LDK 2023

LexExMachinaQA: A framework for the automatic induction of ontology lexica for Question Answering over Linked Data

Mohammad Fazleh Elahi
Bielefeld University, Germany
melahi@techfak.uni-bielefeld.de

Basil Ell
Bielefeld University, Germany
Oslo University, Norway
basile@ifi.uio.no

Philipp Cimiano
Bielefeld University, Germany
cimiano@techfak.uni-bielefeld.de

Abstract

An open issue for Semantic Question Answering Systems is bridging the so called *lexical gap*, referring to the fact that the vocabulary used by users in framing a question needs to be interpreted with respect to the logical vocabulary used in the data model of a given knowledge base or knowledge graph. Building on previous work to automatically induce ontology lexica from language corpora by using association rules to identify correspondences between lexical elements on the one hand and ontological vocabulary elements on the other, in this paper we propose LexExMachinaQA, a framework allowing us to evaluate the impact of automatically induced lexicalizations in terms of alleviating the lexical gap in QA systems. Our framework combines the LexExMachina approach (Ell et al., 2021) for lexicon induction with the QueGG system proposed by Benz et al. (Benz et al., 2020) that relies on grammars automatically generated from ontology lexica to parse questions into SPARQL. We show that automatically induced lexica yield a decent performance i.t.o. F_1 measure with respect to the QLAD-7 dataset, representing a 34% – 56% performance degradation with respect to a manually created lexicon. While these results show that the fully automatic creation of lexica for QA systems is not yet feasible, the method could certainly be used to bootstrap the creation of a lexicon in a semi-automatic manner, thus having the potential to significantly reduce the human effort involved.

1 Introduction

According to (Höffner et al., 2017), the benefit of Semantic Question Answering (SQA) systems from the perspective of end users is that they can access knowledge in knowledge bases or knowledge graphs i) without having to master a formal language such as SPARQL, and ii) without having knowledge about the (ontological) vocabularies used in the knowledge bases. One of the seven challenges identified by the authors for the development of SQA systems is handling the lexical gap, requiring to bridge between the way users refer to certain properties and the way they are modelled in a given knowledge base. Take the following examples involving a (relational) noun, a verb, and an adjective, respectively:

- *'Who is the husband of Julia Roberts?'* In this case, *'husband'* needs to be interpreted with respect to DBpedia as `dbo:spouse`[1] in order to map the question correctly to the following SPARQL query:

```
SELECT ?o WHERE {
dbr:Julia_Roberts dbo:spouse ?o }
```

- *'Who stars in the Matrix?'* In this case, *'stars in'* refers to the property `dbo:actor`, so that the question can be mapped to the following SPARQL query:

```
SELECT ?o WHERE {
dbr:The_Matrix dbo:actor ?o }
```

- *'How high is the Mulhacén?'* In this case, *'high'* needs to be interpreted in terms of the

[1] In this paper we use compact URIs and use namespace prefixes that are defined as follows: `dbr`: http://dbpedia.org/resource/, `dbo`: http://dbpedia.org/ontology/, `dbp`: http://dbpedia.org/property/, `rdf`: http://www.w3.org/1999/02/22-rdf-syntax-ns#, `lemon`: http://lemon-model.net/lemon#, and `lexinfo`: http://www.lexinfo.net/ontology/2.0/lexinfo#.

DBpedia property `dbp:elevation` in order to map the question correctly to the following SPARQL query:

```
SELECT ?o WHERE {
dbr:Mulhacén dbp:elevation ?o }
```

Existing QA systems have attempted to handle the lexical gap by using edit distances or similarity measures to recognize inflected forms of the same lemma and dealing with misspellings or spelling variants (Höffner et al., 2017). A frequently used lexical resource is WordNet (Miller, 1995) and has been used to recognize synonyms in QA systems (e.g., (Walter et al., 2012)). Some QA systems have also relied on pattern databases such as PATTY (Nakashole et al., 2012) to find constructions that verbalize a given relation or property. Word embeddings have also been used to discover related terms (Hakimov et al., 2017).

In this paper, building on our previous work (Benz et al., 2020; Elahi et al., 2021), we follow a different approach and induce a lexicon that is specific for a given knowledge base or vocabulary. We have shown that such lexica can be induced automatically to some extent using our LexExMachina approach (Ell et al., 2021) that builds on association rules to find correspondences between lexical elements and ontological vocabulary elements. However, it is unclear if this approach would help to effectively bridge the lexical gap prevailing in QA systems. In this paper, we thus leverage the LexExMachina approach to induce lexical knowledge relevant for QA, so that we call the approach *LexExMachinaQA*. In order to evaluate the impact of the automatically induced lexica, we build on the QA system proposed by Benz et al. (Benz et al., 2020) that relies on a lexicon-ontology model to automatically generate a grammar that allows to parse questions into SPARQL. While the approach in principle works for multiple languages, in this paper we restrict the evaluation to the English language as a proof-of-concept. Our evaluation is conducted with respect to QALD-7 as a benchmark. We contrast the results obtained with an automatically induced lexicon with the results of a lexicon created manually, comprising 806 lexical entries overall. We show that the automatically induced lexicon yields a decent performance of F_1 for the QA system proposed by Benz et al. (Benz et al., 2020) on the QALD-7 benchmark, corresponding to a performance degradation of between 34–56% relative to the performance of a QA system based on the manually created lexicon. While this shows that it is still worth to invest into manual lexicon creation, the results are encouraging in the sense that the automatically induced lexicon could reduce significantly the human effort involved.

2 Method

In this section, first, we briefly describe our model-based approach to Question Answering (QueGG), detailed in previous work (Benz et al., 2020). QueGG makes use of an ontology lexicon to generate grammars from which questions in natural language are generated. Second, we describe how a lexicon can be created manually. Third, we briefly describe LexExMachina, our previous work on inducing correspondences between natural language and a knowledge base using association rule mining (Ell et al., 2021). Finally, we describe how we make use of the correspondences obtained via LexExMachina to automatically derive a lexicon that can then be used by QueGG.

