Learning an Optimal Sequence of Questions for the Disambiguation of Queries over Structured Data

Achim Rettinger♯, Alexander Hagemann♯, Matthias Nickles♮
♯ Karlsruhe Institute of Technology (KIT), Germany, {rettinger,alexander.hagemann}@kit.edu
♮ INSIGHT / DERI & Department of Information Technology,
National University of Ireland, Galway, matthias.nickles@deri.org

Abstract

Intelligent systems interacting with users often need to relate ambiguous natural language phrases to formal entities which can be further processed. This work strives for learning an optimal sequence of disambiguation questions asked by an agent in order to achieve a perfect interactive disambiguation, setting itself off against previous work on interactive and adaptive dialogue systems for disambiguation in question answering. To this aim, we built a hybrid system that exhibits deductive and statistical inference capabilities by combining techniques from natural language processing, information retrieval, answer set programming and relational reinforcement learning.

Keywords: *Question Answering, Relational Reinforcement Learning, Information Retrieval, Linked Data, Named-Entity Recognition, Disambiguation*

Introduction and Related Work

Human-machine interaction using natural language is typically ambiguous, e.g., because of differences in the perception of the common environment, ambiguous terms, or noise. In human conversations, disambiguation is achieved using contextual information, background knowledge and ultimately by asking questions. Typically, such disambiguation questions are highly efficient, in the sense that only the most ambiguous elements are explicitly discussed and that elements which help to disambiguate other unclear elements are asked first. Thus, humans intuitively learn to optimize for the optimal order and minimal amount of inquiries needed to understand their conversational partner.

A common task of intelligent software systems is to link natural language phrases to a structured knowledge representation which facilitates automated processing. One example is Question Answering (QA) over structured data (in particular Linked Data / RDF), where concepts expressed in natural language need to be related to formal entities (entity recognition), in order to translate the query into a formal (non-ambiguous) SPARQL query. Consider the question *When was Elvis Presley born?*. For a human it is clear that this is about the artist named *Elvis Presley*. However, a keyword search system might associate this with the music album named *Natural Born Elvis*.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The approach proposed in this paper is inspired by the way disambiguation is resolved in human communication. We aim for a system that learns (i) what the most probable disambiguation is, (ii) to what extend interactive disambiguation is needed, and (iii) what the optimal sequence of interactions is to achieve a correct disambiguation dialogue with the minimal number of inquiries by the system. While there is substantial previous work on dialogue systems for disambiguation in QA, this work contrasts with existing approaches by learning an optimal policy that achieves a perfect disambiguation while minimizing the number of interaction steps needed. Our approach is hybrid in that it combines reinforcement learning (RL) with logical reasoning. We implement and evaluate our approach, *DDLearn* (Disambiguation Dialogue Learner), with regard to its disambiguation performance on the *Question Answering over Linked Data* (QALD) challenge. Our results show that our system can learn the optimal order of disambiguation questions, making it the first system capable of this. In contrast to existing approaches which apply *active learning*, our system does not depend on any heuristics.

Related approaches to QA over structured data can be assigned into those approaches that rely on user interactions to remove ambiguity in the question and those that do not. Obviously, we are only interested in the former, and also only in adaptive ones. AutoSPARQL (Lehmann and Bühmann 2011) and FREyA (Damljanovic, Agatonovic, and Cunningham 2012) utilize the answers returned after asking a question for learning. FREyA uses an *active learning* component to improve the heuristic ranking of possible answers during the disambiguation dialogue. Similarly, AutoSPARQL relies on active learning to learn the replies to previously asked questions. Note that active learning - in contrast to reinforcement learning - is not guaranteed to learn an optimal disambiguation sequence, since it optimizes the immediate gain and does not aim at maximizing future rewards. Table 1 summarizes related approaches to question answering over structured data.

An example for an early statistical approach to dialogue learning in a Markov decision process (MDP) setting is (Levin and Pieraccini 1997). Recent results and an extensive survey regarding the use of (Hierarchical) RL for spoken dialogue optimization are provided in (Cuayahuitl 2009), mainly addressing core Natural Language Processing

	SWIP	Querix	Power AQUA	Auto SPARQL	FREyA	DDLearn
QALD	✓	✓	✓	✓	✓	✓
Interaction				✓	✓	✓
Learning				✓	✓	✓
RRL						✓

Table 1: Features of existing QALD approaches

(NLP) problems such as uncertainty in speech recognition. (Ghavamzadeh and Mahadevan 2004) proposes an interesting approach to Hierarchical RL of communication, with a focus on the optimization of communication needs for coordination in terms of communication cost. Like our work, (Ponsen et al. 2009) uses relational reinforcement learning (RRL) in a communication scenario, but aims at learning with whom to communicate. A more closely related approach is pursued in (Rieser and Lemon 2011) and applied to QA for document retrieval. In contrast to our approach the order of inquiries to disambiguate the open "slots" is predefined by a domain specific "process model". A recent approach that is closely related in terms of methodology is (Lison 2013) since it combines RL with probabilistic rules for dialogue management. Transitions are represented as a Bayesian model which can be specified using rules. In contrast, our approach allows deductive reasoning about relational states and actions. None of the aforementioned approaches uses logical reasoning, and none is applied to QA over structured data.

