QaldGen: Towards Microbenchmarking of Question Answering Systems over Knowledge Graphs

Kuldeep Singh[1]*, Muhammad Saleem[2]*, Abhishek Nadgeri[3], Felix Conrads[2],
Jeff Z. Pan[5,6], Axel-Cyrille Ngonga Ngomo[4], and Jens Lehmann[7]

[1] Nuance Communications Deutschland GmbH, Germany,
{kuldeep.singh1}@nuance.com
[2] University of Leipzig, Germany, saleem@informatik.uni-leipzig.de
[3] Service Lee technologies, India abhishek.n@servify.in
[4] University of Paderborn, Germany, axel.ngonga@upb.de
[5] Edinburgh Research Centre, Huawei, UK
[6] University of Aberdeen, UK jeff.z.pan@abdn.ac.uk
[7] Fraunhofer IAIS, Germany, {jens.lehmann}@iais.fraunhofer.de

Resource Type: *Evaluation benchmarks or Methods*
Repository: *https://github.com/dice-group/qald-generator*
License: *GNU General Public License v3.0*

Abstract. Over the last years, a number of Knowledge Graph (KG) based Question Answering (QA) systems have been developed. Consequently, the series of Question Answering Over Linked Data (QALD1–QALD9) challenges and other datasets have been proposed to evaluate these systems. However, the QA datasets contain a fixed number of natural language questions and do not allow users to select micro benchmarking samples of the questions tailored towards specific use-cases. We propose QaldGen, a framework for microbenchmarking of QA systems over KGs which is able to select customised question samples from existing QA datasets. The framework is flexible enough to select question samples of varying sizes and according to the user-defined criteria on the most important features to be considered for QA benchmarking. This is achieved using different clustering algorithms. We compare state-of-the-art QA systems over knowledge graphs by using different QA benchmarking samples. The observed results show that specialised micro-benchmarking is important to pinpoint the limitations of the various QA systems and its components.

1 Introduction

Question answering (QA) systems provide users with an interactive way to extract useful information from various sources such as documents, knowledge graphs, relational tables, etc. by posing questions in natural language or as voice

[*] These two authors contributed equally as first author

input. Since the initial stages of TREC challenge for QA over Web data in the year 1999 [33], researchers have developed several novel approaches that include question answering over structured and unstructured data sources [10,11]. Publicly available Knowledge Graphs (KGs) provide a rich source of structured information. Since 2010 more than 62 QA systems have been developed over KGs including DBpedia [1], Freebase [3] and Wikidata [34] as underlying knowledge source [9]. The question answering approaches over KGs can be broadly categorised into three categories based on their implementation [24]: the first category is *semantic parsing based QA systems* that heavily use linguistic principles such as POS tagging, dependency parsing, and entity recognition for extracting answers of the input question. It is often the case that there is no (or little) training data. The second category is *end-to-end machine learning based QA systems* that require large amounts of training data (e.g., [10]). Lastly, a recently introduced *collaborative QA systems* development focuses on building QA systems by reusing several existing QA systems and components (e.g. OKBQA [4]). Several independent components (e.g. EARL [7], Falcon [18], SQG [36]) perform tasks such as named entity disambiguation, relation linking, and SPARQL query generator for building QA systems in collaborative efforts have also been released by the semantic web research community.

Research Gap: Irrespective of the approaches opted for by researchers for the implementation, QA systems and components over knowledge graphs have been evaluated using several standard datasets such as LC-QuAD [28], QALD[29], and WebQuestion [2]. Nearly all report the results using the global metrics of precision, recall, and F-score as performance indicators [33]. Benchmarking Frameworks such as Gerbil [32] or leader boards[8] also follow the same principle and outline the final results based on the global performance metric. The results are calculated as ***average over all the (test) questions of the dataset*** and indicates the overall gain/loss in the performance w.r.t state of the art. However, it does not shed any light on the strength or weakness of a particular QA system and component. This allows the same issue to persist over time causing performance limitation of the QA system. For example, Muldoven et al. [16] pointed out in the year 2003 that answer type (boolean, count, list) and Wh-type questions (what, who, etc.) have an impact on the performance of the open domain QA systems. Saleem et al. [20] recently raised similar issues pertaining to QA systems over DBpedia. For instance, the overall winner of the 6th edition of the Question Answering over Linked Data Challenge (QALD6) was CANALI, which suffered limitations when the question started with `"Give me"`. CANALI is outperformed by another QA systems UTQA for such type of questions [20]. Similarly, the capitalisation of entity labels (surface forms) in a sentence is an issue reported by Derczynski et al. [5] by performing an in-depth analysis of entity linking tools. Sakor et al. [18] and Singh et al. [27,26] again reported this issue in state of the art entity linking tools evaluated over standard QA datasets. Therefore, it is evident that the common practice of reporting results average over all the questions of the dataset (often referred as macro evaluation) does

[8] `http://qa.mpi-inf.mpg.de/comqa/`

not always reveal details on state-of-the-art pitfalls, limitations, strengths, and potentials for further QA research.

