Improving a Neural Semantic Parser by Counterfactual Learning from Human Bandit Feedback

Carolin Lawrence
Computational Linguistics
Heidelberg University
69120 Heidelberg, Germany
Lawrence@cl.uni-heidelberg.de

Stefan Riezler
Computational Linguistics & IWR
Heidelberg University
69120 Heidelberg, Germany
riezler@cl.uni-heidelberg.de

Abstract

Counterfactual learning from human bandit feedback describes a scenario where user feedback on the quality of outputs of a historic system is logged and used to improve a target system. We show how to apply this learning framework to neural semantic parsing. From a machine learning perspective, the key challenge lies in a proper reweighting of the estimator so as to avoid known degeneracies in counterfactual learning, while still being applicable to stochastic gradient optimization. To conduct experiments with human users, we devise an easy-to-use interface to collect human feedback on semantic parses. Our work is the first to show that semantic parsers can be improved significantly by counterfactual learning from logged human feedback data.

1 Introduction

In semantic parsing, natural language utterances are mapped to machine readable parses which are complex and often tailored specifically to the underlying task. The cost and difficulty of manually preparing large amounts of such parses thus is a bottleneck for supervised learning in semantic parsing. Recent work (Liang et al. (2017); Mou et al. (2017); Peng et al. (2017); *inter alia*) has applied reinforcement learning to address the annotation bottleneck as follows: Given a question, the existence of a corresponding gold answer is assumed. A semantic parser produces multiple parses per question and corresponding answers are obtained. These answers are then compared against the gold answer and a positive reward is recorded if there is an overlap. The parser is then guided towards correct parses using the REINFORCE algorithm (Williams, 1992) which scales the gradient for the various parses by their obtained reward (see the left half of Figure 1). However, learning from question-answer pairs is only efficient if gold answers are cheap to obtain. For complex open-domain question-answering tasks, correct answers are not unique factoids, but open-ended lists, counts in large ranges, or fuzzily defined objects. For example, geographical queries against databases such as OpenStreetMap (OSM) can involve fuzzy operators such as "near" or "in walking distance" and thus need to allow for fuzziness in the answers as well. A possible solution lies in machine learning from even weaker supervision signals in form of human bandit feedback[1] where the semantic parsing system suggests exactly one parse for which feedback is collected from a human user. In this setup neither gold parse nor gold answer are known and feedback is obtained for only one system output per question.

The goal of our paper is to exploit this scenario of learning from human bandit feedback to train semantic parsers. This learning scenario perfectly fits commercial setups such as virtual personal assistants that embed a semantic parser. Commercial systems can easily log large amounts of interaction data between users and system. Once sufficient data has been collected, the log can then be used to improve the parser. This leads to a counterfactual learning scenario (Bottou et al., 2013) where we have to solve the counterfactual problem of how to improve a target system from logged feedback that was given to the outputs of a different historic system (see the right half of Figure 1).

In order to achieve our goal of counterfactual learning of semantic parsers from human bandit feedback, the following contributions are required:

[1]The term "bandit feedback" is inspired by the scenario of maximizing the reward for a sequence of pulls of arms of "one-armed bandit" slot machines.

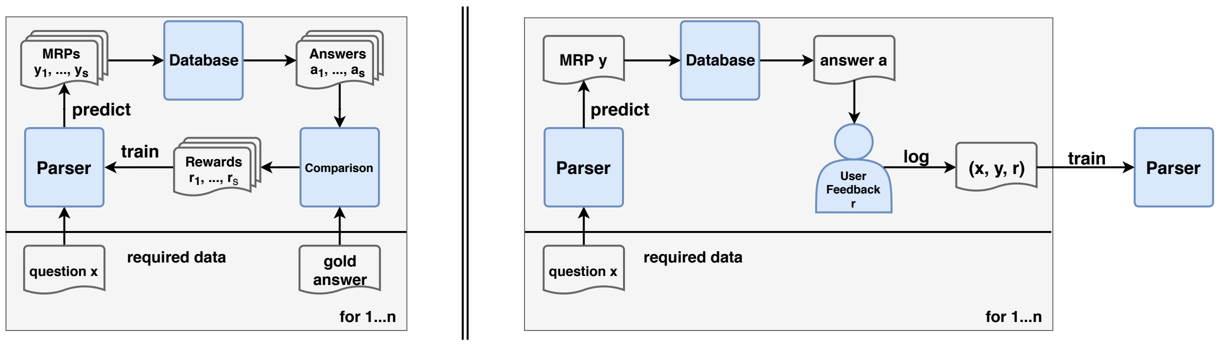


Figure 1: Left: Online reinforcement learning setup for semantic parsing setup where both questions and gold answers are available. The parser attempts to find correct machine readable parses (MRPs) by producing multiple parses, obtaining corresponding answers, and comparing them against the gold answer. Right: In our setup, a question does not have an associated gold answer. The parser outputs a single MRP and the corresponding answer is shown to a user who provides some feedback. Such triplets are collected in a log which can be used for offline training of a semantic parser. This scenario is called counterfactual since the feedback was logged for outputs from a system different from the target system to be optimized.

First, we need to construct an easy-to-use user interface that allows to collect feedback based on the parse rather than the answer. To this aim, we automatically convert the parse to a set of statements that can be judged as correct or incorrect by a human. This approach allows us to assign rewards at the token level, which in turn enables us to perform blame assignment in bandit learning and to learn from partially correct queries where tokens are reinforced individually. We show that users can provide such feedback for one question-parse pair in 16.4 seconds on average. This exemplifies that our approach is more efficient and cheaper than recruiting experts to annotate parses or asking workers to compile large answer sets.

Next, we demonstrate experimentally that counterfactual learning can be applied to neural sequence-to-sequence learning for semantic parsing. A baseline neural semantic parser is trained in fully supervised fashion, human bandit feedback from human users is collected in a log and subsequently used to improve the parser. The resulting parser significantly outperforms the baseline model as well as a simple bandit-to-supervised approach (B2S) where the subset of completely correct parses is treated as a supervised dataset. Finally, we repeat our experiments on a larger but simulated log to show that our gains can scale: the baseline system is improved by 7.45% in answer F1 score without ever seeing a gold standard parse.

Lastly, from a machine learning perspective, we have to solve problems of degenerate behavior in counterfactual learning by lifting the multiplicative control variate technique (Swaminathan and Joachims, 2015b; Lawrence et al., 2017b,a) to stochastic learning for neural models. This is done by reweighting target model probabilities over the logged data under a one-step-late model that decouples the normalization from gradient estimation and is thus applicable in stochastic (minibatch) gradient optimization.

