Maximum Margin Reward Networks for Learning from Explicit and Implicit Supervision

Haoruo Peng[1] **Ming-Wei Chang**[2] **Wen-tau Yih**[2]
[1]University of Illinois, Urbana-Champaign
[2]Microsoft Research, Redmond
[1]hpeng7@illinois.edu [2]{minchang,scottyih}@microsoft.com

Abstract

Neural networks have achieved state-of-the-art performance on several structured-output prediction tasks, trained in a fully supervised fashion. However, annotated examples in structured domains are often costly to obtain, which thus limits the applications of neural networks. In this work, we propose Maximum Margin Reward Networks, a neural network-based framework that aims to learn from both explicit (full structures) and implicit supervision signals (delayed feedback on the correctness of the predicted structure). On named entity recognition and semantic parsing, our model outperforms previous systems on the benchmark datasets, CoNLL-2003 and WebQuestionsSP.

1 Introduction

Structured-output prediction problems, where the goal is to determine values of a set of inter-dependent variables, are ubiquitous in NLP. Structures of such problems can range from simple sequences like part-of-speech tagging (Ling et al., 2015) and named entity recognition (Lample et al., 2016), to complex syntactic or semantic analysis such as dependency parsing (Dyer et al., 2015) and semantic parsing (Dong and Lapata, 2016). State-of-the-art methods of these tasks are often neural network models trained using fully annotated structures, which can be costly or time-consuming to obtain. Weakly supervised learning settings, where the algorithm assumes only the existence of *implicit* signals on whether a prediction is correct, are thus more appealing in many scenarios.

For example, Figure 1 shows a weakly supervised setting of learning semantic parsers using only question–answer pairs. When the system generates a candidate semantic parse during training, the quality needs to be indirectly measured by comparing the derived answers from the knowledge base and the provided labeled answers.

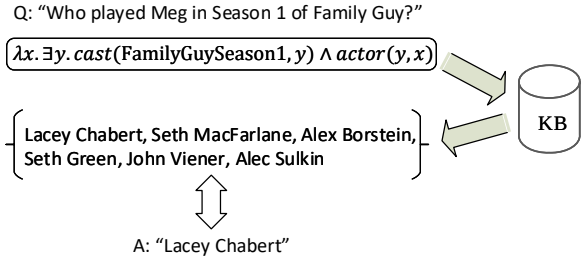

Figure 1: Learning a semantic parser using *implicit* supervision signals (labeled answers). Since there are no gold parses, a model needs to explore different parses, where their quality can only be indirectly verified by comparing retrieved answers and the labeled answers.

This setting of implicit supervision increases the difficulty of learning a neural model, not only because the signals are vague and noisy, but also *delayed*. For instance, among different semantic parses that result in the same answers, typically only few of them correctly represent the meaning of the question. Moreover, the correctness of answers corresponding to a parse can only be evaluated through an external oracle (e.g., executing the query on the knowledge base) *after* the parse is fully constructed. Early model update before the search of a full semantic parse is complete is generally infeasible.[1] It is also not clear how to leverage implicit and explicit signals integrally during learning when both kinds of labels are present.

In this work, we propose Maximum Margin Reward Networks (MMRN), which is a general neural network-based framework that is able to learn from both implicit and explicit supervision signals. By casting structured-output learning as a search problem, the key insight in MMRN is the

[1]Existing weakly supervised methods (Clarke et al., 2010; Artzi and Zettlemoyer, 2013) often leverage domain-specific heuristics, which are not always available.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2351–2361
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

special mechanism of *rewards*. Rewards can be viewed as the training signals that drive the model to explore the search space and to find the correct structure. The explicit supervision signals can be viewed as a source of *immediate* rewards, as we can often instantly know the correctness of the current action. On the other hand, the implicit supervision can be viewed as a source of *delayed* rewards, where the reward of the actions can only be revealed later. We unify these two types of reward signals by using a maximum margin update, inspired by structured SVM (Joachims et al., 2009).

The effectiveness of MMRN is demonstrated on three NLP tasks: *named entity recognition*, *entity linking* and *semantic parsing*. MMRN outperforms the current best results on CoNLL-2003 named entity recognition dataset (Tjong Kim Sang and De Meulder, 2003), reaching 91.4% F_1, in the close setting where no gazetteer is allowed. It also performs comparably to the existing state-of-the-art systems on entity linking. Models for these two tasks are trained using explicit supervision. For semantic parsing, where only implicit supervision signals are provided, MMRN is able to learn from delayed rewards, improving the entity linking component and the overall semantic parsing framework jointly, and outperforms the best published system by 1.4% absolute on the WebQSP dataset (Yih et al., 2016).

