Question Answering over Knowledge Bases with Continuous Learning

---

A dissertation submitted towards the degree
Doctor Engineering (Dr.-Ing)
of the Faculty of Mathematics and Computer Science
of Saarland University

by

Abdalghani Abujabal

Saarbrücken, 2019

Day of Colloquium	12 April 2019
Dean of the Faculty	Univ.-Prof. Dr. Sebastian Hack
Chair of the Committee	Prof. Dr. Vera Demberg
Reporters First Reviewer	Prof. Dr.-Ing. Gerhard Weikum
Second Reviewer	Prof. Dr. Jimmy Lin
Third Reviewer	Prof. Dr.-Ing. Klaus Berberich
Academic Assistant	Rishiraj Saha Roy, Ph.D.

Abstract

Answering complex natural language questions with crisp answers is crucial towards satisfying the information needs of advanced users. With the rapid growth of knowledge bases (KBs) such as Yago and Freebase, this goal has become attainable by translating questions into formal queries like SPARQL queries. Such queries can then be evaluated over knowledge bases to retrieve crisp answers. To this end, three research issues arise: (i) how to develop methods that are robust to lexical and syntactic variations in questions and can handle complex questions, (ii) how to design and curate datasets to advance research in question answering, and (iii) how to efficiently identify named entities in questions. In this dissertation, we make the following five contributions in the areas of question answering (QA) and named entity recognition (NER). For issue (i), we make the following contributions:

We present QUINT, an approach for answering natural language questions over knowledge bases using automatically learned templates. Templates are an important asset for QA over KBs, simplifying the semantic parsing of input questions and generating formal queries for interpretable answers. QUINT is capable of answering both simple and compositional questions.

We introduce NEQA, a framework for continuous learning for QA over KBs. NEQA starts with a small seed of training examples in the form of question-answer pairs, and improves its performance over time. NEQA combines both syntax, through template-based answering, and semantics, via a semantic similarity function. Moreover, it adapts to the language used after deployment by periodically retraining its underlying models.

For issues (i) and (ii), we present TEQUILA, a framework for answering complex questions with explicit and implicit temporal conditions over KBs. TEQUILA is built on a rule-based framework that detects and decomposes temporal questions into simpler sub-questions that can be answered by standard KB-QA systems. TEQUILA reconciles the results of sub-questions into final answers. TEQUILA is accompanied with a dataset called TempQuestions, which consists of 1,271 temporal questions with gold-standard answers over Freebase. This collection is derived by judiciously selecting time-related questions from

existing QA datasets.

For issue (ii), we publish ComQA, a large-scale manually-curated dataset for QA. ComQA contains questions that represent real information needs and exhibit a wide range of difficulties such as the need for temporal reasoning, comparison, and compositionality. ComQA contains paraphrase clusters of semantically-equivalent questions that can be exploited by QA systems. We harness a combination of community question-answering platforms and crowdsourcing to construct the ComQA dataset.

For issue (iii), we introduce a neural network model based on subword units for named entity recognition. The model learns word representations using a combination of characters, bytes and phonemes. While achieving comparable performance with word-level based models, our model has an order-of-magnitude smaller vocabulary size and lower memory requirements, and it handles out-of-vocabulary words.

Kurzfassung

Die Beantwortung komplexer natürlich-sprachlicher Fragen mit treffenden Antworten ist ein wichtiger Schritt Richtung Informationsbedürfnisse fortgeschrittener Benutzer zu erfüllen. Dieses Ziel ist durch das rapide Anwachsen von Wissensbanken (WB), wie Yago und Freebase, in erreichbare Nähe gerückt, indem Fragen in formale Anfrage, wie zum Beispiel SPARQL Anfragen, übersetzt werden. Solche Anfragen werden dann mittels einer Wissensbank ausgewertet um Antworten zu erhalten. In diesem Zusammenhang ergeben sich drei Forschungsprobleme: (i) wie können Methoden entwickelt werden, die robust in Bezug auf lexikalische und syntaktische Veränderungen von Fragen sind und komplexe Fragen handhaben können, (ii) wie müssen Datensätze gestaltet und kuratiert werden um Forschungsanstrengungen zur automatischen Fragebeantwortung voranzutreiben, und (iii) wie können benannte Entitäten effizient in Fragen erkannt werden. In dieser Dissertation leisten wir folgende fünf Beiträge im Bereich der automatischen Fragebeantwortung und der Erkennung benannter Entitäten. Zur Problemstellung (i) leisten wir folgende Beiträge:

Wir präsentieren ein Verfahren, genannt QUINT, zur Beantwortung natürlich-sprachlicher Fragen über Wissensbanken durch automatisch gelernte Mustervorlagen. Diese Mustervorlagen sind wichtige Mittel zur automatischen Fragebeantwortung durch Wissensbanken, da sie das semantische Parsen von Eingabefragen vereinfachen, sowie formale Anfragen fr interpretierbare Antworten generieren. QUINT ist dabei in der Lage einfach als auch zusammengesetzte Fragen zu beantworten.

Wir stellen NEQA, ein Framework fürs kontinuierliche Lernen für die automatische Fragebeantwortung über Wissensbanken vor. NEQA kombiniert Syntax durch vorlagenbasiertes Antworten mit Semantik mittels semantischer Ähnlichkeitsfunktionen. Darüber hinaus adaptiert es die benutzte Sprache zur Laufzeit durch periodisches Neuerlernen des zugrundeliegenden Models.

Für Problemstellungen (i) und (ii) präsentieren wir TEQUILA, welches ein Rahmenwerk zur Beantwortung komplexer Fragen mit expliziten und impliziten temporalen Bedingungen auf Wissensbanken ist. TEQUILA basiert auf einem regelbasierten Rahmenwerk, das temporale Fragen entdeckt und in einfachere

Unterfragen zerlegt, die dann mittels üblicher wissensbasierter Frageantwortsysteme beantwortet werden können. TEQUILA gleicht die Ergebnisse von Unterfragen ab um finale Antworten zu erstellen. TEQUILA geht mit einem Datensatz, genannt TempQuestions, einher. Dieser besteht aus 1271 temporaler Fragen mit Goldstandardantworten aus Freebase. Zur Sammlung der Daten wurden zeitrelevante Fragen aus einem bestehenden Frage-Antwortdatensatz umsichtig ausgewählt.

Für Problemstellung (ii) veröffentlichen wir einen groß angelegten und manuell editierten Datensatz zur Fragebeantwortung, genannt ComQA. ComQA enthält Fragen, die wirkliche Informationsbedürfnisse repräsentieren und ein breites Spektrum an Schwierigkeiten aufweisen, wie z.B. temporales Schlussfolgern, Vergleiche, und eine kompositorische Struktur haben. ComQA enthält Paraphrasgruppen von semantisch gleichen Fragen, die durch Frageantwortsysteme ausgenutzt werden können. Zur Erstellung des Datensatzes nutzen wir eine Kombination aus Frageantwortplattformen und Crowdsourcing.

Für Problemstellung (iii) stellen wir ein neurales Netzwerkmodell vor, welches auf Unterworteinheiten zur Erkennung benannter Entitäten basiert. Das Modell lernt Wortrepräsentationen, indem es eine Kombination aus Buchstaben, Bytes und Phonemen benutzt. Bei gleichbleibender Performanz im Vergleich zu anderen wortlevelbasierten Modellen, hat das Modell einen um eine Größenordnung kleineren Wortschatz, geringere Speicheranforderungen, und es ist in der Lage Wörter zu verarbeiten, welche nicht im Vokabular enthalten sind.

To my mother, Basema.

Acknowledgments

I am gratefully thankful to Gerhard Weikum for his guidance, mentoring and unlimited support over the past couple of years. This dissertation would not have seen the light of day without his contribution.

I wish to thank my friend and coauthor Mohamed Yahya, without whom my PhD journey would have taken a different trajectory. He was patient and always ready to hear me and clarify any points of confusion I have.

I am also thankful to Judith Gaspers and Fabian Triefenbach for taking me on as an intern at Amazon in Aachen. I learned a great deal from both. I wish to thank Prof. Dr. Jimmy Lin and Prof. Dr.-Ing. Klaus Berberich for agreeing to review my dissertation and for their feedback. I thank Dr. Patrick Ernst for translating the abstract of this dissertation to German.

Thanks to my colleagues Mohamed Gad-Elrab and Rishiraj Saha Roy for the great time and the lengthy discussions we had together.

I thank my close circle of friends: Nicole Hartmann for making life easier, teaching me German and unhesitantly helping me in any matter I encountered, and Fidaa Abed for the immense dose of wisdom, which was decisive in many life decisions.

I am gratefully thankful to my parents Basema and Hisham for their unconditional love and support. I am indebted to them beyond repayment. I thank my siblings, Eyad, Nilly, Deaa and Riham, for their encouragement.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Contributions 3

1.3 Organization of the Dissertation 8

2 Background **9**

2.1 Knowledge Bases 9

2.2 Lexica 14

2.3 Named Entity Recognition 15

3 Automated Template Generation for Question Answering over Knowledge Bases **17**

3.1 Introduction 17

3.2 Related Work 21

3.3 Template Generation 24

3.4 Simple Question Answering with Templates 29

3.5 Compositional Question Answering with Templates 32

3.6 Experimental Evaluation 34

4 Never-Ending Learning for Open-Domain Question Answering over Knowledge Bases **41**

4.1 Introduction 41

4.2 Related Work 45

4.3 The NEQA Framework 47

4.4 Experimental Evaluation 52

5 Temporal Question Answering over Knowledge Bases **63**

5.1 Introduction 63

5.2 Related Work 65

5.3 Setup 66

5.4 The TEQUILA Framework 70

5.5 Experimental Evaluation 74

6 ComQA: A Community-sourced Dataset for Complex Factoid Question Answering with Paraphrase Clusters 81
6.1 Introduction . . . 81
6.2 Related Work . . . 84
6.3 Setup . . . 85
6.4 Dataset Collection . . . 88
6.5 Dataset Analysis . . . 91
6.6 Experimental Evaluation . . . 93

7 Neural Named Entity Recognition from Subword Units 99
7.1 Introduction . . . 99
7.2 Related Work . . . 100
7.3 Model . . . 101
7.4 Experimental Evaluation . . . 106

8 Conclusion 111
8.1 Summary . . . 111
8.2 Outlook . . . 112

List of Figures 114

List of Tables 116

Bibliography 118

1 Introduction

1.1 Motivation

Returning crisp answers to users in response to their natural language questions is crucial to satisfying the information needs of users with minimal effort on their part. For example, the answer set for *"Which actors starring in The Departed were born in Cambridge, MA?"* is { `MattDamon`, `MarkWahlberg` }. This goal of crisply answering questions becomes a necessity with the wide availability of smart voice-controlled assistants such as Amazon Echo, Google Home, Siri and Cortana. Moreover, such modalities make it more natural to ask verbose, compositional and complex questions. To this end, models need to understand the meaning of questions by translating them into semantic representations that can be evaluated over a repository of knowledge to retrieve crisp answers.

Traditionally, users receive a list of documents in response to their keyword queries over a typical search engine. For example, the user might receive several documents about The Departed movie and Cambridge, MA as a result for the above question. Going through the retrieved documents to spot relevant information with respect to the user's query is a time-consuming and sometimes hopeless task. Passage retrieval and answer ranking methods have been proposed to distill focused answers from retrieved documents. However, with the increasing complexity of questions, the need to combine different answering sources for cross-topical questions (e.g., evidence from multiple documents) and the need to reason over intermediate results, answers can not be found directly in a single document. Moreover, joining and reasoning over evidence from different, seemingly unrelated documents can be challenging. For instance, the above example requires retrieving the actors of The Departed, retrieving actors who were born in Cambridge, MA, and finally, intersecting the two sets.

Knowledge bases (KBs) are a key enabler and a first step towards satisfying complex information needs and returning crisp answers. KBs store world knowledge in the form of subject-predicate-object (SPO) triples, where each triple represents a real-world fact. For example,

```
TheDeparted hasActor MattDamon
```

and

```
MattDamon placeOfBirth CambridgeMassachusetts
```

KB predicates express crisp relationships among entities. Freebase (Bollacker et al., 2008), Yago (Suchanek et al., 2007), DBpedia (Auer et al., 2007) and, recently, Wikidata (Vrandecic and Krötzsch, 2014) are examples of large-scale knowledge bases with billions of facts, millions of entities and thousands of predicates. Formal query languages like SPARQL are required to access such knowledge, which poses a challenge for the average user. In addition to learning how to construct complex queries using the terminology of a specific formal language, users have to familiarize themselves with the underlying KB.

Question answering over knowledge bases (KB-QA) is a more recent trend, which aims to answer natural language questions over large knowledge bases, and hence, shields users from all the underlying complexity. KB-QA systems take as input questions such as: *"Which actors starring in The Departed were born in Cambridge, MA?"* and translate them into structured queries, in a formal language like SPARQL:

```
SELECT ?x WHERE {
   ?x type movieActor .
   TheDeparted hasActor ?x .
   ?x placeOfBirth CambridgeMassachusetts
}
```

and execute the queries to retrieve answers from the KB (Unger et al., 2012; Yahya et al., 2012). Translating questions into formal queries is a challenging problem due to the countless lexical and syntactic variations users use in formulating questions and the vocabulary mismatch between phrases in the input question and entities, classes and predicates in the KB. For example, mapping '*Cambridge, MA*' to the uniquely identified entity, '*actor*' to the KB type `movieActor` and '*starring*' to the KB predicate `hasActor`.

The initial step towards translating questions into formal queries is to identify mentions of named entities in questions like movies and locations. For instance, the question above contains two mentions: a `movie`, '*The Departed*' and a `location`, '*Cambridge, MA*'. However, while current methods focus only on accuracy, designing efficient entity recognizers is also needed, particularly, in situations where systems operate under certain constraints like memory constraints; in handheld or voice-controlled devices.

To this end, three main research questions arise:

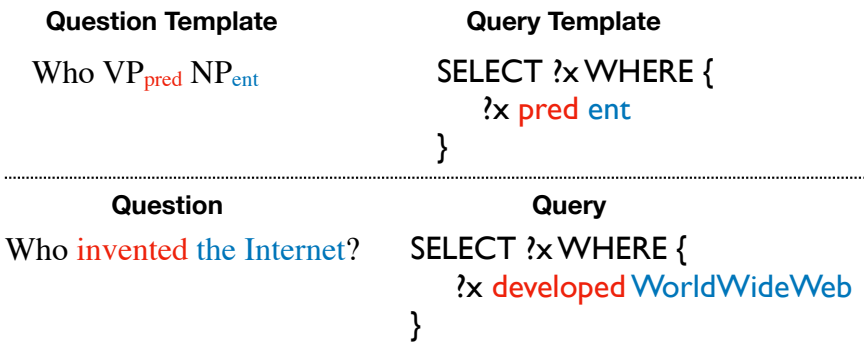


Figure 1.1: An example of a hand-crafted role-aligned question-query template (top). Shared *pred* and *ent* annotations indicate an alignment between a phrase in the question template and a KB semantic item in the corresponding query template. An instantiation of the template is shown at the bottom.

i. how to develop methods that are robust to lexical and syntactic variations in questions and can handle complex questions,

ii. how to design and curate datasets for QA with complex questions that express real information needs to advance research in QA, and

iii. how to efficiently identify mentions of named entities in text.

1.2 Contributions

In this dissertation, we present three approaches for KB-QA: QUINT, NEQA and TEQUILA to address the research question (i). Moreover, to address (ii), we introduce a large-scale dataset, ComQA, for evaluating question answering. Finally, for issue (iii), we present a neural network model based on subword units for named entity recognition.

1.2.1 Question Answering over Knowledge Bases

- **QUINT:** Given the numerous lexical and syntactic variations that users use in formulating natural language questions, a data-driven approach for learning the mapping between syntactic structures on the question side and semantic ones on the KB side is desired. Relying on hand-crafted rules or templates to map natural language questions to formal queries is a common characteristic of many state-of-the-art approaches to KB-QA (Bast

and Haussmann, 2015; Berant et al., 2013; Fader et al., 2013). However, such methods suffer from low coverage. The problem is further exacerbated when dealing with complex and compositional questions. Figure 1.1 shows an example of a hand-crafted question-query template (top) and an instantiation of the template (bottom).

As a remedy, we introduce QUINT, a data-driven approach that automatically learns pairs of role-aligned question-query templates using questions paired with answer sets over knowledge bases. Templates play an important role in KB-QA, guiding the mapping of question constituents onto query components. A benefit of templates is that the mappings to the KB are traceable and can be leveraged to generate explanations for the user to understand why she receives specific answers. QUINT harnesses language compositionality to answer *compositional* questions for which no matching templates could be found like *"actors starring in The Departed who were born in Cambridge, MA"*. In doing so, QUINT uses the templates learned from simple questions to capture sub-questions within the compositional question (*"actors starring in The Departed"* and *"actors were born in Cambridge, MA"* for the above question), and answers them independently. Results of sub-questions are then intersected to deliver the final answer to the full question.

- **NEQA:** QA methods rely on a clear separation between an offline training phase, where a model is either learned or manually crafted, and an online phase where this model is deployed to answer users' questions (Bast and Haussmann, 2015; Berant et al., 2013; Fader et al., 2013; Yahya et al., 2012; Yih et al., 2015). This necessitates the need to access reasonably large training sets with sufficient syntactic and lexical coverage representative of the kinds of questions users pose. These are expensive to construct, particularly, for new domains. Furthermore, methods are limited to the language learned at training time, therefore, they fail on questions from domains not observed previously.

  Our second contribution is NEQA, a *continuous-learning paradigm for KB-QA*. NEQA is initialized with a *small* training set and improves its performance over time. Moreover, it adapts to the language used after deployment by periodically retraining its underlying models. NEQA combines both syntax, through template-based answering, and semantics, through answering via a semantic similarity function, when templates fail to do so.

  NEQA first uses its template bank, learned from a small seed using QUINT,

to answer incoming questions. For example, it uses a template generated from *"Which film awards was Brad Pitt nominated for?"* to answer the syntactically isomorphic question *"Which president was Lincoln succeeded by?"*. Given a new question for which no matching templates were found, say, *"What are the film award nominations that Brad Pitt received?"*, NEQA uses a semantic similarity function to find a correctly answered and semantically-similar question from its history. By harnessing light-weight user feedback on the answers retrieved by the similarity function, NEQA learns new templates, which are then added to the collection of templates learned thus far, improving the ability of the system to directly answer questions using templates.

- **TEQUILA:** Time-related information needs occur very often in web search (Metzler et al., 2009), with explicit or implicit temporal conditions, for example, *"Which teams did Neymar play for before joining PSG?"*. However, handling such complex questions poses a challenge to existing KB-QA systems, as they are geared towards simple questions without any such constraints.

  Our solution, called TEQUILA, is built on a rule-based framework that encompasses four stages of processing: (i) detecting temporal questions (e.g., using temporal prepositions like '*before*' in the above example), (ii) decomposing questions and rewriting sub-questions, if needed, (e.g., decomposing the above question into *"Which teams did Neymar play for"* and *"When did Neymar join PSG?"*) (iii) retrieving candidate answers for sub-questions ({`ParisSaintGermain, FCBarcelona, SantosFC`} and {`2017`} for sub-questions 1 and 2, respectively), and (iv) temporal reasoning to combine and reconcile the results of the previous stage into final answers ({`FCBarcelona, SantosFC`} for the above question). For stage (iii), we leverage existing KB-QA systems (e.g., QUINT or NEQA). Along with TEQUILA, we introduce TempQuestions, a dataset of 1,271 time-related questions with answers over Freebase. The questions are chosen such that many of them require a combination of evaluating sub-questions and reasoning over sub-results (results of the sub-questions). This collection is derived by judiciously selecting time-related questions from existing QA datasets.

- **ComQA:** To advance experimental research on QA, it is important to have access to benchmarks that reflect real user information needs and cover question phenomena users are interested in (e.g., compositionality,

temporal reasoning, comparatives and superlatives, etc.). Moreover, such benchmarks should capture the wide lexical and syntactic variety in expressing these information needs. The benchmarks should be large enough to cover these phenomena and facilitate the use of data-hungry machine learning methods, and they should facilitate providing answers grounded in widely adopted semantic resources like Wikipedia. Existing large datasets with complex questions are mostly composed of synthetically generated questions (Bordes et al., 2015; Su et al., 2016; Talmor and Berant, 2018; Trivedi et al., 2017). Moreover, existing benchmarks are tied to specific answering resources such as a specific knowledge base or a specific textual corpus. Recent research has shown that significant improvements can be obtained from combining various resources for answering (Savenkov and Agichtein, 2016; Sun et al., 2018; Xu et al., 2016b).

We present a *large* dataset for QA, called ComQA, which contains 11,214 *real* user questions with *various challenging phenomena* such as the need for temporal reasoning (e.g., *"Who was Britain's leader during WW1?"*), comparison (e.g., *"Who was the first US president to serve 2 terms?"*, *"What is the population of the largest city in Egypt?"*, *"What was the name of Elvis's first movie?"*), compositionality (e.g., *"Who were the parents of the 13th president of the US?"*), and questions with empty answer sets (e.g., *"Who was the first human being on Mars?"*). Through a large crowdsourcing effort, questions in ComQA are grouped into 4,834 *paraphrase clusters* that express the same information need. Paraphrases are valuable for learning reformulation patterns for questions with equivalent information needs. Each cluster is annotated with its answer(s). Building on the wide adoption of Wikipedia, ComQA answers come in the form of Wikipedia entities wherever possible (e.g., `https://en.wikipedia.org/wiki/Love_Me_Tender_(film)`, the first movie of Elvis). Wherever the answers are temporal or measurable quantities, TIMEX3 and the International System of Units are used for normalization.

1.2.2 Neural Named Entity Recognition

Named Entity Recognition (NER) is an important task in language technology applications. For example, the following utterance *"play we are the champions by queen"* contains two mentions, a `song`, '*we are the champions*' and an `artist`, '*queen*'. Recently, several neural models for NER have been proposed. However, these models mostly rely on word-level representations, which suffer from

three shortcomings: (1) the vocabulary size is large, yielding a large number of parameters, and hence, large memory requirements and training time, which is problematic if large amounts of data are available, (2) out-of-vocabulary (OOV) words can be problematic, and (3) models cannot learn subword representations, which can improve performance by taking advantage of morphology.

Our last contribution in this dissertation is a neural network for NER relying on *subword units*, namely *characters*, *phonemes* and *bytes*. For each word in an utterance, we learn representations from each of the three subword units. The character-level unit looks at the characters of each word, while the phoneme-level unit treats a word as a sequence of phonemes, using lexica that map a given word into its corresponding phoneme sequence. The byte-level unit reads a word as bytes, where we use the variable length UTF-8 encoding. A major advantage of subword-based models is the small vocabulary size which reduces memory requirements and training time of models, which is particularly relevant for large-scale applications and embedded devices. In addition, the use of subword-units improves modelling out-of-vocabulary words and supports learning of morphological features.

1.2.3 Publications

This dissertation is based on results presented at the following conferences:

- Abujabal, A., Yahya, M., Riedewald, M., and Weikum, G. (2017b). Automated Template Generation for Question Answering over Knowledge Graphs. In Proceedings of WWW 2017, pages 1191-1200.

- Abujabal, A., Roy, R. S., Yahya, M., and Weikum, G. (2017a). QUINT: Interpretable Question Answering over Knowledge Bases. In Proceedings of EMNLP 2017, pages 61-66.

- Abujabal, A., Roy, R. S., Yahya, M., and Weikum, G. (2018). Never-Ending Learning for Open-Domain Question Answering over Knowledge Bases. In Proceedings of WWW 2018, pages 1053-1062.

- Jia, Z., Abujabal, A., Roy, R. S., Strötgen, J., and Weikum, G. (2018a). TEQUILA: Temporal Question Answering over Knowledge Bases. In Proceedings of CIKM 2018, pages 1807-1810.

- Jia, Z., Abujabal, A., Roy, R. S., Strötgen, J., and Weikum, G. (2018b).

Tempquestions: A Benchmark for Temporal Question Answering. In Companion of WWW 2018, pages 1057-1062.

- Abujabal, A., Roy, R. S., Yahya, M., and Weikum, G. (2019). ComQA: A Community-sourced Dataset for Complex Factoid Question Answering with Paraphrase Clusters. In Proceedings of NAACL 2019

work on NER is available on arXiv:

- Abujabal, A. and Gaspers, J. (2018). Neural Named Entity Recognition from Subword Units. CoRR, abs/1808.07364.

1.3 Organization of the Dissertation

The thesis is organized as follows. In Chapter 2 we introduce knowledge bases, lexica of natural language phrases and KB semantic items, and the concept of named entity recognition. Chapter 3 presents our first contribution, QUINT, our approach for template-based question answering over knowledge bases, followed by NEQA in Chapter 4, which extends QUINT by continuously learning new templates on-the-fly by harnessing a semantic similarity function and light-weight non-expert user feedback on answers. In Chapter 5, we present TEQUILA, our framework for addressing complex temporal questions with implicit and explicit temporal conditions, accompanied with the TempQuestions dataset. ComQA, a large-scale dataset for complex KB-QA is introduced in Chapter 6. Chapter 7 presents our neural model for named entity recognition. Finally, Chapter 8 concludes this dissertation and presents possible directions for future work.

2 Background

This chapter introduces the concepts needed for this dissertation. In Section 2.1 we present knowledge bases and how semantic items (entities, classes, and predicates) are modeled. We then introduce Freebase, a concrete knowledge base that we use in the rest of this dissertation. Finally, we explain how such knowledge can be queried. Section 2.2 gives an overview of how to construct lexica that bridge the gap between how natural language questions are formulated and how semantic items in KBs are represented. Finally, in Section 2.3 we explain the task of named entity recognition.

2.1 Knowledge Bases

A knowledge base (KB) can be seen as a massive table with three columns: subject, predicate and object. Each entry in this table forms a fact $f \in F$ like:

`AlbertEinstein placeOfBirth Ulm`

where `AlbertEinstein` corresponds to the subject of the fact, `placeOfBirth` is the predicate, and `Ulm` corresponds to the object. This model for representing knowledge is called the *subject-predicate-object (SPO)* model or *triples* model.

Subjects are entities E (e.g., `AlbertEinstein`) or classes C (e.g., `person`). Objects can be entities, classes or literal values L (e.g., dates). Predicates P represent relationships between entities (e.g., `AlbertEinstein placeOfBirth Ulm`), classes (e.g., `physicist subclassOf person`), a pair of an entity and a class (e.g., `AlbertEinstein type physicist`) or a pair of an entity and a literal value (`AlbertEinstein bornOn 14.03.1879`). Entities, predicates, and classes are collectively called semantic items.

Definition 2.1 (Semantic Item)**:** A semantic item s in a knowledge base KB is an element of the set $S = E \cup C \cup P$.

Literals do not serve as subjects of facts in a KB since they express constant values that are not described by other facts in the KB. The purpose of literals in a KB is to describe semantic items, hence, they appear as objects in a fact. Literal values include dates, numbers and strings.

A fact f can be now defined as:

Definition 2.2 (Fact)**:** Given a KB with semantic items E, C and P and literal values L, a fact $f \in F$ is a triple $f \in (E \cup C) \times P \times (E \cup C \cup L)$.

Now, we can define a knowledge base as:

Definition 2.3 (Knowledge Base)**:** A knowledge base KB is a set of facts F.

Figure 2.1(a) shows a fragment of a knowledge base in a tabular form. A knowledge base can also be seen as a massive graph whose nodes correspond to entities, classes or literal values. An edge in this graph represents a relationship between a pair of entities or classes, or a relationship between an entity and a class or between an entity and a literal value. Figure 2.1(b) shows a graphical representation of a KB fragment.

2.1.1 Semantic Items

Entities

Entities are the main kinds of semantic items of a KB. Indeed, knowledge bases were designed to describe entities by connecting them to other entities, classes or literal values through predicates. Note that a clear definition of an entity is controversial (Smith, 2008), however, we follow the definition of Suchanek (2008), which meets our needs: "any abstract or concrete thing that is uniquely identifiable is an entity". Example entities are: `AlbertEinstein`, `Ulm`, `NobelPrize`.

Classes

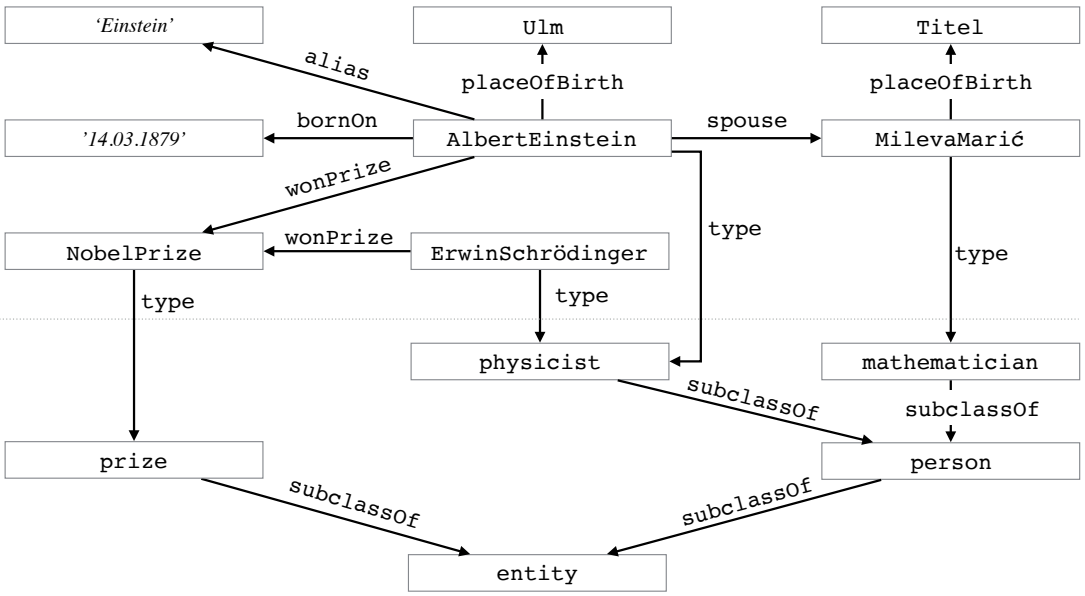
Entities sharing common characteristics are grouped into classes (also called types). A class c is a named set of entities in a KB i.e., $c \subseteq E$. In Figure 2.1 both `AlbertEinstein` and `ErwinSchrödinger` belong to the class of `physicist`, while the entity `MilevaMarić` is a member of the `mathematician` class. In total, we have five distinct classes in Figure 2.1, namely `physicist`, `mathematician`, `person`, `prize`, and `entity`. Typically, entities are connected to the classes they belong to using the `type` predicate. The `subclassOf` predicate connects two classes and defines hyponymy/hypernymy relations. For example, the class `physicist` is a sub-class of `person`, which in turn is a sub-class of `entity`. Using `subclassOf` relation among classes, we can generate the *type taxonomy* of a KB: a directed acyclic graph whose root is the type `entity`.

Predicates

Predicates represent relations between entities or attributes of entities. Predicates have two arguments: a subject and an object, and hence, are called binary

#	Subject	Predicate	Object
1	AlbertEinstein	placeOfBirth	Ulm
2	AlbertEinstein	spouse	MilevaMarić
3	MilevaMarić	placeOfBirth	Titel
4	AlbertEinstein	wonPrize	NobelPrize
5	AlbertEinstein	bornOn	'*14.03-1879*'
6	AlbertEinstein	alias	'*Einstein*'
7	MilevaMarić	type	mathematician
8	AlbertEinstein	type	physicist
9	ErwinSchrödinger	type	physicist
10	NobelPrize	type	prize
11	Ulm	type	city
12	mathematician	subclassOf	person
13	physicist	subclassOf	person
14	person	subclassOf	entity

(a)



(b)

Figure 2.1: A fragment of a knowledge base in (a) tabular and (b) graph form.

relations. When both arguments are entities, the predicate corresponds to a relation between them, however, when the object is a literal value, the predicate represents an attribute of the subject entity. Each predicate in a KB has a *type signature*, which defines the types of both the subject and object. For example, the type signature of `placeOfBirth` is (`person`, `city`).

To accommodate higher-arity relations i.e., n-ary relations, knowledge bases such as Yago attach SPO triples with meta or provenance attributes e.g., spatio-temporal attributes through *reification*. The idea is to decompose the higher-arity relation into a set of binary relations. It works as follows: the main fact is given a unique identifier, which is then used to state more information about the fact. This allows us to create facts (triples) about facts. For example:

```
AlbertEinstein spouse MilevaMarić
```

is deemed the main fact and is given the identifier *fact1*, which is then used to add information about the fact:

```
fact1 startDate 1903
fact1 endDate 1919
fact1 location Bern
```

Note that with reification there has to be a main fact that can be described with further meta information, which is not always the case. As a solution, in Freebase, higher-arity facts are encoded using so-called *Compound Value Types* (CVTs), which we describe in the next section.

2.1.2 Freebase

Freebase (Bollacker et al., 2008) is a large-scale knowledge base that was originally released by Metaweb Technologies, Inc. in 2007 and later acquired by Google in 2010. Freebase is a collaboratively edited knowledge base, with billions of facts, millions of distinct entities and thousands of distinct predicates. Whenever applicable, entities in Freebase are linked to their corresponding Wikipedia articles.