2 Related Work

Semantic parsers have been successfully trained using neural sequence-to-sequence models with a cross-entropy objective and question-parse pairs (Jia and Liang, 2016; Dong and Lapata, 2016)) or question-answer pairs (Neelakantan et al., 2017). Improving semantic parsers using weak feedback has previously been studied (Goldwasser and Roth (2013); Artzi and Zettlemoyer (2013); *inter alia*). More recently, several works have applied policy gradient techniques such as REINFORCE (Williams, 1992) to train neural semantic parsers (Liang et al. (2017); Mou et al. (2017); Peng et al. (2017); *inter alia*). However, they assume the existence of the true target answers that can be used to obtain a reward for any number of output queries suggested by the system. It thus differs from a bandit setup where we assume that a reward is available for only one structure.

Our work most closely resembles the work of

Iyer et al. (2017) who do make the assumption of only being able to judge one output. They improve their parser using simulated and real user feedback. Parses with negative feedback are given to experts to obtain the correct parse. Corrected queries and queries with positive feedback are added to the training corpus and learning continues with a cross-entropy objective. We show that this bandit-to-supervision approach can be outperformed by offline bandit learning from partially correct queries. Yih et al. (2016) proposed a user interface for the Freebase database that enables a fast and easy creation of parses. However, in their setup the worker still requires expert knowledge about the Freebase database, whereas in our approach feedback can be collected freely and from any user interacting with the system.

From a machine learning perspective, related work can be found in the areas of counterfactual bandit learning (Dudik et al., 2011; Swaminathan and Joachims, 2015a), or equivalently, off-policy reinforcement learning (Precup et al., 2000; Jiang and Li, 2016). Our contribution is to modify the self-normalizing estimator (Kong, 1992; Precup et al., 2000; Swaminathan and Joachims, 2015b; Joachims et al., 2018) to be applicable to neural networks. Our work is similar to the counterfactual learning setup for machine translation introduced by Lawrence et al. (2017b). Following their insight, we also assume the logs were created deterministically, i.e. the logging policy always outputs the most likely sequence. Their framework was applied to statistical machine translation using linear models. We show how to generalize their framework to neural models and how to apply it to the task of neural semantic parsing in the OSM domain.

3 Neural Semantic Parsing

Our semantic parsing model is a state-of-the-art sequence-to-sequence neural network using an encoder-decoder setup (Cho et al., 2014; Sutskever et al., 2014) together with an attention mechanism (Bahdanau et al., 2015). We use the settings of Sennrich et al. (2017), where an input sequence $x = x_1, x_2, \dots x_{|x|}$ (a natural language question) is encoded by a Recurrent Neural Network (RNN), each input token has an associated hidden vector $h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i]$ where the former is created by a forward pass over the input, and the latter by a backward pass. $\overrightarrow{h}_i$ is obtained by recursively computing $f(x_i, \overrightarrow{h}_{i-1})$ where f is a Gated Recurrent Unit (GRU) (Chung et al., 2014), and $\overleftarrow{h}_i$ is computed analogously. The input sequence is reduced to a single vector $c = g(\{h_1, \dots, h_{|x|}\})$ which serves as the initialization of the decoder RNN. g calculates the average over all vectors h_i. At each time step t the decoder state is set by $s_t = q(s_{t-1}, y_{t-1}, c_t)$. q is a conditional GRU with an attention mechanism and c_t is the context vector computed by the attention mechanism. Given an output vocabulary $\mathcal{V}_y$ and the decoder state $s_t = \{s_1, \dots, s_{|\mathcal{V}_y|}\}$, a softmax output layer defines a probability distribution over $\mathcal{V}_y$ and the probability for a token y_j is:

$$\pi_w(y_j = t_o | y_{<j}, x) = \frac{\exp(s_{t_o})}{\sum_{v=1}^{|\mathcal{V}_y|} \exp(s_{t_v})}. \quad (1)$$

The model π_w can be seen as parameterized policy over an action space defined by the target language vocabulary. The probability for a full output sequence $y = y_1, y_2, \dots y_{|y|}$ is defined by

$$\pi_w(y|x) = \prod_{j=1}^{|y|} \pi_w(y_j | y_{<j}, x). \quad (2)$$

In our case, output sequences are linearized machine readable parses, called queries in the following. Given supervised data $\mathcal{D}_{sup} = \{(x_t, \bar{y}_t)\}_{t=1}^{n}$ of question-query pairs, where $\bar{y}_t$ is the true target query for x_t, the neural network can be trained using SGD and a cross-entropy (CE) objective:

$$\mathcal{L}_{CE} = -\frac{1}{n} \sum_{t=1}^{n} \sum_{j=1}^{|\bar{y}|} \log \pi_w(\bar{y}_{t,j} | \bar{y}_{t,<j}, x_t), \quad (3)$$

where $y_{t,<j} = y_{t,1}, y_{t,2} \dots y_{t,j-1}$.

4 Counterfactual Learning from Deterministic Bandit Logs

Counterfactual Learning Objectives. We assume a policy π_w that, given an input $x \in \mathcal{X}$, defines a conditional probability distribution over possible outputs $y \in \mathcal{Y}(x)$. Furthermore, we assume that the policy is parameterized by w and its gradient can be derived. In this work, π_w is defined by the sequence-to-sequence model described in Section 3. We also assume that the model decomposes over individual output tokens,

$\nabla_w \hat{R}_{\text{DPM}} = \frac{1}{n}\sum_{t=1}^{n} \delta_t \pi_w(y_t\|x_t) \nabla_w \log \pi_w(y_t\|x_t)$.
$\nabla_w \hat{R}_{\text{DPM+R}} = \frac{1}{n}\sum_{t=1}^{n}[\delta_t \bar{\pi}_w(y_t\|x_t)(\nabla_w \log \pi_w(y_t\|x_t) - \frac{1}{n}\sum_{u=1}^{n} \bar{\pi}_w(y_u\|x_u) \nabla \log \pi_w(y_u\|x_u))]$.
$\nabla_w \hat{R}_{\text{DPM+OSL}} = \frac{1}{m}\sum_{t=1}^{m} \delta_t \bar{\pi}_{w,w'}(y_t\|x_t) \nabla_w \log \pi_w(y_t\|x_t)$.
$\nabla_w \hat{R}_{\text{DPM+T}} = \frac{1}{n}\sum_{t=1}^{n} \left(\sum_{j=1}^{\|y_t\|} \delta_{t,j} \nabla_w \log \pi_w(y_{t,j}\|y_{t,<j}, x_t)\right)$.
$\nabla_w \hat{R}_{\text{DPM+T+OSL}} = \frac{\frac{1}{m}\sum_{t=1}^{m}\left(\sum_{j=1}^{\|y_t\|} \delta_{t,j} \nabla_w \log \pi_w(y_{t,j}\|y_{t,<j}, x_t)\right)}{\frac{1}{n}\sum_{t=1}^{n} \pi_{w'}(y_t\|x_t)}$.

Table 1: Gradients of counterfactual objectives.

i.e. that the model produces the output token by token.

