Discovering Knowledge Graph Schema from Short Natural Language Text via Dialog

Subhasish Ghosh
TCS Research, India
g.subhasish@tcs.com

Arpita Kundu
TCS Research, India
arpita.kundu1@tcs.com

Aniket Pramanick
TCS Research, India
aniket.pramanick@tcs.com

Indrajit Bhattacharya
TCS Research, India
b.indrajit@tcs.com

Abstract

We study the problem of schema discovery for knowledge graphs. We propose a solution where an agent engages in multi-turn dialog with an expert for this purpose. Each mini-dialog focuses on a short natural language statement, and looks to elicit the expert's desired schema-based interpretation of that statement, taking into account possible augmentations to the schema. The overall schema evolves by performing dialog over a collection of such statements. We take into account the probability that the expert does not respond to a query, and model this probability as a function of the complexity of the query. For such mini-dialogs with response uncertainty, we propose a dialog strategy that looks to elicit the schema over as short a dialog as possible. By combining the notion of uncertainty sampling from active learning with generalized binary search, the strategy asks the query with the highest expected reduction of entropy. We show that this significantly reduces dialog complexity while engaging the expert in meaningful dialog.

1 Introduction

Increasingly, within and outside of enterprises, knowledge is represented and stored in knowledge graphs (Pujara and Singh, 2018). Many such knowledge graphs have a schema (Mitchell et al., 2015), so that entities and relationships are labeled with discrete categories. There has been a lot of research on annotating natural language text using the *provided* schema of such knowledge graphs(Chakrabarti and Talukdar, 2017).

A task that is often overlooked in this context is that of designing and updating the knowledge graph schema itself. It is common for such knowledge graphs schemas to be incomplete. The schema is typically designed by a human expert ahead of time based assumptions about the data to be accommodated in the future. Thus schema design is not a one-shot affair, since the expert becomes better aware of the need once the system is deployed. Updating the schema based on new needs forms an integral part of maintaining a knowledge graph.

In this paper, we propose dialog with an expert as a means to construct and update such a schema using as evidence a collection of short natural language statements from the domain. The role of dialog has been explored in the context of updating schema-free knowledge(Hixon et al., 2015; Pappu and Rudnicky, 2014a,b). Our purpose is to recover the expert's desired schema for such knowledge. The proposed overall dialog consists of a sequence of many mini dialogs. Each mini dialog focuses on a single short natural language statement, and recovers the expert's desired schema-based annotation of the statement, taking into account possible enhancements to the current schema. The overall schema is augmented iteratively by performing dialog over the collection of statements. The dialog ends when the agent is able to confidently annotate all statements with its current schema.

As an illustration, consider the following mini dialog between a human (H) and an agent (A).

A: I need help with the statement Tom Mitchell works on never-ending learning. Is Tom Mitchell a Researcher?
H: Yes.
A: Is never ending learning any-of Paper or Topic?
H: It's a topic.
A: Is this about Researcher writing Paper about Topic?
H: No.
A: Is this about a new entity type?
H: Yes.
A: What is the name of the new entity?
H: This is about a project
A: Is this about Researcher related to Project related to Topic?
H: Yes, that is right.
A: How is Project related to Researcher?
H: Researchers lead projects.
A: How is Project related to Topic?
H: Projects focus on topics.

Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue, pages 74 - 84

A: I understand the statement as researcher Tom Mitchell leading project focusing on topic never ending learning.

The agent had existing knowledge of certain entity and relationship types, but needed to introduce the new entity type *Project* and the relation types *leads(Researcher, Project)* and *focusesOn(Project Topic)* to the schema. It was then able to associate a short path from the schema (Research to Project to Topic) with the statement. It elicited this knowledge from the expert using a short dialog.

Queries are sequentially chosen based on their 'utility' - the reduction in uncertainty over path probabilities. This has high-level similarities with active learning for structured prediction (Culotta and McCallum, 2005; Settles and Craven, 2008). This considers eliciting the entire structured label for an instance from the expert *in one shot*. In reality, experts do not always answer queries. If the query is too 'complex' (e.g. "Which of the following 100 paths is the right schema path for this statement?"), the expert is very likely to not answer at all. On the other hand, a simpler query (e.g. "Is this statement about a project?") is more likely to get a response. Therefore, we consider a *mini dialog* for each instance consisting of a sequence of simple queries, based on their utility.

Each query has an associated probability of response from the user, thereby affecting its utility. The response probability depends on the complexity of the query. Active learning literature has studied no-response probability for complex queries(Yan et al., 2016). However, the notion of complexity in this work relates to closeness to the decision boundary. In the structured prediction setting, we hypothesize that the complexity relates more to the 'size' of the query.

In the presence of such response uncertainty, given a collection of statements and an initial knowledge graph, our goal is to design a dialog strategy that picks the statements in some sequence and elicits their desired schema paths from the expert with the shortest overall dialog length.

