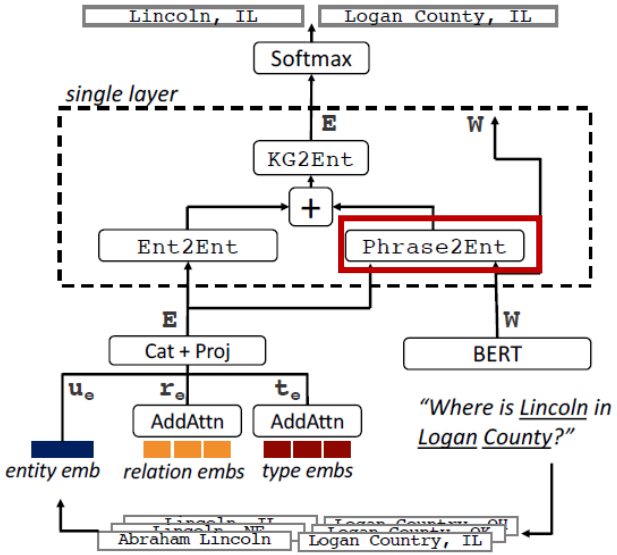
Relation Embedding: Let $\mathcal{R}$ represent the set of possible relationships any entity can participate in. Similar to types, given a mapping from an entity e to its set $\{r_{e,1}, \ldots, r_{e,R} | r_{e,i} \in \mathcal{R}\}$ of R relationships, BOOTLEG assigns an embedding $\mathrm{r}_{e,i}$ to each relation. Because an entity can participate in multiple relations, we use the additive attention to compute $\mathrm{r}_e = \mathrm{AddAttn}([\mathrm{r}_{e,1}, \ldots, \mathrm{r}_{e,R}])$.



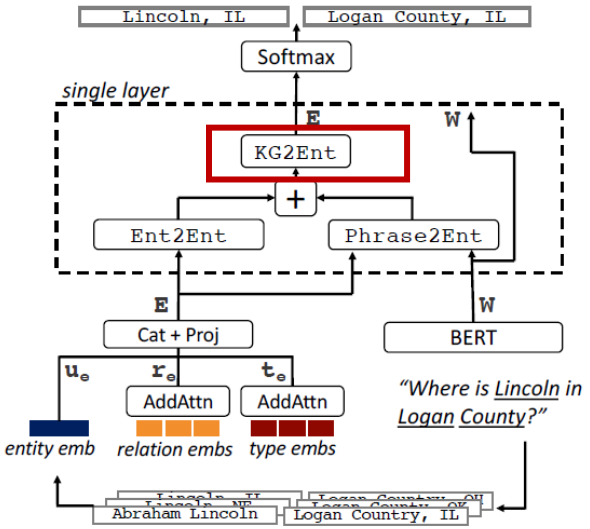
As in existing work [16, 40], given the input sentence of length N and set of M mentions, BOOTLEG generates for each mention m_i a set $\Gamma(m_i) = \{e_i^1, \ldots, e_i^K\}$ of K possible entity candidates that could be referred to by m_i. For each candidate and its associated types and relations, 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.



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.



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].

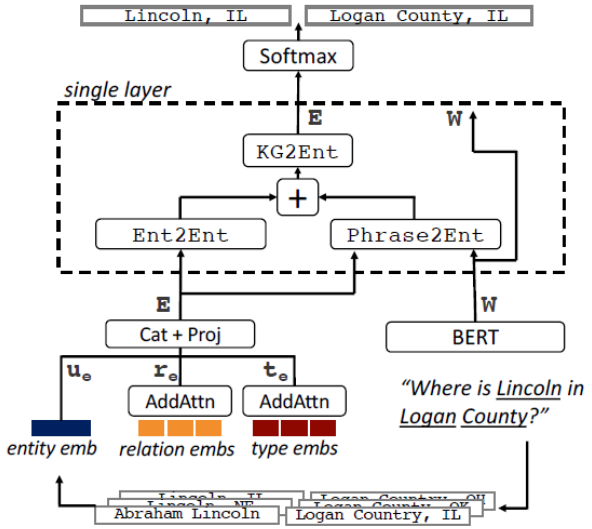


Knowledge Graph (KG) Connection Module:

K represent the adjacency matrix of a (possibly weighted) graph

Given $\mathbf{E}$, KG2Ent computes $\mathbf{E}_k = \text{softmax}(\tilde{\mathbf{K}} + w\mathbf{I})\mathbf{E} + \mathbf{E}$ where $\mathbf{I}$ is the identity and w is a learned scalar weight

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.