In the rest of the paper, we survey the most related work in Sec. 2 and give an in-depth discussion on comparing MMRN and other learning frameworks in Sec. 7. We start the description of our method from the search formulation and the state–action spaces in our targeted tasks in Sec. 3, followed by the reward and learning algorithm in Sec. 4 and the detailed neural model design in Sec. 5. Sec. 6 reports the experimental results and Sec. 8 concludes the paper.

2 Related Work

Structured output prediction tasks have been studied extensively in the field of natural language processing (NLP). Many supervised structured learning algorithms has been proposed for capturing the relationships between output variables. These models include structured perceptron (Collins, 2002; Collins and Roark, 2004), conditional random fields (Lafferty et al., 2001), and structured SVM (Taskar et al., 2004; Joachims et al., 2009). Later, the *learning to search* framework is proposed (Daumé and Marcu, 2005; Daumé et al., 2009), which casts the structured prediction task as a general search problem. Most recently, recurrent neural networks such as LSTM models (Hochreiter and Schmidhuber, 1997) have been used as a general tool for structured output models (Vinyals et al., 2015).

Latent structured learning algorithms address the problem of learning from incomplete labeled data (Yu and Joachims, 2009; Quattoni et al., 2007). The main difference compared to our framework is the existence of the external environment when learning from implicit signals.

Upadhyay et al. (2016) first proposed the idea of learning from implicit supervision, and is the most related paper to our work. Compared to their linear algorithm, our framework is more principled and general as we integrate the concept of margin in our method. Furthermore, we also extend the framework using neural models.

3 Search-based Inference

In our framework, predicting the best structured output, *inference*, is formulated as a state/action search problem. Our search space can be described as follows. The initial state, s_0, is the starting point of the search process. We define $\gamma(s)$ as the set of all feasible actions that can be taken at s, and denote $s' = \tau(s, a)$ as the transition function, where s' is the new state after taking action a from s. A path $\mathbf{h}$ is a sequence of state–action pairs, starting with the initial state: $\mathbf{h} = \{(s_0, a_0), \ldots, (s_k, a_k)\}$, where $s_i = \tau(s_{i-1}, a_{i-1}), \forall i = 1, \ldots, k$. We denote $\mathbf{h} \rightsquigarrow \hat{s}$, if $\hat{s} = \tau(s_k, a_k)$, the final state which the path $\mathbf{h}$ leads to. A path essentially is a partial or complete structured prediction. For each input x, we define $\mathcal{H}(x)$ to be the set of all possible paths for the input. We also define $\mathcal{E}(x) = \{\mathbf{h} \mid \mathbf{h} \in \mathcal{H}(x), \mathbf{h} \rightsquigarrow \hat{s}, \gamma(\hat{s}) = \emptyset\}$, which is all possible paths that lead to terminal states.

Given a state s and an action a, the scoring function $f_\theta(s, a)$ measures the quality of an immediate action with respect to the current state, where θ is the model parameters. The score of a path $\mathbf{h}$ is defined as the sum of the scores for state-action pairs in $\mathbf{h}$: $f_\theta(\mathbf{h}) = \sum_{i=0}^{k} f_\theta(s_i, a_i)$. During test time, inference is to find the *best* path in $\mathcal{E}(x)$: $\arg\max_{\mathbf{h} \in \mathcal{E}(x)} f_\theta(\mathbf{h}; x)$. In practice, inference is often approximated by beam search when no efficient algorithm exists.

In the remaining of this section, we describe the states and actions in the targeted tasks in this work: *named entity recognition*, *entity linking* and *semantic parsing*. The the model and learning algorithm will be discussed in Sec. 4 and Sec. 5.

3.1 Named entity recognition

The task of named entity recognition (NER) is to identify entity mentions in a sentence, as well as to assign their types, such as Person or Location. Following the conventional setting, we treat it as a sequence labeling problem using the standard BIOES encoding. For instance, a "B-LOC" tag on a word means that the word is the beginning of a multi-word location entity.

Given a sentence as input, the states represent the tags assigned to the words. Starting from the initial state, s_0, where no tag has been assigned, the search process explores the sequence tagging from the left-to-right order. For each word, the actions are the legitimate tags that can be assigned to it, which depend on previous actions. For example, if the "S-PER" tag ("S" means a single word entity) has been assigned to the previous word, then an action of labeling the current word with either "I-PER" or "E-PER" cannot can be taken. The search reaches a terminal state when all words in the sentence have been tagged.