Motivation and Contributions There are concrete pieces of evidence in the literature that question features such as "headword", "answer type", number of triples in SPARQL queries, explicit (exact string match between the entity candidate and KG mention) and implicit (no exact match i.e. mapping `NYC` to `dbr:New_York_City`[9]) nature of entities, etc. have an impact on the performance of the QA systems and the QA components [9,18,20,26,27]. Furthermore, question classification has been a long-standing field of research where researchers have focused on identifying several such features [12]. This motivates our work and in this article, we provide a reusable resource `QaldGen` for the community to select personalised question samples for microbenchmarking QA systems over the DBpedia knowledge graph. `QaldGen` uses state-of-the-art clustering algorithms to cluster most prominent questions by using distance metrics. `QaldGen` further allows researchers to select personalised question samples based on specific question features that evidently impact the performance of a QA system or component. We not only provide `QaldGen` for the QA system but also to evaluate several QA components that can be reused in collaborative QA frameworks for tasks such as named entity disambiguation (NED), relation linking (RL) (for mapping natural language relations to KG), and SPARQL query generator (Query Builder). Our contributions are two-fold:

- R1 **`QaldGenData`- An RDF Dataset for Personalised Microbenchmarking**: We automatically annotated a total 5408 questions from QALD9 and LC-QuAD datasets with 51 features and converted it into RDF format. This dataset can be reused in training machine learning approaches related to question answering.
- R2 **`QaldGen`-A Personalised Question Sample Generator**: We collected 51 question features from existing literature that impact the performance of the QA systems and components. A user can choose one or multiple question features to be included in the customised question sample for microbenchmarking. `QaldGen` selects personalised question samples of variable sizes using clustering algorithms from two standard datasets over DBpedia: QALD9 [17], containing 408 questions consolidating previous QALD editions and LC-QuAD [28], having 5000 questions. A user can customise questions (in terms of the number of question and the number of diverse features) using `QaldGen` to evaluate their QA system or the component either independently or using benchmarking frameworks such as Gerbil [32] or the Frankenstein platform [25].

The rest of this paper is organised as follows: the next section describes our two resources and approach to build `QaldGen`. Section 4 presents the importance and impact of this work for the research community. Section 5 presents our plan for availability and sustainability of resources. Section 6 reviews the state of the art and we close with the conclusion and future work in Section 7.

[9] Prefix `dbr` is bound to `http://dbpedia.org/resource/`

2 QaldGen Question Sample Generator

In this section, we present the question sampling process in the `QaldGen`. We first discuss the R1 dataset that we use as input for the `QaldGen` question sample generation framework. We then discuss the question sampling process along with the personalised micro benchmark generation.

2.1 `QaldGen` Dataset

Our framework takes a set of natural language questions as input and selects the required sample of questions according to the user-defined criteria. We use our R1 dataset as input to `QaldGen` where customised question samples will be selected from. As mentioned before, this RDF dataset of QA is selected from QALD9 and LC-QuAD which contains a total of 5408 questions. A QA benchmark should be comprising of questions/tests of varying question features. To this end, we have attached a total of 51 important QA related features summarised in Figure 1. We divide these features according to the question and the corresponding answer. The features attached to the question are related to the entities, relations and classes used in the SPARQL query, along with the natural language features such as headword, POS tags, etc. In addition, we store the number of words in the question and the origin (QALD9 or LC-QuAD) of the question. Each question has a SPARQL query to be executed to get the correct answers. We store SPARQL related features such as the number of projection variables, number of triple patterns, number of BGPs, filters, regex etc. A sample RDF representation of a `QaldGen` question is available in the listing 1.1. We re-used the Linked SPARQL Queries (LSQ) [19] vocabulary to represent the SPARQL query related features. In addition, we re-used properties from the QALD JSON datasets.

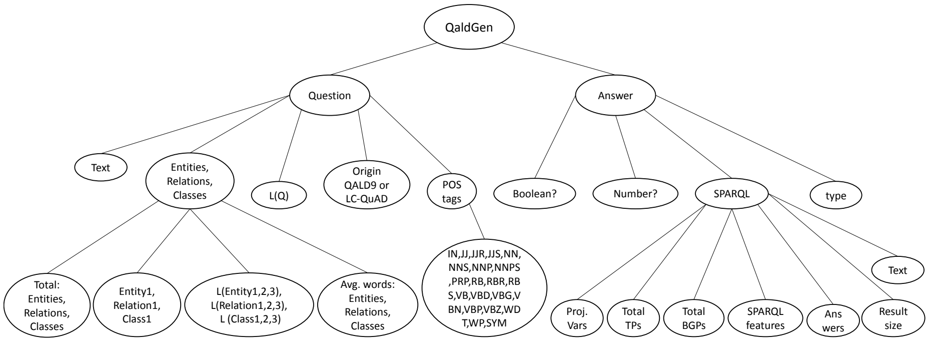
QaldGen
Question
Answer
Text
Entities, Relations, Classes
L(Q)
Origin QALD9 or LC-QuAD
POS tags
Boolean?
Number?
SPARQL
type
Total: Entities, Relations, Classes
Entity1, Relation1, Class1
L(Entity1,2,3), L(Relation1,2,3), L (Class1,2,3)
Avg. words: Entities, Relations, Classes
IN,JJ,JJR,JJS,NN, NNS,NNP,NNPS ,PRP,RB,RBR,RB S,VB,VBD,VBG,V BN,VBP,VBZ,WD T,WP,SYM
Text
Proj. Vars
Total TPs
Total BGPs
SPARQL features
Ans wers
Result size

Fig. 1. A tree of features attached to `QaldGen` dataset questions. L(X) stands for number of words in X.TPs = number of triple patterns, Proj. Vars = number of projection variables

The benefit of this dataset is that it can be directly queried using SPARQL and easily can be used in wide range of Linked Data and Semantic Web applications. In addition, each question has more features attached as compared to the original QALD9 and LC-QuAD datasets.