Freebase follows the triples model to represent its knowledge and extends it to accommodate higher-arity relations using *Compound Value Types* (CVTs). Given a higher-arity relationship like the marriage relation with multiple actors, either of them is chosen as the subject of a binary relation whose object is a CVT with a unique identifier e.g., cvt1. The CVT is then connected to all other actors (entities) or literal values using predicates. For example, the marriage relationship between `AlbertEinstein` and `MilevaMarić` is represented as follows:

```
AlbertEinstein marriage cvt1
  cvt1 spouse MilevaMarić
    cvt1 startDate 1903
     cvt1 endDate 1919
     cvt1 location Bern
```

2.1.3 Querying Knowledge Bases

Given the volume of knowledge bases with billions of facts and the complexity of constructing correct formal queries, it is very hard for a non-expert user to directly query knowledge bases to satisfy her information needs. However, users are quite familiar with search engines where, given a *keyword* query, a list of relevant documents is returned. In *entity search*, work has utilized the familiarity of users with keyword querying to return a set of entities from the KB in response to a keyword query (Balog et al., 2010; Blanco et al., 2011; Joshi et al., 2014; Tran et al., 2007, 2009). For example, in response to *"vice president barack obama"*, the entity `JoeBiden` is returned.

Keyword queries are "telegraphic" in nature (Joshi et al., 2014), and hence, suffer from inherent limited expressiveness. This potentially restricts the type of information needs that users can express to simple ones. To overcome this limitation, natural language questions were adopted as a means to provide users with an easy access to knowledge bases. Natural language questions allow users to express complex information needs in a more intuitive way.

Triple pattern queries

On the knowledge base side we use graph-matching queries based on SPARQL triple patterns. A triple pattern is a fact with one or more of its components replaced by variables (e.g., `?x placeOfBirth CambridgeMassachusetts`). A formal query q is a set of triple patterns, for example:

```
SELECT ?x WHERE {
   ?x type movieActor .
   TheDeparted hasActor ?x .
   ?x placeOfBirth CambridgeMassachusetts
}
```

The variable `?x` is designated as the *projection variable*. An answer a to a query q over a KB is a set of entities which is obtained by mapping variables of q to KB items where the projection variable maps to a. For example, the answer to the above query is { `MattDamon`, `MarkWahlberg` }.

Phrase	Semantic Item	w	Phrase	Semantic Item	w
'*married to*'	`marriage.spouse`	0.7	'*role*'	`cast.character`	0.6
'*married in*'	`marriage.location`	0.6	'*actress*'	`actress`	0.7
'*play on*'	`cast.actor`	0.6	'*actress*'	`person`	0.3

Table 2.1: Fragment of our lexica: $\mathcal{L}_P$ and $\mathcal{L}_C$. Semantic items are from Freebase.

2.2 Lexica

To connect the question vocabulary to semantic items in the knowledge base, we require a lexicon $\mathcal{L}$ that relates natural language tokens/phrases to KB entities, predicates and classes. Our lexicon consists of a predicate lexicon $\mathcal{L}_P$ and a class lexicon $\mathcal{L}_C$. For identifying mentions of entities we rely on off-the-shelf named entity recognition systems. Hakimov et al. (2015) addressed the lexical gap between the vocabulary used by users and the knowledge base semantic items and showed that when high quality lexicons are used, the performance of QA systems substantially improves. We construct $\mathcal{L}_P$ and $\mathcal{L}_C$ using distant supervision (Mintz et al., 2009), similar to the QA work of Bast and Haussmann (2015), Berant et al. (2013), and Yih et al. (2015). We utilize ClueWeb09-FACC1[1], a corpus of 500M Web pages annotated with Freebase entities.

Definition 2.4 (Lexicon)**:** A lexicon $\mathcal{L}$ is a weighted dictionary that maps natural language phrases to KB semantic items.

To create the predicate lexicon, $\mathcal{L}_P$, we run the following extraction pattern over the ClueWeb09-FACC1 corpus: *"e_1 r e_2"*, where e_1 and e_2 are entities and r is a phrase. For example,

*"[[Albert Einstein — **`AlbertEinstein`**]] was born in [[Ulm — **`Ulm`**]] ..."*

Following the distant supervision assumption, if (e_1 p e_2) is a triple in the knowledge base, then r expresses p and we add $r \mapsto p$ to $\mathcal{L}_P$. For the knowledge base triple (`AlbertEinstein placeOfBirth Ulm`) we add *"was born in"* $\mapsto$ `placeOfBirth` to $\mathcal{L}_P$. Of course, this assumption will not always hold, so we assign the mapping a weight w proportional to how many times it was observed in ClueWeb09-FACC1.

For the class lexicon, $\mathcal{L}_C$, we run Hearst patterns (Hearst, 1992) over the annotated corpus, where one argument is an entity and the other is a noun phrase. For example, for *"e and other np"*, we add to $\mathcal{L}_C$ the entry $np \mapsto c$ for each c such that (e `type` c) $\in KB$. For example, given the sentence *"[[Albert Einstein — **`AlbertEinstein`**]] and other scientists ..."* we add, for each type

[1] `http://lemurproject.org/clueweb09/FACC1/`

c to which the entity `AlbertEinstein` belongs, the entry *"scientist"* $\mapsto c$ to our $\mathcal{L}_C$. Entries are assigned a weight w proportional to their frequencies in ClueWeb09-FACC1. Table 2.1 shows a fragment of our lexica $\mathcal{L}_P$ and $\mathcal{L}_C$.

2.3 Named Entity Recognition

Named entity recognition (NER) is the task of identifying mentions of named entities in text and classifying them into one of a predefined set of types like `person` or `organization` (Finkel et al., 2005; Klein et al., 2003; Lample et al., 2016). NER is at the heart of information extraction tasks for recognizing mentions in text. For example, the following sentence *"play we are the champions by queen"* contains two mentions, a `song`, '*we are the champions*' and an `artist`, '*queen*'. NER is an important task for question answering and is applied first in any QA pipeline. Classifying mentions into types reduces the search space of candidate entities and candidate predicates, which leads to efficient answering.

Usually, NER systems operate on a handful of coarse-grained types; `person`, `location` and `organization` (Sang, 2002; Sang and Meulder, 2003). While NER classifies mentions into coarse-grained types, the task of *named entity typing* (NET) handles finer types such as `scientist`, `park` or `primarySchool` (Del Corro et al., 2015; Yosef et al., 2012). However, the main focus of NET is to type mentions, not detecting their boundaries. Therefore, the starting point of many NET approaches is an off-the-shelf NER system to identify mentions. NET then tries to type the identified mentions with the most fine-grained type given the context. In our work, we propose an approach for NER, with a larger set of coarse and fine-grained types.

3 Automated Template Generation for Question Answering over Knowledge Bases

3.1 Introduction

Templates play an important role in question answering (QA) over knowledge bases (KBs), where user utterances are translated to structured queries via semantic parsing (Berant et al., 2013; Unger et al., 2012; Yahya et al., 2012). Utterance templates are usually paired with query templates, guiding the mapping of utterance constituents onto query components.

Table 3.1 shows two utterance-query templates used by Fader et al. (2013). Each template i) specifies how to chunk an utterance into phrases, ii) guides how these phrases map to KB primitives by specifying their semantic roles as predicates or entities, and iii) aligns syntactic structure in the utterance to the semantic predicate-argument structure of the query.

A benefit of templates is that the mappings to the KB are traceable and can be leveraged to generate explanations for the user to understand why she receives specific answers. For example, when the utterance *"Who invented the Internet?"* returns the answer Al Gore (known for funding the Internet), an explanation might tell us that it was answered from the first template of Table 3.1, and that the KB predicate used was `knownFor`, originating from '*invented*'

#	**Utterance Template**	**Query Template**
1	Who VP_{pred} NP_{ent}	(?x, *pred*, *ent*)
2	What NP_{pred} is NP_{ent}	(*ent*, *pred*, ?x)

Table 3.1: Curated templates by Fader et al. (2013). Shared *pred* and *ent* annotations indicate an alignment between a phrase in the utterance template and a KB semantic item in the corresponding query template.

in the utterance through the *pred* alignment.

Many prior approaches to QA over KBs rely on hand-crafted templates or rules, which inevitably results in limited coverage. Some methods use utterance-query templates with alignments as in Table 3.1 (Fader et al., 2013; López et al., 2016; Reddy et al., 2016; Unger et al., 2012; Yahya et al., 2012; Yin et al., 2015). Bast and Haussmann (2015) rely on three manually constructed query templates without any utterance templates. Instead, they exhaustively instantiate each query template from the utterance. Yih et al. (2015) define a sequence of stages for generating queries, each relying on a set of manually defined rules for how query conditions are added. Berant et al. (2013) rely on a manually specified grammar for semantic parsing and mapping text spans onto KB primitives.

Our method supports complex compositional questions that require multiple KB predicates to obtain an answer. For example, the question *"actors starring in The Departed who were born in Cambridge, MA"* is answered by multiple KB predicates. Our system, called QUINT, is able to exploit natural language compositionality and templates learned from simple questions to answer complex questions without any templates that capture the complete question.

Our approach is to automatically learn utterance-query templates with alignments between the constituents of the question utterance and the KB query (Section 3.3). These templates are learned by distant supervision, solely from questions paired with their answer sets as training data (Kwiatkowski et al., 2013; Liang et al., 2011). Prior work used the same input for training QA systems over KBs, but relying on hand-crafted templates (Bast and Haussmann, 2015; Yih et al., 2015) or hand-crafted rules in various formalisms such as DCS (Berant et al., 2013) to generate the structure of KB queries. In contrast, our automatically learned utterance templates are based on standard dependency parse trees, thus benefiting from methodological progress and tools on mainstream parsing (Del Corro and Gemulla, 2013). Dependency parse representations also enable us to cope with compositional utterances. On the KB query side, templates are expressed in a SPARQL-style triple-pattern language 2.1. Our templates include automatic support for semantic answer typing, an important aspect where prior work relied on manually specified typing patterns (Bast and Haussmann, 2015; Berant et al., 2013). The alignments between utterance and query are computed by integer linear programming.

Template learning is an offline step. Online (Section 3.4), when the user interacts with the system, QUINT performs light-weight template matching. QUINT, uses basic templates to answer structurally complex compositional questions without observing such questions during training. QUINT accomplishes this

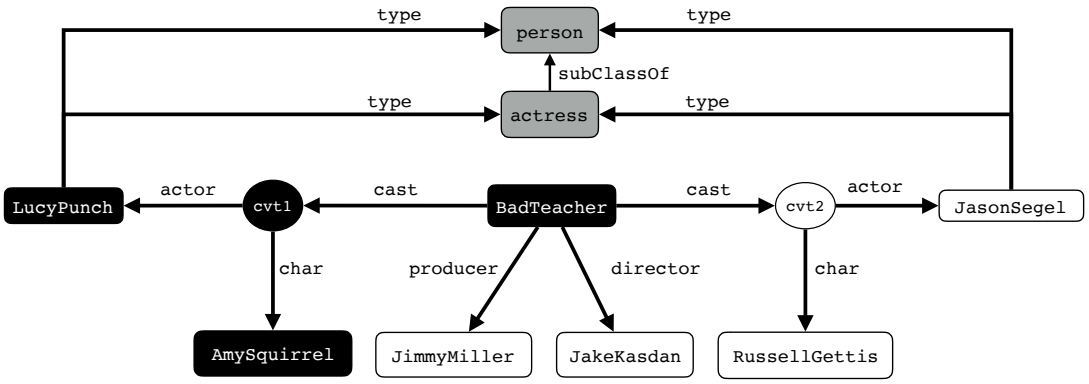


Figure 3.1: Example KB fragment.

by i) automatically decomposing the question into its constituent clauses, ii) computing answers for each constituent using our templates, and iii) combining these answers to obtain the final answer.

Our salient contributions are: i) automatically learning role-aligned question-query templates, ii) handling compositional complex questions, and iii) experiments with the WebQuestions (Berant et al., 2013) and Free917 (Cai and Yates, 2013) benchmarks and with complex questions.

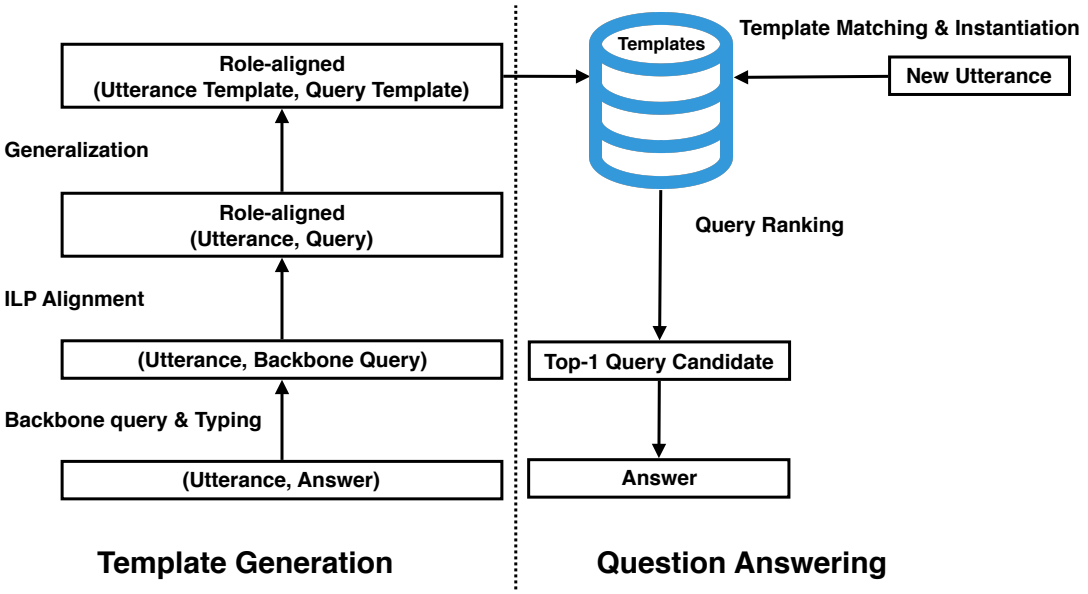


Figure 3.2: An outline of how QUINT generates role-aligned templates at training time and how it uses them for question answering.

3.1.1 Overview

Figure 3.2 outlines our system for learning and subsequently using templates to answer natural language questions over knowledge bases. Templates are generated at training time (Figure 3.2, left), and are used at testing (answering) time (Figure 3.2, right) for answering questions. The input to the training stage is pairs of question *utterance* u and its *answer set* A_u from the KB. An example of a training utterance is $u =$ *"Which actress played character Amy Squirrel on Bad Teacher?"* [sic], which is paired with the answer set $A_u = \{\texttt{LucyPunch}\}$. We use the letter u to refer to both the utterance and its dependency parse tree interchangeably. A *dependency parse tree* (Klein and Manning, 2003) of an utterance is a directed rooted tree whose nodes correspond to utterance tokens and edges represent grammatical relations between the nodes. Our utterance templates are based on the dependency parse of utterances. Our motivation is that a dependency parse (1) can capture long range dependencies between the tokens of an utterance which helps when answering compositional questions (Section 3.5) and (2) gives great flexibility allowing QUINT to skip irrelevant tokens when instantiating query templates (Section 3.4.1). Figure 3.3 shows the dependency parse of the above utterance.

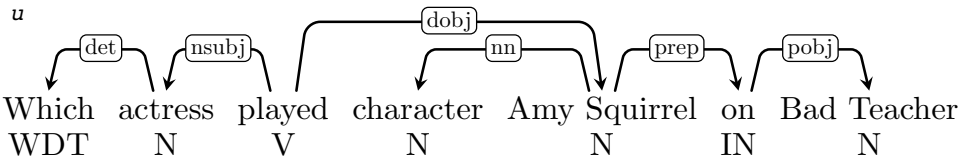


Figure 3.3: The dependency parse tree of our example utterance (top) together with the answer set (bottom).

QUINT starts by heuristically constructing a KB query q that captures u. It constructs a backbone query $\hat{q}$ for q by finding for each $a \in A_u$ the smallest subgraph connecting the KB entities detected in u and a (Section 3.3). For our example in Figure 3.3, this is the subgraph connecting the black nodes in Figure 3.1. QUINT forms $\hat{q}$ from this subgraph by replacing the nodes corresponding to a or a CVT with variables (Figure 3.4). To account for types, it extends $\hat{q}$ by adding the type constraints connected to a to the corresponding variable (Section 3.3.2) – the gray nodes in Figure 3.1. Figure 3.5 shows the resulting $\hat{q}$.

Next, QUINT aligns $\hat{q}$ with u, which gives us $q \subseteq \hat{q}$ that best captures u, a chunking of u, and an alignment between the constituents of u and q. The

alignment is carried over to the templates created from u and q. We formulate the alignment problem as a constrained optimization and find the best alignment m using integer linear programming (ILP) (Section 3.3.3). Figure 3.7 shows u, the resulting q, and alignment m indicated with shared *ent*, *pred*, and *type* annotations between nodes in u and q, which also specify the semantic role for this alignment.

Next, QUINT performs generalization to generate a template from a concrete pair of aligned utterance dependency parse tree and query graph (Section 3.3.4). It removes the concrete text in the nodes participating in m and similarly for the semantic items in q, keeping the annotations *ent*, *pred*, and *type*, thereby turning these nodes into placeholders (Figure 3.8). The result is template $t = (u_t, q_t, m_t)$ composed of an utterance template u_t, a query template q_t, and an alignment m_t between the two.

When the user issues a new question, QUINT matches its dependency parse tree against the template library created during training (Section 3.4.1). In Figure 3.9 the bold edges and nodes in u' are those matched by u_t (Figure 3.8). For each match, the corresponding query template q_t is instantiated using the alignment m_t and the lexicon. Figure 3.9 also shows a query resulting from such an instantiation. Finally, QUINT ranks these candidate queries using a learning-to-rank approach (Section 3.4.2). The answers of the top-ranked query are returned to the user. By showing which template was used and how it was instantiated, QUINT can explain answers to the user.

3.2 Related Work

Question answering. One of the earliest work on question answering was the work of Green et al. (1961), coined BASEBALL, where the answering resource is a database of baseball games. BASEBALL enabled end users to ask full-fledged

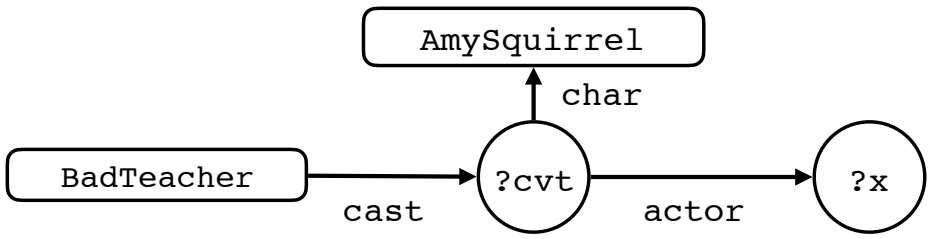


Figure 3.4: Backbone query $\hat{q}$.

natural language questions in a specific domain, namely sport. LUNAR (Woods, 1972) could answer questions about the Apollo 11 mission to the moon. LADDER (Hendrix et al., 1978) is a rule-based QA method over a navy database.

QA has a long tradition in IR and NLP communities, including benchmarking tasks in TREC (Dang et al., 2006; Dietz and Gamari, 2017; Voorhees and Tice, 2000), CLEF Magnini et al. (2004); Herrera et al. (2004) and SemEval. This has predominantly focused on retrieving answers from textual sources (Ferrucci, 2012; Harabagiu et al., 2006; Lin, 2002; Lin and Katz, 2003; Prager et al., 2004; Ravichandran and Hovy, 2002; Saquete et al., 2004, 2009; Yin et al., 2015). TREC QA evaluation series (1999-2007) provide hundreds of factoid questions to be answered over a collection of documents (Dang et al., 2007). Typically, IR-based QA methods answer a question over a collection of documents in two phases: i) question analysis and ii) answer retrieval (Katz, 1997). In in the first phase, the expected type of the answer is identified using either a handful of lexico-syntactic rules or machine-learning methods (Hovy et al., 2000; Li and Roth, 2002; Pasca and Harabagiu, 2001; Ravichandran and Hovy, 2002). Then, a query is formulated from the question at hand to be issued against the underlying collection of documents (Hovy et al., 2000; Xu and Croft, 1996). In the second phase, a list of relevant documents is retrieved in response to the formulated query. Next, a more comprehensive syntactic and semantic analysis of the retrieved list of documents is applied in order to identify highly relevant passages that might contain the answer of interest, including dependency/constituency parsing, chunking, POS tagging, named entity recognition and typing (Salton et al., 1993; Srihari and Li, 1999). Finally, answers are extracted from the retrieved passages (Brill et al., 2002; Lin, 2007). For a more comprehensive overview of IR-based QA (Hirschman and Gaizauskas, 2001; Jurafsky and Martin, 2009).

Semantic search. With traditional Web search over textual corpora reaching maturity, there has been a shift towards semantic search focusing on entities, and more recently on relationships. This shift was brought on by an explosion of structured (Bizer et al., 2009) and semi-structured data (Guha et al., 2016).

Entity search over KBs has gained wide attention (Balog et al., 2010; Blanco et al., 2011; Joshi et al., 2014; Tran et al., 2007, 2009). These are keyword queries asking for entities (or other resources), and have been shown to be very common (Pound et al., 2010).

Question answering over knowledge bases. More recent efforts have focused on natural language questions as an interface for knowledge bases. Questions express complex relation-centric information needs more naturally, allowing

for better KB utilization. They are also more natural when dealing with new modalities such as voice interaction (e.g., Amazon Alexa, Google Home). Multiple benchmarks have been introduced for this problem (Berant et al., 2013; Cai and Yates, 2013; Tsatsaronis et al., 2012; Unger et al., 2015). These differ in the underlying KBs and supporting resources, size, and question phenomena they evoke, resulting in various solutions from those heavily relying on machine learning to more hybrid approaches using a combination of rules and hand-crafted scoring schemes. We presented experimental results for QUINT on the benchmarks introduced by Berant et al. (2013) and Cai and Yates (2013). We could not conduct experiments on benchmarks such as QALD (Unger et al., 2015) and BioASQ (Tsatsaronis et al., 2012) due to the small size of their training sets, and because they emphasize aspects not addressed by QUINT (e.g., aggregation, ordering).

Templates play an important role in QA over KBs. Unger et al. (2012); Unger and Cimiano (2011), Yahya et al. (2012, 2013), and Zou et al. (2014) present approaches that use manually defined templates to handle complex questions with compositionality. These systems use regularities in how syntactic patterns map to semantic ones to create their templates. The drawback of these approaches is the limited coverage of templates, making them brittle when it comes to unconventional question formulations. By automating template generation, we can learn new templates dynamically.

Fader et al. (2013) use a small number of handcrafted templates for QA, focusing on simple queries with a single triple pattern, as illustrated in Table 3.1. In contrast, we use dependency parsing on the utterance side, allowing for better generalization. On the query side, we are not limited to single triple patterns.

Zheng et al. (2015) tackle the problem of utterance-query template construction in a setting different than ours with a query workload and a question repository as input. The task is to pair questions with workload queries that best capture them. Each pair is subsequently generalized to a template. In contrast, our approach does not rely on the availability of observed SPARQL queries.

Berant et al. (2013) use a set of rules for composing logical forms in the DCS semantic representation constructed from all pairs of non-overlapping text spans. Bast and Haussmann (2015) use three manually defined query templates, with no corresponding utterance templates. These are exhaustively instantiated based on the utterance. This is similar to our query generation, but is performed at answering time. Yih et al. (2015) use a staged approach to map utterances to queries. Each stage uses a set of rules for adding conditions to a query at answering time. Other works rely on embedding both questions and KB entities,

paths, and subgraphs in a shared space (Bordes et al., 2014; Dong et al., 2015; Yang et al., 2014).

Yao and Durme (2014) address QA over KBs as an information extraction problem, reminiscent of traditional QA over text corpora. This approach makes heavy use of manually defined templates for extracting cues about the answer entity and the relevant KB predicates.

A popular formalism for mapping utterances into logical forms is CCGs (Steedman, 2000; Zettlemoyer and Collins, 2005). Kwiatkowski et al. (2013) use a probabilistic CCG to build a linguistically motivated logical form. The subsequent translation to a KB-specific semantic representation is performed by a trained ontology matching model. Cai and Yates (2013) combine learning a CCG grammar from question-query pairs with an approach for lexicon extension. We work with dependency parsing to capture question syntax. This is a pragmatic choice to benefit from the methodological progress and tools available (Del Corro and Gemulla, 2013). Reddy et al. (2016) propose an approach to map a dependency parse to a logical form in two steps. First, using a repository of manual rules, a dependency parse is transformed into a logical form whose symbols are words and edge labels. This is transformed into a semantic representation following Reddy et al. (2014) using graph matching.

Combining KBs and text. Work has also been done to combine KBs with supporting textual data for QA. The role of text here is either to support ranking of answer candidates (Savenkov and Agichtein, 2016; Xu et al., 2016b), or as a source of answers (Fader et al., 2013; Usbeck et al., 2015). In our work we use a text corpus linked to our KB for lexicon construction offline. However, we stick to answering exclusively over the KB at answering time.

Multiple KBs. Finally, going beyond a single KB, people have looked at QA over interlinked KBs. PowerAqua (López et al., 2010) answers questions in this setting, focusing on how to combine answers obtained from different sources. Shekarpour et al. (2015) describe a two-stage system for answering questions and keyword queries over linked data, composed of two stages: question segmentation and segment disambiguation, followed by query generation.

3.3 Template Generation

We now present the details of the offline template generation stage. The input to this stage is pairs of utterance u and its answer set A_u as in Figure 3.3.

3.3.1 Backbone Query Construction

To generate the backbone query $\hat{q}$, QUINT starts by annotating a training utterance u with the named entities it contains and disambiguating these to Freebase entities using an off-the-shelf named entity recognition and disambiguation (NERD) system (Yang and Chang, 2015). For our example, the resulting annotated question is:

"Which actress played character [[*Amy Squirrel* — `AmySquirrel`]] *on* [[*Bad Teacher* — `BadTeacher`]]*?"*

Next, for each answer $a \in A_u$, QUINT finds the smallest connected subgraph of the KB that contains the above entities found in the question as well as a. For our example and the KB of Figure 3.1, this is the subgraph with black nodes. To this end, starting with an entity found in the question, QUINT explores all paths of length two when the middle node is a CVT node and paths of length one otherwise, to restrict the search space, similarly to (Yih et al., 2015). If the middle node is a CVT node and the question contains multiple entities, QUINT explores paths of length one connecting the CVT node with these entities. We assume that this subgraph captures the meaning of the question and connects it to one of its answers a. There may be multiple such graphs. QUINT then transforms this subgraph into a query by replacing a with the variable `?x` and any CVT nodes with distinct variables (e.g., `?cvt1`). The above procedure is performed for each $a \in A_u$ for a given u, resulting in multiple queries. We keep the query with the highest F1 with respect to A_u. Figure 3.4 shows $\hat{q}$ for our example.

3.3.2 Capturing Answer Types

Capturing the answer types given in the question is important for precision. In the question *"Which **actor** died in New York?"*, $\hat{q}$ generated thus far would be {`?x deathPlace NewYork`}, which does not capture the fact that the question is only interested in actors, hurting precision. Walker et al. (2015) showed that identifying the expected answer type of an utterance boosts the performance of QA systems. This conclusion is proven in our experiments as well (Section 3.6). Earlier QA systems either use manual rules to find phrases in the question that evoke one of a number of predefined set of possible types in the query (Bast and Haussmann, 2015; Berant et al., 2013), or neglect type constraints altogether (Yih et al., 2015). QUINT automatically creates templates that capture which phrases in the question evoke types in the query, and uses the full Freebase type system as potential mapping targets.

Starting with $\hat{q}$ generated thus far (Figure 3.4), QUINT connects to the answer variable node in $\hat{q}$ one *type constraint* for each $c \in C$ such that the variable originates from the answer entity $a \in A_u$ and $(a \texttt{ type } c) \in KB$. In the KB of Figure 3.1, `LucyPunch` $\in A_u$ has the types `person` and `actress`, the resulting $\hat{q}$ is shown in Figure 3.5. $\hat{q}$ now contains more type constraints than actually given in the question. In the next section we show how the correct one is determined as part of the alignment step.

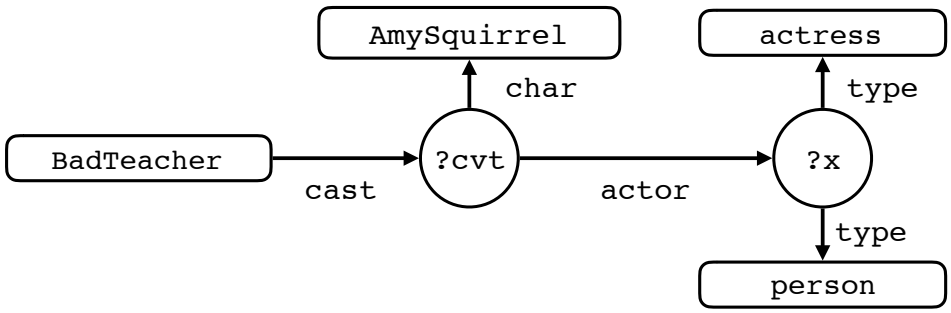


Figure 3.5: Backbone query $\hat{q}$ with types.

3.3.3 Utterance-Query Alignment

With $(u, \hat{q})$ pairs at hand, QUINT proceeds to aligning the constituents of the two. The alignment i) gives us a chunking of u into phrases that map to semantic items in $\hat{q}$, ii) removes spurious type constraints from $\hat{q}$, resulting in $q \subseteq \hat{q}$, and iii) gives us an alignment m between the constituents of u and q.

Alignment is driven by our lexicons $\mathcal{L}_P$ and $\mathcal{L}_C$ (Section 2.2), but faces inherent ambiguity, either from truly ambiguous phrases or from inevitable noise in the automatically constructed lexicons. We model the resolution of this ambiguity as a constrained optimization and use an ILP to address it. We start by building a bipartite graph with Ph, the set of all phrases from u, on one side and $S_{\hat{q}}$, the set of semantic items in $\hat{q}$, on the other as shown in Figure 3.6. $Ph = \{ph_1, ph_2, ...\}$ is generated by taking all subsequences of tokens in u. We add an edge between each $ph_i \in Ph$ and $s_j \in S_{\hat{q}}$ where $(ph_i \mapsto s_j) \in \mathcal{L}_P \cup \mathcal{L}_C$ with a weight w_{ij} from the lexicon. Table 3.2 shows a fragment of our lexicons.

Additionally, we add an edge connecting each entity in $\hat{q}$ to the phrase that evokes it in u. Entities are added to prevent their surface forms in the question from mapping to a class or a predicate. We use an off-the-shelf NERD system to identify entities in u.To resolve the ambiguity in the mapping of types and

predicates and obtain the intended alignment, the ILP decides which subset of the edges we need to keep.

We extend our notation before presenting our ILP. For semantic item s_j, E_j, C_j and P_j are 0/1 constants indicating whether s_j is an entity, type, or predicate, respectively. X_{ij} is a 0/1 decision variable whose value is determined by the solution of the ILP. The edge connecting ph_i to s_j in the bipartite graph is retained iff $X_{ij} = 1$. Given a set of types connected to a variable v from which we want to pick at most one, this set of types is $S(v) = \{c_1, c_2, ...\}$ ({`actress`,`person`} in our example) and the set of phrases that can map to types in $S(v)$ is $Ph(v)$.

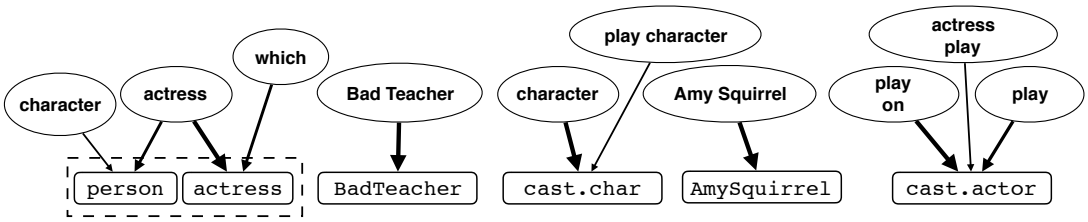


Figure 3.6: Bipartite graph provided to the ILP, with phrases at the top and semantic items at the bottom. An edge corresponds to a mapping from a phrase to a semantic item, and its thickness corresponds to the mapping weight. Dashed box represents $S($`?x`$)$ for `?x` in Figure 3.5.

The objective of the ILP is to maximize the total weight of the mapped phrases:

$$\sum_{i,j} w_{ij} X_{ij},$$

where w_{ij} comes from the lexicon. The constraints make sure that the resulting alignment is a meaningful one:

1. Each semantic item is obtained from at most one phrase: $\forall s_j \in S_{\hat{q}} : \sum_{i,j} X_{ij} \leq 1$. In Figure 3.6, this means that `cast.actor` comes either from '*play on*' or '*play*'.

2. A token contributing to an entity phrase cannot contribute to any other phrase: $\forall ph_i \in Ph, ph_{i'} \in Ph(t), t \in ph_i, s_j, s_{j'} \in S_{\hat{q}} : X_{ij} + E_j \leq \sum_{i',j'} X_{i'j'} + 1$. In Figure 3.6, this means that the tokens of '*Amy Squirrel*' cannot be part of a mention of a semantic item other than `AmySquirrel` (if there was such a candidate).

3. For each variable v, at most one phrase in $Ph(v)$ can map to at most one type in $S(v)$: $\forall v, ph_i \in Ph(v), c_j \in S(v) : \sum_{i,j} X_{ij} \leq 1$. In Figure 3.6, this means that either `person` or `actress` will be chosen, and only one of the phrases mapping to the chosen type among these.