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

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

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

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.

- *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).

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.

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

Dataset

- **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.

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.

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.

Result

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

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

Analysis

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%

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

Global Structure and Local Semantics-Preserved Embeddings for Entity Alignment

Hao Nie[1,3], **Xianpei Han**[1,2 *], **Le Sun**[1,2],
Chi Man Wong[4], **Qiang Chen**[4], **Suhui Wu**[4] and **Wei Zhang**[4]
[1]Chinese Information Processing Laboratory, Institute of Software, Chinese Academy of Sciences
[2]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
[3]University of Chinese Academy of Sciences
[4]Alibaba Group
{niehao2016, xianpei, sunle}@iscas.ac.cn,
{chiman.wcm, lapu.cq, linnai.wsh, lantu.zw}@alibaba-inc.com

IJCAI2020

Task definition of entity alignment:

knowledge graph $G = (E, R, T)$

the source KG is denoted as $G_1 = (E_1, R_1, T_1)$

the target KG is denoted as $G_2 = (E_2, R_2, T_2)$.

$$S = \{(u, v) | u \in E_1, v \in E_2, u \leftrightarrow v\}$$

Entity alignment task can be formulated as automatically discovering more aligned entity pairs based on the existing alignment seeds.

For example, by aligning entities “CA” and “California” from triples <Apple Inc., locatedIn, CA> and <California, country, USA>, we can obtain additional knowledge <Apple Inc., locatedIn, USA>.

Traditional methods

Translation-based embedding methods

——represent an entity by exploiting its local semantics

1. Considering the triples in which it appears, which can provide fine-grained information
2. Not robust enough if some triples are missing or two KGs are represented using different schemas

GCN-based methods

——represent an entity by exploiting the global KG structure

1. Provide comprehensive and robust information for entity alignment
2. Less vulnerable to the missing of partial information and the schema heterogeneity of different KGs
3. May lose fine-grained details for EA

Motivation

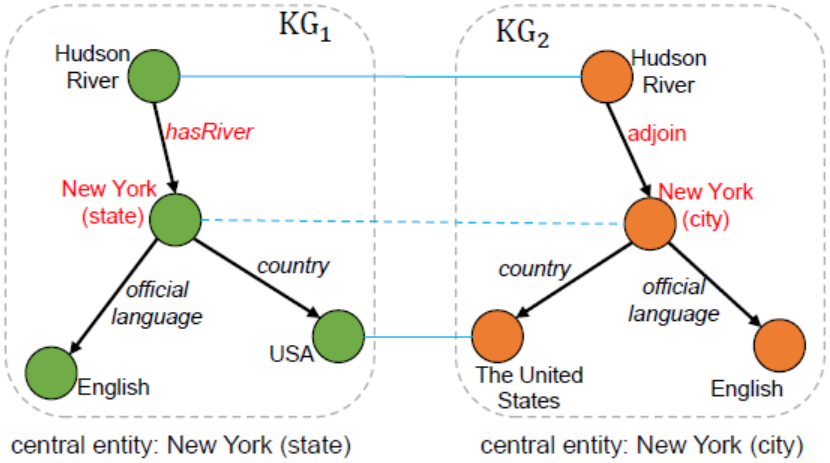


Figure 1: An entity alignment example where *New York (state)* and *New York (city)* will be aligned only using the KG structure, therefore the local relations *hasRiver* and *adjoin* are critical to distinguish them as different real-world objects.

Motivation

1. We propose to jointly leverage the global structure and local semantics for entity alignment.

2. By exploiting both the global structure and local semantics, the entity representations can be both robust and accurate.

3. We design a Structure and Semantics Preserving (SSP) network which can learn the entity representations by simultaneously exploiting both the global structure and local semantics of KGs in a coarse-to-fine manner.

