Abstract

Entity linking (EL), the task of disambiguating mentions in text by linking them to entities in a knowledge graph, is crucial for text understanding, question answering or conversational systems. Entity linking on short text (e.g., single sentence or question) poses particular challenges due to limited context. While prior approaches use either heuristics or black-box neural methods, here we propose LNN-EL, a neuro-symbolic approach that combines the advantages of using interpretable rules based on first-order logic with the performance of neural learning. Even though constrained to using rules, LNN-EL performs competitively against SotA black-box neural approaches, with the added benefits of extensibility and transferability. In particular, we show that we can easily blend existing rule templates given by a human expert, with multiple types of features (priors, BERT encodings, box embeddings, etc), and even scores resulting from previous EL methods, thus improving on such methods. For instance, on the LC-QuAD-1.0 dataset, we show more than 4% increase in F1 score over previous SotA. Finally, we show that the inductive bias offered by using logic results in learned rules that transfer well across datasets, even without fine tuning, while maintaining high accuracy.

1 Introduction

Entity Linking (EL) is the task of disambiguating textual mentions by linking them to canonical entities provided by a knowledge graph (KG) such as DBpedia, YAGO (Suchanek et al., 2007) or Wikidata (Vrandečić and Krötzsch, 2014). A large body of existing work deals with EL in the context of longer text (i.e., comprising of multiple sentences) (Bunescu and Pasca, 2006). The general approach is: 1) extract features measuring some degree of similarity between the textual mention and any one of several candidate entities (Mihalcea and Csomai, 2007; Cucerzan, 2007; Ratinov et al., 2011), followed by 2) the disambiguation step, either heuristics-based (non-learning) (Hoffart et al., 2011; Sakor et al., 2019; Ferragina and Scianna, 2012) or learning-based (Mihalcea and Csomai, 2007; Cucerzan, 2007; Ratinov et al., 2011; Hoffart et al., 2012; Ganea and Hofmann, 2017), to link the mention to an actual entity.

A particular type of entity linking, focused on short text (i.e., a single sentence or question), has attracted recent attention due to its relevance for downstream applications such as question answering (e.g., (Kapanipathi et al., 2021)) and conversational systems. Short-text EL is particularly challenging because the limited context surrounding mentions results in greater ambiguity (Sakor et al., 2019). To address this challenge, one needs to exploit as many features from as many sources of evidence as possible.

Consider the question in Figure 1(a), containing mention1 (Cameron) and mention2 (Titanic).1 DBpedia contains several person entities whose last name matches Cameron. Two such entities are shown in Figure 3(b), James_Cameron and Roderick_Cameron, along with their string similarity scores (in this case, character-level Jaccard similarity) to mention1. In this case, the string similarities are quite close. In the absence of reliable discerning information, one can employ a prior such as using the more popular candidate entity, as measured by the in-degree of the entity in the KG (see Figure 3(b)). Given the higher in-degree, we can (correctly) link mention1 to James_Cameron. However, for mention2, the correct entry is Titanic_1997_film as opposed to

1Note that we assume that mention extraction has already been applied and we are given the textual mentions.
When we start with an input set of rule templates (given by an expert or available as a library), and learn the parameters of these rules (namely, the thresholds of the various similarity predicates as well as the weights of the predicates that appear in the rules), based on a labeled dataset. We use logical neural networks (LNN) (Riegel et al., 2020), a powerful neuro-symbolic AI approach based on real-valued logic that employs neural networks to learn the parameters of the rules. Learning of the rule templates themselves will be the focus of future work.

**Summary of contributions**

- We propose, to the best of our knowledge, the first neuro-symbolic method for entity linking (coined “LNN-EL”) that provides a principled approach to learning EL rules.

- Our approach is extensible and can combine disparate types of local and global features as well as results of prior black-box neural methods, thus building on top of such approaches.

- Our approach produces interpretable rules that humans can inspect toward actionable insights.

- We evaluate our approach on three benchmark datasets and show competitive (or better) performance with SotA black-box neural approaches (e.g., BLINK (Wu et al., 2020)) even though we are constrained on using rules.

- By leveraging rules, the learned model shows a desirable transferability property: it performs well not only on the dataset on which it was trained, but also on other datasets from the same domain without further training.

### 2 Related Work

**Entity Linking Models**. Entity Linking is a well-studied problem in NLP, especially for long text. Approaches such as (Bunescu and Pasca, 2006; Ratinov et al., 2011; Sil et al., 2012; Hoffart et al., 2011; Shen et al., 2015) use a myriad of classical ML and deep learning models to combine priors, local and global features. These techniques, in general, can be applied to short text, but the lack of sufficient context may render them ineffective. The recently proposed BLINK (Logeswaran et al., 2019;
Wu et al., 2020) uses powerful transformer-based encoder architectures trained on massive amounts of data (such as Wikipedia, Wikia) to achieve SotA performance on entity disambiguation tasks, and is shown to be especially effective in zero-shot settings. BLINK is quite effective on short text (as observed in our findings); in our approach, we use BLINK both as a baseline and as a component that is combined in larger rules.

For short-text EL, some prior works (Sakor et al., 2019; Ferragina and Sciaiella, 2012; Mendes et al., 2011) address the joint problem of mention detection and linking, with primary focus on identifying mention spans, while linking is done via heuristic methods without learning. (Sakor et al., 2019) also jointly extracts relation spans which aide in overall linking performance. The recent ELQ (Li et al., 2020) extends BLINK to jointly learn mention detection and linking. In contrast, we focus solely on linking and take a different strategy based on combining logic rules with learning. This facilitates a principled way combining multiple types of EL features with interpretability and learning using promising gradient-based techniques.

**Rule-based Learning.** FOL rules and learning have been successfully applied in some NLP tasks and also other domains. Of these, the task that is closest to ours is entity resolution (ER), which is the task of linking two entities across two structured datasets. In this context, works like (Chaudhuri et al., 2007; Arasu et al., 2010; Wang et al., 2012; Hernández et al., 2013) use FOL rules for ER. Approaches such as (Singla and Domingos, 2006; Pujara and Getoor, 2016) induce probabilistic rules using MLNs (Richardson and Domingos, 2006) and PSL (Bach et al., 2017), respectively. None of these approaches use any recent advances in neural-based learning; moreover, they are focused on entity resolution, which is a related task but distinct from short-text EL.