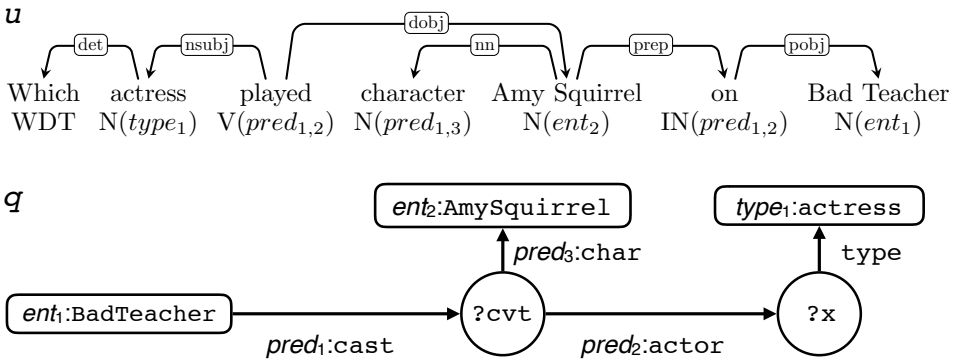


Figure 3.7: Aligned utterance query pair (u, q, m). m is indicated by shared *ent*, *pred*, and *type* annotations (e.g., *"played on"* is aligned with `cast.actor`).

Phrase	**Semantic Item**	w	**Phrase**	**Semantic Item**	w
'*play on*'	`cast.actor`	0.5	'*role*'	`cast.char`	0.6
'*play*'	`cast.actor`	0.3	'*actress*'	`actress`	0.7
'*character*'	`cast.char`	0.6	'*actress*'	`person`	0.3

Table 3.2: Fragment of our lexicons: $\mathcal{L}_P$ and $\mathcal{L}_C$.

We solve the ILP using Gurobi to obtain the intended alignment m, indicated by shared *ent*, *type*, and *pred* semantic annotations between u and $\hat{q}$. Additionally, by discarding all type constraints not connected to a phrase in m (`?x type person` in our example), we obtain q from $\hat{q}$. Figure 3.7 shows the utterance from our running example aligned with q (contrast with $\hat{q}$ in Figure 3.5).

An important by-product of alignment at training time is a lexicon $\mathcal{L}_{P_{train}} \subseteq \mathcal{L}_P$, composed of the phrase-predicate alignments observed during training. This lexicon is much cleaner than the noisier $\mathcal{L}_P$. We will use both at testing time, giving precedence to mappings obtained from $\mathcal{L}_{P_{train}}$ when they exist.

3.3.4 Generalization to Templates

Next, QUINT constructs templates from aligned utterance-query pairs (u, q, m) obtained from the alignment process above. On the utterance side, QUINT takes the utterance u represented using its dependency parse tree and restricts it to the smallest connected subgraph that contains the tokens of all phrases participating in m. In Figure 3.7 this results in removing the node corresponding to '*which*'.

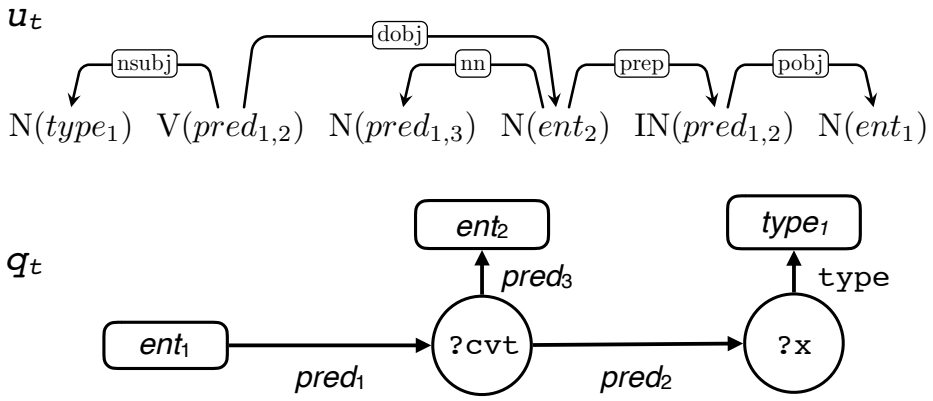


Figure 3.8: A template $t = (u_t, q_t, m_t)$. m_t is indicated by shared *ent*, *pred*, and *type* annotations. Figure 3.7 shows the concrete utterance-query pair used to generate this template.

To create a template from this subgraph, we turn the nodes participating in m into placeholders by removing their text and keeping the POS tags and semantic alignment annotations (*ent*, *type*, *pred*). We use universal POS tags (Petrov et al., 2012) for stronger generalization power. We replace compound nouns with a noun token that can be used to match compound nouns at testing time to ensure generalization. At testing time, our templates allow for robust chunking of an incoming question into phrases corresponding to entities (i.e., as named entity recognizers), predicates (i.e., as relation extractors) and types (i.e., as noun phrase chunkers). For NER, we show that using our templates at testing time gives superior results when compared to using an off-the-shelf NER system.

On the query side, we take the query and remove the concrete labels of edges (predicates) and nodes (entities and types) participating in m, keeping the semantic alignment annotations. We use the number of utterance-query pairs which generate a template as a signal in query ranking (Section 3.4).

3.4 Simple Question Answering with Templates

3.4.1 Candidate Query Generation

At answering time, when a user poses a new utterance u', QUINT matches it against all templates in our repository. u' matches a template (u_t, q_t, m_t) if a subgraph of its dependency parse tree is isomorphic to u_t considering their edge

labels and the POS tags in their nodes. Figure 3.9 shows an example of u' whose dependency parse matches the utterance template u_t in Figure 3.8. Bold edges and nodes are the ones participating in the isomorphism with u_t. Note how using a dependency parse to represent an utterance allows us to ignore the token '*popular*' in u', which does not carry any semantics that can be captured by our specific KB. If the question asked, say, for an '*American*' actress instead, then another template learned at training time would allow us to instantiate a query that captures this important constraint. Our ranking scheme described below decides which is the best match.

For each matching utterance template (usually several), QUINT instantiates the corresponding query template q_t based on the alignment m_t and the lexicon $\mathcal{L}$. Lexicon lookups are guided by the semantic alignment annotations in the query template, which restrict specific parts of the query to types, entities, or predicates. For predicates, QUINT first performs a lookup in the cleaner $\mathcal{L}_{P_{train}}$ created at training time (Section 3.3.3). If no results are returned, QUINT queries the full predicate lexicon $\mathcal{L}_P$ (Section **??**). Figure 3.9 bottom shows a query q' obtained from instantiating the template of Figure 3.8 based on u' shown at the top of the same figure.

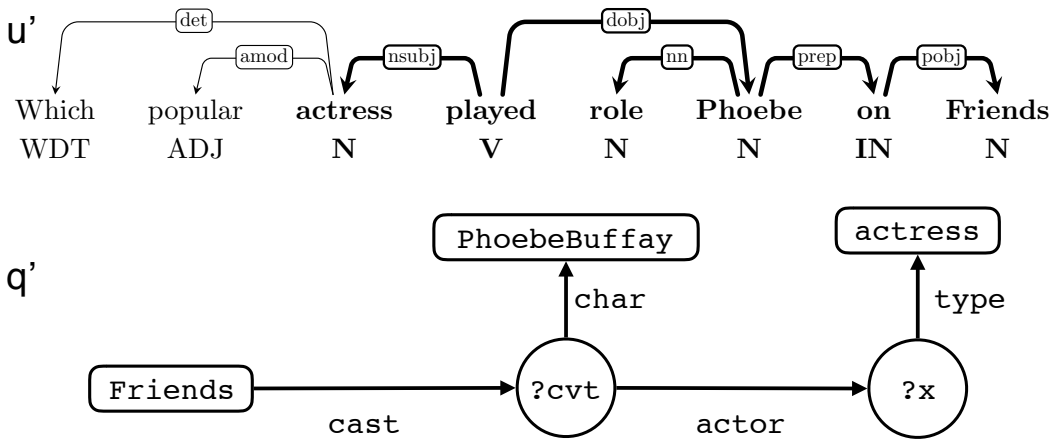


Figure 3.9: Template instantiation (using t in Figure 3.8).

3.4.2 Query Ranking

Query generation yields multiple candidate queries for an utterance, either due to multiple matching templates as discussed above or due to ambiguity of phrases

Category	#	**Description**
Alignment	1,2	Average & sum of lexicon weights for predicates
	3,4	Average & sum of lexicon weights for entities
	5	# of utterance tokens literally matching their predicate
	6	# of entity mentions matching their canonical name in the KB
	7	Indicator: question n-gram $\wedge$ predicate p, $n = 1$, 2 and 3
Semantic	8,9	Average & sum of entity popularity scores
	10	# of predicates in the query
	11	# of entities in the query
Template	12	Indicator: t captures a type constraint
	13	# of training utterances that generate t
	14	t coverage: % of tokens in utterance matched by u_t
	15	Indicator: utterance node with *type* annotation $\wedge$ semantic answer type (if t is typed)
Answer	16	Indicator: answer set size ($=1$, $=2$, $=3$, $\in$ [4-10], > 10)

Table 3.3: Query candidate ranking features.

in the lexicon. We adopt a learning-to-rank approach, analogously to Bast and Haussmann (Bast and Haussmann, 2015), to rank the obtained queries, and return the highest ranking query as the one intended by the question. We use a random forest classifier to learn a preference function between a pair of queries (Breiman, 2001). Table 3.3 lists the features used for the feature vector $\phi(u', t, q')$ associated with the instantiation of template t from utterance u' resulting in query q', which we discuss below. For instantiations of a pair of templates t_1 and t_2 matching utterance u' and resulting in the queries q_1 and q_2, respectively, the feature vector used is a result of concatenating $\phi(u', t_1, q'_1)$, $\phi(u', t_2, q'_2)$, and $\phi(u', t_1, q'_1) - \phi(u', t_2, q'_2)$.

We compute four types of features as shown in Table 3.3. *Alignment* features measure the association between utterance tokens and KB semantic items. *Semantic* features consider the query exclusively. *Template* features measure the appropriateness of a template t for a given an utterance. The *answer* feature template indicates which of the predefined ranges the query answer set size belongs to. Answer size is a good cue as empty answer sets or very large ones are indicators of incorrect queries that are over or under-constrained. Features 7, 15, and 16 define feature templates that are instantiated based on the training

data. For example, instantiations of feature 7 are pairs of utterance n-gram and query predicate. Indicator features take boolean values. For instance, from the utterance and the query in Figure 3.9 we generate the feature "*actress play* $\wedge$ *cast.actor*" with value 1.

3.5 Compositional Question Answering with Templates

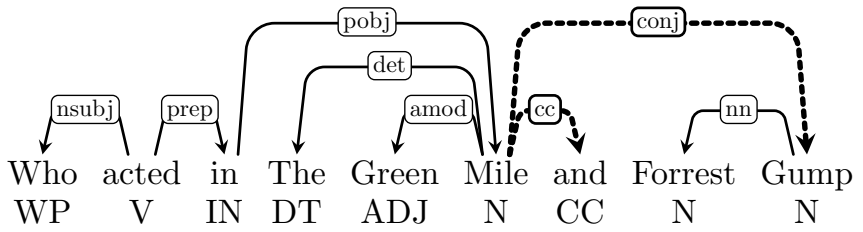
The class of complex questions we target are those composed of multiple clauses, each centered around a relationship, which collectively describe a single "variable" (with possibly multiple bindings in the KB). For example, the question *"actors starring in The Departed who were born in Cambridge, MA"* is composed of two clauses: *"actors starring in The Departed"* (e.g., `MattDamon`, `MarkWahlberg`, `JackNicholson`) and the relative clause *"actors who were born in Cambridge, MA"* (e.g., `MattDamon`, `UmaThurman`, `MarkWahlberg`). `MattDamon` and `MarkWahlberg` are answers to the complete question.

Handling complex questions is a natural extension of the procedure in Section 3.4.1. We do this in three steps: i) automated dependency parse rewriting when necessary, ii) sub-question answering, and iii) answer *stitching*.

Automated dependency parse rewriting. The need for rewriting arises when we have complex questions that we are unable to fully capture with our template repository. An example is *"Who acted in The Green Mile and Forrest Gump?"*. For this question, our templates trained on the simpler WebQuestions dataset, would be unable to capture the second constraint, expressed through a coordinating conjunction (using '*and*'). The problem can be seen in Figure 3.10(a), where there is no direct connection between '*Forst Gump*' and the relation phrase '*acted in*', which our templates would expect given the data used to learn them. To overcome this, we perform simple automated rewriting to get two separate interrogative propositions following Del Corro and Gemulla (2013) for relative clauses and coordinating conjunctions. In our concrete example, this is done by connecting '*Gump*', the target of the *conj* edge, to '*in*', the head of '*Mile*' which is the source of the *conj* edge. We give the new edge the label *pobj*, the same as the one connecting the head of *conj* to its parent. The *conj* and *cc* edges are subsequently removed. The resulting dependency graph and sub-questions captured by our templates are shown in Figure 3.10(b).

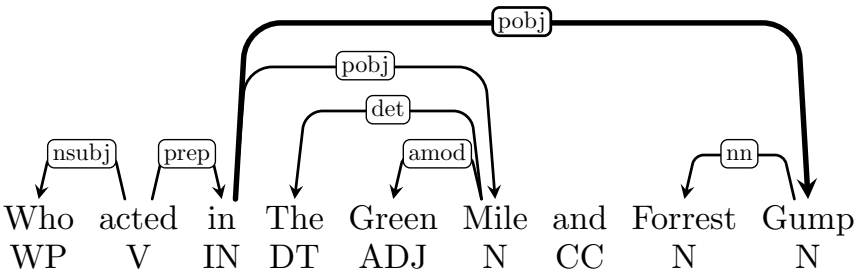
More generally, a rewriting is triggered if i) we detect a coordinating conjunction or relative clause (Del Corro and Gemulla, 2013) and ii) matches against

our template repository result in less sub-questions than expected (e.g., a single coordinating conjunction or relative clause should result in two matched sub-questions). While the question above requires rewriting, the question *"What film directed by Steven Spielberg did Oprah Winfrey act in?"*, where both relations are spelled out, requires no rewriting by QUINT.



"Who acted in The Green Mile and Forrest Gump"

(a)



"Who acted in The Green Mile"
"Who acted in Forrest Gump"

(b)

Figure 3.10: (a) shows the original dependency parse of the question before rewriting (b) shows the dependency parse after rewriting i.e., dropping dashed edges in (a) and adding a new edge (bold) connecting *"Gump"* to *"in"*.

Sub-question answering. We match our templates against the complete automatically rewritten (if needed) dependency parse . Each match corresponds to a sub-question that can be answered independently. Here, we follow the procedure described in Sections 3.4.1 and 3.4.2 for each question independently, and keep the ranked list of queries returned for each subquestion.

Stitching. For each sub-question detected and mapped to a list of queries, we assign a query in its ranked list of queries (Section 3.4.2) a score of $\frac{1}{r}$, where r is the rank of the query within the list. The idea here is to assign a higher numerical score to a higher ranked query. We return the answers resulting from

the combination of queries, one for each sub-question, such that the intersection of their answer sets is non-empty and the sum of their scores is highest.

3.6 Experimental Evaluation

We evaluate QUINT on the WebQuestions and Free917 datasets, as well as a newly introduced set of complex questions. These experiments serve two goals: (1) to show that automatically learned templates can deliver high-quality results, and (2) that our templates provide a natural path toward handling complex questions.

3.6.1 Benchmark

We compare QUINT to previous work using the following benchmarks over Freebase:

- **WebQuestions** (Berant et al., 2013), which consists of 3778 training questions and 2032 test questions, each paired with its answer set, collected using the Google Suggest API and crowdsourcing.
- **Free917** (Cai and Yates, 2013), which consists of 917 questions manually annotated with their Freebase query; 641 training and 276 test questions. For Free917, we made use of a manually crafted entity lexicon provided by the designers of the benchmark for entity linking. All other systems in Table 3.4 used this lexicon. We used the standard train/test splits for WebQuestions and Free917.
- **ComplexQuestions.** This benchmark is composed of 150 test questions that exhibit compositionality through multiple clauses, e.g., *"What river flows through India as well as China?"* Crucially, ComplexQuestions contains *no* training questions: its purpose is to demonstrate that our template-based approach, while trained only on the simpler single-clause WebQuestions, can handle more complex questions. We constructed this benchmark using the crawl of WikiAnswers (`http://wiki.answers.com`), a large, community-authored corpus of natural language questions, collected by Fader et al. (2013). We asked a human annotator to collect questions with multiple clauses and also to provide the gold standard answer set for them from Freebase. We make this benchmark available to the community [1].

[1] `http://qa.mpi-inf.mpg.de/complex-questions-wikianswers-150.json`

3.6.2 Performance Measures

We use the evaluation metric adopted by each of the benchmarks. For WebQuestions this is the average F1 score across all test questions. We also adopt F1 for evaluating systems on ComplexQuestions. For Free917, the metric is accuracy, defined as the fraction of questions answered perfectly i.e., with the exact gold standard answer set.

3.6.3 Template Generation

We analyze our templates generated at training time. For WebQuestions, QUINT generates $\hat{q}$ queries (Sections 3.3.1, 3.3.2) for 3555 of the 3778 training questions. For the others, either no entity candidates were identified by the NERD system or no subgraphs connecting the identified entities were found. The ILP successfully aligned (Section 3.3.3) 3003 of the 3555 u–$\hat{q}$ pairs, resulting in 3003 (u, q, m) triples. The others could not be aligned due to lexicon misses. These produced 1296 distinct (u_t, q_t, m_t) templates. We use these templates to answer test questions in WebQuestions and ComplexQuestions later on.

For Free917, QUINT generates $\hat{q}$ queries for 602 of the 641 training questions. For the others, no subgraphs connecting the identified entities were found. The ILP successfully aligned 571 of the 602 u–$\hat{q}$ pairs, resulting in 571 (u, q, m) triples. These produced 284 distinct (u_t, q_t, m_t) templates.

3.6.4 Results on WebQuestions and Free917

Question answering performance. Table 3.4 shows the results on the test sets for WebQuestions and Free917 with additional entries for earlier work on these benchmarks. QUINT uses the templates generated by our full system described in Sections 3.3 and 3.4. QUINT-untyped uses templates where we skip the step described in Section 3.3.2 that allows the capturing of answer types.

On WebQuestions, QUINT outperforms existing approaches, while performing slightly below the system of Yih et al. (2015). However, the difference in F1 scores between our results and that of Yih et al. is not statistically significant using a two-sided paired t-test at 95% confidence level. Recent work has looked at a different setup combining KBs with additional textual resources for answering questions. For example, Xu et al. (2016b) and Savenkov and Agichtein (2016) use Wikipedia and Web search results combined with community question answering data, respectively. These systems achieve slightly higher F1 scores than our system with these resources (53.3 and 52.2, respectively), but the performance

Method	WebQuestions Average F1	Free917 Accuracy
Cai and Yates (2013)	-	59.0
Berant et al. (2013)	35.7	62.0
Kwiatkowski et al. (2013)	-	68.0
Yao and Durme (2014)	33.0	-
Berant and Liang (2014)	39.9	68.6
Bao et al. (2014)	37.5	-
Bordes et al. (2014)	39.2	-
Yao (2015)	44.3	-
Dong et al. (2015)	40.8	-
Bast and Haussmann (2015)	49.4	76.4
Liang and Potts (2015)	49.7	-
Yih et al. (2015)	**52.5**	-
Reddy et al. (2016)	50.3	78.0
This Work		
QUINT-untyped	50.8	**78.6**
QUINT	51.0	72.8

Table 3.4: Results on the WebQuestions and Free917 test sets.

drops without these (47.1 and 49.4, respectively). QUINT uses entity-annotated text corpora for creating lexicons offline, but following the original benchmark setup, does not invoke any other resources at answering time.

On Free917, QUINT-untyped outperforms all other methods and obtains the best result to date. Interestingly, QUINT-untyped outperforms QUINT on this benchmark. We comment on this below.

Templates as named entity recognizers. In the above experiment, we used our templates for named entity recognition, and relied on the S-MART NERD system for generating the top entity candidates for each recognized entity mention, with the final entity resolution being done by our ranking stage. On average, S-MART generated 2 entity candidates per mention. We tried fully relying on S-MART, by adopting its NER annotations as well. In this case, we restricted matches to those templates compatible with S-MART's annotations (i.e., their *ent* annotations agree with the entity annotations produced by S-MART). In this case, our F1 score drops to 49.0 as the number of questions with

no matching templates increases. This shows the value of our templates as entity recognizers in questions.

Effect of answer typing. In our experiments, answer typing had a positive impact on the results for WebQuestions, and a negative one on Free917. In both cases, this shows the importance of typing. For Free917, a large portion of the test questions ask for literals such as dates or monetary values. However, our type system does not capture these in a fine grained manner, limiting our ability to distinguish between, say monetary values and sizes. This is something to explore in future work. For WebQuestions, our type system comprehensively covers those types used by the entities, so we see a positive impact, which is in line with previous work (Walker et al., 2015).

Table 3.5 shows answers produced by QUINT for four sample questions. In the first and second questions, QUINT (typed) was able to restrict the answer set to only college/high school respectively where the QUINT-untyped failed. In the third question, the gold answer does not conform with answer type constraint in the question (*"city"*). Therefore, QUINT (typed) produced only the city as final answer, however, QUINT-untyped produced the correct answer. For the fourth question, typing the answer entity does not add value since both the answer type (`HumanLanguage`) as well as the type of the object of the KB predicate (`languageSpoken`) agree. The type signature for this KB predicate is `Country` and `HumanLanguage` for the subject and the object, respectively.

3.6.5 Results on ComplexQuestions

Results on ComplexQuestions reveal the full potential of QUINT. For prior works to fully answer ComplexQuestions they would need to have their manually created templates manually extended: a cumbersome task that quickly blows up. In contrast, QUINT trains on the structurally simple training subset of WebQuestions, which does not capture the full complexity of ComplexQuestions. It exploits language compositionality to automatically decompose complex question utterances to their constituent clauses and form simpler "sub-questions" which it answers individually before combining their answers to answer the complete question (Section 3.5).

Table 3.6 shows the results on ComplexQuestions achieved by QUINT and the system of Bast and Haussmann (2015) (AQQU), the best publicly available system on WebQuestions (Table 3.4). It is important to keep in mind that AQQU is not designed to handle ComplexQuestions. To guarantee a fair comparison, we ran two variants of this system; AQQU is the officially published

Question	*"what college did john stockton go to?"*
Gold	Gonzaga University
QUINT (typed)	Gonzaga University
QUINT (untyped)	Gonzaga Prep. School, Gonzaga University
Question	*"where did aaron rodgers go to high school?"*
Gold	Pleasant Valley High School
QUINT (typed)	Pleasant Valley High School
QUINT (untyped)	Butte College, UC Berkeley, Pleasant Valley High School
Question	*"what city is acadia university in?"*
Gold	Canada, Nova Scotia, Wolfville
QUINT (typed)	Wolfville
QUINT (untyped)	Canada, Nova Scotia, Wolfville
Question	*"what language does cuba speak?"*
Gold	Spanish Language
QUINT (typed)	Spanish Language
QUINT (untyped)	Spanish Language

Table 3.5: Anecdotal results from WebQuestions for both variants of QUINT: typed and untyped.

system and AQQU++ where we i) manually decompose each complex question into its constituent sub-questions, ii) answer each sub-question using Bast and Haussmann-basic, and iii) run our stitching mechanism on the answer sets of sub-questions to answer the complete question.

QUINT achieves an F1 of 49.2, outperforming the other two systems. The difference between QUINT and AQQU++ comes from the difference between the two systems already observed in Table 3.4. In both cases, this shows the effectiveness of our procedure for handling complex question detailed in Section 3.5. When looking at the results of AQQU, we see that this system is capturing one sub-question at most, to the detriment of its precision. For the first sample complex question shown in Table 3.7, AQQU captured only one sub-question (movie starred Woody Harrelson) leading to low precision, while QUINT correctly answered it. For the second question, capturing one sub-question is sufficient. QUINT failed on the third question since it could not stitch the answer sets for the sub-questions. It was able to generate the correct query for the first

Method	Average F1
AQQU	27.8
AQQU++	46.7
QUINT	49.2

Table 3.6: Results on ComplexQuestions.

sub-question (*"everton"*) but not for the second (*"leeds"*). AQQU could not rank the correct query for the first sub-question (the only one it could capture) on top.

3.6.6 Discussion

A detailed analysis of the results on the WebQuestions test set reveals that of the 2032 test questions, 3 could not be matched to any of our templates. Another 33 test questions were matched to a template, but no query candidates were generated. The first cause of this issue is the incompleteness of our predicate lexicon, resulting in empty lookups. This suggests that we can benefit from extending our lexicon construction scheme. The second cause is incorrect dependency parse trees and POS tag annotations. For example, for *"what influenced william shakespeare to start writing?"*, the verb '*influenced*' was tagged as a noun, and therefore the question was mapped to templates which could not generate any suitable query. Methodological progress on mainstream parsing and POS tagging would positively affect our system.

For 260 test questions, some query candidates were generated, but none of them returned any correct answers. We identified mistakes made by the NERD system, missing entries in our lexicon $\mathcal{L}$ as well as wrong gold standard as the causes. This meant that QUINT could not generate the right query as input to the ranking stage to begin with. Another important cause for this issue is the lack of any appropriate templates for some questions. For example, for the utterance *"what countries are located near egypt?"*, none of the utterance templates that match this question are paired with the appropriate query template. In this case, we need a query template that connects Egypt to adjoining countries through a CVT node.

This leaves 1736 test questions for which QUINT generates at least one candidate query that returns at least one correct answer. It is also interesting to establish an upper bound on our performance with the above issues. For this, we created an oracle ranker that returns the generated query with the highest

Question	*"which movie starred woody harrelson and wesley snipes?"*
Gold	White Men Can't Jump, Wildcats, Money Train
QUINT	White Men Can't Jump, Wildcats, Money Train
AQQU	White Men Can't Jump, Wildcats, Rampart, Cheers, Money Train, ...
Question	*"what movie included characters named louis tully and dr peter venkman?"*
Gold	Ghostbusters
QUINT	Ghostbusters
AQQU	Ghostbusters
Question	*"which players have played for everton and leeds?"*
Gold	Ross Barkley
QUINT	-
AQQU	Goodison Park

Table 3.7: Sample ComplexQuestions for both QUINT and AQQU (Bast and Haussmann, 2015).

F1 for each question. The result was an F1 of 70.0, which means that further improvement to our ranking scheme is possible.

On Free917, a detailed analysis reveals that of the 276 test questions, 3 could not be matched to any of our templates. Another 31 test questions were matched to a template, but no query candidates were generated. For 13 questions, some query candidates were generated, but none of them returned any correct answer. This leaves 229 test questions for which QUINT generates at least one candidate query that returns at least one correct answer.

On ComplexQuestions, we could answer 81 of 150. The remaining 69 questions were not answered because either one of the sub-questions did not have its correct query in the top-ranked ones, or our templates failed to capture some sub-questions.

4 Never-Ending Learning for Open-Domain Question Answering over Knowledge Bases

4.1 Introduction

Open-domain question answering over knowledge bases (KB-QA) is an active research area where the goal is to provide crisp answers to natural language questions (Abujabal et al., 2017; Bast and Haussmann, 2015; Berant et al., 2013; Fader et al., 2014; Savenkov and Agichtein, 2016; Yahya et al., 2013, 2016; Yin et al., 2015) or telegraphic queries (Sawant and Chakrabarti, 2013; Joshi et al., 2014). An important direction in KB-QA performs this answering via semantic parsing: translating a user's *question* to a SPARQL *query* that is subsequently executed over a KB like Freebase (Bollacker et al., 2008), DBPedia (Auer et al., 2007) or YAGO (Suchanek et al., 2007). Existing approaches rely on a clear separation between an offline training phase, where a model is either learned or manually crafted, and an online phase where this model is deployed to answer users' questions. Such approaches suffer from three major shortcomings: (i) they require access to reasonably large training sets with sufficient syntactic and lexical coverage representative of the kinds of questions users pose, which are expensive to construct, (ii) they provide no mechanism for improving their performance over time by learning from failure cases on questions received after deployment, and (iii) they are limited to the language learned at training time, therefore, they fail on questions from domains not observed previously.

In this work, we present a continuous-learning framework for template-based KB-QA called NEQA (*N*ever *E*nding *QA*) that (i) is initialized with a *small* training set, (ii) improves its performance over time by judiciously invoking user feedback on answers from non-expert users on the failure cases of the underlying template-based answering mechanism, and (iii) adapts to the language used after deployment by periodically retraining its underlying models. A simplified workflow is shown in Figure 4.1.

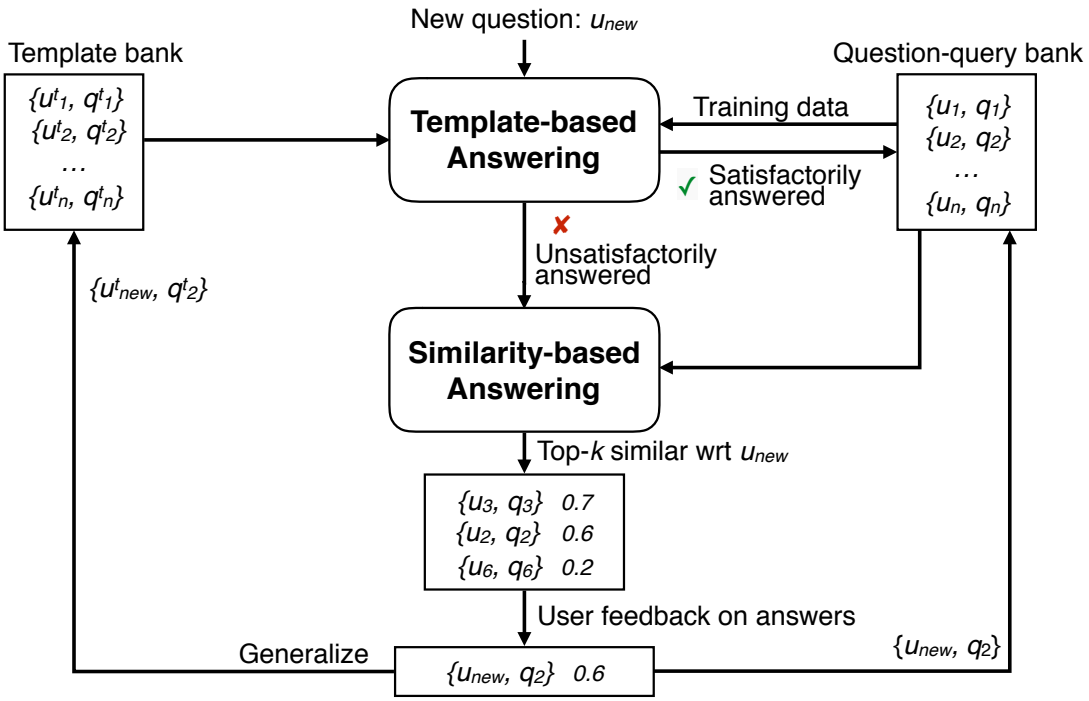


Figure 4.1: Continuous learning: if a new question u_{new} cannot be satisfactorily answered via templates, we utilize user feedback on the output of a semantic similarity function to learn a new template (u^t_{new}, q^t) based on u_{new}.

Training a well-performing open-domain KB-QA system requires a massive annotation effort, in terms of cost, time and expertise. Some methods use labeled SPARQL queries (Cai and Yates, 2013), while others train their systems on question/answer pairs as a form of weak supervision (Bast and Haussmann, 2015; Berant et al., 2013), which has been proven to work well. We adopt this form of supervision, however, for only a small training seed to minimize the annotation effort required. We rely on non-expert user feedback to acquire more question/answer pairs over time. In our experiments, we show that NEQA was able to successfully answer questions from domains it has not seen before.

We harness non-expert user feedback on answer sets generated as a response to a given question, which is related to a number of recent ideas in semantic parsing and natural language interfaces to databases (NLIDB). Li and Jagadish (2014) invoke user feedback to resolve ambiguous words/phrases in the users' questions, while Iyer et al. (2017) ask expert users to provide a full SQL query that answers a question over a database. In Wang et al. (2016), an end user teaches the model new concepts through direct interaction. Other approaches

utilize crowdsourcing as a sort of interactivity (Wang et al., 2015; Werling et al., 2015).

NEQA builds on an established line of work that performs semantic parsing by translating syntactic dependency structures of utterances to semantic predicate-argument structures using templates that are either manually crafted (Fader et al., 2013, 2014; Reddy et al., 2016; Unger et al., 2012; Yahya et al., 2013), or automatically learned (Abujabal et al., 2017). By exploiting syntax, such template-based approaches achieve better generalization (Bender et al., 2015). As an example, such systems can use a template generated from u_1 = *"which film awards was bill carraro nominated for?"* to answer the syntactically isomorphic question *"which president was lincoln succeeded by?"*, despite the fact that it invokes a different semantic KB predicate.

The main drawback of such systems is their inability to handle new syntactic structures beyond those observed in the static training set. Take, for example, a new question u_{new} = *"what are the film award nominations that bill carraro received?"*. Even if the above systems had seen u_1 during training, they cannot answer this new semantically related (but syntactically different) question. This problem is exacerbated if these systems are trained on a small number of training examples. NEQA rectifies this limitation by using a state-of-art similarity function (Zhang et al., 2016) to find a correctly answered and semantically-similar question from its history, and subsequently learns a new template based on the new question.

4.1.1 Overview

NEQA is driven by two intuitions. First, syntactic isomorphism of questions is a strong cue for the isomorphism of their respective predicate-argument structures (SPARQL queries). This intuition underlies template-based approaches outlined above. Second, where syntactic isomorphism fails, NEQA invokes a semantic similarity function together with user feedback to transfer semantics across syntactic structures and triggers the learning of new templates. NEQA combines these intuitions into a *continuous learning framework* that gradually overcomes the limitations of small training sets and evolves over time.

NEQA starts by automatically learning a few templates from a small number of questions offline, using our approach outlined in Section 3.3. Figure 4.1 shows what happens when NEQA receives a new question u_{new} online. It adds satisfactorily answered questions to an ever-growing bank of question-query (u, q) pairs, initially composed of a small training set. Questions in this bank will be called upon when our template-based answering mechanism fails to satisfactorily

answer a new question. In such cases, NEQA learns new syntactic structures to improve its future answering performance.

