LinkedHealthAnswers: Towards Linked Data-driven Question Answering for the Health Care Domain

Artem Ostankov†, Florian Röhrbein†, Ulli Waltinger†

Robotics and Embedded Systems, Technical University Munich, Germany†
Siemens AG, Corporate Technology, Munich, Germany†
{artem.ostankov,florian.roehrbein}@in.tum.de, ulli.waltinger@siemens.com†

Abstract

This paper presents Linked Health Answers, a natural language question answering systems that utilizes health data drawn from the Linked Data Cloud. The contributions of this paper are three-fold: Firstly, we review existing state-of-the-art NLP platforms and components, with a special focus on components that allow or support an automatic SPARQL construction. Secondly, we present the implemented architecture of the Linked Health Answers systems. Thirdly, we propose an statistical bootstrap approach for the identification and disambiguation of RDF-based predicates using a machine learning-based classifier. The evaluation focuses on predicate detection in sentence statements, as well as within the scenario of natural language questions.

Keywords: Question Answering, Linked Data Cloud, Machine Learning

1. Introduction

The Semantic Web (SW) provides huge amount of structured interconnected data. The richness of this data provides new possibilities for research and industry, and opens up new approaches in the human computer interaction. While more and more RDF data is contributed to the SW, questions arise on how the user can access this body of knowledge in an intuitive way. In this context, Linked Data-driven question answering (QA) systems (Cimiano et al., 2013) have caught much attention most recently, as these systems allow users, even with a limited familiarity of technical systems and databases, to pose questions in a natural way and gain insights of the data available. In addition, there has been a renewed interest from industry in having computer systems not only analyze the vast amounts of information (Ferrucci et al., 2010), but also in providing intuitive user interfaces to pose questions in natural language (NL) (Waltinger et al., 2013) in an interactive dialogue manner (Sonntag, 2009; Waltinger et al., 2011; Waltinger et al., 2012). More specifically, several industrial applications of question answering have raised the interest and awareness of this technology as an effective way to interact with a system: IBM Watson’s Jeopardy challenge (Ferrucci et al., 2010) showed that open domain QA can be done accurately and at scale; Wolfram Alpha’s computational knowledge engine centered around Mathematica as one source behind Apple’s Siri, which has proven a successful interaction medium for mobile devices. One of the key challenges in question answering using RDF-based data is the automatic mapping of natural language questions onto their appropriate SPARQL query representation, which subsequently connects a number of interlinked RDF repositories. That is, translating the individual parts of a natural language question to their respective URI representation, as needed for the underlying query language (Waltinger et al., 2012). For example, when trying to answer the question “What are the side effects of penicillin?” with respect to the Unified Medical Language System data set, the name Penicillin needs to be mapped to the resource <http://linkedlifedata.com/resource/umls/id/C0220892>, and side effects needs to be mapped to the predicate side:SideEffect. In this context, the interpretation of natural language questions refers to the process of an automatic triplification for SPARQL query language construction, and its corresponding challenges of an automatic concept identification and URI-based (subject, predicate, object) disambiguation.

The paper is organized as follows. In section 2, we present related work within the context of question answering systems. In addition, we present an analysis of existing state-of-the-art NLP platforms and components. In section 3, present the implemented architecture of Linked Health Answers, a Linked Data-driven question answering system that targets the health care domain. Section 4 describes a novel machine learning approach for the identification and disambiguation of RDF-based predicates. The proposed algorithms will be evaluated and discussed within section 5.

2. Existing Component Review

Most recent work on question answering over linked data that supports an automatic query language construction of questions (Shekarpour et al., 2013; Hakimov et al., 2013), using a template-based triple translation (Unger et al., 2012), and/or utilizing the Yago (Adolphs et al., 2011; Yahya et al., 2012) or DBpedia (Lopez et al., 2010) ontology for the triplification process. In the construction of the Linked Health Answers system, we focused initially on the analysis of existing platforms and components that may serve as a basis for the overall QA and/or query language construction process. As the application area of the work was the medical context, medical targeting software

