UMinho | 2021

Renato Preigschadt de Azevedo **DSL based Automatic Generation of Q&A Systems**

Universidade do Minho
Escola de Engenharia

Renato Preigschadt de Azevedo

DSL based Automatic Generation of Q&A Systems

July, 2021

Universidade do Minho
Escola de Engenharia

Renato Preigschadt de Azevedo

DSL based Automatic Generation of Q&A Systems

Doctorate Thesis
Doctoral Program in Informatics

Work developed under the supervision of:
Pedro Rangel Henriques
Maria João Varanda

July, 2021

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisors Prof. Pedro Rangel Henriques and Prof. Maria João Varanda Pereira, for the continuous support of my Ph.D. study and related research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having better advisors and mentors for my Ph.D. study. I would also like to thank the friendship and support, which were fundamental to making it less challenging to be away from all the family and friends in Brazil. I consider myself a very lucky person to have known and learned so much from you both during these years.

Besides my advisor, I would like to thank the rest of my pre-thesis committee: Prof. Nuno Oliveira, Prof. José João, and Prof. Paulo Novais, for their insightful comments and encouragement.

Most importantly, none of this could have happened without my family. My mother, Maria Augusta, and my father Ricardo, who had offered their encouragement and support (emotional and financial ;) through phone calls and thoughts – despite my limited devotion to correspondence. My brother Ricardo Junior, who, besides the support and my unconditional admiration, also made my absence less felt by my parents and nephews. I also would like to thanks my godson Bruno for the understanding of my long absence. Despite the distance, it is a great pride to be part of your achievements by dedication to your studies.

I am very grateful to Isabela, who was my girlfriend during the start of this Ph.D. and became my wife at the end of 2019. Thank you for understanding me very well. Thank you for always being on my side, even when we lived on different continents. Thank you also for share my passion for research and teach. I love you.

A very special gratitude goes out to all down at Colégio Técnico Industrial (CTISM) and the Federal University of Santa Maria for helping and allowing me to leave Brazil for the doctorate in Portugal. A very special thanks to professor Marcelo Freitas, who encouraged and helped me to get permission to leave the CTISM for four years.

With a special mention to all my colleagues from the computer networks course, who also allowed me to be out of my university for four years. It was fantastic to have the opportunity to do the Ph.D. outside Brazil!

I am also grateful to the following University of Minho staff: Helena, Conceição, Goretti, Carla, Aragão for their unfailing support and assistance during this Ph.D. research.

Thanks to everyone in the gEPL, who is an excellent research laboratory with all of you during the last four years.

In November 2018, I went to Universidad de San Luis - Argentina (UNSL) for several weeks to work with professor Mário Beron. My time at UNSL has been highly productive and working with the professor was an extraordinary experience. I learned a lot about language processing and security with the professors and the students from UNSL. But one of the most significant benefits of being in Argentina was the friendship acquired with professor Mario. Mario, you are the most obstinate worker that I have worked with.

I want to thank also all my friends from Brazil who supported me in several ways, with a barbecue when I visited them, or some random discussions about any kind of subject. A big thanks to the friends from Quarta Gelada who had been by my side during my academic life. A special thanks to the founder of this group Giovani for putting professor Pedro Henriques on my academic and life path. I will always be grateful for this. I have to also thank Rafael Pereira for receiving me the first time in Braga, on which occasion I had met in person professor Pedro Henriques.

Getting through my dissertation required more than academic support, and I have many, many people to thank for listening to and, at times, having to tolerate me over the past four years. I cannot begin to express my gratitude and appreciation for their friendship. All the friends that the Padel made me, you guys were fundamental to maintain clear-mindedness. It would be a lot more challenging to be away from family and friends without you guys.

Also, I would like to thank my long-time friend Ricardo Martini, who accepted me as a roommate since my arrival in Braga. It was a terrific two years and some months of sharing a house and so many triumphs with you. Especially the trophy from the Galaico Durienses padel games. Also have to thank my godsons Leandro and Greice, who shared so many happy moments in Braga, and many trips around Europe. I have also to thank Bruno, Karine, and Walter (and family) for being so present in my life, even that they lived in Lisbon.

I had the luck to know and have the friendship of my professors in Padel Carlos and José Carvalho. You guys were so important for being there during these four years living in Portugal. Thank both of you for crossing the life paths with me.

I have to also thanks professors Victor, Paulo, Solange for all the knowledge and partnership in the area of computer networks and security. I am a very fortunate person to meet you all.

I would also thank the friends from my hometown that I knew in Braga. Luis, Maiara, Ricardo (Alemão), and Tati, it was a pleasure to live with you guys in Braga. I also thank you, Lázaro, for being understanding and teaching me how to brew my own beer, besides so many other teachings.

I probably forgot a lot of special people that were so important in this walk, but I can not forget to thank João Marco and Alice. During my first stay in Braga (years before thinking of doing a Ph.D. there), I had the luck to know both of you. Since then, you were outstanding friends and allowed the quarantine to be a little bit easier for Isabela and me. Thank you, João Marco, for all the scientific, technical, and random discussions over a coffee, beer, party, or any place. I gained not only a few pounds of weight in Braga, but I also gained a new brother.

Thanks for all your encouragement!

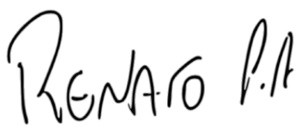
STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the Universidade do Minho.

Braga

(Place)



(Renato Preigschadt de Azevedo)

To my parents, who made this achievement possible.

Resumo

Geração automática de Sistemas de Perguntas e Respostas baseado em DSL

Para auxiliar o usuário na busca por informações relevantes, os sistemas de Perguntas e Respostas (Q&A – Question and Answering) oferecem a possibilidade de formular perguntas através de linguagem natural, obtendo respostas mais adequadas e concisas. Esses sistemas interpretam a pergunta do usuário para entender suas necessidades de informação e retornam as respostas mais adequadas em um sentido semântico; eles não realizam uma pesquisa estatística por palavras chaves, como acontece nos motores de busca existentes. Existem várias abordagens para desenvolver e implantar sistemas de Q&A, tornando difícil escolher a melhor maneira de construir o sistema. O desenvolvedor deve escolher linguagens e técnicas que permitam o processamento de linguagem natural. Também é necessário fornecer uma interface de usuário, permitindo que os usuários dos sistemas de Q&A possam fazer perguntas e obter respostas. Para tornar mais fácil a construção e implantação de sistemas de Q&A, uma linguagem de domínio específico para gerar sistemas de Q&A (AcQA) é proposta nesta tese. A linguagem AcQA permite que os desenvolvedores de sistemas de Q&A se concentrem nos dados que serão utilizados para construir a base de conhecimento e no conteúdo do sistema, em vez dos detalhes de implementação. A linguagem proposta gera código e permite uma implantação completa do sistema de Q&A em um servidor. Um experimento é conduzido para avaliar a viabilidade de usar a linguagem AcQA. O estudo foi realizado principalmente com pessoas da área de informática e mostra que a linguagem AcQA simplifica o desenvolvimento de um sistema de Q&A.

Palavras-chave: linguagens de domínio específico, sistemas de perguntas e respostas, AcQA, geração de código, processamento de linguagem

Abstract

DSL based Automatic Generation of Q&A Systems

In order to help the user to search for relevant information, Question and Answering (Q&A) Systems provide the possibility to formulate the question freely in a natural language, retrieving the most appropriate and concise answers. These systems interpret the user question to understand his information needs and return him the more adequate replies in a semantic sense; they do not perform a statistical word search like happens in the existing search engines. There are several approaches to develop and deploy Q&A Systems, making it hard to choose the best way to build the system. The developer has to choose languages and techniques that allow natural language processing. It is also necessary to provide a user interface where the final users can ask questions and get answers. To turn easier the construction and deployment of Q&A Systems, a way to automatically create Q&A Systems based on a DSL (AcQA) is proposed in this Ph.D. thesis, thus allowing the setup and the validation of the Q&A System independent of the implementation techniques. The proposed AcQA language allows the developers of Q&A Systems to focus on the data and contents instead of implementation details. The proposed language generates code and can do a full deployment of the Q&A System into a destination server. An experiment is conducted to assess the feasibility of using AcQA. The study was carried out with people mainly from the computer science field and shows that the AcQA language simplifies the development of a Q&A System.

Keywords: domain-specific languages, question & answer systems, AcQA, code generation, language processing

Contents

List of Figures

List of Tables

1

Introduction

The increase in the processing power and the availability of information gave rise to Q&A Systems and consequently augmented the need for such systems. Q&A Systems dialog with the end-user in a more natural way, accepting questions formulated in natural language and providing more accurate answers when compared with the traditional search engines.

With the advent of smartphones with personal assistants, which allow the user to ask questions and get answers from various subjects, these systems are being used by a large number of people. Approximately forty or fifty years ago, the study about Q&A Systems began (Bobrow 1964; Fortnow and Homer 2003; Plath 1976; Waltz 1978), but because of computational limitations, these systems had limited scope. Some Q&A Systems were more or less successful; some were discontinued, demonstrating the difficulty of building and maintaining a system capable of understanding natural language queries as humans can do and provide the appropriate answers. Recently more efficient systems have appeared featuring real applicability. However, to improve these tools, more research is necessary.

Questions are asked and answered several times per day by a human. Q&A Systems tries to do the same level of interaction between computers and humans. This approach differs from standard search engines (such as Google[1], Bing[2].) because it makes an effort to understand the intention that the question expresses and try to give concise answers instead of using only keywords from the question asked and provide documents as results.

[1] https://www.google.com

[2] https://www.bing.com

Unlike standard search engines that retrieve documents based on keywords, Q&A Systems aims to recognize a high-level natural language in the input text. The high-level natural language understanding of the user question enables the construction of concise answers instead of a set of possibly related documents.

A simple Q&A System is composed of several processes: question analysis, query processing, and answer formulation (Clark, Fox, and Lappin 2010). Question analysis is done by analyzing the user's input text to extract its meaning; it can be implemented with Natural Language Processing (NLP) techniques. The query processing phase aims at recovering the information necessary to answer the question from relevant documents or Knowledge Base (KB); information retrieval techniques or knowledge base querying can be applied. In the third phase, using the collected information, a list of answer candidates is built, and the elements are ranked according to the probability to satisfy the user's needs.

To be able to create a successful question and answering system, all these processes have to be carefully specified by the domain specialist and implemented by the programmer. The programmer has to be an expert in the chosen programming language. He also needs to master the various libraries required to implement the system (Natural Language Processing, Knowledge Processing, Inference Mechanism, Database, or Triple Storage Access). The complexity of such systems components makes the implementation process complex and error-prone; it is indeed a time-consuming and costly task.

The use of an approach based on formal Specifications written in Domain-specific Languages (DSLs) can simplify and accelerate applications' development. The design of a specific language to support the development of a Q&A System allows the user to specify its components more abstractly and concisely, avoiding implementation details. This approach makes the process of implementing the system more straightforward and less error-prone. To the best of our knowledge, this is the first work that uses a DSL to create Q&A Systems. We did not identify any similar work to compare and discuss.

1.1 Motivation

Considering the present importance of Q&A Systems and the many complex tasks that must be implemented to create such an intelligent tool, the motivation for this thesis, proposed along with this document, is to make the development of Q&A Systems easier following a systematic and rigorous approach. The success of generative approaches to programming and the know-how of the research team under which this thesis takes place also motivates the search for a solution in the area of language processing and automatic generation of software.

1.2 Objectives

This thesis's main objective is to create a language that allows building closed domain Q&A Systems automatically from their formal specification. In order to attain that objective, the following specific objectives must be achieved:

- Choose a generic architecture (among the existing ones or defining a new one) that can always be adopted to build a closed domain Q&A System;
- Identify what components are stable in order to understand which information needs to be specified in each concrete case;
- Define a DSL that allows an end-user to specify the issues that need to be described to build a specific system;
- Develop a system that analyzes descriptions written in that DSL and resorting to standard components generate the desired Q&A System;
- Validate with concrete case studies the approach proposed and the developed engine to process the AcQA language.

1.3 Research Hypothesis

The purpose of this thesis is to prove that it is possible, effective, and advantageous to automatically create Q&A Systems, based on their abstract and rigorous specification formally written in a DSL that allows the end-user to focus on its content and the related knowledge, independently of technical details.

1.4 Thesis Organization

The structure of this document is as follows: Q&A Systems are discussed in Chapter 2; the concepts of DSL are revised in Chapter 3; the AcQA language is presented in Chapter 4. In Chapter 5, the AcQA engine and a discussion of concrete Q&A System that is used as a technological basis for the AcQA language are presented; Three case studies are described and deployed in Chapter 6; Chapter 7, discuss the experiment and results, and finally, in Chapter 8 conclusions, the project schedule, and working methodology are presented.

Figure 1 presents a roadmap with the recommended paths that can be taken to read this thesis.

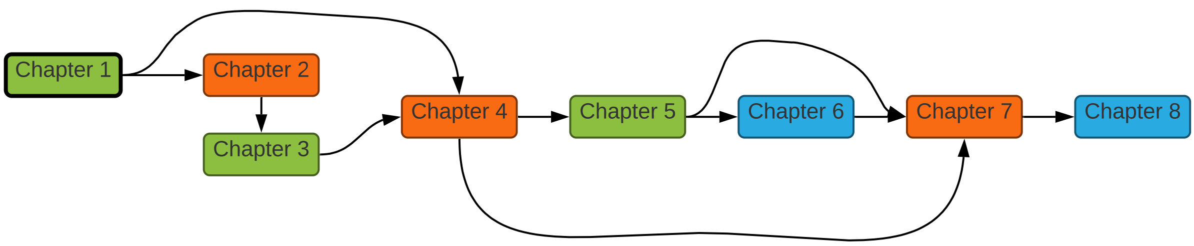


Figure 1: Recommended paths to read this thesis

2

Question Answering Systems - State of the Art

This chapter presents an overview of Q&A Systems along with a broad review of the state of the art.

Q&A Systems provide a way to process natural language inputs from a user extracting their meaning and providing concrete and concise answers. These systems allow users to make questions more naturally and get concise and straightforward answers, thus decreasing the effort necessary to find the correct answer.

Structure of the chapter. Section 2.1 describe a generic Q&A System. In Section 2.3 is presented a review of the literature in the Q&A System field; and Section 2.2 presents a classification of Q&A Systems with the works described in Section 2.2.

2.1 Generic Q&A System

The wideness of information available associated with the demand for direct answers from the users requires a different approach from standard search engines. The use of natural language to communicate with computer systems turns information technology useful for all persons, allowing them to specify the information needed deeply. Q&A Systems are not new, but they still need to be improved in terms of answers accuracy and terms of knowledge domains. The main idea of these systems is to receive the user question and analyze the keywords and the intention. To discover the intent within a question is not an easy task. Besides the capability of understanding the user question, the system must retrieve the best possible answers, ranking them.

There are several approaches in the literature explaining the construction of Q&A Systems (Ferrucci 2010; Kaisser and Becker 2004; Sasikumar and Sindhu 2014; Vargas-Vera and Lytras 2010), explaining the typical sequence of development stages. At first, several technical approaches should be carefully studied to allow the processing of natural language. In the past, small Knowledges Bases were used, allowing the construction of simple Q&A Systems. These systems used simple schemas with small entities and relations, ad-hoc approaches (manually constructed rules), among other strategies to create the knowledge base (KB). The KB was specifically tailored to the specific domain, requiring much effort from the Q&A System's original designers to add new content or add a new domain. These strategies do not support

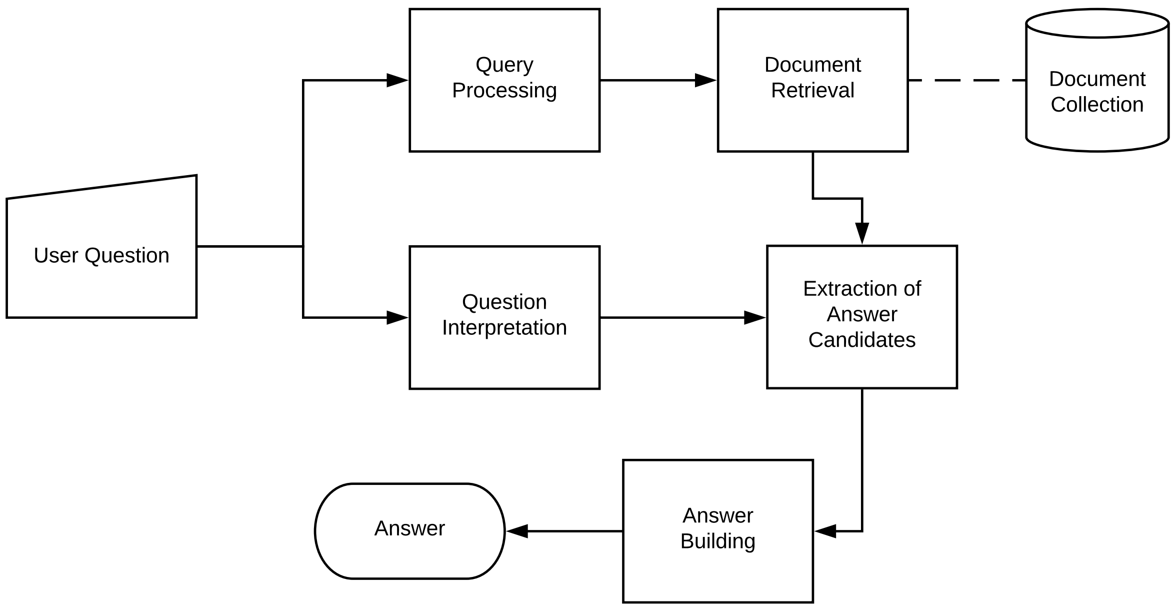


Figure 2: Generic architecture of a Q&A Systems.

scalable systems and turn complex the development of open domain systems. Currently, there are several approaches to the question analysis such as Query Graphs, Topic Entity Linking, Relation Matching using Deep Convolutional Neural Network. Section 2.3 presents papers describing these new approaches.

To construct a Q&A System, we need to develop three processes: question analysis, query processing (extraction of potential answers), and answer formulation. Figure 2 present the main processes needed to a Q&A System accordingly to work in (Pasca 2007). The techniques used for question analysis seek to recover meaning from the input text, sometimes employing Natural Language Processing (NLP) to achieve the goal. Natural Language Processing is an area of computation that includes parsing, part-of-speech (POS) tagging, statistical models, ontologies, and machine learning (A. Jain, Kulkarni, and Shah 2018). Pattern matching and the use of tags can also be used to process the input text. Query processing approaches are responsible for handling the input text to create the queries necessary to extract relevant information from the KB. The answer formulation uses information gathered in question analysis and query processing to generate or retrieve possible answers.

The result of a Q&A System can be fragments of documents, a list of links to web pages, images, a simple and concise sentence, or a ranking of sentences. A basic Q&A System have to process the user's input questions and respond with an answer or a rank-ordered list of candidates' answers.

As examples of Q&A Systems through natural language processing, we have: WolframAlpha: a mathematical Q&A System (Inc., Wolfram Research 2018), which offers knowledge by analyzing the collection of information it possesses in its local database; IBM Watson: A system that was initially used to answer generic questions from the American TV show Jeopardy! (Ferrucci 2010; Packowski, Sarah and Lakhana

2017), but today it is used in several domains.

The techniques necessary to build answers rely on the methods used in query processing and question analysis. They usually use fragments of documents and sentences to define the most appropriate answers and present them to the user. Also, a succinct answer approach can be used, where the technique tries to present a concise answer.

2.2 Classification

These systems are usually classified according to the kind of domain they are able to deal with, being of two types: closed domain or open domain. In this work, we use a broader classification based on the work of (Mishra and S. K. Jain 2016).

Figure 3 presents in the form of an ontology the classification used by this work. The ontology shows the classification proposed by (Mishra and S. K. Jain 2016). Q&A Systems can be classified by the kind of domain, type of questions, analysis type, data source characteristics, type of matching functions, data source types, type of answers generated, and techniques. An interactive version of this image with the explanation is available at [1].

The classification proposed is explained in the following Subsections.

2.2.1 Domain

The domain can be classified as a general or restricted domain. Table 1 classifies the works presented in Section 2.3 according to the domain classification.

- General (or open domain): General domain Question Answering Systems aims to answer anything that the user asks. The questions are domain-independent. This type of Q&A System works with a large repository of information to answer questions about all kinds of subjects. This type of system is supported by large sets of information and generic ontologies.

- Restricted: Restricted domain Question Answering Systems works only with a specific domain, not answering questions outside the proposed field. The information repository is made of data only related to the area, being able to achieve better accuracy than general domain Q&A System. The restricted domain is also known as a closed-domain system, usually based on well-defined, structured databases and ontologies. These systems are limited to the particular domain implemented.

[1] https://acqa.di.uminho.pt/classification

Table 1: Domain classification

	Domain	
System	General	Restricted
Intelligent QAS based on Artificial Neural Network	Ansari, Maknojia, and Shaikh 2016	
R-NET	Etworks 2017	
AskHERMES		Y. G. Cao et al. 2011
FastQA	Weissenborn, Wiese, and Seiffe 2017	
QAS on Education Acts	Lende and Raghuwanshi 2016	
MEANS		Ben Abacha and Zweigenbaum 2015
PythonQA		Ramos, Pereira, and Henriques 2017
WAD	Jayalakshmi and Sheshasaayee 2017	
Dynamic QAS based on Ontology	Rajendran and Sharon 2017	
Ontological-semantic text analysis and the QAS	Mochalova et al. 2015	
Ontology-driven Visual QA Framework	Besbes, Baazaoui-Zghal, and Ghezela 2015	
QAAL		Kalaivani and Duraiswamy 2012
From Captions to Visual Concepts and Back	Fang et al. 2015	
WolframAlpha	Inc., Wolfram Research 2018	
IBM Watson	Ferrucci 2010	
Verb Focused Answering from CORD-19		George 2020

2.2.2 Type of Questions

Type of questions can be classified as a Confirmation, Causal, Factoid, Hypothetical, or List. Table 2 classifies the works presented in Section 2.3 by type of questions. Note that only categories with works are presented in Table 2.

- Factoid: These questions are usually started with What, When, Which, Who, How. All these questions are simple and based on facts that are present in the information repository. Some examples of these types of questions are: 'Who won the trophy for best actor in the 2017 Oscar ceremony?', 'How to declare a String in Python programming language?'

- List: Answer with a list of facts to fulfill the user question is of the List type. For example, these answers can contain a list of pilots to answer 'Which people won the Formula 1 championship?', or a list of students to correctly answer 'Which students have a grade greater than B+?'.

- Hypothetical: Questions asked for information related to any hypothetical scenario. Usually, these questions begin with 'what would happen if' (Kolomiyets, 2011).

- Causal: These types of questions require an explanation about an entity, not only an answer with an entity. An example is 'Why should I use Python instead of Java to write my app?'. These questions require advanced natural language processing techniques to be able to construct an answer for the user. Questions usually start with why or how.

- Confirmation: When the user seeks to get an answer in the form of yes or no from the Q&A System these types of questions require an inference mechanism to extract a yes or no answer from the information repository.

Table 2: Type of Questions classification

	Type of Questions		
System	Factoid	List	Confirmation
Intelligent QAS based on Artificial Neural Network	Ansari, Maknojia, and Shaikh 2016		
R-NET	Etworks 2017		
AskHERMES	Y. G. Cao et al. 2011		
FastQA	Weissenborn, Wiese, and Seiffe 2017	Weissenborn, Wiese, and Seiffe 2017	
QAS on Education Acts	Lende and Raghuwanshi 2016		
MEANS	Ben Abacha and Zweigenbaum 2015		Weissenborn, Wiese, and Seiffe 2017
PythonQA	Ramos, Pereira, and Henriques 2017		
WAD	Jayalakshmi and Sheshasaayee 2017		
Dynamic QAS based on Ontology	Rajendran and Sharon 2017		
Ontological-semantic text analysis and the QAS	Mochalova et al. 2015		
Ontology-driven Visual QA Framework	Besbes, Baazaoui-Zghal, and Ghezela 2015		
QAAL	Kalaivani and Duraiswamy 2012		
From Captions to Visual Concepts and Back	Fang et al. 2015		
WolframAlpha	Inc., Wolfram Research 2018	Inc., Wolfram Research 2018	
IBM Watson	Ferrucci 2010	Ferrucci 2010	
Verb Focused Answering from CORD-19	George 2020		

2.2.3 Analysis Type

Analysis type can be divided into the following types: Morphological, Syntactical, Semantic, Expected Answer type, and Focus Recognition. The works presented in Section 2.3 are arranged according to this classification in Table 3. Only categories with works are presented in Table 3.