Overall Architecture

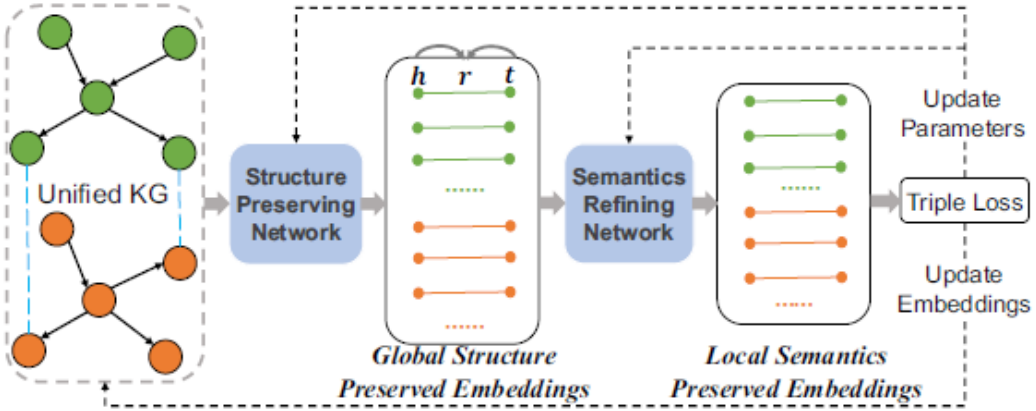


Figure 2: Overall architecture of the Structure and Semantics Preserving networks.

Structure Preserving Network：

The global structure of KG provides useful information for entity alignment, i.e., entities with similar neighboring structures are highly likely to be aligned. This paper employs a GCN to explicitly encode the global KG structure information. Specifically, GCN follows a neighborhood aggregation scheme, where the representation vector of a node is computed by recursively aggregating and transforming representation vectors of its neighboring nodes. Concretely, GCN consists of several stacked layers. Given a set of node features $X^{(l)} = \{x_1^{(l)}, x_2^{(l)}, \cdots, x_n^{(l)} | x_i^{(l)} \in \mathbb{R}^{d^{(l)}}\}$ as input to GCN layer l, where n is the number of nodes (entities) in the unified KG, and $d^{(l)}$ is the number of features in layer l, the output of the l-th layer is obtained following the convolution computation:

$$X^{(l+1)} = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}X^{(l)}W^{(l)}\right) \quad (1)$$

where σ is an activation function; A is a $n \times n$ adjacency matrix that denotes the structure information of the KG; $\hat{A} = A + I$, and I is the identity matrix; $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$; $W^{(l)} \in d^{(l)} \times d^{(l+1)}$ is the weight matrix of the l-th layer in the GCN, $d^{(l+1)}$ is the number of features in the $(l+1)$-th layer.

Highway gates:

Inspired by HGCN [Wu *et al.*, 2019b], we also employ layer-wise highway gates [Srivastava *et al.*, 2015] to control the balance of how much neighborhood information should be passed to a node and reduce noise propagation, where the output of a layer is the weighted sum of its input and the original output via gating weights $T(X^{(l)})$:

$$T(X^{(l)}) = \sigma(X^{(l)}W_T^{(l)} + b_T^{(l)}) \quad (2)$$

$$X^{(l+1)} = T(X^{(l)}) \cdot X^{(l+1)} + (1 - T(X^{(l)})) \cdot X^{(l)} \quad (3)$$

Semantics Refining Network

The assumption:

Intuitively, relations occurred in different entity contexts should have distinct embeddings, regardless of whether they have the same surface forms or not.

Method:

Calculate the relation embeddings based on the adjacent entities and the relation itself.

Contextualized Relation Learning

- Get entity embeddings from the output of the last layer in the HGCN
- Concatenate the embeddings of head entity h and tail entity t
- use a one-layer MLP with non-linear activation to compress the entity contexts

$$\mathbf{n}_{ht} = \sigma([\mathbf{h}; \mathbf{t}]W + \mathbf{b})$$

Inspired by TransEdge (projecting embeddings onto hyperplane has shown promising effects on the processing of disparate feature representations), to further incorporate the relation embedding itself:

$$\mathbf{r}_c = \mathbf{r} - \mathbf{n}_{ht}{}^{\top}\mathbf{r}\mathbf{n}_{ht}$$