---

1 http://www.wolframalpha.com/
2 http://www.apple.com/de/ios/siri/
3 Accessible through linkedlifedata.com
was in preference over the generic one. With reference to the framework and processing architecture decision, we reviewed and analyzed several existing platforms with an emphasis on programming language, feature stack, and licensing options (see Table 1 for an overview). All of the considered frameworks provide a set of natural language processing (NLP) units and/or existing state-of-the-art text annotators, as well as offers high configurability. From the list of processing architectures, we have chosen to use parts of the UIMA-based cTakes system (cta, 2008) for three reasons: First, this architecture focuses on the clinical narrative infrastructure including a broad set (>10) of existing processing annotators (i.e. identifying types of clinical named entities such as drugs, diseases, disorders and anatomical sites). Second, it is based on UIMA framework and allows to integrate standard NLP processing components (i.e. OpenNLP). Third, the open-source license by means of the Apache Software License version 2.0. In the scope of the automatic triplification task, we analyzed various components in terms of their methods used, their ontology mapping support, and their type of available input resources. Amongst others, we evaluated the following state-of-the-art components: PowerAqua (Lopez et al., 2012), FREyA (Damljanovic et al., 2011), QuestIO (Damljanovic et al., 2008), and ORAKEL (Cimiano et al., 2008), all of which offer a natural language interface to an ontology representation and/or RDF-based knowledge base (KB). It implies that the information can be retrieved with a formal semantic request (i.e. SPARQL, MQL). A formal request has the following generalized triple pattern: <?s, ?p, ?o>, where ?s is the subject, ?p is the predicate/property, and ?o is the object. To answer a specific question, we need to construct query with defined subject (?s) and property (?p), and an undefined object (?o), such as:

1. Question:
What are the side effects of penicillin?

2. Query-Template:
SELECT ?s WHERE { ?s ?p ?o. }

3. URI-Mapping:
(?p,property,side effects);
(?o,resource,penicillin);

3. Overview of Linked Health Answers
The Linked Health Answers system can be described as a pipeline that accepts a human composed natural language question as an input and provides list-based answers in a human readable form. Initially, the system was targeting only medical context as its main application area. In the later stages, we have generalized the system, to show its capability to work also within other application areas. With reference to the answering procedure, the system relies on the information accessible through an existing (RDF-based) knowledge base (KB). It implies that the information can be retrieved with a formal semantic request (i.e. SPARQL, MQL). A formal request has the following generalized triple pattern: <?s, ?p, ?o>, where ?s is the subject, ?p is the predicate/property, and ?o is the object. To answer a specific question, we need to construct query with defined subject (?s) and property (?p), and an undefined object (?o), such as:

1. Question:
What are the side effects of penicillin?

2. Query-Template:
SELECT ?s WHERE { ?s ?p ?o. }

3. URI-Mapping:
(?p,property,side effects);
(?o,resource,penicillin);
### Information Extraction
Uses Wikipedia infoboxes
Self-supervised IE from Wikipedia
Classes, taxonomy, datatype properties, instances, property values

### Components of the Ontology Extracted

| System | Method(s) | Features | Components of the Ontology Extracted |
|--------|-----------|----------|--------------------------------------|
| Kylin (Wu et al., 2008) | Uses Wikipedia infoboxes | Self-supervised IE from Wikipedia | Classes, taxonomy, datatype properties, instances, property values |
| OntoSyphon (McDowell and Cafarella, 2006) | Instances search based on the ontology information | Ontology-driven | Instances; relations |
| Text-To-Onto (Maedche and Staab, 2000) | NLP processing; domain lexical processing; relation learning algorithm | Semi-automatically adding discovered conceptual structures to the ontology | Classes, taxonomy, other relationships |
| ontoX (Yildiz and Miksch, 2007) | Linguistic rules; Rules generation module; Utilizes the content and predefined semantics in OWL | Instances, datatype property values |
| KIM (Popov et al., 2004) | Linguistic rules, Gazetteer lists; Uses upper-level ontology; Based on GATE; reasoning | Instances, property values |
| Evolva (Zablith et al., 2009) | Iterative disambiguation | Uses set of ontologies; Plug-in to NeOn; Classes, taxonomy, instances |
| TextRunner (Sarawagi, 2008) | Machine learned algorithms | Created by semi-supervised algorithm | Classes, relations |

Table 3: List of components analyzed that focus on the information extraction and disambiguation task.