Learning Adaptive Disambiguation Dialogues

Our system *DDLearn* (Disambiguation Dialogue Learner) can learn an optimal disambiguation strategy by engaging the user in an efficient dialogue. To achieve this goal, it exhibits deductive and statistical inference capabilities, and combines and extends techniques from natural language processing, information retrieval, formal logic (in form of answer set programming and circumscriptive event calculus) and, most importantly, relational reinforcement learning (Dzeroski, Raedt, and Driessens 2001; van Otterlo 2005). However, for the current evaluation of our framework (next section), we didn't employ human test users but used deterministic rule-based simulations of users.
Our approach is free of heuristics (in contrast to related QA approaches) and thus can be easily extended to solve additional tasks like disambiguating the question type or misunderstandings.

The workflow of the system is shown in Fig. 1. Besides the composition of the system, our contributions are in the *Adaptive Disambiguation Dialogue* component, which is outlined below. All other components are treated as black boxes. Limitations introduced by those components are not targeted by this work.

First, SWIP[1] is used to convert a natural language query about linked data, labeled *iQuestion*[2], formulated by a human user, into a set of keywords. They are passed on to a graph based keyword search to return a set of possible query graphs that link the provided keywords. This is followed by the formal disambiguation dialogue wherein the agent inquires a human user with so called *dQuestions*[2] and learns a policy to do so in an optimal way. During training, a simulated user answers questions based on a set of correct SPARQL queries provided by the QALD benchmark[3]. In the final step, the chosen query graph is converted into a SPARQL query and executed on the linked data source. Examples for iQuestions in the evaluation domain are "Since when does Fugazi exist?" or "Give me all songs by Petula Clark".

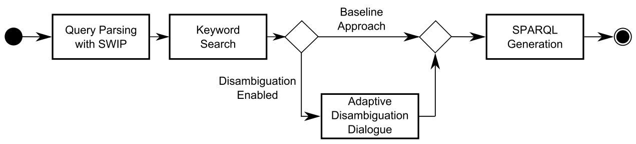

Figure 1: Complete workflow of our DDLearn system.

As input for the Adaptive Disambiguation Dialogue learning phase, our system is provided solely with a set of query segments (such as "Elvis Presley") and possible matching elements (which are in our case formal elements in the RDF schema of a knowledge base, such as "artist"). dQuestions asked by the system are simple yes/no questions about whether a certain query segment (*qs*) matches a certain formal element and about whether the user is fine with the current disambiguation of the query. E.g., in iQuestion "List all members of The Notwist", a query segment would be "Notwist" and the matching formal schema element would be "artist". Any information about the learning environment and also any interaction between system and user (including dQuestions) are formally modeled using a logic approach (answer set programming (ASP) and event calculus (EC)), which allows for the deductive determination of possible states and actions in the relational reinforcement learning task, and, most important, for a *relational* representation of states and actions in reinforcement learning, facilitating relational approaches to state and action generalization and thus a more efficient form of learning (van Otterlo 2005; Driessens 2004).

The actual learning task is performed by an interactive relational reinforcement learning (IRRL) framework which combines the aforementioned formal logic approach (ASP and EC) with reinforcement learning. The IRRL tool QASP ((Nickles 2012)) is configured with the aforementioned scenario and interactively learns an optimal policy of dQuestions which the agent can ask the human interaction partner. The core learning algorithm is a relational and interactive form of Q-learning. The IRRL-approach we use has already shown its potential in toy problems (Nickles and Rettinger 2013). However, this is the first time it is adapted to a question answering scenario and evaluated on a real-world data set of reasonable size.

An ASP solver (a reasoner) is run by QASP to determine the next available states and actions in the Markov

[1]`http://swip.univ-tlse2.fr/SwipWebClient/`

[2]To avoid confusion concerning the term *question*, we distinguish *iQuestion* (the user query in natural language) from *dQuestion* (disambiguation question asked by the agent).