2.1 Background: QueGG

QueGG (Benz et al., 2020), our previous work, is a model-based approach to QA in which a developer of the QA system provides a lexicon using the lemon-OntoLex model (Cimiano et al., 2016), specifying how the vocabulary elements are realized in natural language. The lemon-OntoLex model is an updated version of the lemon model (McCrae et al., 2011) and is the core representation used by the grammar generation in QueGG. The main benefit of the approach is that it is fully controllable in the sense that it can be predicted what the impact of extending the lexicon will have in terms of the questions covered by the system.

Our previous work on QueGG has shown that, leveraging on lemon lexica, question answering grammars can be automatically generated, and these can, in turn, be used to interpret questions and parse them into SPARQL queries. A QA web application developed in previous work (Elahi et al., 2021; Nolano et al., 2022) has further shown that such QA systems can scale to millions of questions and that the performance of the system is practically real-time from an end-user perspective.

The grammar generation from a lexical entry with a specific syntactic frame, detailed in LexInfo (Cimiano et al., 2011), is controlled by a

generic template that describes how specific lexicalized grammar rules can be generated for a given lexical entry. The grammar generation supports the following syntactic frames:

- `NounPPFrame`: corresponding to a relational noun that requires a prepositional object such as *'spouse' (of)*, *'mayor' (of)*, *'capital' (of)*
- `TransitiveFrame`: correspoding to transitive verbs such as *(to) 'direct'* and *(to) 'marry'*.
- `InTransitivePPFrame`: corresponding to intransitive verbs subcategorizing a prepositional phrase such as *'star' (in)*, *'born' (on)* or *'flow' (through)*
- `AdjectivePredicateFrame`: covering intersective adjectives such as *'Spanish'* and *'Afghan'*. This frame is used for both attributive and predicative use of the adjective.
- `AdjectiveSuperlativeFrame`: covering gradable adjectives such as *'high'* and *'highest'*.

For the sake of self-containedness, we describe a lexical entry and the grammar rules for the transitive verb *(to) 'direct'*. The lexicon entry is shown in Figure 1. The semantics of the lexical entry *(to) 'direct'* is expressed by the property `dbo:director`. The lemon entry also specifies that the subject of the property is realized by the direct object of the verb *'direct'*, while the object of the property is realized by the syntactic subject of the verb *'direct'*. The following grammar is generated automatically:

```
Rule 1:
S -> Who directs X? | Who directed X? |
     Which person directs X? | Which
     person directed X?
Rule 2:
S -> What is directed by X? | What was
     directed by X? | Which film is
     directed by X? | Which films are
     directed by X? | Which film was
     directed by X? | Which  films were
     directed by X? | Give me all films
     directed by X?
Rule 3:
S -> How many films are directed by X? |
     How often did X direct?
Rule 4:
S -> film directed by X | films directed
     by X
```

```
:to_direct a lemon:LexicalEntry ;
  lexinfo:partOfSpeech lexinfo:verb ;
  lemon:canonicalForm :form_direct ;
  lemon:otherForm :form_directs ;
  lemon:otherForm :form_directed ;
  lemon:synBehavior
 :direct_frame_transitive ;
  lemon:sense :direct_ontomap .

:form_direct a lemon:Form ;
  lemon:writtenRep "direct"@en ;
  lexinfo:verbFormMood lexinfo:infinitive .

:form_directs a lemon:Form ;
  lemon:writtenRep "directs"@en ;
  lexinfo:person lexinfo:thirdPerson .

:form_directed a lemon:Form ;
  lemon:writtenRep "directed"@en ;
  lexinfo:tense lexinfo:past .

:direct_frame_transitive a
      lexinfo:TransitiveFrame ;
  lexinfo:subject :direct_subj ;
  lexinfo:directObject :direct_obj .

:direct_ontomap a lemon:OntoMap,
      lemon:LexicalSense ;
  lemon:ontoMapping :direct_ontomap ;
  lemon:reference dbo:director ;
  lemon:subjOfProp :direct_obj ;
  lemon:objOfProp :direct_subj ;
  lemon:condition :direct_condition .

:direct_condition a lemon:condition ;
  lemon:propertyDomain dbo:Film ;
  lemon:propertyRange dbo:Person .
```

Figure 1: Lemon entry for the transitive verb *(to) 'direct'*.

2.2 Background: Manual Lexicon Creation

A necessary prerequisite for the grammar generation approach is the availability of a lemon lexicon that describes by which lexical entries the elements (classes, properties) of a particular dataset can be verbalized in a particular language. In particular, a lexicon is needed for each language to be supported by the QA system. We manually created a lexicon for English and DBpedia.[2] The manually created lexical entries,[3] together with the automatically generated grammar, are available online.[4] Table 1 shows the number of manually created lexical entries for QALD-7 training data for different frame types of LexInfo as well as the number of grammar rules automatically generated from these.

The creation of a single lexical entry took approximately 2–3 minutes. The total construction time for the lexicon comprising of 806 entities was approximately 30 hours.

[2] https://downloads.dbpedia.org/2016-10/core-i18n/en/

[3] https://github.com/fazleh2010/multilingual-grammar-generator/tree/main/result/en/lexicalEntries

[4] https://github.com/fazleh2010/multilingual-grammar-generator/tree/main/result/en/grammar

Frame Type	# Lexical Entries	# Grammar Rules
NounPP	722	1,444
Transitive	37	111
InTransitivePP	27	81
AdjPredicate	15	76
AdjSuperlative	5	15
Total	806	1,727

Table 1: An overview over the number of manually created lexical entries for QALD-7 training data for different frame types and the number of automatically generated grammar rules.

2.3 Background: LexExMachina

LexExMachina (Ell et al., 2021) is a methodology that induces correspondences between natural language and a knowledge base by mining class-specific association rules from a loosely-parallel text-data corpus (e.g., Wikipedia + DBpedia). These association rules can help to bridge from natural language to a knowledge base and from a knowledge base to natural language. In the context of question answering, we make use of those rules that bridge from natural language to a knowledge base.

