Domain-Aware Dependency Parsing for Questions

Aparna Garimella[*]
Adobe Research
garimell@adobe.com

Laura Chiticariu
IBM Research
chiti@us.ibm.com

Yunyao Li
IBM Research
yunyaoli@us.ibm.com

Abstract

Parsing natural language questions in specific domains is crucial to a wide range of applications from question-answering to dialog systems. Pre-trained parsers are usually trained on corpora dominated by non-questions, and thus perform poorly on domain-specific questions. Retraining parsers with domain-specific questions labeled with syntactic parse trees is expensive, as these annotations require linguistic expertise. In this paper, we propose an automatic labeled domain question generation framework by leveraging domain knowledge and seed domain questions. We evaluate our approach in two domains, and release the generated question datasets. Our experimental results demonstrate that auto-generated labeled questions indeed lead to significant ($4.9\% - 9\%$) increase in the accuracy of state-of-the-art (SoTA) parsers on domain questions.

1 Introduction

Understanding questions is the first step towards building accurate and reliable natural language interfaces. Recent works on Google Assistant, or IBM Watson focus on building domain-specific conversational agents. In this paper, we focus on syntactic parsing of domain-specific questions, which is crucial in domain-specific agents. The accuracy of syntactic parsers is known to depend on the syntactic similarities between the training data and application text. However, questions are often underrepresented in classic training corpora. In Penn TreeBank, only 0.5% of sentences from the Wall Street Journal are questions, with majority being rhetorical in nature, and those occurring in conversations (starting with interrogatives *wh-*/*how*, imperatives *show me*, *name*, or *yes*/*no* questions) are heavily underrepresented. Recognizing this problem, Judge et al. (2006) introduced QuestionBank, a labeled corpus of 4,000 general questions.

However, domain-specific questions are often underrepresented in general purpose question corpora, leading to their poor parsing performances; e.g., in *Show me Neil's insider transactions since 2011*, a SoTA parser (Nivre et al., 2016a) trained on Universal Dependencies (UD) English TreeBank (Silveira et al., 2014) and QuestionBank attaches *since* to *show*, instead of *transactions*, causing the system to misinterpret Neil's insider transactions since 2011, to all his transactions. In *Will it rain tomorrow by noon?*, *tomorrow* is attached to *rain* with a wrong dependency relation (`dobj` instead of `nmod:tmod`), causing the system to miss the temporal aspect of the question.

A natural solution to obtain accurate domain-specific parsers is to train them on domain-specific corpora. However, obtaining domain-specific questions is difficult. Moreover, annotating questions for parse trees is tedious, prone to errors and inconsistencies, and requires linguistic expertise. Petrov et al. (2010) proposed *uptraining*, training a parser on the output of a slower, more accurate parser. For acceptable performance, the unlabeled corpus must be large (100,000 questions). Our method is applicable when such a large corpus is not available. Inspired by (Wang et al., 2015) who showed that semantic parsers can be built "overnight" using domain expertise, we seek to reduce the effort required to handle a new domain using *domain knowledge*: (1) a *domain schema* modeling the concepts and relationships in a domain, and (2) a *knowledge base* of data instances that populate the schema (Hamon et al., 2017; Julien Gobeill and Ruch, 2015; Damljanovic et al., 2010). To the best of our knowledge, this is the first work to use domain knowledge to improve syntactic parsing.

This paper makes two main contributions. (1) We propose a framework to automatically generate

[*]This work was done when the author was at IBM Research.

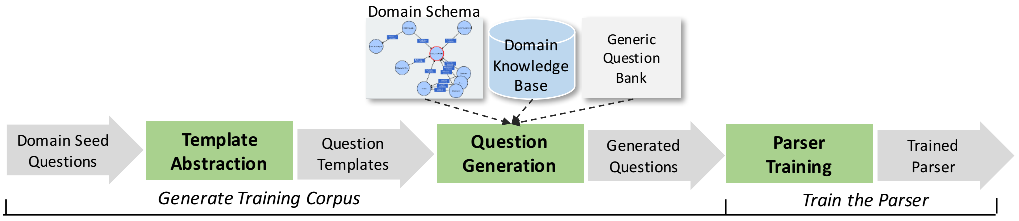


Figure 1: Overview of our approach.