- Morphological: This analysis aims at separating words into individual morphemes and assigns a class to them (Mishra and S. K. Jain 2016). The type of analysis uses stemming and lemmatization of words to do a morphological analysis of the text.

- Syntactical: The syntactical analysis identifies the grammatical construction of words to process the questions. The analysis is done by looking at the class of words (noun, verbs, adjectives, or adverbs), not only the words, to try to find the answer candidates.

- Semantic: It aims to deduce the possible meaning of the input question based on the words used by the user. The semantic analysis usually works with the parse tree generated by the syntactical analysis, interpreting the possible meaning based on the tree.

- Expected Answer type analysis: This analysis tries to identify the answer type required in the answer based on the question category. In the sentence 'Who played Captain Jack Sparrow in the movie Pirates of the Caribean?', the expected answer type is a person. With this information, factoid and list type questions can be answered with the correct answer type.

- Focus Recognition: Tries to identify the focus of the question to be able to give better answers to the user. For example, if a user asks, 'I want a new computer with good processing power and lightweight. Should I buy a mac?', the answer needs to be a yes or no, possible to explain why to choose that answer.

Table 3: Classification of analysis types

	Analysis Type			
System	Morphological	Syntactical	Semantic	Expected Answer type
Intelligent QAS based on Artificial Neural Network	Ansari, Maknojia, and Shaikh 2016	Ansari, Maknojia, and Shaikh 2016		
AskHERMES		Y. G. Cao et al. 2011	Y. G. Cao et al. 2011	
FastQA		Weissenborn, Wiese, and Seiffe 2017		
QAS on Education Acts		Lende and Raghuwanshi 2016		
MEANS		Weissenborn, Wiese, and Seiffe 2017		Weissenborn, Wiese, and Seiffe 2017
PythonQA		Ramos, Pereira, and Henriques 2017		
WAD		Jayalakshmi and Sheshasaayee 2017		Jayalakshmi and Sheshasaayee 2017
Dynamic QAS based on Ontology		Rajendran and Sharon 2017		
Ontological-semantic text analysis and the QAS		Besbes, Baazaoui-Zghal, and Ghezela 2015		
Ontology-driven Visual QA Framework		Kalaivani and Duraiswamy 2012		
IBM Watson		Ferrucci 2010	Ferrucci 2010	

2.2.4 Data Source Type

The data source can be of a Structured, Semi-Structured, and Unstructured type. Table 4 classifies the works presented in Section 2.3 by data source type.

- Structured data source: This type of data source has only structured data in the knowledge base. The documents have a well-defined schema to describe all the data. With this type of data, the information extraction is straightforward since the data is produced according to the schema.

- Semi-Structured data source: There are no guarantees in this kind of data source that all stored data complies with the schema.

- Un-structured data source: The data can be of any type. The data does not have a schema or even the same format.

Table 4: Classification of data source type

	Data source type		
System	Structured	Semi-Structured	Unstructured
Intelligent QAS based on Artificial Neural Network	Ansari, Maknojia, and Shaikh 2016		
R-NET	Etworks 2017		
AskHERMES		Y. G. Cao et al. 2011	
FastQA		Weissenborn, Wiese, and Seiffe 2017	
QAS on Education Acts		Lende and Raghuwanshi 2016	
MEANS	Ben Abacha and Zweigenbaum 2015		
PythonQA	Ramos, Pereira, and Henriques 2017		
WAD		Jayalakshmi and Sheshasaayee 2017	
Dynamic QAS based on Ontology	Rajendran and Sharon 2017		
Ontological-semantic text analysis and the QAS	Mochalova et al. 2015		
Ontology-driven Visual QA Framework	Besbes, Baazaoui-Zghal, and Ghezela 2015		
QAAL		Kalaivani and Duraiswamy 2012	
From Captions to Visual Concepts and Back			Fang et al. 2015
WolframAlpha			Inc., Wolfram Research 2018
IBM Watson			Ferrucci 2010

2.2.5 Data Source Characteristic

The data source can be classified by several characteristics: Source Size, Language, Heterogeneity, Genre, and Media. This classification and the related works are presented in Table 5.

- Source Size: The size of the source data available to the QA System.

- Language: Classifies the Q&A System by the language that it addresses. If the language that the QA System recognizes is comprised of a unique language, it is a single language Q&A System or several languages (multilingual).

- Heterogeneity: This classification concerns the data sources, i.e., whether the data comes from a single data source or several sources.

- Genre: This item classifies the type of language used in data sources (and questions). Formal is when the text is written linguistically correct, and informal when written in a non-correctly way.

- Media: If the data available comprises text or another multimedia data format, such as audio, video.

Table 5: Classification of Data Source Characteristics

	Data Source Characteristics			
System	Language	Heterogeneity	Genre	Media
Intelligent QAS based on Artificial Neural Network	Single Ansari, Maknojia, and Shaikh 2016	Multiple Src Ansari, Maknojia, and Shaikh 2016	Formal Ansari, Maknojia, and Shaikh 2016	Text Ansari, Maknojia, and Shaikh 2016
R-NET	Single Etworks 2017	Multiple Src Etworks 2017	Formal Etworks 2017	Text Etworks 2017
AskHERMES	Single Y. G. Cao et al. 2011	Multiple Src Y. G. Cao et al. 2011	Formal Y. G. Cao et al. 2011	Text Y. G. Cao et al. 2011
FastQA	Single Weissenborn, Wiese, and Seiffe 2017	Multiple Src Weissenborn, Wiese, and Seiffe 2017	Formal Weissenborn, Wiese, and Seiffe 2017	Text Weissenborn, Wiese, and Seiffe 2017
QAS on Education Acts	Single Lende and Raghuwanshi 2016	Unique Src Lende and Raghuwanshi 2016	Formal Lende and Raghuwanshi 2016	Text Lende and Raghuwanshi 2016
MEANS	Single Ben Abacha and Zweigenbaum 2015	Multiple Src Ben Abacha and Zweigenbaum 2015	Formal Ben Abacha and Zweigenbaum 2015	Text Ben Abacha and Zweigenbaum 2015
PythonQA	Single Ramos, Pereira, and Henriques 2017	Unique Src Ramos, Pereira, and Henriques 2017	Formal Ramos, Pereira, and Henriques 2017	Text Ramos, Pereira, and Henriques 2017
WAD	Single Jayalakshmi and Sheshasaayee 2017	Multiple Src Jayalakshmi and Sheshasaayee 2017	Formal Jayalakshmi and Sheshasaayee 2017	Text Jayalakshmi and Sheshasaayee 2017
Dynamic QAS based on Ontology	Single Rajendran and Sharon 2017	Unique Src Rajendran and Sharon 2017	Formal Rajendran and Sharon 2017	Text Rajendran and Sharon 2017
Ontological-semantic text analysis and the QAS	Single Mochalova et al. 2015	Unique Src Mochalova et al. 2015	Formal Mochalova et al. 2015	Text Mochalova et al. 2015
Ontology-driven Visual QA Framework	Single Besbes, Baazaoui-Zghal, and Ghezela 2015	Unique Src Besbes, Baazaoui-Zghal, and Ghezela 2015	Formal Besbes, Baazaoui-Zghal, and Ghezela 2015	Text Besbes, Baazaoui-Zghal, and Ghezela 2015
QAAL	Single Kalaivani and Duraiswamy 2012	Multiple Src Kalaivani and Duraiswamy 2012	Formal Kalaivani and Duraiswamy 2012	Text Kalaivani and Duraiswamy 2012
From Captions to Visual Concepts and Back	Multilingual Fang et al. 2015	Multiple Src Fang et al. 2015	Formal Fang et al. 2015	Image Fang et al. 2015
WolframAlpha	Single Inc., Wolfram Research 2018	Single Src Inc., Wolfram Research 2018	Formal Inc., Wolfram Research 2018	Text Inc., Wolfram Research 2018
IBM Watson	Multilingual Ferrucci 2010	Multiple Src Ferrucci 2010	Formal Ferrucci 2010	Text Ferrucci 2010

2.2.6 Type of Matching Functions

Q&A Systems uses different strategies to match the information in the knowledge base. Some strategies are Set-Theoretic Models, Standard Boolean Model, Algebraic Model, Probability Model, and Feature-Based Models. Table 6 classifies the works presented in Section 2.3 by matching functions type.

- Set-Theoretic Models: This type of matching function uses documents as sets of words or phrases.
- Standard Boolean Model: Use the Boolean model from Information Retrieval to extract answers. Easy to implement but deliver responsibility to the user to write using boolean notation.
- Algebraic Model: This model represents documents and user questions as vectors, allowing a scalar value in the matching function.
- Probability Model: Use probability relevance to classify documents and questions.
- Feature-Based Models: This matching function defines documents as vectors containing the weight value of features to generate a relevance score.

Table 6: Classification of Matching Functions

	Type of Matching Functions		
System	Algebraic Model	Probability Model	Feature-Based
AskHERMES	Y. G. Cao et al. 2011		
QAS on Education Acts	Lende and Raghuwanshi 2016		
MEANS			Ben Abacha and Zweigenbaum 2015
PythonQA		Ramos, Pereira, and Henriques 2017	
WAD		Jayalakshmi and Sheshasaayee 2017	
Dynamic QAS based on Ontology	Rajendran and Sharon 2017		
Ontology-driven Visual QA Framework	Besbes, Baazaoui-Zghal, and Ghezela 2015		
QAAL	Kalaivani and Duraiswamy 2012		
From Captions to Visual Concepts and Back		Fang et al. 2015	

2.2.7 Techniques

Q&A Systems can be classified by the techniques used to extract information and create an answer. These techniques can be Data Mining, Information Retrieval, NLP, and Knowledge Retrieval. Techniques types are classified in Table 7 according to works discussed in Section 2.3.

- Data Mining: Use Data Mining techniques to extract relevant documents to extract answer candidates.

- Information Retrieval: Use information retrieval and NLP techniques to query a large knowledge source, such as the web, to extract possible candidates' answers.

- NLP: Process the answers and the knowledge base with Natural Language Processing and Understanding techniques to seek information that could be subjective or fact-based.

- Knowledge Retrieval: To be able to understand knowledge, this approach uses NLP, Knowledge Acquisition, and data mining techniques to retrieve useful and correct answers.

Table 7: Techniques classification

	Techniques		
System	Information Retrieval	NLP	Knowledge Retrieval
Intelligent QAS based on Artificial Neural Network			Ansari, Maknojia, and Shaikh 2016
AskHERMES	Y. G. Cao et al. 2011	Y. G. Cao et al. 2011	
QAS on Education Acts		Lende and Raghuwanshi 2016	
MEANS		Ben Abacha and Zweigenbaum 2015	
PythonQA		Ramos, Pereira, and Henriques 2017	
WAD	Jayalakshmi and Sheshasaayee 2017	Jayalakshmi and Sheshasaayee 2017	
Dynamic QAS based on Ontology		Rajendran and Sharon 2017	
Ontological-semantic text analysis and the QAS		Mochalova et al. 2015	
Ontology-driven Visual QA Framework		Besbes, Baazaoui-Zghal, and Ghezela 2015	
QAAL		Kalaivani and Duraiswamy 2012	Kalaivani and Duraiswamy 2012
From Captions to Visual Concepts and Back			Fang et al. 2015
WolframAlpha	Inc., Wolfram Research 2018	Inc., Wolfram Research 2018	Inc., Wolfram Research 2018
IBM Watson	Ferrucci 2010	Ferrucci 2010	Ferrucci 2010

2.2.8 Answer Generation

The answer generated by the Q&A Systems can be divided into two categories: Extracted Answer or Generated Answer. The works discussed in Section 2.3 are classified according to answer generation type in Table 8.

- Extracted Answer: This type of QA System extracts answers in the form of sentences or paragraphs directly from the knowledge base.

- Generated Answer: In this type of Q&A System, the answers are generated in the form of yes or no questions, opinionated answers or ratings, or dialog answers.

Table 8: Classification of answer generation

	Type of Answer	
System	Extracted Answer	Generated Answer
Intelligent QAS based on Artificial Neural Network	Ansari, Maknojia, and Shaikh 2016	
R-NET	Etworks 2017	
AskHERMES	Y. G. Cao et al. 2011	
FastQA	Weissenborn, Wiese, and Seiffe 2017	
QAS on Education Acts	Lende and Raghuwanshi 2016	
MEANS	Ben Abacha and Zweigenbaum 2015	Ben Abacha and Zweigenbaum 2015
PythonQA	Ramos, Pereira, and Henriques 2017	
WAD	Jayalakshmi and Shesasaayee 2017	
Dynamic QAS based on Ontology	Rajendran and Sharon 2017	
Ontological-semantic text analysis and the QAS	Mochalova et al. 2015	
Ontology-driven Visual QA Framework	Besbes, Baazaoui-Zghal, and Ghezela 2015	
QAAL	Kalaivani and Duraiswamy 2012	
From Captions to Visual Concepts and Back		Fang et al. 2015
WolframAlpha	Inc., Wolfram Research 2018	Inc., Wolfram Research 2018
IBM Watson	Ferrucci 2010	Ferrucci 2010

2.3 Q&A Systems

In this section, relevant works about Q&A Systems so further developed and described in the literature are introduced.

The PythonQA (Ramos, Pereira, and Henriques 2017) system was developed using the Python programming language, together with some libraries such as Natural Language ToolKit (NLTK) (Bird, Klein, and Loper 2009), Django, among others. To process the user's input, a module called Phrase Analysis divides a phrase into several components and tries to identify three elements: action, keywords, and question type. These three elements are then compared to the knowledge base to retrieve and show answers to the users of the Q&A System.

In MEANS (Ben Abacha and Zweigenbaum 2015) the authors propose a semantic approach to a medical Q&A System. They apply NLP to process the corpora and user questions. The sources documents are annotated with RDF, based on an ontology. The authors propose ten question types to classify the questions.

In work proposed by (Lende and Raghuwanshi 2016), a Q&A System to handle education acts is presented. The knowledge base is created from the data publicly available from the United Kingdom parliament using NLP techniques. Only keywords are extracted from the user question, ignoring the question type and possible actions present in the user's text input. Other works are in the field of education as (Jiang, Xu, and X. Wang 2019), (Agarwal et al. 2019), (Sreelakshmi et al. 2019). The work proposed by (Jiang, Xu, and X. Wang 2019) created a Q&A System to assist traffic controllers in monthly training. This system uses word vectors as inputs for the LSTM (Long Short-Term Memory) artificial recurrent neural network (RNN) to process the user questions for the question analysis. In the paper (Agarwal et al. 2019), is proposed the system EDUQA (Educational Domain Question Answering System). The EDUQA uses a conceptual network model containing educational semantics, capturing the pedagogical meaning of textual content. The EDUQA extract attributes from the conceptual network and generates a similarity coefficient based on WordNet and dynamic generated vectors to produce answers. It is proposed in (Sreelakshmi et al. 2019) the Quiz Q&A System, a system to generate quizzes about the data existing in the knowledge base. The system uses PDF as input files to generate the knowledge base. The Quiz Q&A System uses the Stanford NLP and a feed-forward neural network to generate the KB and pick the most relevant answers.

The work proposed by (Cai et al. 2020) proposes a framework to process Chinese questions using a convolutional neural network (CNN) with a bidirectional long short-term memory network (BiLSTM).

The authors in (Y. G. Cao et al. 2011) created AskHERMES, a Q&A System for complex clinical questions that uses five types of resources as a knowledge base (MEDLINE, PubMed, eMedicine, Wikipedia, and clinical guidelines). The user question is classified by twelve general topics, made by a support vector machine (SVM). The authors developed a question summarization and answers presentation based on a clustering technique to process the possible answers. In a work proposed by Weissenborn et al. (Weissenborn, Wiese, and Seiffe 2017), the authors propose a fast neural network Q&A System. The system uses a simple heuristic, and their results show that the proposed system can achieve the same performance compared to more complex systems. Another work on the domain of medicine is the Traditional Chinese Medicine (TCM), proposed by (Zou, He, and Y. Liu 2020). The authors use a generated knowledge graph using a semantic network produced by the Traditional Chinese Medicine Language System (TCMLS) in this work. The TCM system works with questions only in Chinese.

The authors of (Almansa, Rubio, and Macedo 2020) present the Question-Answering Surveillance architecture (QASF) to answer questions about chronic diseases. The QASF uses scientific papers as a knowledge base to be able to answers questions. To process the user question and generate answers, the authors propose the use of several technologies (dictionaries, ontologies, NLTK Snowball, WordNet, among others).

In work proposed by (Etworks 2017) is introduced the R-NET, a neural network model for answering questions. The neural network tries to answer questions from a given text. The work proposed in (Ansari, Maknojia, and Shaikh 2016) creates a deep neural network from documents provided by the user. They use deep cases and artificial neural network models to understand the contents of the user's information.

WolframAlpha (Inc., Wolfram Research 2018) is a well-established open domain Q&A System that initially was a closed domain system for mathematics. It allows the user to use the version available online with the pre-existing knowledge base or to upload data through a paid subscription.

IBM Watson (Ferrucci 2010) is an open domain Q&A System that was initially created to compete in the Jeopardy TV quiz program. Watson is currently an Artificial Intelligence framework provided by IBM for various areas, one of which is the Q&A Systems and natural language processing. Watson is made available through paid subscriptions.

In the work of Jayalakshmi et al. (Jayalakshmi and Sheshasaayee 2017), they use a similarity measure based on the user-written question and discover the appropriate meaning between the words. The authors propose the WAD Q&A System. It uses ontology and hierarchical web documents to perform entity linking to predict the answers.

Rajendran et al. (Rajendran and Sharon 2017) propose a Q&A System that uses ontology assistance, template assistance, and user modeling techniques to achieve 85% of accuracy in their experiments. The authors of (Mochalova et al. 2015) also use ontologies to assist the Q&A System. This work proposes an algorithm to automatically update the system's ontology and use a semantic analyzer that operates on an ontology to extract answers.

In work (Besbes, Baazaoui-Zghal, and Ghezela 2015), the authors improve question interpretation and the representation of question structure using typed attributed graphs and a question ontology. They also state that using domain ontologies and lexico-syntactic patterns improves the results. The NBAKB (NBA Knowledge Graph) system is proposed in (Y. Li, J. Cao, and Y. Wang 2019). The NBAKB uses a knowledge graph (KG) as a knowledge base. The knowledge graph is created by crawling several NBA-related sites and uses the Basketball Knowledge Graph (BKA) to generate the relationships and annotations on the KG. The system allows Chinese questions and uses BiLSTM-CNN to process the KG and the user question to provide the answer.

It is proposed in (Kalaivani and Duraiswamy 2012) a graph matching algorithm for query matching with an ontology using a spread activation algorithm. The spread activation algorithm uses the WordNet (Miller 1995) to calculate semantic similarity.

An approach to automatically generate image descriptions is proposed in (Fang et al. 2015). Firstly words describing the image are detected. Secondly, sentences relating to the objects in the picture are produced. The final step is to rank the phrases according to the MERT (Och 2003) model and present the best-ranked sentence to the user. The authors of the work (S. Lee et al. 2019), propose a Visual Question Answering (VQA) that uses scene graphs and a model based on memory, attention, and composition (MAC) to classify the answers. This work differs from the work proposed by (Fang et al. 2015), as there is no image processing to extract meaning in the technique. The VQA uses only the already generated scene graph description of the image.

The authors of (Shen et al. 2017) introduce implicit reasoning neural networks (IRNs) to infer information present in the knowledge base without having to processes all the data in the KB. This approach allows the Q&A System to outperform other approaches in the FB15k benchmark. In the work (Shang, J. Liu, and Yang 2020), the authors present an answer generation model that uses deep learning techniques. They also propose a novel position encoding method based on a trigonometric function to achieve better results. It is used the Squad (Rajpurkar et al. 2016) data set to provide results. A Q&A System in the area of Chinese mother-and-child is created in the paper (Yan and J. Li 2018). This system answers questions about the implementation of the two-child policy. The authors propose the generation of a domain-specific dictionary using word2vec based on deep learning techniques.

In work proposed by (Nguyen et al. 2016), they introduce a new dataset to assess machine reading comprehension. The questions are a sample from a real user dataset, and the answers were generated by humans. Some questions have multiple answers to access Q&A Systems.

The work proposed by (George 2020) presents an information retrieval technique where the system tries to find question words, all the nouns, and verbs. The system uses the Stanford NLP library to make structural connections, establishing dependency relations between words. After the system process the question, the selected answer candidates are the sentences that match the verb in the question or have the maximum similarity with the question. This work is similar to the work proposed by (Ramos, Pereira, and Henriques 2017).

2.4 Chapter's Considerations

In this chapter, concepts necessary for the construction of Q&A Systems were presented. It is also presented works that implement Q&A Systems and natural language processing. These works were classified according to their characteristics in Section 2.2. This chapter serves as a basis for developing and constructing the language proposed in chapter 4.

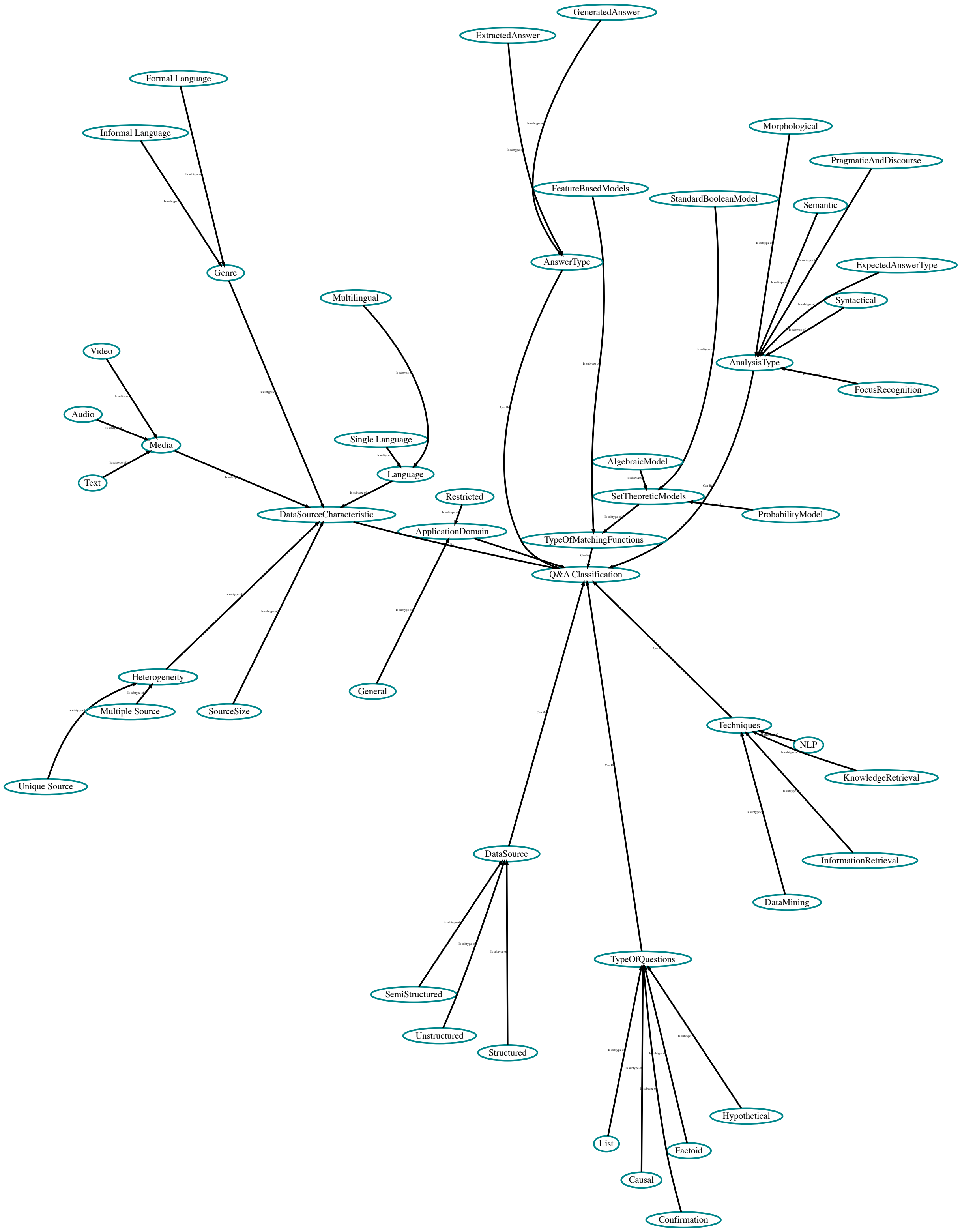


Figure 3: Ontology with classification types of Q&A Systems.