We propose a dialog strategy that combines the notion of uncertainty sampling from the active learning literature with that of generalized binary search. We iteratively pick the best statement to query, and then the query for the statement by considering expected entropy reduction, accounting for no-response probability. We propose two different types of categorical queries, *any-of* queries, where the expert confirms one part of a bi-partition of the current candidate set, and *which-of* queries, where the expert is asked to pick a specific candidate from a set. We model response probability as a parametric function of query complexity, which we represent as the sum of the lengths of the paths included in the query. We show that the standard active learning strategy falls out as a special case of our strategy when there is no response uncertainty. The no-response parameters are learnt by the agent as the dialog progresses.

We evaluate our strategy using a collection of short statements from the web-page of a large organization. We show that our proposed strategy results in meaningful and dialog with an expert that yields the expert's desired schema for the organization via significantly lower dialog complexity compared to multiple baseline strategies.

2 Related Work

Hixon et al. (2015) investigate the problem of augmenting knowledge graphs using an open dialog with the user to improve factual multiple-choice science question answering. Earlier work(Pappu and Rudnicky, 2014a,b) looks at designing dialog for the related task of information retrieval for academic events. In all of these cases, the back-end knowledge graph is type-free and does not involve a schema. Mazumder et al. (2019) address lifelong learning via dialog, where they query the expert back for supporting facts when confronted with queries for entities and relationships with insufficient evidence in the knowledge graph. In contrast, we focus on dialog for augmenting the schema for typed knowledge graphs.

There is existing work on active learning for structured output spaces(Culotta and McCallum, 2005; Settles and Craven, 2008; Tong and Koller, 2001). Of these, sequence annotation (Culotta and McCallum, 2005; Settles and Craven, 2008) also considers paths, but not on schema graphs. More importantly, these pose a single query to the expert for the structured label of an instance. In contrast, we propose to break this into a series of simple queries, in view of answer uncertainty, which is not considered in this line of work.

Active learning with imperfect labelers (Yan et al., 2016) considers no-response and query complexity. However, since this is not for structured labels, query complexity does not account for structural complexity.

3 KG Schema Learning Problem

Schema and statement schema paths: A Knowledge Graph schema is a graph $G = \{E, R\}$ where E is the set of entity types and R is the set of binary relation types. For example, a Research domain may have entity types *Researcher*, *Paper*, *Project* and *Topic*, and relation types *authorOf(Researcher, Paper)*, *affiliatedWith(Researcher, Project)* and *focusesOn(Project, Topic)*.

A typed knowledge graph (or KG in short) $K = (G, I)$ has a schema G and instances I of the entity and relation types in G. For example, our Research KG may contain entity instances *isResearcher(T. Mitchell)*, *isProject(NELL)*, *isPaper(Coupled semi-supervised learning)*, and relation instance *affiliatedWith(T. Mitchell, NELL)*, etc. We assume that every entity type has a single *Name* attribute.

Consider an example statement s_i: "A scientist wrote the paper 'coupled semi-supervised learning'". We restrict ourselves to short statements that refer to at most two entities and two entity types, which we call e_{i1}, e_{i2} and t_{i1}, t_{i2} respectively. In s_i, t_{i1}=*Researcher*, and 'scientist' is a *mention* of t_{i1}, which we denote as $m(t_{i1})$. t_{i2}= *Paper*, and its mention $m(t_{i2})$ is 'paper'. e_{i2} is a specific paper instance and its mention $m(e_{i2})$ is 'Coupled semi-supervised learning'. The first entity e_{i1} is not explicitly mentioned in this statement. Beyond mentioning the entities and their types, s_i also refers to a connection between them in the schema via the defined binary relations. We call this the *statement schema path* p_i, which is a subgraph of the schema. Here, p_i is Researcher-authorOf-Paper. Typically, given a statement, the above variables other than mentions are latent or unobserved. We will assume the availability of a probability model for the posterior distribution of these variables $P(e_{i1}, e_{i2}, t_{i1}, t_{i2}, p_i | m(e_{i1}), m(e_{i2}), m(t_{i1}), m(t_{i2}))$ given the mentions in the statement. In Sec.4, we define such a model and its corresponding inference and learning algorithms.

An important consideration for such a model in the context of schema discovery is its ability to consider entity types (also entities) not contained in the current KG schema (also instances). Consider the statement s_1 from the introduction: "Tom Mitchell works on never ending learning". The true schema path p_1 in this case looks as follows: Researcher-leads-Project-focusesOn-Topic. This includes the entity type *Project* and relation types *focusesOn(Project, Keyword)* and *leads(Researcher, Project)* which are not contained in the observed schema G; i.e., $p_1 \not\subset G$. In order to correctly interpret s_1, its schema path p_1, and therefore the schema, needs to be enhanced to include an additional entity type and two new relation types. In other words, the posterior distribution should be capable of assigning non-zero probability to previously unseen yet likely entity types and schema paths.

Given a set of statements S and an initial schema G^0, our goal is an enhancement G^* of G^0 such that the schema paths p_i for all statements $s_i \in S$ are contained in G^*. Along with G^*, the schema paths for the individual statements are unknown as well, and need to be identified.