The counterfactual learning problem can be described as follows: We are given a data log of triples $\mathcal{D}_{log} = \{(x_t, y_t, \delta_t)\}_{t=1}^{n}$ where outputs y_t for inputs x_t were generated by a logging system under policy π_0, and loss values $\delta_t \in [-1, 0]$[2] were observed for the generated data points. Our goal is to optimize the expected reward (in our case: minimize the expected risk) for a target policy π_w given the data log $\mathcal{D}_{log}$. In case of deterministic logging, outputs are logged with propensity $\pi_0(y_t|x_t) = 1,\ t = 1, \ldots, n$. This results in a *deterministic propensity matching (DPM)* objective (Lawrence et al., 2017b), without the possibility to correct the sampling bias of the logging policy by inverse propensity scoring (Rosenbaum and Rubin, 1983):

$$\hat{R}_{\text{DPM}}(\pi_w) = \frac{1}{n}\sum_{t=1}^{n} \delta_t \pi_w(y_t|x_t). \tag{4}$$

This objective can show degenerate behavior in that it overfits to the choices of the logging policy (Swaminathan and Joachims, 2015b; Lawrence et al., 2017a). This degenerate behavior can be avoided by reweighting using a multiplicative control variate (Kong, 1992; Precup et al., 2000; Jiang and Li, 2016; Thomas and Brunskill, 2016). The new objective is called the *reweighted deterministic propensity matching (DPM+R)* objective in Lawrence et al. (2017b):

$$\begin{aligned}\hat{R}_{\text{DPM+R}}(\pi_w) &= \frac{1}{n}\sum_{t=1}^{n} \delta_t \bar{\pi}_w(y_t|x_t) \qquad (5)\\ &= \frac{\frac{1}{n}\sum_{t=1}^{n} \delta_t \pi_w(y_t|x_t)}{\frac{1}{n}\sum_{t=1}^{n} \pi_w(y_t|x_t)}.\end{aligned}$$

[2]We use the terms loss and (negative) rewards interchangeably, depending on context.

Algorithms for optimizing the discussed objectives can be derived as gradient descent algorithms where gradients using the score function gradient estimator (Fu, 2006) are shown in Table 1.

Reweighting in Stochastic Learning. As shown in Swaminathan and Joachims (2015b) and Lawrence et al. (2017a), reweighting over the entire data log $\mathcal{D}_{log}$ is crucial since it avoids that high loss outputs in the log take away probability mass from low loss outputs. This multiplicative control variate has the additional effect of reducing the variance of the estimator, at the cost of introducing a bias of order $O(\frac{1}{n})$ that decreases as n increases (Kong, 1992). The desirable properties of this control variate cannot be realized in a stochastic (minibatch) learning setup since minibatch sizes large enough to retain the desirable reweighting properties are infeasible for large neural networks.

We offer a simple solution to this problem that nonetheless retains all desired properties of the reweighting. The idea is inspired by one-step-late algorithms that have been introduced for EM algorithms (Green, 1990). In the EM case, dependencies in objectives are decoupled by evaluating certain terms under parameter settings from previous iterations (thus: one-step-late) in order to achieve closed-form solutions. In our case, we decouple the reweighting from the parameterization of the objective by evaluating the reweighting under parameters w' from some previous iteration. This allows us to perform gradient descent updates and reweighting asynchronously. Updates are performed using minibatches, however, reweighting is based on the entire log, allowing us to retain the desirable properties of the control variate.

The new objective, called *one-step-late reweighted DPM* objective (DPM+OSL), optimizes $\pi_{w,w'}$ with respect to w for a minibatch of size m, with reweighting over the entire log of size n under parameters w':

$$\hat{R}_{\text{DPM+OSL}}(\pi_w) = \frac{1}{m}\sum_{t=1}^{m} \delta_t \bar{\pi}_{w,w'}(y_t|x_t) \tag{6}$$
$$= \frac{\frac{1}{m}\sum_{t=1}^{m} \delta_t \pi_w(y_t|x_t)}{\frac{1}{n}\sum_{t=1}^{n} \pi_{w'}(y_t|x_t)}.$$

If the renormalization is updated periodically, e.g. after every validation step, renormalizations under w or w' are not much different and will not hamper convergence. Despite losing the formal justification from the perspective of control variates, we found empirically that the OSL update schedule for reweighting is sufficient and does not deteriorate performance. The gradient for learning with OSL updates is given in Table 1.

Token-Level Rewards. For our application of counterfactual learning to human bandit feedback, we found another deviation from standard counterfactual learning to be helpful: For humans, it is hard to assign a graded reward to a query at a sequence level because either the query is correct or it is not. In particular, with a sequence level reward of 0 for incorrect queries, we do not know which part of the query is wrong and which parts might be correct. Assigning rewards at token-level will ease the feedback task and allow the semantic parser to learn from partially correct queries. Thus, assuming the underlying policy can decompose over tokens, a token level (DPM+T) reward objective can be defined:

$$\hat{R}_{\text{DPM+T}}(\pi_w) = \tag{7}$$
$$\frac{1}{n}\sum_{t=1}^{n}\left(\sum_{j=1}^{|y|} \delta_{t,j} \log \pi_w(y_{t,j}|y_{t,<j}, x_t)\right).$$

Analogously, we can define an objective that combines the token-level rewards and the minibatched reweighting (DPM+T+OSL):

$$\hat{R}_{\text{DPM+T+OSL}}(\pi_w) = \tag{8}$$
$$\frac{\frac{1}{m}\sum_{t=1}^{m}\left(\sum_{j=1}^{|y|} \delta_{t,j} \log \pi_w(y_{t,j}|y_{t,<j}, x_t)\right)}{\frac{1}{n}\sum_{t=1}^{n} \pi_{w'}(y_t|x_t)}.$$

Gradients for the DPM+T and DPM+T+OSL objectives are given in Table 1.

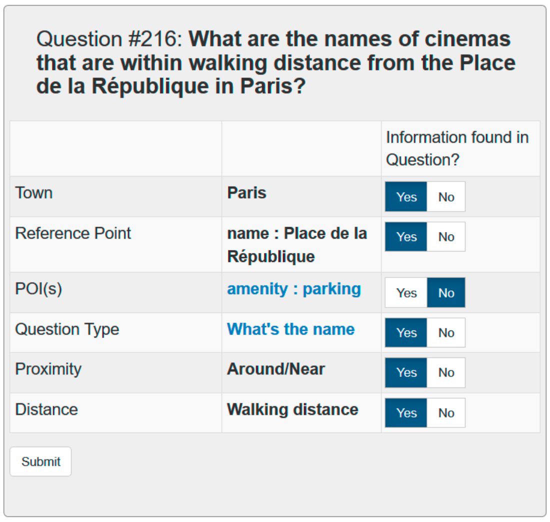
5 Semantic Parsing in the OpenStreetMap Domain

OpenStreetMap (OSM) is a geographical database in which volunteers annotate points of interests in the world. A point of interest consists of one or more associated GPS points. Further relevant information may be added at the discretion of the volunteer in the form of tags. Each tag consists of a key and an associated value, for example "*tourism : hotel*". The NLMAPS corpus was introduced by Haas and Riezler (2016) as a basis to create a natural language interface to the OSM database. It pairs English questions with machine readable parses, i.e. queries that can be executed against OSM.