2.2 Question Sample Generation for Microbenchmarking

First, we define our question sampling generation problem and then explain the generation process. Our Question Sampling problem is defined as follows:

Definition 1 (Question Sampling Problem). *Let L represent the set of input natural language questions. Our goal is to select the N questions that best represent L as well as being more diverse in features, with $N << |L|$.*

Figure 2 shows the the general steps involved in the `QaldGen` question sample generation process. The user provides the input $\boxed{R1}$ RDF dataset, the required number N of questions and the selection criteria (as SPARQL query) to be considered in the question sampling for microbenchmarking. Then, the sampling is carried out in the following four main steps: (1) Select all the questions along with the required features from the input `QaldGenData` RDF dataset. (2) Generate feature vectors and their normalisation for the selected questions. (3) Generate N number of clusters from the questions. (4) Select single most representative questions from each cluster to be included in the final question sample requested by the user.

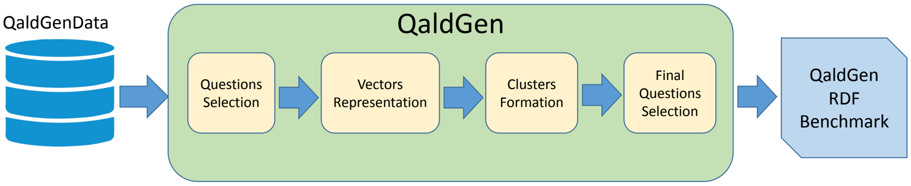

Fig. 2. QaldGen Sampling process.

Selection of questions We can select questions along with the required features by simply using SPARQL queries over `QaldGenData` RDF dataset. The `QaldGen` sampling framework retrieves the set of questions along with the required important features (can be any of the 51 features attached to each question) by using a single SPARQL query. For example, the SPARQL query given in Listing 1.2 retrieves all the questions from `QaldGen` RDF dataset along with the features: the total number of words, entities, relations, classes along with average words per entity in the question. In addition, it considers the number of

```
@prefix lsq: <http://lsq.aksw.org/vocab#> .
@prefix qaldgen: <http://qald-gen.aksw.org/vocab#> .
@prefix qaldgen-res: <http://qald-gen.aksw.org/> .

#Question related
qaldgen-res:question#0 qaldgen:text "What is the time zone of Salt Lake City?
     " ;
qaldgen:length "9"^^xsd:int ;

 # Entities
qaldgen:totalEntities "1"^^xsd:int ; qaldgen:entity1 "Salt_Lake_City, " ;
qaldgen:totalWordsEntity1 "1"^^xsd:int ; qaldgen:entity2 "" ;
qaldgen:totalWordsEntity2 "0"^^xsd:int ; qaldgen:entity3 "" ;
qaldgen:totalWordsEntity3 "0"^^xsd:int ; qaldgen:avgEntitiesWords "1.0"^^
     xsd:double ;

 # Relations
qaldgen:totalRelations "1"^^xsd:int ;
qaldgen:relation1 "timeZone" ;
qaldgen:totalWordsRelation1 "1"^^xsd:int ;
qaldgen:relation2 "" ;
qaldgen:totalWordsRelation2 "0"^^xsd:int ;
qaldgen:relation3 "" ;
qaldgen:totalWordsRelation3 "0"^^xsd:int ;
qaldgen:avgRelationsWords "1.0"^^xsd:double ;

 # Classes
qaldgen:totalClasses "0"^^xsd:int ;
qaldgen:class1 "" ;
qaldgen:totalWordsClass1 "0"^^xsd:int ;
qaldgen:class2 "" ;
qaldgen:totalWordsClass2 "0"^^xsd:int ;
qaldgen:class3 "" ;
qaldgen:totalWordsClass3 "0"^^xsd:int ;
qaldgen:avgClassesWords "0.0"^^xsd:double ;
qaldgen:answerType "resource" ;
qaldgen:isNumberAnswer "0" ;
qaldgen:isBooleanAnswer "0" ;

 # POS tags
qaldgen:IN "true"^^xsd:boolean ; qaldgen:JJ "false"^^xsd:boolean ;
qaldgen:JJR "false"^^xsd:boolean ; qaldgen:JJS "false"^^xsd:boolean ;
qaldgen:NN "true"^^xsd:boolean ; qaldgen:NNS "false"^^xsd:boolean ;
qaldgen:NNP "true"^^xsd:boolean ; qaldgen:NNPS "false"^^xsd:boolean ;
qaldgen:PRP "false"^^xsd:boolean ; qaldgen:RB "false"^^xsd:boolean ;
qaldgen:RBR "false"^^xsd:boolean ; qaldgen:RBS "false"^^xsd:boolean ;
qaldgen:VB "false"^^xsd:boolean ; qaldgen:VBD "false"^^xsd:boolean ;
qaldgen:VBG "false"^^xsd:boolean ; qaldgen:VBN "false"^^xsd:boolean ;
qaldgen:VBP "false"^^xsd:boolean ; qaldgen:VBZ "true"^^xsd:boolean ;
qaldgen:WDT "false"^^xsd:boolean ; qaldgen:WP "true"^^xsd:boolean ;
qaldgen:SYM "false"^^xsd:boolean ; qaldgen:questionOrigin "qald9" ;

# Answer related
lsq:text """SELECT DISTINCT ?uri WHERE { <http://dbpedia.org/resource/
     Salt_Lake_City> <http://dbpedia.org/ontology/timeZone ?uri }""" ;
lsq:tps "1"^^xsd:int ;
lsq:bgps "1"^^xsd:int ;
lsq:usesFeature lsq:Select , lsq:Distinct ;
lsq:projectVars "1"^^xsd:int ;
lsq:answers "( ?uri = <http://dbpedia.org/resource/Mountain_Time_Zone ), " ;
lsq:resultSize "1"^^xsd:int .
```