3

Generative Programming and Domain-Specific Languages (DSLs)

Domain-Specific Languages (DSLs) can simplify and accelerate the development of applications (Adam and Schultz 2015). This advantage comes with the disadvantage of learning a new language (Mernik, Heering, and A. M. Sloane 2005). According to Fowler (Fowler 2010), Domain-Specific Language is a computer programming language of limited expressiveness focused on a particular domain. DSLs are relevant for two main reasons: improve programmer productivity and allow non-programmers to read and understand the source code. The improved programmer productivity is achieved because DSLs try to resolve a minor problem than general-purpose programming languages (GPL) (Ghosh 2010) making it more straightforward to write and modify programs/specifications.

Structure of the Chapter. Section 3.1 presents an overview over DSL. A DSL classification is presented in Section 3.2, and in Section 3.3 discussions about the lifecycle of the development of DSL is presented.

3.1 Domain-Specific Languages

As the approach here proposed to make more accessible the creation of Q&A Systems is based on the design of a DSL specially tailored to allow end-users to describe the domain and the Q&A tool they desire in an abstract level, this Section is devoted to a brief review of the DSL concept and associated techniques as well as Generative Programming concepts.

Since DSLs are smaller and easier to understand than GPLs, they allow domain specialists to see the source code and get a more abstract view of their business. DSLs offer the capacity to domain specialists to create a functional system with no prior knowledge of GPLs.

What distinguishes DSLs from GPLs is the expressiveness of the language: instead of providing all the features that a GPL must contain, such as supporting diverse data types, control, and abstraction structures, the DSL has to support only elements that are necessary to a real domain. Examples of commonly used DSLs, according to (Ghosh 2010), are SQL, Ant, Rake, Make, CSS, YACC, Bison, ANTLR, RSpec, Cucumber, HTML.

Generative programming concerns the construction of specialized and highly optimized systems through the combination and design of modules. According to (Czarnecki 1999), the goals of generative programming are to decrease the conceptual gap between coding and domain concepts, achieve high reusability and adaptability, simplify the management of several components, and increase efficiency (space and execution time).

To be able to achieve the goals proposed by (Czarnecki 1999), generative programming recommends applying some approaches described next.

- Separation of concerns, that is, deal with one important issue at a time and combine these issues to generate a component.
- Parameterization of the components to be able to deal with families of components, allowing the use of the developed component in different scenarios.
- Separation from the problem space to solution space.
- Dependencies and interactions management to allow the combination of components that have parameters that differ and imply in another component.
- Perform domain-specific optimizations through the generation of some components statically or making transformations to allow distributed processing.

Generative programming uses DSL at a modeling level (Cointe 2005) to allow users to operate directly with the domain concepts instead of dealing with implementation details of GPLs.

According to (Czarnecki 2005) generative programming is a system-family approach, which allows the automatic generation of a system-family member, that is, a system that can be automatically generated from a textual or graphical DSL specification. In this thesis, the concepts related to generative programming through the use of DSL are used to allow the creation of Q&A Systems. An engine will be used to process the DSL formal specification (grammar) and generate a Q&A System according to the written description of the DSL proposed. Also, through the DSL's written description, a complete Q&A System is generated without building the system line-by-line by the developer. Section 6.3.2 present the proposed approach.

3.2 DSL Classification

DSLs can be divided into three main categories: internal DSLs, external DSLs, and language workbenches (Fowler 2010).

An internal DSL (also known as embedded DSL) is implemented as a subset of a general-purpose language. The DSL code is valid in the GPL and can be processed by the GPL tools such as the compiler. This type of DSL uses only a subset of the host GPL and has Ruby and Lisp as a few examples (Fowler 2010). According to (Ghosh 2010), internal DSLs are usually implemented as a library for a specific GPL.

External DSL is a language that is independent of other GPLs and their applications. An External DSL has its own (customized) syntax. This DSL type requires a new and specific infrastructure to handle the lexical analysis, parsing, compiling, and code generation. A few examples of external DSLs are SQL, Regular Expressions, and Awk.

The language workbench is an environment designed to help create new DSLs, similar to the integrated development environments (IDEs) for GPLs. As examples of this new type of DSLs are the systems: Intentional's DSL Workbench[1] and JetBrains Meta Programming System (MPS)[2].

3.3 Life cycle of DSLs

According to (Mernik, Heering, and A. M. Sloane 2005) to be able to develop a DSL, five phases should be followed: decision, analysis, design, implementation, and deployment.

3.3.1 Phase one: decision

The first phase recommended by (Mernik, Heering, and A. M. Sloane 2005) is to discuss the need to develop a new or use an existing DSL or even use a GPL. After this discussion, the developer has to decide what approach to follow.

When there is already a DSL that was designed for the domain in question, and it is possible to extend the language, this approach is recommended. If the developer has solid knowledge of the domain, developing a new DSL is suggested.

In a new domain or a domain with little to no developer knowledge, using a GPL is the recommended way to pursue.

After the decision is made by the developer, if the DSL is the chosen way, the following phases should be adopted.

3.3.2 Phase two: analysis

In this phase, which comes after the developer decided to develop a DSL, knowledge from the domain should be gathered. The developer should do a thoughtful analysis of the domain to consolidate the knowledge on the subject. This analysis can be done with technical documents, knowledge extracted from domain specialists, a systematic review of the area, among other ways.

The developer has to define at least the following data about the domain: scope, terminology, vocabulary, concepts. Those data are needed to support the development of the DSL.

[1] http://www.intentsoft.com
[2] http://www.jetbrains.com/mps

3.3.3 Phase three: design

The developer's choice to develop a new DSL or use an existing DSL determines how the design phase should be done.

The recommendations for the development of a new DSL, according to (Mernik, Heering, and A. M. Sloane 2005), are to use two approaches: informal or formal. The informal approach is when the DSL developer wants to specify the language in natural language, with a set of examples. The formal approach requires that the developer specifies the DSL using syntactic and semantic specification methods. These methods can be but are not limited to rules, grammar, regular expressions, abstract state machines.

For the use of an existing DSL, the following approaches are recommended: piggyback, specialization and extension. Piggyback is when parts of the existing DSL are used. In specialization, the developer restricts the existing DSL, tackling a specific problem. The last approach is when the developer extends an existing DSL.

According to (Mernik, Heering, and A. M. Sloane 2005), it is easier for the end-user of the DSL if the developed DSL is based on an existing language. This approach uses the familiarity of the users in a given language, making it easier to understand the language syntax.

3.3.4 Phase four: implementation

After the design of the DSL, the DSL should be implemented so that it can be used. According to (Mernik, Heering, and A. M. Sloane 2005), there are some implementation patterns that can make a big difference in the effort needed to implement the DSL. The seven recommended patterns are described next.

- Interpreter: In this pattern, the DSL constructs are recognized and executed in GPL interpreted languages. A cycle of fetch, decode and execute is used. The advantage of using this pattern instead of the compiler pattern is the simplicity and greater control of the execution environment. Some examples of DSL using this pattern are ASTLOG (Crew 1997), Service Combinators (Davies and Cardelli 1999).

- Compiler: The DSL constructs are translated to library calls and base language constructs. This pattern has the advantages of allowing static analysis and a faster execution speed than the interpreter pattern. ATMOL (Engelen 2002), ESP (Kumar 2002), FIDO (Klarlund and Schwartzbach 1999), Teapot (Chandra, Richards, and Larus 1999) are examples of DSL using this pattern.

- Preprocessor: DSL language constructors are translated to a base language. There are some subpatterns inside this one: macro processing, source-to-source transformation, and lexical processing. The following languages are examples of these subpatterns, respectively, S-XML (Clements et al. 2004), SWUL (Bravenboer and Visser 2004), SSC (Buffenbarger and Gruell 2001).

  – Macro processing: when the DSL use expansion of macro processors.

  - Source-to-source transformation: translation of the DSL into a base language.
  - Lexical processing: when only simple lexical scanning is applied.

- Embedding: When the DSL is constructed as a library of a GPL. Examples of DSL using this pattern are Hawk (Launchbury, Lewis, and Cook 1999) and Nowra (A. Sloane 2002).

- Extensible compiler/interpreter: This pattern extends an existing GPL compiler or interpreter. According to (Mernik, Heering, and A. M. Sloane 2005), interpreters are simple to extend. However, compilers are hard to extend unless they were made with extension as a basic feature. The DSL DiSTiL (Smaragdakis and Batory 1997) uses this pattern.

- Commercial Off-The-Chelf (COTS): It is the use of tools and notations already developed to a specific domain. OWL-Light (Bechhofer 2009) is an example of a DSL using this pattern.

- Hybrid: This approach is when at least two of the above approaches are applied. An example of DSL using this pattern is GAL (Thibault, Marlet, and Consel 1999).

3.3.5 Phase five: deployment

In this last phase, the implemented DSL is used. When the first four DSL lifecycle phases are used as a guideline, a usable DSL is ready to be deployed and used.

The end-user of the DSL can write specifications in the developed DSL in this phase. According to (Tomassetti 2020), it is essential for the DSL developer to provide some tools to support the end-user development. The tools can make it easier to write specifications on the developed DSL. Tools can be necessary to use a language (compiler or an interpreter for the DSL). Some tools can be helpful to the end-user of the DSL, as an editor with syntax highlighting and auto-completion, or even with debugging support.

Developers and domain experts use the DSLs to specify models in this final phase.

3.4 Chapter's Considerations

In this Chapter were discussed concepts about DSL and code generation. These concepts are essentials to the DSL proposed in this thesis. The use of an External DSL is designed precisely to fit adequately in the main objective of this work: the formal and assisted development of Q&A Systems. This decision demanded the construction of a new compiler for the AcQA DSL proposed in this work. The AcQA DSL is presented in Chapter 4.

4

AcQA - Automatic creation of Q&A Systems

As stated in the previous chapters, the use of DSLs and generative programming allows domain specialists to build entire systems without the need for GPLs knowledge. It is proposed in this thesis the AcQA (Automatic creation of Q&A Systems) domain-specific language that allows a specialist to develop a Q&A System. The focus of AcQA language is on the knowledge associated with the domain, allowing the developer to focus on the data to create that knowledge rather than on how to implement Q&A Systems. The language provides question analysis, answer formulation, and the formula to rank answer candidates. This chapter describes the AcQA architecture and presents the main components.

Structure of the chapter. In section 4.1 the arquitecture of AcQA is presented; Section 4.2 shows the design of the AcQA DSL and in Section 4.3 remarks about the proposed language are presented.

4.1 AcQA architecture

Generative programming and domain-specific languages allow domain specialists to develop entire Q&A Systems without the need to code or knowledge in GPLs. This work is based on concepts of generative programming and domain-specific languages. These concepts are used to enforce constraints and validation of inputs, offering correctness guarantees and performance benefits.

This section describes the domain-specific language AcQA (Automatic creation of Q&A Systems). AcQA allows an expert or regular user to specify this kind of system more straightforwardly than in a GPL. The focus is on the behavior of the whole system rather than how to implement them. This approach allows the user to focus on the data made available for building the system knowledge base and several behaviors such as the techniques that need to be used to process the user inputs (questions) and its front-end (Web, WebService, others). Experienced Q&A Systems developers who already have tools to construct Q&A Systems can extend the AcQA language to use these tools. Domain experts with no previous experience developing Q&A Systems can develop a functional Q&A System using only the AcQA DSL.

During the development phase, the developer should address several issues, such as which back-end is used to support the Q&A System (language, framework, server technologies, database). What languages are supported, which are the possible input formats (text, audio, Braille), how to communicate with the user (graphic, speech, Braille). These are questions that the developer has to consider at the beginning of the developing phase when creating a Q&A System in GPL's. When the Q&A System is developed using AcQA, the user can customize the system at any time, not needing to worry about all the techniques that can be used in development. As an example of the complexity underlying the development of a Q&A System, in (Azevedo, Henriques, and Pereira 2018) Python was used as a GPL for the engine combined with Django and the Python Natural Language Toolkit (Bird, Klein, and Loper 2009).

To achieve the objective of generating a Q&A System, the concepts discussed in Chapters 2 and 3 are applied in the AcQA language. Figure 4 shows the architecture of AcQA and how all the steps needed to generate the Q&A System are connected. The specification of the desired Q&A System in AcQA is written by the user. The Core module is responsible for validating the specification written in AcQA and make the connection with the Data and Presentation modules. If the AcQA specification is syntactic and semantical validated by the AcQA Processor, in the Core module, the AcQA Engine interacts with the Data module and the Presentation module. The Data module is responsible for processing and mapping the input files to a format that the AcQA language understands and is used by the AcQA Engine to generate the knowledge base. The Presentation module is responsible for generates the front-end that allows the interaction with the users of the Q&A System.

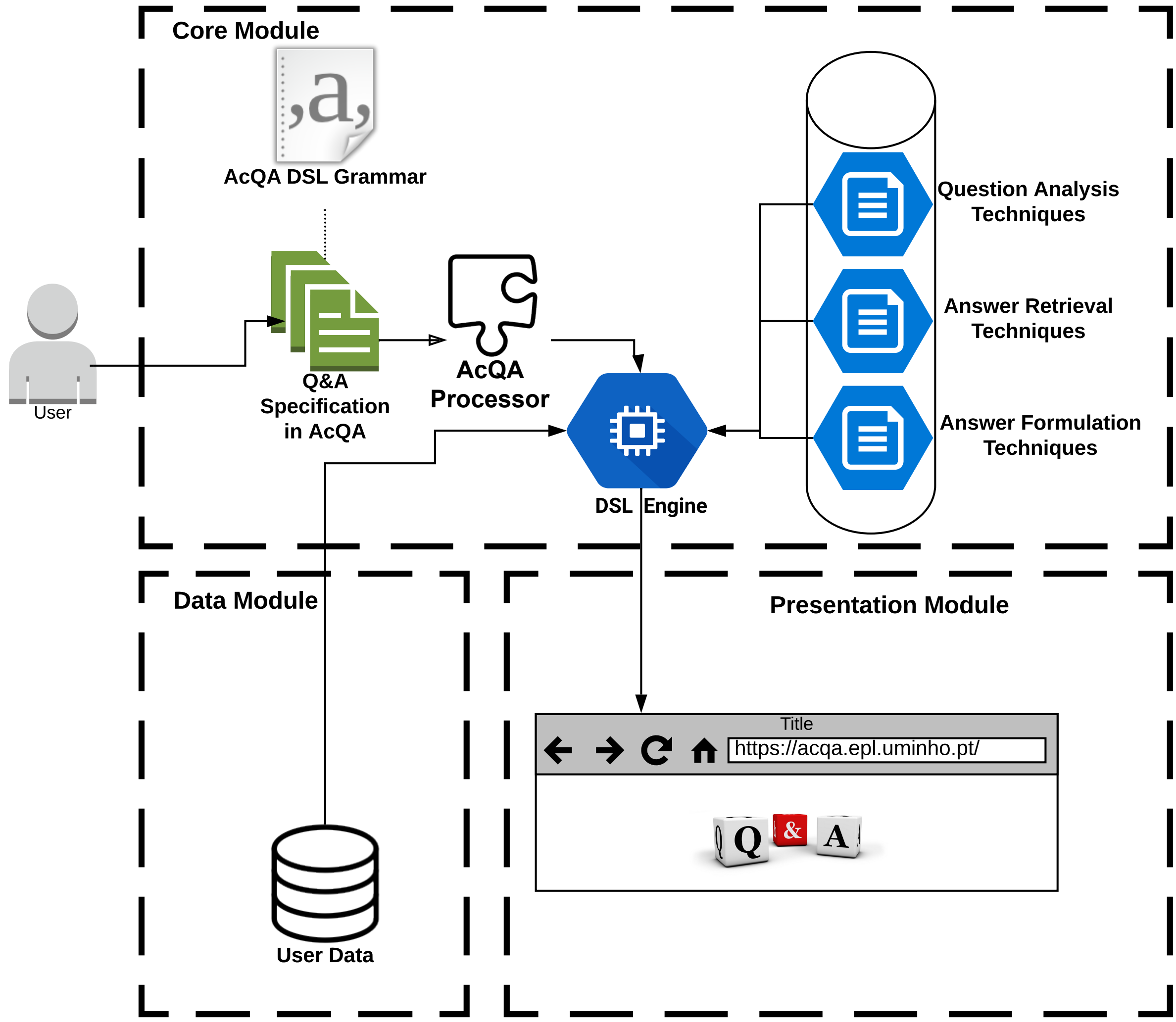


Figure 4: AcQA architecture

4.1.1 Core Module

In this module, the AcQA grammar is processed and validated. All these steps are executed into the AcQA Processor and produce an internal data representation. Suppose the specification written in AcQA is validated by the AcQA Processor. In that case, the AcQA Engine uses the techniques needed to generate the Q&A System, like the question analysis, answer retrieval, and answer formulation techniques. These techniques are described in Chapter 5.

4.1.2 Data Module

User data may be available in various formats and sources, needing to be made available in a format that the AcQA language understands. The Data module can pre-process and process the user data to

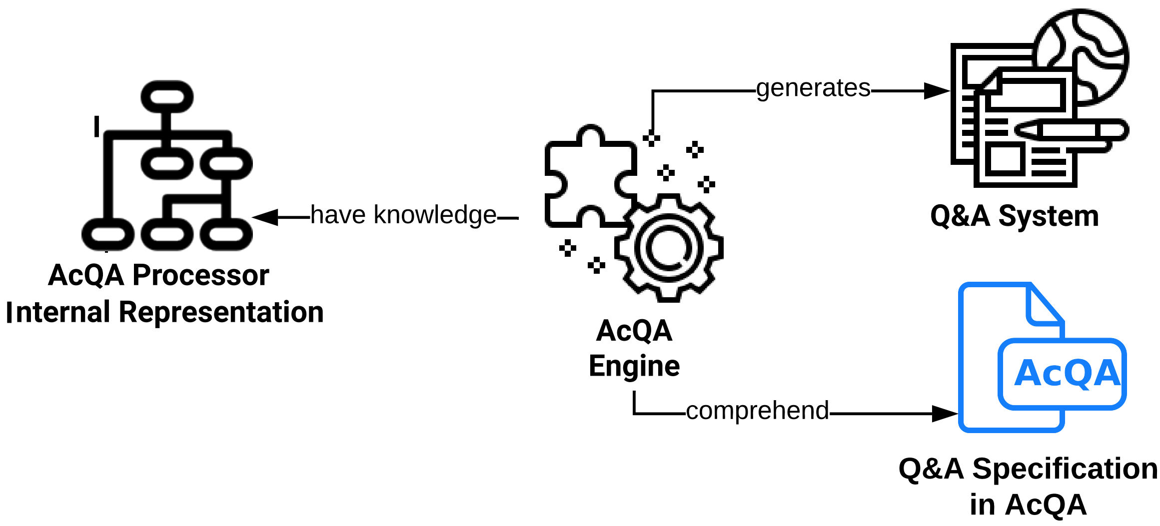


Figure 5: Steps needed to generate the Q&A System

make them available to the AcQA Engine into the Core module. The available data formats that AcQA understands are described in Chapter 5.

4.1.3 Presentation Module

It is responsible for providing an interface where users or systems can interact with the Q&A System. All the interactions with the generated Q&A System are provided by this module. It can add more data into the knowledge base or manage the Q&A System user's permissions.

4.1.4 Steps needed to generate a Q&A System using AcQA

The steps needed for the AcQA language to generate a Q&A System are done inside the AcQA Engine. The engine knows the AcQA Processor internal data representation and uses it to process the specification written in AcQA. If AcQA Engine can successfully validate all the requirements, it generates the Q&A System. Figure 5 shows the steps needed to generate a Q&A System.

The AcQA DSL grammar was specified using the tool ANTLR (Parr 2013). This tool recognizes the grammar file and is used as a base in the AcQA Processor. The AcQA Processor is the compiler for AcQA DSL. This processor is responsible for recognizing specifications written in AcQA and produces an internal representation that is available to the AcQA Engine. The AcQA Engine is responsible for preparing all the required configuration and code generation needed to run the Q&A System. These steps are presented in Figure 6. The python language is used to handle the tasks needed both by the compiler and the generated Q&A System. Chapter 6 presents and discusses examples of Q&A Systems generated

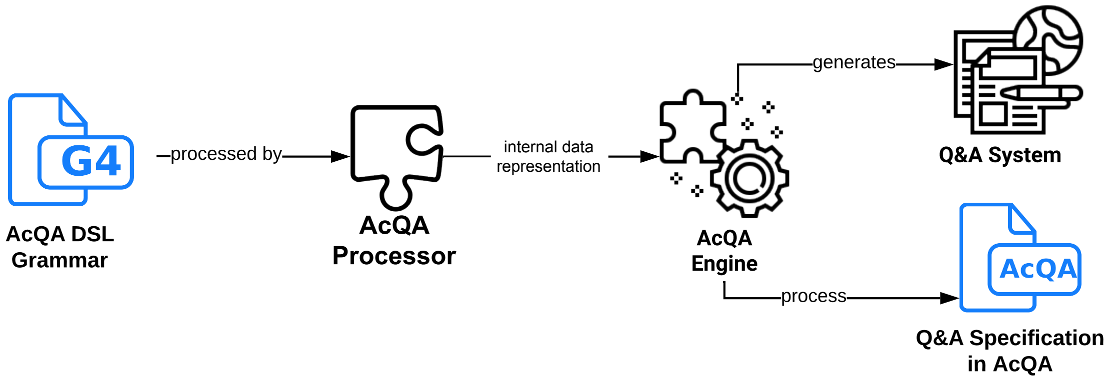


Figure 6: Full steps needed to process AcQA programs

by AcQA to answers questions of Board & Card Games, about Python programming language, and to answer questions about where are classes and meetings by a robot.

The AcQA grammar and design are presented in Section 4.2.

4.2 AcQA DSL Design

When a developer implements a Q&A System, several issues should be addressed, such as back-end technologies to process the user's questions, retrieve the appropriate answers, or front-end technologies to get input from the user and display the built answers. To make the development phase of Q&A System easier, the AcQA language is proposed and described in this section.

The AcQA language is an external DSL, containing a custom syntax to make the specification and parameterization of a Q&A System more user-friendly. There is also a default value for parameters, thus allowing the user to specify only a few values and build the Q&A System automatically.

The AcQA language already has several off-the-shelf elements to allow the construction of a Q&A System. This section presents the elements available to be used by the Q&A System creator.

All the parameters written in AcQA are syntactically and semantically validated. AcQA Engine is responsible for enforcing semantic correctness.