For example, in the context of a question about a person where the question contains the adjective "Greek", the corresponding SPARQL query would contain a triple pattern such as `?x dbo:nationality dbr:Greece`, whereas in the context of a question about a settlement where the question contains the adjective "Greek", the corresponding SPARQL query would contain a triple pattern such as `?x dbo:country dbr:Greece`.

The association rule that specifies that if the term "Greek" occurs in a text about a politician, then this corresponds in DBpedia to the triple pattern with predicate `dbo:nationality` and object `dbr:Greece` is represented as follows:

$$dbo{:}Politician \in c_e \wedge "Greek" \in l_e \Rightarrow (e, dbo{:}nationality, dbr{:}Greece) \in G$$

Here, c_e is the set of classes an entity e is instance of, l_e is a set of linguistic patterns (such as n-grams) that occur in the text that mentions the entity e, and G is the knowledge base that we bridge to (here: DBpedia). This rule is an example for the rule pattern $c_s, l_s \Rightarrow po$, one of the 20 types of association rules regarded by LexExMachina. In particular, the rule expresses that for an entity e that is an instance of the class `dbo:Politician` where the linguistic pattern "Greek" occurs in the text that mentions or describes the entity e, within the knowledge graph G there is (or should be) a triple that expresses that the entity e is in relation `dbo:nationality` with the entity `dbo:Greece`.

The LexExMachina approach was previously applied to a subset of a loosely-parallel text-data corpus consisting of Wikipedia as a corpus and DBpedia as a knowledge graph, which resulted in $447,888,109$ rules, published together with the original paper.

Association rules come with a set of measures. The general form of an association rule is $A \Rightarrow B$. For the types of rules that we regard in this paper, with $sup(A)$ we refer to the number of times the event described by the left hand side of an association rule occurred in the corpus (e.g., how often it occurred in the corpus that a text that mentioned or described a politician contained "Greek"). With $sup(B)$ we refer to the number of times the event described by the right hand side of an association rule occurred in the knowledge graph (e.g., how often it occurred in the knowledge graph that an entity is in relation `dbo:nationality` with the entity `dbo:Greece`). $sup(AB)$ refers to the number of times that both events occurred together (e.g., how often it occurred that a text that mentioned or described an entity of type politician contained "Greek" and this entity is in relation `dbo:nationality` with the entity `dbo:Greece` in the knowledge graph). The confidence of an association rule of the form $A \Rightarrow B$, denoted by $conf(A \Rightarrow B)$, is the estimated conditional probability $P(B|A)$ and is calculated as $sup(AB)/sup(A)$.

In practice, association rules with high confidence do not necessarily disclose truly interesting event relationships (Brin et al., 1997). Therefore, an *interestingness measure* quantifies the interestingness of an association rule. For example, the interestingness measure $Cosine(A \Rightarrow B)$ is defined as $\sqrt{P(A|B)P(B|A)}$. Note that $P(A|B)$ is equal to $conf(B \Rightarrow A)$, i.e., the confidence of the "reversed" rule.

2.4 Lexicon Generation based on LexExMachina

The starting point for our lexicon induction method is a knowledge graph. We retrieve all the prop-

erty URIs from the graph and mine class-specific association rules for each property, yielding lexicalizations for each property.

While LexExMachina defines 20 different types of class-specific association rules, in the context of LexExMachinaQA we rely only on two of those. In fact, we rely only on the two rules that predict a lexicalization for a subject of a given class and a property or for an object of a given class and a property. These rules are described in more detail in the following:

1. The rule pattern with the name $c_s, p \Rightarrow l_s$ has the following meaning: given a subject entity e that is an instance of the class c_s and given that e is in relation p to some term, then the relation can be expressed with the linguistic pattern l. The LexExMachina dataset contains $98,317,655$ rules of this type.

$$\begin{aligned} &dbo{:}FictionalCharacter \in c_e \\ &\wedge \exists o : (e, dbo{:}spouse, o) \in G \\ &\Rightarrow \text{"husband of"} \in l_e \end{aligned} \tag{1}$$

2. The rule pattern with the name $c_o, p \Rightarrow l_o$ has the following meaning: given an object entity e that is an instance of the class c_o and given that some term is in relation p with e, then the relation can be expressed with the linguistic pattern l. The LexExMachina dataset contains $6,499,288$ rules of this type.

$$\begin{aligned} &dbo{:}Person \in c_e \\ &\wedge \exists s : (s, dbo{:}starring, e) \in G \\ &\Rightarrow \text{"star in"} \in l_e \end{aligned} \tag{2}$$

The linguistic patterns found on the right-hand side of the above rules are n-grams found in the corresponding texts. In LexExMachina, n-grams with $1 \leq n \leq 4$ are considered.

Given an association rule, the creation of a lexical entry comprises the following steps:

1. We remove stop words (excluding prepositions) from the linguistic patterns on the right hand sides of the rules.

2. We use a part-of-speech tagger to tag the n-grams on the right-hand side of a rule. We rely on the Stanford tagger in particular.[5]

3. Relying on the part-of-speech sequence, patterns are classified into the syntactic frames discussed in Section 2.1. A noun followed by a preposition is classified as a `NounPPFrame`. A verb is either classified as a transitive verb (i.e., `TransitiveFrame`) or as an intransitive verb (i.e., `InTransitivePPFrame`), based on the English Wiktionary dictionary.[6] Wiktionary also contains inflection forms of verbs, which are added to a lexical entry – see for example Figure 1 line 14 "directs" and line 18 "directed" in the entry for the transitive verb *(to) 'direct'*. An adjective is classified as an attributive adjective (i.e., `AdjectivePredicativeFrame`) or as an superlative adjective (i.e., `AdjectiveSuperlativeFrame`). We use Wiktionary for an adjective's classification and retrieve its inflection forms.