Human Feedback Collection. The task of creating a natural language interface for OSM demonstrates typical difficulties that make it expensive to collect supervised data. The machine readable language of the queries is based on the OVERPASS query language which was specifically designed for the OSM database. It is thus not easily possible to find experts that could provide correct queries. It is equally difficult to ask workers at crowdsourcing platforms for the correct answer. For many questions, the answer set is too large to expect a worker to count or list them all in a reasonable amount of time and without errors. For example, for the question "*How many hotels are there in Paris?*" there are 951 hotels annotated in the OSM database. Instead we propose to automatically transform the query into a block of statements that can easily be judged as correct or incorrect by a human. The question and the created block of statements are embedded in a user interface with a form that can be filled out by users. Each statement is accompanied by a set of radio buttons where a user can select either "*Yes*" or "*No*". For a screenshot of the interface and an example see Figure 2.

In total there are 8 different types of statements. The presence of certain tokens in a query trigger different statement types. For example, the token "*area*" triggers the statement type "*Town*". The statement is then populated with the corresponding information from the query. In the case of "*area*", the following OSM value is used, e.g. "*Paris*". With this, the meaning of every query can be captured by a set of human-understandable statements. For a full overview of all statement types and their triggers see section B of the supplementary material.

OSM tags and keys are generally understandable. For example, the correct OSM tag for "*hotels*" is "*tourism : hotel*" and when searching for


Question #216: **What are the names of cinemas that are within walking distance from the Place de la République in Paris?**

		Information found in Question?
Town	**Paris**	[✓ Yes] No
Reference Point	**name : Place de la République**	[✓ Yes] No
POI(s)	**amenity : parking**	Yes [✓ No]
Question Type	**What's the name**	[✓ Yes] No
Proximity	**Around/Near**	[✓ Yes] No
Distance	**Walking distance**	[✓ Yes] No

Submit

Figure 2: The user interface for collecting feedback from humans with an example question and a correctly filled out form.

websites, the correct question type key would be "*website*". Nevertheless, for each OSM tag or key, we automatically search for the corresponding Wikipedia page on the OpenStreetMap Wiki[3] and extract the description for this tag or key. The description is made available to the user in form of a tool-tip that appears when hovering over the tag or key with the mouse. If a user is unsure if a OSM tag or key is correct, they can read this description to help in their decision making. Once the form is submitted, a script maps each statement back to the corresponding tokens in the original query. These tokens then receive negative or positive feedback based on the feedback the user provided for that statement.

Corpus Extension. Similar to the extension of the NLMAPS corpus by Lawrence and Riezler (2016) who include shortened questions which are more typically used by humans in search tasks, we present an automatic extension that allows a larger coverage of common OSM tags.[4] The basis for the extension is a hand-written, online freely available list[5] that links natural language expressions such as "*cash machine*" to appropriate OSM tags, in this case "*amenity : atm*". Using the list, we generate for each unique expression-tag pair a set of question-query pairs. These latter pairs contain

[3] https://wiki.openstreetmap.org/

[4] The extended dataset, called NLMAPS V2, will be released upon acceptance of the paper.

[5] http://wiki.openstreetmap.org/wiki/Nominatim/Special_Phrases/EN

	NLMAPS	NLMAPS V2
# question-query pairs	2,380	28,609
tokens	25,906	202,088
types	1,002	8,710
avg. sent. length	10.88	7.06
distinct tags	477	6,582

Table 2: Corpus statistics of the question-answering corpora NLMAPS and our extension NLMAPS V2 which additionally contains the search engine style queries (Lawrence and Riezler, 2016) and the automatic extensions of the most common OSM tags.

several placeholders which will be filled automatically in a second step.

To fill the area placeholder `$LOC`, we sample from a list of 30 cities from France, Germany and the UK. `$POI` is the placeholder for a point of interest. We sample it from the list of objects which are located in the prior sampled city and which have a *name* key. The corresponding value belonging to the *name* key will be used to fill this spot. The placeholder `$QTYPE` is filled by uniformly sampling from the four primary question types available in the NLMAPS query language. On the natural language side they corresponded to "*How many*", "*Where*", "*Is there*" and `$KEY`. `$KEY` is a further parameter belonging to the primary question operator FINDKEY. It can be filled by any OSM key, such as *name*, *website* or *height*. To ensure that there will be an answer for the generated query, we first ran a query with the current tag ("*amenity : atm*") to find all objects fulfilling this requirement in the area of the already sampled city. From the list of returned objects and the keys that appear in association with them, we uniformly sampled a key. For `$DIST` we chose between the pre-defined options for walking distance and within city distance. The expressions map to corresponding values which define the size of a radius in which objects of interest (with tag "*amenity : atm*") will be located. If the walking distance was selected, we added "*in walking distance*" to the question. Otherwise no extra text was added to the question, assuming the within city distance to be the default. This sampling process was repeated twice.

Table 2 presents the corpus statistics, comparing NLMAPS to our extension. The automatic extension, obviating the need for expensive manual work, allows a vast increase of question-query pairs by an order of magnitude. Consequently the number of tokens and types increase in a similar vein. However, the average sentence length drops. This comes as no surprise due to the nature of the rather simple hand-written list which contains never more than one tag for an element, resulting in simpler question structures. However, the main idea of utilizing this list is to extend the coverage to previously unknown OSM tags. With 6,582 distinct tags compared to the previous 477, this was clearly successful. Together with the still complex sentences from the original corpus, a semantic parser is now able to learn both complex questions and a large variety of tags. An experiment that empirically validates the usefulness of the automatically created data can be found in the supplementary material, section A.

6 Experiments

General Settings. In our experiments we use the sequence-to-sequence neural network package NEMATUS (Sennrich et al., 2017). Following the method used by Haas and Riezler (2016), we split the queries into individual tokens by taking a pre-order traversal of the original tree-like structure. For example, "*query(west(area(keyval('name','Paris')), nwr(keyval('railway','station'))),qtype(count))*" becomes "*query@2 west@2 area@1 keyval@2 name@0 Paris@s nwr@1 keyval@2 railway@0 station@s qtype@1 count@0*".

The SGD optimizer used is ADADELTA (Zeiler, 2012). The model employs 1,024 hidden units and word embeddings of size 1,000. The maximum sentence length is 200 and gradients are clipped if they exceed a value of 1.0. The stopping point is determined by validation on the development set and selecting the point at which the highest evaluation score is obtained. F1 validation is run after every 100 updates, and each update is made on the basis of a minibatch of size 80.