### 3 Preliminaries

#### 3.1 Entity Linking

Given text $T$, a set $M = \{m_1, m_2, \ldots\}$ of mentions, where each $m_i$ is contained in $T$, and a knowledge graph (KG) comprising of a set $E$ of entities, entity linking is a many-to-one function that links each mention $m_i \in M$ to an entity $e_{ij} \in C_i$, where $C_i \subseteq E$ is a subset of relevant candidates for mention $m_i$. More generally, we formulate the problem as a ranking of the candidates in $C_i$ so that the “correct” entity for $m_i$ is ranked highest. Following existing approaches (e.g. (Sakor et al., 2019; Wu et al., 2020), we use off-the-shelf lookup tools such as DBpedia lookup\(^2\) to retrieve top-100 candidates for each mention. While this service is specific to DBpedia, we assume that similar services exist or can be implemented on top of other KGs.

#### 3.2 Logical Neural Networks

Fueled by the rise in complexity of deep learning, recently there has been a push towards learning interpretable models (Guidotti et al., 2018; Danilevsky et al., 2020). While linear classifiers, decision lists/trees may also be considered interpretable, rules expressed in first-order logic (FOL) form a much more powerful, closed language that offer semantics clear enough for human interpretation and a larger range of operators facilitating the expression of richer models. To learn these rules, neuro-symbolic AI typically substitutes conjunctions (disjunctions) with differentiable $t$-norms ($t$-conorms) (Esteva and Godo, 2001). However, since these norms do not have any learnable parameters (more details in Appendix A.1), their behavior cannot be adjusted, thus limiting their ability to model well the data.

In contrast, logical neural networks (LNN) (Riegel et al., 2020) offer operators that include parameters, thus allowing to better learn from the data. To maintain the crisp semantics of FOL, LNNs enforce constraints when learning operators such as conjunction. Concretely, LNN-$\land$ is expressed as:

\[
\begin{align*}
\text{max}(0, \min(1, \beta - w_1 (1-x) - w_2 (1-y))) \\
\text{subject to:} \\
\beta - (1 - \alpha)(w_1 + w_2) & \geq \alpha \\
\beta - \alpha w_1 & \leq 1 - \alpha \\
\beta - \alpha w_2 & \leq 1 - \alpha \\
w_1, w_2 & \geq 0
\end{align*}
\]

where $\beta$, $w_1$, $w_2$ are learnable parameters, $x, y \in [0, 1]$ are inputs and $\alpha \in [\frac{1}{2}, 1]$ is a hyperparameter. Note that $\max(0, \min(1, \cdot))$ clamps the output of LNN-$\land$ between 0 and 1 regardless of $\beta$, $w_1$, $w_2$, $x$, and $y$. The more interesting aspects are in the constraints. While Boolean conjunction only returns 1 or true when both inputs are 1, LNNs relax this condition by using $\alpha$ as a proxy for 1 (and conversely, $1 - \alpha$ as a proxy for 0). In particular, Constraint (1) forces the output of LNN-$\land$ to be greater than $\alpha$ when both inputs are greater than $\alpha$. Similarly, Constraints (2) and (3) constrain the

\(^2\)https://lookup.dbpedia.org/
behavior of LNN-∧ when one input is low and the other is high. For instance, Constraint (2) forces the output of LNN-∧ to be less than $1 - \alpha$ for $y = 1$ and $x \leq 1 - \alpha$. This formulation allows for unconstrained learning when $x, y \in [1 - \alpha, \alpha]$. By changing $\alpha$, a user can control how much learning to enable (increase to make region of unconstrained learning wider or decrease for the opposite). Figure 2 depicts product $t$-norm and LNN-∧ ($\alpha = 0.7$).

While the former increases slowly with increasing $x$, $y$, LNN-∧ produces a high output when both inputs are $\geq \alpha$ and stays high thereafter, thus closely modeling Boolean conjunction semantics.

In case the application requires even more degrees of freedom, the hard constraints (1), (2) and (3) can be relaxed via the inclusion of slacks:

$$LNN-\wedge(x, y) = \max(0, \min(1, \beta - w_1(1 - x) - w_2(1 - y)))$$

subject to: $\beta - (1 - \alpha)(w_1 + w_2) + \Delta \geq \alpha$

$\beta - \alpha w_1 \leq 1 - \alpha + \delta_1$

$\beta - \alpha w_2 \leq 1 - \alpha + \delta_2$

$w_1, w_2, \delta_1, \delta_2, \Delta \geq 0$

where $\delta_1, \delta_2$, and $\Delta$ denote slack variables. If any of Constraints (1), (2) and (3) in LNN-∧ are unsatisfied then slacks help correct the direction of the inequality without putting pressure on parameters $w_1, w_2,$ and $\beta$ during training. For the rest of the paper, by LNN-∧ we refer to the above formulation. LNN negation is a pass-through operator: LNN-∨($x$) = $1 - x$, and LNN disjunction is defined in terms of LNN-∧:

$$LNN-\vee(x, y) = 1 - LNN-\wedge(1 - x, 1 - y)$$

While vanilla backpropagation cannot handle linear inequality constraints such as Constraint (1), specialized learning algorithms are available within the LNN framework. For more details, please check Riegel et al. (2020).

4 LNN-EL

An overview of our neuro-symbolic approach for entity linking is depicted in Figure 3. We next discuss the details about feature generation component that generates features using a catalogue of feature functions (Section 4.1) followed by proposed model that does neuro-symbolic learning over user provided EL algorithm in Section 4.2.