When u_{new} is unsatisfactorily answered using our template-based answering mechanism, NEQA triggers the learning of a new template from this question. It first consults a semantic similarity function to find the k previously answered questions closest to u_{new}. NEQA then instantiates the corresponding queries with entities from u_{new}. Leveraging user feedback on answer sets generated by executing these queries over the KB, one of the resulting queries (q_2 in Figure 4.1) is determined to be the best fit for u_{new}. NEQA then uses a lexicon and an Integer Linear Program to align the constituents of u_{new} and q_2. A new template (u^t_{new}, q^t_2) is created from this pair, which is then added to the template bank.

Question and Query Templates

Templates play an important role in KB-QA (Abujabal et al., 2017; Fader et al., 2014; Reddy et al., 2016; Unger et al., 2012). They guide the mapping of the syntactic structures of natural language utterances to semantic predicate-argument structures in SPARQL queries. Figure 4.3 shows an example template. It consists of a question template u^t and its corresponding query template q^t, where u^t and q^t are derived from generalizations over the dependency parse and the query, respectively. Alignment of the constituents of u^t and q^t is indicated by shared *ent*, *pred*, and *class* annotations.

Generating templates. We follow our approach outlined in Section 3.3 (Abujabal et al., 2017)) for learning templates. The approach is designed for a weakly supervised setting where a training instance is a question u paired with its answer set A_u (Berant et al., 2013). NEQA uses this form of supervision for the initial training phase. The approach heuristically generates a query q that captures each training question u from the corresponding training pair (u, A_u). For $u =$ *"Which film awards was Bill Carraro nominated for?"*, the corresponding query would be $q =$ `BillCarraro nominatedFor ?x . ?x type movieAward`. We now have a question query pair (u, q). The rest of the discussion explains how a template is generated from such a pair. This process is invoked in NEQA both as part of initial training (where we start with (u, A_u) pairs), and during continuous learning, where NEQA generalizes a (u, q) pair resulting from the similarity function and user feedback to a template (Figure 4.1).

Next, nodes in the dependency parse of u are aligned with semantic items in q. A dependency parse is a tree whose nodes correspond to words in a sentence and edges represent grammatical relations between words. We use the Stanford dependency parser (Chen and Manning, 2014) in this work. For example,

'*film awards*' in u above is aligned with the KB class `movieAward` in the corresponding q. A weighted *lexicon* L (Section 2.2) is used to connect phrases in u to all candidate semantic items in q, forming a weighted bipartite graph. The alignment problem is formulated as a constrained optimization problem solved using an Integer Linear Program (ILP) 3.3.3. The solution to the ILP is a role-aligned question-query pair (Figure 4.2) where phrases in u that are not part of the alignment are dropped (e.g., '*Which*'). Finally, concrete values in both u and q are dropped to produce a role-aligned question-query template pair (u^t, q^t) (Figure 4.3).

Using templates. During answering, when a new question u_{new} is encountered, question templates matching its dependency parse are identified (see Section 4.3.2). Corresponding query templates are then instantiated using alignment information and the lexicon (Section 2.2). This step potentially generates multiple query candidates due to lexicon ambiguity, which are ranked using a learning-to-rank (LTR) model (Section 3.4.2). Finally, the answer of the top-ranked query is presented to the user.

This work presents NEQA, the first continuous learning framework for KB-QA, and make four novel contributions:

- a KB-QA system that can be seeded with a small number of training examples and supports continuous learning to improve its answering performance over time;
- a similarity function-based answering mechanism that enables NEQA to answer questions with previously-unseen syntactic structures, thereby extending its coverage;
- a user feedback component that judiciously asks non-expert users to select satisfactory answers, thus closing the loop between users and the system and enabling continuous learning;
- extensive experimental results on two benchmarks demonstrating the viability of our continuous learning approach, and the ability to answer questions from previously-unseen domains.

4.2 Related Work

Question answering. KB-QA has seen broad interest in recent years with the wide availability and rapid growth of KBs and voice-based interaction with devices (e.g., Alexa, Cortana). We adopt a template-based approach for mapping syntactic structures to semantic predicate-argument structures (Reddy et al.,

2016; Unger et al., 2012; Yahya et al., 2013; Zheng et al., 2015; Zou et al., 2014). Another way of exploiting syntax is to use grammatical formalisms that derive syntax and semantics in tandem, most prominent among these being CCGs (Cai and Yates, 2013; Kwiatkowski et al., 2013). Some techniques disregard syntax altogether, and rely on combinatorial over-generation of queries followed by a ranking of such candidate queries (Bast and Haussmann, 2015; Berant et al., 2013; Yih et al., 2015; Yao and Durme, 2014). Finally, with the recent popularity of deep learning, some methods use large amounts of training data to learn a function for embedding questions and answer entities in a shared latent space (Bordes et al., 2015; Yang et al., 2014). We opted to base NEQA on the systems that use syntax as they achieve better generalization (Bender et al., 2015), allowing the transfer of semantics between syntactic structures. We rely on dependency parsing for capturing syntax to exploit the rapid progress on this task (Chen and Manning, 2014). In contrast to all the above, our system uses continuous learning that allows starting from small training sets and improving over time.

The framing of the QA task depends on the type of the underlying data and associated annotations. An important QA setting is answering over textual corpora. One way to approach this is using traditional IR methods to retrieve relevant documents and extract passages or phrases that answer the question (Dumais et al., 2002; Harabagiu et al., 2000; Brill et al., 2002; Ribarov, 2004). Another way has been to use OpenIE to turn such corpora into open-vocabulary knowledge bases and answer over these (Fader et al., 2013, 2014; Krishnamurthy and Mitchell, 2015). Finally, in a setting where both textual and structured data are used, hybrid approaches have been explored for QA (Sun et al., 2015; Xu et al., 2016a; Savenkov and Agichtein, 2016; Xu et al., 2016b; Yahya et al., 2016).

In *entity search* (Balog et al., 2011; Dalton et al., 2014; Blanco et al., 2011; Joshi et al., 2014; Tran et al., 2007; Sawant and Chakrabarti, 2013), the user searches for a list of entities using keyword-based queries (e.g., '*dutch artists paris*'). The underlying corpus for retrieval may be Wikipedia pages (Balog et al., 2011), documents with Freebase annotations (Sawant and Chakrabarti, 2013), or general RDF-stores (Blanco et al., 2011). Techniques vary from probabilistic language models (Blanco et al., 2011), query segmentation (Joshi et al., 2014), to category models for entities (Balog et al., 2011; Tran et al., 2007).

Continuous learning and user feedback. Our work draws inspiration from the never-ending learning paradigm (Mitchell et al., 2015) and its use case NELL (Never-Ending Language Learning) in machine reading (Carlson et al.,

2010). NEQA also leans on the principle of *online learning* (Kivinen et al., 2004) where incoming questions are fed into the system in a sequential order, thus improving the system's performance over time.

User feedback has always been vital for IR systems: be it solely for evaluation as relevance judgments in the early days (Voorhees, 2001), or in more implicit forms like clicks (Joachims, 2002) and reformulations (Radlinski and Joachims, 2005) for improving personalized ranking models (Agichtein et al., 2006a,b; Bendersky et al., 2017) and automatically completing queries (Li et al., 2017b). User interactions play a key role in closing the loop in a continuous learning framework, where they improve the system iteratively. In the NELL system (Mitchell et al., 2015), feedback is incorporated as periodic expert judgment on extracted beliefs. Recently, user feedback was leveraged on graph queries, and evaluated with simulated judgments (Su et al., 2015). User feedback has been leveraged in natural language interfaces to databases (NLIDB), where Li and Jagadish (2014) invoke user feedback to resolve ambiguous words/phrases in the users' questions, while Iyer et al. (2017) ask expert users to provide a full SQL query that answers a question over a database.

Question retrieval. NEQA relies on question retrieval to drive continuous learning when templates fail, where it looks for previously answered questions most similar to the current one. This is a central task in community question answering (CQA) (Jeon et al., 2005; Chen et al., 2016; Zhang et al., 2016; Zhou et al., 2015; Wang et al., 2009; Zhang et al., 2014), where the goal is to answer a user's question by presenting answers to similar questions that have already been answered. Various methods have been proposed, including those that use syntactic parse trees (Wang et al., 2009) and language models (Zhang et al., 2016). Our method takes inspiration from the latter work.

Notions of similarity have been leveraged in KB-QA systems based on paraphrasing. Berant and Liang (2014) use a supervised paraphrasing model that finds the logical form whose machine-generated verbalization best paraphrases the input question. Fader et al. (2013, 2014) perform QA over an open-predicate KB by learning a paraphrase model for rewriting a question to a set of similar canonical question forms, each of which maps to a unique query.

4.3 The NEQA Framework

Initially, NEQA goes through an offline training stage that populates its question-query and template banks with their seed data (Section 4.3.1). Online, when NEQA is deployed, a stream of questions arrives from users. NEQA attempts

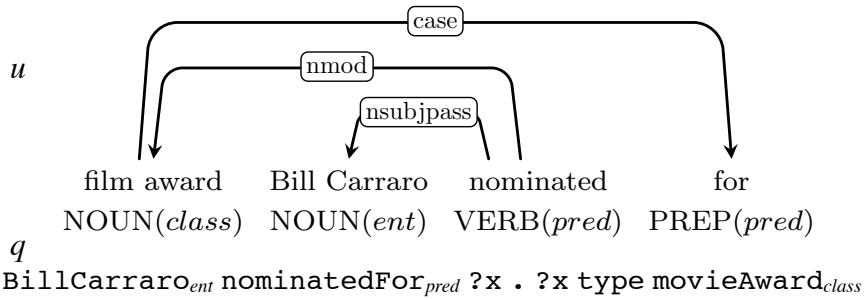


Figure 4.2: An aligned question-query pair (u, q). Alignment is indicated by shared *ent*, *pred*, and *class* annotations.

to answer each incoming question using the templates it has learned so far (Section 4.3.2). If this fails, it falls back to answering using the semantic similarity function against the set of already answered questions in the question-query bank (Section 4.3.3). In both cases, NEQA utilizes user feedback on answer sets to extend its banks (Section 4.3.4). After each *batch* of questions, NEQA retrains its learning-to-rank (LTR) ranking model on the accumulated data in its banks to improve system performance for subsequent questions.

4.3.1 Initial Training

NEQA is initialized through an automated template generation stage. This stage relies on weak supervision through a small number of questions, each paired with its answer set. This training stage results in populating the question-query and template banks (Figure 4.1) with their seed data that are used for bootstrapping the continuous learning process. Moreover, it results in NEQA's first LTR model. NEQA's continuous learning improves all three components once the system goes online.

Questions in the question-query bank are stored in a generalized form that facilitates improved matching by the semantic similarity function. Specifically, entities in both questions and queries in the question-query bank are replaced by placeholders. For example, *"Which film award was ENTITY nominated for?"* (question) is paired with `ENTITY nominatedFor ?x . ?x type movieAward` (query).

4.3.2 Question Answering with Templates

Once the system goes online, it starts receiving new question utterances from users. Given a new question u_{new}, NEQA identifies matching question templates $\{u^t\}$ in its template bank. A match is deemed successful if edge labels and

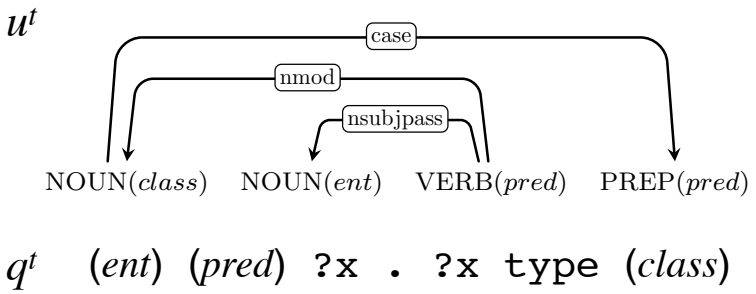


Figure 4.3: A template composed of aligned question and query templates (u^t, q^t). Shared *ent*, *pred*, and *class* annotations indicate alignment between u^t and q^t.

POS tags in the dependency parse of u_{new} and u^t agree. For example, the dependency parse of *"which president was lincoln succeeded by?"* matches the utterance template in Figure 4.3. When this happens, the associated query template q^t is instantiated with concrete semantic items using the phrases in u_{new}, the alignment information between u^t and q^t, and the underlying lexicon L. For example, based on the alignment information in Figure 4.3, the verb phrase '*succeeded by*' is used to instantiate a KB predicate.

Note that a single utterance may match multiple utterance templates, and these templates may result in multiple queries due to ambiguity in L. The learning-to-rank (LTR) model is used to rank this set of candidate queries. Features for training the LTR model are borrowed from past work (Abujabal et al., 2017; Bast and Haussmann, 2015), and are derived from lexicon weights, entity popularity scores, answer type constraints, and sizes of answer sets, among others. The full list of features is depicted in Table 3.3 in Section 3.4.2. The top-ranked queries, as detailed below, are then executed over the KB to fetch answer sets.

Next, user feedback is obtained on these retrieved answer sets. To be realistic, note that we obtain feedback on answer sets of the top-k queries where k is small. If an answer set is chosen (e.g., {`AndrewJohnson`}) by the user, then this validates the choice of the query q^* that generated this answer set as correct (e.g., `AbrahamLincoln succeededBy ?x . ?x type president`). On the other hand, when none of the shown answer sets is chosen, NEQA proceeds differently (Section 4.3.3). Finally, the correct query q^* is paired with u_{new} and is then added to our question-query bank after entity generalization. For the example above, q^* after entity generalization is: `ENTITY succeededBy ?x . ?x type president`. Such an augmentation of the question-query bank potentially results in the system

gaining questions with unseen KB predicates. We validate this postulate in our experiments on open-domain answering. When a batch of questions has been received, we use the questions answered satisfactorily by the templates to retrain the LTR model to further boost the performance of the system on subsequent questions.

4.3.3 Question Answering via Similarity Function

A core contribution of NEQA is to extend coverage of template-based answering using a semantic similarity function. A typical template-based KB-QA system fails when an input utterance represents previously unseen syntactic structure (Abujabal et al., 2017; Fader et al., 2013; Unger et al., 2012; Yahya et al., 2013). Further, even when a matching utterance template is identified, the KB-QA system might fail to deliver answers due to errors in the alignment information between the question and the query templates.

NEQA, on the other hand, builds on failure cases to improve its future QA performance. Whenever a question cannot be answered satisfactorily using templates, NEQA uses a *semantic similarity function* to retrieve the k most semantically similar questions to u_{new} from its question-query bank. For example, say, the utterance u_{new} = *"what are the film award nominations that bill carraro received?"* represents a syntactic structure beyond the coverage of our current templates. However, our question-query bank contains a similar question: *"which film awards was bill carraro nominated for?"*. The goal of our similarity function is to identify such questions and allow the *transfer of semantics* across syntactic structures.

We first use an off-the-shelf NERD system to link mentions of entities in u_{new} to KB entities (Yang and Chang, 2015). Identified entities in u_{new} are then replaced by placeholders to ensure better generalization. Similar generalization is also done on (u, q) pairs in our question-query bank (Section 4.3.1). Next, the corresponding queries $\{q_1 \dots q_k\}$ for these similar utterances are instantiated with entities from u_{new} and then executed over the KB to retrieve answer sets.

Next, we obtain user feedback on the answer sets of the k queries. If an answer set is chosen (e.g., {`BlackReel`}), the corresponding query q^* (e.g., `BillCarraro nominatedFor ?x . ?x type movieAward`) is paired with u_{new}. The newly generated pair (u_{new}, q^*) is then added to our question-query bank after entity generalization. A vital step of NEQA is the subsequent on-the-fly alignment and generalization of u_{new} and q^*, to obtain a new template (u^t, q^t). This is performed by casting the problem as an integer linear program (Section 3.3.3). The new template (u^t, q^t) is then added to NEQA's template bank. By acquiring more

templates, the system's capability to handle syntactic variation increases, i.e., it learns how to *directly* answer questions with new syntactic structures.

Similarity Function

Following recent work on *question retrieval* in community question answering (Zhang et al., 2016), we opt for an unsupervised semantic similarity function. Note that we treat the similarity function as a plug-in, where supervised methods can also be used if required. Our similarity function consists of two components: (i) question likelihood based on a language model, and (ii) word embedding-based similarity obtained through *word2vec*.

Given a new question u_{new} and a question u_i from our question-query bank, our first component, based on language model, computes question likelihood as follows:

$$score_{LM}(u_{new}, u_i) = \prod_{w \in u_{new}} [(1-\lambda) \cdot P_{ml}(w|u_i) + \lambda \cdot P_{ml}(w|C)] \tag{4.1}$$

where $P_{ml}(w|u_i)$ represents the maximum likelihood probability estimate of w estimated from u_i and w is a unigram, bigram or trigram generated directly from u_{new} or from paths of lengths one and two in the dependency parse of u_{new}. $P_{ml}(w|C)$ is a smoothing term calculated as the maximum likelihood of w in a corpus C of questions from our question-query bank, and $\lambda \in [0, 1]$ is a smoothing parameter.

The second component uses a *word2vec* model pre-trained on Google News corpora (Mikolov et al., 2013):

$$score_{w2v}(u_{new}, u_i) = \frac{1}{|\mathcal{P}|} \sum_{(w_j, w_k) \in \mathcal{P}} cos(w2v(w_j), w2v(w_k)), \tag{4.2}$$

where, $w_j \in u_{new}$, $w_k \in u_i$, $w2v(w)$ is the *word2vec* embedding vector of w, and $\mathcal{P}$ is the set of word pairs from u_{new} and u_i whose cosine similarity is above a threshold τ.

The final score is a linear combination of the two components presented above, where α is a trade-off parameter:

$$score_{sim}(u_{new}, u_i) = \alpha \cdot score_{LM}(u_{new}, u_i) + (1-\alpha) \cdot score_{w2v}(u_{new}, u_i) \tag{4.3}$$

4.3.4 Harnessing User Feedback

NEQA resorts to user feedback in two cases. The first is when an incoming utterance u_{new} is answered using templates in the template bank. In this case,

the user is asked to give feedback on the relevance of the answer sets shown to her by either choosing the one that satisfies her information needs or none of them, if none is satisfactory. By propagating answer quality back to queries, this feedback is leveraged to extend the question-query bank. The second case is when NEQA returns answers using the semantic similarity function. The answers obtained from the top-k previously answered questions that are most similar to u_{new} are shown to the user for assessment. This feedback is used to extend both the template and the question-query banks.

In both cases above, it is important to keep k small to ensure the feasibility of asking for user feedback. In the experiments, we show that this is the case for our choices of LTR and semantic similarity functions. Additionally, we look at the extreme case where $k = 1$ and user feedback is bypassed by making the assumption that answers returned by our system are correct, and can be used for continuous learning.

4.4 Experimental Evaluation

We present extensive experimental evaluation and analysis of continuous learning in NEQA. Our experiments demonstrate NEQA's ability to continuously improve its answering performance over time starting with a very limited training set. We show that the answering performance of traditional state-of-the-art QA systems where periodic re-training is done is inferior to that of NEQA, which was designed specifically to support continuous learning. We also show that the manner in which NEQA exploits the interaction between syntax and semantics allows it to support truly open-domain QA by answering questions requiring predicates it has not seen before.

4.4.1 Benchmark

We use the following KB-QA benchmarks over Freebase to evaluate NEQA:

- **WebQuestions** (Berant et al., 2013): This benchmark was created using Google's suggest API and crowdsourcing, and is composed of 5810 questions, each paired with its answer set. These are split into 3778 training and 2032 test instances.
- **ComplexQuestions** (Bao et al., 2016): This very recent benchmark focuses on more challenging multi-constraint questions. It contains 2100 question-answer pairs from (i) a commercial search engine, (ii) WebQuestions and,

Property	WebQuestions	ComplexQuestions
Size of initial training set	300	105
Size of development set	300	300
Initial templates acquired	223	85

Table 4.1: NEQA initialization statistics.

(iii) the benchmark released by Yin et al. (2015). It is split into 1300 training and 800 test cases.

We base our extended analyses and comparisons with baseline systems on WebQuestions due to the lack of publicly available KB-QA systems designed for handling complex questions in ComplexQuestions. As detailed below, small subsets of the respective training sets are used for initial training, followed by streaming the complete test sets in batches to simulate online answering and continuous learning.

4.4.2 Training

Table 4.1 gives a summary of the initial training stage. A main motivation for resorting to continuous learning is the cost associated with obtaining a large training set upfront. To simulate small seed training sets, we randomly sample only about 8% of the standard WebQuestions and ComplexQuestions training sets. These seed training sets were used to initialize the (i) question-query and template banks (Section 4.3.1), (ii) learning-to-rank (LTR) models (Section 4.3.2), and (iii) language model component of the similarity function (Section 4.3.3). The development sets (randomly sampled from the training set) were used to tune the λ and α parameters of the similarity function. Crucially, NEQA is never exposed to the full WebQuestions or ComplexQuestions training sets in our experiments beyond the above seed examples. After initial training, NEQA is deployed to answer incoming questions, performing continuous learning when necessary.

Continuous learning. During answering, NEQA receives test questions from the respective benchmark in batches. At the end of each batch, we retrain the LTR component and re-estimate the language model with the data seen thus far. We set our batch size to 100 questions, so that we can observe the effect of continued learning over a larger number of batches (20 for WebQuestions, 8 for ComplexQuestions). Varying batch size did not have any significant effect on

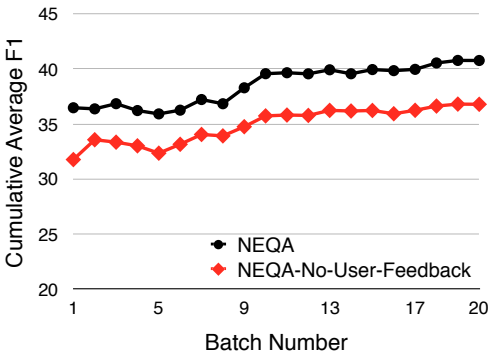
the observed trends.

We report system performance in two modes, which differ in their invocation of user feedback during continuous learning:

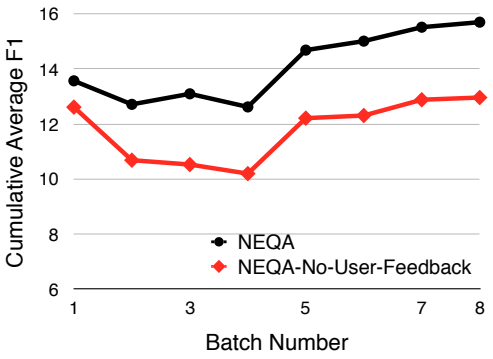
- **NEQA** (with user feedback): Here, we invoke user feedback to select the most appropriate answer set among the top-k answer sets obtained using templates and, if none are appropriate, among the top-k obtained using the similarity function. We use gold answer labels provided with the benchmarks to simulate user feedback. We use $k = 5$ in both cases, as we find that it provides a good balance between recall and the number of answer sets a user needs to look at.
- **NEQA-No-User-Feedback**: In this configuration, we perform continuous learning without user feedback. Instead, we take the top-ranked answer set in either of the two lists of answer sets above to be the correct one. We consider the list of answer sets obtained using the similarity function only if the list obtained using templates (ranked by the LTR function) is empty. This configuration demonstrates the quality of our template-based and similarity function-based answering mechanisms and helps understand the gap filled by user feedback.

Method	**Avg. Precision**	**Avg. Recall**	**Avg. F1**
QUINT (Abujabal et al., 2017) - No Feedback	25.5	30.2	25.7
QUINT (Abujabal et al., 2017) - Feedback	35.2	44.1	35.9
AQQU (Bast and Haussmann, 2015) - No Feedback	24.5	29.6	24.8
AQQU (Bast and Haussmann, 2015) - Feedback	36.3	45.2	37.6
NEQA-No-User-Feedback	36.6	45.4	37.0
NEQA	**40.6**	**49.5**	**40.8**

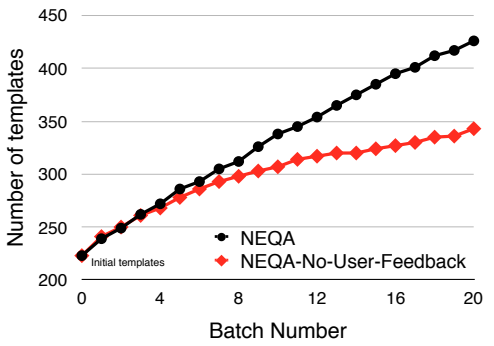
Table 4.2: Performance of continuous learning-based methods on the WebQuestions test set. User Feedback is used to re-train the systems after each batch.



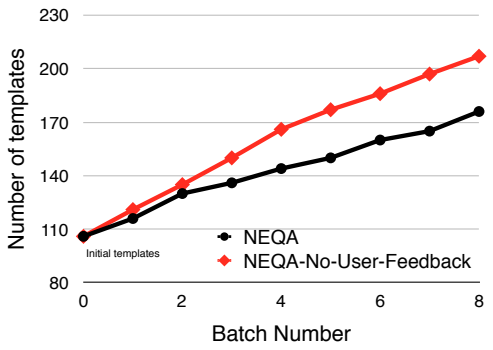
(a) F1-scores (WebQuestions)



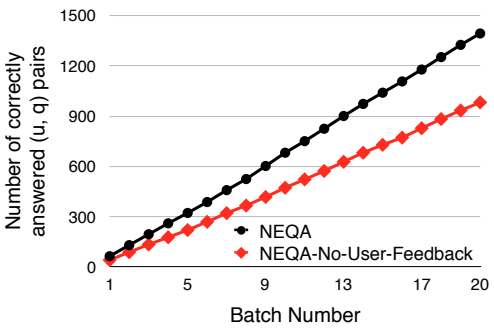
(b) F1-scores (ComplexQuestions)



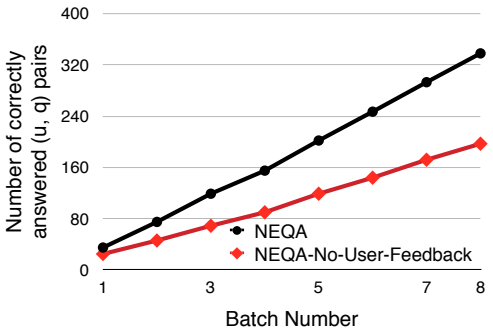
(c) Templates acquired (WebQuestions)



(d) Templates acquired (ComplexQuestions)



(e) Correct (u, q) pairs acquired (WebQuestions)



(f) Correct (u, q) pairs acquired (ComplexQuestions)

Figure 4.4: Performance of NEQA with and without user feedback as a function of batch number, on the WebQuestions and ComplexQuestions datasets.

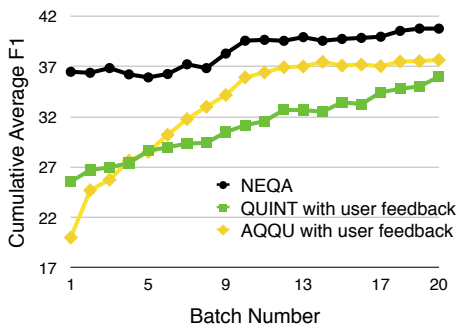
4.4.3 Results

Answering performance over time. Figure 4.4 shows how NEQA performs after deployment, as it receives user questions and invokes continuous learning where necessary. The cumulative average F1 scores in Figures 4.4a and 4.4b show an improvement over time for both benchmarks and for both feedback configurations. We compute the F1 score of a given batch after observing all questions in that batch, but before updating our ranking models based on that batch. The cumulative F1 score at batch n is the mean of these scores over all n batches. On both benchmarks, the general trend is for the cumulative F1 to increase as NEQA sees more questions and invokes continuous learning as needed to learn new templates and improve its ranking model. In general, the numbers on ComplexQuestions are lower due to its more challenging nature stemming from multi-relation questions. We can see some fluctuation in the initial batches for both datasets. We attribute such variations to the modest ranking performance of the underlying LTR model during the very first iterations due to the small number of instances it was trained on. As expected, NEQA's F1 increases significantly when user feedback is invoked. We observe an F1 increase of 3.8 and 2.8 points over the no-feedback configuration for WebQuestions and ComplexQuestions, respectively.

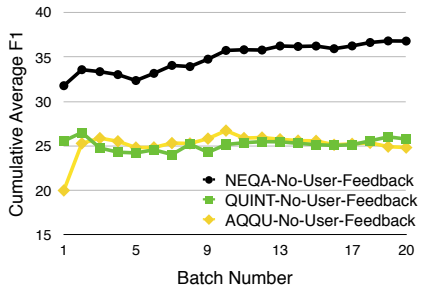
Augmentation of banks. NEQA extends its banks with new templates and question-query pairs over time. Figures 4.4c and 4.4d show the number of templates learned over time. Each template captures a distinct syntactic structure and its mapping to the appropriate semantic predicate-argument structure. Each template in the template bank corresponds to one or more question-query pairs in the question-query bank: it was either generated from such a pair during training or continuous learning, or was used to answer u in that pair by mapping it to q. In general, having more correct (u, q) pairs in the question-query bank means: (i) NEQA has learned more correct templates, and, (ii) NEQA can better transfer these new templates to new syntactic structures with semantics similar to that of q. Figures 4.4e and 4.4f show the numbers of *correctly answered* new (u, q) pairs for WebQuestions and ComplexQuestions, respectively with the two modes of feedback. A (u, q) pair is deemed correct if the gold answer set of u overlaps with the answer set of q when executed over the KB. For 1393 out of 2032 test questions in WebQuestions (338 out of 800 in ComplexQuestions), user feedback indeed yielded the ground-truth answer set.

Contrasting these figures gives interesting insights. For WebQuestions, the number of templates obtained from user feedback is higher than that obtained without. This is because the similarity function does a good job at surfacing

the correct answer set to the top-k from which the user selects the correct one. However, without feedback, the top-1 may be incorrect, in which case alignment between the corresponding query and the question at hand fails (as opposed to generating a spurious alignment), resulting in no templates. The question-query bank in the configuration with feedback contains more correct (u, q) pairs, meaning that we have more correct alignments (and hence less spurious templates). When looking at ComplexQuestions, we see that the no-feedback configuration results in more templates. The reason here is that with the complexity of the dataset and the limitations of the similarity function, users are more likely to decide that no answer set produced through the similarity function is appropriate. In the no-feedback case, the topmost answer set from the similarity function is always chosen, and alignments (including spurious ones) are more likely here due to the length of the questions. Despite the large number of templates for the no-feedback configuration, we can see that NEQA with user feedback results in more correct (u, q) pairs (Figure 4.4f), indicating that the no-feedback configuration has more spurious templates than the one with feedback, as expected.



(a) Harnessing user feedback.



(b) No user feedback. The top-1 query is deemed correct.

Figure 4.5: Performance of AQQU (Bast and Haussmann, 2015), QUINT (Abujabal et al., 2017) and NEQA over the 20 batches of the WebQuestions test set.

Comparison with state-of-the-art. An intuitive baseline for evaluating continuous learning in NEQA is to extend existing static-learning based methods to our setting of continuous learning through user feedback and periodic retraining. Concretely, we trained both QUINT (Abujabal et al., 2017), and AQQU (Bast and Haussmann, 2015) on the same initial training seed as NEQA (300 question-answer pairs for WebQuestions). Then, we streamed the 2032 test questions in batches of size 100, where user feedback is harnessed on the top-5 answer sets generated by the two systems. After each batch, the baseline sys-

tems were re-trained using the initial seed plus those questions from the batches seen thus far. Additionally, similar to NEQA-No-User-Feedback configuration, we performed continuous learning without user feedback for the two baselines.

Aggregate results over the WebQuestions test questions are shown in Table 4.2, where NEQA outperformed both QUINT and AQQU in the two modes of operation (with and without user feedback). To gain more insights, the per-batch cumulative average F1 for the three systems in both configurations is depicted in Figures 4.5a and 4.5b. NEQA starts off with a high F1 score (36.5) in batch one, compared to AQQU and QUINT with F1 scores of 20.0 and 25.5, respectively. This is due to two reasons: (i) NEQA learns templates online and adds them directly to the template bank, while QUINT, for example, learns new templates after each batch and (ii) when a question has no matching templates, our similarity function is invoked and might be able to correctly answer the question at hand, and hence, positively affects the performance. AQQU disregards the syntax of questions and relies on three query templates with exhaustive instantiation. This makes the underlying ranking module of AQQU very crucial to the overall performance, which explains the low values for the very first batches (the ranking module was trained on a relatively small number of training instances). When user feedback is bypassed, the performance of the baselines deteriorates, as shown in Figure 4.5b, where the two baselines were not able to improve their performance over time. On the other hand, NEQA showed improvement in performance over time. We attribute this to our *unsupervised* similarity function, which is used to answer questions when the template-based answering mechanism fails and subsequently trigger the learning of new templates with new syntactic structures. The similarity function plays a vital role in distinguishing our system, built with continuous learning in mind, from systems where continuous learning is achieved through simple periodic retraining.

For completeness, we also show results for NEQA when used as a traditional static-learning based QA system, with distinct training and testing phases and with *continuous learning disabled*. Table 4.3 shows the results for NEQA and baseline systems on the WebQuestions test set after training each on the full WebQuestions training set. The results show that NEQA achieves competitive results, with no significant differences from the best system, but with the added advantage of being able to perform continuous learning when limited training data is available. As described in Sections 4.1.1 and 4.3, NEQA is a continuous-learning extension of Abujabal et al. Abujabal et al. (2017), hence when used in the classical QA setup it obtains the same results as that work.