In this context, the object (?o) represented as a list of values, refers to the answer(s) of the initial question. Therefore, the main goal of the system is to parse the natural language question and produce an accurate formal semantic request with reference to the KB used.

#### 3.1. System pipeline
The pipeline paradigm was picked as an architecture pattern for the developed system. The pipeline consists of several nodes, connected by the data flow interface. Each node refers to a specific component, that induces the query transformation process. The main components of the system are (1) linguistic component; (2) concept identifier; (3) query mapper; (4) ontology connector; (5) output provider. All components contain one or more sub-components, which provide different approaches to the same task. From a development perspective, the system was build using a vertical slicing approach, which was horizontally extended (see Figure 1). In other words, we have first created the stable system, which used only one sub-component of each component. Subsequently, we have enhanced the system with additional sub-components (i.e. dynamic predicate detection) and connected it to additional KBs (i.e. DBpedia, Freebase).

#### 3.2. Linguistic Component
The linguistic component refers to a standard NLP processing phase, utilizing tokenization, lemmatization, part-of-speech-tagging, and named entity recognition. In addition, the main role of this component is to break the question into a list of terms/phrases, which are holding the semantic load of the question, and to induce the triplification of the given question. From the broad set of analyzed natural language interfaces to ontology tools, we have used *Power Aqua* for the triplification process. However on the later stages, the *Power Aqua* system was re-engineered and only the linguistic component of the system was reused. This component is the part of the GATE framework and its main role is to parse natural language question and generate question terms in the triplified form: $s$, $p$, $o$. In addition, we have used the question type annotation of *Power Aqua*. Question types are predefined set of different types of question, such as description or wh-specific questions. (see Table 4). To give an example, for the question "What drugs cure asthma?", the component will generate the following triple: question term ($s$): drug; relation term ($p$): cure; second term ($o$): asthma.

#### 3.3. Concept Identifier
The next stage of the formal request construction focuses on the entity identification task. That is, any term/phrase candidate of the prior component needs to be represented as a unique resource identifier. From the example above: "What drugs cure asthma?", the term asthma needs to be mapped to its according URI representation as specified in the target KB-based ontology. As the system targets the health care domain, we utilize the UMLS (Unified Medical Language System) as a reference plane for the identification of medical concepts. We have used the UMLS not only because it is the most widely used metathesaurus reference system in the medical domain, but it also provides unique resource identifier (called UMLS CUI i.e. C0220892 for *penicillin*), which are accessible through the Linked Life Data platform and its associated SPARQL endpoint (i.e. the URI for *penicillin*: <http://linkedlifedata.com/resource/umls/id/C0220892>). For the task of Concept-to-URI mapping, we have reviewed various tools from a medical context perspective (see Table 3). Finally, we have decided to use MetaMap (Aronson, 2001), as provided from the National Library of Medicine for the task of concept identification. MetaMap provides an effective mapping of bio-medical text to the UMLS Metathesaurus. In addition, it is also included as one of the components within the cTakes framework. To give an
Figure 1: System schematic pipeline of the Linked Health Answer system

| Question                                                                 | Query term | Relation term | Second term               | Type                    |
|--------------------------------------------------------------------------|------------|---------------|---------------------------|-------------------------|
| Which drugs helps against Epilepsy?                                      | drug       | help          | Epilepsy                  | wh-generic term         |
| Find all literature about cancer?                                        | literature | cancer        |                           | wh-unknown relation     |
| What side effects has aspirin?                                           | side effects | aspirin     |                           | wh-generic term         |
| What is influenza?                                                       | desc type  | influenza     |                           | description             |
| Which trials are made by Munich Municipal Hospital?                      | trials     | Munich Municipal? | Hospital       | wh-generic term         |

Table 4: Example questions after the triplification process by question type.

example, MetaMap generates the following UMLS-CUIs (http://linkedlifedata.com/resource/umls/id/):