Listing 1.1. An example QaldGenData RDF representation of a question

```
Prefix qaldGen: <http://qald-gen.aksw.org/vocab#>
Prefix lsq: <http://lsq.aksw.org/vocab#>
SELECT  DISTINCT ?qId ?totalWords ?totalEntities ?totalRelations ?totalClasses ?avgEntitiesWords ?tps ?rs ?bgps ?pvars
{
?qId  qaldGen:length ?totalWords .
?qId  qaldGen:totalEntities ?totalEntities .
?qId  qaldGen:totalRelations ?totalRelations .
?qId  qaldGen:totalClasses ?totalClasses .
?qId  qaldGen:avgEntitiesWords ?avgEntitiesWords .
?qId  lsq:tps ?tps .
?qId  lsq:resultSize  ?rs .
?qId  lsq:bgps   ?bgps .
?qId  lsq:projectVars ?pvars .
}
```

Listing 1.2. Natural language questions selection along with required features from QaldGenData RDF dataset

triple patterns, the number of answers, the number of BGPs, and the number of projection variables. In other words, the user can select any number of features that are considered important for microbenchmarking. The result of this query execution is stored in a map that is used in the subsequent sampling steps. In Section 2.3, we show how this query can be modified to select customised samples for microbenchmarking.

Normalised Feature Vectors The cluster generation algorithms (explained in the next section) require distances between questions to be computed. Each question (that was retrieved in the previous step) from the input `QaldGenData` dataset is mapped to a vector of length equal to the number of retrieved features. The vector stores the corresponding *question features* that were retrieved along with the input questions. To ensure that dimensions with high values do not bias the selection of questions for benchmarking, we normalise the question feature vectors with values between 0 and 1. This is to ensure that all questions are located in a unit hypercube. At this point, each of the individual values in every feature vector is divided by the overall maximal value (across all the vectors) for that question feature.

Generation of Clusters As a next step, we generate N clusters from the given input QaldGen question represented as normalised feature vectors. For this step we used 5 existing well-known algorithms – FEASIBLE [21], FEASIBLE-Exemplars [21], KMeans++, DBSCAN+KMeans++, Random selection – which allow the generation of the required fixed number of clusters. Note DBSCAN+KMeans++ means that we applied DBSCAN first to remove the outlier queries and then applied KMeans++ to generate the required number of clusters. Please also consider that we need an additional normalisation of the remaining vectors after outliers are removed. Moreover, our framework is flexible enough to integrate any other clustering algorithm and allows the generation of a fixed number of clusters.

Selection of Most Representative Questions Finally, we perform the selection of a single prototypical question from each cluster. This step is exactly the same as performed in FEASIBLE [21]: For each cluster S, compute the centroid c which is the average of the feature vectors of all the queries in S. Following this, compute the distance of each query in S with c and select the query of minimum distance to include in the resulting benchmark.The final output of the QaldGen sampling generator is an RDF file containing the finally selected natural language questions along with the complete list of attached features. Thus the RDF output can be directly queried using SPARQL. Current QA benchmark execution frameworks such as GERBIL [32] require the benchmark to be represented in JSON format. Thus, `QaldGen` is also able to select GERBIL compatible QA benchmarks.

Our framework also allows the generation of question samples using random selection. In addition, it allows the generation of samples using Agglomerative clustering [21]. However, Agglomerative clustering does not allow the creation of fixed size samples. The QaldGen website contains the CLI options for the generation of benchmarks.

2.3 Question Sample Personalisation

As mentioned before, our framework allows customised question sample generation according to the criteria specified by the user. This can be done by simply specialising the query given in Listing 1.2. For example, imagine the user wants to select customised samples with the following features: The question sample should only be selected from QALD9 and hence skipping LC-QuAD questions. The personalised sample should only contain "what"-type questions, and the number of triple patterns should be greater than 1 and there should be at least one answer of this question. The query for the selection of such a personalised benchmark is given in Listing 1.3. Users can use such personalised question samples to micro benchmark the QA system.

