SlotGAN: Detecting Mentions in Text via Adversarial Distant Learning

Daza, D.; Cochez, M.; Groth, P.

DOI
10.18653/v1/2022.spnlp-1.4

Publication date
2022

Document Version
Final published version

Published in
Sixth Workshop on Structured Prediction for NLP

License
CC BY

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)
SlotGAN: Detecting Mentions in Text via Adversarial Distant Learning

Daniel Daza¹,²,³, Michael Cochez¹,³, Paul Groth²,³
¹Vrije Universiteit Amsterdam
²University of Amsterdam
³Discovery Lab, Elsevier, The Netherlands
d.dazacruz@vu.nl, m.cochez@vu.nl, p.groth@uva.nl

Abstract

We present SlotGAN, a framework for training a mention detection model that only requires unlabeled text and a gazetteer. It consists of a generator trained to extract spans from an input sentence, and a discriminator trained to determine whether a span comes from the generator, or from the gazetteer. We evaluate the method on English newswire data and compare it against supervised, weakly-supervised, and unsupervised methods. We find that the performance of the method is lower than these baselines, because it tends to generate more and longer spans, and in some cases it relies only on capitalization. In other cases, it generates spans that are valid but differ from the benchmark. When evaluated with metrics based on overlap, we find that SlotGAN performs within 95% of the precision of a supervised method, and 84% of its recall. Our results suggest that the model can generate spans that overlap well, but an additional filtering mechanism is required.

1 Introduction

Detecting mentions of entities in text is an important step towards the extraction of structured information from natural language sources. Mention Detection (MD) components can be found frequently in systems for Named Entity Recognition (NER) (Straková et al., 2019; Wang et al., 2021), entity linking (Wu et al., 2020; Cao et al., 2021), relationship extraction (Katiyar and Cardie, 2017; Zhong and Chen, 2021), and coreference resolution (Joshi et al., 2019; Xu and Choi, 2020; Kirstain et al., 2021), where accurately modeling mentions is crucial for downstream performance.

The MD task is often subsumed under NER, where most effective approaches employ supervised learning with exhaustively annotated datasets. These methods become less feasible in cases where we need to rapidly build MD systems, for example, when moving to a domain with incompatible NER classes; or when there are not enough resources to create a labeled dataset. In contrast, we assume that we have access to an unlabeled corpus, and a list of known entity names (i.e. a gazetteer).

We propose SlotGAN– a framework for detecting mentions that uses a generator to extract spans conditioned on some input text, and a discriminator that determines whether a span comes from the generator, or from the gazetteer (see Fig. 1). In contrast with distant supervision methods that require training with false negatives (Ratner et al., 2016; Giannakopoulos et al., 2017; Shang et al., 2018), SlotGAN relies on the discriminator to learn patterns that are not likely to be names of entities (such as verb phrases, or very long spans, which rarely occur in a gazetteer), thereby improving the generator’s ability to detect valid mentions.

We evaluate the method in a MD task using the CoNLL 2003 English dataset (Tjong Kim Sang and De Meulder, 2003). We observe that the absence of strong supervision in SlotGAN results in different, yet valid notions of what constitutes an entity. For instance, while in the sentence “On the road to Tripoli airport...” the word Tripoli is selected as a gold mention, SlotGAN selects Tripoli airport. In this case, exact match metrics for NER underestimate performance, assigning zero precision and recall. To account for this, we introduce overlap-based metrics into the evaluation.

When using exact boundary match metrics, SlotGAN exhibits lower performance compared to different baselines. When evaluating overlap, precision (how much of the predicted span overlaps with the gold span) is within 95% of the performance of the supervised baseline, while recall (how much of the gold span is actually predicted) is within 84%. We observe that SlotGAN tends to generate more and longer spans than those in the benchmark, and in some cases it relies only on capitalization.

Our contributions are the following: 1) A framework towards distantly-supervised MD that avoids explicit training with false negatives, and an imple-
Figure 1: SlotGAN consists of a generator $G$ trained to extract spans from an input sentence. We represent spans as matrices containing embeddings of words in a span, padded with zeros to a fixed length $L$. True spans are generated from a gazetteer. A discriminator $D$ is trained to determine if a span was generated from $G$ or from the gazetteer.