Dialog for schema path discovery: The problem above is difficult to solve in practice, even without considering schema enhancements, and requires very large volumes of training data. To aid this generally intractable search, we propose a dialog with an expert. Our task is to design a dialog strategy that reduces the uncertainty (entropy) about the schema path p_i for each $s_i \in S$ — and as a result about the complete schema graph G^* — as much as possible given a dialog length as a budget. The length of the dialog is the aggregated complexity of all the queries posed to the expert during the dialog.

Our strategy uses one mini dialog for each question statement in S. The j^{th} mini dialog considers some statement $s_i \in S$, and poses a sequence of queries to identify the true schema path p_i for s_i. Each mini dialog consists of a sequence of simple mini-queries, denoted as $q(s_i)$. Specifically, the agent poses two kinds of close-ended mini-queries to the expert. The first is a binary (yes/no) query of the form "Is the statement about *any of* p_1 OR p_2 OR ... p_k?", where each p_1 is a possible path in G^*. The second is an n-ary query of the form "This statement is about *which of* p_1 OR p_2 OR ... p_k?" Note that the response for this query can be 'none'. The paths in the queries can include new entity and relation types that are not in the current schema graph. Note that such categorical queries can only recover the *structure* of p_i. An additional type of query therefore asks for the names of any new entity or relation types in p_i.

Any such query $q(s_i)$ has a *utility*. Intuitively, utility measures potential reduction in entropy of the posterior distribution over possible responses to the query. There reduction is 0 if the query gets no

response from the expert. The response probability depends on the complexity of the query. Our goal is design a strategy that can evaluate the utilities for the different query types taking into account no-response probability, and then select the mini-query with the best utility at each step.

4 Candidate Schema Paths

In this section, we discuss a probabilistic model for schema paths for a statement, and then an algorithm for finding the best schema path given a statement along with their probabilities. This is not the focus of our work, and ideally we would prefer to use a state of the art method for this task. There has been recent progress on joint linking of mentions in short statements to entities and relationships in a knowledge graph (Sakor et al., 2019). This is similar to our task, but does not consider the possibility of extending the current schema with new entity and relationship types. There is also work on inferring paths in schema-based knowledge graph given a query node (Lao et al., 2011) using random walks. We extend this line of work for statements with mentions while also considering new schema nodes in the random walk. In this section, we first explain our probabilistic model, and then the candidate path sampling algorithm.

4.1 Model for Schema Paths

For a statement s_i such as "Researcher Tom Mitchell works on never ending learning", and a current KG $K(G, I)$, our first goal is to define a posterior distribution for the types and path for s_i. This statement has entity mentions $m(e_{i1})$='Tom Mitchell' and $m(e_{i2})$='never ending learning', and one type mention $m(t_{i1})$='Researcher'. The second type mention is absent.

Mention identification is not the focus of our work, and we use simple unsupervised NLP techniques which were sufficient for our purposes. These may be substituted with more sophisticated supervised techniques when needed without affecting the remaining components of our solution. We assume that a verb phrase separates the first $(m(e_{i1}), m(t_{i1}))$ and second $(m(e_{i2}), m(t_{i2}))$ set of entity and type mentions, such as 'works on' in this statement. We use a combination of noun phrase detection and named entity detection from *nltk*[1] to identify m_{e1} and m_{e2}.

[1] https://www.nltk.org/

We factorize the posterior distribution as

$$\begin{aligned} &P(e_{i1}, e_{i2}, t_{i1}, t_{i2}, p_i | m_{i1}, m_{i2}) \\ =&P(e_{i1}, t_{i1} | m_{i1}) P(e_{i2}, t_{i2} | m_{i2}) P(p_i | t_{i1}, t_{i2}) \end{aligned} \tag{1}$$

We have used m_{i1} and m_{i2} as shorthand for $m(e_{i1}), m(t_{i1})$ and $m(e_{i2}), m(t_{i2})$ respectively. Here, the first term is the posterior distribution over the entity and type for the first mention pair, the second that for the second mention pair, and the third is the posterior for the schema path given the two entity types.

We assume the first two distributions to be identical. We associate two distributions with each entity type t_i in the current KG. The first θ_i^t is a distribution over entity instances e currently associated with t_i. For example, the *Researcher* type may have a non-zero probability over entity instances *Tom Mitchell*, *Will Cohen*, etc. The second ϕ_i^t is over possible mentions of this type. For example, the type *Researcher* may have non-zero probability over mentions *researcher*, *scientist*, *professor*, etc. An individual entity instance e_i also has a distribution ϕ_i^e over possible mentions of it. For example, entity *Tom Mitchell* has non-zero probability over different ways of mentioning the name, e.g. *Tom Mitchell*, *T. Mitchell*, etc. Accordingly, the posterior distribution over type and entity is further factorized as

$$\begin{aligned} &P(e_{i1}, t_{i1} | m(e_{i1}), m(t_{i1})) \propto \\ &P(m(e_{i1}) | e_{i1}; \phi^e) P(m(t_{i1}) | t_{i1}; \phi^t) P(e_{i1} | t_{i1}; \theta^t) \end{aligned} \tag{2}$$

For previously encountered mentions, we use these two distributions to identify the most likely type. For new mentions, the type could be one of the existing types in E or a new type. For this, all three sets of distributions are smoothed to allow for previously unseen mentions and entities.