Open-domain question answering. NEQA exploits the interaction be-

Method	Avg. Precision	Avg. Recall	Avg. F1
Berant et al. (2013)	48.0	41.3	35.7
Yao and Durme (2014)	-	-	33.0
Bordes et al. (2014)	-	-	39.2
Yao (2015)	-	-	44.3
Bast and Haussmann (2015)	49.8	60.4	49.4
Yin et al. (2015)	**52.8**	**60.7**	**52.5**
Reddy et al. (2016)	-	-	50.3
Savenkov and Agichtein (2016) (w/o text)	49.8	60.4	49.4
Xu et al. (2016b) (w/o text)	-	-	47.1
Abujabal et al. (2017)	52.1	60.3	51.0
NEQA	52.1	60.3	51.0

Table 4.3: Performance of state-of-the-art static learning-based methods on the WebQuestions test set.

tween syntax and semantics to perform continuous learning on questions coming while the system is deployed. This interaction allows NEQA to perform truly open-domain question answering, where it answers questions that require previously-unseen semantic predicates. To test how well NEQA performs on this task, we restrict the test questions from WebQuestions to 693 questions whose corresponding query contains predicates from the following three domains: `sports`, `government` and `people`. Note that the domain information is encoded in the names of Freebase predicates (e.g., `sports.sports_team.championships`), allowing us to systematically make this restriction. We then removed the 56 questions with queries containing predicates from the above domains from the seed training set used for NEQA (resulting in $300 - 56 = 244$ new seed training examples). To provide a baseline, we used the best publicly available traditional KB-QA system of Bast and Haussmann (2015) as a baseline. We train this system on the standard WebQuestions training set with the 1315 questions from the three domains above excluded (a total of $3778 - 1315 = 2463$ training samples).

NEQA achieved an F1 score of 50.3 and 41.5 with and without user feedback, respectively, on the above test set. The system of Bast and Haussmann (2015), AQQU, had an F1 score of 20.3. The exhaustive instantiation of the three query templates with all possible KB predicates explains the F1 score achieved by AQQU. This experiment shows that NEQA, in both modes of feedback, answered questions from domains it had never seen during the initial training with a high F1 on par with the results obtained without filtering the seed training set. We attribute these gains to a combination of (i) using templates that account

for syntax and using lexicons for instantiating predicate-argument queries as a foundation for NEQA, (ii) re-training the underlying models using test questions which helps NEQA to adapt to the terminology of the new domain, and (ii) the extension of these methods to allow for continuous learning to account for new syntactic structures.

4.4.4 Discussion

Impact of templates and similarity function. We studied the two branches of NEQA individually: answering with templates, and via the similarity function when the basic answering with templates fails and continuous learning is triggered. Note that we cannot completely decouple both branches since the similarity function feeds our template-based answering with new templates over time, resulting in better coverage and performance. On WebQuestions, with user feedback, 1184 questions were answered with templates, while 848 were answered via the similarity function. For the no-feedback configuration, 1788 out of 2032 were handled by the learned templates, and the similarity function answered 244 questions. The contrast between 1184 and 1788 shows cases where feedback helped weed out erroneous answers obtained through templates.

There are various possible failure cases in our pipeline. During the very first batches, the LTR model had a modest ranking performance due to the small number of examples used to train it. However, as more questions were observed, the ranking performance of the LTR model improved substantially, especially when user feedback was harnessed. The impact of this was either incorrect answering, triggering continuous learning, or, once continuous learning was triggered, no answer sets in the top-k being correct, triggering answering via the similarity function. In some cases, none of the generated queries that were fed to the LTR model was correct to start with. This is explained either by the lack of appropriate templates, especially in early batches, or the incompleteness of the underlying lexicons used to instantiate our templates with concrete SPARQL queries. As future work, we plan to add textual resources to build better lexicons. In other cases, the NERD system failed to link mentions to the correct KB entities.

For some questions, our similarity function failed to retrieve semantically similar questions from our question-query bank. In some of these cases, our bank did not contain any question that is semantically similar to the question at hand. In other cases, although a correct similar question was retrieved, no new template was generated. Again, this is explained by deficiencies in our lexicons, or the NERD system we use.

Components	NEQA	NEQA-No-User-Feedback
Both	**40.8**	**37.0**
Only LM	38.3	35.1
Only word2vec	35.0	33.4

Table 4.4: F1 scores for a component ablation analysis of our similarity function.

"what is the name of the currency used in italy?"
"what is the head judge of the supreme court called?"
"where did the battle of waterloo occur?"

Table 4.5: Sample questions correctly answered via templates learned online.

Similarity function ablation study. Our similarity function consists of two components: (i) a language model (LM), and (ii) *word2vec* similarity. We conducted an ablation study to measure the effect of each component on the overall performance of the system. F1-scores are shown in Table 4.4. The highest F1 score is achieved when the two components are used. While the LM plays the most vital role, the word2vec component also contributes significantly to the final performance.

Anecdotal results. Table 4.5 shows sample test questions from the WebQuestions that were correctly answered using templates learned online. These questions represent new syntactic structures that NEQA learned online. Table 4.6 shows questions that were correctly answered using our similarity function together with the top-1 most similar question retrieved by the similarity function. For every pair of questions in this table, note the differing syntactic structures conveying similar semantics, e.g., '*what is the currency*' and '*what kind of money*'.

Question:	*"what is the currency in [italy?]"*
Most similar:	*"what kind of money is used in [israel]?"*
Question:	*"what films has [scarlett johansson] been in?"*
Most similar:	*"what movies did [zoe saldana] play in?"*
Question:	*"what was [sir isaac newton]'s inventions?"*
Most similar:	*"what inventions did [robert hooke] made?"*

Table 4.6: Sample questions correctly answered using the similar questions retrieved from the question-query bank. Entities are generalized using [...] placeholders.

5 Temporal Question Answering over Knowledge Bases

5.1 Introduction

Question answering over knowledge bases (KB-QA) aims to answer natural language questions over large knowledge bases (e.g., DBpedia, Wikidata, Yago, Freebase etc.) or other structured data. KB-QA systems take as input questions such as:

Q1: *"Which teams did Neymar play for?"*

and translate them into structured queries, in a relational language like SPARQL, and execute the queries to retrieve crisp answers from the KB. In doing so, KB-QA methods need to address the vocabulary mismatch between phrases in the input question and entities, classes and predicates in the KB: mapping '*Neymar*' to the uniquely identified entity, '*teams*' to the KB type `footballClub` and '*played for*' to the KB predicate `memberOf`. State-of-the-art KB-QA (see surveys (Diefenbach et al., 2018; Moschitti et al., 2017)) can handle simple questions like the above example very well, but struggle with complex questions that involve multiple conditions on different entities and need to join the results from corresponding sub-questions. For example, the question:

Q2: *"After whom did Neymar's sister choose her last name?"*

would require a three-way join that connects Neymar, his sister Rafaella Beckran, and David Beckham.

An important case of complex questions are temporal information needs. Search often comes with explicit or implicit conditions about time (Metzler et al., 2009). Consider the two examples:

Q3: *"Which teams did Neymar play for before joining PSG?"*

Q4: *"Under which coaches did Neymar play in Barcelona?"*

In Q3, no explicit date (e.g., August 2017) is mentioned, so a challenge is to detect its temporal nature. The phrase '*joining PSG*' refers to an event (Neymar's transfer to that team). We could detect this, but have to properly disambiguate it to a normalized date. The temporal preposition '*before*' is a strong cue as well,

but words like '*before*', '*after*', etc. are also used in non-temporal contexts; Q2 is an example for this. Q4 does not seem to be time-dependent at all, when looking at its surface form. However, it is crucial for correct answers that only coaches are selected whose job periods at FC Barcelona overlap with that of Neymar. Here, detecting the temporal nature is a big challenge. A second challenge is how to decompose such questions and ensure that the execution contains an overlap test for the respective time periods.

The key idea of this work is to judiciously decompose such temporal questions and rewrite the resulting sub-questions so that they can be separately evaluated by a standard KB-QA system. The answers for the full questions are then computed by combining and reasoning on the sub-question results. For example, Q3 should be decomposed and rewritten into

Q3.1: *"Which teams did Neymar play for?"* and

Q3.2: *"When did Neymar join PSG?"*.

For the results of Q3.1, we could then retrieve time scopes from the KB, and compare them with the date returned by Q3.2, using a BEFORE operator. Analogously, Q4 would require an OVERLAP comparison as a final step. With the exception of the work by Bao et al. (2016), to which we experimentally compare our method, we are not aware of any KB-QA system for such composite questions.

Our solution, called TEQUILA, is built on a rule-based framework that encompasses four stages of processing: (i) detecting temporal questions, (ii) decomposing questions and rewriting sub-questions, (iii) retrieving candidate answers for sub-questions, and (iv) temporal reasoning to combine and reconcile the results of the previous stage into final answers. For stage (iii), we leverage existing KB-QA systems that are geared for answering simple questions. In experiments, we use AQQU (Bast and Haussmann, 2015) and QUINT (Abujabal et al., 2017), but TEQUILA allows plugging in any other KB-QA engine.

The quality of QA is usually evaluated by benchmarks. As a first step towards addressing the challenge of handling temporal questions, we offer a new benchmark set of temporal questions. The questions are chosen such that many of them require a combination of evaluating sub-questions and reasoning over sub-results (results of the sub-questions).

There already exists a variety of QA benchmarks. For KB-QA, the Free917 (Cai and Yates, 2013) and WebQuestions (Berant et al., 2013) collections are the most popular. Both are vastly dominated by simple questions and do not exercise a system's capability to decompose and process complex questions. The QALD series of evaluation tasks (Usbeck et al., 2017) includes both simple and complex

questions. However, the number of questions per year is relatively small ($50-250$ questions). The ComplexQuestions collection of Bao et al. (2016) contains various types of complex questions: however, temporal questions present only a small fraction. For text-oriented QA, the TREC (Voorhees, 2010; Agichtein et al., 2015) and CLEF (Peñas et al., 2015) conference series offer a wealth of benchmark questions, but there is no design consideration on harnessing structured data at all.

In this work we propose a benchmark, called *TempQuestions*, consists of 1,271 temporal questions with gold-standard answers over Freebase. This collection is derived by judiciously selecting time-related questions from the Free917, WebQuestions and ComplexQuestions sets, with additional curation and tagging of temporal cues.

5.2 Related Work

Question answering. QA has a long tradition in IR and NLP, including benchmarking tasks in TREC, CLEF and SemEval. This has predominantly focused on retrieving answers from textual sources. The recent TREC CAR (complex answer retrieval) resource (Dietz and Gamari, 2017), explores multi-faceted passage answers, but information needs are still simple. In IBM Watson (Ferrucci, 2012), structured data played a role, but text was the main source for answers. Question decomposition was leveraged, for example, in (Ferrucci, 2012; Saquete et al., 2009; Yin et al., 2015) for QA over text. However, re-composition and reasoning over answers work very differently for textual sources (Saquete et al., 2009), and are not directly applicable for KB-QA. Compositional semantics of natural-language sentences has been addressed by Liang et al. (2011) from a general linguistic perspective. Although applicable to QA, existing systems support only specific cases of composite questions.

KB-QA is a more recent trend, starting with (Berant et al., 2013; Cai and Yates, 2013; Fader et al., 2014; Unger et al., 2012; Yahya et al., 2012). Most methods have focused on simple questions, whose SPARQL translations contain only a single variable (and few triple patterns for a single set of qualifying entities). For popular benchmarks like WebQuestions (Berant et al., 2013), the best performing systems use templates and rules or grammars (e.g., (Abujabal et al., 2017; Bast and Haussmann, 2015; Reddy et al., 2014)), leverage additional text sources (e.g., (Savenkov and Agichtein, 2016; Xu et al., 2016b)), or rely on extensive training data for end-to-end learning (e.g., (Li et al., 2017a; Xu et al., 2016b; Yih et al., 2015)). These techniques do not cope well with complex questions.

Bao et al. (2016) combined rules with deep learning to address different types of complex questions accompanied by the ComplexQuestions dataset.

Datasets for KB-QA. Multiple datasets have been proposed for question answering over knowledge bases, which differ in the underlying knowledge base (DBpedia or Freebase), size (a couple of hundreds to a few thousands), and question phenomena they evoke (simple, compositional, and/or questions with conditions, among others) (Abujabal et al., 2017; Bao et al., 2016; Berant et al., 2013; Bordes et al., 2015; Cai and Yates, 2013; Unger et al., 2015; Usbeck et al., 2017).

Datasets with complex questions are still ad hoc. QALD (Unger et al., 2015; Usbeck et al., 2017) is a series of evaluation campaigns on QA over linked data, and releases datasets every year to evaluate KB-QA systems. Questions in QALD cover many interesting phenomena such as aggregation, count, and additional conditions, (for example, *"Which German cities have more than 250000 inhabitants?"*). However, the main shortcoming is the very small size ($50 - 250$ questions). Recently, Abujabal et al. (2017) released 150 questions paired with their answers over Freebase. While all questions in this dataset contain more than one entity/relation, the underlying SPARQL query would still require joining over a single variable only. Questions were collected using a public crawl of WikiAnswers, a large, community-authored corpus of questions. The WebQuestions (Berant et al., 2013) and SimpleQuestions (Bordes et al., 2015) datasets contain a majority of simple factoid questions, e.g., *"what language does cuba speak?"*, with a few exceptions. While questions in WebQuestions (Berant et al., 2013) are only paired with answers, they are paired with SPARQL queries in Bordes et al. (2015). Bao et al. (2016) release a new dataset with complex questions paired with their answers over Freebase ($2,100$ question-answer pairs). The LC-QuAD dataset (Trivedi et al., 2017) contains $5,000$ questions and their corresponding SPARQL queries over DBpedia. Questions in LC-QuAD exhibit high syntactic and structural variation. These were generated using a set of handwritten templates that verbalize SPARQL queries, which are then corrected and paraphrased by humans.

5.3 Setup

There are diverse types of temporal aspects in questions. Questions can contain temporal expressions or signals to express temporal relations. Furthermore, questions may ask for temporal information, e.g., a date. We first explain what temporal expressions and temporal signals are. Then, we define temporal ques-

tions.

Temporal Expressions

The temporal markup language TimeML (Pustejovsky et al., 2005) is used for annotating temporal information in text documents. It is also the annotation standard adopted by most tools that perform temporal tagging automatically (Strötgen and Gertz, 2016).

TimeML contains `TIMEX3` tags for temporal expressions and `SIGNAL` tags for temporal signals. The `TIMEX3` tag is used to annotate temporal expressions of four types: date, time, duration, and set expressions. The semantics of all temporal expressions can be normalized to some value in a standard format, which allows the comparison between temporal expressions – a characteristic of temporal information, which can also be exploited for temporal QA. TimeML's most important attribute to capture the temporal information of temporal expressions is the `value` attribute. In the case of duration and set expressions, the `value` attribute captures the length of the interval, and the `value` attribute of date and time expressions contains information how to anchor the point in time on a timeline of the respective granularity.

According to TimeML's specifications, set expressions refer to the re-occurring nature of an event. Examples are '*once a week*' and '*daily*'. Duration expressions are used to specify the length of an interval. For instance, '*three weeks*' and '*several years*' are two duration expressions. Note that the temporal information might be concrete as in '*three weeks*' or vague as in '*several years*'. Date and time expressions both refer to points in time – though the points in time are of different granularities: all granularities smaller than '*day*' are considered as time expressions, for instance, expressions referring to parts of a day (e.g., '*Monday morning*' and '*yesterday night*') and expressions referring to a specific time (e.g., '*9 pm*', '*three o'clock*' and '*February 5, 2018 23:59:59 CET*'). In contrast, date expressions may refer to a particular day (e.g., '*last Thursday*' and '*23rd of November*') or to any point in time of a coarser granularity (e.g., '*the 21st century*', '*last year*' and '*September 2016*').

Note that these examples directly show that date and time expressions can be realized in different ways: fully-specified, relatively specified, underspecified, or implicitly specified (Strötgen and Gertz, 2016). Fully-specified expressions can be normalized without any further context information (e.g., '*September 2016*' as `2016-09`). In contrast, relative expressions require a reference time (e.g., '*last Thursday*') and underspecified expressions need a reference time and a relation to the reference time (e.g., '*(on) Thursday*'). In both cases, the reference time might be the time of the sentence or a date mentioned in the textual context.

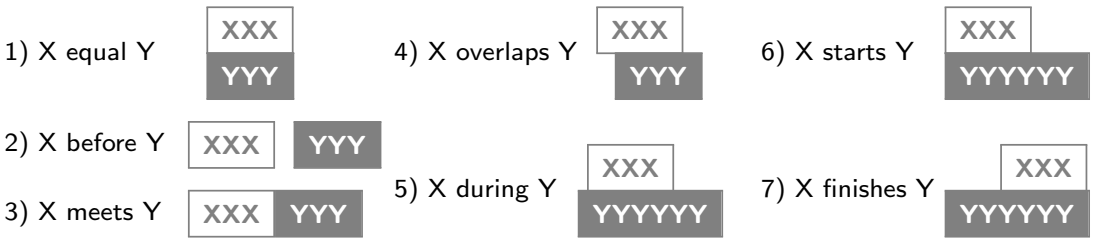


Figure 5.1: The 13 temporal relations (nos. 2 through 7 have inverses) between two intervals X and Y, as in Allen (1983).

If relative and underspecified date and time expressions occur in questions, it is thus important that the information about when the question was formulated is also available. Otherwise, questions such as *"Who was the US president two years ago?"* cannot be answered as it is impossible to determine to which year '*two years ago*' refers.

Finally, non-standard temporal knowledge is required for normalizing implicit expressions such as holidays (e.g., '*Columbus Day 2018*' – which is, in the US, the second Monday in October). In some works, the definition of implicit temporal expressions has been extended to further include all types of free-text temporal expressions, such as event names or other textual phrases with temporal scopes (Kuzey et al., 2016) (e.g., '*Obama's presidency*', which can be normalized to an interval with a particular start and end date).

Temporal Signals

TimeML defines *temporal signals* as textual elements that make explicit the *temporal relation* between two TimeML entities (events or temporal expressions), such as '*before*' or '*during*'. In natural language questions, signals occur, for instance, to explicitly specify a valid time interval for the searched information, as in: *"Which movies did Besson work on before his marriage to Jovovich?"*.

In general, any of the 13 temporal relations defined in Allen's interval algebra for temporal reasoning (Allen, 1983) can be the described relation, that is, the `equal` relation as well as the six relations `before`, `meets`, `overlaps`, `during`, `starts`, and `finishes` with respective inverses (see Figure 5.1 for visualizations of the relations). However, due to ambiguities, it is often not possible to select a unique temporal relation for a temporal question. For example, the question *"What did Besson work on before his marriage to Jovovich?"* could be interpreted as asking for either the movie he was working on directly before his marriage or all movies which he was working on any time before his marriage.

It is crucial to point out that questions are often formulated with even further

ambiguities. While the question *"Which movies did Besson work on before his marriage to Jovovich?"* as well as *"Which movie did Besson work on before his marriage to Jovovich?"* concisely describe the required number of answer movies (several due to plural and one due to singular, respectively), the latter requires the movie which Besson worked on directly before his marriage, i.e., the temporal constraint cannot be simply validated, but valid answers have to be sorted and the closest one has to be chosen. In addition, the slightly reformulated question *"What did Besson work on before his marriage to Jovovich?"* could be interpreted one way or the other (singular or plural) – a fact that also makes it sometimes difficult, even for humans, to determine the correct answer of a question.

Due to such ambiguities, in the context of temporal QA, temporal relations could be simplified as the following three types:

(i) `before` and `meet` are treated as the relation `BEFORE`

(ii) `before_inverse` and `meet_inverse` are treated as `AFTER`

(iii) all other relations are treated as `OVERLAP`

Typical *trigger words* suggesting the three temporal relations above, respectively, are the temporal signals:

(i) '*before*', '*prior to*'

(ii) '*after*', '*following*'

(iii) '*during*', '*while*', '*when*', '*until*', '*in*', '*at the same time*'

In addition to the trigger terms defined in TimeML, we add ordinals to the class of temporal signals, as they are often used in questions to specify particular instances of items which can be sorted chronologically. An example is '*last*' in *"What was Besson's last movie before his marriage to Jovovich?"*.

Based on our discussion of temporal expressions and temporal signals above, we can define a temporal question:

Definition 5.1: A *temporal question* is any question, which contains a temporal expression, a temporal signal, or whose answer is of temporal nature.

Expected input: wh∗ $w_1 \ldots w_n$ SIGNAL $w_{n+1} \ldots w_p$?
Case 1: Constraint has both an entity and a relation
Sub-question 1 pattern: wh∗ $w_1 \ldots w_n$? **Sub-question 2 pattern:** when $w_{n+1} \ldots w_p$? **Example:** *"where did neymar play before he joined barcelona?"* **Sub-question 1:** *"where did neymar play?"* **Sub-question 2:** *"when neymar joined barcelona?"*
Case 2: Constraint has a relation only
Sub-question 1 pattern: wh∗ $w_1 \ldots w_n$? **Sub-question 2 pattern:** when sq1-entity $w_{n+1} \ldots w_p$? **Example:** *"where did neymar live before playing for clubs?"* **Sub-question 1:** *"where did neymar live?"* **Sub-question 2:** *"when neymar playing for clubs?"*
Case 3: Constraint has an entity only
Sub-question 1 pattern: wh∗ $w_1 \ldots w_n$? **Sub-question 2 pattern:** when $w_{n+1} \ldots w_p$ $w_1 \ldots w_n$? **Example:** *"who was the brazil team captain before neymar?"* **Sub-question 1:** *"who was the brazil team captain?"* **Sub-question 2:** *"when neymar was the brazil team captain?"*
Case 4: Constraint is an event name
Sub-question 1 pattern: wh∗ $w_1 \ldots w_n$? **Sub-question 2 pattern:** when did $w_{n+1} \ldots w_p$ happen? **Example:** *"where did neymar play during south africa world cup?"* **Sub-question 1:** *"where did neymar play?"* **Sub-question 2:** *"when did south africa world cup happen?"*

Table 5.1: Decomposition and rewriting of questions. The *constraint* is the fragment after the SIGNAL word. *wh∗* is the question word, e.g., *who*, and w_i are tokens in the question.

5.4 The TEQUILA Framework

Given an input question, TEQUILA works in four stages: (i) detect if the question is temporal (in the sense of Section 5.3), (ii) decompose the question into simpler sub-questions with some form of rewriting, if needed, (iii) obtain candidate answers and dates for temporal constraints from a KB-QA system (and

possibly direct KB-lookups), and (iv) apply constraint-based reasoning on the candidates using the temporal signals and constraints to produce final answers. Our method builds on ideas from the literature on question decomposition for general QA (Abujabal et al., 2017; Bao et al., 2014; Saquete et al., 2009), but goes much further and provides a comprehensive solution for temporal QA.

5.4.1 Detecting Temporal Questions

A question is identified as temporal if it contains any of the following: (a) explicit or implicit temporal expressions (dates, times, events), (b) temporal signals (i.e., cue words for temporal relations), (c) ordinal words (e.g., *first*), (d) an indication that the answer type is temporal (e.g., the question starts with '*When*'). We use HeidelTime (Strötgen and Gertz, 2010) to tag TIMEX3 expressions in questions. Named events e.g., '*World War II*' are identified using a dictionary curated from Freebase. Specifically, if the type of an entity is '*time.event*', its surface forms are added to the event dictionary. SIGNAL words and ordinal words are detected using a small dictionary, and a list of temporal prepositions. To spot questions whose answers are temporal, we use a small set of patterns like *when*, *what date*, *in what year*, and *which century*.

5.4.2 Decomposing and Rewriting Questions

TEQUILA decomposes a composite temporal question into one or more *non-temporal sub-questions* (returning candidate answers), and one or more *temporal sub-questions* (returning temporal constraints). Results of non-temporal sub-questions are combined by intersecting their answers. The constraints are applied to time scopes associated with the results of non-temporal sub-question. For brevity, we focus on the case with one non-temporal sub-question, and zero/one temporal sub-question.

For questions with either an ordinal word, an explicitly stated temporal constraint or no temporal constraints at all, we generate zero temporal sub-questions. Concretely, for those questions with ordinal constraints e.g., *"who was the first coach of the bucaneers?"*, we simply drop the ordinal value ('*first*') and proceed to answering the non-temporal sub-question *"who was the coach of the bucaneers?"* (Section 5.4.3). During reasoning, we make use of the ordinal value by sorting candidate answers over temporal dimension. For questions with explicit temporal constraints e.g., *"what kind of government does iran have after 1979?"*, we generate a non-temporal sub-question e.g., *"what kind of government does iran have?"* by dropping the signal word and what follows it. We use the temporal

relation ('*after*') and the explicit temporal value ('*1979*') during reasoning to obtain the final answer set (Section 5.4.4). Questions with no temporal constraints are directly answered by the underlying KB-QA system. For example, *"when was the united nations founded?"*. Such questions require no further processing.

The more interesting questions are those with implicit temporal constraints, where converting such implicit constraints to explicit ones requires generating sub-questions seeking temporal information. For example, the question *"Which teams did Neymar play for before joining PSG?"* is decomposed into the non-temporal sub-question *"Which teams did Neymar play for?"*, which returns candidate answers, and the temporal sub-question *"When Neymar joining PSG?"*, which returns temporal constraints to filter the candidate answers from the non-temporal sub-question. To decompose a composite question, we use a set of lexico-syntactic rules designed from first principles to decompose and rewrite a question into its components. The intuition behind generating temporal sub-questions is to convert implicit temporal constraints into explicit ones that can be evaluated over the time scopes of the candidate answers. Basic intuitions driving these rules are as follows:

- The signal word usually separates the non-temporal sub-question and the implicit temporal constraint, acting as a pivot for decomposition;
- Each sub-question needs to have an entity and a relation (generally represented using verbs), so as to enable the underlying KB-QA systems to handle sub-questions;
- If the second sub-question lacks the entity or the relation, it is borrowed from the first sub-question;
- KB-QA systems are robust to ungrammatical constructs, thus precluding the need for linguistically correct sub-questions.

Table 5.1 shows our rules to decompose and rewrite questions with implicit temporal constraints, accompanied with examples. The rules cover four cases, which differ in how the implicit temporal constraint is represented:

1. Constraint constitutes a full sub-question: This means, the constrains has both an entity and a relation. In this case, the constraint is used as the second sub-question and nothing is borrowed from the first sub-question.
2. Constraint has an entity only: In this case, we borrow the relation from the first sub-question.

Relation	Signal word(s)	Constraint
BEFORE	'*before*', '*prior to*'	$end_{ans} \leq begin_{cons}$
AFTER	'*after*'	$begin_{ans} \geq end_{cons}$
OVERLAP	'*during*', '*while*', '*when*'	$begin_{ans} \leq end_{cons} \leq end_{ans}$
	'*since*', '*until*', '*in*'	$begin_{ans} \leq begin_{cons} \leq end_{ans}$
	'*at the same time as*'	$begin_{cons} \leq begin_{ans} \leq end_{ans} \leq end_{cons}$

Table 5.2: Temporal reasoning constraints.

3. Constraint has a relation only: Here, we borrow the entity from the first sub-question.

4. Constraint corresponds to a named event: In this case, we simply generate a sub-question asking for the date of the event.

5.4.3 Answering Sub-questions

The sub-questions are passed on to a KB-QA system, which translates them into SPARQL queries and executes them on the KB. This produces a result set for each sub-question. Results from the non-temporal sub-question(s) are entities of the same type (e.g., football teams). These are the candidate answers for the full user question. With multiple sub-questions, the candidate sets are intersected. The temporal sub-questions, on the other hand, return temporal results such as dates, which act as constraints to filter the non-temporal candidate set. The candidate answers need to be associated with time scopes, so that we can evaluate the temporal constraints.

Retrieving time scopes. To obtain time scopes for the candidate answers, we introduce additional KB lookups; the details depend on the specifics of the underlying knoweldge base. Freebase, for example, associates SPO triples with time scopes by means of compound value types (CVTs); other KBs may use n-tuples ($n > 3$) instead of triples to attach spatio-temporal attributes to facts (Section 2.1.2). For example, in Freebase, marriage relationship is a CVT with attributes including `marriage.spouse`, `marriage.startDate` and `marriage.endDate`. When the underlying KB-QA system return answers using the KB predicate `marriage.spouse`, the time scope is easily retrieved by looking up `marriage.startDate` and `marriage.endDate` in the KB.

Representation of time scopes is not always properly reflected in the knowledge base structure. For example, playing for a football club could be captured in a predicate like `team.players` without temporal information attached, and the job periods are represented as events in predicates like `footballPlayer.team.joinedOnDate` and `footballPlayer.team.leftOnDate`. In such cases, TEQUILA considers all kinds of temporal predicates associated with the candidate entity, and chooses one based on a *similarity measure* between the non-temporal predicate (e.g., `team.players`) and the potentially relevant temporal predicates (e.g., `footballPlayer.team.joinedOnDate`, `footballPlayer.award.date`, etc.). The similarity measure is implemented by selecting tokens in a predicate name (e.g., `footballPlayer`, `team`, etc.), computing *word2vec* embeddings for them (to contextualize the tokens), superimposing the per-token vectors, and comparing the cosine distance between predicate vectors (Wieting et al., 2015). The best-matching temporal predicate is chosen for use. When time periods are needed (e.g., for a temporal constraint using OVERLAP), a pair of begin/end predicates is selected (e.g., `footballPlayer.team.joinedOnDate` and `footballPlayer.team.leftOnDate`).

5.4.4 Reasoning on Temporal Intervals

By now, time scopes of candidate answers have been retrieved from the KB, and implicit temporal constraints have been disambiguated and normalized through generating temporal sub-questions. In case the question contains ordinal word or explicit temporal expression, no normalization is needed. We cast time scopes and temporal constraints into intervals with start point $begin_{ans}$, $begin_{cons}$ and end point end_{ans}, end_{cons}, respectively. The test itself depends on the temporal operator derived from the input question (e.g., BEFORE, OVERLAP, etc.) (Table 5.2). For questions with ordinal constraints (e.g., *last*), we sort the time scopes to select the appropriate answer.

5.5 Experimental Evaluation

We evaluate TEQUILA on a newly introduced set of temporal questions, called TempQuestions. Our experiments serve two goals: (1) to show that TEQUILA enables existing standard KB-QA systems to handle temporal complex questions, and (2) that TEQUILA outperforms KB-QA systems that address complex questions.

Question Tag	Free917	WebQuestions	ComplexQuestions	Total
Explicit temporal	41	344	222	607
Implicit temporal	3	81	125	209
Temporal answer	88	254	51	393
Ordinal constraint	18	111	26	155
Total	150	790	424	1,364

Table 5.3: Distribution of question types by source. The total is greater than 1,271 as some questions have multiple tags.

5.5.1 Setup

Benchmark. Current KB-QA benchmarks datasets (Bao et al., 2016; Berant et al., 2013; Cai and Yates, 2013) individually contain small fractions of temporal questions. Thus, methods that ignore temporal conditions in questions manage to get good overall performance. This motivated us to compile a new benchmark, coined *TempQuestions*. TempQuestions is a high-quality benchmark with a total of 1,271 temporal questions paired with their answers over Freebase. To this end, we ran our temporal question detection (Section 5.4.1) on three existing datasets: Free917 (Cai and Yates, 2013) (917 question-query pairs), WebQuestions (Berant et al., 2013) (5,810 question-answer pairs), and ComplexQuestions (Bao et al., 2016) (2,100 question-answer pairs). For Free917, we evaluated the SPRAQL queries paired with questions over Freebase to retrieve answers. While the first two have been popular in the KB-QA community over the last years, ComplexQuestions is recent and contains questions that are syntactically more complex than those in the other two datasets. The output of the automated detection step was manually curated, which resulted in removing 245 non-temporal questions (e.g., *"what countries speak german as a first language?"*). a misinterpretation of the ordinal tag), along with redundant and noisy answers.

Questions in our benchmark are between 4 and 15 words long, and the average question length is 8.28 words. Sample questions are depicted in Table 5.4, segmented by the following three dimensions: temporal category, numbers of entities and relations, and question source.

We provide a breakdown into the four classes of temporal questions (explicit, implicit, temporal answer and ordinals), along with the input source, in Table 5.3. TempQuestions has a good number of questions with implicit temporal expressions (209) and ordinals (155) – both these classes require *additional rea-*

Property	Question
Segmentation by temporal category	
Explicit	*"who won the state of texas in 2008?"* *"what kind of government does iran have after 1979?"*
Implicit	*"who was the president after jfk died?"* *"what team did michael jordan play for after the bulls?"*
Temporal answer	*"what years did the knicks win the championship?"* *"when was the united nations founded?"*
Ordinal constraint	*"who was the first coach of the bucaneers?"* *"who was andy williams second wife?"*
Segmentation by question concepts	
Multi-entity	*"what did france lose to the british in the treaty of paris in 1763?"* *"when was the last time the oakland raiders won the super bowl?"*
Multi-relation	*"who won best supporting actor when alfred junge won best art direction?"* *"what book was written by george orwell and published in 1945?"*
Segmentation by question source	
Free917 (Cai and Yates, 2013)	*"when was the airspeed oxford first flown?"* *"in 1981 what award did danny devito win?"*
WebQuestions (Berant et al., 2013)	*"what was the currency in france before euro?"* *"who is julia roberts married to 2012?"*
ComplexQuestions (Bao et al., 2016)	*"who was us president when vietnam war started?"* *"who did michael jordan play for after the bulls?"*

Table 5.4: Representative examples from TempQuestions.

soning and ranking on part of the QA-system, and thus add a level of difficulty; (b) the total $1,364$ is higher than $1,271$, showing that there are several questions that belong to more than one category, and are thus quite challenging for current QA systems (like *"who was elected the first governor of virginia in 1776?"*, with both explicit and ordinal tags).

Baselines. We use three state-of-the-art KB-QA systems as baselines: AQQU (Bast and Haussmann, 2015), QUINT (Abujabal et al., 2017), and the system of Bao et al. (2016). The first two are template-based methods geared for simple questions, while the system of Bao et al. (2016) handle complex questions, including temporal ones. We use TEQUILA as a plug-in for the first two systems, and directly evaluate against the system of Bao et al. (2016) on 341 temporal questions from the ComplexQuestions test set (Bao et al., 2016).