4.2.1 AcQA main elements

Figure 7 shows the main elements of AcQA grammar. This Figure shows the declaration of the main definitions needed to set up the initial working Q&A System. Each line of a specification in AcQA needs to have a comment or a declaration (lines 1 and 2). AcQA DSL has six main declaration blocks (line 3): Input File, Techniques, UI, Server, NoDeploy, and CleanKB. These blocks specify the behavior of the generated Q&A System. Lines 4-6 present the definition that allows comments in AcQA specification.

```
acqaFile: lines+ EOF;
lines:(comment | decl | TERMINATOR | EOF) ;
decl: inputfile | techniques | ui | server | nodeploy | cleankb;
comment:COMMENT | SINGLE_LINE_COMMENT | MULTILINE_COMMENT;
SINGLE_LINE_COMMENT: '--' ~[\n]*;
MULTILINE_COMMENT: '/*' .*? ( '*/' | EOF );
...
```

Figure 7: AcQA grammar main fragment

4.2.2 AcQA data input block

Figure 8 presents the fragment of AcQA grammar to define the input file of the Q&A System. Line 1 specifies the input keyword *INPUT* into the grammar to define data that has to be imported to create the Q&A System's knowledge base of the Q&A System. The input keyword needs to have at least the *path* param containing the file location that needs to be processed to generate the Q&A System. The input block also allows the user to set optional parameters (line 2) to change the parser's behavior, such as parser type, parser options. The user can choose between several parser types for parsing the data needed to build the knowledge base. In this release of AcQA, it is possible to parse the following file formats: eXtensible Markup Language (XML), Raw Text (in any encoding, as long as it works with Python), SQL, HTML, DOC, XLS, JSON, or PDF. Other file formats can be processed through the extension of an interface provided by the AcQA language, and it is the developer's responsibility to develop this extension. The optional parameters are $key => value$ (as defined in lines 3-5) to change the parser's behavior.

```
1 inputfile: INPUT '(' PATH (',' input_options)* ')';
2 input_options: INPUT_PARSER | params;
3 params:key'='value;
4 key: IDENTIFIER;
5 value:NUMERIC_LITERAL | STRING_LITERAL | INT | PATH;
6 PATH: STRING_LITERAL;
7 STRING_LITERAL: '\'' ( ~'\'' | '\'\'' )* '\'';
8 ...
```

Figure 8: AcQA grammar input block

4.2.3 AcQA techniques block

The Techniques block (line 1 - Figure 9) defines which techniques are used in the question analysis, answer retrieval, and answer formulation processes. If this block is not specified, the default behavior uses techniques associated with the Triplets approach. These Triplets techniques are initially based on works described in (Azevedo, Henriques, and Pereira 2018) and (Ramos, Pereira, and Henriques 2017), where a closed-domain Q&A System is implemented to answer Python-related questions. The Triplets technique is presented in more detail in Section 5.2.3.1. These techniques were used as the initial approaches to accelerate the development of AcQA and use the know-how from the language processing group (gEPL)[1] of the University of Minho. The not obligatory options are defined as $key => value$.

```
1 techniques: TECHNIQUES '(' TECHNIQUES_TYPE (','
    ↪ techniques_options)* ')';
2 techniques_options: params;
3 params:key '=' value;
4 key: IDENTIFIER;
5 value:NUMERIC_LITERAL | STRING_LITERAL | INT |
    ↪ PATH;
6 PATH: STRING_LITERAL;
7 STRING_LITERAL: '\'' ( ~'\'' | '\'\'' )* '\'';
8 ...
```

Figure 9: AcQA grammar techniques block

4.2.4 AcQA UI block

The UI block is responsible for specifying which type of UI the system deploys (line 1 - Figure 10). The available front-ends to provide access to the Q&A System are twofold: HTTP and RESTful WebService. The HTTP front-end is a graphical interface available through the HTTP protocol, having a responsive interface and can be accessed through computers, tablets, or cell phones. Using the RESTful front-end allows the creators of the Q&A System, who already have some developed platform, to provide access to the user who wants to ask questions by integrating their platform with the AcQA generated Q&A System.

[1] https://epl.di.uminho.pt

```
ui: UI '(' UI_TYPE (',' ui_options)* ')';
ui_options: params;
params:key'='value;
key: IDENTIFIER;
value:NUMERIC_LITERAL | STRING_LITERAL | INT | PATH;
PATH: STRING_LITERAL;
STRING_LITERAL: '\'' ( ~'\'' | '\'\'' )* '\'';
...
```

Figure 10: AcQA grammar ui block

4.2.5 AcQA Server block

The block of AcQA that configures and deploys the system to a given location is the *Server* (line 1 - Figure 11). The user needs to specify at least the hostname of the server, the user name, and the password or key to access the server. If the Server block is not defined, the AcQA language cannot generate the Q&A System, as it requires a fully functional server to deploy the generated code. The parameters path and hostname are syntactically verified in the grammar of AcQA. AcQA Engine is responsible for enforcing the semantic correctness of these parameters.

```
1 server: SERVER '(' HOST (',' server_options)*')';
2 server_options: (USER | PASSWORD | KEY ) '=' value;
3 params:key'='value;
4 key: IDENTIFIER;
5 value:NUMERIC_LITERAL | STRING_LITERAL | INT | PATH;
6 PATH: STRING_LITERAL;
7 STRING_LITERAL: '\'' ( ~'\'' | '\'\'' )* '\'';
8 ...
```

Figure 11: AcQA grammar server block

4.2.6 AcQA NoDeploy and CleanKB definitions

To allow the programmer to change the knowledge base or clean a knowledge base of the generated Q&A System, these two definitions were made, as defined in Figure 12.

The developer can define in the specification the keyword *nodeploy*, so the AcQA Engine does not update the code generated and deployed into a server. It only updates the knowledge base of the resulting Q&A System.

If the developer wants to clean the knowledge base of the generated Q&A System, it can use the keyword *cleankb*. When used the keyword *cleankb*, the AcQA Engine connects to the Q&A System running server and removes all the data from the knowledge base.

```
                    nodeploy: NODEPLOY;
                    cleankb: CLEANKB;
```

Figure 12: AcQA grammar NoDeploy and CleanKB definitions

4.3 Chapter's Considerations

In this chapter, the main aspects of the AcQA language were presented. The main objective of AcQA language is to allow users to develop Q&A Systems using a systematic and rigorous approach based on a DSL instead of GPL.

The specification in AcQA does not require extensive knowledge of general programming languages from the developer, leaving the effort to be focused on the data and upon which techniques are used in the generated Q&A System.

In Chapter 5, the concrete aspects of AcQA are discussed, along with the techniques and parameters that can be used in the language.

5

Code Generation

This chapter presents the transformation processes needed to generate the Q&A System code from the specification of AcQA DSL.

Structure of the chapter. Section 5.1 present the AcQA Processor, responsible for processing and transform a specification written in AcQA into an intermediate representation that later is used in the AcQA Engine; The AcQA Engine is described in detail along with the techniques available on the AcQA language in Section 5.2. The main remarks are presented in Section 5.3.

5.1 AcQA Processor

To generate the Q&A System, first, an interpretation and translation of the specification written in AcQA are needed to generate an intermediate representation of the code. In this thesis, this is done by the AcQA Processor. The AcQA language compiler is the front-end in the internal structure of the compiler, generating the intermediate representation through the parsing of AcQA code and the lexical and semantic analysis. The internal representation is then used by the AcQA Core to generate the fully functional Q&A System. Figure 13 shows the steps needed to generate the Q&A System through a compiler perspective. The AcQA Processor generates the internal representation, and the AcQA Engine (discussed in Section 5.2) uses the internal representation to generate the Q&A System.

As already stated in Chapter 4, the AcQA DSL grammar was specified using the tool ANTLR (Parr 2013). In AcQA Processor, the ANTLR was used to process the specification (written in AcQA) and is responsible for all the required analysis (lexical, syntactic, and semantical). To maintain coherence, all the modules that comprise the AcQA Processor are written in python 3 language in conjunction with the python 3 ANTLR library.

5.1.1 AcQA Specification

Figure 14 presents a code fragment of a specification written in AcQA DSL to configure a Q&A System for Board Games. Line 1 in Figure 25 specifies the file name (with a valid path) and which parser is

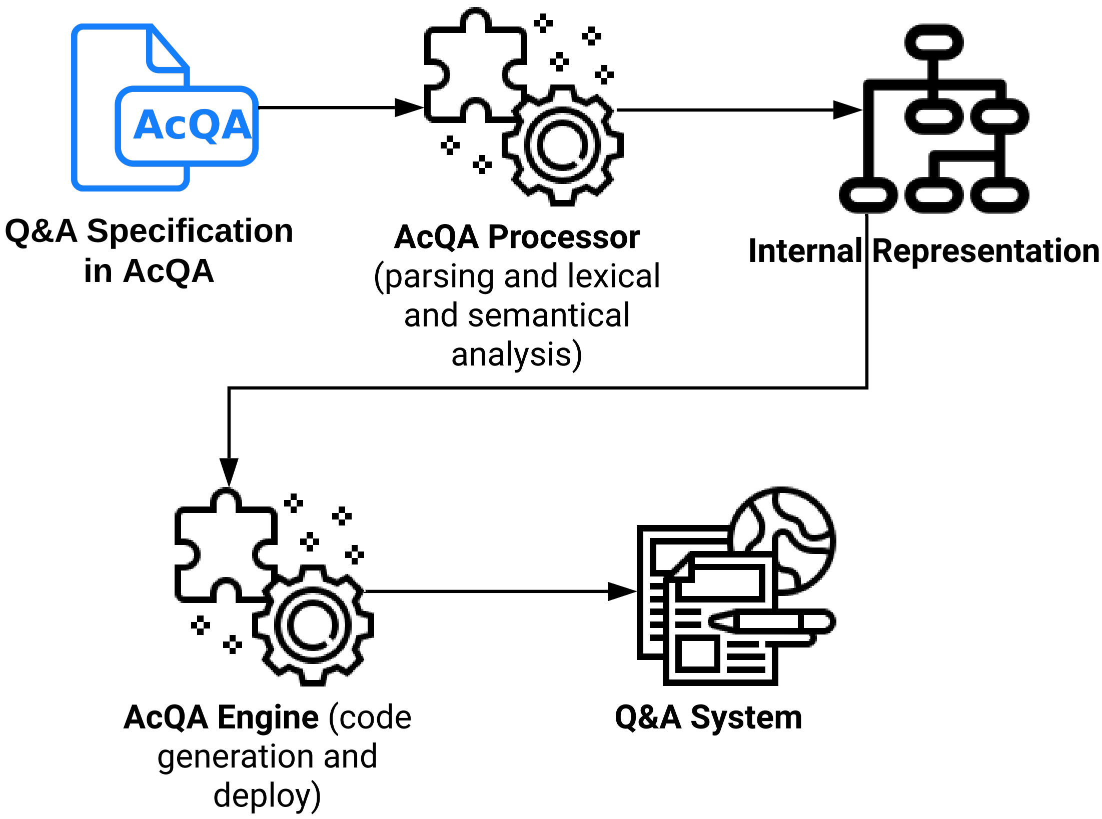


Figure 13: Steps needed to generate the Q&A System

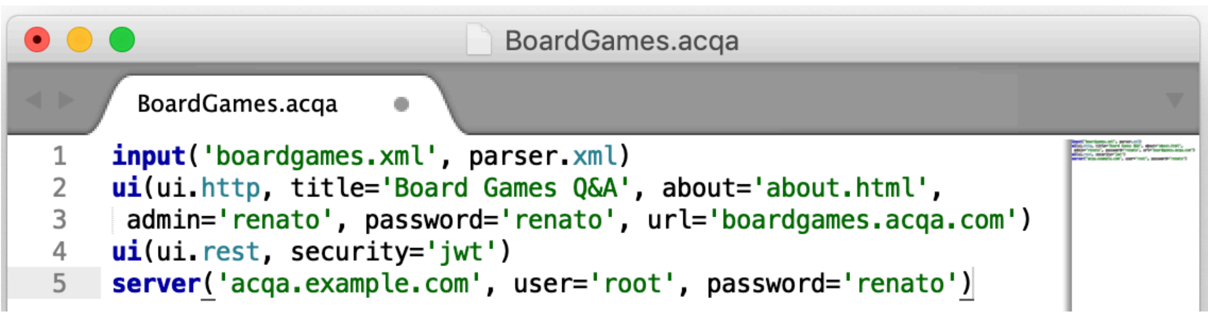


Figure 14: Q&A System specification in AcQA

used to process and load the user data into the knowledge base. It is possible to set optional variables to determine a specific behavior of the parser. In this example, the techniques block was omitted, so the generated system uses by default the triplets technique.

The UI's are specified in lines 2-4. It is obligatory to specify at least one parameter: the UI type. In lines 2-3, an HTTP UI is defined with some parameters of the UI: the title of Q&A System, the HTML used in section About, the admin credentials to access the admin page, and the URL used to configure the services in the Server, respectively. In line 4, a setup of a RESTful web service is shown. The parameter security defines which type of security is used by the RESTful web service. In this example, the JSON Web Token (Beltran 2016) is used to provide authentication and authorization in the web service.

The code at line 5 (Figure 25) configures in which Server the Q&A System is deployed, and the first parameter is the hostname of the Server. The last two parameters are the login information that is used to connect to the server through an SSH (Secure Shell) protocol (Ylonen 1996).

The AcQA DSL allows the user to change the behavior of the whole Q&A System. For example, the user can change the language of the system to Portuguese instead of English using the parameter *language="portuguese"* inside the UI block. The changes are applied in tokenizer, POS (Part-of-Speech) tagger, lemmatizer, and Wordnet language. These concepts are discussed in Section 5.2.3.1.

To generate the internal representation of the specification shown in Figure 14 a multidimensional array is defined to store all the required information. The AcQA Processor store all the data read from the specification. The processor verifies if the file 'boardgames.xml' exists and if the parser is known by the AcQA language. All the specification written in AcQA language is syntactically and semantically validated. For example, if the file 'boardgames.xml' does not exist or has invalid content, an error is thrown. These errors are discussed in Section 5.2.

5.2 AcQA Engine

To be able to generate the fully functional Q&A System, the AcQA Engine uses the internal representation produced by the AcQA Processor.

The steps needed to generate the fully operational Q&A System are depicted in Figure 26. The AcQA grammar is processed by the AcQA Processor to generate the internal representation that is used by the AcQA Engine. The specification is written by the user and recognized by the AcQA compiler inside the AcQA Processor. The data from the user is imported through the Input Parser set in the AcQA specification. The AcQA Engine then generates a Q&A System specified by the user. The Q&A System is deployed into the Server when AcQA Engine executes the Deploy Engine. After that, the Deploy Engine processes the User data to create the knowledge base used by the fully operational Q&A System.

The AcQA Engine generates all the code that is used by the Q&A System. The Engine and the code generated are all written in python 3. The Sections 5.2.1, 5.2.2, and present the techniques that can be used in AcQA.

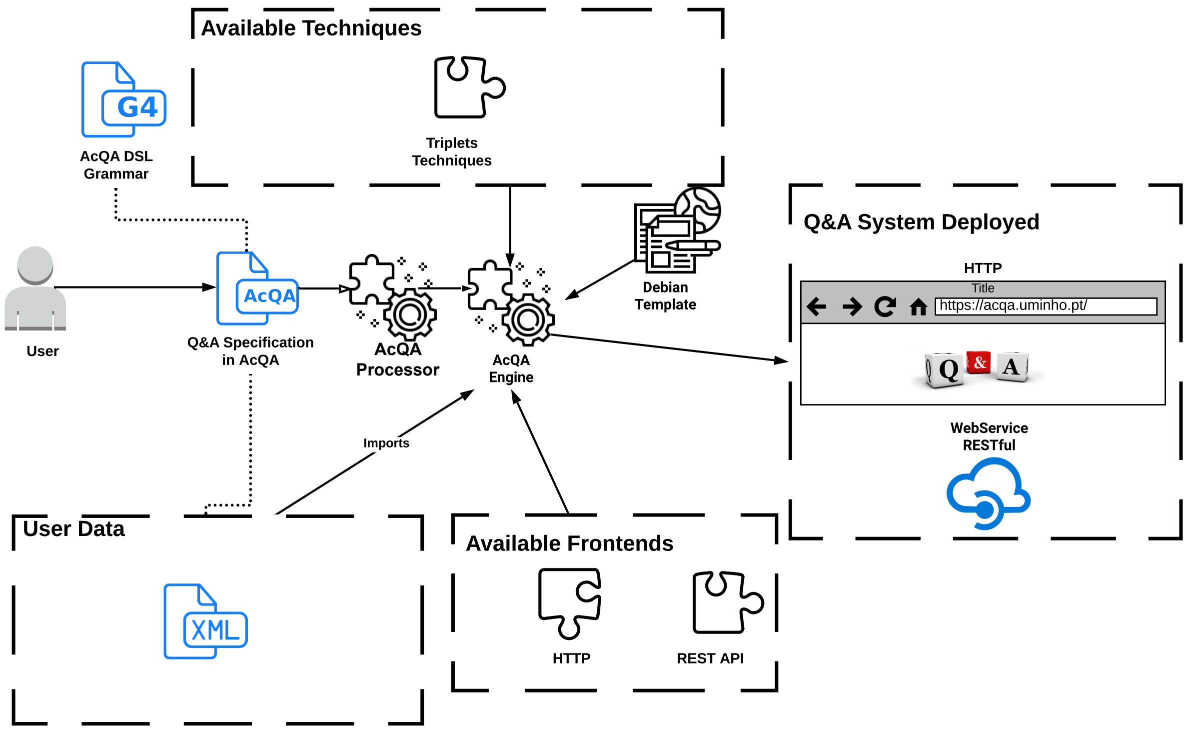


Figure 15: Steps to generate a Q&A System

5.2.1 AcQA Data Input Techniques

To be able to create the knowledge base that is used by NLP techniques, as discussed in Section 5.2.3.1, the AcQA language allows the user to choose from a set of parsers. There are several parser types for parsing the user data needed to build the knowledge base. It is possible to parse the following file formats: eXtensible Markup Language (XML), Raw Text (in any encoding, as long as it works with python), SQL, HTML, DOC, XLS, CSV, or JSON. Other file formats can be added through the extension of an interface provided by the AcQA language, and it is the responsibility of the developer to write this extension. These techniques create an abstract data model used as input data to the natural language processing techniques. Figures 16 and 17 present a fragment of an XML and JSON input files that can be used to generate the knowledge base of a Board & Card Games Q&A System.

5.2.2 AcQA Frontends

To allow the interaction with the Q&A System, AcQA generates at least one front-end. As already stated in Section 4.2.4, AcQA has two options to use as a front-end: an HTML5 UI and a REST web service. The HTML5 user interface allows a user to access the resulting Q&A System with any modern browser. This UI is accessible both on computers and mobile devices. Figure 18 shows a screenshot of the desktop version of the generated Q&A System of the Board & Card Games Q&A. This user interface allows a user

```
<acqa>
<qa>
<question>Playing exploding kittens and the
    ↪ game is unclear
about the effect of nope cards mid attack. If a
    ↪ player has already drawn
one of their attack and it's a nope, can they
    ↪ play it? What happens?
</question>
<answer questionid="1">While not explicity
    ↪ stated in the rules, my
understanding is that Nope reacts to a card (or
    ↪ pair, or triple) just
played, in which case once you've drawn the
    ↪ Attack has already
resolved and you can't Nope it.
</answer>
</qa>
</acqa>
```

Figure 16: XML fragment of an input file needed to generate the knowledge base of a Q&A System

to ask a question in two ways: textual and by voice. If the user chooses to ask a question using their voice, the server sends a request to an external API to process the audio and return a textual representation of the user's asked question. With the required credentials, the API can be set by the options in the AcQA UI block. If the user chooses to use the default Google API, it is possible to use between 36 languages to process the user's voice. The HTML5 UI uses the Django and the bootstrap framework.

AcQA also has a REST web service that can be used to allow interaction with the Q&A System. In this interface, a REST web service is exposed as an endpoint, thus allowing that interaction with the AcQA generated Q&A System can be done by software or different user interfaces. The REST web service exposes several interfaces to the programmer (Figure 18), such as uploading a new file to be used as a knowledge base, removing all the data in the knowledge base, asking questions, logging in, and logging out.

5.2.3 AcQA Server

To be able to deploy the resulting Q&A System, a server is needed. The AcQA Engine can be deployed to a variety of Linux Servers. The AcQA language uses templates to automatically deploy the Q&A System and the requirements to run the generated code. As already stated, all the generated code is written in python 3 language so that the Q&A System can run in several operational systems. The AcQA Engine can work with Debian-like and RedHat-like Linux distributions.

```json
{
    "acqa": {
        "qa": {
            "question": "Playing exploding kittens and the game is unclear \n\tabout the effect of nope cards mid attack. If a player has already drawn one of their attack and it's a nope, can they play it? What happens?",
            "answer": "While not explicity stated in the rules, my \n\tunderstanding is that Nope reacts to a card (or pair, or triple) just \n\tplayed, in which case once you've drawn the Attack has already \n\tresolved and you can't Nope it."
        }
    }
}
```

Figure 17: JSON fragment of an input file needed to generate the knowledge base of a Q&A System