- 'drug' -> C0013227
- 'cure' -> C1880198
- 'asthma' -> C0004096

For concepts that could not be mapped to UMLs URIs by the MetaMap component, we utilized the DBpedia dataset as a URI representation. More precisely, we connected DBpedia Spotlight (Mendes et al., 2011), to the system pipeline. This open source library is able to disambiguate text fragments by means of their DBpedia representation (DBpedia=http://dbpedia.org/page/). To give an example:

- 'asthma' -> DBpedia:Asthma

3.4. Query Mapper

The Query Mapper component uses the concept and URI information from the preceding nodes, in order to construct a formal request pattern by means of either SparQL or MQL. This component uses a template-slot-based approach similar to (Unger et al., 2012). That is, the input representation is matched against a given formal query template representation, and subsequently populated with the appropriate slot values. The matching builds upon a rule-based decision tree, which uses a fuzzy match algorithm. More precisely, the system utilizes a predefined set of the synonym terms which are related to a specific predicate or entire query template. To give an example, the relation term 'cause', identified by the triplification component, is mapped onto the predicate rdf:type sider:SideEffect within the Linked Life Data sub-graph - so that a query template for the question "What drugs cause vomit?", would generate:

```sparql
SELECT distinct ?d ?n
WHERE {
  ?o sideEffect:umlS <SLOT-VALUE>.
  i.e. umls-concept:C0042963
  ?d <PREDICATE-VALUE> ?o.
  i.e sider-drugs:sideEffect
  ?d sider-drugs:drugName ?n.
}
```

with the predicate value sider:SideEffect for cure and the slot value umls-concept:C0042963 for vomit. As another example, the reference to symptoms will map the input question to the following MQL template:

```mql
["type": "/medicine/disease","id": null, "name": <SLOT-VALUE>, i.e "Meningitis" "symptoms": [{"id": null,"name": null}]
}
```

4. RDF-based Predicate Learning

One of the most important aspects in the automatic query construction process is in this context, the identification and disambiguation of relationship types - the actual predicates/property reference that connects the different URIs. While the current components (i.e. Spotlight and MetaMap) are not able to map these kinds of relations onto the ontology representation of the given KBs, we have implemented a new method to enhance the existing mapping functionalities. That is, at this stage of the pipeline, URI objects have been already identified, however, the predicate relationship (i.e. sider:SideEffect) was detected by using the rule-based approach - utilizing manually created gazetteers. The algorithm proposed follows the idea of DIPRE (Brin, 1999), by extracting patterns and relations from a reference corpus. The main steps of our algorithm are:

1. Property Selection: Select a number of properties for a given RDF-based KB to classify i.e. DBpedia with the property:
2. **Seed Generation**: In this step, we perform a formal SPARQL request on DBpedia, to get the set of objects and values bound to the requested property. Found entities are presented in the form of triples as \(?o \?p \?s\), where \(?p\) is the same for all of them, as for example:

"Roger Zelazny"  
dbprop:author  
"Blood of Amber"  

"Roger Zelazny"  
dbprop:author  
"The Hand of Oberon"

3. **Candidate Generation**: On the basis of the generated RDF seeds, a reference text corpus is processed, to find sentences with occurrences of both object (\(?o\)) and subject (\(?s\)). For our experiments, we have used the extended summary (dbpedia-owl:abstract) collection provided by DBpedia as the reference corpus. As result, we generate a number of candidates, such as:

Roger Zelazny wrote a Blood of Amber.

Roger Zelazny is author of The Hand of Oberon

4. **Candidate Transformation**: All occurrences of subjects and objects are converted into an abstract representation:

\((?s)\) wrote a \((?o)\).

\((?s)\) is author of \((?o)\)

The remaining term features are processed by the NLP component including stemming, POS tagging, named entity recognition.

5. **Classifier Training**: Given the feature representation of the candidate transformation, we applied the learning procedure. In the experiments, we have used SVM as the machine learning algorithm.

6. **Property Identification**: In the last step, we can use the multi-class classifier, to assign identified properties for given input question.