a large training corpus of domain-specific questions for syntactic parsing in English, from a few seed domain questions using domain knowledge. We evaluate our approach in general (Weather) and specialized (Finance) domains. The parsers trained on the data augmented with our method have significant improvements ($4.9\% - 9\%$ LAS, $4.8\% - 8.8\%$ UAS) over those trained on UD Treebank and QuestionBank. Our method is robust to small seed, improving accuracy with as few as 10 seed questions. (2) We release the datasets and generated questions to the community.[1]

2 Question Generation

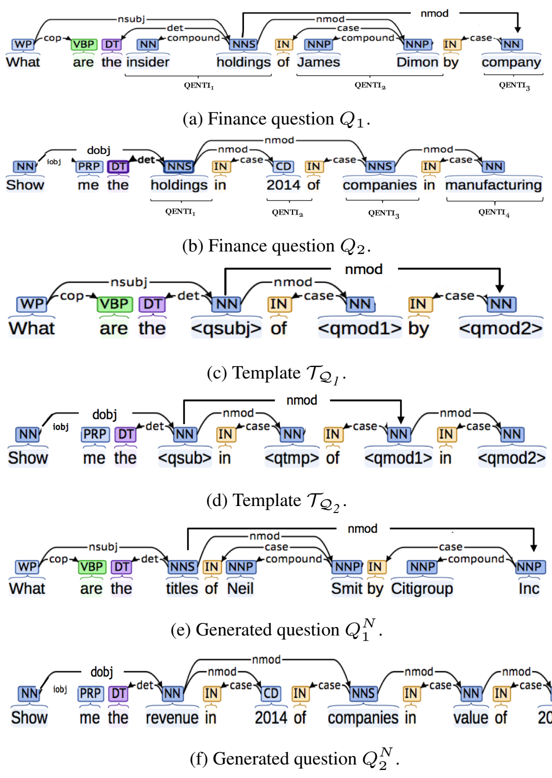
We automatically generate large training corpora by combining (1) domain seed questions labeled with syntactic dependencies using the Universal Dependencies v1.4 (Nivre et al., 2016b) guideline, and (2) domain knowledge. We use a two-stage pipeline: For each seed question, a *question template* is created that maintains its general structure, but abstracts away details of specific entities. Next, new questions are generated by automatically filling the templates with new entities obtained using domain schema and knowledge base. The domain schema is further annotated to ensure naturalness of generated questions (Figure 1).

2.1 Template Abstraction

Given labeled seed, this stage involves abstracting *templates* from their parse trees. We focus on two entity types, based on the dependency relation of nodes to their parent in parse trees of questions:

Subject Entity (`qsubj`) is the subject in a question. In *wh*-interrogatives, it occurs as the nominal subject (`nsubj`) of root. (QENT$_1$ in Fig. 2a). In imperatives, it occurs as the direct object (`dobj`) of the imperative verb. (QENT$_1$ in Fig. 2b).

[1]https://github.com/System-T


What are the insider holdings of James Dimon by company
Show me the holdings in 2014 of companies in manufacturing
What are the <qsubj> of <qmod1> by <qmod2>
Show me the <qsub> in <qtmp> of <qmod1> in <qmod2>
What are the titles of Neil Smit by Citigroup Inc
Show me the revenue in 2014 of companies in value of 20,000

(a) Finance question Q_1.

(b) Finance question Q_2.

(c) Template $\mathcal{T}_{Q_1}$.

(d) Template $\mathcal{T}_{Q_2}$.

(e) Generated question Q_1^N.

(f) Generated question Q_2^N.

Figure 2: Finance questions (a,b), their templates (c,d) and examples of generated questions (e,f).

Modifier Entity (`qmod`) is a (noun phrase) node that modifies another node. In parse trees, they usually relate via an `nmod` dependency.[2] In Fig. 2a, 'James Dimon' and 'company' are `qmod` entities of 'insider holdings'. Given a question, its *template* has `qsubj` and `qmod` entities replaced by placeholders. Algorithm 1 details template abstraction.