We now come to the posterior distribution $P(p_i | t_{i1}, t_{i2})$ of the connecting schema path given the two entity types. We model the statement path as a random path in a partially-observed schema graph, with start probabilities over entity types and transition probabilities over relations between entity types. The path needs to start at t_{i1} and end at t_{i2}. This is similar to random walks used in (Lao et al., 2011) for link prediction in knowledge graphs. The difference is that we admit the possibility of previously unseen entity and relation types.

The probability of a statement path is defined as:

$$P(p_i|t_{i1}, t_{i2}; \pi, Q) = P_s(t_{i1}) \prod_{(t_j,t_k)\in p_i} P_t(t_k|t_j) \quad (3)$$

where $P_s(\_)$ is the distribution over start entities of the random walk, and $P_t(\_|t_j)$ is the transition distribution from entity type t_j to other entity types.

The definitions for π and Q need to account for new entity and relationship types. Accordingly, we define the probability $\pi(k)$ of the random path starting at entity type k as follows:

$$\begin{aligned} P_s(k) \quad &\propto n_k + \alpha_e \text{ for existing k} \\ &\propto \alpha_n \text{ for new k} \end{aligned}$$

Here, n_k is the number of previously seen edges from entity type k, and $\alpha_e > 0$ allows new start entities. Similarly, the probability $P_t(k|j)$ of entity type k following entity type j in the path is defined as follows:

$$\begin{aligned} P_t(k|j) \quad &\propto n_{jk} + \beta_{ee} \text{ for existing j and k} \\ &\propto \beta_{en} \text{ for existing j, new k} \\ &\propto \beta_{ne} \text{ for new j, existing k} \\ &\propto \beta_{nn} \text{ for new j and k} \end{aligned}$$

Here, n_{jk} is the number previously seen transitions from entity type j to entity type k, and $\beta_{ee} > \beta_{ne}, \beta_{en} > \beta_{nn} > 0$ allows transitions from and to previously unrelated and unseen entity types. The intuition is that (a) more frequently seen transitions are more likely, and (b) while encountering new entities and relationships is possible, that probability progressively reduces with increasing training count.

4.2 Sampling Algorithm for Schema Paths

Since all the distributions involved are multinomials, their parameters can be estimated in a closed form, given an initial KG and training statements labeled with schema paths. Given estimates of these parameters, we now address the problem of identifying possible candidate paths for a statement along with their probabilities.

Having defined the distribution over schema paths p for a statement s, we need to identify the top-k most likely schema paths. For this, we use a MCMC technique based on Metropolis Hasting sampling (Neal, 1993; Andrieu et al., 2003) that performs a random walk over the space of schema paths for a statement. This requires a proposal distribution $q(p'|p)$ over possible next paths p' given the current path p. We define the neighbors p' of any statement path p using two operations. (a) Vertex insertion: This introduces a vertex between currently adjacent entity types in p. For example, Researcher-rel-Topic has Researcher-rel-Paper-rel'-Topic as a vertex-insertion neighbor. Note that this inserted entity type can be an existing type or a new one. Vertex insertion can operate on any currently adjacent pair of types in the statement schema path. (b) Vertex collapse: This is the inverse of vertex insertion. This collapses an intermediate vertex of p by introducing a direct edge between its neighbors. Vertex collapse can operate any current intermediate vertex of the schema path. At each step, we sample a neighbor using a uniform proposal distribution $q(p'|p)$ over all the neighbors p' of the current schema path p defined by these two operations, and accept the sample with probability $A(p', p) = \min\{1, \frac{p(p')q(p|p')}{p(p)q(p'|p)}\}$, where the path probabilities follow Eqn.1.

5 Dialog for Schema Path Discovery

At this point, for each statement s_i, we have n candidate schema paths $p_{i1}, p_{i2}, \ldots$, and their probabilities. Our task now is to minimize the entropy of schema path predictions over all statements with a dialog of length L, knowing that the expert may not answer all queries.

Let us recall the standard active learning paradigm: (a) select the next statement for expert labeling, (b) acquire expert's preferred label for selected statement, (c) retrain model with newly labeled statement in training data, and continue until budget is exhausted. For selecting the next statement, the principle of uncertainty sampling is followed, with entropy as the notion of uncertainty (Hwa, 2004). In step (b), the expert provides the preferred label in one shot, even for structured output spaces (Culotta and McCallum, 2005). We call this interaction format the *active learning dialog format* and the overall strategy the entropy-based active learning strategy (**E-AL**).

Our major departure from this strategy is in step (b). We may present the expert with candidate schema paths and ask the expert to pick one. We call this a *which-of* query. When the list is long, this is unlikely to receive a response.