3.2 Entity linking

The problem of entity linking (EL) is similar to NER, but instead of tagging the mention using a small set of generic entity types, the goal here is to ground the mention to a specific entity, stored in a knowledge base or described by a Wikipedia page. For example, consider the sentence "*nfl news: draft results for giants*" and assume that the mention candidates "*nfl*" and "*giants*" are given. A state reflects how we have assigned the entity labels to these candidates. Following the same left-to-right order and starting from the empty assignment s_0, the first action to take is to assign the entity label to the first candidate "*nfl*". A legitimate action set can be all the entities that have been associated with this mention in the training set (e.g., "*National Football League*" or "*National Fertilizers Limited*"). Once the action is completed, the transition function will bring the focus to the next mention candidate (i.e., "*giants*"). The search reaches a terminal state when all the candidate mentions in the sentence have been linked.

$$\lambda x.\, \exists y.\, cast(\text{FamilyGuySeason1}, y) \wedge actor(y, x) \wedge character(y, \text{MegGriffin})$$

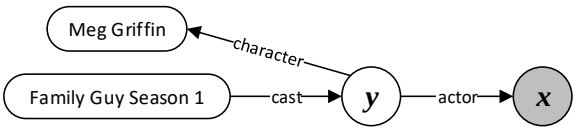

Figure 2: Semantic parses in λ-calculus (top) and query graph (bottom) of the question "*who played meg in season 1 of family guy?*"

3.3 Semantic parsing

Our third targeted task is semantic parsing (SP), which is a task of mapping a text utterance to a formal meaning representation. In this paper, we focus on a specific type of semantic parsing problem that maps a natural language question to a structured query, which is executed on a knowledge base to retrieve the answer to the original question.

Figure 2 shows the semantic parses of an example question "*who played meg in season 1 of family guy*", assuming the knowledge base is Freebase (Bollacker et al., 2008). An entity linking component plays an important role by mapping "*meg*" to `MegGriffin` and "*season 1 of family guy*" to `FamilyGuySeason1`. Predicates like `cast`, `actor` and `character` are also from the knowledge base that define the relationships between these entities and the answer. Together the semantic parse in λ-calculus is shown in the top of Figure 2. Equivalently, the semantic parse can be represented as a *query graph* (Figure 2 bottom), which is used in the STAGG system (Yih et al., 2015). The nodes are either grounded entities or variables, where x is the answer entity. The edges denote the relationship between two entities.

Regardless of the choice of the formal language, the process of constructing the semantic parse is typically formulated as a search problem. A state is essentially a partial or complete semantic parse, and an action is to extend the current semantic parse by adding a new relation or constraint.

Different from previous systems which treat entity linking as a static component, our search space consists of the search space of ***both*** entity linking and semantic parsing. That is, the search space is the union of the search space of entity linking described in Section 3.2 and the search space of the semantic parses, which we describe below. Integrating search spaces allows the model to use implicit signals to update both the semantic parsing

and the entity linking systems. To the best of our knowledge, this is the first work that jointly learns the entity linking and semantic parsing systems.

Our search space is defined as follows. Starting from the initial state s_0, the model first explores the entity linking search space. Once the entity linking assignment are assigned (e.g. `FamilyGuySeason1` in Figure 2.) The second phase is then to determine the main relationship between the topic entity and the answer (e.g., the `cast-actor` chain between `FamilyGuySeason1` and x). Constraints (e.g., the `character` is `MegGriffin`) that describe the additional properties that the answer needs to have are added last. In this case, any state that is a legitimate semantic parse (consisting of one topic entity and one main relationship, as well as zero or more constraints) can lead to a terminal state.

4 Maximum Margin Reward Networks

In this section, we introduce the learning framework of `MMRN`, which includes two main components: *reward* and *max-margin loss*. The former is a mechanism for using implicit and explicit supervision signals in a unified way; the latter formally defines the learning objective.

4.1 Reward

The key insight of `MMRN` is that different types of supervision signals can be represented using the appropriate design of the reward function. A reward function is defined over a state–action pair $R(s, a)$, representing the *true* quality of taking action a in the state s. The reward for a path can be formally defined as: $R(\mathbf{h}) = \sum_{i=0}^{k} R(s_i, a_i)$. Intuitively, when the annotated action sequences (explicit supervision signals) exist, the model only needs to learn to imitate the annotated sequence. For instance, when learning NER in the fully supervised setting, the equivalent way of using Hamming distance is to define the reward $R(s, a)$ to be 1 if a matches the annotated sequence at the current state, and 0 otherwise.

In the setting where only implicit supervision is available, the reward function can still be designed to capture the signals. For instance, when only the question–answer pairs exist for learning the semantic parser, the reward can be defined by comparing the answers derived from a candidate parse and the labeled answers. More formally, assume that $s = \tau(s', a)$ is the state after applying action a to state s'. Let $Y(s)$ be the set of predicted answers generated from state s, and $Y(s) = \{\}$ when s is not a legitimate semantic parse. The reward function $R(s', a)$ can be defined by comparing $Y(s)$ and the labeled answers, A, to the input question. While a set similarity function like the Jaccard coefficient can be used as the reward function, we chose the F_1 score in this work as it was used as the evaluation metric in previous work (Berant et al., 2013). Figure 3 shows an example of this reward function.