Who (wrote) Blood of Amber.  
\(\rightarrow\) dbprop:author

Who (is the author of) The Hand of Oberon  
\(\rightarrow\) dbprop:author

This algorithm allows the Linked Health Answer system, to identify formal property types with reference to a given knowledge repository.

### 4.1. Ontology connector

In the scope of the current work, we have evaluated several online knowledge bases as the sources of the structured information. For the medical context, we have chosen Linked Life Data (Momtchev et al., 2009). This RDF-repository is an aggregation of more than 25 popular bio-medical data source. The data from the 3rd party sources is merged and interconnected in Linked Life Data, most of the entities can be referenced by UMLS CUIs. These features make Linked Life Data to be a reasonable pick for the formal semantic queries in the medical context. As we wanted to show that the current system can be extended to other areas and to general context, we have added additional general purpose knowledge sources. We have added DBpedia (Auer et al., 2007), a semantic repository, as it is known as the core of Semantic Web and used as golden standard in evaluation process. Another rich open knowledge source is Freebase (Bollacker et al., 2008), it is known as the knowledge base for Google Knowledge Graph. Freebase uses MQL as formal request language. For each of the RDF repositories comprised, we have created a connector component. These components are taking the role of the request wrapper and executor.

### 4.2. Output provider

This component is the sink node of the pipeline. First it determines the type of the result, whether it is rich text answer or the list of short answers. Then it processes the outcome provided by the ontology connector nodes, deletes duplicates and merges the result. The output then is sent to the web user interface using AJAX server push.

### 5. Experiments

We have conducted two different experiments for the RDF-based predicate learning approach as describe above. The first goal was to create a precise relation classifier in the scope of normal statement sentences (StS). The second goal was to create a classifier that would be able to classify question sentences (QeS) with relation id in the scope of the question-answering system.

#### 5.1. Data Preparation

In order to train an accurate classifier, we had to obtain a reasonable amount of training data. The training data consists of the raw data itself plus the corresponding classes. In our case the raw data is represented in form of natural language sentences, each of which has a corresponding class in form of the predicate id (i.e. dbprop:author). We have used the QALD-3 (QALD dataset (Cimiano et al., 2013)) questions set to obtain the questions and the corresponding SPARQL request. We decided to skip the complex questions with more than one relation. In addition, we wanted the system to work with a single graph from the Linked Data, that is why SPARQL queries referring to the YAGO (Suchanek et al., 2007) ontology were skipped too. Each SPARQL request was executed on DBpedia, to get the set of objects and values bound to the requested property (see Seed Generation within previous section). On the basis of the RDF seeds, the extended abstracts of the DBpedia were used to identify sentences with occurrences of both object
and subject (*Candidate Generation*). Finally, all occurrences of subject and object are converted into the abstract representation (*Candidate Transformation*). As a result, we collected 18 different RDF-based predicates with 15,790 abstract candidate sentences, which were further processed by means feature construction and selection.

### 5.2. Features Selection

For the feature extraction task, we applied several NLP processing techniques, including sentence boundary detection, tokenization, lemmatization, POS-tagging. Additionally, we have applied named entity recognition (NER) technique that classifies terms as Person, Organization or Location and the dependency parse tree using *StanfordNLP*. During the tokenization, we allowed to construct n-grams of up to size 5, in order to detect frequent phrases within the training data set.

### 5.3. Learning Algorithm

For the classification experiments, we evaluated various ML algorithms on the basis of different input representations. Input data files were varying in size of number of classes and number of total features. By the number of the classes the data was divided into the categories: 18 classes, 10 classes and 5 classes. The number of the features was reduced by the slicing out the features that appear less then N times in the whole data set. For instance N=8 means that input data file contains no features with frequency less than 8. We have used Nave Bayes and the sequential minimal optimization (SMO) algorithm for the classification task. Both algorithms were trained using the 6 different learning data sets. The learning process was validated by usage of cross-validation with 10 folds.