Given the input text $T$, together with labeled data in the form $(m_i, C_i, L_i)$, where $m_i \in M$ is a mention in $T$, $C_i$ is a list of candidate entities $e_{ij}$ (drawn from lookup services$^3$) for the mention $m_i$, and where each $l_{ij} \in L_i$ denotes a link/not-link label for the pair $(m_i, e_{ij})$. The first step is to generate a set $F_{ij} = \{f_k(m_i, e_{ij})\}$ of features for each pair $(m_i, e_{ij})$, where $f_k$ is a feature function drawn from a catalog $F$ of user provided functions.

4.1 Feature Functions

Our collection of feature functions include both non-embedding and embedding based functions.

Non-embedding based. We include here a multitude of functions (see Table 1) that measure the similarity between the mention $m_i$ and the candidate entity $e_{ij}$ based on multiple types of scores.

| Features | Description |
|----------|-------------|
| Name | $sim(m_i, e_{ij})$, where $sim$ is a general purpose string similarity function such as Jaccard ($jacc$), Jaro-Winkler ($jw$), Levenshtein ($lev$), Partial Ratio ($pr$), etc. |
| Context | $Ctx(m_i, e_{ij})$ |
| | $= \sum_{m_k \in M \setminus \{m_i\}} pr(m_k, e_{ij}.desc)$ |
| | where $m_k$ is a mention in the context of $m_i$ |
| Type | $Type(m_i, e_{ij})$ |
| | $= \begin{cases} 1 & \text{if } m_i.\text{type} \in e_{ij}.\text{dom} \\ 0, & \text{otherwise} \end{cases}$ |
| | where $m_i.\text{type}$ is the type of the mention and $e_{ij}.\text{dom}$ is the set of domains |
| Entity | $Prom(e_{ij}) = \text{indegree}(e_{ij})$, |
| Prominence | i.e., number of links pointing to entity $e_{ij}$ |

Table 1: Non-embedding based feature functions.

Of course, the non-embedding based features are not sufficient to capture the similarity between mentions and candidates. Therefore, we include embedding based functions as well. The non-embedding based features compute the similarity between mentions and candidates in the text, together with labeled KG resources using: $\text{Jaccard}$ ($jacc$), $\text{Jaro-Winkler}$ ($jw$), $\text{Levenshtein}$ ($lev$), $\text{Partial Ratio}$ ($pr$), $\text{DBpedia}$, etc. are used to compute the similarity between $m_i$ and candidate entity $e_{ij}$.

**Name**: a set of general purpose string similarity functions such as Jaccard, Jaro-Winkler, Levenshtein, Partial Ratio, etc. are used to compute the similarity between $m_i$ and the candidate entity $e_{ij}$ based on multiple types of scores.

**Context**: aggregated similarity of $m_i$’s context to the description of $e_{ij}$. Here, we consider the list of all other mentions $m_k \in M \setminus \{m_i\}$ as $m_i$’s context, together with $e_{ij}$’s textual description obtained using KG resources. The exact formula we use is shown in Table 1, where $\text{Partial Ratio}(pr)$ measures the similarity between each context mention and the description. $\text{(Partial Ratio)}$ computes the

---

3https://lookup.dbpedia.org

4pypi.org/project/py-stringmatching

5dbpedia.org/sparql
maximum similarity between a short input string and substrings of a second, longer string. For normalizing the final score, we apply a min-max rescaling over all entities \( e_{ij} \in C_i \).

**Type:** the overlap similarity of mention \( m_i \)'s type to \( e_{ij} \)'s domain (class) set, similar to the domain-entity coherence score proposed in (Nguyen et al., 2014). Unlike in (Nguyen et al., 2014), instead of using a single type for all mentions in \( M \), we obtain type information for each mention \( m_i \) using a trained BERT-based entity type detection model. We use KG resources \(^5\) to obtain \( e_{ij} \)'s domain set, similar to Context similarity.

**Entity Prominence:** measure the prominence of entity \( e_{ij} \) as the number of entities that link to \( e_{ij} \) in target KG, i.e., \( \text{indegree}(e_{ij}) \). Similar to Context score normalization, we apply min-max rescaling over all entities \( e_{ij} \in C_i \).

**Embedding based.** We also employ a suite of pre-trained or custom trained neural language models to compute the similarity of \( m_i \) and \( e_{ij} \).

**Pre-trained Embedding Models.** These include SpaCy's semantic similarity\(^6\) function that uses Glove (Pennington et al., 2014) trained on Common Crawl. In addition to SpaCy, we also use scores from an entity linking system such as BLINK (Wu et al., 2020) (a state-of-the-art entity linking model) as a feature function in our system.

**BERT Embeddings.** To further explore the semantics of the context in \( T \) and the inherent structure of the target KG, we incorporate an embedding-based similarity by training a mini entity linking model without any aforementioned prior information. We first tag the input text \( T \) with a special token [MENT] to indicate the position of mention \( m_i \), and then encode \( T \) with BERT, i.e., \( m_i = \text{BERT}(m_i, T) \). Each candidate \( e_{ij} \) is encoded with a pre-trained graph embedding Wiki2Vec (Yamada et al., 2020), i.e., \( e_{ij} = \text{Wiki2Vec}(e_{ij}) \). The candidates are ranked in order of the cosine similarity to \( m_i \), i.e., \( \text{Sim}_\text{cos}(m_i, e_{ij}) \). The mini EL model is optimized with margin ranking loss so that the correct candidate is ranked higher.

**BERT with Box Embeddings.** While features such as Context (see Table 1) can exploit other mentions appearing within the same piece of text, they only do so via textual similarity. A more powerful method is to jointly disambiguate the mentions in text to the actual entities in the KG, thus exploiting the structural context in the KG. Intuitively, the simultaneous linking of co-occurring mentions in text to related entities in the KG is a way to reinforce the links for each individual mention. To this end, we adapt the recent Query2Box (Ren et al., 2020), whose goal is to answer FOL queries over a KG. The main idea there is to represent sets of entities (i.e., queries) as contiguous regions in embedded space (e.g., axis-parallel hyper-rectangles or boxes), thus reducing logical operations to geometric operations (e.g., intersection).