2.4 Diversity of Question Samples

The selected sample should not be mostly comprised of a similar type of natural language question. To ensure the overall quality of the selected sample, sufficient diversity in the micro benchmarking questions is important, which `QaldGen` is able to select using different clustering algorithms. We define their diversity as follows:

```
Prefix qaldGen: <http://qald-gen.aksw.org/vocab#>
Prefix lsq: <http://lsq.aksw.org/vocab#>
SELECT  DISTINCT ?qId ?totalWords ?totalEntities ?totalRelations ?totalClasses ?avgEntitiesWords ?tps ?rs ?bgps ?pvars
{
?qId  qaldGen:length ?totalWords .
?qId  qaldGen:totalEntities ?totalEntities .
?qId  qaldGen:totalRelations ?totalRelations .
?qId  qaldGen:totalClasses ?totalClasses .
?qId  qaldGen:avgEntitiesWords ?avgEntitiesWords .
?qId  lsq:tps ?tps .
?qId  lsq:resultSize  ?rs .
?qId  lsq:bgps   ?bgps .
?qId  lsq:projectVars ?pvars .

# Options for Personalisation
?qId  qaldGen:questionOrigin ”qald9” .
?qId  qaldGen:questionType  ?qType .
Filter Regex (?qType, ”What”)
Filter (?tps > 1 && ?rs>0)
}
```

Listing 1.3. Benchmark Personalisation

Definition 2 (Question Sample Diversity). *Let μ_i mean and σ_i the standard deviation of a given distribution w.r.t. the i^{th} feature of the said distribution. Let B be a question sample extracted from a set of queries L.The diversity score D is the average standard deviation of the query features k included in the sample B:*

$$D = \frac{1}{k}\sum_{i=1}^{k}(\sigma_i(B)).$$

The command line tool provided on the project website will report the diversity score after the generation of the desired QA benchmark.

3 Evaluation and Results

In this section, we describe our evaluation setup and the results.

3.1 Experiment Setup

Micro Benchmarking: We used three personalised question samples in our evaluation at a micro level: (1) a 200 question sample was used to compare the QA systems. We used the FEASIBLE-Exemplars clustering method because it had the highest diversity score. We used the query given in Listing 1.2 to select the input questions for `QaldGen`. (2) a 100 question sample was selected using FEASIBLE-Exemplars to test the named entity disambiguation (NED) tools. In this benchmark we only consider features related to the named entities like the number of entities, the number of words in entities etc. (3) a 100 question sample was generated using FEASIBLE-Exemplars to test the relation linking (RL) tools. In this sample, we only considered features related to the relations used in the questions like number relations, number of words in relations, etc.

QA Systems: We compared all the systems which took part in the QALD9 challenge [17] and which are currently part of Gerbil framework. However, only ganswer2 and QUEPY were online and returned answers. We used `QaldGen` extension integrated in Gerbil to calculate the results. This also illustrates the adaptability and compatibility of $\boxed{\text{R2}}$ into a generic benchmarking framework. The results can be also found at persistent URI provided by Gerbil[10].

NED and RL Tools: We also extended our evaluation study at component level. We evaluated the top-2 components over LC-QuAD[11] performing NED (DBpedia Spotlight [15] and TagMe [8]) and RL tasks (RNLIWOD[12] and EARL [7]), respectively. For this study, we utilised the Frankenstein framework because these tools are already part of the framework[13]. The customised questions selected by `QaldGen` are uploaded in Frankenstein to calculate the final results.

3.2 Experiment Results

Table 1 summarises our evaluation results and we employ the performance metric of Precision (P), Recall (R), and F-Score (F) for reporting the results[14]. It is clearly observable in the table that the performance of QA systems and components fluctuate when customised question samples were selected using `QaldGen`. For instance, ganswer2 is the overall winner of QALD9, however, for a diverse set of 200 questions selected by `QaldGen`, QUEPY is the winner. This clearly show that performance of QA systems vary with the diversity of questions used in the evaluation. It is highly possible that a QA system is tuned for a particular type of question and may not perform well when exposed to question samples of varying diversity while performing microbenchmarking. In addition, a QA system designed for a particular use-case should only be tested with a use-case specific micro benchmark. Such systems will likely not perform well when tested with general QA benchmarks. For NED tools, TagMe remains the overall winner for macro and micro evaluation but its F-score drops sharply over customised sampling questions. When we consider specific question features (total number

[10] `http://gerbil-qa.aksw.org/gerbil/experiment?id=201903190000`

[11] as reported by [27] and [7]

[12] `https://github.com/dice-group/NLIWOD`

[13] `http://frankenstein.qanary-qa.com`

[14] `https://github.com/dice-group/gerbil/wiki/Precision,-Recall-and-F1-measure`

Table 1. Performance of QA systems and components. We can observe fluctuation of P,R,F values when customised question samples have been selected using R2. For instance, QUEPY is the new baseline outperforming ganswer2(overall baseline of QALD9 dataset) on the customised benchmark selected by `QaldGen`. Similar performance variation has been observed in the tools performing NED and RL tasks.