Entity Embeddings Refinement

Entity Embeddings Refinement. Having obtained the contextualized relation embedding $\mathbf{r}_c$, we apply it to the translational model to refine entity embeddings. Like TransE, we define a scoring function to estimate the plausibility of relational triples as:

$$f(h,r,t) = \|\mathbf{h} + \mathbf{r}_c - \mathbf{t}\| \tag{6}$$

where relational triples that exist in the KG are expected to be more plausible with low energy score. By distinctly modeling relational triples in the KG, we can further incorporate complex relation semantics information into entity embeddings and obtain more expressive knowledge embeddings. Therefor better entity alignment performance can be achieved.

Alignment Prediction. Having obtained entity embeddings with global structure and local semantics information preserved, we align entities using the similarities between their embeddings. Specifically, for each entity to be aligned, we rank all candidate entities based on their similarities to the target entity.

The Calculation of Loss

4.3 End-to-End Learning

For training, we regard all observed relational triples in KGs as positive samples, while all unobserved ones as negative samples (either false or missing triples). It is expected that positive samples are supposed to have lower energy than their negative counterparts. We thus minimize the following limit-based loss function as in [Zequn *et al.*, 2019] to distinguishably separate positive triples from negative ones, aiming to gather aligned entities together, and in the meantime tell unaligned ones apart.

$$\mathcal{L} = \sum_{(h,r,t)\in\mathcal{T}^+} [f(h,r,t) - \beta_1]_+ + \sum_{(h',r',t')\in\mathcal{T}^-} \alpha\,[\beta_2 - f\,(h',r',t')]_+ \tag{7}$$

α is a hyper parameter for balancing positive and negative samples, β_1 and β_2 denote positive and negative margins respectively. $[x]_+$ denotes $max(0,x)$. $\mathcal{T}^+$ is the set of observed triples, and $\mathcal{T}^-$ is the negative triples set with each triple created using the truncated uniform sampling strategy.

Results

DBP15K: Popular entities from English to Chinese, Japanese and French respectively. Each dataset has 15 thousand interlingual links (ILLs) as reference alignments

Methods	$DBP15K_{ZH-EN}$				$DBP15K_{JA-EN}$				$DBP15K_{FR-EN}$			
	Hits@1	Hits@10	MRR	MR	Hits@1	Hits@10	MRR	MR	Hits@1	Hits@10	MRR	MR
MtransE	0.308	0.614	0.364	154	0.279	0.575	0.349	159	0.244	0.556	0.335	139
IPTransE	0.406	0.735	0.516	-	0.367	0.693	0.474	-	0.333	0.685	0.451	-
BootEA	0.629	0.848	0.703	-	0.622	0.854	0.701	-	0.653	0.874	0.731	-
MMR†	0.679	0.867	-	-	0.654	0.858	-	-	0.675	0.890	-	-
TransEdge	0.659	0.903	0.748	50	0.646	0.907	0.741	36	0.649	0.921	0.746	25
TransEdge‡	0.735	0.919	0.801	32	0.719	0.932	0.795	25	0.710	0.941	0.796	**12**
GCN-Align	0.413	0.744	-	-	0.399	0.745	-	-	0.373	0.745	-	-
AliNet	0.539	0.826	0.628	-	0.549	0.831	0.645	-	0.552	0.852	0.657	-
SSP(-HW)	0.506	0.823	0.618	75	0.520	0.835	0.632	52	0.480	0.819	0.600	51
SSP(-RC)	0.667	0.908	0.758	42	0.665	0.918	0.757	30	0.682	0.934	0.775	17
SSP	**0.739**	**0.925**	**0.808**	**26**	**0.721**	**0.935**	**0.800**	**19**	**0.739**	**0.947**	**0.818**	13

Table 1: Overall performance on DBP15K datasets. All comparable results are taken from their original papers. The first two parts of the results separated by the dashed line denote the translational knowledge embedding methods and GCN-based methods respectively. The last part of the table denotes the results of our models. - denotes unreported results on their original papers. † denotes that the entities are aligned from two directions, and we just take the average of them. ‡ denotes the model uses a bootstrapping strategy to iteratively select likely-aligned entity pairs to enlarge the training set.