We describe how the actual lexical entries in RDF format are created by way of OTTR templates (Skjæveland et al., 2018). OTTR is a language for defining templates over RDF data. Thereby, consistency can be ensured and RDF graph instantiations are more human-readable than plain RDF data. Using OTTR enables us to separate the data about a lexical entry that we collect from LexExMachina and from Wiktionary from how we represent it. For example, in order to create the lemon entry for the relational noun *'husband' (of)*, shown in Figure 2, we need to have collected the canonical, singular and plural form of the noun, the preposition, the corresponding DBpedia property, and the property's domain and range. Then, when the OTTR template shown in the appendix in Figure 3 is instantiated using the OTTR template instantiation statement shown below, then RDF data similar[7] to the data shown in Figure 2 is generated.

```
quegg:NounPPFrame(
    "husband"@en, "husband"@en,
    "husbands"@en, "of"@en,
    dbo:husband, dbo:Person,
    dbo:Person).
```

[5] https://nlp.stanford.edu/software/tagger.shtml

[6] http://en.wiktionary.org/

[7] Instead of showing the actual RDF data as it is generated, which contains blank nodes such as `_:b0`, `_:b1` etc., for the purpose of readability we have replaced these with meaningful URIs.

Lexicon	# Entries NounPP*	# Entries Transitive*	# Entries InTransitivePP*	# Entries AdjPred*+AdjSuper*	# Entries Total
Rule Pattern $c_s, p \Rightarrow l_s$					
s-L1	280,219	27,703	26,072	34,175	368,169
s-L2	286,127	28,246	26,724	34,818	375,915
s-L3	572,254	56,492	53,448	69,636	751,830
s-L4	248,963	24,954	23,825	31,067	328,809
s-L5	497,926	49,908	47,650	62,134	657,618
Rule Pattern $c_o, p \Rightarrow l_o$					
o-L1	66,454	4,598	4,422	8,618	84,092
o-L2	42,416	4,701	3,908	7,203	58,228
o-L3	57,713	3,626	3,437	6,636	71,412
o-L4	43,654	2,644	2,597	4,739	53,634
o-L5	38,712	2,742	1,092	4,798	47,344

Table 2: The table shows the number of lexical entries per frame type generated with the two rule patterns for the best 5 lexicon configurations according to F-score. Here, AdjPred* refers to `AdjectivePredicateFrame` and AdjSuper* refers to `AdjectiveSuperlativeFrame`.

Lexicon	$sup(A)$	$sup(B)$	$sup(AB)$	$P(A\|B)$	$P(B\|A)$	$Cos.$	micro-P	Micro-R	Micro-F_1	Macro-P	Macro-R	Macro-F_1
Rule Pattern $c_s, p \Rightarrow l_s$												
s-L1	5	5	5	0.02	0.09	0.1	0.32	0.44	0.37	0.40	0.40	0.40
s-L2	5	5	5	0.02	0.02	0.1	0.32	0.44	0.37	0.40	0.40	0.40
s-L3	250	50	50	0.02	0.10	0.1	0.31	0.47	0.37	0.39	0.39	0.38
s-L4	250	5	5	0.02	0.10	0.1	0.30	0.46	0.36	0.38	0.39	0.39
s-L5	250	5	5	0.02	0.60	0.1	0.26	0.46	0.33	0.38	0.37	0.38
Rule Pattern $c_o, p \Rightarrow l_o$												
o-L1	5	5	5	0.09	0.02	0.1	0.15	0.36	0.21	0.22	0.23	0.27
o-L2	5	5	5	0.02	0.02	0.09	0.14	0.41	0.21	0.24	0.24	0.24
o-L3	5	5	5	0.1	0.02	0.09	0.14	0.37	0.21	0.23	0.23	0.23
o-L4	5	5	5	0.02	0.1	0.1	0.13	0.40	0.20	0.24	0.24	0.24
o-L5	5	5	5	0.02	0.1	0.09	0.13	0.40	0.20	0.23	0.23	0.23

Table 3: The table shows the configurations as well as micro-averaged and macro-averaged precision, recall, and F_1 scores for the 5 best lexicon configurations according to F-measure with respect to QALD-7 training data.

3 Evaluation

In this section we describe how we evaluate the manually created and the automatically generated ontology lexica and describe how we have optimized threshold values based on the parameters of LexExMachina rules to yield the best settings for LexExMachinaQA. We compare the results of the automatically generated lexica to the results obtained using the manually created lexicon as an upper baseline.

3.1 Lexicon Evaluation

We evaluate each lexicon using the QALD-7 benchmark (Usbeck et al., 2017). A QALD dataset consists of a set of tuples of the form (q, s) where q is a question in natural language and s is a corresponding SPARQL query that retrieves the answers to q from a knowledge graph (here: DBpedia).

An example (q, s) pair is the following: (*'Who was the wife of U.S. president Lincoln?'*, `SELECT ?o WHERE { dbr:Abraham_Lincoln dbo:spouse ?o }`).

Given a lexicon, our approach generates grammars from which questions are generated – we call these QueGG questions. These questions have corresponding queries. Thus, we generate a set of (question, query) tuples.

We evaluate the QueGG answers for each QALD question using *Precision* (Eq. 3), *Recall* (Eq. 4) and *F-Measure* as defined by the QALD task (Usbeck et al., 2017).