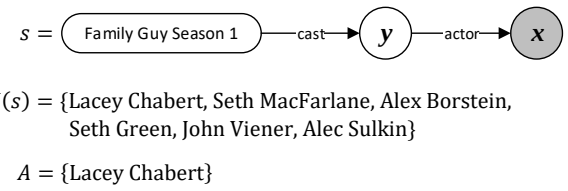

Figure 3: For the question "*who played meg in season 1 of family guy?*", the candidate semantic parse s lists all the actors in "Family Guy Season 1" ($Y(s)$). By comparing $Y(s)$ to the answer set A, the precision is $\frac{1}{6}$ and the recall is 1. Therefore, the F_1 score used for the reward is $\frac{2}{7}$.

4.2 Max-Margin Loss & Learning Algorithm

The `MMRN` learning algorithm can be viewed as an extension of M^{3}N (Taskar et al., 2004) and Structured SVM (Joachims et al., 2009; Yu and Joachims, 2009). The learning algorithm takes three steps, where the first two involve two different search procedures. The final step is to update the models with respect to the inference results.

Finding the best path The first search step is to find the *best* path $\mathbf{h}^*$ by solving the following optimization problem:

$$\mathbf{h}^* = \arg\max_{\mathbf{h} \in \mathcal{E}(x)} R(\mathbf{h}; y) + \epsilon f_\theta(\mathbf{h}). \quad (1)$$

The first term defines the path that has the highest reward. Because it is possible that several paths share the same reward, the second term leverages the current model and serves as the tie-breaker, where ϵ is a hyper-parameter that is set to a small positive number in our experiments.

When explicit supervision is available, solving Eq. (1) is trivial – the search simply returns the annotated sequence. In the case of implicit supervision, where true rewards are only revealed for complete action sequences, the search problem becomes difficult as the rewards of early state–action

pairs are zeros. In this situation, the search algorithm uses the model score f_θ to guide the search. One possible design is to use beam search for the optimization problem, where the search procedure follows the current model in the early stage (given that $R(\mathbf{h}) = 0$). After generating several complete action sequences, the true reward function is then used to find $\mathbf{h}^*$. The tie-breaker also picks the best sequence when there are multiple sequences that lead to the same reward. Note that $\mathbf{h}^*$ can change between iterations because of the tie-breaker.

Finding the most violated path Once $\mathbf{h}^*$ is found, it is used as our reference path. We would like to update the model so that the scoring function f_θ will behave similarly to the reward R. More formally, we aim to update the model parameters θ to satisfy the following constraint.

$$f_\theta(\mathbf{h}^*) - f_\theta(\mathbf{h}) \geq R(\mathbf{h}^*) - R(\mathbf{h}), \forall \mathbf{h}.$$

The constraint implies that the "best" action sequence should rank higher than any other sequence by a margin computed from rewards as $R(\mathbf{h}^*) - R(\mathbf{h})$. The degree of violation of this constraint, with respect to $\mathbf{h}$, is thus $(R(\mathbf{h}^*) - R(\mathbf{h})) - (f_\theta(\mathbf{h}^*) - f_\theta(\mathbf{h})) = f_\theta(\mathbf{h}) - R(\mathbf{h}) - f_\theta(\mathbf{h}^*) + R(\mathbf{h}^*)$. The max-margin loss is defined accordingly:

$$\mathcal{L}(\mathbf{h}, \mathbf{h}^*) = \max(f_\theta(\mathbf{h}) - R(\mathbf{h}) - f_\theta(\mathbf{h}^*) + R(\mathbf{h}^*), 0)$$

$\mathcal{L}(\mathbf{h}, \mathbf{h}^*)$ is our optimization goal, where we want to update the model by fixing the biggest violation. Note that the associated constraint is only violated when $\mathcal{L}(\mathbf{h}, \mathbf{h}^*)$ is positive. To find the path $\mathbf{h}$ in this step that maximizes the violation is equivalent to maximizing $f_\theta(\mathbf{h}) - R(\mathbf{h})$, given that the rest of the terms are constant with respect to $\mathbf{h}$.

When there exist only explicit supervision signals, our objective function reduces to the one for optimizing structured SVM without regularization. For implicit signals, we find $\mathbf{h}^*$ approximately before we optimize the margin loss. In this case, the search is not exact as the reward signals are delayed. Nevertheless, we found the margin loss worked well empirically, as it kept decreasing in general until being stable.

Algorithm 1 summarizes the learning procedure of MMRN. Search is used in both Line 2 and 3. In Line 4, the algorithm performs a gradient update to modify all the model parameters.