2 SlotGAN

In the MD task, we are given a sentence from a corpus as a sequence of words $(w_1, w_2, ..., w_n)$. The output of the system is a set of spans that contain a mention, and each span is a tuple $(s_i, i_e)$ where $i_s$ is an integer indicating the position where the span starts, and $i_e$ the position where it ends. Additionally, we have access to a gazetteer $E = (e_1, e_2, ..., e_N)$ containing names of entities relevant to a particular domain.

SlotGAN is a method for MD based on Generative Adversarial Networks (Goodfellow et al., 2014; Mirza and Osindero, 2014). It consists of a generator trained to extract spans from a sentence, and a discriminator that determines whether a span comes from the generator or from the gazetteer.

We define the embedding of a sentence $w = (w_1, ..., w_n)$ as a matrix $\text{emb}(w) \in \mathbb{R}^{d \times n}$, where $\text{emb}$ is a function that maps words to $d$-dimensional pretrained embeddings (for example, from the input embedding layer of BERT (Devlin et al., 2019)).

We represent each mention span in a sequence as a matrix in a space $S = \mathbb{R}^{d \times L}$, where $L$ is the length of the sequence. For a span $(i_s, i_e)$, the matrix contains the embeddings of the words within the span, from column $i_s$ to column $i_e$, and is zero in the remaining columns.

The generator takes as input the embedding matrix $\text{emb}(w)$ of a sentence, and assigns each of its columns to one of $k$ slots. The output is a sequence of $k$ span representations $(S_i)_{i=1}^k$ with $S_i \in S$, such that the $j$-th column of $S_i$ contains the $j$-th column of the input matrix, if it was assigned to slot $i$. Unused columns of $S_i$ are filled with zeros.

When sampling a name $e$ of an entity in the gazetteer, we embed it as $\text{emb}(e)$ and then add zero padding via a pad function until reaching a maximum length $L$, to obtain a span representation in $S$. The amount of padding is added randomly to both sides of an entity name, with the purpose of emulating how in a sentence, a mention can appear at an arbitrary position. The discriminator takes as input span representations in $S$, and outputs a score that should be high for samples from the gazetteer, and low for samples from the generator.

Denoting as $p_w$ the distribution used to sample sentences from the corpus, and as $p_e$ the distribution for sampling names from the gazetteer, the generator and discriminator are trained via gradient descent using the W-GAN (Arjovsky et al., 2017) minimax optimization objective:

$$\min_G \max_D \mathbb{E}_{e \sim p_e} [D(\text{pad}(\text{emb}(e)))] - \mathbb{E}_{w \sim p_w} \left[ \sum_{i=1}^{k} D(G(\text{emb}(w))_i) \right], \quad (1)$$

where we have denoted as $G(\text{emb}(w))_i$ the $i$-th span representation produced by the generator.

To allow also not extracting any mentions when not required, we randomly introduce empty spans in the gazetteer, and we reformulate the generator objective with an equality constraint. Following Bastings et al. (2019), we define the constraint in terms of a differentiable function $C$ such that $C(G(\text{emb}(w))_i)$ counts the number of transitions from zero to non-zero, and vice versa, in a span representation. For valid spans, this should be equal to 2. We solve the problem introducing a Lagrange
multiplier $\lambda$, and the term in Eq. 1 that depends on the generator becomes

$$\min_{\lambda, G} \mathbb{E}_{w \sim p_w} \left[ \sum_{i=1}^{k} -D(S_i(w)) - \lambda(2 - C(S_i(w))) \right]$$

(2)

where $S_i(w)$ is a shorthand for $G(\text{emb}(w))_i$. This constraint prevents the generator from producing only empty spans.

At test time, we can use the spans produced by the generator as predictions for mentions. Alternatively, we can balance precision and recall by leveraging the discriminator, by only keeping spans with a score $D(S_i(w)) > t$ where $t$ is a threshold.

We implement the generator using BERT (Devlin et al., 2019), followed by a modified Slot Attention layer (Locatello et al., 2020) to model discrete selections of distinct spans. The discriminator is a temporal CNN. For more details on the architecture, we refer the reader to Appendix A.

### 3 Related Work

The task of MD has been addressed under NER effectively via supervised learning (Devlin et al., 2019; Straková et al., 2019; Peters et al., 2018; Yu et al., 2020; Wang et al., 2021). Some works address the lack of labeled data in a target domain by applying adaptation techniques from a source domain with labeled data (Zhou et al., 2019; Li et al., 2019; Zhang et al., 2021). In this work we focus on the case where annotations are not available.