Given a question-query pair (q, s) from QALD, we find the question-query pair (q', s') from QueGG such that the similarity between the questions q and q' is maximal. We use Jaccard similarity to measure the similarity between two questions:

$$(q', s') = \max_{(q', s') \in QueGG} JS(q, q')$$

The reason for using the Jaccard similarity measure is because it ignores word order and duplicate words, thus it emphasizes unique words shared by two questions. For example, for the QALD-7 question *'When was the Titanic completed?'* we retrieve the QueGG question *'When was RMS Ti-*

```
:husband_of a lemon:LexicalEntry ;
  lexinfo:partOfSpeech lexinfo:noun ;
  lemon:canonicalForm :husband_of_form ;
  lemon:otherForm :husband_of_singular ;
  lemon:otherForm :husband_of_plural ;
  lemon:sense :husband_of_sense_1 ;
  lemon:synBehavior :husband_of_nounpp .

:husband_of_form a lemon:Form ;
  lemon:writtenRep "husband"@en .

:husband_of_singular a lemon:Form ;
  lemon:writtenRep "husband"@en ;
  lexinfo:number lexinfo:singular .

:husband_of_plural a lemon:Form ;
  lemon:writtenRep "husbands"@en ;
  lexinfo:number lexinfo:plural .

:husband_of_nounpp a lexinfo:NounPPFrame ;
  lexinfo:copulativeArg :arg1 ;
  lexinfo:prepositionalAdjunct :arg2 .

:husband_of_sense_1 a lemon:OntoMap,
lemon:LexicalSense ;
  lemon:ontoMapping :husband_of_sense_1 ;
  lemon:reference dbo:spouse ;
  lemon:subjOfProp :arg2 ;
  lemon:objOfProp :arg1 ;
  lemon:condition :husband_of_sense_1_condition .

:husband_of_sense_1_condition a lemon:condition ;
  lemon:propertyDomain dbo:Person ;
  lemon:propertyRange dbo:Person .

:arg2 lemon:marker :husband_of_form_preposition .
## Prepositions ##
:husband_of_form_preposition a
lemon:SynRoleMarker ;
  lemon:canonicalForm
[ lemon:writtenRep "of"@en ] ;
  lexinfo:partOfSpeech lexinfo:preposition .
```

Figure 2: Lemon entry for the relational noun *'husband' (of)*.

tanic completed on?' as most similar question. As a downside, the use of the Jaccard measure introduces some unwanted artefacts as it is possible that truly different questions can have a similarity of 1 , e.g. *'Who is the daughter of the daughter of Jan Delay?'* gets 100% similarity with the question *'Who is the daughter of Jan Delay?'*.

$$precision(q,s) := \frac{|\Omega_{s,G} \cap \Omega_{s',G}|}{|\Omega_{s',G}|} \quad (3)$$

$$recall(q,s) := \frac{|\Omega_{s,G} \cap \Omega_{s',G}|}{|\Omega_{s,G}|} \quad (4)$$

3.2 Parameter Optimization

The rules created by LexExMachina have a number of parameters (see Section 2.3). We make use of these parameters to specify which rules to use based on threshold values when creating a lexicon. We carry out grid search to find the best values (according to F_1-measure) for these parameters on the QALD-7 training dataset.

The threshold parameters that we optimize and the grid intervals we explore are the following:

$$\begin{aligned} sup(A) &\in \{5, 50, 250\} \\ sup(B) &\in \{5, 50, 250\} \\ sup(AB) &\in \{5, 50, 250\} \\ P(B|A) &\in \{0.02, 0.09, 0.1, 0.6\} \\ P(A|B) &\in \{0.02, 0.09, 0.1, 0.6\} \\ Cosine(A \Rightarrow B) &\in \{0.02, 0.09, 0.1, 0.6\} \end{aligned}$$

In principle, this yields $3^3 \times 3^4 = 1728$ configurations to explore. However, there cannot be a rule where $sup(A)$ or $sup(B)$ is smaller than $sup(AB)$. For two configurations that only differ in, e.g., the $sup(A)$ threshold and both $sup(A)$ values are less or equal to $sup(AB)$, both configurations would yield the same lexicon. Thus, we exclude configurations where either $sup(A)$ or $sup(B)$ is set to a value lower than $sup(AB)$. Thereby, the number of configurations we explore in grid search is 896.

3.3 Results

Table 3 shows the parameters and scores for the 5 best lexicon configurations according to F_1-measure. In general, we see that the variation of scores is low for the top 5 configurations within a pattern class. For example, the micro F_1-measures for the rule pattern $c_s, p \Rightarrow l_s$ vary between 0.33 and 0.37. The micro F_1-measures for rule pattern $c_o, p \Rightarrow l_o$ are generally lower, but show also smaller variation across configurations, ranging between 0.2 and 0.21.

Table 2 shows the number of lexical entries induced per frame type separately for the 5 best configurations for each rule in addition to the overall number of lexical entries. Over all configurations, a clear pattern emerges. First of all, it can be seen that the configurations for rule pattern $c_s, p \Rightarrow l_s$ are more productive, creating an order of magnitude more lexical entries compared to the pattern $c_o, p \Rightarrow l_o$. In terms of distribution of frame types, about 75% of the induced lexical entries are of type `NounPPFrame`, representing relational nouns. About 15% of the induced lexical entries are verb frames, with more or less an equal share of Transitive and IntransitivePP verb frames, and about 10% are adjective frames.

As can be seen in Table 4, in terms of micro F_1 measure the results using the automatically induced lexicon are 0.42 under the upper baseline using the manually created lexicon (micro F_1 of 0.79). This corresponds to a relative performance degradation of about 53%.

Training Data						
Lexicon	Micro-P	Micro-R	Micro-F_1	Macro-P	Macro-R	Macro-F_1
s-L1	0.32	0.44	0.37	0.40	0.40	0.40
o-L1	0.15	0.36	0.21	0.22	0.23	0.27
manual	0.84	0.75	0.79	0.61	0.62	0.61
Test Data						
Lexicon	Micro-P	Micro-R	Micro-F_1	Macro-P	Macro-R	Macro-F_1
s-L1	0.023	0.005	0.008	0.139	0.139	0.139
o-L1	0.015	0.004	0.007	0.093	0.093	0.093
manual	0.63	0.01	0.02	0.24	0.24	0.24

Table 4: Comparison of the evaluation results on the QALD-7 training data and test data for the best-performing lexicon automatically induced for $c_s, p \Rightarrow l_s$ rules and for the best-performing lexicon automatically induced for $c_o, p \Rightarrow l_o$ rules with results for the manually created lexicon.

Overall, these results clearly show that, while our method successfully induces many appropriate lexical entries, with the completely automatically generated lexicon the performance is far from the results obtained with a manually created lexicon.