Algorithm 1 Maximum Margin Reward Networks

1: **for** a random labeled data (x, y) **do**
2: $\quad \mathbf{h}^* \leftarrow \arg\max_{\mathbf{h} \in \mathcal{E}(x)} R(\mathbf{h}; y) + \epsilon f_\theta(\mathbf{h})$
3: $\quad \hat{\mathbf{h}} \leftarrow \arg\max_{\mathbf{h} \in \mathcal{E}(x)} f_\theta(\mathbf{h}) - R(\mathbf{h}; y)$
4: $\quad$ update θ by minimizing $\mathcal{L}(\hat{\mathbf{h}}, \mathbf{h}^*)$
5: **end for**

4.3 Practical Considerations

Although the learning algorithm of MMRN is simple and general, the quality of the learned model is dictated by the effectiveness of the search procedure. Increasing the beam size generally helps improve the model, but also slows down the training, and has a limited effect when dealing with a large search space. Domain-specific heuristics for pruning search space should thus be used when available. For instance, in the task of semantic parsing, when the reward of a legitimate semantic parse is 0, it implies that none of the derived answers is included in the labeled set of answers. When all the possible follow-up actions can only make the semantic parse stricter (e.g., adding constraints), and result in a subset of the current derived answers, it is clear that the rewards of all these new states are 0 as well. Paths from this state can thus be pruned.

Another strategy for improving search quality is to use *approximated* reward in the early stage of search. Very often the true rewards at this stage are 0, and are not useful to guide the search to find the best path. The approximated reward function can be thought of as estimating whether there exists a high-reward state that is reachable from the current state. The effectiveness of this strategy has been demonstrated successfully by several recent efforts (Mnih et al., 2013; Krishnamurthy et al., 2015; Silver et al., 2016; Narasimhan et al., 2016).

5 Neural Architectures

While the learning algorithm of MMRN described in Sec. 4 is general, the exact model design is task-dependent. In this section, we describe in detail the neural network architectures of the three targeted tasks, *named entity recognition*, *entity linking* and *semantic parsing*.

5.1 Named Entity Recognition

Recall that NER is formulated as a sequence labeling problem, and each action is to label a word with a tag using the BIOES encoding (cf. Sec. 3.1).

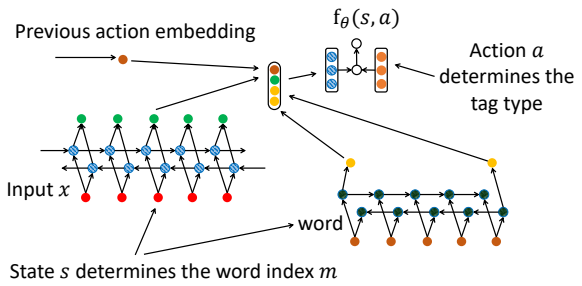

Figure 4: The action scoring model for NER.

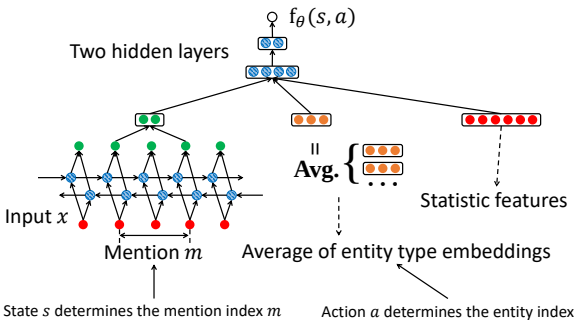

Figure 5: The action scoring model for EL.

The model of the action scoring function $f_\theta(s, a)$ is depicted in Figure 4, which is basically the dot product of the *action embedding* and *state embedding*. The action embedding is initialized randomly for each action, but can be fine-tuned during training (i.e. back-propagate the error through the network and update the word/entity type embeddings). The state embedding is the concatenation of bi-LSTM word embeddings of the current word, the character-based word embeddings, and the embedding of the previous action. We also include the orthographic embeddings proposed by Limsopatham and Collier (2016).

5.2 Entity Linking

An action in entity linking is to determine whether a mention should be linked to a particular entity (cf. Sec. 3.2). As shown in Figure 5, we design the scoring function as a feed-forward neural network that takes as input three different input vectors: (1) surface features from hand-crafted mention-entity statistics that are similar to the ones used in (Yang and Chang, 2015); (2) mention context embeddings from a bidirectional LSTM module; (3) entity embeddings constructed from entity type embeddings. All these embeddings, except the feature vectors, are fine-tuned during training.

Some unique properties of our entity linking model are worth noticing. First, we add mention context embeddings from a bidirectional LSTM module as additional input. While using LSTMs is a common practice for sequence labeling, it is not usually used for short-text entity linking. For each mention, we only extract the output from the bi-LSTM module at the start and end tokens of the mention, and concatenate them as the mention context embeddings. Second, we construct entity embeddings using the average of its Freebase (Bollacker et al., 2008) type embeddings[2], initialized using pre-trained embeddings. Adding these two types of embeddings has shown to improve the performance in our experiments.