Since Query2Box assumes a well-formed query as input, one complication in directly applying it to our setting is that we lack the information necessary to form such an FOL query. For instance, in the example from Section 1, while we may assume that the correct entities for our Cameron and Titanic mentions are connected in the KG, we do not know how these are connected, i.e., via which relation. To circumvent this challenge, we introduce a special neighborhood relation \( \mathcal{N} \), such that \( v \in \mathcal{N}(u) \) whenever there is some KG relation from entity \( u \)

---

\(^6\) spacy.io/usage/vectors-similarity
to entity $v$. We next define two box operations:

$$\text{Box}(C_i) = \{v \mid \min_{e_j \in C_i} \{e_{ij} \mid e_{ij} \in C_i\} \leq v \leq \max_{e_j \in C_i} \{e_{ij} \mid e_{ij} \in C_i\}\}$$

$$\text{Box}(\mathcal{N}(C_i)) = \text{Box}(C_i) + \text{Box}_v$$

The first operation represents mention $m_i$ as a box, by taking the smallest box that contains the set $C_i$ of candidate entities for $m_i$. This can be achieved by computing the dimension-wise minimum (maximum) of all entity embeddings in $C_i$ to obtain the lower-left (upper-right) corner of the resulting box. The second operation takes $m_i$’s box and produces the box containing its neighbors in the KG. Query2Box achieves this by representing $\text{Box}_v$ via a center vector $\psi$ and offset vector $\omega$, both of which are learned parameters. The box of neighbors is then obtained by translating the center of $m_i$’s box by $\psi$ and adding the offset $\omega$ to its side.

Figure 4 shows how these operations are used to disambiguate Titanic while exploiting the co-occurring mention Cameron and the KG structure. We take the box for Cameron, compute its neighborhood box, then intersect with the Titanic box. This intersection contains valid entities that can disambiguate Titanic and are connected to the entity for Cameron. For the actual score of each such entity, we take its distance to the center of the intersection box and convert it to a similarity score $\text{Sim}_{\text{box}}(m_i, e_{ij})$. We then linearly combine this with the BERT-based similarity measure: $\beta_{\text{box}} \cdot \text{Sim}_{\text{box}}(m_i, e_{ij}) + \text{Sim}_{\text{cos}}(m_i, e_{ij})$, where $\beta_{\text{box}}$ is a hyper-parameter that adjusts the importance of the two scores. The approach described can be easily extended to more than two mentions.

### 4.2 Model

In this section, we describe how an EL algorithm composed of a disjunctive set of rules is reformulated into LNN representation for learning. **Entity Linking Rules** are a restricted form of FOL rules comprising of a set of Boolean predicates connected via logical operators: conjunction ($\land$) and disjunction ($\lor$). A Boolean predicate has the form $f_k > \theta$, where $f_k \in \mathcal{F}$ is one of the feature functions, and $\theta$ can be either a user provided or a learned threshold in $[0, 1]$. Figure 5(a) shows two example rules $R_1$ and $R_2$, where, for instance, $R_1(m_i, e_{ij})$ evaluates to $\text{True}$ if both the predicate $\text{jacc}(m_i, e_{ij}) > \theta_1$ and $\text{Ctx}(m_i, e_{ij}) > \theta_2$ are $\text{True}$. Rules can be disjuncted together to form a larger EL algorithm, as the one shown in Figure 5(b), which states that $\text{Links}(m_i, e_{ij})$ evaluates to $\text{True}$ if any one of its rules evaluates to $\text{True}$. The $\text{Links}$ predicate is meant to store high-quality links between mention and candidate entities that pass the conditions of at least one rule. The EL algorithm also acts as a scoring mechanism. In general, there are many ways in which scores can be computed. In a baseline implementation (no learning), we use the scoring function in Figure 5(c), where $r_{w_1}$ denote manually assigned feature weights, while $f_{w_1}$ are manually assigned feature weights.

An EL algorithm is an explicit and extensible description of the entity linking logic, which can be easily understood and manipulated by users. However, obtaining competitive performance to that of deep learning approaches such as BLINK (Wu et al., 2020) requires a significant amount of manual effort to fine tune the thresholds $\theta_i$, the feature weights ($f_{w_i}$) and the rule weights ($r_{w_i}$).

**LNN Reformulation.** To facilitate learning of the thresholds and weights in an EL algorithm, we map the Boolean-valued logic rules into the LNN formalism, where the LNN constructs – LNN-$\land$ (for logical OR) and LNN-$\lor$ (for logical AND) – allow for continuous real-valued numbers in $[0, 1]$. As described in Section 3.2, LNN-$\land$ and LNN-$\lor$ are a weighted real-valued version of the classical logical operators, where a hyperparameter $\alpha$ is used as a proxy for 1. Each LNN operator produces a value in $[0, 1]$ based on the values of the inputs, their weights and bias $\beta$. Both the weights and $\beta$ are learnable parameters. The score of each link is based on the score that the LNN operators give, with an added complication related to how we score the feature functions. To illustrate, for the EL rules in Figure 5, the score of a link is computed as:

$$s(m_i, e_{ij}) = \begin{cases} \text{LNN-$\land$} & \left(\text{TL}(\text{jacc}(m_i, e_{ij}), \theta_1)\right), \\ \text{LNN-$\lor$} & \left(\text{TL}(\text{Ctx}(m_i, e_{ij}), \theta_2)\right) \end{cases}$$

Figure 5: Example of entity linking rules and scoring.
We also discuss the training and inference time. We first evaluate our approach w.r.t performance & extensibility, interpretability and transferability. For the feature functions with thresholds, a natural scoring mechanism would be to use score($f > \theta$) = \( f \) if \( f > \theta \) else 0, which filters out the candidates that do not satisfy the condition \( f > \theta \), and gives a non-zero score for the candidates that pass the condition. However, since this is a step function which breaks the gradient flow through a neural network, we approximate it via a smooth function TL($f, \theta$) = \( f - \sigma(f - \theta) \), where \( \sigma \) is Sigmoid function and \( \theta \) is the learnable threshold that is generated using \( \sigma \), i.e., \( \theta = \sigma(\gamma) \), to ensure that it lies in \([0, 1]\).