Systems	**Dataset**	**P**	**R**	**F**
baseline QA System(ganswer2) [17]	QALD9	0.29	0.33	0.30
ganswer2	QaldGen	0.24	0.24	0.24
QUEPY	QaldGen	**0.27**	**0.27**	**0.27**
baseline NED(TagMe) [27]	LCQuAD	0.69	0.67	0.68
TagMe	QaldGen	**0.44**	**0.40**	**0.41**
DBpedia Spotlight	QaldGen	0.37	0.38	0.36
baseline RL (RNLIWOD) [27]	LCQuAD	0.25	0.22	0.23
RNLIWOD	QaldGen	0.23	0.19	0.20
EARL	QaldGen	**0.38**	**0.39**	**0.37**

of entities as two and more than two words in the entity label) for generating a personalised question sample, the performance of TagMe is limited. In the case of RL tools, EARL outperforms RNLIWOD when we selected a personalised question sample with particular question features (explicit relations, number of relations=1). It provides a clear indication of the strength of the EARL tool for a particular type of questions. Please note that in the scope of this paper, we are not analysing the architecture of the QA systems and components to understand the performance variation for specific question features. Our aim is to illustrate with empirical results that the term "baseline or state of the art" solely depends on the type of questions and considered features. Therefore, using our reusable resources, developers can better understand the strength or weaknesses of their tools by evaluating their tools at micro level.

4 Impact

38 QA systems from over 40 research groups in the semantic web community have participated in nine editions of QALD [17]. The LC-QuAD dataset has also gained recognition and is already cited 25 times since its release in October 2018 [28]. However the same issues of question ambiguity, capitalisation of entity labels, complex questions, implicit/explicit nature of entity and relation label, etc. are reported repeatedly in the evaluation studies [26,20,9].

In this article, we provide the semantic web community with two reusable resources for micro benchmarking of QA components and systems. Please note that we are **not** proposing any new benchmarking dataset or a benchmarking framework such as Frankenstein [25] or Gerbil [31]. We reuse the existing QA datasets (QALD9 and LC-QuAD) over DBpedia and in addition have developed R2 that can be reused by QA developers to select personalised question samples for micro benchmarking their systems and components. This is the first step towards a more fine grained evaluation of QA systems and we expect the QA community to utilise $\boxed{\text{R2}}$ to dig deeper into the strength and weaknesses of their systems. Similar effort has also been started towards micro-benchmarking entity linking tools [35], however micro benchmarking of QA components and systems is a major research gap. Our second resource $\boxed{\text{R1}}$ can be reused by developers as a rich source of questions represented in RDF format with its features. Furthermore, we hope fine grained benchmarking of QA systems will trigger discussion within the community to release datasets with diverse question features.

5 Adoption and Reusability

The resources $\boxed{\text{R2}}$ and $\boxed{\text{R1}}$ are licensed under the GNU General Public License v3.0 and publicly available for reuse. Detailed instructions are provided for the easy adaptability of the resources. Developers can either use the proposed resources independently or can select personalised benchmarking questions using $\boxed{\text{R2}}$ and adapt it for benchmarking frameworks. For community adaptation, both resources are made compatible with Gerbil [31] and Frankenstein framework [25]. Furthermore, Gerbil supports an easy extension of its core architecture and it is a widely used platform for calculating global performance metrics of QA systems and entity linking components. The maven version of $\boxed{\text{R2}}$ has already been integrated with Gerbil. Therefore, researchers can choose customised benchmarking questions based on their needs, and use Gerbil to calculate the global performance metric for a sub-set of customised questions selected by $\boxed{\text{R2}}$. The extension of Gerbil with `QaldGen` is already completed and it is available for public reuse in official Github repository of Gerbil[15].

6 Related Work

QA Systems and Macro Benchmarking: TREC QA series [33] for evaluating open domain question answering was one of the earlier attempts in the direction of providing researchers with standard datasets and a performance metric for benchmarking QA systems. Question answering over KG gained momentum in the last decade after the inception of publicly available KGs such as DBpedia and Freebase. Datasets such as SimpleQuestions[16] and WebQuestions [2] are commonly used for evaluating QA systems that employ Freebase as the underlying KG. For benchmarking QA systems over DBpedia, the QALD series was launched in 2011[17] and is currently running its 9th edition. In the last 8 years, over 38 QA systems using DBpedia as the underlying KG have been evaluated using QALD [9]. However, the maximum number of questions in QALD

[15] `https://github.com/dice-group/gerbil/wiki/QALD-Generation`

[16] `https://research.fb.com/downloads/babi/`

[17] `http://qald.aksw.org/index.php?x=home&q=1`

is 408 which hinders the development of machine learning based approaches. In contrast with QALD, LC-QuAD dataset provides a rich and diverse set of 5000 complex questions for DBpedia [28]. Recently developed QA systems and frameworks also report results on LC-QuAD [6,13,27]. However, the reported results across these datasets are on **macro level** i.e. an average on all questions of the dataset.

Question Classification and Micro Benchmarking: Question classification techniques aim to classify questions based on several features that may impact the overall performance of the QA system [12]. In the semantic web community, Usbeck et al. [30] first attempted to use question classification and question features such as headword, answer type, entity type, etc were extracted to provide a labelled representation of each question. Classifiers were trained using these features to choose a QA system among six that can potentially answer an input question. This approach resulted in an increase in the overall performance of the proposed hybrid QA system. Saleem et al. [20] used question classification to **micro benchmark** (i.e. reporting results based on the type of the questions containing specific features) QA systems to understand the strengths and the weaknesses. The authors conclude that a QA system which is the overall winner for all questions of the QALD-6 dataset is not always the winner for the questions with particular features. This empirical study also revealed that macro F-score (average on all questions) varies a lot based on the type of questions. For example, the highest reported F-score for all questions is 0.89 (CANALI QA system [14]). However, for a specific type of questions starting with "Give me" (e.g. Give me all cosmonauts.), F-score sharply drops to 0.34 and the UTQA QA system outperforms CANALI. Singh et al [26,27] extended the concept of micro-benchmarking to QA component level where exhaustive evaluations of 28 QA components including 20 named entity disambiguation, five relation linking, two class linking, and two SPARQL query generator have been performed using over 3000 questions from LC-QuAD. The authors observe significant fluctuation of the F-score even at the component level published in the extended study of Frankenstein framework [26]. For example, SINA [23] generates SPARQL queries which requires DBpedia URIs of entities and predicates present in the question as input. SINA is the baseline (F-Score 0.80) for the subset of 729 questions from LC-QuAD having SPARQL queries with two triples compared to the overall winner (F-score 0.48 reported by NLIWOD component) on all the questions of LC-QuAD dataset considered by authors. However, when the number of triples in SPARQL queries is four, SINA reports F-score 0.0 for a subset of 1256 questions. The above-mentioned micro-benchmarking studies provide a foundation to our work. We reuse all the question features reported by [20,27] in `QaldGen` for micro benchmarking.

