TEQUILA: Temporal Question Answering over Knowledge Bases

Zhen Jia
Southwest JiaoTong University, China
zjia@swjtu.edu.cn

Abdalghani Abujabal
MPI for Informatics, Germany
abujabal@mpi-inf.mpg.de

Rishiraj Saha Roy
MPI for Informatics, Germany
rishiraj@mpi-inf.mpg.de

Jannik Strötgen
Bosch Center for AI, Germany
jannik.stroetgen@de.bosch.com

Gerhard Weikum
MPI for Informatics, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

Question answering over knowledge bases (KB-QA) poses challenges in handling complex questions that need to be decomposed into sub-questions. An important case, addressed here, is that of temporal questions, where cues for temporal relations need to be discovered and handled. We present TEQUILA, an enabler method for temporal QA that can run on top of any KB-QA engine. TEQUILA has four stages. It detects if a question has temporal intent. It decomposes and rewrites the question into non-temporal sub-questions and temporal constraints. Answers to sub-questions are then retrieved from the underlying KB-QA engine. Finally, TEQUILA uses constraint reasoning on temporal intervals to compute final answers to the full question. Comparisons against state-of-the-art baselines show the viability of our method.

1 INTRODUCTION

Motivation and Problem. Knowledge-based question answering (KB-QA) aims to answer questions over large knowledge bases (e.g., DBpedia, Wikidata, Yago, 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 formal language like SPARQL or SQL, and execute the queries to retrieve answers from the KB. In doing so, KB-QA methods need to address the vocabulary mismatch between phrases in the input question and entities, types, 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 [9, 18]) 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 Rafaela Beckran, and David Beckham.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CIKM’18, October 2018, Turin, Italy
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-5996-7/18/11...
$15.00
https://doi.org/10.1145/3243734.3243837

An important case of complex questions are temporal information needs. Search often comes with explicit or implicit conditions about time [17]. 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.

Approach and Contributions. The key idea of this paper 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 [4], 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 (state-of-the-art systems QUINT [2] and AQQU [6] used in experiments), that are geared for answering simple questions.

To the best of our knowledge, this is the first paper that presents a complete pipeline specific to temporal KB-QA. Novel contributions also include: (i) a method for decomposing complex questions, and (ii) the time-constraint-based reasoning for combining sub-question results into overall answers. All data and code are

2 CONCEPTS

In NLP, the markup language TimeML (now, timeML.org) is widely used for annotating temporal information in text documents. Our definition of temporal questions is based on two of its concepts (tags for temporal expressions and temporal signals).

Temporal expressions. Timex3 tags demarcate four types of temporal expressions. Dates and times refer to points in time of different granularities (e.g., ‘May 1, 2010’ and ’9 pm’, respectively). They occur in fully- or under-specified forms (e.g., ‘May 1, 2010’ vs. ’last year’). Durations refer to intervals (e.g., ‘two years’), and sets to periodic events (e.g., ‘every Monday’). Going beyond TimeML, implicit expressions (e.g., ‘the Champions League final’) are used to capture events and their time scopes [14]. Expressions can be normalized into standard format (e.g., ‘May 2nd, 2016’ into 2016–05–02).

Temporal signals. SIGNAL tags mark textual elements that denote explicit temporal relations between two TimexML entities (i.e., events or temporal expressions), such as ’before’ or ’during’. We extend the TimeML definition to also include cues when an event is mentioned only implicitly, such as ’joining PSG’. In addition, we consider ordinals like ’first’, ’last’, etc. These are frequent in questions when entities can be chronologically ordered, such as ’last’ in ’Neymar’s last club before joining PSG’.

Temporal questions. Based on these considerations, we can now define a temporal question as any question that contains a temporal expression or a temporal signal, or whose answer type is temporal.

Temporal relations. Allen [3] introduced 13 temporal relations between time intervals for temporal reasoning: EQUAL, BEFORE, MEETS, OVERLAPS, DURING, STARTS, FINISHES, and their inverses for all but EQUAL. However, for an input temporal question, it is not always straightforward to infer the proper relation. For example, in Q3 the question should be BEFORE; but if we slightly vary

Q5: “Which team did Neymar play for before joining PSG?”

the singular form ‘team’ suggests that we are interested in the MEETS relation, that is, only the last team before the transfer. Frequent trigger words suggesting such relations are, for instance, the signals before, prior to (for BEFORE or MEETS), after, following (for AFTER), and during, while, when (for OVERLAP).