Training. We train the LNN formulated EL rules over the labeled data and use a margin-ranking loss over all the candidates in \( C_i \) to perform gradient descent. The loss function \( L(m_i, C_i) \) for mention \( m_i \) and candidates set \( C_i \) is defined as

\[
\sum_{e_{ip} \in C_i \setminus \{e_{ip}\}} \max(0, -(s(m_i, e_{ip}) - s(m_i, e_{im})) + \mu)
\]

Here, \( e_{ip} \) is a positive candidate, \( C_i \setminus \{e_{ip}\} \) is the set of negative candidates, and \( \mu \) is a margin hyper parameter. The positive and negative labels are obtained from the given labels \( L_i \) (see Figure 3).

Inference. Given mention \( m_i \) and candidate set \( C_i \), similar to training, we generate features for each mention-candidate pair \((m_i, e_{ij})\) in the feature generation step. We then pass them through the learned LNN network to obtain final scores for each candidate entity in \( C_i \) as shown in Figure 3.

5 Evaluation

We first evaluate our approach w.r.t performance & extensibility, interpretability and transferability. We also discuss the training and inference time.

Datasets. As shown in Table 2, we consider three short-text QA datasets. LC-QuAD and QALD-9 are datasets comprising of questions (Q) over DBpedia together with their corresponding SPARQL queries. We extract entities (E) from SPARQL queries and manually annotate mention spans. WebQSP\textsubscript{EL} dataset (Li et al., 2020) comprises of both mention spans and links to the correct entity. Since the target KG for WebQSP is Wikidata, we translate each Wikidata entity to its DBpedia counterpart using DBpedia Mappings. In addition, we discard mentions that link to DBpedia concepts (e.g., heaviest player linked to dbo:Person) and mentions \( m_i \) with empty result (i.e., \( C_i = \phi \)) or all not-link labels (i.e, \( \forall l_{ij} \in L_i, l_{ij} = 0 \)).

Baselines. We compare our approach to (1) BLINK (Wu et al., 2020), the current state-of-the-art on both short-text and long-text EL, (2) three BERT-based models - (a) BERT: both mention and candidate entity embeddings are obtained via BERT\textsubscript{base} pre-trained encoder, similar to (Gillick et al., 2019), (b) BERTwiki: mention embeddings are obtained from BERT\textsubscript{base}, while candidate entity is from pretrained Wiki2Vec (Yamada et al., 2020), (c) Box: BERTWiki embeddings finetuned with Query2Box embeddings (see Section 4.1). In addition to the aforementioned black-box neural models, we also compare our approach to (3) two logistic regression models that use the same feature set as LNN-EL: LogisticRegression without BLINK and LogisticRegression\textsubscript{BLINK} with BLINK.

Furthermore, we use the following variants of our approach: (4) RuleEL: a baseline rule-based EL approach with manually defined weights and thresholds, (5) LogicEL: a baseline approach built on RuleEL where only the thresholds are learnable, based on product t-norm (see Section 3.2), (6) LNN-EL: our core LNN-based method using non-embedding features plus SpaCy, and (7) LNN-EL\textsubscript{ens}: an ensemble combining core LNN-EL with additional features from existing EL approaches, namely BLINK and Box (we consider Box, as it outperforms BERT and BERTWiki on all datasets). Detailed rule templates are provided in Appendix A.3.

Setup. All the baselines are trained for 30 epochs, except for BLINK which we use as a zero-shot approach. For BERT approaches, we use BERT\textsubscript{base} as pretrained model. We used two Nvidia V100 GPUs with 16GB memory each. We perform hyper-parameter search for margin \( \mu \) and learning rates in the range \([0.6, 0.95], [10^{-5}, 10^{-1}] \) respectively.

| Dataset          | Train | Test |
|------------------|-------|------|
|                  | I{\textsubscript{Q}} | I{\textsubscript{E}} | I{\textsubscript{Q}} | I{\textsubscript{E}} |
| LC-QuAD 1.0 (Trivedi et al., 2017) | 4,000 | 6,823 | 1,000 | 1,721 |
| QALD-9 (Usbeck et al., 2018) | 408 | 568 | 150 | 174 |
| WebQSP\textsubscript{EL} (Li et al., 2020) | 2974 | 3,237 | 1603 | 1,798 |

Table 2: Characteristics of the datasets.
Table 3: Performance comparison of various baselines with our neuro-symbolic variants.

| Model         | LC-QuAD | QALD-9 | WebQSP EL |
|---------------|---------|--------|-----------|
|               | Precision | Recall | F1  | Precision | Recall | F1  | Precision | Recall | F1  |
| BLINK         | 87.04    | 87.04  | 87.04 | 89.14     | 89.14  | 89.14 | 92.15     | 92.05  | 92.10|
| BERT          | 57.14    | 63.09  | 59.97 | 55.46     | 61.11  | 58.15 | 70.26     | 72.15  | 71.20|
| BERTWiki      | 66.96    | 73.85  | 70.23 | 66.16     | 72.90  | 69.37 | 81.11     | 83.29  | 82.19|
| Box           | 67.31    | 74.32  | 70.64 | 68.91     | 75.93  | 72.25 | 81.53     | 83.72  | 82.61|
| LogisticRegression | 87.04    | 86.83  | 86.93 | 84.73     | 84.73  | 84.73 | 83.39     | 83.33  | 83.36|
| LogisticRegression BLINK | 90.50    | 90.30  | 90.40 | 90.94     | 90.94  | 90.94 | 89.33     | 89.28  | 89.31|
| RuleEL        | 79.82    | 80.10  | 79.96 | 81.55     | 75.15  | 78.22 | 76.56     | 74.55  | 75.54|
| LogicEL       | 86.68    | 86.48  | 86.58 | 83.05     | 83.05  | 83.05 | 82.60     | 82.58  | 82.59|
| LNN-EL        | 87.74    | 87.54  | 87.64 | 88.52     | 88.52  | 88.52 | 85.11     | 85.05  | 85.08|
| LNN-EL_{ens}  | 91.10    | 90.90  | 91.00 | 91.38     | 91.38  | 91.38 | 92.17     | 92.08  | 92.12|