Domain Schema and Knowledge Base We assume domain schema as a set of *classes* with *properties*, *relations* between classes, and knowledge base conforming to the schema with few data instances; e.g., `title` is a property of class `AssignmentHistory` with instances 'CEO' and

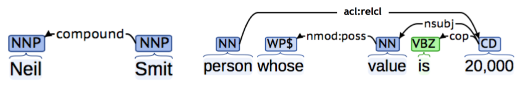
[2]We also account for other dependencies (`nmod:poss`, `appos`, `xcomp`) modifying `qsubj`.

'COO' in the Finance domain. Figure 5 in Section B shows an example schema for financial domain and a few data instances in a knowledge base $\mathcal{KB}$ conforming to this schema.

2.2 Template Filling

Given a template, questions are generated by systematically filling `qsubj` and `qmod` with new values obtained from domain schema and knowledge base. The main challenges include: (1) which new values are suitable for filling? and (2) how to automatically construct parse trees for generated questions? To address the first challenge, we propose ***template replacement heuristics***: `qsubj` is replaced with properties of a class, while `qmod` is replaced with properties of other classes in relation to that class. Further, we introduce one-level nesting on the filled `qmod` by expanding them using the relative pronoun *whose* (further details in Section C).

We address the second challenge by constructing ***parse trees for template fillers*** as follows. (1) For each class property, we manually provide the parse tree. This incurs a small one-time effort proportional to the size of the domain schema, which is small compared to the knowledge base. (2) For each instance of a class property, we automatically generate the parse tree by making the last word as root, and attach the preceding words (Fig. 3a). (3) For `qmod`, we construct relative pronoun expansion node by attaching to it the parse tree of a property of a related class with tag `acl:relcl` and *whose*. Fig. 3b expands $\texttt{qmod}_1$ = `Person:name='Neil Smit'` using the parse tree of `Holding:value` with a prepositional attachment for a value (*i.e.,* 20,000). The preposition is changed to an appropriate copular verb (*is*, *are*).



(a) (b)

Figure 3: Parse trees for template fillers based on heuristics for data instances (a), and and relative pronoun expansion (b).

Tab. 1 shows example questions generated in the Finance domain (Q_1^N to Q_4^N).

While the generated questions exhibit variety, most are unnatural; more natural formulations are Q_1^F to Q_4^F. The common sources of unnaturalness are the following.

Incorrect preposition for `qmod` In Q_1^N, the original preposition *by* is incorrect for the choice of `qsubj` and $\texttt{qmod}_2$; it should instead be *in*.

Incomplete usage of dependent property In Q_2^N, the choice of property for `qsubj` is incomplete; start date is not independent by itself and should be associated with a title the person holds.

Incomplete semantics of property In Q_3^N, the usage of property for `qsubj` is misleading as it refers to the value of a company, while the intent is to query the value of holdings in the company.

Incorrect question word In Q_4^N, *what* should be replaced with *who*, since `qsubj` is filled with a person, as opposed to an object.

2.3 Schema Annotations

A random sample of 100 generated questions contains 68 incorrect prepositions, 47 incomplete dependents, 64 incomplete semantics, and 2 incorrect question words. We address them using simple annotations to the domain schema, provided by a domain expert, in a one-time effort that is linear with the size of the schema.

Class Relations. between classes are annotated with *connective words* (usually prepositions), *e.g.*, $\texttt{AssignmentHistory} \xrightarrow{\text{in}} \texttt{Company}$ and $\texttt{AssignmentHistory} \xrightarrow{\text{of}} \texttt{Person}$ in Figure 5. This annotation addresses the incorrect prepositions for `qmod` (e.g. Q_1^F).

Heading Properties. are those that can be queried independently without referencing others. Each property is annotated as heading or non-heading for all the classes in the schema. While heading properties and their instances can be used to fill `qsubj` and `qmod` independently, we devise rules to use non-heading ones (Appendix D). This annotation modifies the use of non-heading property *start date* in Q_2^N by associating it with an instance 'CEO' of heading property *title* in Q_2^F.