Closer to our work are methods for weakly or distantly supervised learning, where heuristics and domain-specific rules are used to generate a noisy training set, often using external sources like gazetteers (Safranchik et al., 2020; Lison et al., 2020; Zhao et al., 2021; Ratner et al., 2016; Shang et al., 2018; Li et al., 2021a). These methods are limited by false negatives that reduce recall in MD. Furthermore, even though rules can be used to annotate a dataset at a large scale, the process of devising these rules in the first place can be tedious, and might require domain expert knowledge.

Luo et al. (2020) recently introduced a fully unsupervised method for NER that uses a pipeline of clustering over word embeddings, a generative model, and reinforcement learning to solve the NER task without any labels or external sources. These elements are optimized separately, whereas SlotGAN provides an end-to-end architecture.
Table 1: Mention detection results evaluated via exact match precision (P), recall (R), and F1 score; and overlap metrics (preceded with O). The “Data” column indicates what is required to train the model in addition to a corpus.

<table>
<thead>
<tr>
<th>Method</th>
<th>Data</th>
<th>Precision (P)</th>
<th>Recall (R)</th>
<th>F1 score</th>
<th>Overlap Precision (OP)</th>
<th>Overlap Recall (OR)</th>
<th>Overlap F1 (OF1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>String matching Gazetteer</td>
<td>Gazetteer</td>
<td>76.2</td>
<td>54.0</td>
<td>63.2</td>
<td>57.4</td>
<td>61.3</td>
<td>58.6</td>
</tr>
<tr>
<td>ACE (Wang et al., 2021) Gold labels</td>
<td>96.0</td>
<td>97.1</td>
<td>96.5</td>
<td>98.3</td>
<td>98.1</td>
<td>98.1</td>
<td>98.1</td>
</tr>
<tr>
<td>AutoNER (Shang et al., 2018) Type dictionary</td>
<td>88.4</td>
<td>94.2</td>
<td>91.2</td>
<td>97.4</td>
<td>97.2</td>
<td>96.9</td>
<td></td>
</tr>
<tr>
<td>Unsupervised (Luo et al., 2020) Domain concepts</td>
<td>80.0</td>
<td>72.0</td>
<td>76.0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>SlotGAN - no pretraining Gazetteer</td>
<td>55.9</td>
<td>66.1</td>
<td>60.6</td>
<td>82.9</td>
<td>79.5</td>
<td>82.9</td>
<td></td>
</tr>
<tr>
<td>SlotGAN - pretrained Gazetteer</td>
<td>60.1</td>
<td>71.1</td>
<td>65.2</td>
<td>93.2</td>
<td>83.0</td>
<td>84.7</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Comparison of gold spans and spans predicted by SlotGAN.

Results We present MD results in Table 1. We observe that pretraining with Wikipedia and Wikidata entity names helps to improve the performance over a version trained with the CoNLL 2003 data only. The higher recall of SlotGAN in comparison with the string matching baseline shows that the generator is not simply memorizing the gazetteer and can thus detect mentions not seen during training. However, its precision and recall are low compared to other systems. We attribute this partly to the lack of strong supervision of the generator, which results in boundaries that differ from gold spans, and detection of more mentions than those present in the dataset. The overlap-based metrics show that on average, predicted spans overlap 93% and gold spans overlap 83% with the intersection. This indicates that extra words are added to predicted spans, and boundary mismatch, though these values of precision and recall are within 95% and 84% of the supervised baseline, respectively.

A closer analysis of the length of overlapping spans shows that in 69.4% of the cases the length is the same as gold spans, in 21.1% the predicted span is longer, and in 9.5% it is shorter. This often leads to mentions that are actually correct, as shown in Table 2. However, SlotGAN also produces spans that do not overlap with any gold span. This can be observed by plotting the average number of words assigned to a mention by the model versus the gold annotations, as shown in Fig 2. We see that across different numbers of mention words for the gold annotations, SlotGAN produces a higher number in average. We also find cases where it relies on capitalization only, which becomes problematic in upper case sentences: for regular sentences, there is no exact boundary match in 11% of the cases. For sentences in upper case, this increases to 23%.

5 Conclusion We have presented SlotGAN, a method for training a mention detector that only requires unlabeled text and a list of entity names, that relies on implicit supervision provided by a discriminator that is also optimized during training. This results in spans that overlap well with gold spans, but also a tendency towards generating more and longer spans, and relying on capitalization only. This suggests that spans predicted by SlotGAN are likely to be correct,
but an additional mechanism is needed to filter them. This can be enforced via tighter constraints on generated spans, or a stronger discriminator.