Question	User answer	Reward
isType(qs, formal element)	*yes*	0
	no	0
	don't know	0
ishappy	*happy*	500
	not happy	-10

Table 2: Rewards for answers received from simulated user

DDLearn	AVG Steps in Optimal Policy	Optimal Policies Learned	Episodes Until Optimal Policy
heuristic-free	4.5 ± 1.3	100%	25 ± 17
heuristic-free series	4.3 ± 0.7	76%	27 ± 32

Table 3: Percentage of optimal policies learned by DDLearn

decision process. At this, states consist of logical fluents formulated in the EC. Further, the agent observes the current environment (object domain states and user interaction states) at each time step and interactively receives feedback in terms of a numerical reward. Please find details in (Nickles and Rettinger 2013) and (Nickles 2012). Message forms and rewards are as shown in Table 2. To generalize over similar relational (state, action) pairs accumulated during Q-learning, a k-nearest-neighbor (kNN) approach using a relational instance-based regression function (Driessens 2004) is employed for the prediction of Q-values. One way to use an interactive relational reinforcement learning approach would be to treat every iQuestion individually, motivated by the fact that it is difficult to re-use prior knowledge from past experiments. To avoid this, QASP is able to use additional prior knowledge by loading the results (learned state/action values) from the respective previous experiment (a form of *transfer learning*) and making use of it by means of generalization over similar states and actions.

The output of the learning step is a disambiguated user query from which a SPARQL query is generated ((Tran et al. 2009)).

Evaluation and Conclusions

We implement and evaluate DDLearn with regards to its disambiguation performance on the Question Answering over Linked Data challenge (QALD)[3], which uses MusicBrainz[4] as a knowledge base. MusicBrainz describes 595,765 music artists, 878,630 release and 1,571,575 tracks with a total of almost 15 million statements. Query segments are mapped (disambiguated) to classes of MusicBrainz (e.g., artist, album, track) or to a relation or a property. From the QALD-1 data set 14 iQuestions were filtered for which the keyword search did not return any results and thus no disambiguation was possible. Out of the 100 natural language iQuestions (i.e., user queries in natural language), 86 remained for evaluation.

Our experiments are evaluated in terms of learning success, i.e. does the agent learn to ask an optimal series of dQuestions. This is measured by the number of interactions needed to fully disambiguate the iQuestion. In our experiments, 100 learning episodes are repeated ten times (trials) and the results are averaged.

We first report results concerning the question whether *DDLearn* is able to learn the *optimal* disambiguation strategy for single, independent questions (each disambiguated from scratch), before investigating its behavior regarding learning to disambiguate series of questions:

Optimal Disambiguation of Single Queries We compare our approach to several other approaches. Under the assumption that all query segments can be disambiguated, the optimum (minimal number of interactions) is equal to the number of query segments which should be disambiguated, since this is the minimum number of affirmations needed from the user if the matching is guessed correctly every single time (labeled *#Segments*). Counting this way indicates how complex each iQuestions is to disambiguate. Similarly, the theoretical maximum, called *Worst Case*, shows the maximum number of dQuestions until disambiguation is complete. In this case every potential disambiguation for every query segment is requested and the correct one is always picked last. We also report the performance when dQuestions are randomly selected, called *Random*. Note that the *#Segments*, worst and random approaches do not use any learning component and are heuristic-free, since they don't make use of ranking information. What is called *Heuristic* in Fig. 2 refers to a heuristic ranking of potential mappings. From all query segments and every potential combination of mappings, a set of query graphs is built which is then ranked based on string similarity and length of paths in each graph. A top-k threshold algorithm is used to terminate graph exploration if the algorithm cannot find higher-scored graphs (see (Tran et al. 2009) for details).

Our DDLearn approach was able to learn an optimal policy for all of the iQuestions and required on average 25 episodes to find the optimal policy. The optimal policy found by DDLearn has an average of 4.5 steps. Fig. 2 summarizes the results by showing the number of steps averaged over all individual iQuestions together with the variance. Please note that DDLearn performs better than *#Segments* because it learns which qss cannot be disambiguated (if the correct mapping is not provided by the keyword matching component for some subset of the query segments) and thus does not ask for this qss. In contrast, *#Segments* asks for every qs exactly once. Also note that *Heuristic* shows better performance than *Random* but with a relatively large variance.