[2]We use only the 358 most frequent Freebase entity types.

5.3 Semantic Parsing

Our semantic parsing model follows the STAGG system (Yih et al., 2015), which uses a stage-wise search procedure to expand the candidate semantic parses gradually (cf. Sec. 3.3). Compared to the original system, we make two notable changes. First, we use a two-layer feed-forward neural network to replace the original linear ranker that scores the candidate semantic parses. Second, instead of using a separately trained entity linking system, we incorporate our entity linking networks described in Sec. 5.2 as part of the semantic parsing model. The training process will thus fine tune the entity linking component to improve the semantic parsing system.

6 Experiments

It is important to have a general machine learning model working for both implicit and explicit supervision signals. We valid our learning framework when the explicit supervision signals are presented, as well as demonstrate the support of the scenario where supervision signals are mixed.

Specifically, in this section, we report the experimental results of `MMRN` on named entity recognition and entity linking, both using explicit supervision, and on semantic parsing, using implicit supervision. In all our experiments, we tuned hyper-parameters on the development set (each task respectively), and then re-trained the models on the combination of the training and development set.

6.1 Named entity recognition

We use the CoNLL-2003 shared task data for the NER experiments, where the standard evaluation

System	F_1
Collobert et al. (2011)	89.59
Huang et al. (2015)	90.10
Chiu and Nichols (2015)	90.77
Ratinov and Roth (2009)	90.88
Lample et al. (2016)	90.94
Ma and Hovy (2016)	91.21
MMRN-NER Beam = 5	90.03
MMRN-NER Beam = 20	**91.39**

Table 1: **Explicit Supervision: Named Entity Recognition**. Our MMRN with beam size 20 outperforms current best systems, which are based on neural networks.

	NEEL-Test F_1	TACL F_1
S-MART	77.7	63.6
NTEL	77.9	**68.1**
MMRN-EL	**78.5**	67.5
MMRN-EL - Entity	77.4	66.5
MMRN-EL - LSTM	76.6	66.0

Table 2: **Explicit Supervision: Entity Linking.** Our system trained with MMRN is comparable to the state-of-art NTEL system.

metric is the F_1 score. The pre-trained word embeddings are 100-dimension GloVe vectors trained on 6 billion tokens (Pennington et al., 2014)[3]. The search procedure is conducted using beam search, and the reward function is simply the number of correct tag assignments to the words.

The results are shown in Table 1, compared with recently proposed systems based on neural models. When the beam size is set to 20, MMRN achieves 91.4, which is the best published result so far (without using any gazetteers). Notice that when beam size is 5, the performance drops to 90.03. This demonstrates the importance of search quality when applying MMRN.

6.2 Entity linking

For entity linking, we adopt two publicly available datasets for tweet entity linking: NEEL (Cano et al., 2014)[4] and TACL (Guo et al., 2013; Fang and Chang, 2014; Yang and Chang, 2015; Yang et al., 2016). We follow prior works (Guo et al., 2013; Yang and Chang, 2015) and perform the standard evaluation for an end-to-end entity linking system by computing precision, recall, and F_1 scores, according to the entity references and the system output. An output entity is considered correct if it matches the gold entity and the mention boundary overlaps with the gold mention boundary. Interested readers can refer to (Carmel et al., 2014) for more detail.

We initialize the word embeddings from pre-trained GloVe vectors trained on the twitter corpus, and type embeddings from the pre-trained skip-gram model (Mikolov et al., 2013)[5]. Sizes of both word embeddings are set to 200. Inference is done using a dynamic programming algorithm.

Results of entity linking experiments are presented in Table 2, which are compared with those of S-MART (Yang and Chang, 2015)[6] and NTEL (Yang et al., 2016)[7], two state-of-the-art entity linking systems for short texts. Our MMRN-EL is comparable to the best system. We also conducted two ablation studies by removing the entity type vectors (MMRN-EL - Entity), and by removing the LSTM vectors (MMRN-EL - LSTM). Both show significant performance drops, which validates the importance of these two additional input vectors.

6.3 Semantic parsing

For semantic parsing, we use the dataset WebQSP[8] (Yih et al., 2016) in our experiments. This dataset is a clean and enhanced version of the widely used WebQuestions dataset (Berant et al., 2013), which consists of pairs of questions and answers found in Freebase. Compared to WebQuestions, WebQSP excludes questions with ambiguous intent, and provides verified answers and full semantic parses to the remaining 4,737 questions.

We follow the implicit supervision setting in (Yih et al., 2016), using $3,098$ question–answer pairs for training, and $1,639$ for testing. A subset of 620 pairs from the training set is used for hyper-parameter tuning. Because there can be multiple answers to a question, the quality of a semantic parser is measured using the averaged F_1 score of the predicted answers.