Metrics. For evaluating baselines, the full question was fed directly to the underlying system. We report precision, recall, and F1-scores of the retrieved answer sets with respect to the gold answer sets, and average them over all test questions.

5.5.2 Results

Results on TempQuestions and the 341 temporal questions in ComplexQuestions are shown in Table 5.5. AQQU + TEQUILA and QUINT + TEQUILA refer to the TEQUILA-enabled versions of the respective baseline systems.

TEQUILA enables standard QA over KBs systems to answer composite questions with temporal conditions. Overall and category-wise F1-scores show that TEQUILA-enabled systems to significantly outperform the baselines. For example, while AQQU achieved an F1 score of 26.9 on TempQuestions, the TEQUILA-enabled version of AQQU achieved 36.7 F1 value (9.8 F1 points increase). Note that these systems neither have capabilities for handling compositional syntax nor specific support for temporal questions. Our decomposition and rewrite methods are crucial for compositionality, and constraint-based reasoning on answers is decisive for the temporal dimension. The improvement in F1-scores stems from a systematic boost in precision, across most categories. Despite ignoring temporal constraints, baseline systems were able to deliver answers to the full question, which explains their reasonable performance on TempQuestions. With TEQUILA framework, temporal constraints were taken care of by filtering out candidate answers that do not respect the constraints, and hence, boosting precision.

TEQUILA-enabled systems outperform the system of Bao et al. (2016) on the temporal slice of ComplexQuestions. The system of Bao et al. (2016) represents the state-of-the-art in KB-QA, with a generic mechanism for handling constraints in questions. This shows that a tailored method for temporal information needs is worthwhile. Some questions that TEQUILA enabled both QUINT and AQQU to answer are: *"who is the first husband of julia roberts?"*, and *"who was governor of oregon when shanghai noon was released?"*.

5.5.3 Discussion

Analyzing cases when TEQUILA fails yields various insights towards future work: (i) Incorrect decomposition and rewriting: For example, in *"where did the pilgrims come from before landing in america?"*, '*landing*' is incorrectly labeled as a noun, triggering case 3 instead of case 1 in Table 5.1; (ii) Incorrect time scope predicate: These cases happen when the correct temporal predicate is not found due to limitations of the similarity function. (iii) Reasoning errors: Wrongly identifying the temporal constraint or the time scope to use during reasoning, is another point of failure for TEQUILA.

TEQUILA was able to improve the performance of QUINT across categories

TempQuestions (1,271 questions)	**Aggregate results**			**Explicit constraint**			**Implicit constraint**			**Temporal answer**			**Ordinal constraint**		
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
AQQU (Bast and Haussmann, 2015)	24.6	48.0	27.2	27.6	60.7	31.1	12.9	34.9	14.5	26.1	33.5	27.4	28.4	**57.4**	32.7
AQQU+TEQUILA	**36.0***	42.3	**36.7***	**43.8***	53.8	**44.6***	**29.1***	34.7	**29.3***	27.3*	29.6	**27.7***	**38.0***	41.3	**38.6***
QUINT (Abujabal et al., 2017)	27.3	**52.8**	30.0	29.3	**60.9**	32.6	25.6	**54.4**	27.0	25.2	**38.2**	27.3	21.3	54.9	26.1
QUINT+TEQUILA	33.1*	44.6	34.0*	41.8*	51.3	42.2*	13.8	43.7	15.7	**28.6***	34.5	**29.4***	37.0*	42.2	37.7*
ComplexQuestions (341 questions)	**Aggregate results**			**Explicit constraint**			**Implicit constraint**			**Temporal answer**			**Ordinal constraint**		
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
Bao et al. (2016)	34.6	48.4	35.9	41.1	53.2	41.9	26.4	36.5	**27.0**	18.6	**40.2**	22.3	31.1	**60.8**	36.1
AQQU (Bast and Haussmann, 2015)	21.5	50.0	23.3	25.0	**60.1**	28.4	11.2	31.2	11.4	19.6	35.7	19.2	22.2	54.9	25.3
AQQU+TEQUILA	**36.2***	45.9	**37.5***	**41.2***	54.7	**43.5***	**27.5***	32.6	**27.0***	29.5*	32.1	29.9*	40.2*	45.1	40.8*
QUINT (Abujabal et al., 2017)	22.0	**50.3**	24.5	24.7	54.7	27.5	18.8	**47.9**	19.0	16.6	37.5	20.7	20.9	51.3	26.0
QUINT+TEQUILA	29.6*	44.9	31.1*	34.6*	47.3	36.3*	12.3	42.1	13.9	**33.4***	37.5	**33.9***	**44.9***	51.6*	**45.8***

Table 5.5: Detailed performance of TEQUILA-enabled systems on TempQuestions and ComplexQuestions. The highest value in a column for each dataset is in **bold**. An asterisk (*) indicates statistical significance of TEQUILA-enabled systems over their standalone counterparts, under the 2-tailed paired t-test at $p < 0.05$ level.

except for questions with implicit constraints. While QUINT achieved an F1 score of 25.4 on the slice of TempQuestions with implicit constraints, TEQUILA-enabled version of QUINT achieved an F1 value of 17.4. We attribute this deterioration in performance to the fact that QUINT relies heavily on well-formed questions, however, our decomposition and rewriting rules might generate linguistically ill-formed temporal sub-questions, and hence, were not correctly answered by QUINT.

6 ComQA: A Community-sourced Dataset for Complex Factoid Question Answering with Paraphrase Clusters

6.1 Introduction

Factoid QA is the task of answering questions whose answer is one or a small number of entities (Voorhees and Tice, 2000). Traditionally, approaches have mostly tapped into textual sources using passage retrieval and answer ranking techniques (Ferrucci, 2012; Harabagiu et al., 2003; Ravichandran and Hovy, 2002; Saquete et al., 2009). In the last few years, the paradigm of translating questions into formal queries over structured knowledge bases (KB-QA) has become prevalent (Bast and Haussmann, 2015; Berant et al., 2013; Unger et al., 2012; Yahya et al., 2013). Recently, methods have combined both textual and structured resources to boost answering performance (Savenkov and Agichtein, 2016; Xu et al., 2016b). To advance research on QA in a manner consistent with the needs of end users, it is important to have access to QA benchmarks that reflect *real* user information needs by covering various *question phenomena*, capture the wide *lexical and syntactic variety* in expressing these information needs, and are large enough to facilitate the use of data-hungry machine learning methods. In this paper, we present ComQA, a large dataset of 11,214 real user questions collected from the WikiAnswers community QA website. As Figure 6.1 shows, the dataset contains various question phenomena. ComQA questions are grouped into 4,834 paraphrase clusters through a large-scale crowdsourcing effort, which capture lexical and syntactic variety. Crowdsourcing is also used to pair paraphrase clusters with answers to serve as a supervision signal for training and as a basis for evaluation.

Table 6.1 contrasts ComQA with other publicly available QA datasets. The foremost issue ComQA tackles is ensuring research is driven by real information

Figure 6.1: Paraphrase clusters from ComQA, covering a range of question aspects, with lexical and syntactic diversity.

needs. Most benchmarks resort to synthetically generated questions (Bordes et al., 2015; Cai and Yates, 2013; Su et al., 2016; Talmor and Berant, 2018; Trivedi et al., 2017). Other benchmarks utilize search engine logs to collect their questions Berant et al. (2013), which creates a bias towards simpler questions that search engines can already answer reasonably well. In contrast, ComQA questions come from WikiAnswers, a community QA website where users pose questions to be answered by other users. This is often a reflection of the fact that such questions are beyond the capabilities of commercial search engines and QA systems. Questions in our dataset exhibit a wide range of interesting aspects such as the need for temporal reasoning (Figure 6.1, cluster 1), comparison (e.g., comparatives, superlatives, ordinals) (Figure 6.1, cluster 2), compositionality (multiple, possibly nested, subquestions with multiple entities) (Figure 6.1, cluster 3), and unanswerable questions e.g., Figure 6.1, cluster 4.

ComQA is the result of a carefully designed large-scale crowdsourcing effort to group questions into paraphrase clusters and pair them with answers. Past work has demonstrated the benefits of paraphrasing for QA (Abujabal et al., 2018; Berant and Liang, 2014; Dong et al., 2017; Fader et al., 2013). Motivated by this, we judiciously use crowdsourcing to obtain clean paraphrase clusters

Dataset	Large scale (> 5K)	Real Information Needs	Complex Questions	Question Paraphrases
ComQA (This work)	✓	✓	✓	✓
Free917 (Cai and Yates, 2013)	✗	✗	✗	✗
WebQuestions (Berant et al., 2013)	✓	✓	✗	✗
SimpleQuestions (Bordes et al., 2015)	✓	✗	✗	✗
QALD (Usbeck et al., 2017)	✗	✗	✓	✗
LC-QuAD (Trivedi et al., 2017)	✓	✗	✓	✗
ComplexQuestions (Bao et al., 2016)	✗	✓	✓	✗
GraphQuestions (Su et al., 2016)	✓	✗	✓	✓
ComplexWebQuestions (Talmor and Berant, 2018)	✓	✗	✓	✗
TREC (Voorhees and Tice, 2000)	✗	✓	✓	✗

Table 6.1: Comparison of ComQA with existing QA datasets over various dimensions.

from WikiAnswer's noisy ones, resulting in ones like those shown in Figure 6.1, with both lexical and syntactic variations. The only other existing dataset to provide such clusters is that of Su et al. (2016), but that is based on synthetic information needs as detailed above.

For answering, recent research has shown that combining various resources for answering significantly improves performance (Savenkov and Agichtein, 2016; Sun et al., 2018; Xu et al., 2016b). Therefore, unlike earlier work, we do not pair ComQA with a specific knowledge base (KB) or text corpus for answering. We call on the research community to innovate in combining different answering sources to tackle ComQA and advance research in QA. We also use crowdsourcing to pair paraphrase clusters with answers. ComQA answers are primarily Wikipedia entity URIs. This has two motivations: (i) it builds on the example of search engines that use Wikipedia as a primary way of answering entity-centric queries (e.g., through knowledge cards), and (ii) most modern KBs ground their entities in Wikipedia. Wherever the answers are temporal or measurable quantities, TIMEX3[1] and the International System of Units (SI)[2] are used for normalization. Providing canonical answers allows for better comparison of different systems.

We present an extensive analysis of ComQA, where we introduce the various question phenomena in the dataset. Finally, we analyze the results of running five state-of-the-art QA systems on ComQA. The main result is that ComQA exposes major shortcomings in these systems, mainly related to their inability to handle compositionality, time, and comparison. Our detailed error analysis

[1] `http://www.timeml.org`
[2] `https://en.wikipedia.org/wiki/SI`

provides inspiration for avenues of future work to ensure that QA systems meet the expectations of real users. To summarize, in this paper we make the following contributions:

- We present a large dataset of 11,214 real user questions collected from a community QA website. The questions exhibit a wide range of phenomena that are important for end users and challenging for existing QA systems. Using crowdsourcing, questions in the dataset are grouped into 4,834 paraphrase clusters that are annotated with their Wikipedia-grounded answers. The dataset is available at `http://qa.mpi-inf.mpg.de/comqa`.
- We present an extensive analysis of the dataset, and quantify the various phenomena found within. Moreover, we present the results of various state-of-the art QA systems on ComQA, and a detailed error analysis.

6.2 Related Work

There are two main variants of the factoid QA task, with the distinction tied to the underlying resources used for answering and the nature of these answers. Traditionally, the problem of QA has been explored over large textual corpora (Cui et al., 2005; Dietz and Gamari, 2017; Ferrucci, 2012; Ravichandran and Hovy, 2002; Saquete et al., 2009; Voorhees and Tice, 2000) with answers being textual phrases. More recently the problem has been explored over large structured data sources such as knowledge bases (Berant et al., 2013; Unger et al., 2012; Yahya et al., 2013), with answers being semantically grounded entities. Very recent work demonstrated that the two variants are complementary, and a combination of the two results in the best answering performance (Savenkov and Agichtein, 2016; Sun et al., 2018; Xu et al., 2016b).

QA over textual corpora. QA has a long tradition in IR and NLP communities, including benchmarking tasks in TREC (Dang et al., 2006; Dietz and Gamari, 2017; Voorhees and Tice, 2000), CLEF Magnini et al. (2004); Herrera et al. (2004) and SemEval. This has predominantly focused on retrieving answers from textual sources (Ferrucci, 2012; Harabagiu et al., 2006; Lin, 2002; Prager et al., 2004; Ravichandran and Hovy, 2002; Saquete et al., 2004, 2009; Yin et al., 2015). In IBM Watson system (Ferrucci, 2012), structured data played a role, but text was the main source for answers combined with learned models for question types. TREC QA evaluation series provide hundreds of questions to be answered over a collection of documents, which have become one of the most widely adopted benchmarks for answer sentence selection (Wang and Nyberg,

2015; Yang et al., 2016a). Wang et al. (2007) constructed a benchmark using the data from TREC QA 8-13, where questions in TREC 8-12 are used for training and questions in TREC 13 are used for development and testing. ComQA is orders of magnitude larger than TREC QA data, accompanied with paraphrase clusters.

Reading comprehension is a recently introduced task, where the goal is to answer a question from a *given* textual paragraph (Joshi et al., 2017; Kociský et al., 2018; Lai et al., 2017; Rajpurkar et al., 2016; Trischler et al., 2017; Yang et al., 2015; Welbl et al., 2018). This setting requires understanding of natural language and is intended to improve answering from a given text (mimicking school children), rather than the general ability to efficiently obtain answers from large amounts of data. ComQA reflects the general QA setting, where finding answers requires searching through a collection of documents.

QA over knowledge bases. Recent efforts have focused on natural language questions as an interface for knowledge bases, where questions are translated to structured queries via semantic parsing (Bao et al., 2016; Bast and Haussmann, 2015; Berant et al., 2013; Fader et al., 2013; Reddy et al., 2014; Mohammed et al., 2018; Savenkov and Agichtein, 2016; Talmor and Berant, 2018; Xu et al., 2016b; Yang et al., 2014; Yao and Durme, 2014; Yih et al., 2015). Over the past five years, many datasets were introduced, however, they are either small in size (QALD, Free917, and ComplexQuestions), composed of synthetically generated questions (SimpleQuestions, GraphQuestions, LC-QuAD and ComplexWebQuestions), or structurally simple (SimpleQuestions and WebQuestions). ComQA tackles all these shortcomings. The ability to return semantic entities as answers allows users to further explore these entities in various resources such as their Wikipedia pages, Freebase entries, etc. It also allows QA systems to tap into various interlinked resources for improvement (e.g., to obtain better lexicons, or train better NER systems). Because of this, ComQA proivdes semantically grounded reference answers where possible. ComQA answers are primarily Wikipedia entities (without committing to Wikipedia as an answering resource). For numerics and dates, ComQA adopts the SI and TIMEX3 standards, respectively.

6.3 Setup

In this work, a factoid question is a question whose answer is one or a small number of entities (Voorhees and Tice, 2000). For example, *"Who were the secretaries of state under Barack Obama?"* and *"When was Germany's first*

post-war chancellor born?".

6.3.1 Questions in ComQA.

A question in our dataset can exhibit *one or more* of the following phenomena:

- **Simple:** These are questions that ask about a property of a named entity. e.g., *"Where was Einstein born?"*
- **Compositional:** A question is compositional if obtaining its answer requires answering more primitive questions and combining these. These can be *intersection* or *nested* questions. Intersection questions are ones where two or more subquestions can be answered independently, and their answers intersected (e.g., *"Which films featuring Tom Hanks did Spielberg direct?"*). Nested questions are those where the answer of one subquestion is necessary to answer another e.g., *"Who were the parents of the 13th president of the US?"*
- **Temporal:** These are questions that require temporal reasoning for deriving the answer, be it explicit (e.g., '*in 1998*'), implicit (e.g., '*during the WWI*'), relative (e.g., '*current*'), or latent (e.g. '*Who is the US president?*'). Temporal questions also include those whose answer is an explicit temporal expression.
- **Comparison:** We consider three types of comparison questions, namely *comparatives* (e.g., *"Which rivers in Europe are longer than the Rhine?"*), *superlatives* (e.g., *"What is the population of the largest city in Egypt?"*) and *ordinals* (e.g., *"What was the name of Elvis's first movie?"*).
- **Telegraphic** (Joshi et al., 2014): These are short questions formulated in a very informal manner similar to keyword queries e.g., *"First president India?"*. Systems that rely on linguistic analysis of questions often fail on such questions.
- **Answer tuple**: Where an answer is a tuple of connected entities as opposed to a single entity, e.g., *"When and where did Geroge H. Bush go to college, and what did he study?"*

6.3.2 Answers in ComQA.

Recent work has showed that the choice of answering resource, or the combination of resources significantly improves answering performance (Savenkov and Agichtein, 2016; Sun et al., 2018; Xu et al., 2016b). Inspired by this, ComQA is

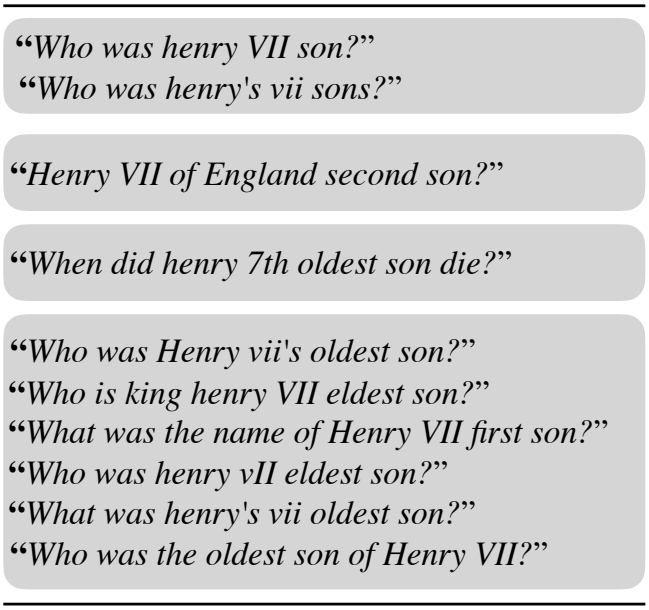


Figure 6.2: All ten questions belong to the same original WikiAnswers cluster. AMT Turkers split the original cluster into four new ones.

not tied to a specific resource for answering. To this end, answers in ComQA are Wikipedia URIs, wherever this is possible. This enables QA systems to combine different answering resources which are linked to Wikipedia (e.g., DBpedia, Freebase, Yago, Wikidata, Wikipedia, ClueWeb09-FACC1, etc). This also enables seamless comparison across QA systems whose individual answering resources are different, but are linked to Wikipedia. Literal value answers follow the TIMEX3 and SI standards. An answer in ComQA can be:

- **Entity:** ComQA entities are grounded in Wikipedia. However, Wikipedia is inevitably incomplete, so answers that cannot be grounded in Wikipedia are represented as plain text. For example the answer for *"What is the name of Kristen Stewart adopted brother?"* is {Taylor Stewart, Dana Stewart}.
- **Literal value:** Temporal answers follow the TIMEX3 standard. For measurable quantities, we follow the International System of Units.
- **Empty:** Some questions are based on false premises, and hence, are unanswerable e.g., *"Who was the first human being on Mars?"* (no human has been on Mars, yet). The correct answer to such questions is the empty set. Such questions allow systems to cope with these cases. Recent work has started looking at this problem (Rajpurkar et al., 2018).

6.4 Dataset Collection

Our goal is to collect a dataset of factoid questions that represent real information needs and cover a wide range of question phenomena. Moreover, we want to have different paraphrases for each question. To this end, we tap into the potential of CQA platforms. Questions posed there represent real information needs and often cannot be answered by commercial search engines or QA technology, making them more interesting for driving future research compared to those collected from an engine's query log. Moreover, users of CQA platforms provide (noisy) annotations around questions. Marking questions with the same information needs as duplicates, by the users of CQA, is harnessed in constructing the ComQA dataset. Concretely, we started with the WikiAnswers crawl by Fader et al. (2014). We obtained ComQA from this crawl primarly through a large scale crowdsourcing effort to ensure it is of high quality. We describe this effort in what follows.

The crawl contains 763M questions. While our focus is on factoid questions, that crawl contains other types of questions like *Why?* or comparison questions. Moreover, the questions are grouped into 30M paraphrase clusters based on feedback from the users of WikiAnswers. This clustering has low accuracy (Fader et al., 2014), and therefore needs to be cleaned for a high-accuracy dataset.

6.4.1 Preprocessing of WikiAnswers

To remove non-factoid questions we applied the following two filters: (i) removing questions starting with '*why*' and (ii) removing those containing words like *(dis)similarities, differences, (dis)advantages, benefits* and their synonyms. Questions matching these filters require a narrative as an answer, and are therefore out of scope.We also removed questions with less than three or more than twenty words, as we found these to typically be noisy or non-factoid questions. This left us with about 21M questions belonging to 6.1M clusters.

To further focus on factoid questions, we automatically classified the remaining questions into one or more of the following four classes: (1) temporal, (2) comparison, (3) single entity and (4) multi-entity questions. We used the SUTime to identify temporal questions (Chang and Manning, 2012) and the Stanford named entity recognizer to detect named entities (Finkel et al., 2005). We used part-of-speech patterns to identify comparatives, superlatives and ordinals. Clusters which did not have questions belonging to any of the above classes were discarded from further consideration. Although these clusters contain false negatives e.g., *"What official position did Mendeleev hold until his death?"* due to errors by

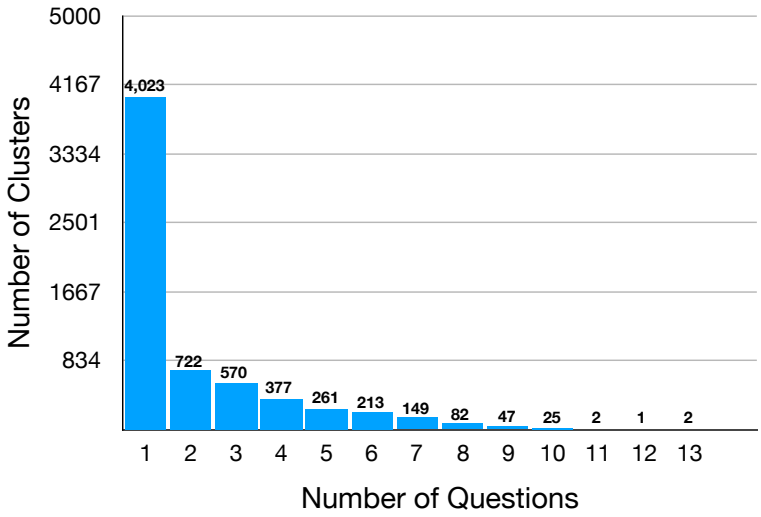


Figure 6.3: Distribution of the number of questions per cluster.

the tagging tools, most discarded questions are out-of-scope e.g., *"How does the government help fight poverty?"*.

Manual inspection. We next applied the first stage of human curation of the dataset. Each WikiAnswers cluster was assigned a class label based on the majority label of the questions within. We then randomly sampled 15K clusters from each of the four classes (60K clusters in total with 482K questions). We then sampled a representative question from each cluster at random (60K questions). We relied on the assumption that questions within the same cluster are semantically equivalent. These 60K questions were manually *examined by the authors* and those with unclear or non-factoid intent were removed along with the cluster that contains them. As a result, we ended up with 2.1K clusters with 13.7K questions.

6.4.2 Curating Paraphrase Clusters

We manually inspected a random subset of the 2.1K clusters, and found that questions of the same cluster are semantically *related but not equivalent* as has been observed by previous work (Fader et al., 2014). Dong et al. (2017) reported that 45% of question pairs were related rather than genuine paraphrases. For example, Figure 6.2 shows an original WikiAnswers cluster. The accuracy of paraphrase clustering can have a large impact on QA performance (Abujabal et al., 2018; Berant and Liang, 2014). We therefore utilized crowdsourcing to clean the Wikianswers paraphrase clusters. We used Amazon Mechanical Turk (AMT) to

identify semantically equivalent questions within a WikiAnswers cluster, thereby obtaining cleaner clusters for ComQA. Once we had the clean clusters, we set up a second AMT task to collect answers for each ComQA cluster of questions.

Task design. In designing the AMT task to clean up clusters of question paraphrases from WikiAnswer's question clusters, we had to ensure that the task was simple enough to obtain high quality results. Therefore, rather than giving workers a WikiAnswers cluster and asking them to partition it into clusters of paraphrases, we showed them pairs of questions from a cluster and asked them whether they are paraphrases. To improve the efficiency of this annotation effort, we utilized the transitivity of the paraphrase relationship. Given a WikiAnswers cluster $Q = \{q_1, ..., q_n\}$, we proceed in rounds to form ComQA paraphrase clusters. In the first round, we collect annotations for each pair (q_i, q_{i+1}). The majority annotation among five annotators is taken. An initial clustering is formed accordingly. In subsequent iterations, a question is chosen from a cluster and paired with questions from other clusters for annotation to determine whether these two clusters can be combined. This continues until no new clusters can be formed.

Task statistics. We obtained annotations for 18,890 question pairs. Each pair was shown to five different workers, with 65.7% of the pairs receiving unanimous agreement, 21.4% receiving four agreements and 12.9% receiving three agreements. By design, with five judges and binary annotations, no pair can have less three agreements. This resulted in questions being placed in paraphrase clusters, and no questions were discarded at this stage. At the end of this step, the original 2.1K WikiAnswers clusters became 6.4K ComQA clusters with a total of 13.7K questions. Figure 6.3 shows the distribution of questions in clusters.

To test whether relying on the transitivity of the paraphrase relationship is suitable to reduce the annotation effort, we asked annotators to annotate 1,100 random pairs (q_1, q_3), where we had already received positive annotations for the pairs (q_1, q_2) and (q_2, q_3) being paraphrases of each other. In 93.5% of the cases there was agreement. Additionally, as experts on the task, the authors manually assessed 600 pairs of questions, which serve as honeypots. There was 96.6% agreement with our annotations. An example result of this task is shown in Figure 6.2, where Turkers split the original WikiAnswers cluster into the four clusters shown.

6.4.3 Answering Questions

With clean paraphrase clusters, we were now in a position to obtain an answer annotations for each of the 6.4K clusters.

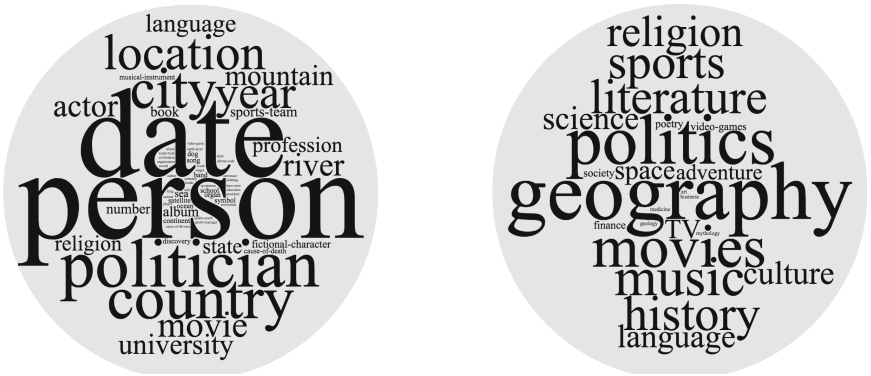
Task design. To collect answers, we designed another AMT task, where workers were shown a question randomly drawn from a cluster and asked to use the Web to find answers and to provide the URL of Wikipedia entities that are suitable answers. Due to the inevitable incompleteness of Wikipedia, workers were asked to provide the surface form of an answer entity in case it does not have a Wikipedia page. If the answer is a full date, workers were asked to follow dd-mmm-yyyy format. For other temporal answers e.g., centuries, workers were asked to provide the value followed by a temporal unit e.g., '*12th century*'. For measurable quantities, workers were asked to provide units. TIMEX3 and the international system of units are used for standardizing temporal answers and measurable quantities e.g., '*12th century*' to `11XX`. If no answer is found, workers were asked to type in '*no answer*' as an answer.

Task statistics. Each question was shown to three different workers. An answer is deemed correct if it is common between at least two workers. 1.3K representative questions received a single unanimous answer, 1.7K received at least one unanimous answer, while 4.8K out of 6.4K received at least one common answer between two annotators or more. This resulted in 1.6K clusters (containing 2.4K questions) with no agreed-upon answers, which were dropped. We manually inspected some questions with no agreed-upon answers. Some questions were subjective, for example, *"Who was the first democratically elected president of Mexico?"*. Other questions received related answers e.g., *"Who do the people in Iraq worship?"* with `Allah`, `Islam` and `Mohamed` as answers from the three annotators. Other questions were underspecified e.g., *"Who was elected the vice president in 1796?"*, which misses the entity, making agreement harder. At the end of the task, we had 4,834 clusters with 11,214 question-answer pairs, which form ComQA.

6.5 Dataset Analysis

In this section, we present a manual analysis of 300 questions sampled at random from the ComQA dataset. This analysis helps understand the different aspects of our dataset. A summary of the analysis is presented in Table 6.2.

Question categories. We categorized each question into either a simple or complex question. A question is deemed complex if it contains temporal and/or



(a) Answer types (b) Question topics

Figure 6.4: Answer types and question topics on 300 annotated examples as word clouds. The bigger the font, the more frequent the concept.

comparison conditions or it is of a compositional nature (see Section 6.3). 56.33% of the questions were complex, with 32% compositional, 23.67% temporal, and 29.33% contain comparison conditions. Note that a question might contain a combination of conditions e.g., *"What country has the **highest** population in the year **2008**?"* (comparison and temporal constraint).

We also identified questions of telegraphic nature e.g., *"Julia Alvarez's parents?"*, with 8% of our questions are telegraphic. Such questions pose a challenge for systems that rely on linguistic analysis of questions as shown in Joshi et al. (2014).

We counted the number of named entities in questions, with 23.67% contain two or more entities, indicating the compositional nature of questions. 2.67% have no entities e.g., *"What public company has the most employees in the world?"*. Such questions are hard since current methods assume the existence of a pivotal entity.

Finally, 3.67% of questions are unanswerable, e.g., *"Who was the first human being on Mars?"*. Such questions incentivise QA systems to return non-empty answer sets only when suitable. We also compared ComQA with other current datasets over different question categories (Table 6.3).

Answer types. We annotated each question with the most fine-grained answer type given the context. Answers in ComQA belong to a diverse set of types that range from coarse-grained e.g., `person` to fine-grained e.g., `sports manager`. Types include literals e.g., `number` and `date`. Figure 6.4 (a) shows the set of answer types on 300 annotated examples as word cloud.

Question topics. We annotated questions with topics they belong to e.g.,

geography or movies. The topics, shown in Figure 6.4 (b), demonstrate that questions belong to a diverse set of topics.

Question length. Questions in ComQA are fairly long, with a mean length of 7.73 words, indicating the compositional nature of questions.

6.6 Experimental Evaluation

We experimented with various state-of-the-art QA systems, which achieved humble performance on ComQA, highlighting the need to new methods to handle the question phenomena within ComQA.

6.6.1 Experimental Setup

Splits. In total, ComQA contains 11,214 question-answer pairs. We generated random split of 70% (7,850), 10% (1,121) and 20% (2,243), which serve as train, development and test sets.

Evaluation Metrics. We follow standard evaluation metrics adopted by the community: we compute average precision, recall, and F1 scores across all test questions. However, because ComQA includes unanswerable questions whose correct answer is the empty set, we define precision and recall to be 1 for a system that returns an empty set in response to an unanswerable question, and 0 otherwise (Rajpurkar et al., 2018).

Baselines We evaluated two types of QA systems that differ in the underlying answering resource: either KBs or textual extractions. We ran the following QA systems using their publicly available code: 1) Abujabal et al. (2017), which generates question-query templates using question-answer pairs. 2) Berant and Liang (2015), which relies on agenda-based parsing to train QA systems. 3) Bast and Haussmann (2015), which instantiates query templates at answering time. Candidate queries are ranked using a learning-to-rank model. 4) Berant et al. (2013), which uses rules to build logical forms from questions. 5)Fader et al. (2013), which maps questions to queries over open vocabulary facts extracted from a large text corpus of Web documents. Note that our intention is not to assess the quality of current systems, but to show the challenging nature of ComQA.

All systems were run over the data sources for which they were designed. The first four baselines are over Freebase, therefore, we mapped ComQA answers (Wikipedia entities) to the corresponding Freebase names using the information stored with entities in Freebase. We observe that the Wikipedia answer entities

have no counterpart in Freebase for only 7% of the ComQA questions. This suggests an oracle F1 score of 93.0. For Fader et al. (2013), which is over web extractions, we mapped Wikipedia URLs to their titles.

6.6.2 Results

Table 6.4 shows the performance of baselines on the ComQA test set. Overall, the systems achieved poor performance, suggesting that current methods cannot handle the complexity of our dataset, and that new models for QA are needed. Table 6.5 compares the performace of the systems on different datasets. For example, while Abujabal et al. (2017) achieved an F1 score of 51.0 on WebQuestions, it achieved 22.4 on ComQA.

The performance of Fader et al. (2013) is worse than the others due to the incompleteness of its underlying extractions and the complexity of ComQA questions that require higher-order relations and reasoning. However, the system answered complex questions, which KB-QA systems failed to answer. For example, it successfully answered *"What is the highest mountain in the state of Washington?"*. The answer to such a question is more readily available in Web text, compared to a KB, where more sophisticated "reasoning" would be required to handle the superlative. While text can easily answer questions like the one above, a slightly modified question e.g., *"What is the **fifth** highest mountain in the state of Washington?"* might not be explicitly found in text, which can be answered using KBs. Both examples above demonstrate the benefits of combining text and structured resources.

6.6.3 Discussion

For the two best performing systems on ComQA, QUINT (Abujabal et al., 2017) and AQQU (Bast and Haussmann, 2015), we manually inspected ComQA questions on which they failed, 100 questions per system. We classified failure sources into four categories: compositionality, temporal, comparison or NER. Table 6.6 shows the distribution of failure sources of both systems.