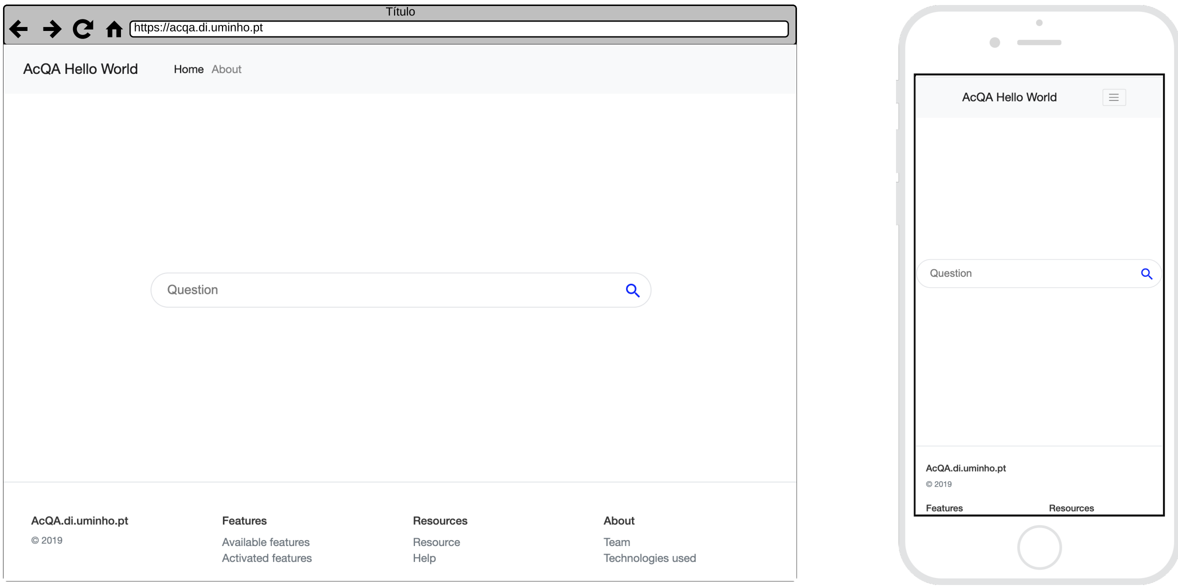


Figure 18: Screenshot of the HTML5 UI generated by AcQA

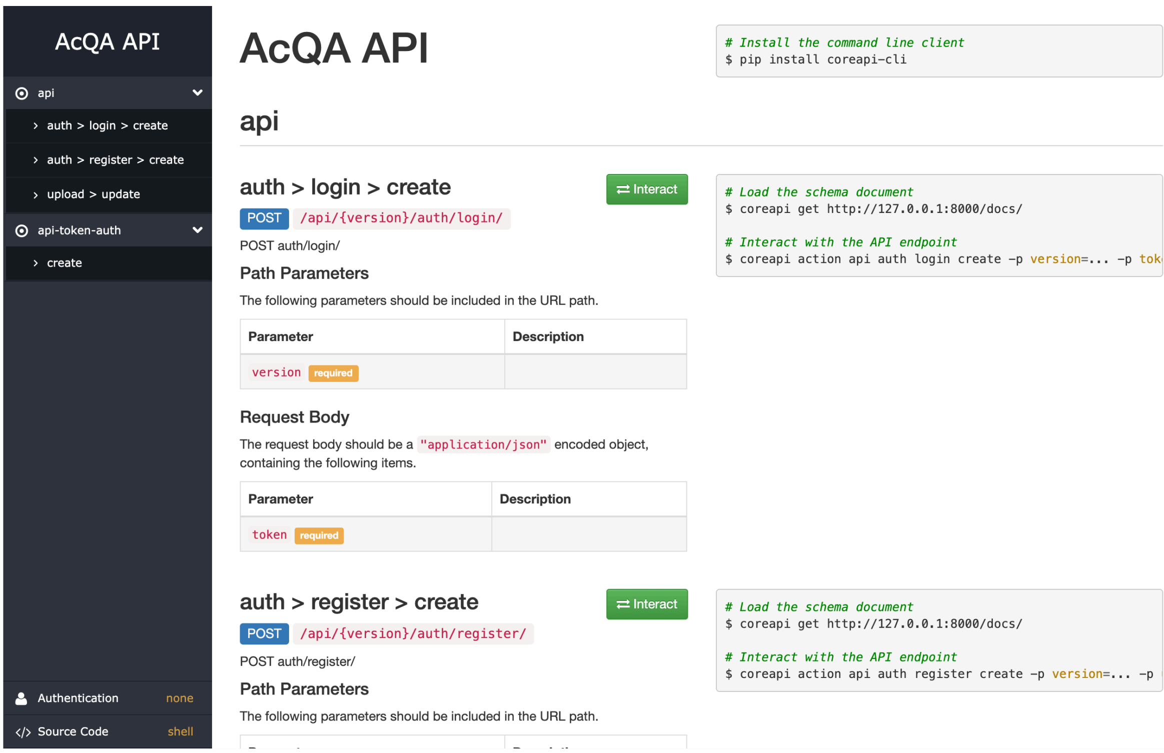


Figure 19: Screenshot of REST API UI generated by AcQA

```
server {
        server_name {{ server_name }};
        charset     utf-8;
        access_log  /var/log/nginx/qacore-access.log;
        error_log   /var/log/nginx/qacore-error.log;
        client_max_body_size 75M;
        location /media {
                alias /var/acqaQA/acqacore/media;
        }
        location /static {
                alias   /var/acqaQA/acqacore/qaSystem/static;
        }
        location / {
                include proxy_params;
                proxy_pass http://unix:/var/acqaQA/acqacore/qacore.sock;
        }
}
```

Figure 20: Configuration written in Jinja template template mechanism to generate a correct configuration file for the webserver Nginx

When the AcQA Engine process performs all the required processes for the correct installation of the Q&A System, the Engine connects to the server using the SSH credentials written in the server block. The password parameter can be interchanged by the RSA (Rivest-Shamir-Adleman) cryptographic key for added security (Stallings 2017). There is also an optional parameter to specify which server type (Debian, Fedora, among others). Several installation processes are executed if the AcQA Engine successfully connects to the server: python 3 language; Natural Language ToolKit (NLTK); Nginx WebServer (Nedelcu 2013), gUnicorn (Chesneau 2021) wrapper for python and Django (Django Software Foundation 2021); several minor libraries and configurations. After the correct installation of all the requirements, the AcQA generated code is then uploaded to the server to start processing the input data to generate the knowledge base of the Q&A System. All these steps are made by the AcQA Engine. To be able to configure the server according to the options defined by the user in the AcQA specification, several templates are written in the template mechanism Jinja (Ronacher 2008). These templates generate the correct configuration files needed to run the generated Q&A System on the Linux server. Figure 20 presents a jinja template to configure the Nginx according to the server name configured by the user on the UI block.

5.2.3.1 AcQA **NLP Techniques**

In this thesis, a natural language processing approach was used to extract meaning from the user's questions. The triplets technique was developed using the Python programming language and some libraries such as Natural Language ToolKit (NLTK) and WordNet.

To process the input from the user, a module called Phrase Analysis divides a phrase into several components and tries to identify three elements: action, keywords, and question type.

An example of the triplets generated by the question *Is there an easy way I can tell if an MTG card is "rare"just by looking at it?* is: <action: *tell*; keywords: *mtg, card, rare, look*; question type:*tell* >. The question and the triplets were extracted from the running example, described in subsection 6.2.1. With these three elements, the PythonQA can discover and store the intent from the user question.

Figure 21 describes the significant phases of the Phrase analysis. Firstly the question is processed with the NLTK library to replace contractions, converting them to their complete form. The following two steps use the NLTK library to divide the phrase into multiple strings using the Tokenizer package, allowing the use of the POS (Part-of-Speech) tagger.

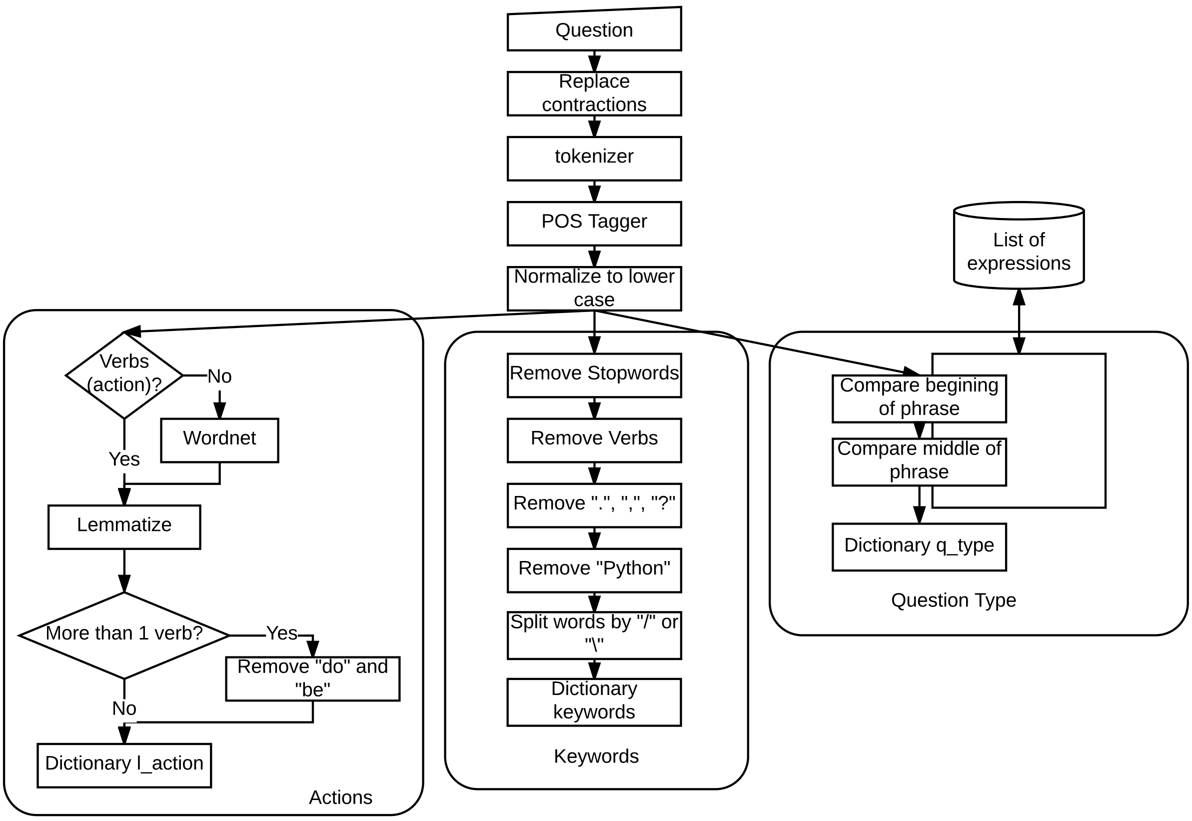


Figure 21: Phrase Analysis: Process to extract the three main elements: action, keywords, and question type

After the POS tagger is applied, the words are converted to their lower form, avoiding problems when comparing words. This conversion has to be done after the POS tagger because this can decrease the efficiency of the tagger. The verbs are then processed to find actions in the question. If no verb were found, the system tries to analyze the phrase in WordNet, to detect if a word can be a verb. The next step is to convert these verbs found in the infinitive mode using the NLTK WordLemmatizer package. If more than one verb were found, the system would try to identify and exclude false-positive verbs. A value for quality is assigned for each verb recognized in the previous steps. To identify keywords, firstly, the following information is removed: stopwords, verbs, unwanted characters. After the removal of unwanted information, the keyword candidates are processed to split words that may have a slash (”/”or ”\”) between them. A dictionary is then created with the keywords found in the previous steps and the value of assertiveness.

The Triplets techniques contain a list of expressions that were extracted with the analysis of the PyFAQ (Python Frequently Asked Questions) (**python**). This list has expressions like *How*, *When*, *Where*. This expression list is used to discover the question type of the phrase. The system searches for the presence of these words and generates a dictionary of question types. Depending on the position in the sentence (beginning or middle) is assigned a weight for the question type.

The Knowledge Base is constructed with the entries retrieved by the AcQA parser. All the inputs are processed by the Phrase Analysis module of the Triplets techniques. For each input found, an entry is created in the KB with the actions, keywords, and question type. Then the knowledge base is constructed with the structure: <actions, keywords, question type, Answer>. The information stored in the KB is crucial for the information module to be able to extract and present concise answers to users.

Answer retrieval is the technique responsible for processing the information gathered in the Phrase Analysis module and presenting an answer to the user. Figure 22 depicts the steps necessary to find and handle the answer candidates. The analysis of Actions and Keywords is made looking for a direct match with the KB. After the previous phase, the Triplets techniques use an NLTK Stemmer package to get the base word. With the base word, the system tries to find synonyms to match more answers from the KB. A trust value is assigned to each answer retrieved in these steps. The search for answers that equal the question type is done first with a direct match and then with a similar question type. A trust value is assigned for each answer retrieved from the KB. All these steps are made to recover more answers from candidates that match the intents from user questions (stored as the triplet: actions, keywords, and question type).

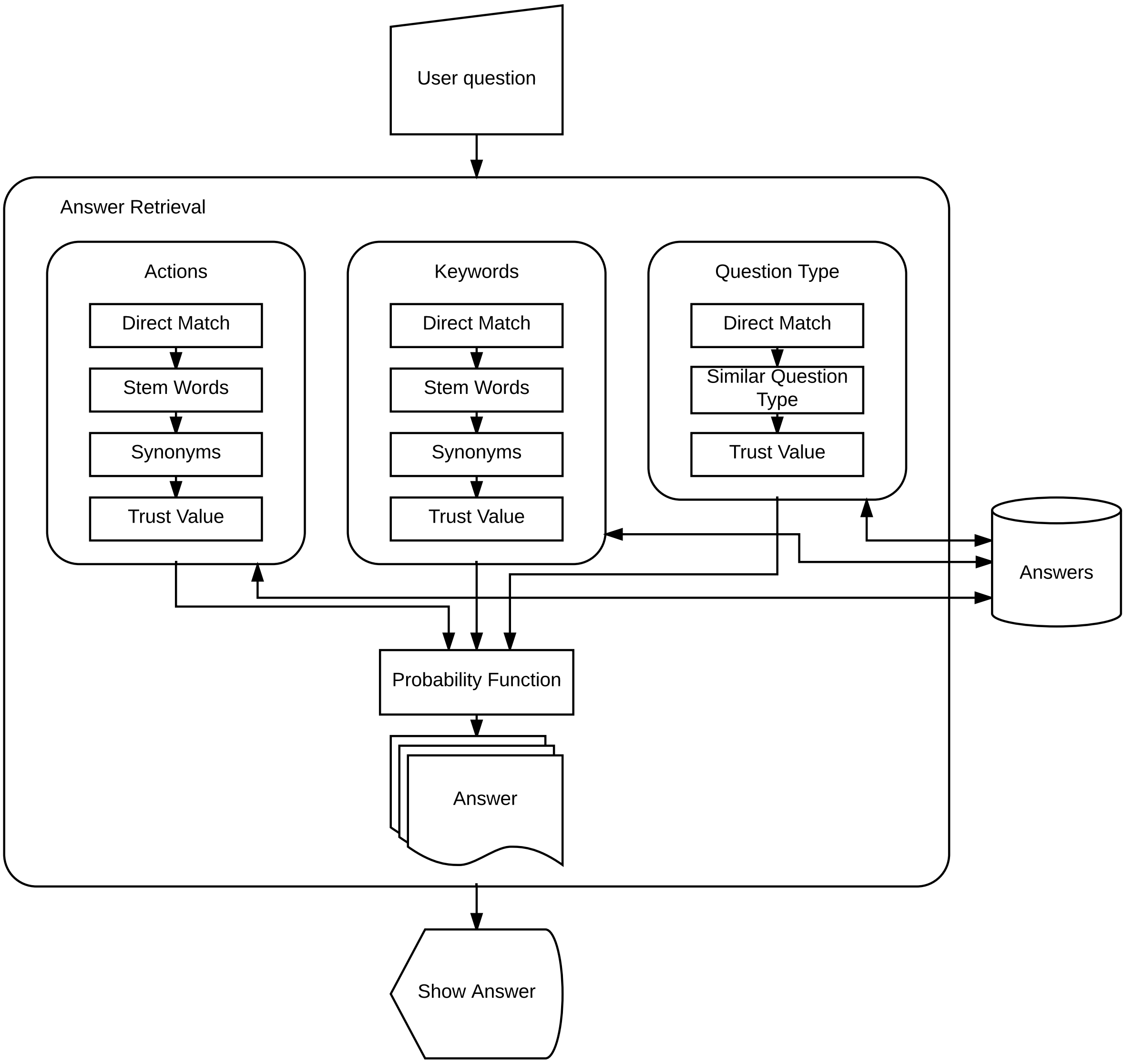


Figure 22: Answer Retrieval

With all these candidate answers retrieved, a probability function is applied to rank them and present to the user the most likely answer. The less probable answers are made available to the user if they are not satisfied with the answer provided by the system (the best ranked one).

5.3 Chapter's Considerations

In this chapter, the main concepts of the AcQA language were presented. This chapter discussed how the AcQA Engine generates and deploys the Q&A System. The AcQA processor generates an intermediary representation of the code written by the user and exposes this data so the AcQA Engine can generate the code needed to construct the Q&A System. After the correct execution of steps needed by the AcQA

Engine, the Q&A System is deployed in a Linux server. The deployed system is accessible through the UI defined in the AcQA specification.

6

Case Studies (CS)

This chapter contains three case studies presenting the use of the AcQA language in different scenarios.

Structure of the chapter. Section 6.1 introduces concepts of Community Q&A Sites needed to introduce the first two case studies: Board & Card Games Q&A System (Section 6.2) and the PythonQA (Python Q&A System) (Section 6.3). The third case study, to enable a robot to answer questions about classes and meetings time and location, is presented in Section 6.4.

6.1 Community Q&A Sites

Community Q&A Sites allows users to post questions and answers to questions already asked. StackExchange (SE) is an Online Social Question and Answering site which contains several Q&A subjects. Stack Overflow is one of the 166 Stack Exchange Community[1]. Python programming language is one of several languages discussed on the Stack Overflow site. The choice to use StackExchange in this thesis case studies among another Community Question Answer site (CQAS) is because of the public availability of the data and is regularly updated.

StackExchange works as a regular CQAS with users posting questions, answers, commenting, and voting positively or negatively in posts and comments. Users can only modify their posts, being the author of the question responsible for choosing an answer as a correct one. Users can register in the SE to be able to vote and maintain a reputation, gaining rights and badges based on the reputation.

6.2 CS1: Boards & Cards Games Q&A System Specification in AcQA DSL

This case study demonstrates theAcQA DSL use to generate a Q&A System to answer questions about Board & Cards Games. Section 6.2.2 presents the AcQA DSL syntax by specifying a board & card games Q&A System. The language syntax of AcQA was defined to be simple, allowing the user to create

[1] www.stackexchange.com

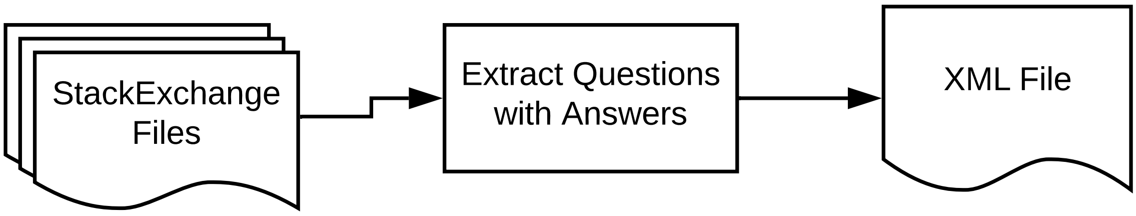


Figure 23: Steps needed for data cleaning and processing from SE

a specification of a Q&A System without the need to have prior knowledge of GPL's. The language syntax also allows the user to parameterize the techniques used to build the knowledge base and process answers and other parameters.

6.2.1 Overview of Board & Card Games

The Q&A System specified in AcQA is intended to answer questions concerning board & card games. This subsection gives an overview of the domain of the Q&A System. It also discusses the data used in the running example.

According to Gobet, Retschitzki, and Voogt 2004, board games are games with a fixed set of rules that limit the number of pieces on a board, the number of positions for these pieces, and the number of possible moves. There are several discussion groups on the Internet about these types of games; they allow users to ask questions about the rules and strategies or even questions of the games themselves. An example of one question about the game *Exploding Kittens* is *How many persons can play the original game?*. In this case study, we use data available from StackExchange (SE)[2] in Archive.org[3].

Board & Cards Games is one of the 166 Stack Exchange Community. The data handwrote into the system by community members is used as a base for the knowledge base of this case study.

SE provides an anonymized data dump facility through the Archive.org Site [4]. All user contributions are made over the Creative Commons cc-by-sa 3.0 license[5] allowing the data to be made available for any purpose, even commercially. We used in this work the data made available on 2019-03-04 and have approximately 40,7 MB of size and 31395 Posts (questions or answers).

The data from SE was pre-processed to extract Questions that have answers from the Posts XML file. Figure 31 details the steps necessary to process the data from the StackExchange and insert it in the Knowledge Base of AcQA generated Q&A System. Firstly questions that have answers are extracted from the posts file. After the extraction of all question → answer pairs, is created an XML file to be consumed

[2] www.stackexchange.com
[3] https://archive.org/details/stackexchange
[4] https://archive.org/details/stackexchange
[5] http://creativecommons.org/licenses/by-sa/3.0/

```
<question id="1">Playing exploding kittens and the game is unclear
about the effect of nope cards mid attack. If a player has already drawn
one of their attack and it's a nope, can they play it? What happens?
</question>
<answer questionid="1">While not explicity stated in the rules, my
understanding is that Nope reacts to a card (or pair, or triple) just
played, in which case once you've drawn the Attack has already
resolved and you can't Nope it.
</answer>
```

Figure 24: XML fragment of preprocessed Posts.xml from Stack Exchange Boards & Cards Games

later by the AcQA parser. This XML file is specified in AcQA to generate the KB of the Q&A System. Figure 24 shows a fragment of the output data pre-processed from the StackExchange input file, exhibiting a question about the game Exploding Kittens [6].

6.2.2 AcQA **Specification**

Figure 25 presents a code fragment of a specification written in AcQA DSL to generate a Q&A System for Board Games. Line 1 in Figure 25 specifies the file name (with a valid path on the developer operating system) and which parser is used to process and load the user data into the knowledge base. It is possible to set optional variables to determine a specific behavior of the parser. In this example, the techniques block was omitted, so the generated system uses by default the triplets techniques.

The UI's are specified in lines 2-4. It is obligatory to specify at least one parameter: the UI type. In lines 2-3, an HTTP UI is defined with some parameters of the UI: the title of Q&A System, the HTML used in section About, the admin credentials to access the admin page, and the URL used to configure the web server in the Linux server, respectively. In line 4, a setup of a RESTful web service is shown. The parameter security defines which type of security is used by the RESTful web service. In this example, the JSON Web Token (Beltran 2016) is used to provide authentication and authorization in the web service.

The code at line 5 (Figure 25) configures in which Server the Q&A System is deployed, and the first parameter is the hostname of the server. The last two parameters are the login information that is used to connect to the server through an SSH (Secure Shell) protocol Ylonen 1996. There is also an optional parameter to specify the server Linux distribution (Debian, Fedora, among others). Currently, the default

[6] https://explodingkittens.com/

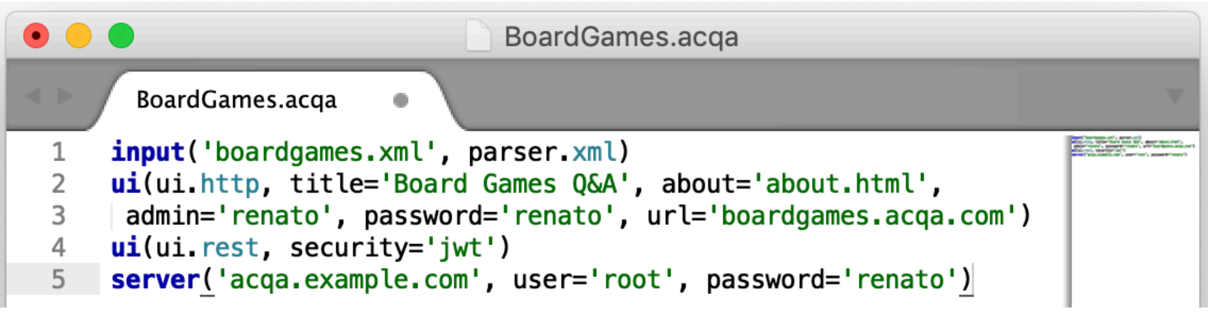


Figure 25: Q&A System specification in AcQA

value is Debian-like[7] and is the Linux server used in this case study.

The AcQA DSL allows the user to change the behavior of the whole Q&A System. For example, the user can change the language of the system to Portuguese instead of English using the parameter *language="portuguese"* inside the UI block. The changes are applied in tokenizer, POS (Part-of-Speech) tagger, lemmatizer, and Wordnet language.

The steps needed to generate the fully operational Q&A System are depicted in Figure 26. The AcQA grammar is processed by the compiler from AcQA to generate the AcQA Engine. The specification is written by the user and recognized by the AcQA compiler. The data from the user is imported through the Input Parser set in the AcQA specification. The AcQA Engine then generates a Q&A System specified by the user. The Q&A System is deployed into the server when AcQA Engine executes the Deploy Engine. After that, the Deploy Engine processes the User data to create the knowledge base used by the fully operational Q&A System.

When a User of the Q&A System accesses the provided URL in the AcQA specification (in our example: boardgames.acqa.com), an HTTP Web UI is displayed, as shown in Figure 27. Figure 29 displays the RESTful web service endpoint, with the included documentation needed to use the web service. Figure 28 shows the result from the user question: "How many turns can an attack card skip in exploding kittens?". It is presented an answer with 53.7% of confidence that contains an explanation with an image from the game website. This answer is part of the knowledge extracted from StackExchange and correctly answers the user question. If more than one answer is generated, the answers are presented to the user according to the ranking generated by the trust value. In the example presented in Figure 28 there is only one answer retrieved by the Q&A System.