Class-dependency of Properties. This addresses the incomplete semantics of properties. Certain properties are ambiguous, and querying them requires specifying their class names to add context; *e.g.*, one would ask *What are the values of holdings in Citigroup?* as oppose to *What are the values in Citigroup?* Properties in the schema are annotated as class-dependent or not. This annotation leads to Q_3^F, a more natural version of Q_3^N.

Possible Question Words. To address the incorrect question words (Q_4^N), we annotate all properties and their instances with corresponding possible *wh-* question words (Q_4^F).

Naively generated question	**Final question**
Q_1^N: What are the titles of Neil Smit *by* Citigroup Inc?	Q_1^F: What are the titles of Neil Smit *in* Citigroup Inc?
Q_2^N: What are the *start dates* of Neil Smit by Citigroup Inc?	Q_2^F: What are the *start dates as CEO* of Neil Smit *in* Citigroup Inc?
Q_3^N:What are the *values* of Citigroup Inc by Neil Smit?	Q_3^F: What are the values *of holdings in* Citigroup Inc *of* Neil Smit?
Q_4^N: *What* are the CEOs of Citigroup Inc?	Q_4^F: *Who* are the CEOs *in* Citigroup Inc?

Table 1: Questions generated from "What are the insider holdings of James Dimon by company" with template: "What are the `qsubj` of `qmod₁` by `qmod₂`?"

DATASET		DOMAIN	# SENT.
UD	UD Treebank	General	16,622
GQ	QuestionBank	General	4,000
DQ_F	Domain-specific questions	Finance	250
DQ_W	labeled by linguist	Weather	250

Table 2: Datasets in our experiments.

	FINANCE		WEATHER	
DATASET	LAS	UAS	LAS	UAS
UD	78.44	80.71	74.26	78.09
UD+GQ	80.62	82.7	79.48	82.44
UD+DQ	85.06	86.96	79.06	82.26
UD+N_{DQ}	86.34	88.61	79.47	82.89
UD+GQ+DQ	86.05	87.88	81.97	84.42
UD+GQ+N_{DQ}	88.31*	90.41†	84.37*	87.21†
UD+GQ+DQ+N_{DQ}	**89.67***	**91.53†**	**84.82***	**87.49†**

Table 3: Performance of Malt with different training sets. LAS with * (UAS with †) are statistically significant compared to all other settings using McNemar's test ($p < 0.05$).

3 Experiments and Results

We evaluate our approach with 5-fold cross-validation in the Finance (9 classes, 16 relations, 63 props., 3,028 instances) and Weather (9 classes, 1 relation, 85 props., 66 instances) domains.

Data. Table 2 summarizes different datasets used in our experiments. For each domain, we randomly sample 250 questions from the query logs of an internal QA system, and manually annotate them with dependency parse information (DQ_F and DQ_W). We group the questions based on their template structure, resulting in 148 and 166 unique templates for Finance and Weather, respectively. Questions corresponding to 20% templates are used for training in each fold, resulting on average in 38 and 54 seed questions from 30 and 34 templates for Finance and Weather, respectively. The remaining questions (212 and 196 questions for Finance and Weather) are used as test sets to evaluate the trained parsers. In this way, the train and test sets do not have any overlap even at the template-level.

From DQ, we generate new questions N_{DQ} using our approach. As the number of questions generated from each seed question is very large (140,325 for Finance), and they have similar syntactic structures, we include in N_{DQ} a maximum of 50 questions randomly selected from those generated from each seed question (this value is chosen from validation experiments).

	FINANCE		WEATHER	
DATASET	LAS	UAS	LAS	UAS
UD	72.31	80.19	69.87	76.27
UD+GQ	76.41	84.50	72.76	78.44
UD+DQ	80.47	86.10	77.30	82.24
UD+N_{DQ}	80.56	86.34	77.98	83.07
UD+GQ+DQ	81.99*	87.84†	80.44*	84.80†
UD+GQ+N_{DQ}	**82.43***	**88.54†**	**81.03***	**85.34†**

Table 4: SyntaxNet results with different training sets.