System	Micro-P	Micro-R	Micro-F
WDAqua-core1	0.37	0.39	0.39
CNN-QA	–	–	0.29
$c_s, p \Rightarrow l_s$	0.32	0.44	0.37
$c_o, p \Rightarrow l_o$	0.15	0.36	0.21
manual	0.84	0.75	0.79

Table 5: Comparison of best result of LexExMachinaQA (i.e., $c_s, p \Rightarrow l_s$ and $c_o, p \Rightarrow l_o$) with the systems evaluated on QALD-7 dataset.

Table 5 shows the results of the evaluations using the best configurations for the rule patterns $c_s, p \Rightarrow l_s$ and $c_o, p \Rightarrow l_o$, for the rule-based systems WDAqua-core1 (Diefenbach et al., 2020), and for the machine learning-based approach CNN-QA (Sorokin and Gurevych, 2017). We compare the results of our approach to these two approaches as they have also been evaluated on QALD-7 training dataset. As can be seen from Table 5, the approach using the manually created lexicon outperforms state-of-the-art systems by a large margin (F_1 of 0.79 compared to 0.39 by the WDAqua-core1 system). This clearly shows the potential of our lexicon-based approach. Concerning the results using the automatically induced lexicon for rule pattern $c_s, p \Rightarrow l_s$, we see that our approach outperforms the CCN-QA approach (F_1 of 0.37 vs. 0.29) and has comparable performance to WDAqua-core1 (F_1 of 0.37 vs. 0.39). This is a remarkable result, showing that our approach can outperform state-of-the-art systems using a fully automatically generated lexicon. If a high quality lexicon is available, our approach outperforms SOTA systems by almost doubling performance.

3.4 Qualitative Analysis

In order to illustrate the working of our system, we analyze its behaviour in more detail by discussing 6 types of cases. Hereby, we rely on the best lexicon obtained from $c_s, p \Rightarrow l_s$ rules (i.e., s-L1). In particular, we sample 150 questions from the QALD-7 training set and classify them into six cases.[8]

Case 1 (Exact lexicalization): There are many cases in which the grammar generation based on an automatically induced lexicon generates exactly the same (question, query) pair as contained in the QALD-7 dataset. This is the case for 59 out of 150 (i.e., 39.33%) questions. An example here is the question *'In which year was Rachel Stevens born?'*

Case 2 (different variations but correct lexicalization): A second case is the one where our grammar generation based on the automatically induced lexicon generates a question that is semantically equivalent to a QALD-7 question, but that contains a synonym or variant of the lexical element in the ground truth question. In many cases, the generated question is grammatically correct and expresses the same meaning. According to our analysis, 12 out of 150 (i.e., 8%) questions are not identical but semantically equivalent. An example is the QALD-7 question *'When was the Titanic completed?'* In this case, the most similar automatically generated question is *'When was RMS Titanic completed on?'*

Case 3 (different variations but incorrect lexicalization): For 9 out of 150 (i.e., 6%) questions, our approach generates a question that features an incorrect lexicalization of the relevant

[8] 23% do not belong to any of these classes.

property. Consider the QALD-7 question *'What is the currency of the Czech Republic?'* In this question, *'currency of'* refers to the property `dbo:currency`. Our approach incorrectly induces that *'republic of'* denotes the property `dbo:currency` and thus generates the question: *'What is the republic of Czech Republic?'* which nevertheless retrieves the correct answer.

Case 4 (same lexicalization but different SPARQL query): There are cases where a question generated by an automatically induced lexicon is equivalent to a question in QALD-7, but the corresponding SPARQL queries differ. The question *'Who is the president of Eritrea?'* is generated, but instead of relating *'president of'* to `dbo:leader` as required to retrieve the correct answer in QALD-7, our lexicon induction approach relates *'president of'* to `dbo:office`, thus generating the same question but with a different SPARQL query, thus retrieving a different answer. This is the case for 6 out of 150 (i.e., 4%) questions.

Case 5 (Ask query): 20 out of 150 (i.e., 13.33%) questions in QALD-7 are ASK queries. The grammar generation excludes ASK queries because many of these questions are those whose answer is No. In this case, the SPARQL query of the question generated by automatically induced lexicalization is different from QALD-7 ones.

Case 6 (complex query): QueGG allows handling questions that are realized by a simple query.[9] QueGG has limited support for questions for which the corresponding query is complex, such as the following question-query pair:

```
Who is the mayor of the
capital of French Polynesia?

SELECT ?uri WHERE { res:French_
Polynesia dbo:capital ?x .
?x dbo:mayor ?uri .}
```

10 out of 150 (i.e., 6.6%) questions in QALD-7 are complex queries. The most similar question generated by the automatically induced grammar is *'What is the capital of French Polynesia?'*. In our case, none of these questions retrieves all answers as one or more lexicalization is not correct.

The qualitative evaluation thus shows that in some cases our approach generates correct questions with alternative but valid interpretations that do not match the QALD-7 gold standard. The evaluation thus underestimates the performance of our approach in some cases.

4 Related Work

The automatic acquisition of a lexicon from a corpus is not a new idea. For example, (Zernik, 1989) describes a method to automatically extract lexical entries, where an entry's semantics is expressed via a semantic template, different configurations in which the syntactic arguments can be organized are recorded etc. Furthermore, *semi-automated semantic knowledge base construction and multilingual lexicon acquisition* was one of the foci of the Penman project, which started in 1978 (Hovy, 1993).

In the context of the task of Automatic Question Generation, one can distinguish between the generation of questions from natural language text, e.g., (Heilman and Smith, 2009; Curto et al., 2012; Zhang et al., 2021) and the generation of questions from a knowledge base, e.g., (Chaudhri et al., 2014; Bordes et al., 2015; Raynaud et al., 2018; Bi et al., 2020).

Question generation from text makes use of manually created rules or trained models that transform a sentence into a question.