The evaluation of all models is based on the answers obtained by executing the most likely query obtained after a beam search with a beam of size 12. We report the F1 score which is the harmonic mean of precision and recall. Recall is defined as the percentage of fully correct answers divided by the set size. Precision is the percentage of correct answers out of the set of answers with non-empty strings. Statistical significance between models is measured using an approximate randomization test (Noreen, 1989).

Baseline Parser & Log Creation. Our experiment design assumes a baseline neural semantic parser that is trained in fully supervised fashion, and is to be improved by bandit feedback obtained for system outputs from the baseline system for given questions. For this purpose, we select 2,000 question-query pairs randomly from the full extended NLMAPS V2 corpus. We will call this dataset $\mathcal{D}_{sup}$. Using this dataset, a baseline semantic parser is trained in supervised fashion under a cross-entropy objective. It obtains an F1 score of 57.45% and serves as the logging policy π_0.

Furthermore we randomly split off 1,843 and 2,000 pairs for a development and test set, respectively. This leaves a set of 22,765 question-query pairs. The questions can be used as input and bandit feedback can be collected for the most likely output of the semantic parser. We refer to this dataset as $\mathcal{D}_{log}$.

To collect human feedback, we take the first 1,000 questions from $\mathcal{D}_{log}$ and use π_0 to parse these questions to obtain one output query for each. 5 question-query pairs are discarded because the suggested query is invalid. For the remaining question-query pairs, the queries are each transformed into a block of human-understandable statements and embedded into the user interface described in Section 5. We recruited 9 users to provide feedback for these question-query pairs. The resulting log is referred to as $\mathcal{D}_{human}$. Every question-query pair is purposely evaluated only once to mimic a realistic real-world scenario where user logs are collected as users use the system. In this scenario, it is also not possible to explicitly obtain several evaluations for the same question-query pair. Some examples of the received feedback can be found in the supplementary material, section C.

To verify that the feedback collection is efficient, we measured the time each user took from loading a form to submitting it. To provide feedback for one question-query pair, users took 16.4 seconds on average with a standard deviation of 33.2 seconds. The vast majority (728 instances) are completed in less than 10 seconds.

Learning from Human Bandit Feedback. An analysis of $\mathcal{D}_{human}$ shows that for 531 queries all corresponding statements were marked as correct. We consider a simple baseline that treats completely correct logged data as a supervised data set with which training continues using the cross-entropy objective. We call this baseline bandit-to-supervised conversion (B2S). Furthermore, we present experimental results using the log $\mathcal{D}_{human}$ for stochastic (minibatch) gradient descent optimization of the counterfactual objectives introduced in equations 4, 6, 7 and 8. For the token-level feedback, we map the evaluated statements back to the corresponding tokens in the original query and assign these tokens a feedback of 0 if the corresponding statement was marked as wrong and 1 otherwise. In the case of sequence-level feedback, the query receives a feedback of 1 if all statements are marked correct, 0 otherwise. For the OSL objectives, a separate experiment (see below) showed that updating the reweighting constant after every validation step promises the best trade-off between performance and speed.

Results, averaged over 3 runs, are reported in Table 3. The B2S model can slightly improve upon the baseline but not significantly. DPM improves further, significantly beating the baseline. Using the multiplicative control variate modified for SGD by OSL updates does not seem to help in this setup. By moving to token-level rewards, it is possible to learn from partially correct queries. These partially correct queries provide valuable information that is not present in the subset of correct answers employed by the previous models. Optimizing DPM+T leads to a slight improvement and combined with the multiplicative control variate, DPM+T+OSL yields an improvement of about 1.0 in F1 score upon the baseline. It beats both the baseline and the B2S model by a significant margin.

		F1	Δ F1
1	baseline	57.45	
2	B2S	57.79±0.18	+0.34
3	DPM^{1}	58.04±0.04	+0.59
4	DPM+OSL	58.01±0.23	+0.56
5	DPM+T^{1}	58.11±0.24	+0.66
6	DPM+T+OSL^{1,2}	58.44±0.09	+0.99

Table 3: Human Feedback: Answer F1 scores on the test set for the various setups, averaged over 3 runs. Statistical significance of system differences at $p < 0.05$ are indicated by experiment number in superscript.

		F1	Δ F1
1	baseline	57.45	
2	B2S^{1,3}	63.22±0.27	+5.77
3	DPM^{1}	61.80±0.16	+4.35
4	DPM+OSL^{1,3}	62.91±0.05	+5.46
5	DPM+T^{1,2,3,4}	63.85±0.2	+6.40
6	DPM+T+OSL^{1,2,3,4}	64.41±0.05	+6.96

Table 4: Simulated Feedback: Answer F1 scores on the test set for the various setups, averaged over 3 runs. Statistical significance of system differences at $p < 0.05$ are indicated by experiment number in superscript.

Learning from Large-Scale Simulated Feedback. We want to investigate whether the results scale if a larger log is used. Thus, we use π_0 to parse all 22,765 questions from $\mathcal{D}_{log}$ and obtain for each an output query. For sequence level rewards, we assign feedback of 1 for a query if it is identical to the true target query, 0 otherwise. We also simulate token-level rewards by iterating over the indices of the output and assigning a feedback of 1 if the same token appears at the current index for the true target query, 0 otherwise.

An analysis of $\mathcal{D}_{log}$ shows that 46.27% of the queries have a sequence level reward of 1 and are thus completely correct. This subset is used to train a bandit-to-supervised (B2S) model using the cross-entropy objective.

Experimental results for the various optimization setups, averaged over 3 runs, are reported in Table 4. We see that the B2S model outperforms the baseline model by a large margin, yielding an increase in F1 score by 6.24 points. Optimizing the DPM objective also yields a significant increase over the baseline, but its performance falls short of the stronger B2S baseline. Optimizing the DPM+OSL objective leads to a substantial improvement in F1 score over optimizing DPM but still falls slightly short of the strong B2S baseline. Token-level rewards are again crucial to beat the B2S baseline significantly. DPM+T is already able to significantly outperform B2S in this setup and DPM+T+OSL can improve upon this further.

Analysis. Comparing the baseline and DPM+T+OSL, we manually examined all queries in the test set where DPM+T+OSL ob-

Error Type	Human	Simulated
OSM Tag	90%	86.75%
Question Type	6%	8.43%
Structure	4%	4.82%

Table 5: Analysis of which type of errors DPM+T+OSL corrected on the test set compared to the baseline system for both human and simulated feedback experiments.

tained the correct answer and the baseline system did not (see Table 5). The analysis showed that the vast majority of previously wrong queries were fixed by correcting an OSM tag in the query. For example, for the question "*closest Florist from Manchester in walking distance*" the baseline system chose the tag "*landuse : retail*" in the query, whereas DPM+T+OSL learnt that the correct tag is "*shop : florist*". In some cases, the question type had to be corrected, e.g. the baseline's suggested query returned the location of a point of interest but DPM+T+OSL correctly returns the phone number. Finally, in a few cases DPM+T+OSL corrected the structure for a query, e.g. by searching for a point of interest in the east of an area rather than the south.