3 METHOD

Given an input question, TEQUILA works in four stages: (i) detect if the question is temporal, (ii) decompose the question into simpler sub-questions with some form of rewriting, (iii) obtain candidate answers and dates for temporal constraints from a KB-QA system, and (iv) apply constraint-based reasoning on the candidates to produce final answers. Our method builds on ideas from the literature on question decomposition for general QA [2, 5, 20]. Standard NLP tasks like POS tagging, NER, and coreference resolution, are performed on the input question before passing it on to TEQUILA.

3.1 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

<table>
<thead>
<tr>
<th>Case</th>
<th>Expected input: who</th>
<th>SIGNAL</th>
<th>who?</th>
<th>wh?</th>
<th>w?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1:</td>
<td>Constraint has both an entity and a relation</td>
<td>who</td>
<td>who</td>
<td>SIGNAL</td>
<td>who?</td>
</tr>
<tr>
<td>Sub-question 1 pattern:</td>
<td>who</td>
<td>wh?</td>
<td>w?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub-question 2 pattern:</td>
<td>when</td>
<td>who</td>
<td>SIGNAL</td>
<td>who?</td>
<td>wh?</td>
</tr>
</tbody>
</table>
| E.g.: | "when did neymar play before he joined barcelona?" | "where did neymar play?" | "when neymar joined barcelona?"
| Case 2: | Constraint has no entity but a relation | who | SIGNAL | who? | wh? | w? |
| Sub-question 1 pattern: | who | wh? | w? |
| Sub-question 2 pattern: | when | who | SIGNAL | who? | wh? | w? |
| E.g.: | "where did neymar live before playing for clubs?" | "when did neymar live?"
| Case 3: | Constraint has no relation but an entity | SIGNAL | who | SIGNAL | who? | wh? | w? |
| Sub-question 1 pattern: | SIGNAL | who | SIGNAL | SIGNAL | who? | wh? | w? |
| Sub-question 2 pattern: | when | SIGNAL | SIGNAL | SIGNAL | SIGNAL | SIGNAL | SIGNAL |
| E.g.: | "when neymar was the brazil team captain?" | "when neymar was the brazil team captain?"
| Case 4: | Constraint is an event name | SIGNAL | who | SIGNAL | who? | wh? | w? |
| Sub-question 1 pattern: | SIGNAL | who | SIGNAL | SIGNAL | SIGNAL | SIGNAL | SIGNAL |
| Sub-question 2 pattern: | when | SIGNAL | SIGNAL | SIGNAL | SIGNAL | SIGNAL | SIGNAL |
| E.g.: | "when did neymar play during south africa world cup?" | "when did neymar play during south africa world cup?"
| Sub-question 1: | "when did neymar play?" | "when did neymar play?"
| Sub-question 2: | "when south africa world cup happen?"

Table 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? are tokens in the question.

is temporal (e.g., the question starts with ’When’). We use Heidel-Time [23] to tag TIMEX3 expressions in questions. Named events 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 as per suggestions from Setzer [22], 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.

3.2 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 sub-questions are combined by intersecting their answers. The constraints are applied to time scopes associated with results of the non-temporal sub-questions. For brevity, the following explanation focuses on the case with one non-temporal sub-question, and one temporal sub-question. We use a set of lexico-syntactic rules (Table 1) designed from first principles to decompose and rewrite a question into its components. Basic intuitions driving these rules are as follows:

- The signal word separates the non-temporal and temporal sub-questions, acting as a pivot for decomposition;
- Each sub-question needs to have an entity and a relation (generally represented using verbs) 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;
With multiple sub-questions, the candidate sets are intersected. The time scopes, so that we can evaluate the temporal constraints. TEQUILA: Temporal Question Answering over Knowledge Bases CIKM’18, October 2018, Turin, Italy

vals with start point vals, or sets of dates (e.g., a set of consecutive years during which the non-temporal candidate answers, also cast into intervals, are retrieved by looking up marriage.date, marriage.spouse, etc.). The similarity measure is implemented by selecting tokens in predicate names (footballPlayer, team, etc.), contextualizing the tokens by computing word2vec embeddings for them, averaging per-token vectors to get a resultant vector for each predicate [25], and comparing the cosine distance between two predicate vectors. 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).