In addition, we can elicit the schema path with a mini-dialog, which is a sequence of simple mini-

queries. The basic idea is to iteratively prune the set of remaining candidates by splitting into two parts at each step, showing any one part to the expert and querying if it contains his preference. We call this an *any-of* query. Within a mini dialog (step (b)), our strategy repeatedly chooses the best (which-of or any-of) mini-query until the complete schema path is obtained from the expert. We call this the *mixed any-which mini dialog format*.

We can see that this is more general than the active learning dialog format.

Lemma 1. *The mixed any-which mini dialog format reduces to the active learning dialog format when each mini dialog has a single which-of query.*

For completion, we point out that if the elicited schema path in a mini dialog includes new entities and relationships, their names are also elicited from the expert via name queries. An example name query may be "What is the name of the new entity related to both researchers and topics?".

Generalized binary search and Entropy Reduction: At any step within a mini dialog, the strategy needs to choose between the which-of query and many possible any-of queries, one for each bi-partition of the remaining candidate set. To evaluate different queries, we define the utility $u(q)$ of a query q borrowing from entropy-based uncertainty sampling and generalized binary search (Pelc, 2002).

Let P_i^k be the set of remaining candidates at step k of the mini dialog for statement s_i, and e_i^k denote the entropy of the candidate distribution. A bi-partition π_i^k splits P_i^k into P_{i1}^k and P_{i2}^k such that the entropies of the two splits are e_{i1}^k and e_{i2}^k. Depending on the expert's response to the any-of query with this bi-partition, either P_{i1}^k or P_{i2}^k becomes the next candidate set. So the residual entropy after this query is either $(e_i^k - e_{i1}^k)$ or $(e_i^k - e_{i2}^k)$. We define the utility $u(q)$ of this bi-partition (any-of) query q as the *average reduction of entropy* after the query, which is $p_{i1}^k(e_i^k - e_{i1}^k) + p_{i2}^k(e_i^k - e_{i2}^k)$, where p_{i1}^k (p_{i2}^k) is the sum of candidate probabilities in the first (second) split.

Instead of all partitions, we order the candidate schema paths by probability, and consider each position in the order as a possible splitting point.

The alternative to partitioning is to present the entire set of candidates P_i^k to the expert and pose a which-of query. If the expert responds to this query, this mini dialog concludes and the entropy becomes 0, so that the utility (reduction in entropy) of the which-of query is e_i^k. But this query is 'complex' and may not be answered by the expert.

Response uncertainty and Expected utility: Utility as defined above is not sufficient when queries are not answered with certainty. We define the complexity $c(q)$ of a query q as the sum of lengths of the paths specified in the query. For example, the query "Is this about Researchers-relatedTo-Projects-relatedTo-Topics?" has complexity 3, while the query "Is this about Researchers-relatedTo-Projects-relatedTo-Topics OR Researchers-relatedTo-Topics?" has complexity 5. We model the no-reponse probability $r(q)$ for a query as a function of its complexity. We have the option of various squashing functions which map positive integers to $[0, 1]$. We use the generalized logistic function:

$$r(q) = \frac{1}{1 + \exp(-w \times (c(q) - t))} \qquad (4)$$

Using this definition of no-response probability, we define the *expected utility* of a query as $\bar{u}(q) = u(q)(1 - r(Q))$. Our strategy picks that mini-query at step k of a mini dialog that maximizes this expected utility. If the expert does not respond, then the next best mini-query is posed.

Having introduced the notion of expected utility to deal with no-response probability, we also modify step (a) of the framework by using expected utility instead of entropy to select the next statement for mini dialog. The expected utility of a statement is taken to be the maximum of the expected utilities of the which-of query and the possible any-of queries for its candidate set. This completes the description of our overall expected utility based dialog strategy (**EU**).

Theorem 2. *When the expert's response probability is 1, and the dialog strategy is aware of this, the* **EU** *dialog strategy recovers the entropy-based active learning strategy (***E-AL***).*

The proof follows from the observation that when response probability is 1, the query with the highest expected utility is the which-of query on the entire set of candidates, and a mini dialog reduces to a single which-of query. So the dialog format becomes identical to that of active learning. Further, expected utility and entropy lead to the same statement being selected for querying.

Thus **E-AL** is a special case of the **EU** dialog strategy which is meaningful when the expert al-

ways responds. Our experiments show that **EU** far outperforms **E-AL** under response uncertainty.

Query for a partition: Having selected a bi-partition for an any-of query, the query for it needs to be formulated. Recall that *any-of* queries have the form "Is the statement about any of p_1 OR p_2 OR ...?". The naive query for a partition enumerates all the paths in the smaller split. Instead, we identify the smallest *discriminating path feature* for the two parts of the partition — such as nodes, edges, length-2 paths, etc. For example, if all the paths in one of the splits contain the entity *Paper*, and no path in the other split contains it, then a possible query for the partition is the following: "Is the statement about papers?". Being a less complex query, this is much more likely to get a response.