Table 4: Recall@k performance of LNN-EL models

| Dataset     | Model       | R@5 | R@10 | R@64 |
|-------------|-------------|-----|------|------|
| LC-QuAD     | BLINK       | 94.69 | 96.01 | 96.92 |
|             | LNN-EL      | 93.66 | 94.39 | 97.56 |
|             | LNN-EL_{ens} | 97.07 | 97.20 | 97.68 |
| QALD-9      | BLINK       | 93.39 | 93.39 | 94.29 |
|             | LNN-EL      | 92.72 | 95.94 | 98.04 |
|             | LNN-EL_{ens} | 94.63 | 94.63 | 95.48 |
| WebQSP EL   | BLINK       | 97.40 | 97.64 | 98.61 |
|             | LNN-EL      | 93.54 | 95.12 | 96.59 |
|             | LNN-EL_{ens} | 96.34 | 96.59 | 96.95 |

Table 5: F1 scores of LNN-EL with additional features coming from various black-box EL approaches.

| Dataset     | Model         | Precision | Recall | F1  |
|-------------|---------------|-----------|-------|-----|
| LC-QuAD     | BLINK         | 87.64     | 90.24 | 88.23 | 89.05 | 91.00 |
| QALD-9      | LNN-EL_{ens}  | 88.52     | 90.06 | 86.41 | 88.52 | 91.38 |
| WebQSP EL   | LNN-EL_{ens}  | 85.08     | 92.32 | 91.70 | 91.44 | 92.12 |

5.1 Results

Overall Performance. As seen in Table 3, among logic-based approaches, LNN-EL outperforms LogicEL and RuleEL, showing that parameterized real-valued LNN learning is more effective than the non-parameterized version with t-norm (LogicEL) and the manually tuned RuleEL. Logistic regression models which also learn weights over features achieve competitive performance to LNN-EL models; however they lack the representation power that LNN-EL offer in the form of logical rules comprising of conjunctions and disjunctions. In other words, LNN-EL allows learning over a richer space of models that help in achieving better performance as observed in Table 3.

On the other hand, simple BERT-based approaches (BERT, BERTWiki, Box) that are trained on the QA datasets underperform the logic-based approaches, which incorporate finer-grained features. BLINK (also a BERT-based approach, but trained on the entire Wikipedia) is used as zero-shot approach and achieves SotA performance (when not counting the LNN-EL variants). The core LNN-EL version is competitive with BLINK on LC-QuAD and QALD-9, despite being a rule-based approach. Furthermore, LNN-EL_{ens}, which combines the core LNN-EL with both BLINK and Box features, easily beats BLINK on LC-QuAD and QALD-9 and slightly on WebQSP EL.

Table 4 shows the Recall@k performance of LNN-EL against the BLINK model. Both LNN-EL and LNN-EL_{ens} have better Recall@k performance against BLINK on LC-QuAD and QALD-9 datasets, however BLINK’s Recall@k achieves a slightly better performance for WebQSP EL dataset.

Extensibility. Here, we inspect empirically how a multitude of EL features coming from various black-box approaches can be combined in a principled way with LNN-EL, often leading to an overall better performance than the individual approaches. A detailed ablation study of the core LNN-EL version can be found in Appendix A.2. As seen in Table 5, approaches like BERTWiki and Box which in isolation underperform compared to LNN-EL, help boost the latter’s performance if they are included as predicates. Similarly, LNN-EL which has comparable performance to BLINK, can accommodate the latter’s score to produce better performance (see LNN-EL_{+BLINK}). We also note that adding features is not a guarantee to improve performance, as LNN-EL_{+BLINK} (which includes both BLINK and Box) slightly underperforms LNN-EL_{+BLINK} on WebQSP EL. For such cases, the interpretability of LNN-EL (discussed next) can help users select the right features based on their relative importance.

Interpretability. Unlike black-box models, rule-based approaches provide the capability to inspect the model, specifically on how the features impact performance. This inspection can help in dropping or adjusting features that are detrimental. For instance, consider our case of LNN-EL_{+BLINK} and LNN-EL_{ens} trained on WebQSP EL dataset, where we observed that LNN-EL_{ens}’s performance is inferior to LNN-EL_{+BLINK} even though the former model has more features. A human expert can find
insights into this behavior by looking at the feature weights in each model. In Figure 6 (left), the disjunction tree with the Box feature is given a low weight of 0.26, thus discounting some of the other useful features in the same tree. Removal of the Box feature leads to a re-weighting of the features in the model; the modified disjunction tree (Figure 6 (left)) has now a weight of 0.42. Such visualization can help the rule designer to judiciously select features to combine towards building a performant model.

Transferability. To study the transferability aspect, we train LNN-EL on one dataset and evaluate the model on the other two, without any finetuning. We use the core LNN-EL variant for this, but similar properties hold for the other variants. Table 6 shows F1 scores on different train-test configurations, with diagonal (underlined numbers) denoting the F1 score when trained and tested on the same dataset. We observe that LNN-EL transfers reasonably well, even in cases where training is done on a very small dataset. For example, when we transfer from QALD-9 (with only a few hundred questions to train) to WebQSPEL, we obtain an F1-score of 83.06 which is within 2 percentage points of the F1-score when trained directly on WebQSPEL. We remark that the zero-shot BLINK by design has very good transferability and achieves F1 scores of 87.04, 89.14, 92.10 on LC-QuAD, QALD-9, WebQSPEL respectively. However, BLINK is trained on the entire Wikipedia, while LNN-EL needs much less data to achieve reasonable transfer performance.