### 3.3 Answering sub-questions

Sub-questions are passed on to the underlying 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 candidate answers for the full question. For example, the Freebase predicate marriage.marriage.spouse and marriage.date. When the predicate marriage.marriage.spouse is used to retrieve answers, the time scope is retrieved by looking up marriage.date in the KB. On the other hand, 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 for the candidate entity, and chooses one based on a similarity measure between the non-temporal predicate (team.players) and potentially relevant temporal predicates (footballPlayer.team.joinedOnDate, footballPlayer.team.leftOnDate). The similarity measure is implemented by selecting tokens in predicate names (footballPlayer, team, etc.), contextualizing the tokens by computing word2vec embeddings for them, averaging per-token vectors to get a resultant vector for each predicate [25], and comparing the cosine distance between two predicate vectors. 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 leftOnDate).

### 3.4 Reasoning on temporal intervals

For temporal sub-questions, the results are time points, time intervals, or sets of dates (e.g., a set of consecutive years during which someone played for a football team). We cast all these into intervals with start point begincons and end point endcons. These form the temporal constraints against which we test the time scopes of the non-temporal candidate answers, also cast into intervals [begincons, endcons]. The test itself depends on the temporal operator derived from the input question (e.g., BEFORE, OVERLAP, etc.)

### 4 EXPERIMENTS

#### 4.1 Setup

We evaluate TEQUILA on the TempQuestions benchmark [13], which contains 1,271 temporal questions labeled as questions with explicit, implicit, and ordinal constraints, and those with temporal answers. Questions are paired with their answers over Freebase. We use three state-of-the-art KB-QA systems as baselines: AQQU [6], QUINT [2] (code from authors for both), and Bao et al. [4] (detailed results from authors). The first two are geared for simple questions, while Bao et al. handle complex questions, including temporal ones. We use TEQUILA as a plug-in for the first two, and directly evaluate against the system of Bao et al. on 341 temporal questions from the ComplexQuestions test set [4]. 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 w.r.t. the gold answer sets, and average them over all test questions.

#### 4.2 Results and insights

Results on TempQuestions and the 341 temporal questions in ComplexQuestions are shown in Table 3. AQQU + TEQUILA and QUINT + TEQUILA refer to the TEQUILA-enabled versions of the respective baseline systems. We make the following observations.

**TEQUILA enables KB-QA systems to answer composite questions with temporal conditions.** Overall and category-wise F1-scores show that TEQUILA-enabled systems significantly outperform the baselines. 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.

**TEQUILA outperforms state-of-the-art baselines.** Bao et al. [4] represents the state-of-the-art in KB-QA, with a generic mechanism for handling constraints in questions. TEQUILA-enabled systems outperform Bao et al. on the temporal slice of ComplexQuestions, showing that a tailored method for temporal information needs is worthwhile. TEQUILA enabled QUINT and AQQU to answer questions like: "who is the first husband of julia roberts?", "when did francesco sabatini start working on the puerta de san vicente?", and "who was governor of oregon when shanghai noon was released?".

**Error analysis.** Analyzing cases when TEQUILA fails yields insights towards future work: (i) Decomposition and rewriting were incorrect (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 1); (ii) The correct temporal predicate was not found due to limitations of the similarity function; and (iii) The temporal constraint or the time scope to use during reasoning was wrongly identified.

### 5 RELATED WORK

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 [10], explores multi-faceted passage answers, but information needs are still simple. In IBM Watson [12], structured data played a role, but text was the main source for answers. Question decomposition was leveraged, for example, in [12, 20, 29] for QA over text. However, re-composition and reasoning over answers works very differently for textual sources [20], and are not directly applicable for KB-QA. Compositional semantics of natural language sentences has been addressed by [16] 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 [7, 8, 11, 24, 27]. Most methods have focused on simple questions, whose SPARQL translations contain only a single variable (and a few triple patterns for a single set of qualifying entities). For popular benchmarks like WebQuestions [7], the best performing systems use templates and grammars [1, 2, 6, 19, 29], leverage additional text [21, 26], or learn end-to-end with extensive training data [15, 26, 28]. These methods do not cope well with complex questions. Bao et al. [4] combined rules with deep learning to address a variety of complex questions.

### 6 Conclusion

Understanding the compositional semantics of complex questions is an open challenge in QA. We focused on temporal question answering over KBs, as a major step for coping with an important slice of information needs. Our method showed boosted performance on a recent benchmark, and outperformed a state-of-the-art baseline on general complex questions. Our work underlines the value of building reusable modules that improve several KB-QA systems.

### References