6.3 CS2: PythonQA

PythonQA is a closed-domain Question and Answering system that answer questions about the Python programming language. As a closed domain Q&A System, PythonQA can provide concise answers rather

[7]https://debian.org

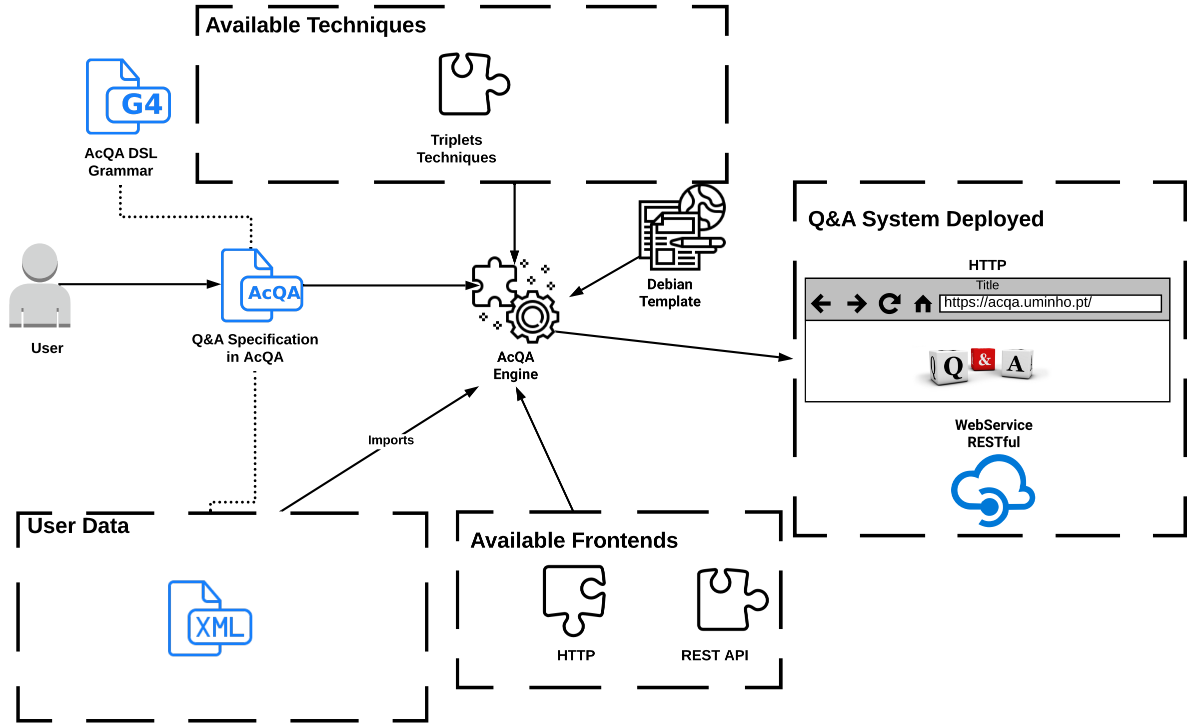


Figure 26: Steps to generate a Q&A System

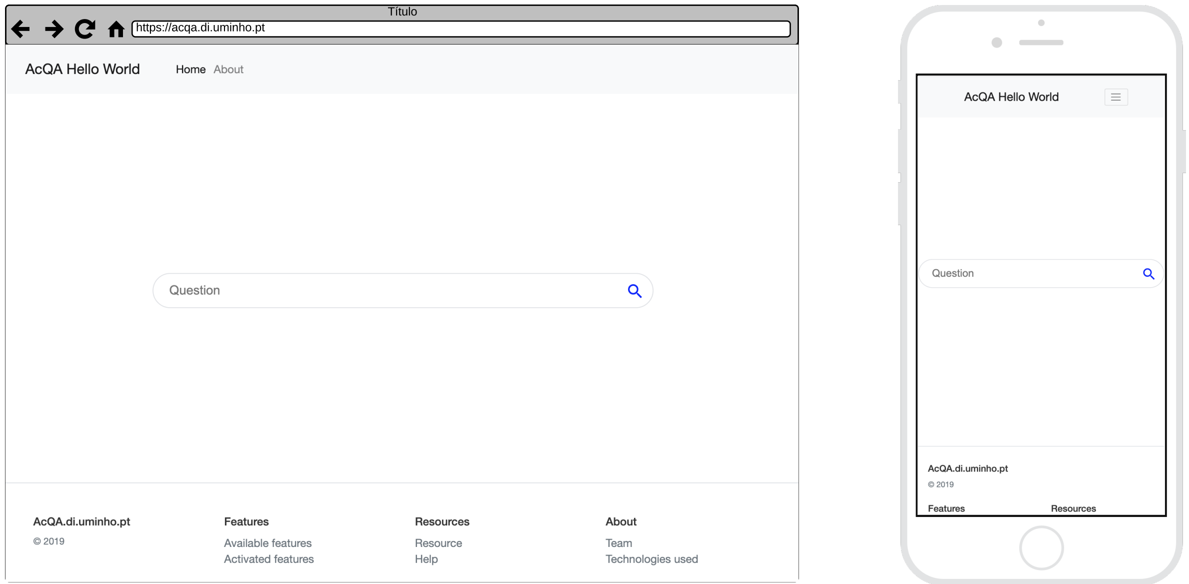


Figure 27: Screenshot of the Board Games Q&A System generated by AcQA

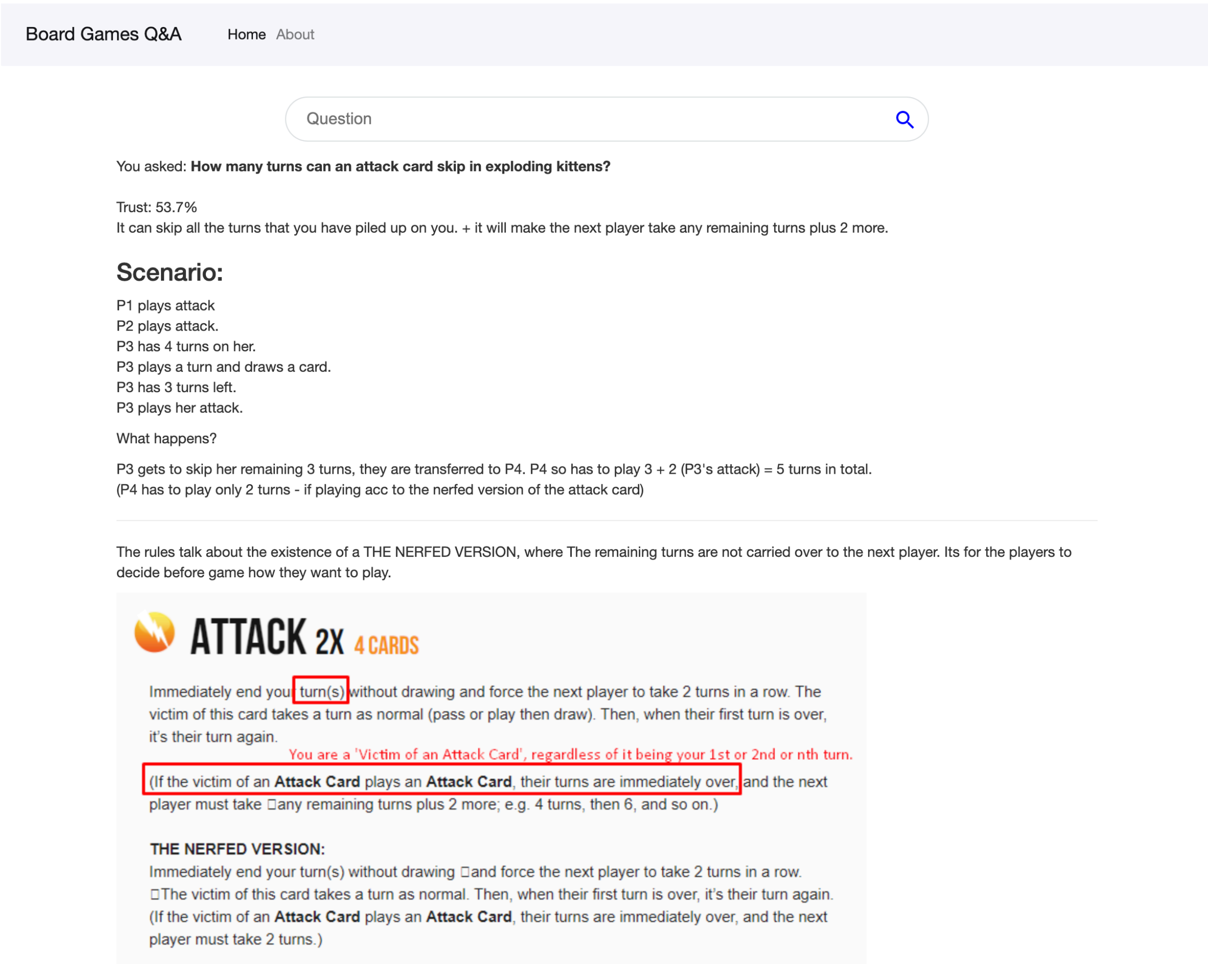


Figure 28: Screenshot of the Board Games Q&A System generated by AcQA

than a set of related documents, depending on the quality and size of the knowledge base.

Python has gained attention from the scientific community and programmers around the world, from both beginners and experienced programmers. Many Community Question and Answering Sites (CQAS) address the Python language because of the demand created by users who use the programming language regularly. Thus, Python was chosen as the domain of the Q&A System for this case study. Nonetheless, other languages such as Java, Haskell, or Julia could serve as the domain of the PythonQA without the need for structural changes. Some examples of this CQAS are StackExchange [8], Yahoo Answers [9], among others.

The architecture of PythonQA is shown in Figure 30. The system receives a question from the user and sends it to the Question Analysis module. In this module, the question is parsed to produce a query that will be used to retrieve relevant information from the knowledge base. The information is processed in the Answer Retrieval module to compose the final answer.

[8] https://stackexchange.com
[9] https://answers.yahoo.com/

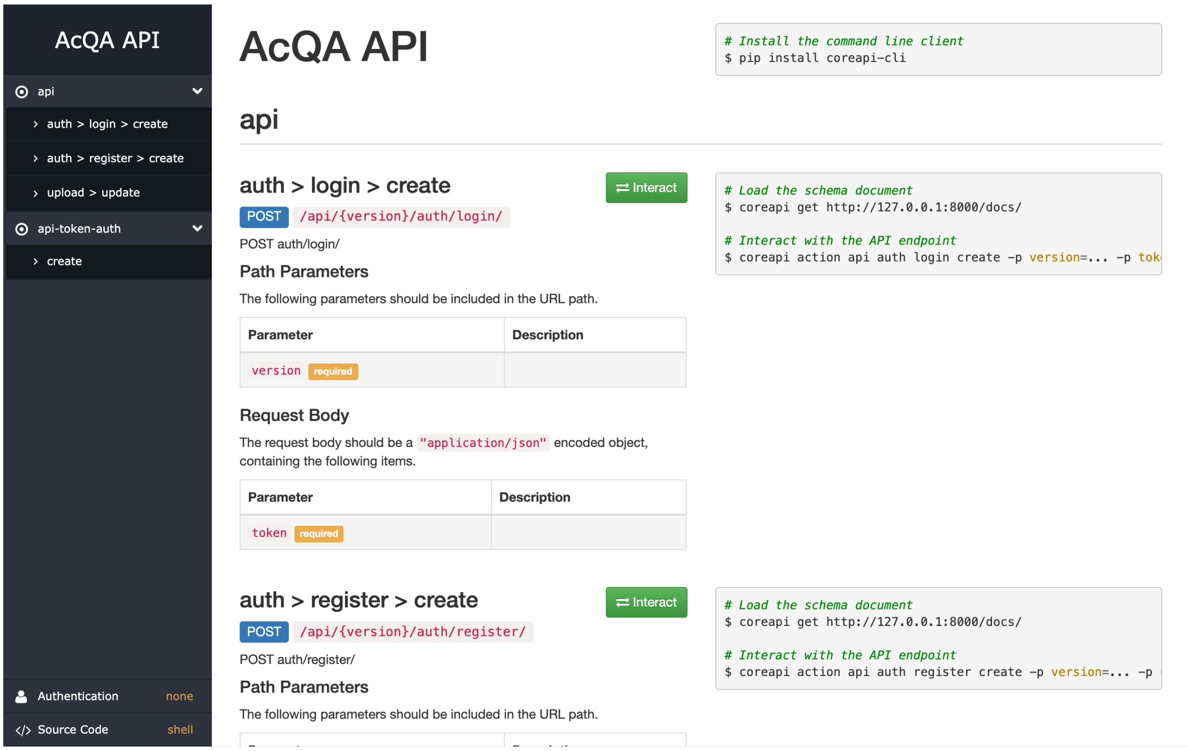


Figure 29: Screenshots of the API endpoint of the Board Games Q&A System generated by AcQA

The PythonQA system was developed using the AcQA DSL, using as a knowledge base the python frequently asked questions.

6.3.1 Extending PythonQA with Knowledge from Stack Overflow

The PythonQA was able to return satisfactory answers, but the Knowledge Base is too narrow. The only source of knowledge is extracted from the Python Frequently Asked Questions. The PyFAQ has only 169 pairs of Question-Answer Ramos, Pereira, and Henriques 2017, restricting the knowledge of the system.

To increase the KB, we have to choose between CQAS such as StackExchange[10] or Yahoo Answers[11]. We decided to extend the PythonQA with data from the StackExchange because of the public availability of the data, as well as being regularly updated.

The data is available as a direct download through the Archive.org Site[12]. The size of all compressed datasets is approximately 40 GB. Each SE file has at least 8 XML files: Votes, Tags, Users, PostLinks, Posts, PostHistory, Comments, and Badges. The Users file contains the information about the users, like Display Name, Creation Data, and other information. The Badges file includes a relationship between badges and users. Tags used in the SE are inside the Tags file. The contents of the questions and answers are in the

[10] www.stackexchange.com

[11] answers.yahoo.com

[12] https://archive.org/details/stackexchange

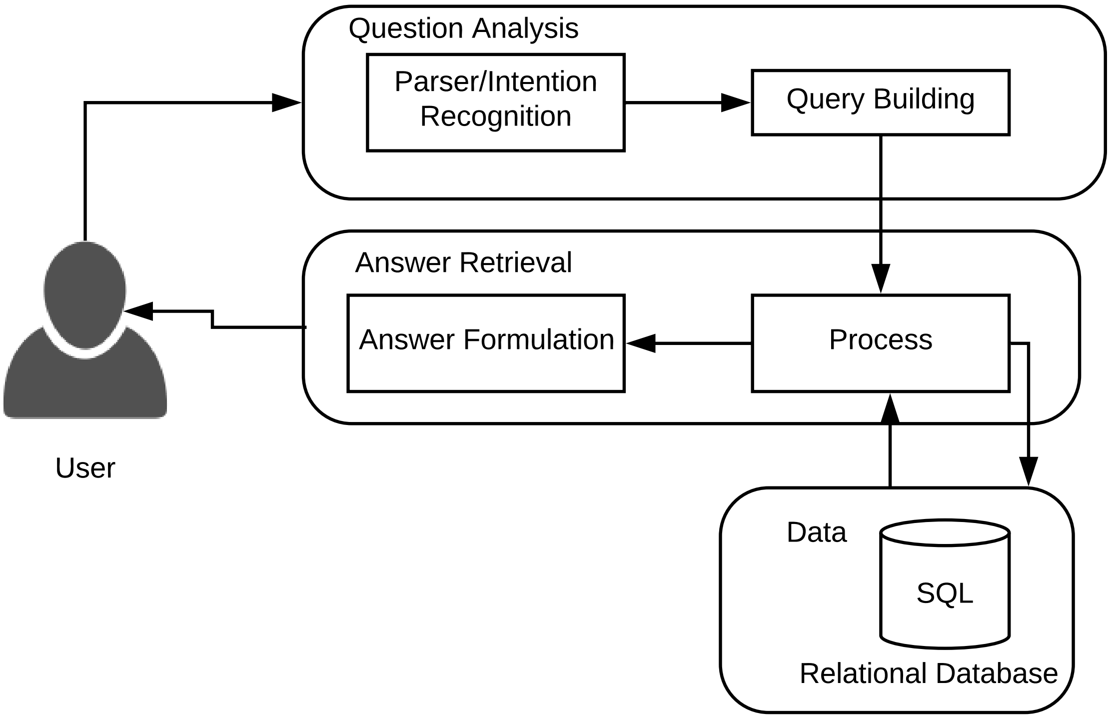


Figure 30: PythonQA Architecture

Posts file. This XML file defines if the post is a question or an answer, the creation date, page views, score, owner, title, and the content of the question. The Comments file contains comments produced by users of SE about the questions and answers.

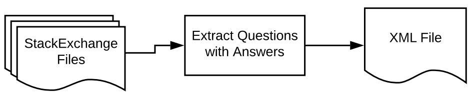


Figure 31: Extending PythonQA

Data from StackOverflow was downloaded to create the knowledge base of PythonQA. Figure 31 detail the steps necessary to process the data from the StackExchange and insert it in the Knowledge Base of PythonQA. Firstly is extracted the questions that have answers from the Posts file. Next, only questions and answers with the python tag associated with the pair Question $\longrightarrow$ Answer are selected. After the extraction of all Question $\longrightarrow$ Answer pairs, an XML file is produced to serve as input data to the AcQA data module (parser). The last process is to upload the data to the Q&A System, so the AcQA generated code can be able to generate the knowledge base through the phrase analysis module, as already discussed in Section 5.2.3.1.

The generated Q&A System was able to handle 480 thousand Question $\longrightarrow$ Answer pairs. These questions and answers were uploaded to the Q&A System and processed in the background, allowing the use of the PythonQA during the construction of the knowledge base.

Some preliminary tests were made, with ten random questions extracted from StackOverflow that were not imported to the knowledge base. The original KB was only able to correctly answer 20% of the analyzed questions, while the extended KB successfully fulfilled 80%. This result was due to limited information on the original KB. When looking in the alternative answers, the PythonQA with the extended KB was able to provide the correct answer in 50% of the unanswered questions on the first answer alternative. The extended PythonQA presented more details in the answers, providing solutions that contained code fragments and links to more relevant information. The extraction of more detailed answers was possible because the information available in StackOverflow is curated by a large community of developers. For instance, with the following questions: q1: "How can I create a stand-alone binary from a Python script?", and q2: "How do I validate an XML against a DTD in Python". The q1 is correctly answered in PythonQA with original and extended KB. However, with the question q2 only with the extended KB, a relevant answer is presented.

6.3.2 PythonQA Q&A System Specification in AcQA DSL

Figure 32 presents the specification written in AcQA DSL to generate the PythonQA Q&A System. This case study specification is very similar to the specification written in the case study discussed in Section 6.2. Line 1 in Figure 32 defines the data that is used to create the knowledge base of the Q&A System and which parser is used. In this case study, the AcQA language processed the specification more than one time: the first run was to process the data from the python frequently asked questions, and later to import the data originated from the StackOverflow site (Figure 33). In both times, the data was generated in XML, so for this reason, the parameter parser was *parser.xml*. Figure 33 besides the different input data also have the *nodeploy* command (Line 5), configuring the AcQA Engine to not reinstall the resulting Q&A System, only to upload and process the new input data (with questions and answers from StackOverflow).

As PythonQA used the default triplets technique, the technique block was omitted. This Q&A System was deployed with only the HTML5 user interface, as specified in lines 2-3. The parameters in the UI block define the username, password of the administrator, and the URL that the resulting Q&A System is deployed.

The last line in the specification (line 4 in Figure 25) defines the server information where the PythonQA is deployed.

```
1   input('faqdata.xml', parser.xml)
2   ui(ui.http, title='PythonQA',  admin='renato',
3       password='renato', url='pythonQA.farpa.org.br')
4   server('172.25.10.50', user='root', password='acqapasswd1')
```

Figure 32: Specification of PythonQA using the python FAQ as KB, written in AcQA

```
input('stackoverflowdata.xml', parser.xml)
ui(ui.http, title='PythonQA',  admin='renato',
    password='renato', url='pythonQA.farpa.org.br')
server('172.25.10.50', user='root', password='acqapasswd1')
nodeploy
```

Figure 33: Specification of PythonQA using the StackOverflow as KB, written in AcQA

6.4 CS3: Where is my Class?

This case study shows the use of the AcQA DSL to produce a Q&A System to answer questions about where and when are classes and meetings to students on a university campus. Some extensions were made to the AcQA user interface to answer questions asked by voice instead of by text. A mobile application was made to allow students to interact with a robot in the halls of the university buildings. This application is detailed in Section 6.4.4.

6.4.1 Description of Where is my Class? case study

Students need to know where and when are the classes or meetings occurring on the campus. To be able to answer these types of questions, a Q&A System was proposed. *Where is my Class?* is a Q&A System that allow students or university staff to use their voice to discover in which rooms or buildings are some specific class or meeting. This study case has to deal with the following requirements: temporal data and students speaking several languages.

The rooms schedule is made available every week and comprises a one-week data span, so the Q&A System knowledge base has to be updated every week. The users (students and university staff) also do not usually use a specific day in the question if the question is for the location of an event occurring on the same day. For example, to know the location of the class *Algorithm I*, the user can ask *Where is the class algorithm I?* or can ask *Where is the class algorithm I on Monday morning?*. The Q&A System generated by the AcQA DSL allows the developer to deal with this situation. If temporal elements are not present in the question, the Q&A System automatically insert the day and day-shift so that the answer can be correctly answered.

The students and university staff are from several countries around the world, so the language is a relevant requirement in this study case. The system has to answer questions in a variety of languages. To be able to allow the internationalization of the recognized languages, an API can be used in AcQA specification.

6.4.2 AcQA Specification

Figure 34 presents the specification written in AcQA DSL to generate the *Where is my Class?* Q&A System. This case study specification is also very similar to the specification written in the case studies discussed in Section 6.2 and 6.3. Line 1 in Figure 34 defines the data that is used to create the knowledge base of

```
1 input('ipb.json', parser.json)
2 ui(ui.http, title='Where is My Class?',  admin='useradmin',
3     password='password', url='whereismyclass.farpa.org.br')
4 ui(ui.rest, security='jwt')
5 server('172.16.10.50', user='root', password='acqapasswd1')
```

Figure 34: Mobile version written in flutter and deployed on an iPhone device

the Q&A System and which parser is used. This Q&A System was generated with the data available by a JSON file, so for this reason, the parameter parser is *parser.json*.

This Q&A System also used the default triplets technique, information that can be inferred in the specification as there is no techniques definition. The UI's defined in the specification are HTML5 and REST, as specified in lines 2-4. The parameters in lines 2-3 the UI block define the username, password of the administrator, and the URL that the resulting Q&A System is deployed. In line 4, the JSON Web Token is defined as an authentication mechanism.

The last line in the specification (line 5 in Figure 34) defines the server information where the PythonQA is deployed.

6.4.3 Data Importing

The University Information Technology (IT) department provides information about the rooms allocation in their proprietary JSON format. A conversion is needed to use the university JSON data format as input data to generate the knowledge base. The data contains information about rooms, professors, classes, and meetings. A pre-processing step is needed to generate pairs of questions and answers to use as a seed to generate the knowledge base.

A parser was developed to use as the input the data available from the university and generates as the output data understandable by the AcQA parser.

The created parser connects to the REST API generated by the AcQA Engine. It automatically updates the knowledge base of the Q&A System to use the new data when available from the university IT department.

6.4.4 Generated Systems

To achieve the requirements of this study case, the Q&A System generated by AcQA produces an HTML5 user interface and generates code in the flutter language to allow the deployment in mobile devices and a robot running Android.

The HTML5 user interface is generated in the same way as in the previous case studies. In this case, the only difference is that an option is defined in the UI to allow the resulting Q&A System to pre-process the user question and add temporal data if the question misses this information.