OSL Update Variation. Using the DPM+T+OSL objective and the simulated feedback setup, we vary the frequency of updating the reweighting constant. Results are reported in Table 6. Calculating the constant only once at the beginning leads to a near identical result in F1 score as not using OSL. The more frequent update strategies, once or four times per epoch, are more effective. Both strategies reduce variance further and lead to higher F1 scores. Updating four times per epoch compared to once per epoch, leads to a nominally higher performance in F1. It has the additional benefit that the re-calculation is done at the same time as the validation, leading to no additional slow down as executing the queries for the development set against the database takes longer than the re-calculation of the constant. Updating after every minibatch is infeasible as it slows down training too much. Compared to the previous setup, iterating over one epoch takes approximately an additional 5.5 hours.

	OSL Update	F1	Δ F1
1	no OSL (DPM+T)	63.85 ± 0.2	
2	once	63.82 ± 0.1	-0.03
3	every epoch	64.26 ± 0.04	+0.41
4	every validation / 4x per epoch	64.41 ± 0.05	+0.56
5	every minibatch	N/A	N/A

Table 6: Simulated Feedback: Answer F1 scores on the test set for DPM+T and DPM+T+OSL with varying OSL update strategies, averaged over 3 runs. Updating after every minibatch is infeasible as it significantly slows down learning. Statistical significance of system differences at $p < 0.05$ occur for experiment 4 over experiment 2.

7 Conclusion

We introduced a scenario for improving a neural semantic parser from logged bandit feedback. This scenario is important to avoid complex and costly data annotation for supervise learning, and it is realistic in commercial applications where weak feedback can be collected easily in large amounts from users. We presented robust counterfactual learning objectives that allow to perform stochastic gradient optimization which is crucial in working with neural networks. Furthermore, we showed that it is essential to obtain reward signals at the token-level in order to learn from partially correct queries. We presented experimental results using feedback collected from humans and a larger scale setup with simulated feedback. In both cases we show that a strong baseline using a bandit-to-supervised conversion can be significantly outperformed by a combination of a one-step-late reweighting and token-level rewards. Finally, our approach to collecting feedback can also be transferred to other domains. For example, (Yih et al., 2016) designed a user interface to help Freebase experts to efficiently create queries. This interface could be reversed: given a question and a query produced by a parser, the interface is filled out automatically and the user has to verify if the information fits.

Acknowledgments

The research reported in this paper was supported in part by DFG grant RI-2221/4-1.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1).

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, San Diego, CA.

Léon Bottou, Jonas Peters, Joaquin Qui nonero Candela, Denis X. Charles, D. Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. 2013. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *ArXiv e-prints*, 1412.3555.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany.

Miroslav Dudik, John Langford, and Lihong Li. 2011. Doubly robust policy evaluation and learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, New York, NY.

Michael C. Fu. 2006. Gradient estimation. *Handbook in Operations Research and Management Science*, 13.

Dan Goldwasser and Dan Roth. 2013. Learning from natural instructions. *Machine Learning*, 94(2).

Peter J. Green. 1990. On the use of the EM algorithm for penalized likelihood estimation. *Journal of the Royal Statistical Society B*, 52(3).

Carolin Haas and Stefan Riezler. 2016. A corpus and semantic parser for multilingual natural language querying of openstreetmap. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, San Diego, California.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany.

Nan Jiang and Lihong Li. 2016. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, New York, New York, USA.

Thorsten Joachims, Adith Swaminathan, and Maarten de Rijke. 2018. Deep learning with logged bandit feedback. In *International Conference on Learning Representations (ICLR)*.

Augustine Kong. 1992. A note on importance sampling using standardized weights. Technical Report 348, Department of Statistics, University of Chicago, Illinois.

Carolin Lawrence, Pratik Gajane, and Stefan Riezler. 2017a. Counterfactual learning for machine translation: Degeneracies and solutions. In *Proceedings of the NIPS WhatIF Workshop*, Long Beach, CA.

Carolin Lawrence and Stefan Riezler. 2016. Nlmaps: A natural language interface to query openstreetmap. In *Proceedings of the 26th International Conference on Computational Linguistics: System Demonstrations (COLING)*, Osaka, Japan.

Carolin Lawrence, Artem Sokolov, and Stefan Riezler. 2017b. Counterfactual learning from bandit feedback under deterministic logging : A case study in statistical machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Copenhagen, Denmark.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada.

Lili Mou, Zhengdong Lu, Hang Li, and Zhi Jin. 2017. Coupling distributed and symbolic execution for natural language queries. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, Sydney, Australia.

Arvind Neelakantan, Quoc V. Le, Martín Abadi, Andrew McCallum, and Dario Amodei. 2017. Learning a natural language interface with neural programmer. In *International Conference on Learning Representations (ICLR)*, Toulon, France.

Eric W. Noreen. 1989. *Computer Intensive Methods for Testing Hypotheses: An Introduction*. Wiley, New York.

Haoruo Peng, Ming-Wei Chang, and Wen-tau Yih. 2017. Maximum margin reward networks for learning from explicit and implicit supervision. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Copenhagen, Denmark.

Doina Precup, Richard S. Sutton, and Satinder P. Singh. 2000. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, San Francisco, CA, USA.

Paul R. Rosenbaum and Donald B. Rubin. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1).

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Valencia, Spain.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, Montreal, Canada.

Adith Swaminathan and Thorsten Joachims. 2015a. Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research*, 16.

Adith Swaminathan and Thorsten Joachims. 2015b. The self-normalized estimator for counterfactual learning. In *Advances in Neural Information Processing Systems (NIPS)*, Montreal, Canada.

Philip Thomas and Emma Brunskill. 2016. Data-efficient off-policy policy evaluation for reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning (ICML)*, New York, NY.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 20.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. *ArXiv e-prints*, 1212.5701.

Appendix

8 Empirical Validation of the NLMAPS Corpus Extension

To empirically validate the usefulness of the automatically created data, we compare two parsers trained with NEMATUS (Sennrich et al., 2017) (see Section 6 for more details). The first model is trained using the original NLMAPS training data. The second receives an additional 15,000 instances from the synthetic data. Both systems are tested on the original NLMAPS test data and on the new test set of NLMAPS V2 which consists of a random set of 2,000 pairs from the remaining data. Results may be found in Table **??**. On the original test set, adding the 15,000 synthetic instances allows the parser to significantly improve by 2.09 in F1 score. The parser trained on the original training data performs badly on the new test set because it is ignorant of many OSM tags that were introduced with the extension.

		Train		
		V1	V1+15K	Δ
Test	V1	73.56±0.61	75.65±0.34	+2.09
	V2	28.31±0.25	79.17±0.11	+50.86

Table 7: Answer F1 scores on the NLMAPS V1 and NLMAPS V2 test sets for models trained on either only NLMAPS V1 training data or with an additional 15k synthetic instances. Results are averaged over 3 runs. Using the NLMAPS V2 training set leads to significant system differences on both test sets at $p < 0.05$.