Models. We use two parsers with default parameters: Malt (Nivre et al., 2016a), as it is fast to train, and SyntaxNet, a SoAT neural model (Andor et al., 2016). We measure parser performance using UAS (Eisner, 1996) and LAS (Nivre et al., 2004) metrics, on the test portions of DQ_F and DQ_W.

Effectiveness. Tables 3 and 4 shows the results of Malt trained on different combinations of datasets. SyntaxNet displays similar trend, though with lower scores. Adding GQ to UD improves performance in both domains. Adding DQ leads to greater improvements for Finance, though GQ contains many more questions than DQ. This confirms the importance of domain-specific questions in the training data for specialized domains such as Finance. Adding generated questions N_{DQ} leads to comparable, if not better, performances over UD+GQ and UD+GQ+DQ for both domains, with small sizes (1,150 questions for Finance, 1,136 for Weather, on average), illustrating the effectiveness of our algorithm in generating labeled domain questions. UD+GQ+N_{DQ} and UD+GQ+DQ+N_{DQ} achieve the best performance: LAS gain of $4.9\% - 9\%$ over UD+GQ, and

Dataset	Finance	Weather
UD	64.54	64.56
UD+DQ	65.14	64.75
UD+DQ+N$_{\mathrm{DQ}}$	66.00	65.26

Table 5: LAS of Malt parser on GQ as test.

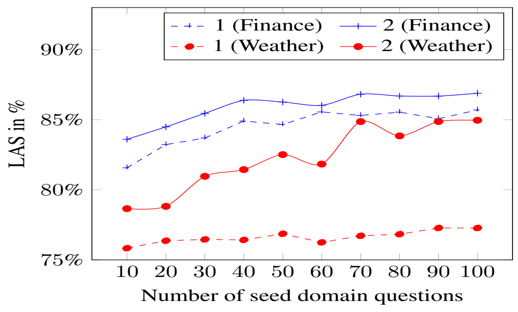


Figure 4: Malt performance trained with increasing sizes of DQ (1: UD+DQ, 2: UD+DQ+N$_{DQ}$).

$2.3\% - 3.6\%$ over UD+GQ+DQ. This shows that our framework generates effective labeled domain-specific questions which help improve parser performances when used for training them.

Table 5 shows LAS of Malt parser trained on domain-specific seed and generated questions, and evaluated on GQ. The improvements with the addition of DQ and N$_{\mathrm{DQ}}$ on general questions illustrate that our framework does not overfit to a specific domain; instead, the augmented training sets only facilitate an increase in the performances.

Effect of Schema Annotations. Schema annotation is a one-time effort and is proportional to the schema size; it required an average of 2 hours for the authors to annotate each schema. The domain understanding required for this can be acquired in a fairly small amount of time as the annotations are straight-forward, as opposed to the heavy linguistic expertise required to annotate questions with parse trees, which generally requires weeks or even months of effort to obtain a decently large training data. Moreover, large-scale human dependency tree annotations are error-prone, inconsistent and intensive, as annotators may tend to forget the many linguistic rules involved, and need to constantly ensure that the same rules are applied everywhere. Another sample of 100 questions, generated using the schema annotations, do not exhibit the anomalies listed in Section 2.2. Moreover, the inclusion of schema annotations leads to $0.4\% - 0.94\%$ gains in LAS, thus compensating the effort required to annotate the ontology.

Robustness to Size of Domain Seed. To study the performance variation with the size of training data, we randomly sample $1/3^{rd}$ templates as test (49 Finance and 55 Weather questions), and train the parser on questions from remaining templates with varying sizes. Figure 4 shows LAS averaged over 5 runs with Malt in two settings: UD+DQ and UD+DQ+N$_{\mathrm{DQ}}$. Our framework leads to significantly better performances in both domains, and even with only 10 seed questions, LAS improves by 2.03% (Finance) and 2.82% (Weather). We also note that in the Finance, adding 10 domain questions to UD leads to comparable performance to adding entire QuestionBank, and 30 questions are needed for achieve the same in Weather (UD+GQ in Table 3). As Finance is a specialized domain, N$_{\mathrm{DQ}}$ from even a small seed set have a higher effect compared to a more general domain like Weather.