The mobile code generated by the AcQA Engine is written in the flutter framework, thus allowing the deployment in the Android and the Apple stores and devices. The generated mobile code also allows the

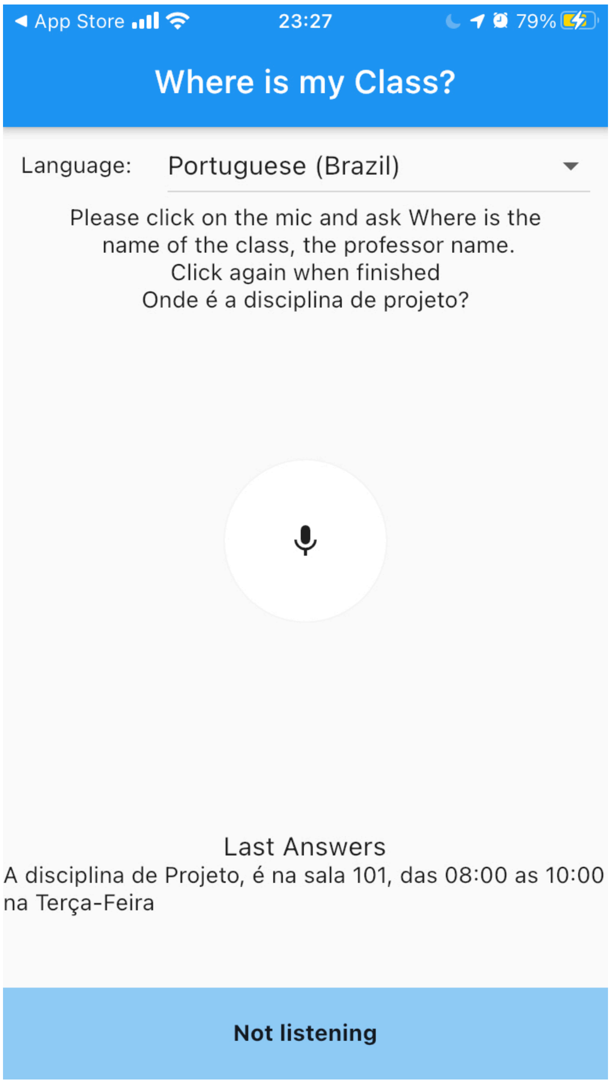


Figure 35: Mobile version written in flutter and deployed on an iPhone device

use of the localization and voice recognition of the running device, thus not needing to process the input data across some API to convert the voice into text.

The interface of the mobile version is similar to the HTML5 version of the generated Q&A System. Figure 35 shows the user interface on the mobile version of the generated system.

6.5 Chapter's Considerations

In this chapter, three case studies were presented to the reader. Three different Q&A System were described and deployed using the AcQA DSL. The AcQA language was able to construct the Q&A Systems according to the specifications. It made it possible for the developer to add data in the knowledge base using the exposed REST web service. One of the main benefits of deploying an Q&A System using the AcQA DSL is that the infrastructure needed to run all the required services can be deployed inside the developer employee premises, not needing to use third parties-providers to be able to deploy the Q&A System.

7 Assessment

An experiment was designed and conducted to assess the use of AcQA language and test the performance and usability of the resulting Q&A System. This experiment aims at recognizing if it is feasible to use a language like AcQA to create in a short time and with a minimum effort Q&A Systems.

Structure of the chapter. Section 7.1 present the experiment design, Section 7.2 present information about the participants of the experiment. The hypothesis definition is made in Section 7.3. The survey with the questions asked to the participants is presented in Section 7.3.1, and the results of the experiments are discussed in 7.4.

7.1 Experiment Design

A complete infrastructure was developed to allow the use of a graphic editor (SublimeText) to write the AcQA specification and a Linux server to deploy the resulting Q&A System, enabling participants to use and test the AcQA language.

The AcQA language was introduced to the participants through a tutorial on the development and deployment of a complete Q&A System. The tutorial contains a complete description of a specification written in AcQA to generate a Q&A System on the domain of *Board & Cards Games*, similar to the description provided in Section 6.3. It is described the input file needed to create the knowledge base of the Q&A System. As input file, the participant can choose among seven ready to use XML files extracted from Stack Exchange data dump (as was the examples presented in Sections 6.2,6.3):

- cooking.xml: Seasoned Advice is a question and answer site for professional and amateur chefs
- diy.xml: Home Improvement Stack Exchange is a question and answer site for contractors and serious DIYers.
- fitness.xml: Physical Fitness Stack Exchange is a question and answer site for physical fitness professionals, athletes, trainers, and those providing health-related needs.

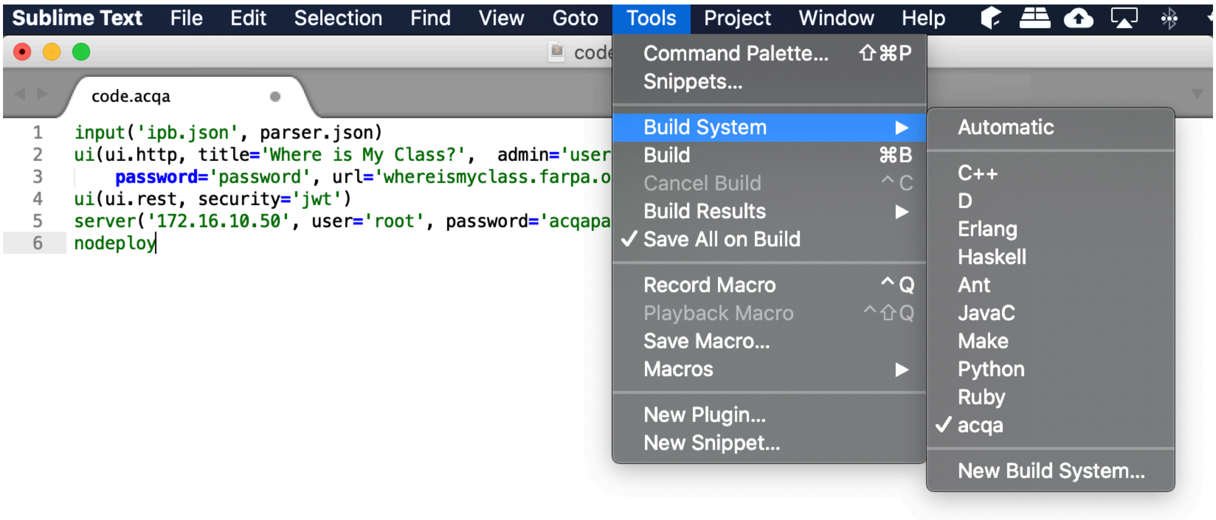


Figure 36: Support to the AcQA language inside the SublimeText editor

- hardware.xml: Hardware Recommendations Stack Exchange is a question and answer site for people seeking specific hardware recommendations.
- lifehacks.xml: Lifehacks Stack Exchange is a question and answer site for people looking to bypass life's everyday problems with simple tricks.
- mechanics.xml: Motor Vehicle Maintenance & Repair Stack Exchange is a question and answer site for mechanics and DIY enthusiast owners of cars, trucks, and motorcycles.
- parenting.xml: Parenting Stack Exchange is a question and answer site for parents, grandparents, nannies, and others with a parenting role.

A Windows Server was deployed to provide the participant with a fully configured Integrated Development Environment (IDE). This Windows Server exposes a remote desktop server accessible through the remote desktop connection application, allowing the participant to connect to a fully functional remote desktop. The user has in the desktop folders containing the datasets available to use in the deployment of the Q&A System. It also has a shortcut to a fully functional Integrated Development Environment (IDE) based on Sublime Text Editor, configured to provide support and syntax highlighting for development under AcQA eco-system.

The support for the AcQA language inside the SublimeText editor was made to enable syntax highlighting and code coloring, making easier the understanding of the code written in AcQA by the developer. The AcQA support into SublimeText also contains the tools to run the AcQA language through the build tools of the editor. Figure 36 shows the support to AcQA eco-system in the SublimeText editor.

The remote desktop server also has the AcQA language installed for all system users, eliminating the need for the developer to install anything on their computer. The remote access server makes it possible

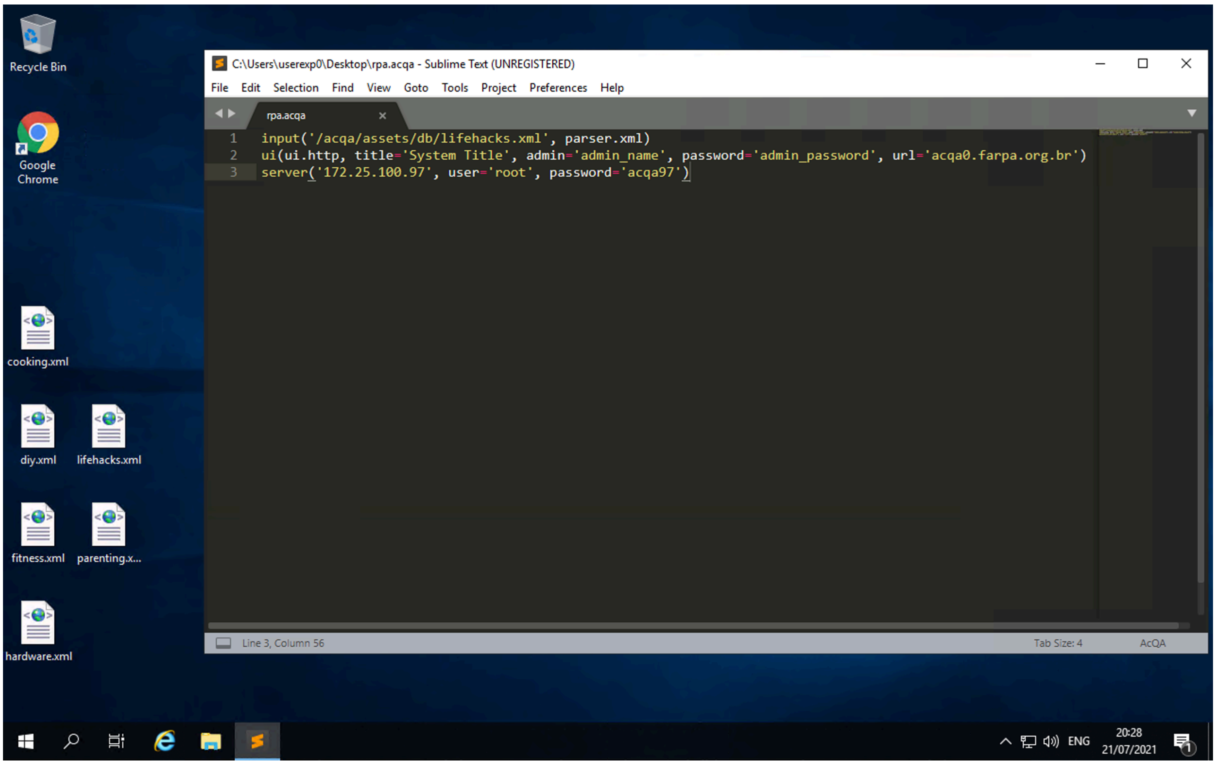


Figure 37: Desktop available to the participants

for the participant to access the remote desktop to write specifications in the AcQA language. Figure 37 presents the desktop available to the participant when accessing the Windows Server.

There is also another option for the participants to code and use the AcQA language, which is using an HTML5 editor developed to recognize the AcQA eco-system. This developed editor also has support for AcQA syntax highlight, code completion, and code coloring. Figure 38 presents the developed HTML editor for the AcQA language. This HTML editor was implemented using the following technologies:

- Server side:
  - Python
  - Flask framework
  - Nginx WebServer
  - gUnicorn
  - AcQA language and utilities
- Client side:
  - JavaScript
  - Bootstrap library

Figure 38: Support to the AcQA language inside a developed HTML editor

- jQuery library
- Socket.io library
- Resumable library
- Monaco-editor Editor

The AcQA eco-system was implemented into the HTML editor and into the server that executes the AcQA editor. With the support build for the AcQA eco-system, the user can create and run AcQA code direct inside a modern web browser.

When the participant access the experiment URL, a Linux server is automatically deployed inside a virtualization system to allow access as a superuser for the participant. Credentials to a clean install of a Linux server (Ubuntu Server) are also provided to each participant to allow that a Q&A System can be deployed and tested by the experiment participant.

The participants are asked to write a specification in AcQA and run the code to deploy a Q&A System. After the execution of the AcQA specification and the resulting system is deployed with success, the participant can access the Q&A System to ask some questions. The Linux server is accessible on the internet, allowing the participant to use the Q&A System on their computer or mobile device.

After writing the AcQA specification and after building and testing the Q&A System generated by AcQA Engine, the participants were requested to answer a survey organized in four parts: section one contains a questionnaire that gathers information about the participants' prior experience and academic background; section two collects information about the participant's programming experience; section three asks the subject about his experience in the design and development of Q&A Systems; and finally, Section four enquires the participant about AcQA usage. The questionnaire is detailed in Section 7.3.1. The survey was implemented using the Django framework and the library Django-survey-and-report. This library so we can guarantee some requirements, such as preserving anonymity.

The experiment described here is accessible at https://acqa.di.uminho.pt/experiment/. It presents the tasks needed to evaluate AcQA language. It is available to anyone who wants to participate and send feedback about the experience, thus helping develop the language.

7.2 Participants

The participants in the experiment received a description of a possible scenario within which they were supposed to create, resorting to AcQA, to generate a Q&A System to answer questions about a specific domain.

This experiment was applied to people with distinct education levels, reaching undergraduate, M.S., or Ph.D. students, masters, and doctors. All the participants are from the computer science area and have prior programming experience, ranging from beginners to experts. There were seventeen participants that successfully carried out the experiment and answered the survey.

7.3 Hypothesis definition

The experiment was planned to understand if the following research questions (RQ) are true:

- **RQ1**. Does the AcQA usage help to understand Q&A Systems design?
- **RQ2**. Does the AcQA usage affect the time required to deploy a Q&A System?
- **RQ3**. The tools provided with AcQA can successfully assist the user in the development of the Q&A System?
- **RQ4**. Can AcQA effectively help the user in the deployment of the Q&A System?

7.3.1 Questionnaire

A survey was made to collect information about the participants and how AcQA performed in the experiment. The questionnaire was divided into four sections to gather different information from the participants: information about the participants' prior experience and academic background; information about the participant's programming experience; information about his experience in the design and development of Q&A Systems; and finally, section four enquires the participant about the AcQA usage and the resulting Q&A System.

The information regarding the academic background was the highest educational qualification, in which area the participant acquired the qualification and the current occupation. The last two questions are not mandatory.

The second section of the questionnaire questions the participant about their prior experience with programming languages and if they ever developed a Q&A System. The only mandatory question is if the participant has ever written a computer program using a programming language. In this section, there are other questions: which programming languages (including languages for special purposes like SQL or Matlab) have the participant used so far; how would they rate the level of their programming skills; how interested are they in programming in general; and if they have experience with development of Q&A System.

The section about the experience in the design and development of Q&A Systems gathers information about the usage of technologies needed to develop these systems, like natural language processing, artificial intelligence, or even proprietary technologies. The only mandatory question is if the participant already used some of these technologies.

The fourth section of questions is requested to be answered only after ending the experiment. It contains Likert Robinson 2014 scale questions about using the AcQA language and general-purpose languages experience. This section has the following mandatory questions:

- Q1: Concepts of Q&A Systems can easily be specified with the AcQA language.
- Q2: Concepts of Q&A Systems can easily be specified with a GPL language.
- Q3: The AcQA DSL seems simple to use.
- Q4: Developing a Q&A System in any GPL seems simple to use.
- Q5: AcQA programs are easy to understand.
- Q6: GPL programs are easy to understand.
- Q7: AcQA seems too technical to specify concepts of Q&A Systems.
- Q8: GPL seems too technical to specify concepts of Q&A Systems.
- Q9: It is easy to make changes to existing AcQA programs.
- Q10: It is easy to understand the meaning of AcQA source code quickly.

The last two questions in the fourth section of the questionnaire asked the participant about difficulties using the AcQA language and suggestions to improve AcQA. These questions are optional.

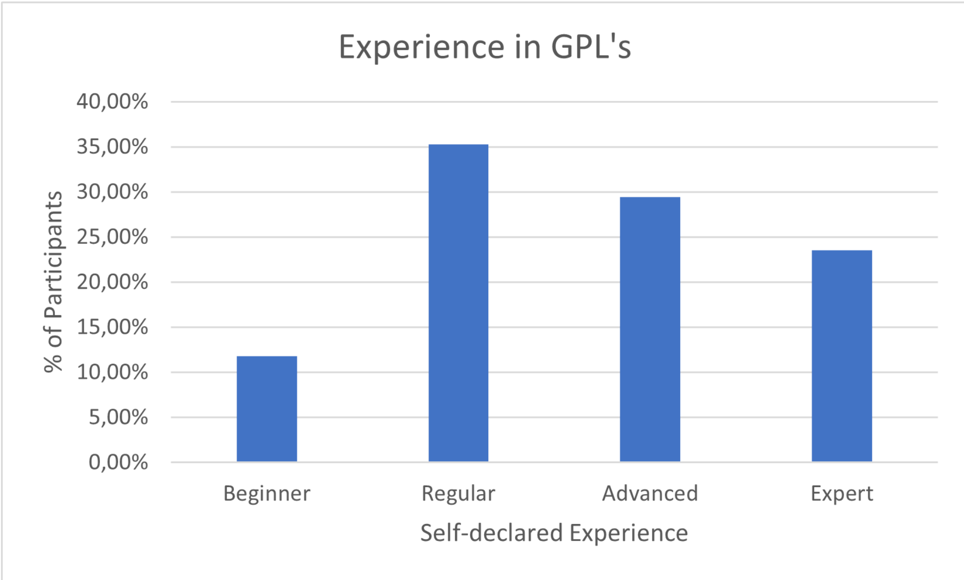


Figure 39: Chart describing the participants prior experience with programming in GPL

7.4 Experiment results

As said above, the described experiments were conducted involving seventeen participants. There was one undergraduate student, four Ph.D. students, three master's students, six masters, and three Ph.D. The participants are from the computer science area and have programming experience, ranging from beginners to experts, as presented in the graph in Figure 39. According to the answers, the majority of participants self-declared as beginners concerning the development of Q&A Systems. Figure 40 shows the chart with the percentage of the respondents according to the prior experience developing Q&A System.

The participants have heterogeneous experience with programming languages, with some knowing a few GPL and some knowing some DSL. Figure 41 shows the languages known by the participants and the percentage of the participants who know that languages. The most mentioned language was the DSL SQL.

The answers provided by the participants in this experiment were subjected to a reliability test using the Cronbach alpha scale. As the values for the Cronbach alpha scale computed were higher than the threshold of 0.66, the reliability of the measuring instrument can be confirmed.

Figure 42 shows a graph with the answers to the questionnaire, quantified on a 1-5 scale (Likert-scale), and used as a base to calculate the research questions.

Table 9 shows the average, median, and standard deviation of the research questions quantified on a 1-5 scale. The research question 1 (Does the AcQA usage help to understand Q&A Systems design) got an average rate of 4, mainly because the respondents did not have prior experience (52.94% of respondents) or are beginners (35.29%) developing Q&A Systems. Only 5.88% of participants stated that they were regular Q&A System developers and 5.88% self-declared as experts.

Analyzing the experiment outcomes and the answers collected from the seventeen questionnaires, it

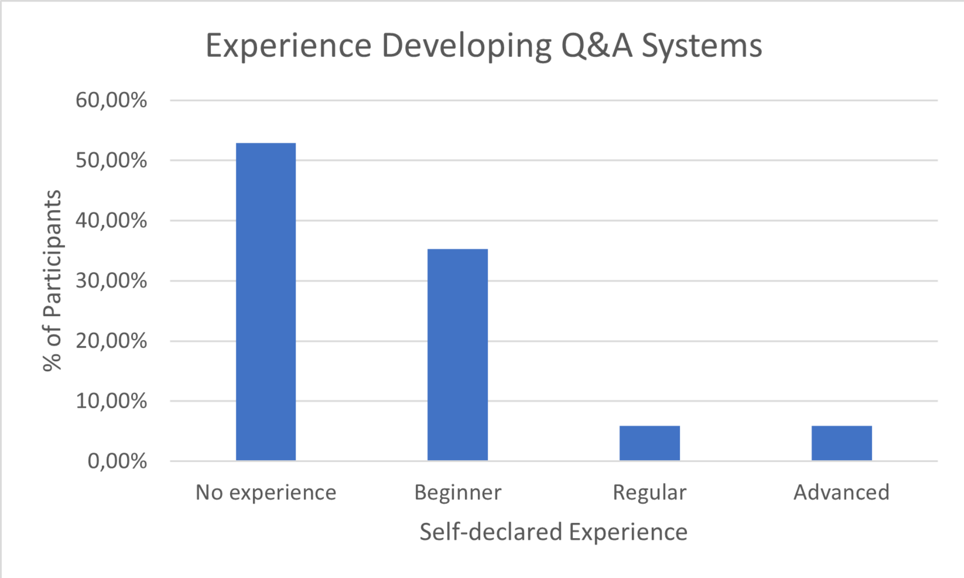


Figure 40: Graph describing the experience in developing Q&A Systems answers from participants of the experiment

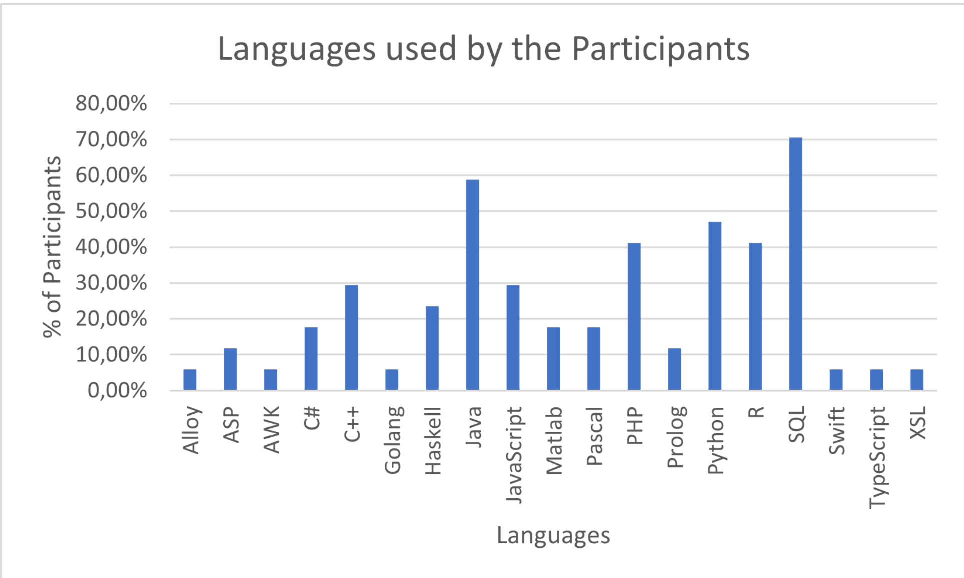


Figure 41: Graph presenting the programming languages known by the participants

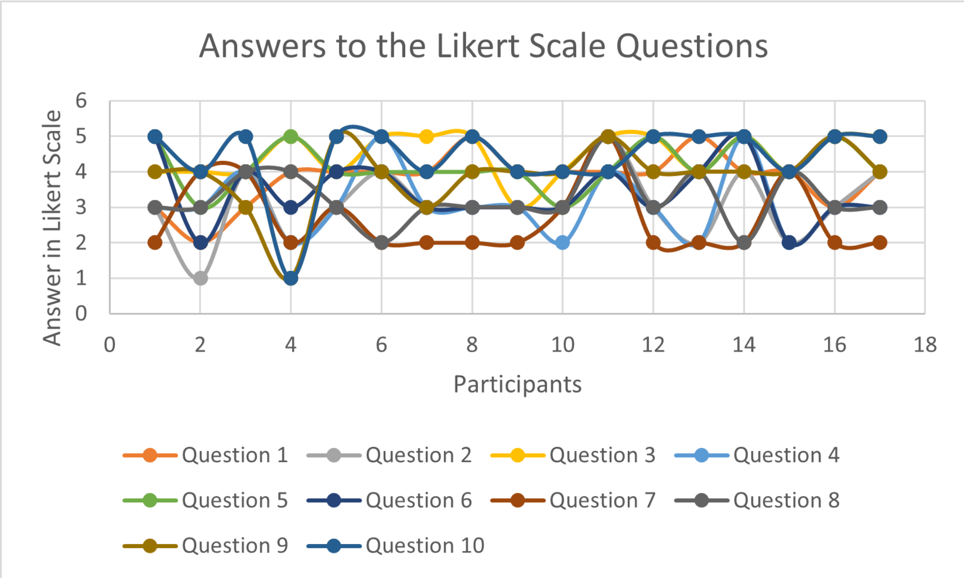


Figure 42: Graph presenting participants answers to the ten likert-scale questions

is fair to conclude that the four research questions were positively confirmed: (1) the exercise of writing a specification in AcQA DSL contributes for a clear understanding about the design of a Q&A System; (2) using AcQA specification language and engine, the time required to deploy a Q&A System is reduced; (3) the editing tools provided with AcQA aid the user during the development of a Q&A System; (4) resorting to AcQA system, the deployment of a Q&A System becomes more accessible and faster.