DWY100K

SSP(-HW)	0.617	0.898	0.716	61	0.726	0.928	0.798	35
SSP(-RC)	0.710	0.949	0.796	15	0.799	0.964	0.859	14
SSP(Concate)	0.591	0.721	0.639	83	0.638	0.715	0.714	69
SSP	0.772	**0.960**	**0.842**	**12**	**0.811**	**0.968**	**0.869**	**11**

RC: $\mathbf{n}_{ht} = \sigma([\mathbf{h};\mathbf{t}]W + \mathbf{b})$

$\mathbf{r}_c = \mathbf{r} - \mathbf{n}_{ht}{}^{\top}\mathbf{r}\mathbf{n}_{ht}$

DBP-WD (DBpedia-Wikidata) and DBP-YG (DBpedia-YAGO3). Each dataset has 100,000 aligned entity pairs and hundreds of thousands of relational triples.

Discussion:

Impact of Number of Alignment Seeds

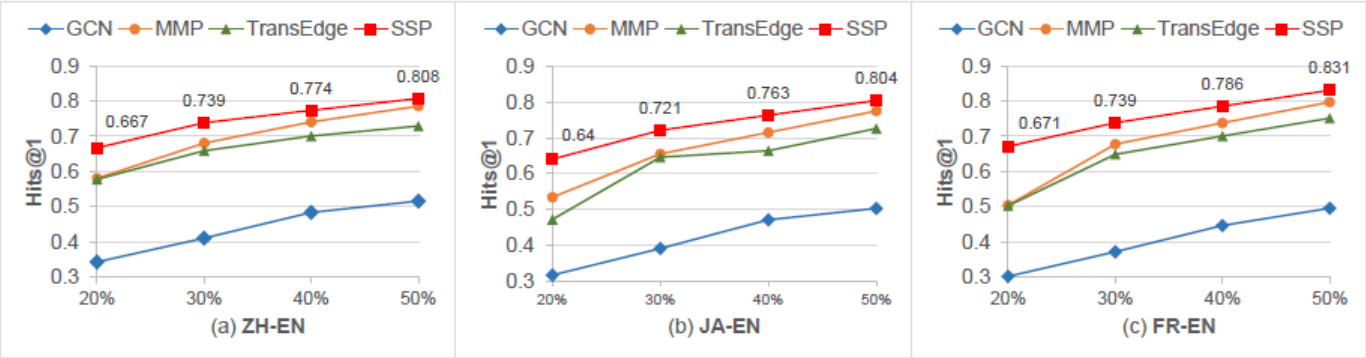


Figure 3: (a)-(c) report the Hits@1 performance of GCN, MMR, TransEdge and SSP on DBP15K datasets when they are given different proportions of alignment seeds as training data.

Discussion:

Impact of Entity Sparsity

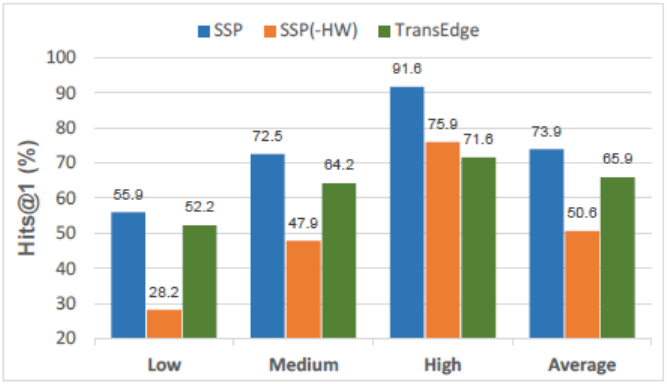


Figure 4: Model performance on different levels of node degree. Gray bar "avg" denotes model performances on the original test set.

In real-world KGs, the node degree (number of adjacent entities) usually follows a long-tail distribution, which means a large proportion of entities have a moderate degree and the proportion of entities with a very high or low degree is relatively small. This practical situation can have a big critical impact on EA. For sparse nodes (entities) with little structure information, it is also very hard for an embedding method to learn high-quality entity embeddings, thus would deteriorate the EA performance

Conclusion:

- Succeed to improve performance by combining the advantages of translation-based and GCN-based models.

- Not so innovative.