### 5.4. Evaluation Analysis

The obtained results (see Table 5) of the evaluation process are demonstrating relatively high precision and the percentage of the correctly classified instances. The trend-line of the correctly classified instances percentage lies above 80% and hits the maximum at 95.50% (see Table 6) for the 5-classes-8-N classifier. The evaluation average accuracy is going down with the number of classes and scores 85.95% and 83.86% (Table 5) accuracy result for best classifiers in 10 classes and 18 classes respectively. The lowest average accuracy for SMO algorithm of 81.80% is obtained by the 18-classes-50-N classifier. Provided results can be used to define several observations. The first expected observation is that the percentage of the correctly predicted classes is rising up with the number of classes falling down. However, the difference in accuracy of the 18-classes classifier and 10-classes classifier is much smaller then between 10-classes and 5-classes. The same observation remains true for the weighted average recall and weighted average precision.

### 5.5. Question-based Predicate Evaluation

Our next goal was to adapt the generated classifier in the pipeline in order to improve our question answering system. Following this idea, we have created a test set of questions typical for the QA input. On this test data we have evaluated the obtained classifiers. Having the best obtained accuracy equals to 46% led us to the idea that question sentences are carrying less important information (which is used for classification) than statement sentences. For example, the statement sentence contains both SUBJECT and OBJECT term:

```
SUBJECT has written OBJECT in 1915 under the impression of the wars.
```

However, the question sentence contains only one of the both, OBJECT or SUBJECT:

```
Who wrote OBJECT?
```

Semantic and NLP information obtained from the question is quite different from the same types of information collected from the statement sentence. That is why we have created two classifier groups. Classifiers from the first group (created in previous steps) were optimized to find a relation id in the statement sentences text, while classifiers

| Number Predicates | ML Algo. | Avg. Accuracy |
|-------------------|----------|---------------|
| 18 classes 50 N   | SMO      | 81.80%        |
| 18 classes 8 N    | SMO      | 83.86%        |
| 10 classes 50 N   | SMO      | 83.95%        |
| 10 classes 8 N    | SMO      | 85.95%        |
| 5 classes 50 N    | SMO      | 94.06%        |
| 5 classes 8 N     | SMO      | 95.50%        |
| 18 classes 50 N   | nave Bayes | 36.74%       |
| 5 classes 50 N    | nave Bayes | 75.13%       |

Table 5: Results of RDF-based predicate learning by number of classes and features comprised using sentence statement-focused classifier.

| Precision | Recall | F-Measure | RDF predicate |
|-----------|--------|-----------|---------------|
| 0.879     | 0.880  | 0.879     | AuthorBook    |
| 0.848     | 0.722  | 0.780     | BirthPlace    |
| 0.799     | 0.820  | 0.809     | Country       |
| 0.833     | 0.730  | 0.778     | CountryLanguage |
| 0.889     | 0.951  | 0.919     | Developer     |
| 0.875     | 0.500  | 0.636     | Elevation     |
| 0.758     | 0.686  | 0.721     | Foundation    |
| 0.560     | 0.203  | 0.298     | Genre         |
| 0.952     | 0.784  | 0.860     | League        |
| 0.800     | 0.763  | 0.781     | Location      |
| 0.784     | 0.642  | 0.705     | Owner         |
| 0.011     | 0.011  | 0.011     | Producer      |
| 0.400     | 0.371  | 0.385     | ProducerFilm  |
| 0.750     | 0.293  | 0.421     | ProgrammerLanguage |
| 0.467     | 0.347  | 0.398     | PublisherSoft |
| 0.915     | 0.794  | 0.850     | Spouse        |
| 0.768     | 0.566  | 0.652     | Starring      |
| 0.632     | 0.462  | 0.534     | WebsiteAuthor |
| 0.831     | 0.839  | 0.832     | Weighted Avg. |

Table 6: Results of SMO utilizing 18 classes - 8 N by precision, recall and F1-measure using sentence statement-focused classifier.
from the second one were worked out to classify the relations in the question sentence. The observation led us to the intention to change the learning data in order to try to improve the exact question classification precision. Our intention however was to use the same data source for the classification on the QeS as for the classification on the StS. That is why the task was to find the most important features and slice off the features that cannot appear in the question statement. Following these conclusions, we have modified the training data, in the way that it will be more fit to the question-based predicate detection task. 1. We have processed the raw sentences and left only the text in between the OBJECT and the SUBJECT terms. 2. In the NLP preprocessing phase of the training data we have left only lemma and POS annotators.