Several works mine relation-specific patterns from corpora. The approach M-ATOLL by Walter et al. (Walter et al., 2014) mines textual patterns that denote binary relations between entities. The text corpus is dependency-parsed and natural language patterns are identified via a set of manually defined dependency graph patterns that are matched against the parsed text. The resulting patterns are represented in *lemon* format. In contrast to M-ATOLL, the LexExMachina approach does not rely on a pre-definend set of patterns, but mines the patterns inductively from data (that has not been dependency-parsed).

A good overview about Natural Language Generation (NLG) from RDF can be found in the context of the WebNLG challenge[10] (Gardent et al., 2017). Approaches that tackle this challenge need to be able to carry out tasks such as sentence segmentation, lexicalization, aggregation, and surface real-

[9] A simple SPARQL query consists of a triple pattern with the predicate `rdf:type`, a triple pattern with the predicate `rdfs:label` and one more triple pattern.

[10] https://webnlg-challenge.loria.fr/

isation. Several of these tasks could make use of an automatically generated lexicon as we generate from LexExMachina rules. Recent work by Moussallem et al. (Moussallem et al., 2020) presents an approach based on a encoder-decoder architecture that is capable of generating multilingual verbalizations. Explicit linguistic knowledge in the form of automatically generated lexica could probably be incorporated into their approach.

The (syntactic) frames we used represent only a small set of possible syntactic frames and overlap with frames defined in `VerbNet` (Kipper et al., 2008). Our frames are by nature mainly syntactically defined and differ from the more semantic frames defined in `FrameNet` (Baker et al., 1998).

5 Conclusions and Future Work

We have presented LexExMachinaQA, a framework that allows to evaluate the impact of automatically induced ontology lexica on Question Answering over Linked Data. The framework builds on the LexExMachina approach that mines class-specific association rules over a loosely coupled text and KG dataset. We show how the association rules can be transformed into lemon lexical entries and rely on the QueGG approach to automatically create a grammar from the induced lexicon that can be used to parse questions into SPARQL queries over the corresponding vocabulary. We have evaluated the impact of the automatically induced lexica with respect to the English part of the QALD-7 dataset in terms of F_1-measure. While our method for lexicon induction yields many reasonable lexical entries that provide a baseline QA performance, our results show that it is not yet feasible to induce a lexicon that comes close to a manually created lexicon by fully automatic means. While not being able to fully replace a manually created lexicon, our method has clearly the potential to contribute to overcoming the lexical gap in Question Answering over Linked Data. In future work we will investigate if the proposed method works for other loosely-coupled datasets beyond Wikipedia/DBpedia and examine if the induced lexical knowledge can be used by QA approaches other than QueGG.

6 Acknowledgements

This research is part of the project eTaRDiS[11] (Exploration of Temporal and Spatial Data in Immersive Scenarios), which is funded by the Federal Ministry of Education and Research (BMBF). Basil Ell is partially funded by the SIRIUS centre: Norwegian Research Council project No 237898.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *36th Annual Meeting of the Association for Computational Linguistics, Volume 1*, pages 86–90.

Viktoria Benz, Philipp Cimiano, Mohammad Fazleh Elahi, and Basil Ell. 2020. Generating Grammars from Lemon Lexica for Questions Answering over Linked Data: a Preliminary Analysis. In *NLIWOD Workshop*, volume 2722 of *CEUR Workshop Proceedings*, pages 40–55.

Sheng Bi, Xiya Cheng, Yuan-Fang Li, Yongzhen Wang, and Guilin Qi. 2020. Knowledge-enriched, Type-constrained and Grammar-guided Question Generation over Knowledge Bases. In *COLING 2020*, pages 2776–2786.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale Simple Question Answering with Memory Networks. *CoRR*, abs/1506.02075.

Sergey Brin, Rajeev Motwani, and Craig Silverstein. 1997. Beyond Market Baskets: Generalizing Association Rules to Correlations. In *ACM SIGMOD 1997*, pages 265–276.

Vinay K Chaudhri, Peter E Clark, Adam Overholtzer, and Aaron Spaulding. 2014. Question generation from a knowledge base. In *EKAW 2014*, pages 54–65.

Philipp Cimiano, Paul Buitelaar, John P. McCrae, and Michael Sintek. 2011. LexInfo: A declarative model for the lexicon-ontology interface. *JWS*, 9(1):29–51.

Philipp Cimiano, John P. McCrae, and Paul Buitelaar. 2016. Lexicon Model for Ontologies: Community Report. In *W3C Community Group Final Report*.

Sérgio Curto, Ana Cristina Mendes, and Luisa Coheur. 2012. Question Generation based on Lexico-Syntactic Patterns Learned from the Web. *Dialogue & Discourse*, 3(2):147–175.

Dennis Diefenbach, Andreas Both, Kamal Singh, and Pierre Maret. 2020. Towards a question answering system over the semantic web. *Semantic Web*, 11(3):421–439.

[11] `https://digital-history.uni-bielefeld.de/etardis/`

Mohammad Fazleh Elahi, Basil Ell, Frank Grimm, and Philipp Cimiano. 2021. Question Answering on RDF Data based on Grammars Automatically Generated from Lemon Models. In *SEMANTiCS 2021*.

Basil Ell, Mohammad Fazleh Elahi, and Philipp Cimiano. 2021. Bridging the Gap Between Ontology and Lexicon via Class-Specific Association Rules Mined from a Loosely-Parallel Text-Data Corpus. In *LDK 2021*, pages 33:1–33:21.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *INLG*, pages 124–133.

Sherzod Hakimov, Soufian Jebbara, and Philipp Cimiano. 2017. AMUSE: Multilingual Semantic Parsing for Question Answering over Linked Data. In *ISWC 2017*, page 329–346.

Michael Heilman and Noah A. Smith. 2009. Question Generation via Overgenerating Transformations and Ranking. Technical report, Carnegie Mellon University.

Konrad Höffner, Sebastian Walter, Edgard Marx, Ricardo Usbeck, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo. 2017. Survey on Challenges of Question Answering in the Semantic Web. *Semantic Web*, 8(6):895–920.