Compositionality. Both systems could not handle the compositional nature of questions. For example, they returned the father of Julius Caesar as an answer for *"What did Julius Caesar's father work as?"*. However, the question requires another KB predicate that connects the father to his profession. For *"John Travolta and Jamie Lee Curtis starred in this movie?"*, both systems returned the movies Jamie Lee Curtis appeared in, ignoring the part constraint that John

Travolta should appear in them as well. Answering multi-relation questions over KBs remains an open problem given the large number of KB relations.

Temporal. Our analysis reveals that the tested systems often fail to capture temporal constraints in questions, be it explicit or implicit. For example, for *"Who won the Oscar for Best Actress in 1986?"*, both systems returned all winners and ignored the explicit year in the question together with the temporal aspect '*in 1986*'. Implicit temporal constraints e.g., named events like '*Vietnam war*' in *"Who was the president of the US during Vietnam war?"* pose a challenge to current methods. Such constraints need first to be detected and normalized to a canonical time interval (Novermber 1st, 1955 to April 30th, 1975 for the Vietnam war). Then, these systems need to compare the terms of the US presidents with above interval to account for the temporal relation of '*during*'. While detecting explicit time expressions can be done reasonably well using existing time taggers (Chang and Manning, 2012), identifying implicit ones is still a challenge. Furthermore, retrieving the correct temporal scopes of entities in questions (e.g., the terms of the US presidents) is hard due to the large number of temporal KB predicates associated with entities.

Comparison. Both systems perform poorly on comparison questions (comparatives, superlatives, and ordinals), which is expected since they were not designed to address those. To the best of our knowledge, no existing KB-QA system can handle comparison questions. Note that our goal is not to assess the quality the of current methods, but to highlight that these methods miss categories of questions that are important to real users. For *"What is the first film Julie Andrews made?"* and *"What is the largest city in the state of Washington?"*, both systems returned the list of Julie Andrews's films and the list of Washington's cities, for the first and the second questions, respectively. While the first question requires the temporal attribute of `filmReleasedIn` to order by, the second question needs the attribute of `hasArea`. Identifying the correct attribute to order by as well as determining the order direction (ascending for the first and descending for the second) is challenging and out of scope for current methods.

NER. NER errors come from false negatives, where entities are not detected, or false positives, where systems identify entities not intended as such. For example, in *"On what date did the Mexican Revolution end?"* QUINT identified '*Mexican*', rather than '*Mexican Revolution*' as an entity. For the latter case, the question *"What is the first real movie that was produced in 1903?"* does not ask about a specific entity. QUINT could not generate SPARQL queries and returned empty answer. Existing QA methods expect a pivotal entity in a

question, which is not always the case.

Note that while baseline systems achieved low precision, they achieved higher recall (21.2 vs 38.4 for QUINT, respectively) (Table 6.4). This reflects the fact that these systems often cannot cope with the full complexity of ComQA questions, and instead end up evaluating underconstrained "interpretations" of the question.

To conclude, current methods can handle simple questions very well, but struggle with complex questions that involve multiple conditions on different entities or need to join the results from sub-questions. Handling such complex questions, however, is important if we are to satisfy information needs expressed by real users.

Property	**Example**	**Percentage%**
Compositional questions		
Conjunction	*"What is the capital of the country whose northern border is Poland and Germany?"*	17.67
Nested	*"When is Will Smith's oldest son's birthday?"*	14.33
Temporal questions		
Explicit time	*"Who was the winner of the World Series* ***in 1994?****"*	4.00
Implicit time	*"Who was Britain's leader* ***during WW1?****"*	4.00
Temporal answer	*"****When*** *did Trenton become New Jerseys capital?"*	15.67
Comparison questions		
Comparative	*"Who was the first US president to serve* ***2 terms****?"*	1.00
Superlative	*"What ocean does the* ***longest*** *river in the world flow into?"*	14.33
Ordinal	*"When was Thomas Edisons* ***first*** *wife born?"*	14.00
Question formulation		
wh- word	*"****When*** *did Trenton become New Jerseys capital?"*	92.00
Telegraphic	*"Neyo first album?"*	8.00
Entity distribution in questions		
Zero entity	*"What public company has the most employees in the world?"*	2.67
Single entity	*"Who is* ***Brad Walst****'s wife?"*	75.67
Multi-entity	*"What country in* ***South America*** *lies between* ***Brazil*** *and* ***Argentina****?"*	21.67
Other features		
Answer Tuple	*"****Where*** *was Peyton Manning born and* ***what year was he born****?"*	2.00
Empty answer	*"Who was the first human being on Mars?"*	3.67

Table 6.2: Results of the manual analysis of 300 questions. Note that properties are not mutually exclusive.

Dataset	**Size**	**Compositional**	**Temporal**	**Comparison**	**Telegraphic**	**Empty Answer**
ComQA	11,214	32%	24%	30%	8%	4%
WebQuestions (Berant et al., 2013)	5,810	2%	7%	2%	0%	0%
ComplexQuestions (Bao et al., 2016)	2,100	39%	34%	9%	0%	0%

Table 6.3: Comparison of ComQA with existing datasets over various phenomena. We manually annotated 100 random questions from each dataset.

	Avg. Precision	**Avg. Recall**	**Avg. F1**
Abujabal et al. (2017)	21.2	38.4	22.4
Bast and Haussmann (2015)	20.7	37.6	21.6
Berant and Liang (2015)	10.7	15.4	10.6
Berant et al. (2013)	13.7	20.1	12.0
Fader et al. (2013)	7.22	6.59	6.73

Table 6.4: Results of baselines on ComQA test set.

	WebQuestions F1	**Free917 Accuracy**	**ComQA F1**
Abujabal et al. (2017)	51.0	78.6	22.4
Bast and Haussmann (2015)	49.4	76.4	21.6
Berant and Liang (2015)	49.7	–	10.6
Berant et al. (2013)	35.7	62.0	12.0

Table 6.5: Results of baselines on different KB-QA datasets including ComQA.

Category	**QUINT**	**AQQU**
Compositionality error	39%	43%
Missing comparison	31%	26%
Missing temporal constraint	19%	22%
NER error	11%	9%

Table 6.6: Distribution of failure sources on ComQA questions on which QUINT and AQQU failed.

7 Neural Named Entity Recognition from Subword Units

7.1 Introduction

Named Entity Recognition (NER) is an important task in language technology applications, such as smart voice-controlled devices like the Amazon Echo or Google Home. For example, if a user requests a voice-controlled assistant to *"play we are the champions by queen"*, a named entity recogniser determines that the phrase '*we are the champions*' refers to a `song` and '*queen*' to an `artist`. As new utterances are collected over time, which are annotated with named entities, regular retraining with increasing data amounts is needed.

Recently, several neural models for NER have been proposed (e.g., (Chiu and Nichols, 2016; Lample et al., 2016)), indicating promising performance on a rather small and artificially generated dataset (Sang, 2002; Sang and Meulder, 2003). However, these models rely mostly on dedicated word-level representations, which suffer from three main shortcomings:

- The vocabulary size is large, yielding a large number of parameters, and hence, large memory requirements and training time, which is particularly problematic if large amounts of data are available.
- The models cannot learn subword representations, which can potentially improve performance by taking advantage of morphology.
- Out-of-vocabulary words cannot be handled.

In this paper, we adopt a neural solution relying on subword units, namely characters, phonemes and bytes. For each word in an utterance, we learn representations from each of the three subword units. The character-level unit looks at the characters of each word to learn morphological information, while the phoneme-level unit treats a word as a sequence of phonemes, using lexica that map a given word into its corresponding phoneme sequence. The byte-level unit reads a word as bytes, where we use the variable length UTF-8 encoding.

Subword-based models have much smaller vocabulary sizes compared to mod-

els that rely on dedicated word-level embeddings (332 vs 74K for English). Furthermore, subword units enable the model to mitigate the out-of-vocabulary problem. Character-level networks have already proven to boost the performance of many sequence tagging tasks, including POS tagging and NER, in particular for morphologically rich languages (Chiu and Nichols, 2016; Klein et al., 2003; Lample et al., 2016; Yang et al., 2016b). However, while characters have been successfully used to boost NER performance, combining different types of subword units has not yet been explored.

We opt for a neural solution based on bidirectional LSTMs and conditional random fields (CRF) (Chiu and Nichols, 2016; Huang et al., 2015; Lample et al., 2016) in which we integrate the subword units. The output of the three subword units concatenated is fed into a bidirectional LSTM that captures word-level dependencies, and subsequently, into a CRF layer for decoding (Figure 7.1).

We present experiments on a real-world large dataset covering four languages (English, German, and two more Eurpoean languages), with up to 5.5M utterances per language. The utterances constitute customer requests to voice-controlled devices; all utterances were manually transcribed and annotated with named entities. Our experiments show that:

- With increasing training data size, performance of models trained solely on subword units becomes closer to that of models with dedicated word-level embeddings (91.35 vs 93.92 F1 for English), however, with much smaller vocabulary size (332 vs 74K).
- Subword units can enhance models with dedicated word-level embeddings, in particular, for languages with smaller training data sets.
- Combining the three subword units (character-, phoneme- and byte-level) yields better results than using only one or two of them.

7.2 Related Work

Traditional NER. NER is a widely studied problem in the NLP community, where methods have been characterized by the use of CRFs with heavy feature engineering, gazetteers and external knowledge resources (Finkel et al., 2005; Florian et al., 2003; Kazama and Torisawa, 2007; Klein et al., 2003; Lin and Wu, 2009; Radford et al., 2015; Ratinov and Roth, 2009; Zhang and Johnson, 2003).

Ratinov and Roth (2009) use non-local features and gazetteers extracted from Wikipedia, while Kazama and Torisawa (2007) harness type information of candidate entities. In our work, we opt for a neural solution without hand-crafted

features or external resources.

Neural NER. Recently, the focus has shifted towards adopting neural architectures for NER (Chiu and Nichols, 2016; Collobert et al., 2011; Gillick et al., 2016; Huang et al., 2015; Lample et al., 2016; dos Santos and Guimarães, 2015; Yadav et al., 2018; Yang et al., 2016b). Huang et al. (2015) use a word-level bidirectional LSTM-CRF for several sequence tagging problems including POS tagging, chunking and NER. However, instead of using subword units e.g., character-level, they made use of heavy feature engineering to extract character-level features. Lample et al. (2016) extend the previous model by appending a character-level bidirectional LSTM-based unit, where a word is represented by concatenating dedicated word-level embeddings and embeddings learned from its characters. Bharadwaj et al. (2016) represent words as sequences of phonemes, which serve as universal representation across languages to facilitate cross-lingual transfer learning. Chiu and Nichols (2016) use a convolutional nerual network to learn character-level embeddings and LSTM units on the word level. dos Santos and Guimarães (2015) propose the CharWNN network, a similar neural architecture to that of Chiu and Nichols (2016). Gillick et al. (2016) employ a sequence-to-sequence model with a novel tagging scheme. The model relies on bytes, allowing the joint training on different languages for NER and eliminating the need for tokenization. Finally, Yang et al. (2016b) adopt a very similar architure to that of Lample et al. (2016), however, they replaced LSTMs with GRUs units. Furthermore, they stud the multi-lingual and multi-task joint training, which we plan to address in the future.

Overall, existing methods mostly utilize dedicated word embeddings rather than subword units. While some work has also addressed characters or bytes, using phonemes or a combination of different subword units have not been explored, which we address in this work.

7.3 Model

We follow recent works on named entity recognition and base our soultion on bidirectional LSTMs and conditional random fields (CRF) (Chiu and Nichols, 2016; Huang et al., 2015; Lample et al., 2016). Our model relies on bidirectional LSTM-based subword units to learn word representations, namely character-level, phoneme-level and byte-level units. For each word in an utterance, our model learns a low-dimensional representation from each subword unit, which are then concatenated together and fed into a word-level bidirectional LSTM. The output of the word-level bidirectional LSTM is fed into a CRF layer for decoding.

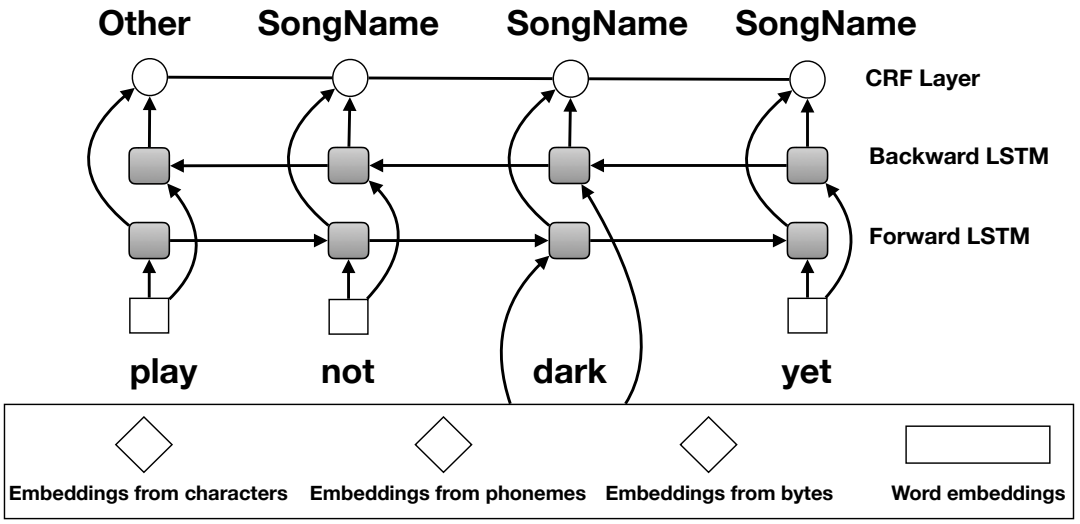


Figure 7.1: Our model with a bidirectional LSTM layer and CRF layer for decoding. For each word in an utterance, our model learns embeddings from the three subword units. Note that the word embeddings are optional.

Our model is depicted in Figure 7.1. Next we describe the different components of our model, including the existing bidirectional LSTM-CRF network and our subword units.

7.3.1 Recurrent Neural Networks

Recurrent neural networks (RNNs) are powerful models for sequential data that enable modeling dependencies among input tokens, be they words, characters, phonemes or bytes. By using RNNs we can condition on entire utterances, allowing us to abandon the Markov assumption that was prevalent in statistical NLP (Goldberg, 2017). The input of an RNN network is a sequence of fixed-sized d-dimensional vectors $(\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n)$, where each vector $\mathbf{x}_t$ represents the token at position t and n is the number of tokens. The intermediate output of an RNN is another sequence of fixed-sized vectors $(\mathbf{h}_1, \mathbf{h}_2, ...\mathbf{h}_n)$ that encodes information about the input sequence at every time step:

$$h_t = f(Ux_t + Wh_{t-1}), y_t = g(Vh_t), \tag{7.1}$$

where U, W, V are shared weight matrices, f is a nonlinearity such as tanh or ReLU, g is a softmax function and y_t is the final output at time t. RNNs are also good at summarizing/encoding sequences by translating an input sequence into a

low-dimensional fixed-sized vector that encompasses the important information required for the task at hand. In this scenario, the last hidden state $\mathbf{h}_n$ represents the summary.

7.3.2 Bidirectional LSTMs

In theory, RNNs are capable of capturing long-term dependencies, however, in practice, they fail to do so due to the vanishing/exploding gradient problem (Bengio et al., 1994). As a remedy, new architectures have been proposed including Long Short-term Memory networks (LSTMs) and Gated Recurrent Units (GRUs), where *memory-cells* are incorporated to retain relevant information from the history and to propagate it through the network Cho et al. (2014); Hochreiter and Schmidhuber (1997). Different *gates* control the proportion of the input and history to retain in memory cells. In this work we use the following LSTM implementation, which was adopted by existing works (Lample et al., 2016):

$$\begin{aligned} i_t =& \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\ c_t =& (1 - i_t) \odot c_{t1} + \\ & i_t \odot tanh(W_{xc}x_t + W_{hc}h_{t1} + b_c) \\ o_t =& \sigma(W_{xo}x_t + W_{ho}h_{t1} + W_{co}c_t + b_o) \\ h_t =& o_t \odot tanh(c_t), \end{aligned} \tag{7.2}$$

where W's are shared weight matrices, b's are the biases, σ is the element-wise sigmoid function, and $\odot$ is the element-wise product.

A *forward* LSTM network, as described above, captures the past at a given token. However, for certain tasks including NER, capturing the future is also desirable and can lead to significant performance improvement. This can be achieved by reading the same input sequence in a reverse order using another LSTM network. A network that captures the future is called *backward* LSTM. To capture both the past and future at a given token, Graves and Schmidhuber (2005) coined the *bidirectional* LSTM network, where the outputs of the forward ($\overrightarrow{h_t}$) and backward ($\overleftarrow{h_t}$) networks are concatenated together at each time step of the input sequence $h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$. For sequence tagging problems, a softmax layer is used on top of the output of the bidirectional LSTM network to calculate a probability distribution of output tags for a given token.

7.3.3 Conditional Random Fields

Bidirectional LSTM networks capture dependencies among input tokens, however, they assume independence among output tags, which is not practical for several tasks including NER. To remedy this, instead of using a softmax layer, a CRF layer is incorporated Lafferty et al. (2001), where h_t's are treated as features of the CRF layer (Huang et al., 2015; Yang et al., 2016b).

Concretely, a CRF layer has a transition matrix $\mathbf{A}$ of size $k \times k$ as parameters, where k is the number of distinct tags, and $A_{i,j}$ represents the score of a transition from the tag i to tag j. For a given input sequence $\mathbf{x} = (x_1, x_2, ..., x_n)$, we treat the output matrix $\mathbf{P}$ of size $n \times k$ of the word-level bidirectional LSTM network as the network scores, where $P_{i,j}$ represents the score of the j^{th} tag of the i^{th} token in a sequence. For an output tag sequence $\mathbf{y} = (y_1, y_2, ..., y_n)$, we define its score to be the sum of transition scores and network scores:

$$s(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} A_{y_{i-1}, y_i} + \sum_{i=1}^{n} P_{x_i, y_i},$$

where we set y_0 to be a START token. During training, we maximize the log-probability of the correct tag sequence $log(p(\mathbf{y}|\mathbf{x}))$, where

$$p(\mathbf{y}|\mathbf{x}) = \frac{e^{s(\mathbf{x}, \mathbf{y})}}{\sum_{\tilde{y} \in \mathbf{Y}} e^{s(\mathbf{x}, \tilde{\mathbf{y}})}},$$

where $\mathbf{Y}$ represents all possible output tag sequences. We use dynamic programming to compute A and the optimal output tag during decoding. See (Lafferty et al., 2001) for details.

7.3.4 Subword Units

In this work we rely on subword units, namely characters, phonemes and bytes to learn embeddings that represent the full word. Subword units enable the model to mitigate the out-of-vocabulary problem and to have smaller vocabulary sizes compared to models that rely on dedicated word-level embeddings. Our model extends the state-of-the-art by using phoneme-level and byte-level networks as shown in Figure 7.2. Each subword unit in our model is a bidirectional LSTM network, where the last hidden state of the forward and the backward networks are concatenated, which constructs V_c, V_{ph} and V_{by} from the character-, phoneme- and byte-level networks, respectively. The vectors V_c, V_{ph} and V_{by} are, in turn, concatenated to represent the final embeddings of a word. Optionally, we can

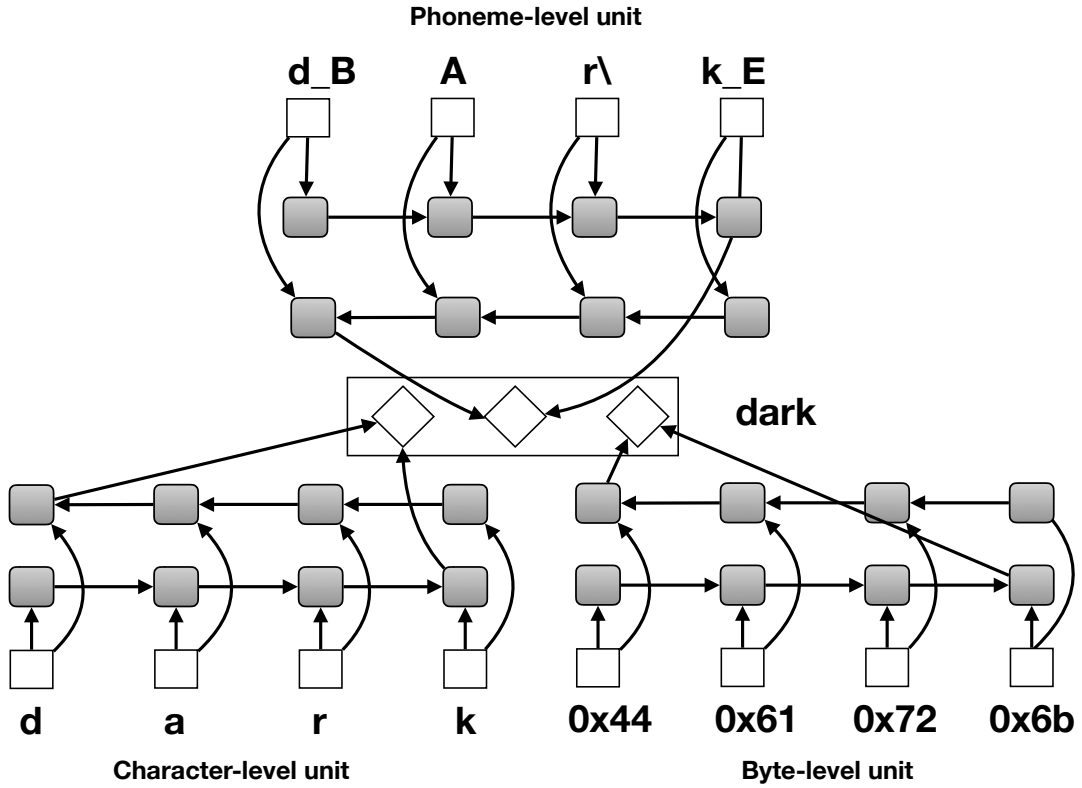


Figure 7.2: Our bidirectional LSTM-based subword units. The outputs of the three units are concatenated to learn embeddings for the whole word.

concatenate dedicated word embeddings that are either randomly initialized or pre-trained.

For the phoneme-level unit, we use lexica that map a given word to its phoneme sequence. We use the X-SAMPA phoneme set. For example, the word '*dark*' is mapped to $\{d\_B, A, r\backslash, k\_E\}$, where $\_B$ marks the first phoneme in a word, while $\_E$ marks the last one. We add a special symbol UNK to the phoneme set, to which we map missing words in our lexica. With these extra symbols and others to express stress and diacritics we have 265 phonemes in total, with 60 unique ones. In general, for the setting explored in our work, i.e. voice-controlled devices, phoneme lexica with good coverage are developed for the agents text to speech (TTS) and automated speech recognition (ASR) components which can be re-used. In case lexica with good coverage are not available, tools for grapheme to phoneme conversion can be used.

For the byte-level unit, we use the variable length UTF-8 encodings to keep the vocabulary as small as possible. For example, '*Guimarães*' is represented as {0x47 0x75 0x69 0x6d 0x61 0x72 0xc3 0xa3 0x65 0x73}. Note that the character

	EN	DE	FR	ES
Size of train set	3.3M	0.6M	12K	8K
Size of dev set	1.1M	0.2M	4K	2.6K
Size of test set	1.1M	0.2M	4K	2.6K

Table 7.1: Number of utterances of per language.

ã corresponds to two bytes, {0xc3 0xa3}. This distinguishes this unit from the character-level one.

7.4 Experimental Evaluation

Our experiments serve two goals: 1) to show that models trained on the three subword units combined achieve reasonable performance across languages, which is very close to the performance of those models with dedicated word-level embeddings, in particular, as the size of train data increases, and that 2) our subword units can enhance models that harness dedicated word-level embeddings, in particular, for resource-poor languages.

7.4.1 Benchmark

We use a large dataset covering four different languages, namely American English (EN), German (DE), French (FR) and Spanish (ES). The data is representative of user requests to voice-controlled devices, which were manually transcribed and annotated with named entities. Overall, the data covers several domains, comprising different intents and named entities. The dataset represents a slice of the original data, which was sampled at random. Table 7.1 shows the size of train, dev, and test set splits for each language. While a large number of utterances is available for languages which were collected with deployed systems (EN and DE), rather few are available for the other two languages. On average, we have 36 types of named entities per language.

7.4.2 Performance Measures

We use the CoNLL script (Sang, 2002) to compute precision, recall and an F1 scores on a per-token basis i.e., we allow for partial matching. We report the average F1 score.

Lang	Subwords	Word-level
EN	332	74K
DE	225	46K
FR	148	18K
ES	120	3.7K

Table 7.2: Vocabulary size of models trained solely on subword units versus models used word embeddings.

Lang	Char	Phoneme	Byte	Char + Phoneme	Char + Byte	Phoneme + Byte	All
EN	89.63	90.03	89.7	91.15	90.58	91.1	**91.35**
DE	84.94	84.21	84.95	86.81	86.32	86.76	**87.37**
FR	80.57	73.65	80.4	80.1	**82.44**	82.15	81.05
ES	67.64	62.4	67.07	69.06	70.33	68.94	**71.07**

Table 7.3: F1 scores of the subword-only models with different units being used. The model with the three subword units combined achieved best performance across languages, except for FR.

7.4.3 Training

We used a mini-batch Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.0007 for all the models presented in this paper. We tried different optimizers with different learning rates (e.g., stochastic gradient descent), however, they performed worse than Adam. The batch size was set to 1024, 256, 4 and 4 utterances for EN, DE, FR and ES, respectively. The embedding dimension of the subword units is set to 35, while its counterpart of the word-level network is set to 64 (in case dedicated word-level representations are used). Both subword and word-level networks have a single layer for the forward and the backward LSTMs whose dimensions are set to 35 and 128, respectively. We tried several different values, however, the performance was inferior to the one reported with the above values. When a given number of epochs is reached (40), training is terminated. The model with the best F1 score on the development set is used to make predictions [1].

We used dropout training Hinton et al. (2012), applying a dropout mask in two places: 1) to the final embedding layer just before the input to the word-level bidirectional LSTM in Figure 7.1, with dropout rate set to 0.5, and 2) to the input of each subword unit, with dropout rate set to 0.2. The latter is

[1]We extended the model at `https://github.com/glample/tagger`

Lang	Subwords	Word-level	Combined
EN	91.35	93.92(+2.57)	94.02(+0.1)
DE	87.37	90.12(+2.75)	90.25(+0.13)
FR	82.44	86.87(+4.43)	87.45(+0.58)
ES	71.07	79.43(+8.36)	80.23(+0.8)

Table 7.4: Comparison of subword only models versus word-level models and models combined word-level and subwrod units. Numbers correspond to F1 values.

used whenever dedicated word-level embeddings are harnessed. Otherwise, no dropout is applied to the input of the subword units.

A notable advantage of relying on subword units is the very small vocabulary compared to models with dedicated word embeddings. Table 7.2 shows the vocabulary size of different languages for models that rely on the three suboword units combined versus models with dedicated word embeddings. In terms of model complexity, subword-based models have less number of parameters. For example, for EN, they have 74K $*$ 64 = 4.7M fewer parameters to fine-tune during training.

7.4.4 Results and Discussion

Subwords only models. Table 7.3 shows the performance of our models that solely rely on subword units. In particular, we trained models with different combinations of subword units. Models with the three subword units combined achieved the best results across languages, demonstrating that our subword units work in concert. Combining the different subword units always yields superior results to models with only one subword unit. Additionally, we trained models for the different languages using only a single subword unit, however, with higher embedding and LSTM hidden dimensions. The performance was inferior to that reported in Table 7.3 (last column). This shows that gains result from the better representations our models learn from the three units combined and not from the higher dimensionality of the hidden embedding representation when we combine multiple types of subword units.

Word-level models. We train models with dedicated word-level embeddings, however, with no subword units as well as models that combine both subword units and word-level embeddings. Results are shows in Table 7.4. With increasing training data size, performance for models trained solely on subword

Lang	Char	Phoneme	Byte	Char + Phoneme	Char + Byte	Phoneme + Byte	All
EN	93.96	93.99	93.99	**94.02**	93.89	93.97	93.88
DE	90.17	90.02	90.17	**90.25**	90	90.19	90.1
FR	86.37	86.38	86.49	**87.45**	85.98	85.86	86.38
ES	79	80.03	79.08	79.57	79.1	**80.23**	78.72

Table 7.5: F1 scores of models that combine both word-level embeddings and different subword units.

units becomes closer to that of models with dedicated word-level embeddings (91.35 vs 93.92 F1 for EN), however, with much smaller vocabulary size (332 vs 74K). The gap in performance increases as the size of train data decreases (71.07 vs 79.43 F1 for ES). In other words, with sufficient training data, subword units achieve comparable results to models with dedicated word-level embeddings.

Combined models. Finally, models that combine both dedicated word-level embeddings and subword units achieved best results as shown in Table 7.4 (last column). As train data decreases (resource-poor languages), the positive effect of subword units increases (+0.1 F1 point for EN and +0.8 F1 point for ES). In Table 7.5 we show combined models with different combinations of subword units. The combination of character- and phoneme-level units together with the dedicated word embeddings achieved the best results for three out of the four languages. This shows the that a phoneme-level unit is useful for NER.

Out-of-vocabulary words. 625 utterances of the ES test set contain at least one out-of-vocabulary word, with 703 out-of-vocabulary words in total. F1 scores on the 625 utterances are 44.61, 50.15 and 51.01 for subwords only, word-level and combined models, respectively. These results follow the trends observed in Table 7.4.

We also computed F1 scores on the out-of-vocabulary words, where, interestingly and for the first time, subwords only model outperformed the corresponding word-level model (34.93 vs 34.81), while combined model achieved an F1 score of 37.11. Note that we have here a generally higher boost using combined models (2.3 F1 points increase). This experiment shows that our subword units help in the presence of out-of-vocabulary words.

8 Conclusion

8.1 Summary

In this dissertation, we presented four methods and a curated dataset in the areas of question answering over knowledge bases and named entity recognition.

Our first contribution is QUINT, a KB-QA system that automatically learns question-query templates with fine-grained alignments between phrases in questions and conditions in queries. In addition to answering simple questions, QUINT answers compositional complex questions by utilizing language compistionality, where a compositional question is decomposed into simpler sub-questions that can be answered independently.

Our second contribution is NEQA, a never-ending learning framework for template-based KB-QA. NEQA starts with a small seed of training examples and acquires more after deployment by relying on a light-weight user feedback on answers. NEQA supports true open-domain KB-QA by harnessing the interplay between syntax and semantics. On the syntax side, NEQA relies on role-aligned templates to answer user questions. When templates fail, NEQA consults a semantic similarity function to retrieve a semantically-similar and correctly-answered question from its history. NEQA then learns a new template from the question the templates-based answering mechanism failed on. The new template is used for answering subsequent questions with similar syntactic structures. NEQA periodically retrains its underlying models on newly acquired examples which allows it to improve its performance over time as well as adapt itself to the terminology being used after deployment.

Our third contribution, TEQUILA, allows users to ask complex questions with explicit and implicit temporal constraints. Moreover, TEQUILA handles questions with ordinal conditions and questions seeking temporal information. TEQUILA relies on a rule-based framework that decomposes temporal questions into sub-questions, with a form of rewriting, if needed. Sub-questions can be answered using standard KB-QA systems e.g., QUINT and NEQA. Along with TEQUILA, we introduced TempQuestions, a dataset of 1,271 temporal-only questions with answers over Freebase. The questions are chosen such that

many of them require a combination of evaluating sub-questions and reasoning over sub-results (results of the sub-questions). This collection is derived by judiciously selecting time-related questions from the Free917, WebQuestions and ComplexQuestions sets.

Our fourth contribution is ComQA, a large-scale dataset of real user questions with several types of complex questions such as the need for temporal reasoning, comparison (e.g., comparatives, superlatives, ordinals), compositionality (multiple, possibly nested, subquestions with multiple entities), and questions with empty answer sets. Through a large crowdsourcing effort, the 11,214 questions in ComQA are grouped into 4,834 paraphrase clusters that express the same information need. Each cluster is annotated with its answer(s). Building on the wide adoption of Wikipedia, ComQA answers come in the form of Wikipedia entities wherever possible. Wherever the answers are temporal or measurable quantities, TIMEX3 and the International System of Units (SI) are used for standardization. In this manner, the dataset is largely resource independent.

Our fifth and last contribution is a neural model based on subword units for named entity recognition. Designing efficient, in terms of memory requirements and training time, and yet high-accuracy solutions for NER for such assistants is challenging. To this end, we rely on subword units, namely *characters*, *phonemes* and *bytes*. For each word in an utterance, we learn representations from each of the three subword units. The character-level unit looks at the characters of each word, while the phoneme-level unit treats a word as a sequence of phonemes, using lexica that map a given word into its corresponding phoneme sequence. The byte-level unit reads a word as bytes, where we use the variable length UTF-8 encoding. A major advantage of subword-based models is the small vocabulary size which can positively affect memory requirements and training time of models, which is particularly relevant for large-scale applications. In addition, the use of subword-units could improve modelling of out-of-vocabulary words and support learning of morphological information.

8.2 Outlook

We conclude this dissertation by discussing future research directions that we believe are crucial for advancing question answering systems with the ability to interact with users in a more natural way.

Complex QA

Although both QUINT and TEQUILA address complex questions, the breadth

of such questions is large. While QUINT focuses on questions that require decomposition into simpler sub-questions, TEQUILA handles questions with temporal conditions. Count, aggregation and comparative questions, among others are interesting kinds of complex questions. Such questions require new innovative methods as translating such questions into logical representations on the KB side is challenging. Our dataset, ComQA, which contains complex questions of several types would help the research community to pursue this goal.