4 Conclusions

We proposed a method to automatically generate labeled domain-specific questions from small seed set using domain knowledge, to compensate for the lack of training data. We introduced ontology annotations that enhance the naturalness of the automatically generated questions. Our approach resulted in a significant increase in the LAS of $2.3\% - 2.4\%$ over training with standard corpora and domain seed in two domains, and is robust to the seed set size. With sufficient labeled data, some of these heuristics could potentially be learned from the data. We believe our work paves way to develop domain-independent question parsing methods with very little or possibly no training data.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2016, pages 2442–2452, Berlin, Germany.

Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. 2010. Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In *Proceedings of the 7th International Conference on The Semantic Web: Research and Applications - Volume Part I*, ESWC'10, pages 106–120, Heraklion, Crete, Greece.

Jason M Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational* *linguistics-Volume 1*, COLING 1996, pages 340–345, Copenhagen, Denmark.

Thierry Hamon, Natalia Grabar, and Fleur Mougin. 2017. Querying biomedical linked data with natural language questions. *Semantic Web*, 8(4):581–599.

John Judge, Aoife Cahill, and Josef Van Genabith. 2006. QuestionBank: Creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL 2006, pages 497–504, Sydney.

Emilie Pasche Dina Vishnyakova Pascale Gaudet Amos Bairoch Julien Gobeill, Arnaud Gaudinat and Patrick Ruch. 2015. Deep question answering for protein annotation. *Database: The Journal of Biological Databases and Curation*, bav081.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, New York City, USA.

Joakim Nivre, Johan Hall, and Jens and Nilsson. 2016a. Malt Parser, v1.9, released June 4 2016. `http://www.maltparser.org/download.html`.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016b. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*, LREC 2016, pages 1659–1666, Portorož, Slovenia.

Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2010, pages 705–713, Cambridge, MA, USA.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, LREC 2014, pages 2897–2904, Reykjavik, Iceland.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, ACL IJCNLP 2015, pages 1332–1342, Beijing, China.

A Template Abstraction

We refer to `qmod` entities that have temporal values as `qtmp`.

Algorithm 1 Template Abstraction

```
Input: Question $Q$ and its syntactic parse tree $P_Q$.
Output: Template $\mathcal{T}_Q$ for $Q$.
1: procedure AbstractTemplate($Q$, $P_Q$)
2:     Identify the qsubj entity in $P_Q$.
3:     Identify all qmod entities attached to qsubj's head in $P_Q$.
4:     Recursively identify qmod entities for each of qmod.
5:     Among all qmod entities, mark those that are qtmp.
6:     Let $\mathcal{T}_Q$ be $P_Q$ with qsubj and qmod replaced with placeholders.
```

B Domain Schema

Fig. 5 shows an example schema for financial domain and a few data instances in a knowledge base $\mathcal{KB}$ conforming to this schema. The schema models a subset of the Finance domain, including people and companies with some of their related classes $\mathcal{R}$ such as address and financial metrics. It also models several relations of interest including a person's holding in a company, and a person's job assignment as an officer of a company. Ellipses represent *classes*; rectangles show class *properties* and their data types; For example, `title` is a property of class `AssignmentHistory` with instances 'CEO' and 'COO' in Finance domain.

We denote properties as `class:property` and their instances as `class:property='instance'`. For classes with a property `name`, we consider instances of `name` as instances of the class (e.g., 'Citigroup Inc' is an instance of class `Company`). Labeled arrows denote relations between classes. For example, in Fig. 5, `AssignmentHistory` is related to `Company` and `Person`, modeling the relationships between a person's assignment within a company; e.g., Neil Smit is the CEO of Citigroup.

C Template Filling

We detail the template replacement heuristics for each entity type here. If sufficient labeled data is available, some of these heuristics could potentially be learned from the data.

***Entity* `qsubj`** can be filled with either (1) a class property,[3] or (2) an instance of a class property of type string, provided that the property is not `name` and the instance is not a

[3]If the part-of-speech tags of the property and `qsubj` do not match, appropriate changes are made to the template.