[3] Available at http://nlp.stanford.edu/projects/glove/

[4] NEEL dataset was originally created for an entity linking competition: http://microposts2016.seas.upenn.edu/challenge.html

[5] Available at https://code.google.com/archive/p/word2vec/

[6] The winning system of the NEEL challenge.

[7] To have a fair comparison, we compare to the results of NTEL which do not use pretrained user embedding.

[8] Available at http://aka.ms/WebQSP

We experiment with two configurations of incorporating the entity linking component. MMRN-PIPELINE trains an MMRN-EL model using the entity linking labels in WebQSP separately. Given a question, the entities in it are first predicted, and used as input to the semantic parsing system. In contrast, MMRN-JOINT incorporates the MMRN-EL model in the whole framework. During this joint training process, 15 entity link results are sampled according to the current MMRN-EL model, and passed to the downstream networks. In both cases, we use the previous entity linking model trained on the NEEL dataset to initialize the parameters. As discussed in Sec. 4.1, in this implicit supervision setting, we directly set the (delayed) reward function to be the F_1 score, which can be obtained by comparing the annotated answers with predicted answers.

Table 3 summarizes the results of the MMRN-based semantic parsing systems and other strong baselines. The SP column reports the averaged F_1 scores. Compared to the pipeline approach (MMRN-PIPELINE), the joint learning framework (MMRN-JOINT) improves significantly, reaching 68.1% F_1. To compare different learning methods, we also apply REINFORCE (Williams, 1992), a popular policy gradient algorithm, to train our joint model using the same setting and reward function.[9] MMRN-JOINT outperforms REINFORCE and its variant, REINFORECE+, which re-normalizes the probabilities of the sampled candidate sequences. Its result is also better than the state-of-the-art STAGG system. Note that we use the same architectures and initialization procedures for MMRN-PIPELINE/JOINT and REINFORCE/REINFORCE+. Therefore, the superior performance of MMRN-JOINT shows that the joint learning plays a crucial role in addition to the choices of architecture. Comparing to STAGG, note that Yih et al. (2016) did not jointly train the entity linker and semantic parser together, but they did improve the results by taking the top 10 predictions of their entity linking system for re-ranking parses. Our algorithm further allows to update the entity linker with the labels for semantic parsing and shows superior performance.

Our joint model also improves the entity linking prediction on the questions in WebQSP using the implicit signals (the EL columns in Table 3). The F_1 score of MMRN-JOINT on entity linking is 2.4 points higher than the baseline MMRN-PIPELINE. Note that the entity linking results of MMRN-PIPELINE (line 1) are exactly the results of the entity linking component MMRN-EL. The result is also better than REINFORCE, and comparable to REINFORCE+.

	SP	EL		
	Avg. F_1	P	R	F_1
MMRN-PIPELINE	62.5	85.6	77.5	81.3
MMRN-JOINT	**68.1**	89.3	78.9	**83.7**
REINFORCE	62.9	87.5	76.6	81.7
REINFORCE+	66.7	91.1	76.9	83.4
STAGG	66.8	–	–	–

Table 3: **Implicit Supervision: Semantic Parsing**. By updating the entity linking and semantic parsing models jointly, MMRN-JOINT improves over MMRN-PIPELINE by 5 points in F_1 and outperforms REINFORCE+ (SP). It also improves the entity linking result on the WebQSP questions (EL).

Recently Liang et al. (2016) proposed Neural Symbolic Machine (NSM) and reported the best result of 69.0 F1 score on the WebQSP dataset using the weak supervision settings.[10] The NSM architecture for semantic parsing is significantly different from the architecture used in (Yih et al., 2016) and the one used in this paper. In contrast, MMRN is a general learning framework that allows joint training on existing models (i.e. entity linking and semantic parsing modules). This allows MMRN to use the labels of semantic parsing task as implicit supervision signals for the entity linking module. It would be interesting to apply MMRN on the newly proposed architectures as well.

7 Discussion

We discuss several issues that are highly related to MMRN in this section.

Learning to Search There are two main differences between MMRN and search-based algorithms, such as SEARN (Daumé et al., 2009) and DAGGER (Ross et al., 2011). First, both SEARN and DAGGER focus on imitation learning, assuming explicit supervision signals exist. They use a two-step model learning approach:

[9] The REINFORCE algorithm uses warm initialization—the entity linking parameters are initialized using the model trained on the NEEL dataset.

[10] The paper is published after the submission of this paper.