Estimating response probability: At the start of a dialog, the agent has an initial guess about the expert's response probability model. After receiving responses or non-responses for specific queries, the parameters of this model can be estimated. On the conclusion of a mini dialog, we update the parameters w and t in the standard way for logistic regression using gradient descent.

6 Experiments

For empirical evaluation, we experimented with statements collected from the website of a large multi-national company, to see how accurately an expert's desired schema behind the website data can be recovered. From 64 web-pages on the company's website, we picked short statements representative about the company's business, each containing at most 2 pairs of entity / type mentions. This resulted in a dataset of 850 short statements. In addition, we obtained from an expert a schema for the company's business, which covers the statements that were picked. The resultant schema contains 15 entity types and 17 relationships between these. Using this schema, we manually annotated the statements with entities, entity types and type paths. About 40% statements had schema paths of length 1 and remaining of length 2. We used 200 the statements for train and the remaining 650 for test, ensuring that all entity and relation types used in the test schema paths are covered in the train.

To simulate the expert for large-scale experimentation, we used an 'expert bot' that had knowledge of the gold standard schema paths. The no-response parameters of the expert bot were manually specified. For each query, the expert bot sampled from a Bernoulli distribution for the query to decide whether or not to respond.

To evaluate a dialog, we use the learning curve, where we plot the length of the dialog (terms of number of mini-queries) against the correctness of the inferred schema paths for the statements in the test set as compared with the gold-standard schema paths, evaluated using the F1 measure.

Our proposed dialog strategy **EU - L**, picks the next statement for mini-dialog and also the next mini-query within a mini-dialog using expected utility (EU) , and further learns (L) the no-response parameter based on the expert's action (response / no-response). We compare our strategy against a few baselines. **EU** does not learn the expert's no-response probability. **Random + EU** picks the next statement for dialog uniformly at random (R), instead of expected utility, but uses expected utility (EU) to selected the query within a mini dialog. **E-AL++** uses entropy (E) reduction assuming certain response for selecting the next statement for dialog as well as the next query within a mini dialog. Note that this is still more powerful than the *E-AL* strategy(Culotta and McCallum, 2005), which only poses *which-of* queries assuming the expert will respond. This makes no progress for non-zero no-response probability. In contrast, **E-AL++** uses the mixed any-which mini-dialog format and has the flexibility to pose *any-of* queries by partitioning. In addition, we also evaluate as a skyline **EU-O**, where the agent is an oracle (O) with perfect knowledge about the expert's no-response parameters.

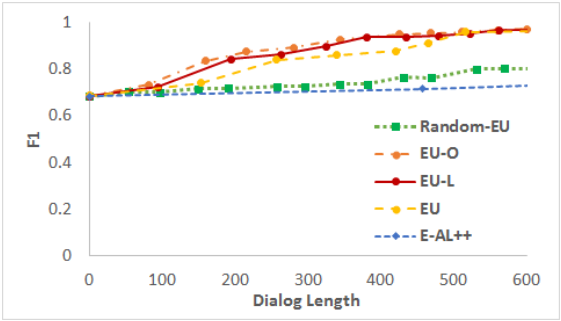

Figure 1: Learning curve for various dialog strategies

In the first experiment, the agent has knowledge of the complete schema, and does not need to consider new entity or relation types. In Fig.1, we plot the performances of **EU-L** along with the baselines. For each strategy, we plot the number of mini-queries on the x-axis and cumulative F1 on the test set after re-estimating parameters and re-

inferring paths at the end of a mini-dialog on the y-axis. **E-AL++** performs quite poorly and gets very little improvement in accuracy. Accounting for no-response via expected utility for selecting the query within a mini dialog (**Random + EU**) makes some improvement over this, but a **EU** makes a major improvement by using expected utility when selecting the next statement for dialog. Finally, **EU-L** makes a steady improvement by estimating the no-response parameters. Note that the skyline **EU-O** with perfect knowledge of the no-response parameters performs the best, but **EU-L** catches up with it as the dialogs progress, showing the effectiveness of our learning strategy. We also note that **E-AL++** can be seen as a special case of **EU** where the agent assumes that the expert always answers and does not update this model.

Average number of mini-queries per statement is very different for different strategies. This is 39 for **E-AL++**, 7.0 for **Random EU**, 7.0 for **EU** and finally to 6.0 for **EU-L**. Average query complexity (number of entity type nodes in a query) also varies significantly across the strategies. For **E-AL** (only *which-of* queries), average query complexity is 73.5, which explains why it makes no progress for non-zero no-response probability. This drops to 46 for **E-AL++**, 5.4 for **Random EU**, 4.8 for **EU** and finally to 2.7 for **EU-L**. Beyond dialog length, this also helps to highlight the benefit of expected utility and learning.

In the first experiment, the agent had knowledge of the complete schema, and only needed to detect occurrences of known entity and relation types in schema paths. We call this agent **EU-L Detect**. In the second experiment we compare this with an agent **EU-L Discover**, which only has partial knowledge of the schema at the start, and discovers new entity and relation types via dialog. In the training data for **EU-L Discover**, we randomly removed 5 entity types from the schema, which appear either as intermediate nodes as well as end nodes in the statement paths. We also removed the 9 relationships involving these entity types. This resulted in a pruned schema with 10 entities and 8 relationships. Statements associated with prune relationships were removed from the training data. In contrast, **EU-L Detect** was given the complete training data and the complete schema.