9 Automatic Feedback Form Creation

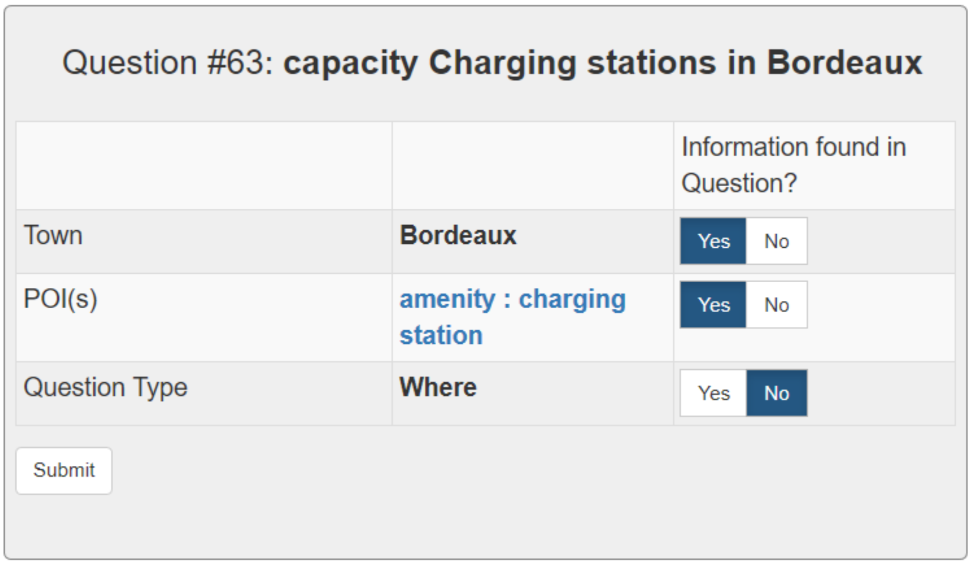
A query can be automatically converted into a set of statements which can be judged as correct or incorrect by non-expert users. There are 8 different statement types and each is triggered based on the shape of the query and certain tokens. An overview of the statement types, their triggers and the value a statement will hold, can be found in Table **??**.

10 Screenshots of Human User Feedback.

Figures **??**, **??**, **??**, **??** and **??** present screenshots of forms as filled out by the recruited human users.

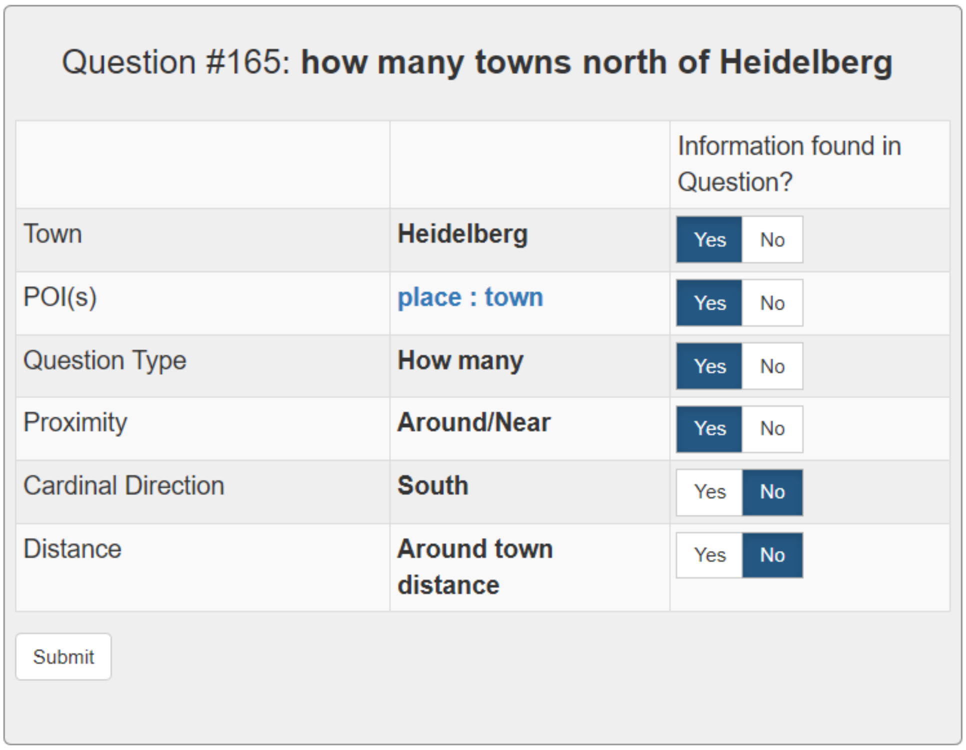
Type	Explanation
Town	OSM tags of "*area*"
Reference Point	OSM tags "*center*"
POI(s)	OSM tags of "*search*" if "*center*" is set, else of "*nwr*"
Question Type	Arguments of "*qtype*"
Proximity : Around/Near	If "*around*" is present
Restriction : Closest	If "*around*" and "*topx*" are present
Distance	Argument of "*maxdist*"
Cardinal Direction	"*north*", "*east*", "*south*" or "*west*" are present

Table 8: Overview of the possible statements types that are used to transform a parse into a human-understandable block of statements.


Question #63: **capacity Charging stations in Bordeaux**

		Information found in Question?
Town	**Bordeaux**	[✓ Yes] No
POI(s)	**amenity : charging station**	[✓ Yes] No
Question Type	**Where**	Yes [✓ No]

Figure 3: Feedback form for question #63 as filled out by a human user.