Runtime Analysis. We study the efficiency of LNN-EL across three aspects: 1) candidate & feature generation, 2) training, and 3) inference. Candidate & feature generation involve using the DBpedia lookup API to obtain candidates for each mention, pruning non-entity candidates (i.e., categories, disambiguation links, etc.), obtaining any missing descriptions for candidates using SPARQL endpoint, and finally generating feature vectors for each mention-candidate pair using the feature functions described in Section 4.1. The generated features for the train and test data are then used, respectively, to train and test the LNN-EL models. The number of parameters in an LNN-EL model is linearly proportional to the combined number of disjunctions and conjunctions, which typically is in the order of few 10s. For example, LNN-EL comprises of 72 parameters, which is several orders of magnitude smaller than in neural black box models such as BLINK. Table 7 provides the time (in seconds) taken per question for candidate & feature generation, as well as 5-run average training and inference time per epoch.

6 Conclusions

We introduced LNN-EL, a neuro-symbolic approach for entity linking on short text. Our approach complements human-given rule templates through neural learning and achieves competitive performance against SotA black-box neural models, while exhibiting interpretability and transferability without requiring a large amount of labeled data. While LNN-EL provides an extensible framework where one can easily add and test new features in existing rule templates, currently this is done manually. A future direction is to automatically learn the rules with the optimal combinations of features.

Acknowledgements

We thank Ibrahim Abdelaziz, Pavan Kapanipathi, Srinivas Ravishankar, Berthold Reinwald, Salim Roukos and anonymous reviewers for their valuable inputs and feedback.
References

Arvind Arasu, Michaela Götz, and Raghav Kaushik. 2010. On active learning of record matching packages. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD ’10, page 783–794, New York, NY, USA. Association for Computing Machinery.

Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2017. Hinge-loss markov random fields and probabilistic soft logic. J. Mach. Learn. Res., 18(1):3846–3912.

Razvan Bunescu and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06), pages 9–16, Trento, Italy.

S. Chaudhuri, Bee-Chung Chen, V. Ganti, and R. Kaushik. 2007. Example-driven design of efficient record matching queries. In VLDB.

Silviu Cucerzan. 2007. 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, Prague, Czech Republic. Association for Computational Linguistics.

Marina Danilevsky, Kun Qian, Ranit Aharonov, Yannis Katsis, Ban Kawas, and Prithviraj Sen. 2020. A survey of the state of explainable AI for natural language processing. In Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pages 447–459, Suzhou, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

F. Esteva and L. Godo. 2001. Monoidal t-norm based logic: Towards a logic for left-continuous t-norms. Fuzzy Sets and Systems.

Richard Evans and Edward Grefenstette. 2018. Learning explanatory rules from noisy data. JAIR.

Paolo Ferragina and Ugo Scaiella. 2012. Fast and accurate annotation of short texts with wikipedia pages. IEEE Softw., 29(1):70–75.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2619–2629, Copenhagen, Denmark. Association for Computational Linguistics.

Lise Getoor and Ben Taskar. 2007. Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning). The MIT Press.

Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Balbridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. ACM Computing Surveys.

Mauricio Hernández, Georgia Koutrika, Rajasekar Krishnamurthy, Lucian Popa, and Ryan Wisnesky. 2013. Hil: A high-level scripting language for entity integration. In Proceedings of the 16th International Conference on Extending Database Technology, EDBT ’13, page 549–560, New York, NY, USA. Association for Computing Machinery.

Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2012. Kore: Keyphrase overlap relatedness for entity disambiguation. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM ’12, page 545–554, New York, NY, USA. Association for Computing Machinery.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordin, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 782–792, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander Gray, Ramon Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, et al. 2021. Leveraging abstract meaning representation for knowledge base-question answering. Findings of the Association for Computational Linguistics: ACL 2021.

Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. In EMNLP.

Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. 2019. Zero-shot entity linking by reading entity descriptions. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3449–3460, Florence, Italy. Association for Computational Linguistics.

Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. Dbpedia spotlight: shedding light on the web of documents. In Proceedings of the 7th international conference on semantic systems, pages 1–8.
Rada Mihalcea and Andras Csomai. 2007. Wikify! linking documents to encyclopedic knowledge. In Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, CIKM ’07, page 233–242, New York, NY, USA. Association for Computing Machinery.

Stephen Muggleton. 1996. Learning from positive data. In Workshop on ILP.

D. Nguyen, Johannes Hoffart, M. Theobald, and G. Weikum. 2014. Aida-light: High-throughput named-entity disambiguation. In LDOW.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543.

J. Pujara and L. Getoor. 2016. Generic statistical relational entity resolution in knowledge graphs. ArXiv, abs/1607.00992.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to Wikipedia. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 1375–1384, Portland, Oregon, USA. Association for Computational Linguistics.

Hongyu Ren, Weihua Hu, and Jure Leskovec. 2020. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. Mach. Learn., 62(1–2):107–136.

Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndiuvhuwo Makondo, Ismail Yunus Akhalwaya, HaiFeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, Shajith Ikbai, Hima Karanam, Sumit Neelam, Ankita Likhyan, and Santosh Srivastava. 2020. Logical neural networks. arXiv.

Ahmad Sakor, Isaiah Onando Mulang’, Kuldeep Singh, Saeedeh Shekarpour, Maria Esther Vidal, Jens Lehmann, and Sören Auer. 2019. Old is gold: Linguistic driven approach for entity and relation linking of short text. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 2336–2346, Minneapolis, Minnesota. Association for Computational Linguistics.

W. Shen, J. Wang, and J. Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. IEEE Transactions on Knowledge and Data Engineering, 27(2):443–460.
A Appendix

A.1 t-norm and t-conorm

While linear classifiers, decision lists/trees may also be considered interpretable, rules expressed in first-order logic (FOL) form a much more powerful, closed language that offer semantics clear enough for human interpretation and a larger range of operators facilitating the expression of richer models. To learn these rules, neuro-symbolic AI substitutes conjunctions (disjunctions) with differentiable t-norms (t-conorms) (Esteva and Godo, 2001). However, since it does not have any learnable parameters, this behavior cannot be adjusted, which limits how well it can model the data. For example, while linear classifiers such as logistic regression can only express a (weighted) sum of features which is similar to logic’s disjunction (∨) operator, logic also contains other operators including, but not limited to, conjunction (∧), and negation (¬).