Fig.2 shows the performance of the two agents. Unsurprisingly, performance of *EU-L Discover* trails that of *EU-L Detect*, but it makes steady improvement as the dialog progresses.

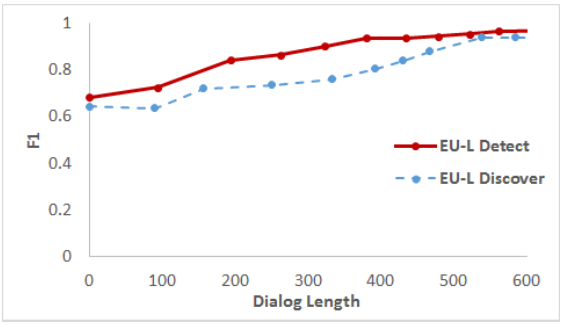

Figure 2: Learning curves for discovery and detection

In summary, we have shown that with response uncertainty, the **EU** strategy works significantly better than **E-AL** for detecting and discovering schema paths for short natural language statements, thereby enabling the enhancement of the underlying schema for the domain. On top of this, estimating the no-response parameters of the expert leads to further improvements in the learning curve.

7 Discussion and Future Work

We have introduced the problem of discovering the schema for the knowledge graph for a domain by engaging in dialog with an expert about interpreting short natural language statements in terms of the desired schema. We have proposed dialogue strategy that is aware of no-response probability for the expert, and accounts for it in its strategy by splitting the interaction for a statement into a sequence of short and simple queries, which are chosen using expected utility. The agent also estimates the no-response parameters for an expert. We have demonstrated that the proposed strategy is able to discover a schema starting from an initial partial observation. This goes well beyond the state of the art in active learning for structured spaces.

However, much remains to be done. One shortcoming of our expert model is that we have considered no-response to be the only form of 'noise'. In reality, the expert, when presented with a complex question, may provide an incorrect response, and the algorithm should be resilient to a small probability of such erroneous responses. Next, we have only considered simple statements for which the statement graphs are paths. In general, such statement graphs can be trees or directed acyclic graphs. We will investigate these aspects in future work.

References

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael Jordan. 2003. An introduction to mcmc for machine learning. *Machine Learning*, 50:5–43.

Soumen Chakrabarti and Partha Talukdar. 2017. Tutorial on knowledge extraction and inference from text: Shallow, deep, and everything in between. In *International Conference on Information and Knowledge Management (CIKM)*.

Aron Culotta and Andrew McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *Annual Conference of the American Association for Artificial Intelligence (AAAI)*.

Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. 2015. Learning knowledge graphs for question answering through conversational dialog. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Rebecca Hwa. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30(3):73–77.

Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Sahisnu Mazumder, Bing Liu, Shuai Wang, and Nianzu Ma. 2019. Lifelong and interactive learning of factual knowledge in dialogues. In *Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapa Nakashole, Emmanouil Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. 2015. Never-ending learning. In *Conference on Artificial Intelligence (AAAI)*.

Radford Neal. 1993. Probabilistic inference using markov chain monte carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto.

Aasish Pappu and Alexander Rudnicky. 2014a. Knowledge acquisition strategies for goal-oriented dialog systems. In *Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Aasish Pappu and Alexander Rudnicky. 2014b. Learning situated knowledge bases through dialog. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*.

Andrzej Pelc. 2002. Searching games with errors fifty years of coping with liars. *Theoretical Computer Science*, 270(1):71–109.

Jay Pujara and Sameer Singh. 2018. Mining knowledge graphs from text. In *Tutorial at ACM International Conference on Web Search and Data Mining (WSDM)*.

Ahmad Sakor, Isaiah Onando Mulang, Kuldeep Singh, Saeedeh Shekarpour, Maria Esther Vidal, Jens Lehmann, and Sren Auer. 2019. Old is gold: Linguistic driven approach for entity and relation linking of short text. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Simon Tong and Daphne Koller. 2001. Active learning for structure in bayesian networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Songbai Yan, Kamalika Chaudhuri, and Tara Javidi. 2016. Active learning from imperfect labelers. In *International Conference on Neural Information Processing Systems (NIPS)*.

A Appendices

As appendix, we provide more details on the statement data preparation and include some example mini-dialogs generated on test statements by the **EU-L** strategy.