Eduard H Hovy. 1993. Natural Language Processing by the Penman Project at USC/ISI. Technical report, University of Southern California Marina del Rey Information Sciences Institute.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of english verbs. *Language Resources and Evaluation*, 42:21–40.

John P. McCrae, Dennis Spohr, and Philipp Cimiano. 2011. Linking Lexical Resources and Ontologies on the Semantic Web with Lemon. In *ESWC 2011*, volume 6643, pages 245–259.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Diego Moussallem, Dwaraknath Gnaneshwar, Thiago Castro Ferreira, and Axel-Cyrille Ngonga Ngomo. 2020. NABU – Multilingual Graph-Based Neural RDF Verbalizer. In *ISWC 2020*, pages 420–437.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: A taxonomy of relational patterns with semantic types. In *EMNLP 2012*, pages 1135–1145.

Gennaro Nolano, Mohammad Fazleh Elahi, Maria Pia di Buono, Basil Ell, and Philipp Cimiano. 2022. An Italian Question Answering System based on grammars automatically generated from ontology lexica. In *CLiC-it 2022*.

Tanguy Raynaud, Julien Subercaze, and Frédérique Laforest. 2018. Thematic Question Generation over Knowledge Bases. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 1–8.

Martin G. Skjæveland, Daniel P. Lupp, Leif Harald Karlsen, and Henrik Forssell. 2018. Practical Ontology Pattern Instantiation, Discovery, and Maintenance with Reasonable Ontology Templates. In *ISWC 2018*, pages 477–494.

Daniil Sorokin and Iryna Gurevych. 2017. End-to-end representation learning for question answering with weak supervision. In *Semantic Web Challenges*, pages 70–83, Cham. Springer International Publishing.

Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Bastian Haarmann, Anastasia Krithara, Michael Röder, and Giulio Napolitano. 2017. 7th Open Challenge on Question Answering over Linked Data (QALD-7). In *Semantic Web Evaluation Challenge*, pages 59–69.

Sebastian Walter, Christina Unger, and Philipp Cimiano. 2014. M-ATOLL: A Framework for the Lexicalization of Ontologies in Multiple Languages. In *ISWC 2014*, pages 472–486.

Sebastian Walter, Christina Unger, Philipp Cimiano, and Daniel Bär. 2012. Evaluation of a Layered Approach to Question Answering over Linked Data. In *ISWC 2012*, page 362–374.

Uri Zernik. 1989. Lexicon Acquisition: Learning from Corpus by Capitalizing on Lexical Categories. In *IJCAI 1989*, pages 1556–1564.

Ruqing Zhang, Jiafeng Guo, Lu Chen, Yixing Fan, and Xueqi Cheng. 2021. A review on question generation from natural language text. *ACM Transactions on Information Systems (TOIS)*, 40(1):1–43.

A OTTR template definition: NounPPFrame

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ottr: <http://ns.ottr.xyz/0.4/> .
@prefix quegg: <http://example.org/quegg#> .
@prefix lemon: <http://example.org/lemon#> .
@prefix lexinfo: <http://example.org/lexinfo#> .

quegg:NounPPFrame[
   ?main_URI, ?canonical, ?singular, ?plural, ?property, ?domain,
   ?range, ?marker] :: {

  ottr:Triple(?main_URI, rdf:type, lemon:LexicalEntry),
  ottr:Triple(?main_URI, lexinfo:partOfSpeech, lexinfo:noun),
  ottr:Triple(?main_URI, lemon:canonicalForm, _:form_1),
  ottr:Triple(?main_URI, lemon:canonicalForm, _:form_2),
  ottr:Triple(?main_URI, lemon:synBehavior, _:nounpp),
  ottr:Triple(?main_URI, lemon:sense, _:sense_ontomap),

  ottr:Triple(_:form_1, rdf:type, lemon:Form),
  ottr:Triple(_:form_1, lemon:writtenRep, ?singular),

  ottr:Triple(_:form_2, rdf:type, lemon:Form),
  ottr:Triple(_:form_2, lemon:writtenRep, ?plural),


  ottr:Triple(_:nounpp, rdf:type, lexinfo:NounPPFrame),
  ottr:Triple(_:nounpp, lexinfo:copulativeArg, quegg:arg1),
  ottr:Triple(_:nounpp, lexinfo:prepositionalAdjunct, quegg:arg1),


  ottr:Triple(_:sense_ontomap, rdf:type, lemon:OntoMap),
  ottr:Triple(_:sense_ontomap, rdf:type, lemon:LexicalSense),

  ottr:Triple(_:sense_ontomap, lemon:ontoMapping, _:sense_ontomap),
  ottr:Triple(_:sense_ontomap, lemon:ontoMapping, _:sense_ontomap),
  ottr:Triple(_:sense_ontomap, lemon:reference, ?property),
  ottr:Triple(_:sense_ontomap, lemon:subjOfProp, quegg:arg2),
  ottr:Triple(_:sense_ontomap, lemon:objOfProp, quegg:arg1),
  ottr:Triple(_:sense_ontomap, lemon:condition, _:condition),

  ottr:Triple(_:condition, rdf:type, lemon:condition),
  ottr:Triple(_:condition, lemon:propertyDomain, ?domain),
  ottr:Triple(_:condition, lemon:propertyRange, ?range),

  ottr:Triple(_:condition, lemon:propertyRange, ?range),

  ottr:Triple(quegg:arg2, lemon:marker, ?marker),

  ottr:Triple(quegg:of, rdf:type, lemon:SynRoleMarker),

  ottr:Triple(quegg:of, lemon:canonicalForm, _:b1),
  ottr:Triple(_:b1,lemon:writtenRep, ?marker),
  ottr:Triple(_:b1, lexinfo:partOfSpeech, lexinfo:preposition)
} .
```

Figure 3: Definition of an OTTR template that can be used to create a lexical entry of type NounPPFrame.