Disambiguating Series of Queries In a preliminary experiment, we investigate how our approach compares to *active learning* (see Sec. 1). A greedy selection strategy selects the next dQuestion to be asked according to a heuristic (see above). Since active learners do also "learn" from experience by remembering all correct mappings for each

[3]`http://greententacle.techfak.uni-bielefeld.de/ cunger/qald/index.php?q=1`

[4]`http://musicbrainz.org/`

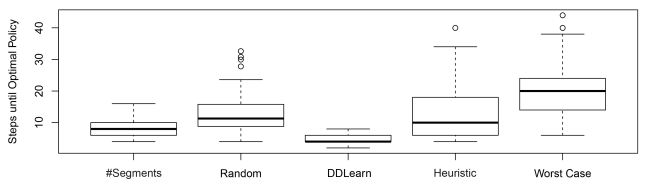

Figure 2: Summary of results for individual query disambiguation

iQuestion, they would also show an optimal disambiguation when the same iQuestion is presented twice. However, active learning methods do not explore different policies and do not consider future rewards. In contrast, our approach does not rely on a deterministic greedy strategy based on a fixed heuristic ranking, but can learn an abstract disambiguation strategy that generalizes beyond single iQuestions. Thus, DDLearn can improve with every iQuestion, beyond the individual answer of this particular iQuestion.

As reported in Table 3, *DDLearn* needs a training phase to learn the optimal sequence. We trained DDLearn on a sequence of three iQuestions, namely: *Give me all songs by Petula Clark.*, *Since when does Fugazi exist?*, *How many artists are called John Willams?*. This setup is called *DDLearn heuristic-free series* and its performance is tested on the remaining unseen 83 iQuestions.

The performance is compared to active learning in order to assess whether the *series* setup requires less dQuestions in the first episode of an unseen iQuestion. For about half of the test-iQuestions, DDLearn required fewer dQuestions for disambiguation than active learning. For 28 (34%) of iQuestions, *DDLearn heuristic-free series* was able to beat active learning on average by two dQuestions. Please note that this obviously does not apply to any combination of train- and test-iQuestions, specifically for the QALD iQuestions which are very heterogeneous. Still, for this small number of question it is a convincing proof of concept, although of course more experiments will be required to investigate this further.

Summing up, we presented the first reinforcement learning based disambiguation dialogue learner (*DDLearn*) that is evaluated on a reasonably sized real-world *linked data* set. Although it learns perfect disambiguation policies and appears to have an advantage over active learning techniques in our preliminary experiments, the question remains how such an approach is applied in broader domains with "real" human users. In this respect, we are mainly restricted by the availability of suitable training data, but it is likely that more sophisticated generalization techniques are needed to deploy this in a large real-world system with a large number of user interactions. One idea to generate such training data would be to exploit implicit user feedback as it is generated by user interactions with structured content as well as expressive natural language content like user ratings, comments and discussions. Besides the data, the conversational repertoire needs to be extended e.g., to also disambiguate the query type or to cover even more general purpose communications beyond question answering. This would also reduce the dependency on the pre- and post-processing components. Eventually, this might allow to train a flexible and adaptive conversational agent for any purpose.

References

Cuayahuitl, H. 2009. *Hierarchical Reinforcement Learning for Spoken Dialogue Systems*. Ph.D. Dissertation, University of Edinburgh.

Damljanovic, D.; Agatonovic, M.; and Cunningham, H. 2012. Freya: An interactive way of querying linked data using natural language. In *ESWC 2011 Workshops*.

Driessens, K. 2004. Relational reinforcement learning.

Dzeroski, S.; Raedt, L. D.; and Driessens, K. 2001. Relational reinforcement learning. *Machine Learning* 43(1/2):7–52.

Ghavamzadeh, M., and Mahadevan, S. 2004. Learning to communicate and act using hierarchical reinforcement learning. In *Procs. AAMAS'04*.

Lehmann, J., and Bühmann, L. 2011. Autosparql: Let users query your knowledge base. In *The Semantic Web: Research and Applications*.

Levin, E., and Pieraccini, R. 1997. A stochastic model of computer-human interaction for learning dialog strategies. 1883–1886.

Lison, P. 2013. Model-based bayesian reinforcement learning for dialogue management. In *Procs. of the 14th Annual Conference of the International Speech Communication Association (Interspeech 2013).* (accepted).

Nickles, M., and Rettinger, A. 2013. Interactive relational reinforcement learning of concept semantics. *Machine Learning* 1–36.

Nickles, M. 2012. A system for the use of answer set programming in reinforcement learning. In *Logics in Artificial Intelligence*, volume 7519 of *LNCS*. Springer.

Ponsen, M.; Croonenborghs, T.; Tuyls, K.; Ramon, J.; and Driessens, K. 2009. Learning with whom to communicate using relational reinforcement learning. In *Procs. AAMAS'09*.

Rieser, V., and Lemon, O. 2011. *Reinforcement learning for adaptive dialogue systems*. Springer.

Tran, T.; Wang, H.; Rudolph, S.; and Cimiano, P. 2009. Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In *Procs. of the 2009 IEEE International Conference on Data Engineering*, 405–416.

van Otterlo, M. 2005. A survey of reinforcement learning in relational domains. Technical report.