The work by Waitelonis et al. [35] proposes fine-grained benchmarking of entity linking tools using the Gerbil framework [32] based on the features of entity type (person, organisation, place). This work is most closely related to our approach. Unlike `QaldGen`, the above-mentioned work is limited to the entity linking tools whereas the novelty of `QaldGen` is to provide a reusable resource for the fine-grained micro-benchmarking study of the QA systems and reusable components for QA frameworks performing various tasks(e.g. NED, RL, Query Builder).

Finally, there are benchmark generators available for SPARQL queries [21] to test the runtime performances of triplestores, and SPARQL query containments [22] to test the query containments solvers. However, to the best of our knowledge, there was no QA over Linked Data benchmark generator available to generator customised benchmarks.

7 Conclusion and Future Work

In this paper, we present two reusable resources for generating personalised question samples for micro benchmarking question answering systems over knowledge graphs, more specifically DBpedia. Our first offered resource $\boxed{\mathsf{R\,2}}$ is `QaldGen`. `QaldGen` uses state of the art clustering algorithms to cluster the most diverse or similar questions based on the features that impact the overall performance of QA systems. `QaldGen` is compatible with Gerbil and Frankenstein framework. Hence developers can directly use these frameworks to compare their system with state of the art on specific questions selected by `QaldGen` for personalised micro benchmarking.

The second resource $\boxed{\mathsf{R\,1}}$ is a collection of 5408 questions from two standard datasets with a diverse representation of 51 features in each question. In the previous works [27,30,21], QA developers extracted such features multiple times for different research studies. Using $\boxed{\mathsf{R\,1}}$ researchers can now select question features they would like to consider for training machine learning algorithms rather than extracting the features again from scratch. We believe that using our resources, researchers can now evaluate their systems on their specific needs. We also hope that our work will trigger discussion in the QA community to come up with a dataset containing more diverse question features and start reporting performance at the micro level. We plan to extend this work in three directions 1) extend questions to other knowledge graphs such as Wikidata 2) include more datasets in the $\boxed{\mathsf{R\,1}}$ and 3) develop a similar micro-benchmarking approach for open domain question answering datasets such as reading comprehension.

Acknowledgments

This work has been supported by the project LIMBO (Grant no. 19F2029I), OPAL (no. 19F2028A), KnowGraphs (no. 860801), and SOLIDE (no. 13N14456)

References

1. S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. DBpedia: A Nucleus for a Web of Open Data. In *ISWC*, pages 722–735, 2007.
2. J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*, pages 1533–1544. ACL, 2013.

3. K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD*, pages 1247–1250, 2008.
4. K.-S. Choi, T. Mitamura, P. Vossen, J.-D. Kim, and A.-C. N. Ngomo. SIGIR 2017 Workshop on Open Knowledge Base and Question Answering (OKBQA2017). In *Proceedings of the ACM SIGIR*, pages 1433–1434, 2017.
5. L. Derczynski, D. Maynard, G. Rizzo, M. van Erp, G. Gorrell, R. Troncy, J. Petrak, and K. Bontcheva. Analysis of named entity recognition and linking for tweets. *Inf. Process. Manage.*, 51(2):32–49, 2015.
6. D. Diefenbach, A. Both, K. Singh, and P. Maret. Towards a Question Answering System over the Semantic Web. *arXiv preprint arXiv:1803.00832*, 2018.
7. M. Dubey, D. Banerjee, D. Chaudhuri, and J. Lehmann. EARL: joint entity and relation linking for question answering over knowledge graphs. In *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*, pages 108–126, 2018.
8. P. Ferragina and U. Scaiella. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, pages 1625–1628. ACM, 2010.
9. K. Höffner, S. Walter, E. Marx, R. Usbeck, J. Lehmann, and A. N. Ngomo. Survey on challenges of Question Answering in the Semantic Web. *Semantic Web*, 8(6):895–920, 2017.
10. X. Huang, J. Zhang, D. Li, and P. Li. Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 105–113. ACM, 2019.
11. F. Li and H. V. Jagadish. Constructing an Interactive Natural Language Interface for Relational Databases. *PVLDB*, 8(1):73–84, 2014.
12. B. Loni. A survey of state-of-the-art methods on question classification. 2011.
13. G. Maheshwari, P. Trivedi, D. Lukovnikov, N. Chakraborty, A. Fischer, and J. Lehmann. Learning to rank query graphs for complex question answering over knowledge graphs. *arXiv preprint arXiv:1811.01118*, 2018.
14. G. M. Mazzeo and C. Zaniolo. Answering controlled natural language questions on rdf knowledge bases. In *EDBT*, pages 608–611, 2016.
15. P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. DBpedia spotlight: shedding light on the web of documents. In *Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011*, pages 1–8. ACM, 2011.
16. D. Moldovan, M. Paşca, S. Harabagiu, and M. Surdeanu. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems (TOIS)*, 21(2):133–154, 2003.
17. N. Ngomo. 9th challenge on question answering over linked data (qald-9). *language*, 7:1.
18. A. Sakor, O. I. Mulang, K. Singh, S. Shekarpour, M. E. Vidal, J. Lehmann, and S. Auer. Conference paper: Old is gold: Linguistic driven approach for entity and relation linking of short text. In *NAACL 2019 (to appear)*. ACL, 2019.
19. M. Saleem, M. I. Ali, A. Hogan, Q. Mehmood, and A.-C. N. Ngomo. Lsq: the linked sparql queries dataset. In *ISWC*, pages 261–269, 2015.
20. M. Saleem, S. N. Dastjerdi, R. Usbeck, and A. N. Ngomo. Question Answering Over Linked Data: What is Difficult to Answer? What Affects the F scores? In *Joint Proceedings of BLINK2017: co-located with (ISWC 2017), Austria*, 2017.