(1) create cost-sensitive examples by listing state–action pairs and their corresponding (estimated) losses; (2) apply cost-aware training algorithms. In contrast, MMRN directly updates the parameters using back-propagation based on search results of each example. Second, SEARN mixes the optimal and current policies during learning, while MMRN performs search twice and simply pushes the current policy towards the optimal one. Recently, Chang et al. (2015) extend this line of work and discuss different roll-in and roll-out strategies during training for structured contextual bandit settings. As MMRN uses two search procedures, there is no need to mix different search policies.

Reinforcement Learning In many reinforcement learning scenarios, the search space is not fully controllable by the agent. For example, a chess playing agent cannot control the move made by its opponent, and has to commit a single move and wait for the opponent. Note that the agent can still think ahead and build a search tree, but only one move can be made in the end. In contrast, in scenarios like semantic parsing, the whole search space is controlled by the agent itself. Therefore, from the initial state, we can explore several search paths and get their real rewards. This may explain why MMRN can be more efficient than REINFORCE, as MMRN can use the reward signals of multiple paths more effectively. In addition, MMRN is not a probabilistic model, so it does not need to handle normalization issues, which often causes large variance in estimating the gradient direction when optimizing the expected reward.

Semantic Parsing MMRN can be applied for many semantic parsing tasks. One key step is to design the right approximated reward for a given task to guide the beam search to nd the reference parses in MMRN, given that the actual reward is often very sparse. In our companion paper, (Iyyer et al., 2017), we used a simple form of approximated reward to get feedback as early as possible during search. In other words, the semantic parse will be executed as soon as the parse is executable (even if the parse is still not completed) *during search*. The execution results will be used to calculate the Jaccard coefficient with respect to the labeled answers as the approximated rewards. The use of approximated reward has been proven to be effective in (Iyyer et al., 2017).

An important research direction for semantic parsing is to reduce the supervision cost. In (Yih et al., 2016), the authors demonstrated that labeling semantic parses is possible and often more effective with a sophisticated labeling interface. However, collecting answers may still be easier or faster for certain problems or annotators. This suggests that we could allow the annotators to choose to label semantic parses or answers in order to minimize the supervision cost. MMRN would be an ideal learning algorithm for this scenario.

8 Conclusion

This paper proposes Maximum Margin Reward Networks, a structured learning framework that can learn from both explicit and implicit supervision signals. In the future, we plan to apply Maximum Margin Reward Networks on other structured learning tasks. Improving MMRN for dealing with large search space is an important future direction as well.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. The first author is partly sponsored by DARPA under agreement number FA8750-13-2-0008. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *TACL*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *EMNLP*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *ICDM*.

Amparo E Cano, Giuseppe Rizzo, Andrea Varga, Matthew Rowe, Milan Stankovic, and Aba-Sah Dadzie. 2014. Making sense of microposts:(# microposts2014) named entity extraction & linking challenge. In *CEUR Workshop*.

David Carmel, Ming-Wei Chang, Evgeniy Gabrilovich, Bo-June Paul Hsu, and Kuansan Wang. 2014. ERD'14: entity recognition and disambiguation challenge. In *ACM SIGIR Forum*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. In *ICML*.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional LSTM-CNNs. *arXiv preprint arXiv:1511.08308*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *CoNLL*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *SIGDAT*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *ACL*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*.

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*.

Hal Daumé and Daniel Marcu. 2005. Learning as search optimization: approximate large margin methods for structured prediction. In *ICML*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *ACL*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *ACL*.

Yuan Fang and Ming-Wei Chang. 2014. Entity linking on microblogs with spatial and temporal signals. *TACL*.

Yuhang Guo, Bing Qin, Ting Liu, and Sheng Li. 2013. Microblog entity linking by leveraging extra posts. In *EMNLP*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *ACL*.

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. 2009. Cutting-plane training of structural svms. *Machine Learning*.

Akshay Krishnamurthy, CMU EDU, Hal Daumé III, and UMD EDU. 2015. Learning to search better than your teacher. *arXiv preprint arXiv:1502.02206*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*.

Nut Limsopatham and Nigel Collier. 2016. Bidirectional LSTM for named entity recognition in twitter messages. In *WNUT*.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNS-CRF. *arXiv preprint arXiv:1603.01354*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. In *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Ariadna Quattoni, Sybor Wang, Louis-Philippe Morency, Morency Collins, and Trevor Darrell. 2007. Hidden conditional random fields. *PAMI*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*.

Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin markov networks. In *NIPS*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *NAACL*.

Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen-tau Yih. 2016. Learning from explicit and implicit supervision jointly for algebra word problems. In *EMNLP*.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *NIPS*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*.

Yi Yang and Ming-Wei Chang. 2015. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. In *ACL*.

Yi Yang, Ming-Wei Chang, and Jacob Eisenstein. 2016. Toward socially-infused information extraction: Embedding authors, mentions, and entities. In *EMNLP*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL*.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *ACL*.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural SVMs with latent variables. In *ICML*.