Interactive QA

With the inherent ambiguity of natural language, which is exacerbated when the user poses complex questions, and the wide availability of voice-controlled assistants such the Amazon Echo, it becomes more natural to teach machines to ask the user back clarification questions on phrases or segments in questions, which could not be parsed. This requires fine-grained alignments between phrases in questions and the constituents of queries, so as to enable the system to pin point sources of failures and generate clarification questions on these. Additionally, natural language generation would be a crucial component.

List of Figures

1.1 An example of a hand-crafted role-aligned question-query template (top). Shared *pred* and *ent* annotations indicate an alignment between a phrase in the question template and a KB semantic item in the corresponding query template. An instantiation of the template is shown at the bottom. 3

2.1 A fragment of a knowledge base in (a) tabular and (b) graph form. 11

3.1 Example KB fragment. 19
3.2 An outline of how QUINT generates role-aligned templates at training time and how it uses them for question answering. 19
3.3 The dependency parse tree of our example utterance (top) together with the answer set (bottom). 20
3.4 Backbone query $\hat{q}$. 21
3.5 Backbone query $\hat{q}$ with types. 26
3.6 Bipartite graph provided to the ILP, with phrases at the top and semantic items at the bottom. An edge corresponds to a mapping from a phrase to a semantic item, and its thickness corresponds to the mapping weight. Dashed box represents $S(\texttt{?x})$ for `?x` in Figure 3.5. 27
3.7 Aligned utterance query pair (u, q, m). m is indicated by shared *ent*, *pred*, and *type* annotations (e.g., *"played on"* is aligned with `cast.actor`). 28
3.8 A template $t = (u_t, q_t, m_t)$. m_t is indicated by shared *ent*, *pred*, and *type* annotations. Figure 3.7 shows the concrete utterance-query pair used to generate this template. 29
3.9 Template instantiation (using t in Figure 3.8). 30
3.10 (a) shows the original dependency parse of the question before rewriting (b) shows the dependency parse after rewriting i.e., dropping dashed edges in (a) and adding a new edge (bold) connecting *"Gump"* to *"in"*. 33

4.1 Continuous learning: if a new question u_{new} cannot be satisfactorily answered via templates, we utilize user feedback on the output of a semantic similarity function to learn a new template (u^t_{new}, q^t) based on u_{new}. 42
4.2 An aligned question-query pair (u, q). Alignment is indicated by shared *ent*, *pred*, and *class* annotations. 48
4.3 A template composed of aligned question and query templates (u^t, q^t). Shared *ent*, *pred*, and *class* annotations indicate alignment between u^t and q^t. 49
4.4 Performance of NEQA with and without user feedback as a function of batch number, on the WebQuestions and ComplexQuestions datasets. 55
4.5 Performance of AQQU (Bast and Haussmann, 2015), QUINT (Abujabal et al., 2017) and NEQA over the 20 batches of the WebQuestions test set. 57

5.1 The 13 temporal relations (nos. 2 through 7 have inverses) between two intervals X and Y, as in Allen (1983). 68

6.1 Paraphrase clusters from ComQA, covering a range of question aspects, with lexical and syntactic diversity. 82
6.2 All ten questions belong to the same original WikiAnswers cluster. AMT Turkers split the original cluster into four new ones. 87
6.3 Distribution of the number of questions per cluster. 89
6.4 Answer types and question topics on 300 annotated examples as word clouds. The bigger the font, the more frequent the concept. . 92

7.1 Our model with a bidirectional LSTM layer and CRF layer for decoding. For each word in an utterance, our model learns embeddings from the three subword units. Note that the word embeddings are optional. 102
7.2 Our bidirectional LSTM-based subword units. The outputs of the three units are concatenated to learn embeddings for the whole word. 105

List of Tables

2.1 Fragment of our lexica: $\mathcal{L}_P$ and $\mathcal{L}_C$. Semantic items are from Freebase. 14

3.1 Curated templates by Fader et al. (2013). Shared *pred* and *ent* annotations indicate an alignment between a phrase in the utterance template and a KB semantic item in the corresponding query template. 17
3.2 Fragment of our lexicons: $\mathcal{L}_P$ and $\mathcal{L}_C$. 28
3.3 Query candidate ranking features. 31
3.4 Results on the WebQuestions and Free917 test sets. 36
3.5 Anecdotal results from WebQuestions for both variants of QUINT: typed and untyped. 38
3.6 Results on ComplexQuestions. 39
3.7 Sample ComplexQuestions for both QUINT and AQQU (Bast and Haussmann, 2015). 40

4.1 NEQA initialization statistics. 53
4.2 Performance of continuous learning-based methods on the WebQuestions test set. User Feedback is used to re-train the systems after each batch. 54
4.3 Performance of state-of-the-art static learning-based methods on the WebQuestions test set. 59
4.4 F1 scores for a component ablation analysis of our similarity function. 61
4.5 Sample questions correctly answered via templates learned online. 61
4.6 Sample questions correctly answered using the similar questions retrieved from the question-query bank. Entities are generalized using [...] placeholders. 62

5.1 Decomposition and rewriting of questions. The *constraint* is the fragment after the SIGNAL word. *wh∗* is the question word, e.g., *who*, and w_i are tokens in the question. 70

5.2 Temporal reasoning constraints. 73
5.3 Distribution of question types by source. The total is greater than 1,271 as some questions have multiple tags. 75
5.4 Representative examples from TempQuestions. 76
5.5 Detailed performance of TEQUILA-enabled systems on TempQuestions and ComplexQuestions. The highest value in a column for each dataset is in **bold**. An asterisk (*) indicates statistical significance of TEQUILA-enabled systems over their standalone counterparts, under the 2-tailed paired t-test at $p < 0.05$ level. 78

6.1 Comparison of ComQA with existing QA datasets over various dimensions. 83
6.2 Results of the manual analysis of 300 questions. Note that properties are not mutually exclusive. 97
6.3 Comparison of ComQA with existing datasets over various phenomena. We manually annotated 100 random questions from each dataset. 97
6.4 Results of baselines on ComQA test set. 98
6.5 Results of baselines on different KB-QA datasets including ComQA. 98
6.6 Distribution of failure sources on ComQA questions on which QUINT and AQQU failed. 98

7.1 Number of utterances of per language. 106
7.2 Vocabulary size of models trained solely on subword units versus models used word embeddings. 107
7.3 F1 scores of the subword-only models with different units being used. The model with the three subword units combined achieved best performance across languages, except for FR. 107
7.4 Comparison of subword only models versus word-level models and models combined word-level and subwrod units. Numbers correspond to F1 values. 108
7.5 F1 scores of models that combine both word-level embeddings and different subword units. 109

Bibliography

Abujabal, A., Roy, R. S., Yahya, M., and Weikum, G. (2018). Never-ending learning for open-domain question answering over knowledge bases. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 1053–1062.

Abujabal, A., Yahya, M., Riedewald, M., and Weikum, G. (2017). Automated template generation for question answering over knowledge graphs. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 1191–1200.

Agichtein, E., Brill, E., and Dumais, S. T. (2006a). Improving web search ranking by incorporating user behavior information. In *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, pages 19–26.

Agichtein, E., Brill, E., Dumais, S. T., and Ragno, R. (2006b). Learning user interaction models for predicting web search result preferences. In *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, pages 3–10.

Agichtein, E., Carmel, D., Pelleg, D., Pinter, Y., and Harman, D. (2015). Overview of the TREC 2015 liveqa track. In *Proceedings of The Twenty-Fourth Text REtrieval Conference, TREC 2015, Gaithersburg, Maryland, USA, November 17-20, 2015*.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. G. (2007). Dbpedia: A nucleus for a web of open data. In *The Semantic Web,*

6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007., pages 722–735.

Balog, K., Bron, M., and de Rijke, M. (2011). Query modeling for entity search based on terms, categories, and examples. *ACM Trans. Inf. Syst.*, 29(4):22:1–22:31.

Balog, K., Meij, E., and De Rijke, M. (2010). Entity search: building bridges between two worlds. In *Proceedings of the 3rd International Semantic Search Workshop*, page 9. ACM.

Bao, J., Duan, N., Yan, Z., Zhou, M., and Zhao, T. (2016). Constraint-based question answering with knowledge graph. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2503–2514.

Bao, J., Duan, N., Zhou, M., and Zhao, T. (2014). Knowledge-based question answering as machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 967–976.

Bast, H. and Haussmann, E. (2015). More accurate question answering on freebase. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pages 1431–1440.

Bender, E. M., Flickinger, D., Oepen, S., Packard, W., and Copestake, A. A. (2015). Layers of interpretation: On grammar and compositionality. In *Proceedings of the 11th International Conference on Computational Semantics, IWCS 2015, 15-17 April, 2015, Queen Mary University of London, London, UK*, pages 239–249.

Bendersky, M., Wang, X., Metzler, D., and Najork, M. (2017). Learning from user interactions in personal search via attribute parameterization. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*, pages 791–799.

Bengio, Y., Simard, P. Y., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks*, 5(2):157–166.

Berant, J., Chou, A., Frostig, R., and Liang, P. (2013). Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544.

Berant, J. and Liang, P. (2014). Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1415–1425.

Berant, J. and Liang, P. (2015). Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.

Bharadwaj, A., Mortensen, D. R., Dyer, C., and Carbonell, J. G. (2016). Phonologically aware neural model for named entity recognition in low resource transfer settings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1462–1472.

Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22.

Blanco, R., Mika, P., and Vigna, S. (2011). Effective and efficient entity search in RDF data. In *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, pages 83–97.

Bollacker, K. D., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250.

Bordes, A., Chopra, S., and Weston, J. (2014). Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014,*

Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 615–620.

Bordes, A., Usunier, N., Chopra, S., and Weston, J. (2015). Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Brill, E., Dumais, S. T., and Banko, M. (2002). An analysis of the askmsr question-answering system. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, EMNLP 2002, Philadelphia, PA, USA, July 6-7, 2002*.

Cai, Q. and Yates, A. (2013). Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 423–433.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Jr., E. R. H., and Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*.

Chang, A. X. and Manning, C. D. (2012). Sutime: A library for recognizing and normalizing time expressions. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pages 3735–3740.

Chen, D. and Manning, C. D. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 740–750.

Chen, L., Jose, J. M., Yu, H., Yuan, F., and Zhang, D. (2016). A semantic graph based topic model for question retrieval in community question answering. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016*, pages 287–296.

Chiu, J. P. C. and Nichols, E. (2016). Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, pages 103–111.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Cui, H., Sun, R., Li, K., Kan, M., and Chua, T. (2005). Question answering passage retrieval using dependency relations. In *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*, pages 400–407.

Dalton, J., Dietz, L., and Allan, J. (2014). Entity query feature expansion using knowledge base links. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, pages 365–374.

Dang, H. T., Kelly, D., and Lin, J. J. (2007). Overview of the TREC 2007 question answering track. In *Proceedings of The Sixteenth Text REtrieval Conference, TREC 2007, Gaithersburg, Maryland, USA, November 5-9, 2007*.

Dang, H. T., Lin, J. J., and Kelly, D. (2006). Overview of the TREC 2006 question answering track 99. In *Proceedings of the Fifteenth Text REtrieval Conference, TREC 2006, Gaithersburg, Maryland, USA, November 14-17, 2006*.

Del Corro, L., Abujabal, A., Gemulla, R., and Weikum, G. (2015). FINET: context-aware fine-grained named entity typing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 868–878.

Del Corro, L. and Gemulla, R. (2013). Clausie: clause-based open information extraction. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 355–366.

Diefenbach, D., López, V., Singh, K. D., and Maret, P. (2018). Core techniques of question answering systems over knowledge bases: a survey. *Knowl. Inf. Syst.*, 55(3):529–569.

Dietz, L. and Gamari, B. (2017). Trec car: A data set for complex answer retrieval. version 1.4.(2017).

Dong, L., Mallinson, J., Reddy, S., and Lapata, M. (2017). Learning to paraphrase for question answering. In *EMNLP*, pages 875–886.

Dong, L., Wei, F., Zhou, M., and Xu, K. (2015). Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 260–269.

dos Santos, C. N. and Guimarães, V. (2015). Boosting named entity recognition with neural character embeddings. In *Proceedings of the Fifth Named Entity Workshop, NEWS@ACL 2015, Beijing, China, July 31, 2015*, pages 25–33.

Dumais, S. T., Banko, M., Brill, E., Lin, J. J., and Ng, A. Y. (2002). Web question answering: is more always better? In *SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 11-15, 2002, Tampere, Finland*, pages 291–298.

Fader, A., Zettlemoyer, L., and Etzioni, O. (2014). Open question answering over curated and extracted knowledge bases. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 1156–1165.

Fader, A., Zettlemoyer, L. S., and Etzioni, O. (2013). Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 1608–1618.

Ferrucci, D. A. (2012). Introduction to "this is watson". *IBM Journal of Research and Development*, 56(3):1.

Finkel, J. R., Grenager, T., and Manning, C. D. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 363–370.

Florian, R., Ittycheriah, A., Jing, H., and Zhang, T. (2003). Named entity recognition through classifier combination. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 168–171.

Gillick, D., Brunk, C., Vinyals, O., and Subramanya, A. (2016). Multilingual language processing from bytes. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1296–1306.

Goldberg, Y. (2017). *Neural Network Methods for Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Green, Jr., B. F., Wolf, A. K., Chomsky, C., and Laughery, K. (1961). Baseball: An automatic question-answerer. In *Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '61 (Western).

Guha, R. V., Brickley, D., and Macbeth, S. (2016). Schema.org: evolution of structured data on the web. *Commun. ACM*, 59(2):44–51.

Hakimov, S., Unger, C., Walter, S., and Cimiano, P. (2015). Applying semantic parsing to question answering over linked data: Addressing the lexical gap. In *Natural Language Processing and Information Systems - 20th International Conference on Applications of Natural Language to Information Systems, NLDB 2015 Passau, Germany, June 17-19, 2015 Proceedings*, pages 103–109.

Harabagiu, S. M., Lacatusu, V. F., and Hickl, A. (2006). Answering complex questions with random walk models. In *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, pages 220–227.

Harabagiu, S. M., Maiorano, S. J., and Pasca, M. (2003). Open-domain textual question answering techniques. *Natural Language Engineering*, 9(3):231–267.

Harabagiu, S. M., Pasca, M., and Maiorano, S. J. (2000). Experiments with open-domain textual question answering. In *COLING 2000, 18th International Conference on Computational Linguistics, Proceedings of the Conference, 2 Volumes, July 31 - August 4, 2000, Universität des Saarlandes, Saarbrücken, Germany*, pages 292–298.

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *14th International Conference on Computational Linguistics, COLING 1992, Nantes, France, August 23-28, 1992*, pages 539–545.

Hendrix, G. G., Sacerdoti, E. D., Sagalowicz, D., and Slocum, J. (1978). Developing a natural language interface to complex data. *ACM Trans. Database Syst.*, 3(2):105–147.

Herrera, J., Peñas, A., and Verdejo, F. (2004). Question answering pilot task at CLEF 2004. In *5th Workshop of the Cross-Language Evaluation Forum, CLEF*, pages 581–590.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Hirschman, L. and Gaizauskas, R. J. (2001). Natural language question answering: the view from here. *Natural Language Engineering*, 7(4):275–300.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hovy, E. H., Gerber, L., Hermjakob, U., Junk, M., and Lin, C. (2000). Question answering in webclopedia. In *Proceedings of The Ninth Text REtrieval Conference, TREC 2000, Gaithersburg, Maryland, USA, November 13-16, 2000*.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Iyer, S., Konstas, I., Cheung, A., Krishnamurthy, J., and Zettlemoyer, L. (2017). Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 963–973.

Jeon, J., Croft, W. B., and Lee, J. H. (2005). Finding similar questions in large question and answer archives. In *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 - November 5, 2005*, pages 84–90.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 133–142.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. (2017). Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611.

Joshi, M., Sawant, U., and Chakrabarti, S. (2014). Knowledge graph and corpus driven segmentation and answer inference for telegraphic entity-seeking queries. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1104–1114.

Jurafsky, D. and Martin, J. H. (2009). *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition*. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International.

Katz, B. (1997). Annotating the world wide web using natural language. In *Computer-Assisted Information Retrieval (Recherche d'Information et ses Applications) - RIAO 1997, 5th International Conference, McGill University, Montreal, Canada, June 25-27, 1997. Proceedings*, pages 136–159.

Kazama, J. and Torisawa, K. (2007). Exploiting wikipedia as external knowledge for named entity recognition. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pages 698–707.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Kivinen, J., Smola, A. J., and Williamson, R. C. (2004). Online learning with kernels. *IEEE Trans. Signal Processing*, 52(8):2165–2176.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 7-12 July 2003, Sapporo Convention Center, Sapporo, Japan.*, pages 423–430.

Klein, D., Smarr, J., Nguyen, H., and Manning, C. D. (2003). Named entity recognition with character-level models. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 180–183.

Kociský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. (2018). The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Krishnamurthy, J. and Mitchell, T. M. (2015). Learning a compositional semantics for freebase with an open predicate vocabulary. *Transactions of the Association for Computational Linguistics*, 3:257–270.

Kuzey, E., Setty, V., Strötgen, J., and Weikum, G. (2016). As time goes by: Comprehensive tagging of textual phrases with temporal scopes. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 915–925.

Kwiatkowski, T., Choi, E., Artzi, Y., and Zettlemoyer, L. S. (2013). Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1545–1556.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289.

Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. H. (2017). RACE: large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017*

Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017, pages 785–794.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 260–270.

Li, F. and Jagadish, H. V. (2014). Nalir: an interactive natural language interface for querying relational databases. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 709–712.

Li, H., Xiong, C., and Callan, J. (2017a). Natural language supported relation matching for question answering with knowledge graphs. In *Proceedings of the First Workshop on Knowledge Graphs and Semantics for Text Retrieval and Analysis (KG4IR 2017) co-located with the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017), Shinjuku, Tokyo, Japan, August 11, 2017.*, pages 43–48.

Li, L., Deng, H., Dong, A., Chang, Y., Baeza-Yates, R. A., and Zha, H. (2017b). Exploring query auto-completion and click logs for contextual-aware web search and query suggestion. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 539–548.

Li, X. and Roth, D. (2002). Learning question classifiers. In *19th International Conference on Computational Linguistics, COLING 2002, Howard International House and Academia Sinica, Taipei, Taiwan, August 24 - September 1, 2002*.

Liang, P., Jordan, M. I., and Klein, D. (2011). Learning dependency-based compositional semantics. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 590–599.

Liang, P. and Potts, C. (2015). Bringing machine learning and compositional semantics together. *Annu. Rev. Linguist.*, 1(1):355–376.

Lin, D. and Wu, X. (2009). Phrase clustering for discriminative learning. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 1030–1038.

Lin, J. J. (2002). The web as a resource for question answering: Perspectives and challenges. In *Proceedings of the Third International Conference on Language Resources and Evaluation, LREC 2002, May 29-31, 2002, Las Palmas, Canary Islands, Spain*.

Lin, J. J. (2007). An exploration of the principles underlying redundancy-based factoid question answering. *ACM Trans. Inf. Syst.*, 25(2):6.

Lin, J. J. and Katz, B. (2003). Question answering from the web using knowledge annotation and knowledge mining techniques. In *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management, New Orleans, Louisiana, USA, November 2-8, 2003*, pages 116–123.

López, V., Nikolov, A., Sabou, M., Uren, V. S., Motta, E., and d'Aquin, M. (2010). Scaling up question-answering to linked data. In *Knowledge Engineering and Management by the Masses - 17th International Conference, EKAW 2010, Lisbon, Portugal, October 11-15, 2010. Proceedings*, pages 193–210.

López, V., Tommasi, P., Kotoulas, S., and Wu, J. (2016). Queriodali: Question answering over dynamic and linked knowledge graphs. In *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part II*, pages 363–382.

Magnini, B., Vallin, A., Ayache, C., Erbach, G., Peñas, A., de Rijke, M., Rocha, P., Simov, K. I., and Sutcliffe, R. F. E. (2004). Overview of the CLEF 2004 multilingual question answering track. In *Multilingual Information Access for Text, Speech and Images, 5th Workshop of the Cross-Language Evaluation Forum, CLEF 2004, Bath, UK, September 15-17, 2004, Revised Selected Papers*, pages 371–391.

Metzler, D., Jones, R., Peng, F., and Zhang, R. (2009). Improving search relevance for implicitly temporal queries. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009*, pages 700–701.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 1003–1011.

Mitchell, T. M., Cohen, W. W., Jr., E. R. H., Talukdar, P. P., Betteridge, J., Carlson, A., Mishra, B. D., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohamed, T., Nakashole, N., Platanios, E. A., Ritter, A., Samadi, M., Settles, B., Wang, R. C., Wijaya, D., Gupta, A., Chen, X., Saparov, A., Greaves, M., and Welling, J. (2015). Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 2302–2310.

Mohammed, S., Shi, P., and Lin, J. (2018). Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 291–296.

Moschitti, A., Tymoshenko, K., Alexopoulos, P., Walker, A. D., Nicosia, M., Vetere, G., Faraotti, A., Monti, M., Pan, J. Z., Wu, H., and Zhao, Y. (2017). Question answering and knowledge graphs. In *Exploiting Linked Data and Knowledge Graphs in Large Organisations*, pages 181–212.

Pasca, M. and Harabagiu, S. M. (2001). High performance question/answering. In *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA*, pages 366–374.

Peñas, A., Unger, C., Paliouras, G., and Kakadiaris, I. A. (2015). Overview of the CLEF question answering track 2015. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 6th International Conference of the CLEF Association, CLEF 2015, Toulouse, France, September 8-11, 2015, Proceedings*, pages 539–544.

Petrov, S., Das, D., and McDonald, R. T. (2012). A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language*

Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012, pages 2089–2096.

Pound, J., Mika, P., and Zaragoza, H. (2010). Ad-hoc object retrieval in the web of data. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 771–780.

Prager, J. M., Chu-Carroll, J., and Czuba, K. (2004). Question answering using constraint satisfaction: Qa-by-dossier-with-contraints. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain.*, pages 574–581.

Pustejovsky, J., Knippen, R., Littman, J., and Saurí, R. (2005). Temporal and event information in natural language text. *Language Resources and Evaluation*, 39(2-3):123–164.

Radford, W., Carreras, X., and Henderson, J. (2015). Named entity recognition with document-specific KB tag gazetteers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 512–517.

Radlinski, F. and Joachims, T. (2005). Query chains: learning to rank from implicit feedback. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, pages 239–248.

Rajpurkar, P., Jia, R., and Liang, P. (2018). Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392.

Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning, CoNLL 2009, Boulder, Colorado, USA, June 4-5, 2009*, pages 147–155.

Ravichandran, D. and Hovy, E. H. (2002). Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 41–47.

Reddy, S., Lapata, M., and Steedman, M. (2014). Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Reddy, S., Täckström, O., Collins, M., Kwiatkowski, T., Das, D., Steedman, M., and Lapata, M. (2016). Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Ribarov, K. (2004). Review: Open-domain question answering from large text collections, by marius pasca. *Prague Bull. Math. Linguistics*, 81:65–68.

Salton, G., Allan, J., and Buckley, C. (1993). Approaches to passage retrieval in full text information systems. In *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Pittsburgh, PA, USA, June 27 - July 1, 1993*, pages 49–58.

Sang, E. F. T. K. (2002). Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning, CoNLL 2002, Held in cooperation with COLING 2002, Taipei, Taiwan, 2002*.

Sang, E. F. T. K. and Meulder, F. D. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 142–147.

Saquete, E., González, J. L. V., Martínez-Barco, P., Muñoz, R., and Llorens, H. (2009). Enhancing QA systems with complex temporal question processing capabilities. *J. Artif. Intell. Res.*, 35:775–811.

Saquete, E., Martínez-Barco, P., Muñoz, R., and González, J. L. V. (2004). Splitting complex temporal questions for question answering systems. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain.*, pages 566–573.

Savenkov, D. and Agichtein, E. (2016). When a knowledge base is not enough: Question answering over knowledge bases with external text data. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 235–244.

Sawant, U. and Chakrabarti, S. (2013). Learning joint query interpretation and response ranking. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 1099–1110.

Shekarpour, S., Marx, E., Ngomo, A. N., and Auer, S. (2015). SINA: semantic interpretation of user queries for question answering on interlinked data. *J. Web Sem.*, 30:39–51.

Smith, B. (2008). Ontology (science). In *Formal Ontology in Information Systems, Proceedings of the Fifth International Conference, FOIS 2008, Saarbrücken, Germany, October 31st - November 3rd, 2008*, pages 21–35.

Srihari, R. K. and Li, W. (1999). Information extraction supported question answering. In *Proceedings of The Eighth Text REtrieval Conference, TREC 1999, Gaithersburg, Maryland, USA, November 17-19, 1999*.

Steedman, M. (2000). The syntactic process.

Strötgen, J. and Gertz, M. (2010). Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval@ACL 2010, Uppsala University, Uppsala, Sweden, July 15-16, 2010*, pages 321–324.

Strötgen, J. and Gertz, M. (2016). *Domain-Sensitive Temporal Tagging*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Su, Y., Sun, H., Sadler, B., Srivatsa, M., Gur, I., Yan, Z., and Yan, X. (2016). On generating characteristic-rich question sets for QA evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 562–572.

Su, Y., Yang, S., Sun, H., Srivatsa, M., Kase, S., Vanni, M., and Yan, X. (2015). Exploiting relevance feedback in knowledge graph search. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 1135–1144.

Suchanek, F. M. (2008). Automated construction and growth of a large ontology. *PhDThesis. Saarbrücken University, Germany*, page 8.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706.

Sun, H., Dhingra, B., Zaheer, M., Mazaitis, K., Salakhutdinov, R., and Cohen, W. W. (2018). Open domain question answering using early fusion of knowledge bases and text. *EMNLP*.

Sun, H., Ma, H., Yih, W., Tsai, C., Liu, J., and Chang, M. (2015). Open domain question answering via semantic enrichment. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 1045–1055.

Talmor, A. and Berant, J. (2018). The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 641–651.

Tran, T., Cimiano, P., Rudolph, S., and Studer, R. (2007). Ontology-based interpretation of keywords for semantic search. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.*, pages 523–536.

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 *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, pages 405–416.

Trischler, A., Wang, T., Yuan, X., Harris, J., Sordoni, A., Bachman, P., and Suleman, K. (2017). Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017, Vancouver, Canada, August 3, 2017*, pages 191–200.

Trivedi, P., Maheshwari, G., Dubey, M., and Lehmann, J. (2017). Lc-quad: A corpus for complex question answering over knowledge graphs. In *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II*, pages 210–218.

Tsatsaronis, G., Schroeder, M., Paliouras, G., Almirantis, Y., Androutsopoulos, I., Gaussier, É., Gallinari, P., Artières, T., Alvers, M. R., Zschunke, M., and Ngomo, A. N. (2012). Bioasq: A challenge on large-scale biomedical semantic indexing and question answering. In *Information Retrieval and Knowledge Discovery in Biomedical Text, Papers from the 2012 AAAI Fall Symposium, Arlington, Virginia, USA, November 2-4, 2012*.

Unger, C., Bühmann, L., Lehmann, J., Ngomo, A. N., Gerber, D., and Cimiano, P. (2012). Template-based question answering over RDF data. In *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012*, pages 639–648.

Unger, C. and Cimiano, P. (2011). Pythia: Compositional meaning construction for ontology-based question answering on the semantic web. In *Natural Language Processing and Information Systems - 16th International Conference on Applications of Natural Language to Information Systems, NLDB 2011, Alicante, Spain, June 28-30, 2011. Proceedings*, pages 153–160.

Unger, C., Forascu, C., López, V., Ngomo, A. N., Cabrio, E., Cimiano, P., and Walter, S. (2015). Question answering over linked data (QALD-5). In *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015.*

Usbeck, R., Ngomo, A. N., Bühmann, L., and Unger, C. (2015). HAWK - hybrid question answering using linked data. In *The Semantic Web. Latest Advances and New Domains - 12th European Semantic Web Conference, ESWC 2015, Portoroz, Slovenia, May 31 - June 4, 2015. Proceedings*, pages 353–368.

Usbeck, R., Ngomo, A. N., Haarmann, B., Krithara, A., Röder, M., and Napolitano, G. (2017). 7th open challenge on question answering over linked data (QALD-7). In *Semantic Web Challenges - 4th SemWebEval Challenge at ESWC 2017, Portoroz, Slovenia, May 28 - June 1, 2017, Revised Selected Papers*, pages 59–69.

Voorhees, E. M. (2001). The philosophy of information retrieval evaluation. In *Evaluation of Cross-Language Information Retrieval Systems, Second Workshop of the Cross-Language Evaluation Forum, CLEF 2001, Darmstadt, Germany, September 3-4, 2001, Revised Papers*, pages 355–370.

Voorhees, E. M. (2010). Reflections on TREC QA. In *CLEF 2010 LABs and Workshops, Notebook Papers, 22-23 September 2010, Padua, Italy.*

Voorhees, E. M. and Tice, D. M. (2000). Building a question answering test collection. In *SIGIR 2000: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 24-28, 2000, Athens, Greece*, pages 200–207.

Vrandecic, D. and Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Walker, A. D., Alexopoulos, P., Starkey, A., Pan, J. Z., Gómez-Pérez, J. M., and Siddharthan, A. (2015). Answer type identification for question answering - supervised learning of dependency graph patterns from natural language questions. In *Semantic Technology - 5th Joint International Conference, JIST 2015, Yichang, China, November 11-13, 2015, Revised Selected Papers*, pages 235–251.

Wang, D. and Nyberg, E. (2015). A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 707–712.

Wang, K., Ming, Z., and Chua, T. (2009). A syntactic tree matching approach to finding similar questions in community-based qa services. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009*, pages 187–194.

Wang, M., Smith, N. A., and Mitamura, T. (2007). What is the jeopardy model? A quasi-synchronous grammar for QA. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pages 22–32.

Wang, S. I., Liang, P., and Manning, C. D. (2016). Learning language games through interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Wang, Y., Berant, J., and Liang, P. (2015). Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computa-*

tional Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers, pages 1332–1342.

Welbl, J., Stenetorp, P., and Riedel, S. (2018). Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

Werling, K., Chaganty, A. T., Liang, P., and Manning, C. D. (2015). On-the-job learning with bayesian decision theory. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3465–3473.

Wieting, J., Bansal, M., Gimpel, K., and Livescu, K. (2015). Towards universal paraphrastic sentence embeddings. *CoRR*, abs/1511.08198.

Woods, W. (1972). The lunar sciences natural language information system. *BBN report*.

Xu, J. and Croft, W. B. (1996). Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'96, August 18-22, 1996, Zurich, Switzerland (Special Issue of the SIGIR Forum)*, pages 4–11.

Xu, K., Feng, Y., Huang, S., and Zhao, D. (2016a). Hybrid question answering over knowledge base and free text. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2397–2407.

Xu, K., Reddy, S., Feng, Y., Huang, S., and Zhao, D. (2016b). Question answering on freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Yadav, V., Sharp, R., and Bethard, S. (2018). Deep affix features improve neural named entity recognizers. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics, *SEM@NAACL-HLT, New Orleans, Louisiana, USA, June 5-6, 2018*, pages 167–172.

Yahya, M., Barbosa, D., Berberich, K., Wang, Q., and Weikum, G. (2016). Relationship queries on extended knowledge graphs. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016*, pages 605–614.

Yahya, M., Berberich, K., Elbassuoni, S., Ramanath, M., Tresp, V., and Weikum, G. (2012). Natural language questions for the web of data. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 379–390.

Yahya, M., Berberich, K., Elbassuoni, S., and Weikum, G. (2013). Robust question answering over the web of linked data. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, pages 1107–1116.

Yang, L., Ai, Q., Guo, J., and Croft, W. B. (2016a). anmm: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 287–296.

Yang, M., Duan, N., Zhou, M., and Rim, H. (2014). Joint relational embeddings for knowledge-based question answering. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 645–650.

Yang, Y. and Chang, M. (2015). S-MART: novel tree-based structured learning algorithms applied to tweet entity linking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 504–513.

Yang, Y., Yih, W., and Meek, C. (2015). Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 2013–2018.

Yang, Z., Salakhutdinov, R., and Cohen, W. W. (2016b). Multi-task cross-lingual sequence tagging from scratch. *CoRR*, abs/1603.06270.

Yao, X. (2015). Lean question answering over freebase from scratch. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 66–70.

Yao, X. and Durme, B. V. (2014). Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 956–966.

Yih, W., Chang, M., He, X., and Gao, J. (2015). Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1321–1331.

Yin, P., Duan, N., Kao, B., Bao, J., and Zhou, M. (2015). Answering questions with complex semantic constraints on open knowledge bases. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pages 1301–1310.

Yosef, M. A., Bauer, S., Hoffart, J., Spaniol, M., and Weikum, G. (2012). HYENA: hierarchical type classification for entity names. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, 8-15 December 2012, Mumbai, India*, pages 1361–1370.

Zettlemoyer, L. S. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 658–666.

Zhang, K., Wu, W., Wang, F., Zhou, M., and Li, Z. (2016). Learning distributed representations of data in community question answering for question retrieval. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016*, pages 533–542.

Zhang, K., Wu, W., Wu, H., Li, Z., and Zhou, M. (2014). Question retrieval with high quality answers in community question answering. In *Proceedings*

of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014, pages 371–380.

Zhang, T. and Johnson, D. (2003). A robust risk minimization based named entity recognition system. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 204–207.

Zheng, W., Zou, L., Lian, X., Yu, J. X., Song, S., and Zhao, D. (2015). How to build templates for RDF question/answering: An uncertain graph similarity join approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 1809–1824.

Zhou, G., He, T., Zhao, J., and Hu, P. (2015). Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 250–259.

Zou, L., Huang, R., Wang, H., Yu, J. X., He, W., and Zhao, D. (2014). Natural language question answering over RDF: a graph data driven approach. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 313–324.