A.1 Statement Data Preparation

We prepare our statement data from the website of Tata Consultancy Services (TCS), which covers different Industries, Services, Products and Platforms and has web-pages under these covering Solutions, Insights, Customer Testimonials and Success Stories, White papers, etc. Each web-page is partially structured mentioning key people and their designations, names of clients, services and industries, along with descriptions of challenges, innovations and results. We model each web-page as a rooted directed acyclic graph, with each of the above as individual nodes. Then we consider pairs of nodes from within each web-page and create a statement with these by connecting them with templatized relation strings. The following are examples: *Client Woolworths Supermarkets appears in the context of Technology Operations*, *Malaysia Airlines is mentioned in the context of Industries Airline Industry*, etc. Note that the original web-pages mention entity types in some cases, such as in *Client Woolworths Supermarkets*, but not in others, such as in *Malaysia Airlines*. The statements are created to

match this data. For each statement, the unique connecting path between the two end nodes is initially set as the gold-standard path and then verified manually.

A.2 Example Mini Dialogs

We present three example mini-dialogs between the agent (A) and the Expert (E) bot. Each agent query is annotated with type of the query and its subject. The query type can be any-of, which-of or name query. Further, an any-of query can be enumerated or posed using a discriminating feature, such as node or short sub-path. The subject of the query can be the entity, its type or the path connecting two types. In addition to queries, the agent also provides context to the expert by stating facts that about which it is confident. These illustrate that the agent has basic NLU capabilities so that the expert does not have to stick to categorical answers.

The first example mini-dialog is a short one and includes examples of any-of enumeration queries on path and type and a which-of path query.

A: I need help with the statement *Client Woolworths Supermarkets appears in the context of Technology Operations*.
A: In *Client Woolworths Supermarkets*, I know *Client* means type CLIENT.
A: I know *Woolworths Supermarkets* means CLIENT Woolworths Supermarkets.
A: Is the type of *Technology Operations* any-of the following: SERVICE? [any-of-enumeration, type]
E: yes
A: Is the statement about any-of the following? CLIENT related-to SERVICE? [any-of-enumeration, path]
E: no
A: Is the statement about any-of the following? [any-of-enumeration, path]
1.CLIENT provides CUSTOMER TESTIMONIAL for SERVICE
2.CLIENT has SUCCESS STORY for SERVICE
E: yes
A: Which of the above? [which-of, path]
E: first one
A: Ok. I understand the statement as Client Woolworth provides Customer Testimonial for Service Technology Operations.

The second example mini-dialog additionally includes examples of which-of query on type, any-of discriminating node query on path, and discovery of a new entity.

A: I need help with the statement *Malaysia Airlines appears in the context of Industries Airline Industry*.
A: Is the entity for *Malaysia Airlines* any-of the following: new entity? [any-of-enumeration, entity]
E: yes
A: The type for *Malaysia Airlines* is which of the following? [which-of, type]
1.CUSTOMER TESTIMONIAL
2.RESULTS
3.PARTNERSHIP
4.LEADERSHIP
5.CLIENT
6.SERVICE
7.INDUSTRIES
E: It is a Client
A: In *Industries Airline Industry*, I know *Industries* means type INDUSTRIES.
A: Is the entity for *Airline Industry* any-of the following: new entity of type INDUSTRIES? [any-of-enumeration, entity]
E: yes
A: Is the statement about any-of the following: SUCCESS STORY? [any-of-discriminating-node, path]
E: yes
A: Ok. I understand the statement Client Malaysia Airlines has Success Story for Industries Airline Industry.

The third example mini-query additionally includes examples of a no-response from the expert, discovery of new types and relations and name queries for these.

A: I need help with the statement *Client Ceridian appears in the context of Industries Communications, Media & Technology*.
A: In *Client Ceridian*, I know *Client* means type CLIENT.
A: Is the entity for *Ceridian* any-of the following: new entity of type CLIENT? [any-of-enumeration, entity]
E: Yes
A: In *Industries Communications, Media & Technology*, I know *Industries* means type INDUSTRIES.
A: Is the entity for *Communications, Media & Technology* any-of the following: new entity of type INDUSTRIES? [any-of-enumeration, entity]
E: yes
A: Is the statement about any-of the following: CLIENT has SUCCESS STORY for INDUSTRIES? [any-of-enumeration, path]
E: no
A: Is the statement about any-of the following: CLIENT related to INDUSTRIES? [any-of-enumeration, path]
E: no
A: Is the statement about any-of the following? [any-of-enumeration, path]
1.CLIENT related to SERVICE mentioned in SUCCESS STORY for INDUSTRIES

2.CLIENT has SUCCESS STORY for SERVICE related to INDUSTRIES

3.CLIENT related to SERVICE related to INDUSTRIES

4.CLIENT related to BENEFITS related to INDUSTRIES

E: [no answer]

A: Is the statement about any of the following: NEW TYPE related to INDUSTRIES? [any-of-discriminating-edge, path]

E: yes

A: What is the name of the new type? [name, entity]

E: customer testimonial

A: Could you tell me how CLIENT is related to CUSTOMER TESTIMONIAL? [name, relation]

E: client provides customer testimonial

A: Could you tell me how CUSTOMER TESTIMONIAL is related to INDUSTRIES? [name, relation]

E: customer testimonial belongs to industries

A: Ok. I understand Client Ceridian provides Customer Testimonial belongs to Industries Communication, Media & Technology.