Even though its performance is close to a supervised model according to overlap-based metrics, it cannot match other methods that also use a gazetteer or are unsupervised. In spite of this, we consider SlotGAN a promising framework for other IE tasks with less supervision, for example, where relations between slots could be induced. The end-to-end architecture also presents an opportunity for fine-tuning with gold labels, which we plan to explore in future work.

Acknowledgments
This project was funded by Elsevier’s Discovery Lab.

References


### A Architectures

In our implementation of SlotGAN, the embedding function $\text{emb}(w)$ used to obtain embeddings of sentences and names in the gazetteer uses the pretrained WordPiece embeddings from the input layer of BERT (Devlin et al., 2019).

The generator consists of BERT, which for an input sentence of length $n$, outputs a matrix $\mathbf{H} \in \mathbb{R}^{d \times n}$ where $d$ is the dimension of the output layer of BERT, equal to 768. We use the bert-base-cased implementation in HuggingFace’s Transformer library (Wolf et al., 2019).

The output matrix is passed to a modified Slot Attention layer (Locatello et al., 2020), which we use as a differentiable mechanism to assign $n$ input embeddings to $k$ slots. In the original implementation, Slot Attention would assign each of the $n$ outputs to the columns of $\mathbf{H}$ to $k$ slots, by using a differentiable clustering algorithm. This algorithm works for a variable number of slots, by sampling $k$ initial slot representations from a Gaussian distribution. In our experiments we use $k = 10$, and the number of iterations of the clustering algorithm is set to 3.

In the MD case, for words that do not belong to any mention, we want the generator to be able to assign them to a “default” slot. We achieve this by introducing an extra slot, whose representation, instead of sampled, is a single vector with a learned representation. Slot Attention in the generator thus contains $k+1$ slots, but the default slot is discarded when passing generated spans to the discriminator.

After discarding the default slot, the result is an attention mask $\mathbf{M} \in \mathbb{R}^{k \times n}$ where the $m_{ij}$ entry indicates the fraction of input $j$ assigned to slot $i$, and each column is normalized to 1. The $i$-th span representation is then obtained as

$$S_i = \text{emb}(w) \odot M_{i:},$$

where $M_{i:}$ is the $i$-th row of $\mathbf{M}$, and $\odot$ is broadcast element-wise multiplication.

For the discriminator we use a temporal CNN, where convolutions are applied along the sequence axis. The input is a matrix of span representations of shape $d \times L$, and the output is a scalar. The architecture is described in Table 3.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Output features</th>
<th>Activation</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 $\times$ 3 Conv</td>
<td>128</td>
<td>ReLU</td>
</tr>
<tr>
<td>3 $\times$ 3 Conv</td>
<td>64</td>
<td>ReLU</td>
</tr>
<tr>
<td>3 $\times$ 3 Conv</td>
<td>64</td>
<td>ReLU</td>
</tr>
<tr>
<td>3 $\times$ 3 Conv</td>
<td>64</td>
<td>—</td>
</tr>
<tr>
<td>Flatten</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Linear</td>
<td>32</td>
<td>ReLU</td>
</tr>
<tr>
<td>Linear</td>
<td>1</td>
<td>—</td>
</tr>
</tbody>
</table>

Table 3: Architecture of the discriminator used in our experiments.
B Training Procedure

We train SlotGAN with mini-batches of 32 sentences. We update the generator once for every 5 updates of the discriminator. To let the discriminator accept empty spans as valid, we replace names from the gazetteer with an empty span with a probability of 0.5. We use a gradient penalty coefficient (Gulrajani et al., 2017) of 10 when computing the discriminator loss.

We use a learning rate of $2 \times 10^{-5}$, with a linear warm-up schedule for the first 10% of epochs. For the Lagrange multiplier, we use the Modified Differential Method of Multipliers (Platt and Barr, 1987) with a constant learning rate of $1 \times 10^{-3}$.

We run our experiments in a workstation with an Intel Xeon processor, 1 NVIDIA GeForce GTX 1080 Ti GPU with 11GB of memory, and 60GB of RAM. When pretraining with Wikipedia and Wikidata, we train SlotGAN with 20,000 updates of the generator, and 5,000 when training with the CoNLL 2003 dataset.