As opposed to inductive logic programming (Muggleton, 1996) and statistical relational learning (Getoor and Taskar, 2007), neuro-symbolic AI utilizes neural networks to learn rules. Towards achieving this, the first challenge to overcome is that classical Boolean logic is non-differentiable and thus, not amenable to gradient-based optimization (e.g., backpropagation). To address this, neuro-symbolic AI substitutes conjunctions (disjunctions) with differentiable t-norms (t-conorms) (Esteva and Godo, 2001). For example, product t-norm, used in multiple neuro-symbolic rule-learners (Evans and Grefenstette, 2018; Yang et al., 2017), is given by \( x \land y \equiv xy \), where \( x, y \in [0, 1] \) denote input features in real-valued logic. Product t-norm agrees with Boolean conjunction at the extremities, i.e., when \( x, y \) are set to 0 (false) or 1 (true). However, when \( x, y \in [0, 1] \setminus \{0, 1\} \), its behavior is governed by the product function. More importantly, since it does not have any learnable parameters, this behavior cannot be adjusted, which limits how well it can model the data.

A.2 Ablation Study

To understand the roles of each rule in LNN-EL, we also conduct ablation study on the largest benchmark dataset LC-QuAD (see Table 8). We observe that Context is the most performant rule alone. Although PureName rule is behind the other two alone, PureName + Context improves the performance of Context by 1%. Meanwhile, Context + Type only improves Context’s performance by 0.05%. Interestingly, the combination of three rules performs slightly worse than PureName + Context by 0.35%. These results show that Type rule is less important among the three rules. To be consistent with the RuleEL system, we apply “PureName + Context + Type” setting for LNN-EL in our experiments.

| Dataset | Precision | Recall | F1   |
|---------|-----------|--------|------|
| PureName  | 76.03     | 75.83  | 75.93|
| + Context | 88.09     | 87.89  | 87.99|
| + Type    | 87.74     | 87.54  | 87.64|
| + Type    | 81.46     | 81.26  | 81.36|
| Context  | 88.09     | 86.83  | 86.93|
| + Type    | 87.04     | 86.88  | 86.98|
| Type     | 87.04     | 86.83  | 86.93|

Table 8: LNN-EL Ablation Analysis on LC-QuAD

Additionally, we also show the transferability of LR in Table 9. This must be compared with the corresponding LNN-EL results in the earlier Table 6. In particular, we observe that LNN-EL outperforms LR in 4 out of 6 transferability tests, demonstrating that LNN-EL has superior transferability.

A.3 LNN-EL Rules

In our experiments, we explore the following modules, implemented in PyTorch.

Name Rule:

\[
R_{\text{name}} \leftarrow [f_{\text{acc}}(m_i, e_{ij}) > \theta_1 \lor f_{\text{lev}}(m_i, e_{ij}) > \theta_2 \\
\lor f_{\text{w}}(m_i, e_{ij}) > \theta_3 \lor f_{\text{spacy}}(m_i, e_{ij}) > \theta_4] \\
\land f_{\text{prom}}(m_i, e_{ij})
\]

Context Rule:

\[
R_{\text{ctx}} \leftarrow [f_{\text{acc}}(m_i, e_{ij}) > \theta_1 \lor f_{\text{lev}}(m_i, e_{ij}) > \theta_2 \\
\lor f_{\text{w}}(m_i, e_{ij}) > \theta_3 \lor f_{\text{spacy}}(m_i, e_{ij}) > \theta_4] \\
\land f_{\text{ctx}}(m_i, e_{ij}) > \theta_5 \\
\land f_{\text{prom}}(m_i, e_{ij})
\]

Type Rule:

\[
R_{\text{type}} \leftarrow [f_{\text{acc}}(m_i, e_{ij}) > \theta_1 \lor f_{\text{lev}}(m_i, e_{ij}) > \theta_2 \\
\lor f_{\text{w}}(m_i, e_{ij}) > \theta_3 \lor f_{\text{spacy}}(m_i, e_{ij}) > \theta_4] \\
\land f_{\text{type}}(m_i, e_{ij}) > \theta_5 \\
\land f_{\text{prom}}(m_i, e_{ij})
\]

Blink Rule:

\[
R_{\text{blink}} \leftarrow [f_{\text{acc}}(m_i, e_{ij}) > \theta_1 \lor f_{\text{lev}}(m_i, e_{ij}) > \theta_2 \\
\lor f_{\text{w}}(m_i, e_{ij}) > \theta_3 \lor f_{\text{spacy}}(m_i, e_{ij}) > \theta_4] \\
\land f_{\text{link}}(m_i, e_{ij})
\]
| Train | LC-QuAD | Test QALD-9 | WebQSPxL |
|------|---------|-------------|-----------|
| LC-QuAD | 86.93 | 84.73 | 76.72 |
| QALD-9 | 87.14 | 84.73 | 80.03 |
| WebQSPxL | 83.42 | 86.83 | 83.59 |

Table 9: F1 scores of LR in transfer settings.

Box Rule:

$$R_{box} \leftarrow [f_{jacc}(m_i, e_{ij}) > \theta_1 \lor f_{lev}(m_i, e_{ij}) > \theta_2 \lor f_{jw}(m_i, e_{ij}) > \theta_3 \lor f_{spacy}(m_i, e_{ij}) > \theta_4] \lor f_{box}(m_i, e_{ij}) > \theta_5$$

BERT Rule:

$$R_{bert} \leftarrow [f_{jacc}(m_i, e_{ij}) > \theta_1 \lor f_{lev}(m_i, e_{ij}) > \theta_2 \lor f_{jw}(m_i, e_{ij}) > \theta_3 \lor f_{spacy}(m_i, e_{ij}) > \theta_4] \lor f_{bert}(m_i, e_{ij}) > \theta_5$$

LNN-EL:

$$R_{LNN-EL} \leftarrow R_{name} \lor R_{ctx} \lor R_{type}$$

LNN-EL+BLINK:

$$R_{LNN-EL+BLINK} \leftarrow R_{LNN-EL} \lor R_{blink}$$

LNN-ELens:

$$R_{LNN-ELens} \leftarrow R_{LNN-EL} \lor R_{blink} \lor R_{box}$$