Table 7: Results of RDF-based predicate learning task for sentence statements (StS) and question statements (QeS) by number of instances, features, and accuracy using question statement-focused classifier (5 classes)

| NLP pipeline | N | Algo. | Num. of instances | Num. of features | Precision | Recall | Avg. Accuracy |
|--------------|---|-------|-------------------|------------------|-----------|--------|---------------|
| StS          | 8 | SMO   | 11904             | 2895             | 0.377     | 0.462  | 46.15%        |
| QeS          | 20| SMO   | 8418              | 812              | 0.654     | 0.615  | 61.54%        |
| QeS          | 20| J48 (C4.5) | 8418         | 812              | 0.769     | 0.692  | 69.23%        |
| QeS          | 8 | SMO   | 8418              | 1795             | 0.885     | 0.769  | 76.92%        |
| QeS          | 8 | J48 (C4.5) | 8418         | 1795             | 0.744     | 0.692  | 69.23%        |
| QeS          | 2 | SMO   | 8418              | 5520             | 0.891     | 0.846  | 84.61%        |
| QeS          | 2 | J48 (C4.5) | 8418         | 5520             | 0.744     | 0.692  | 69.23%        |

Figure 2: Comparison of the 5 classes formal relation classifiers in statement and question sentences.

5.6. Question-based Predicate Result

During the validation process we have obtained a set of the results, which allowed us to make several assumptions about the nature of relation classification in question statements (QeS). The most important fact from our point of view is that the original 5 classes classifier, which has shown the best accuracy of 94.06%. In the task of relation identification on statements sentences (StS) (see Table 7), has shown only 46.15% average accuracy on the questions evaluation data set. These results led us to re-design of NLP-pipeline and data-preprocessing in order to prepare another set of features from the original data ‘seeds’, this on our assumption could improve overall accuracy of the relation identification over question statements. The classifiers that were trained on this enhanced data showed indeed better results, with the best result equal to 84.61% for 5 classes (see Table 7). The diversity of the results for relation identification task over QeS is related to the N-value (feature minimum frequency) and to variance of applied ML algorithms (Table 2).

6. Conclusion

In this paper we presented an analysis of existing methods, frameworks and software libraries that can be used in the creation of a Linked Data-driven question answering system. After the overview of the existing approaches, we have proposed the architecture of the Linked Health Answer system. The system was built to target specifically the medical context. Our work has additionally targeted the problem of semantic property identification. The proposed machine learning-based algorithm can be used to enhance the transformation of natural language questions into their respective semantic query request representation.

7. References

Adolphs, Peter, Theobald, Martin, Schafer, Ulrich, Uszkoreit, Hans, and Weikum, Gerhard. (2011). Yago qa: Answering questions by structured knowledge queries. In Proceedings of the 2011 IEEE Fifth International Conference on Semantic Computing, ICSC ‘11, pages 158–161, Washington, DC, USA. IEEE Computer Society.

Aronson, A. R. (2001). Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program. Proceedings / AMIA ... Annual Symposium. AMIA Symposium, pages 17–21.

Auer, Sren, Bizer, Christian, Kobilarov, Georgi, Lehmann, Jens, and Ives, Zachary. (2007). Dbpedia: A nucleus for a web of open data. In In 6th Intl Semantic Web Conference, Busan, Korea, pages 11–15. Springer.

Bär, Daniel, Zesch, Torsten, and Gurevych, Iryna. (2013). Dkpro similarity: An open source framework for text similarity. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (System Demonstrations) (ACL 2013), pages 121–126, Stroudsburg, PA, USA, August. Association for Computational Linguistics.

Bollacker, Kurt, Evans, Colin, Paritosh, Praveen, Sturge, Tim, and Taylor, Jamie. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In SIGMOD ’08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pages 1247–1250, New York, NY, USA. ACM.

Brin, Sergey. (1999). Extracting patterns and relations from the world wide web. In Selected papers from the International Workshop on The World Wide Web and Databases, WebDB ’98, pages 172–183, London, UK, UK. Springer-Verlag.