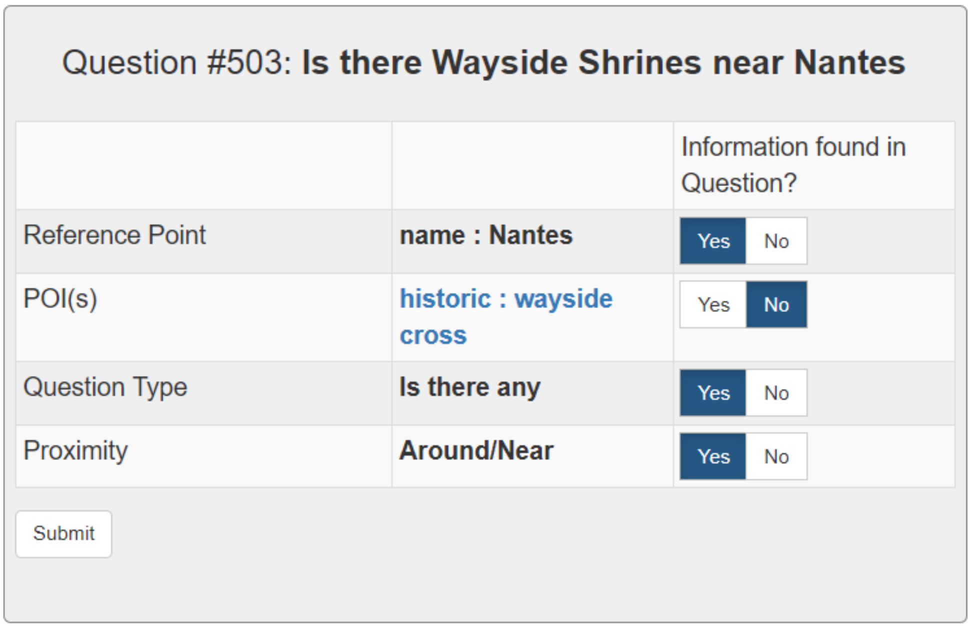


Question #165: **how many towns north of Heidelberg**

		Information found in Question?
Town	**Heidelberg**	[✓ Yes] No
POI(s)	**place : town**	[✓ Yes] No
Question Type	**How many**	[✓ Yes] No
Proximity	**Around/Near**	[✓ Yes] No
Cardinal Direction	**South**	Yes [✓ No]
Distance	**Around town distance**	Yes [✓ No]

Submit

Figure 4: Feedback form for question #165 as filled out by a human user.

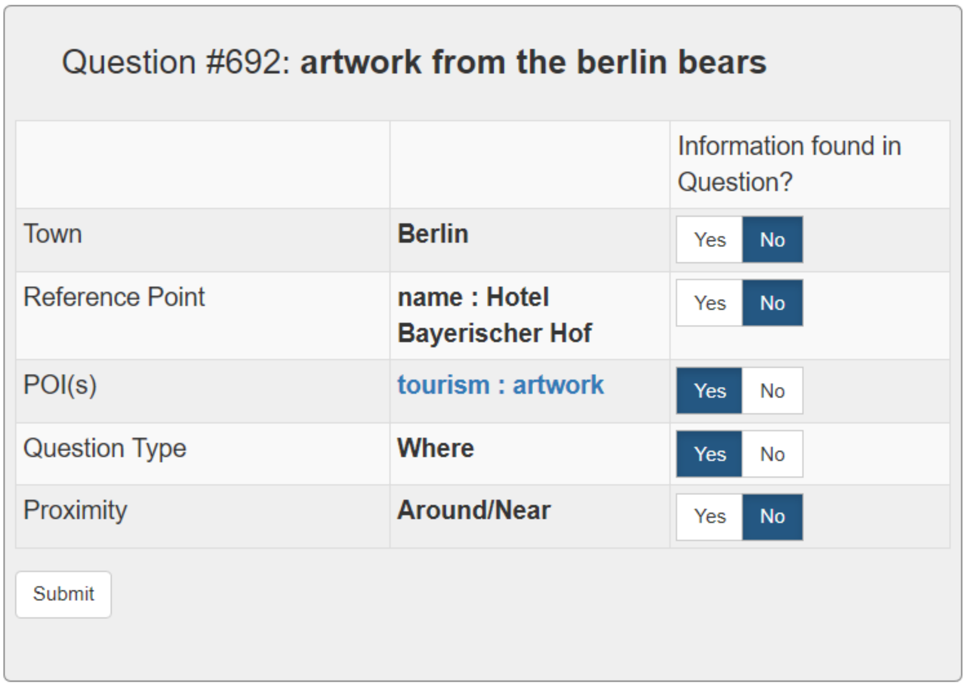


Question #503: **Is there Wayside Shrines near Nantes**

		Information found in Question?
Reference Point	**name : Nantes**	[✓ Yes] No
POI(s)	**historic : wayside cross**	Yes [✓ No]
Question Type	**Is there any**	[✓ Yes] No
Proximity	**Around/Near**	[✓ Yes] No

Submit

Figure 5: Feedback form for question #503 as filled out by a human user.

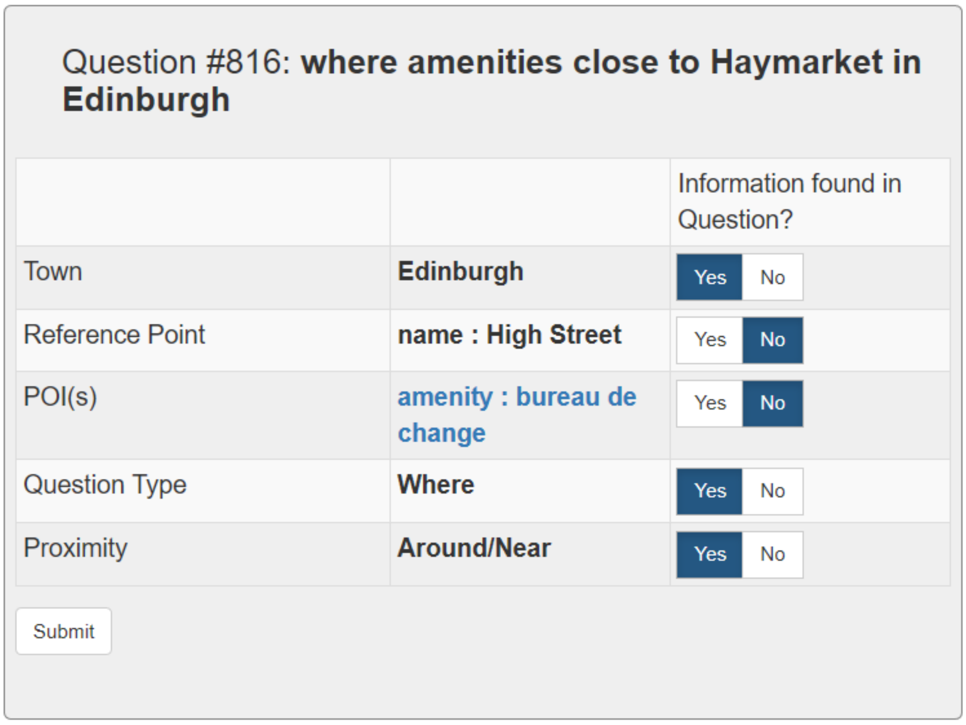


Question #692: **artwork from the berlin bears**

		Information found in Question?
Town	**Berlin**	Yes [✓ No]
Reference Point	**name : Hotel Bayerischer Hof**	Yes [✓ No]
POI(s)	**tourism : artwork**	[✓ Yes] No
Question Type	**Where**	[✓ Yes] No
Proximity	**Around/Near**	Yes [✓ No]

Submit

Figure 6: Feedback form for question #692 as filled out by a human user.



Question #816: **where amenities close to Haymarket in Edinburgh**

		Information found in Question?
Town	**Edinburgh**	[✓ Yes] No
Reference Point	**name : High Street**	Yes [✓ No]
POI(s)	**amenity : bureau de change**	Yes [✓ No]
Question Type	**Where**	[✓ Yes] No
Proximity	**Around/Near**	[✓ Yes] No

Submit

Figure 7: Feedback form for question #816 as filled out by a human user.