21. M. Saleem, Q. Mehmood, and A.-C. N. Ngomo. Feasible: A feature-based sparql benchmark generation framework. In *International Semantic Web Conference*, pages 52–69. Springer, 2015.
22. M. Saleem, C. Stadler, Q. Mehmood, J. Lehmann, and A.-C. N. Ngomo. Sqcframework: Sparql query containment benchmark generation framework. In *Proceedings of the Knowledge Capture Conference*, page 28. ACM, 2017.
23. S. Shekarpour, E. Marx, A. N. Ngomo, and S. Auer. SINA: Semantic interpretation of user queries for question answering on interlinked data. *J. Web Sem.*, 30:39–51, 2015.
24. K. Singh. *Towards Dynamic Composition of Question Answering Pipelines.* PhD thesis, University of Bonn, Germany, 2019.
25. K. Singh, A. Both, A. S. Radhakrishna, and S. Shekarpour. Frankenstein: A Platform Enabling Reuse of Question Answering Components. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, pages 624–638. Springer, 2018.
26. K. Singh, I. Lytra, A. S. Radhakrishna, S. Shekarpour, M.-E. Vidal, and J. Lehmann. No one is perfect: Analysing the performance of question answering components over the dbpedia knowledge graph. *arXiv:1809.10044*, 2018.
27. K. Singh, A. S. Radhakrishna, A. Both, S. Shekarpour, I. Lytra, R. Usbeck, A. Vyas, A. Khikmatullaev, D. Punjani, C. Lange, M. Vidal, J. Lehmann, and S. Auer. Why Reinvent the Wheel: Let's Build Question Answering Systems Together. In *Web Conference*, pages 1247–1256, 2018.
28. P. Trivedi, G. Maheshwari, M. Dubey, and J. Lehmann. LC-QuAD: A Corpus for Complex Question Answering over Knowledge Graphs. In *ISWC*, pages 210–218, 2017.
29. C. Unger, C. Forascu, V. López, A. N. Ngomo, E. Cabrio, P. Cimiano, and S. Walter. 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.* CEUR-WS.org, 2015.
30. R. Usbeck, M. Hoffmann, M. Röder, J. Lehmann, and A. N. Ngomo. Using Multi-Label Classification for Improved Question Answering. *CoRR*, 2017.
31. R. Usbeck, M. Röder, M. Hoffmann, F. Conrads, J. Huthmann, A.-C. N. Ngomo, C. Demmler, and C. Unger. Benchmarking Question Answering Systems. *Semantic Web Journal*, 2017.
32. R. Usbeck, M. Röder, A. N. Ngomo, C. Baron, A. Both, M. Brümmer, D. Ceccarelli, M. Cornolti, D. Cherix, B. Eickmann, P. Ferragina, C. Lemke, A. Moro, R. Navigli, F. Piccinno, G. Rizzo, H. Sack, R. Speck, R. Troncy, J. Waitelonis, and L. Wesemann. GERBIL: General Entity Annotator Benchmarking Framework. In *WWW 2015*, pages 1133–1143, 2015.
33. E. M. Voorhees and D. K. Harman, editors. *Proceedings of The Eighth Text REtrieval Conference, TREC 1999, Gaithersburg, Maryland, USA, November 17-19, 1999*, volume Special Publication 500-246. National Institute of Standards and Technology (NIST), 1999.
34. D. Vrandecic. Wikidata: a new platform for collaborative data collection. In *Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume)*, pages 1063–1064. ACM, 2012.
35. J. Waitelonis, H. Jürges, and H. Sack. Remixing entity linking evaluation datasets for focused benchmarking. *Semantic Web*, 10(2):385–412, 2019.
36. H. Zafar, G. Napolitano, and J. Lehmann. Formal query generation for question answering over knowledge bases. In *European Semantic Web Conference*, pages 714–728. Springer, 2018.