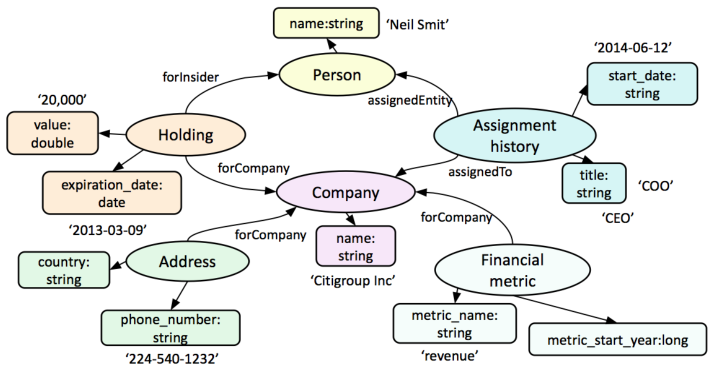


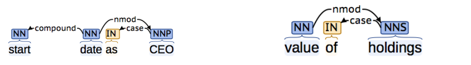
Figure 5: Domain schema for the finance domain, and example data instances. Classes, properties (their data types), and relations between classes are shown as ovals, rectangles and labeled arrows. Data instances are shown next to corresponding properties in quotes.

proper noun. For example, `qsubj` in $\mathcal{T}_{Q_1}$ can be filled with `AssignmentHistory:title` or `AssignmentHistory:'CEO'` resulting in *What is the title of ..?* and *Who is the CEO of ..?* The proper noun restriction avoids generating meaningless questions such as *What is the 224-540-1232 of ..?* The restriction on `name` and string type avoids questions such as *What is the Citigroup Inc ..?*, and *What is the 20,000 of ..?*

***Entity* `qmod`** is filled based on the QENT it modifies. If QENT is a class property, `qmod` is filled with an instance of a related class. In $\mathcal{T}_{Q_1}$, when `qsubj` is filled with `AssignmentHistory:title`, the $\texttt{qmod}_1$ slot can be filled with `Person:name='Neil Smit'`, resulting in *What are the titles of Neil Smit ..?* If QENT is a class instance, it is changed to the name of the class, and `qmod` is filled with a V_p of property p of a related class. For example, in $\mathcal{T}_{Q_2}$, if $\texttt{qmod}_1$ is filled with `Company:name='Citigroup Inc'`, it will be changed to 'company', and $\texttt{qmod}_2$ can be filled with V_p *value of 20,000* of `Holding:value` as `Holding` is related to `Company`.

Relative pronouns for* `qmod` *expansion. To generate more complex questions, we introduce one-level nesting on an already filled `qmod` by expanding it using relative pronoun *whose*. If `qmod` is not a V_p of any property p, we replace it with its corresponding class name, and attach a relative modifier clause using V_p of a property p of one of its related classes. In $\mathcal{T}_{Q_1}$, when $\texttt{qmod}_1$ is filled with `Person:name='Neil Smit'`, the relative pronoun expansion of $\texttt{qmod}_1$ is *person whose value is 20,000* using related class `Holding` of `Person`.

***Entity* `qtmp`** is always retained as temporal values do not change the syntactic context of questions. In Fig. 2f, see Q_2^N generated from $\mathcal{T}_{Q_2}$.



(a) `AssigH:start_date` (b) `Holding:value`

Figure 6: Templates for non-heading property (a) and class-dependency (b).

D Heading Property Heuristics

While heading properties and their instances can be used to fill QENT entities independently, we adopt the following rules for non-heading properties:

- For classes with a single heading property with an instance (e.g. `AssignmentHistory:title` with 'CEO'), the instance is used along with a non-heading property to query it using prepositional connectives which are also annotated along with heading properties (`AssignmentHistory:start_date` *as* CEO). The QENT replacement is automatically constructed by attaching the heading property's instance as `nmod` using prepositional connective to the parse tree of non-heading property (Fig. 6a).
- For classes with more than one heading property, only they can be queried (`Holding:value`, `Holding:expiration_date`). E.g., *what is the value of ..?*, *what is the expiration date of ..?*