Table 9: Statistics about the answers to the research questions (N=17)

	Average[1]	Median	St. Dev.
RQ1	4.00	4.00	0.50
RQ2	3.66	3.66	0.50
RQ3	3.91	4.00	0.41
RQ4	4.23	4.50	0.49

The participants answered two optional questions at the end of the survey, asking what the difficulties occurred when using the AcQA language and what suggestions to improve AcQA they suggest. Two participants had some difficulties understanding the included datasets, with one suggesting that the experiment tutorial should be revised and improved to provide more details about the datasets. One participant stated that they had no difficulty using the AcQA language, writing that the language is pretty straightforward for people that are familiar with computer syntax. In the suggestions questions, some participants had written answers. Some participants suggest improvements on AcQA support into the SublimeText, making different colors for the syntax highlighting. One recommended that the language accepts additional sources of data, making the content more reliable. The AcQA language already supports several

[1] A five-grade scale, starting from strongly disagree (1) to strongly agree (5) was used in the questionnaires

types of input data, not only the example datasets provided in the experiment. One user also stated that writing code in AcQA was very straightforward, being the AcQA language very easy to understand and to deploy the Q&A System.

Some participants stated that the support tools provided (syntax highlighting and build help on a code editor) made it easier to program and understand the AcQA code.

The respondents think that AcQA helps design a Q&A System and decreases the time required to deploy these systems. Respondents also gave a positive answer that AcQA helps to understand Q&A System design.

7.5 Chapter's Considerations

In this chapter, experiments made with participants were discussed. The experiment and tools developed to support AcQA developers were made available to the participants and presented in this chapter.

The experiment results, described in Section 7.4, present the analysis of the respondent's answers to a questionnaire. This analysis has confirmed that the AcQA language can support the development of Q&A Systems, making it easier for developers to deploy a functional solution. The heterogeneity of the participant's knowledge also supports this statement. All the research questions were positively confirmed using the Cronbach alpha scale reliability test.

8

Conclusion

This thesis presented the domain-specific language AcQA, which allows the specification of a Q&A System. The specification in AcQA does not require extensive knowledge of general programming languages from the developer, leaving the effort to be focused on the data upon which techniques are used in the generated Q&A System.

Three case studies were presented in Chapter 6 to show the viability of the proposed approach and expressiveness of AcQA language. These case studies describe the domain and how to use the proposed AcQA language. Two Q&A Systems were created to answer questions from data obtained from the StackExchange CQAS. These case studies described how to create a Q&A System to answer questions about Board & Cards Games and about the Python GPL. The result systems successfully were able to answer questions about the proposed domains. One third case study presented a system to answer students from a university about classes and meeting times and places. This case study also successfully dealt with internationalization being able to answer questions in several languages.

The discussion presented in Chapter 7 presented the AcQA language to several participants to evaluate the feasibility to develop Q&A System using the proposed DSL. The results corroborated that using AcQA to develop Q&A System is possible and even more accessible than using GPLs.

8.1 Discussing objectives and results

The main objective discussed in this Ph.D. thesis was to create a language that allows developers to be able to build closed domain Q&A Systems automatically, using a formal specification. Five specifics objectives were made to achieve the main objective. Next, we revisit these objectives, and the main results are presented.

- *Choose a generic architecture (among the existing ones or defining a new one) that can always be adopted to build a closed domain Q&A System*

  The generated code from the running of the AcQA language, presented in details in Chapters 4, 5 used a generic architecture (Figure 2) to serve as a base of the resulting Q&A System. Section

5.2 describes the techniques and technologies used to generate code and deploy a functional Q&A System. The techniques available to process natural language are also described in detail in Section 5.2.3.1, and used in the case studies presented in Chapter 7.

- *Identify what components are stable in order to understand which information needs to be specified in each concrete case.*

  The study conducted in Chapter 2 supported the definition of the elements required in the DSL. The information collected in this study helped define which elements are mandatory to develop a Q&A System, and which ones are helpful to the developers.

  With the required and optional components to specify a Q&A System defined, the definition of the AcQA DSL was possible.

- *Define a DSL that allows an end-user to specify the issues that need to be described to build a specific system.*

  To tackle this objective, the information collected in the review of Q&A System was used to apply the lifecycle of developing a DSL, as discussed in Section 3.3. A DSL named AcQA was proposed to generate Q&A System through a formal specification written in that language. AcQA DSL is an external DSL and aims to have a simple syntax, yet allowing powerful customization of the resulting Q&A System. The syntax of AcQA is presented in Chapter 4.

- *Develop a system that analyzes descriptions written in that DSL and resorting to standard components generate the desired Q&A System.*

  To allow the processing of the AcQA language and achieve this objective, a set of tools was made to process specifications written in AcQA, as discussed in Chapter 5. The tools that support writing specification in AcQA language are the AcQA Processor and AcQA Engine. These tools are responsible for processing and transform a specification written in AcQA, generating code (AcQA Processor), and deploy the result code into a destination server (AcQA Engine).

  Along with the required tools to process AcQA specifications, some supporting tools were developed to assist developers in writing code in AcQA and deploying the Q&A System into a destination Linux server. Support for the AcQA language was implemented into the SublimeText editor. Developers have code highlighting and code completion, assisting the process of writing AcQA code. To allow the development of AcQA without requiring any setup or software installation, a web editor was developed to assist the users of the AcQA language. This web editor also has the code highlighting and code completion, and suggestion. The supporting technologies were discussed in Section 7.1.

- *Validate with concrete case studies the approach proposed and the developed engine to process the AcQA language.*

To be able to achieve this objective, some case studies (three in Chapter 6) were created. The specifications wrote in AcQA successfully generated a Q&A System and correctly answered the user's questions about a specific domain.

8.2 Main contributions of this Thesis

Within the specified objectives and results achieved, the main contributions of this Ph.D. work are:

- Proposal of a DSL (AcQA) to generate and automatically deploy a fully functional Q&A System.
- Proposal of the necessary and supporting tools to aid the development of specification written in AcQA.
- The deployment of the code generated by AcQA into a running server makes it easier to deploy a Q&A System, even when the developer does not has knowledge of servers technologies and operating systems.
- Detailed analysis and review of Q&A System classification and technologies used.
- Development of the case studies, detailing the usage of AcQA. Especially the case study Where is my Class?, developed to answer a demand from the Polytechnic Institute of Bragança (IPB).
- Design and implementation of an AcQA experiment to evaluate the AcQA language usage feasibility by users.

8.3 Other activities

During the fruitful period of this doctorate, some projects and academic cooperation with other professors were made, as the participation in some events, as detailed next.

- Participation in the international cooperation project *Reinforcing the security of software systems through reverse engineering methods, techniques and tools*. This is a bilateral research project between Instituto Politecnico de Braganca (IPB) and Universidad de San Luis (UNSL), Argentina.
- The course Computación Forense (Computer forensics) was presented to undergraduate students from Universidad de San Luis. The purpose of this course was to present tools and techniques that can be used in forensic analysis. This course was held in Argentina.
- Participation in the International Summer School on Deep Learning 2017, learning techniques used in this Ph.D. thesis.

- A master class on Computer Systems Security was taught with the professors João Marco Silva and Vítor Fonte at the University of Minho.
- An undergraduate class on Computer Networks was taught together with the professors João Marco Silva, Solange Lima, and Paulo Carvalho, also at the University of Minho.
- Supervision of the participation of the University of Minho in the event Cyber Cloud Expo, held in Braga.
- Assisted a University of Minho committee to join the Android Training Program - Portugal (ATP), made by Google.
- Participation in the project Privacy Shield, from Federal University of Santa Maria (UFSM). Professors Walter Filho from UFSM, and João Marco Silva from the University of Minho, also participate in this project.

During this doctorate, these activities helped to start and keep several productive connections with professors around Portugal and Argentina.

8.4 Future work

As future work, an extension of AcQA language to integrate more techniques to process the user's questions (Ben Abacha and Zweigenbaum 2015,Weissenborn, Wiese, and Seiffe 2017,Gondek et al. 2012,J. Lee et al. 2019) can be made. There is also possible to add support to other user interfaces, extending the already defined HTML5, REST API, and the Flutter application. As the servers are already implemented as templates, it is also possible to create new templates to deploy the resulting Q&A System into different servers operating systems.

Improve the customization of the techniques, allowing the developer to make more significant changes in the algorithms and parameters while writing an AcQA specification.

Apply the AcQA experiment with more and diverse users, extracting more information from the users, helping the development of the AcQA language. With more information from the users of AcQA, it is possible to define the directions to follow during AcQA language development.

Keep the cooperation with the University of Minho (UM), Instituto Politécnico de Bragança (IPB), Universidad Nacional de San Luis (UNSL - Argentina), adding in the cooperation the Federal University of Santa Maria (UFSM), where this author is an adjunct professor.

Bibliography

Adam, S. and U. P. Schultz (2015). "Towards tool support for spreadsheet-based domain-specific languages". In: *ACM SIGPLAN Notices*. Vol. 51. 3. ACM, pp. 95–98 (cit. on p. 24).

Agarwal, A. et al. (May 2019). "EDUQA: Educational Domain Question Answering System Using Conceptual Network Mapping". In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. Vol. 2019-May. Institute of Electrical and Electronics Engineers Inc., pp. 8137–8141. isbn: 9781479981311. doi: 10.1109/ICASSP.2019.8683538. arXiv: 1911.05013 (cit. on p. 20).

Almansa, L., G. Rubio, and A. Macedo (Sept. 2020). "A Question Answering System over Chronic Diseases and Epigenetics Knowledge". In: *Anais Principais do Simpósio Brasileiro de Computação Aplicada à Saúde (SBCAS)*. Sociedade Brasileira de Computacao - SB, pp. 203–214. doi: 10.5753/sbcas.2020.11514. url: http://www.ncbi.nlm.nih.gov/pubmed (cit. on p. 20).

Ansari, A., M. Maknojia, and A. Shaikh (Mar. 2016). "Intelligent question answering system based on Artificial Neural Network". In: *2016 IEEE International Conference on Engineering and Technology (ICETECH)*. IEEE, pp. 758–763. isbn: 978-1-4673-9916-6. doi: 10.1109/ICETECH.2016.7569350. url: http://ieeexplore.ieee.org/document/7569350/ (cit. on pp. 8, 10, 12, 13, 15, 18, 19, 21).

Azevedo, R., P. R. Henriques, and M. J. V. Pereira (2018). "Extending PythonQA with Knowledge from StackOverflow". In: *Trends and Advances in Information Systems and Technologies, WorldCist2018*. Ed. by Á. Rocha et al. 1st ed. Vol. 745. Advances in Intelligent Systems and Computing. Springer International Publishing, pp. 568–575. isbn: 978-3-319-77702-3. doi: https://doi.org/10.1007/978-3-319-77703-0_56 (cit. on pp. 30, 35).

Bechhofer, S. (2009). "OWL: Web Ontology Language". In: *Encyclopedia of Database Systems*. Ed. by L. LIU and M. T. ÖZSU. Boston, MA: Springer US, pp. 2008–2009. isbn: 978-0-387-39940-9. doi: 10.1007/978-0-387-39940-9_1073. url: https://doi.org/10.1007/978-0-387-39940-9_1073 (cit. on p. 28).

Beltran, V. (2016). "Characterization of web single sign-on protocols". In: *IEEE Communications Magazine* 54.7, pp. 24–30 (cit. on pp. 40, 52).

Ben Abacha, A. and P. Zweigenbaum (2015). "MEANS: A medical question-answering system combining NLP techniques and semantic Web technologies". In: *Information Processing and Management* 51.5, pp. 570–594. issn: 03064573. doi: 10.1016/j.ipm.2015.04.006. url: http://dx.doi.org/10.1016/j.ipm.2015.04.006 (cit. on pp. 8, 10, 13, 15, 16, 18–20, 76).

Besbes, G., H. Baazaoui-Zghal, and H. B. Ghezela (2015). "An ontology-driven visual question-answering framework". In: *Proceedings of the International Conference on Information Visualisation* 2015-Septe, pp. 127–132. issn: 10939547. doi: 10.1109/iV.2015.32 (cit. on pp. 8, 10, 12, 13, 15, 16, 18, 19, 21).

Bird, S., E. Klein, and E. Loper (2009). *Natural Language Processing with Python*. 1st. O'Reilly Media, Inc. isbn: 0596516495, 9780596516499 (cit. on pp. 19, 30).

Bobrow, D. G. (1964). "A question-answering system for high school algebra word problems". In: *Proceedings of the October 27-29, 1964, fall joint computer conference, part I*. ACM, pp. 591–614 (cit. on p. 1).

Bravenboer, M. and E. Visser (2004). "Concrete Syntax for Objects: Domain-Specific Language Embedding and Assimilation without Restrictions". In: *Proceedings of the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*. OOPSLA '04. Vancouver, BC, Canada: Association for Computing Machinery, pp. 365–383. isbn: 1581138318. doi: 10.1145/1028976.1029007. url: https://doi.org/10.1145/1028976.1029007 (cit. on p. 27).

Buffenbarger, J. and K. Gruell (2001). "A Language for Software Subsystem Composition". In: *Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 9 - Volume 9*. HICSS '01. USA: IEEE Computer Society, p. 9072. isbn: 0769509819 (cit. on p. 27).

Cai, L. Q. et al. (2020). "Intelligent Question Answering in Restricted Domains Using Deep Learning and Question Pair Matching". In: *IEEE Access* 8, pp. 32922–32934. issn: 21693536. doi: 10.1109/ACCESS.2020.2973728 (cit. on p. 20).

Cao, Y. G. et al. (2011). "AskHERMES: An online question answering system for complex clinical questions". In: *Journal of Biomedical Informatics* 44.2, pp. 277–288. issn: 15320464. doi: 10.1016/j.jbi.2011.01.004. arXiv: NIHMS150003. url: http://dx.doi.org/10.1016/j.jbi.2011.01.004 (cit. on pp. 8, 10, 12, 13, 15, 16, 18–20).

Chandra, S., B. Richards, and J. Larus (1999). "Teapot: a domain-specific language for writing cache coherence protocols". In: *IEEE Transactions on Software Engineering* 25.3, pp. 317–333. doi: 10.1109/32.798322 (cit. on p. 27).

Chesneau, B. (2021). *Gunicorn - Python WSGI HTTP Server for UNIX*. url: https://gunicorn.org/#docs (cit. on p. 45).

Clark, A., C. Fox, and S. Lappin (2010). *The Handbook of Computational Linguistics and Natural Language Processing*. Wiley-Blackwell (cit. on p. 2).

Clements, J. et al. (Mar. 2004). *Fostering Little Languages*. English (US) (cit. on p. 27).

Cointe, P. (2005). "Towards generative programming". In: *Unconventional Programming Paradigms*. Springer, pp. 315–325 (cit. on p. 25).

Crew, R. F. (Oct. 1997). “ASTLOG: A Language for Examining Abstract Syntax Trees”. In: *Conference on Domain-Specific Languages (DSL 97)*. Santa Barbara, CA: USENIX Association. url: https://www.usenix.org/conference/dsl-97/astlog-language-examining-abstract-syntax-trees (cit. on p. 27).

Czarnecki, K. (1999). “Generative Programming: Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models”. PhD Thesis. Technical University of Ilmenau (cit. on p. 25).

Czarnecki, K. (2005). “Overview of generative software development”. In: *Unconventional Programming Paradigms*. Springer, pp. 326–341 (cit. on p. 25).

Davies, R. and L. Cardelli (May 1999). “Service Combinators for Web Computing”. In: *IEEE Transactions on Software Engineering* 25.03, pp. 309–316. issn: 1939-3520. doi: 10.1109/32.798321 (cit. on p. 27).

Django Software Foundation (July 1, 2021). *Django*. Version 3.2. url: https://djangoproject.com (cit. on p. 45).

Engelen, R. A. V. (2002). “Atmol: A domain-specific language for atmospheric modeling”. In: *the Journal of Computing and Information Technology* (cit. on p. 27).

Etworks, S. E. L. F. A. N. (2017). “R-Net: Machine Reading Comprehension With Self-Matching Networks *”. In: *arXiv*, pp. 1–11. url: https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf (cit. on pp. 8, 10, 13, 15, 19, 20).

Fang, H. et al. (2015). “From captions to visual concepts and back”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 07-12-June, pp. 1473–1482. issn: 10636919. doi: 10.1109/CVPR.2015.7298754. arXiv: 1411.4952 (cit. on pp. 8, 10, 13, 15, 16, 18, 19, 21).

Ferrucci, D. (2010). “Build watson: An overview of DeepQA for the Jeopardy! Challenge”. In: *2010 19th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, p. 1 (cit. on pp. 5, 6, 8, 10, 12, 13, 15, 18, 19, 21).

Fortnow, L. and S. Homer (2003). *A short history of computational complexity*. Tech. rep. Boston University Computer Science Department (cit. on p. 1).

Fowler, M. (2010). *Domain-specific languages*. Pearson Education (cit. on pp. 24, 25).

George, E. J. (Sept. 2020). “Verb focused answering from cord-19”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 12284 LNAI. Springer Science and Business Media Deutschland GmbH, pp. 206–213. isbn: 9783030583224. doi: 10.1007/978-3-030-58323-1_22 (cit. on pp. 8, 10, 22).

Ghosh, D. (2010). *DSLs in action*. Manning Publications Co. (cit. on pp. 24, 25).

Gobet, F., J. Retschitzki, and A. de Voogt (2004). *Moves in mind: The psychology of board games*. Psychology Press (cit. on p. 51).

Gondek, D. C. et al. (2012). “A framework for merging and ranking of answers in DeepQA”. In: *IBM Journal of Research and Development* 56.3.4, pp. 14–1 (cit. on p. 76).

Inc., Wolfram Research (2018). *Wolfram Alpha* (cit. on pp. 6, 8, 10, 13, 15, 18, 19, 21).

Jain, A., G. Kulkarni, and V. Shah (2018). "Natural language processing". In: *International Journal of Computer Sciences and Engineering* (cit. on p. 6).

Jayalakshmi, S. and A. Sheshasaayee (2017). "Automated Question Answering System Using Ontology and Semantic Role". In: *International Conference on Innovative Mechanisms for Industry Applications (ICIMIA 2017)*. Icimia, pp. 528–532. isbn: 9781509059607 (cit. on pp. 8, 10, 12, 13, 15, 16, 18, 19, 21).

Jiang, W., Q. Xu, and X. Wang (July 2019). "Research and application of question answering system in the field of air traffic control". In: *Chinese Control Conference, CCC*. Vol. 2019-July. IEEE Computer Society, pp. 8622–8626. isbn: 9789881563972. doi: 10.23919/ChiCC.2019.8865973 (cit. on p. 20).

Kaisser, M. and T. Becker (2004). "Question Answering by Searching Large Corpora With Linguistic Methods." In: *TREC* (cit. on p. 5).

Kalaivani, S. and K. Duraiswamy (2012). "Comparison of question answering systems based on ontology and semantic web in different environment". In: *Journal of Computer Science* 8.8, pp. 1407–1413. issn: 15493636. doi: 10.3844/jcssp.2012.1407.1413 (cit. on pp. 8, 10, 12, 13, 15, 16, 18, 19, 21).

Klarlund, N. and M. Schwartzbach (1999). "A domain-specific language for regular sets of strings and trees". In: *IEEE Transactions on Software Engineering* 25.3, pp. 378–386. doi: 10.1109/32.798326 (cit. on p. 27).

Kumar, S. (2002). "Esp: A Language for Programmable Devices". AAI3033049. PhD thesis. isbn: 0493457933 (cit. on p. 27).

Launchbury, J., J. R. Lewis, and B. Cook (1999). "On Embedding a Microarchitectural Design Language within Haskell". In: *Proceedings of the Fourth ACM SIGPLAN International Conference on Functional Programming*. ICFP '99. Paris, France: Association for Computing Machinery, pp. 60–69. isbn: 1581131119. doi: 10.1145/317636.317784. url: https://doi.org/10.1145/317636.317784 (cit. on p. 28).

Lee, J. et al. (2019). "Biobert: pre-trained biomedical language representation model for biomedical text mining". In: *arXiv preprint arXiv:1901.08746* (cit. on p. 76).

Lee, S. et al. (Sept. 2019). "Visual Question Answering over Scene Graph". In: *Proceedings - 2019 1st International Conference on Graph Computing, GC 2019*. Institute of Electrical and Electronics Engineers Inc., pp. 45–50. isbn: 9781728141299. doi: 10.1109/GC46384.2019.00015 (cit. on p. 21).

Lende, S. P. and M. M. Raghuwanshi (2016). "Question answering system on education acts using NLP techniques". In: *IEEE WCTFTR 2016 - Proceedings of 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare*. isbn: 9781467392143. doi: 10.1109/STARTUP.2016.7583963 (cit. on pp. 8, 10, 12, 13, 15, 16, 18–20).

Li, Y., J. Cao, and Y. Wang (Dec. 2019). "Implementation of Intelligent Question Answering System Based on Basketball Knowledge Graph". In: *Proceedings of 2019 IEEE 4th Advanced Information Technology,* *Electronic and Automation Control Conference, IAEAC 2019*. Institute of Electrical and Electronics Engineers Inc., pp. 2601–2604. isbn: 9781728119076. doi: 10.1109/IAEAC47372.2019.8997747 (cit. on p. 21).

Mernik, M., J. Heering, and A. M. Sloane (2005). "When and how to develop domain-specific languages". In: *ACM computing surveys (CSUR)* 37.4, pp. 316–344 (cit. on pp. 24, 26–28).

Miller, G. A. (1995). "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11, pp. 39–41. issn: 00010782. doi: 10.1145/219717.219748. url: http://portal.acm.org/citation.cfm?doid=219717.219748 (cit. on p. 21).

Mishra, A. and S. K. Jain (2016). "A survey on question answering systems with classification". In: *Journal of King Saud University - Computer and Information Sciences* 28.3, pp. 345–361. issn: 22131248. doi: 10.1016/j.jksuci.2014.10.007. url: http://dx.doi.org/10.1016/j.jksuci.2014.10.007 (cit. on pp. 7, 11).

Mochalova, V. A. et al. (2015). "Ontological-semantic text analysis and the question answering system using data from ontology". In: *ICACT Transactions on Advanced Communications Technology (TACT) Vol.* 4.4, pp. 651–658 (cit. on pp. 8, 10, 13, 15, 18, 19, 21).

Nedelcu, C. (2013). *Nginx HTTP Server - Second Edition*. 2nd. Packt Publishing. isbn: 1782162321 (cit. on p. 45).

Nguyen, T. et al. (2016). "MS MARCO: A human generated MAchine reading COmprehension dataset". In: *CEUR Workshop Proceedings* 1773.Nips, pp. 1–10. issn: 16130073. arXiv: 1611.09268 (cit. on p. 22).

Och, F. (2003). "Minimum error rate training in statistical machine translation". In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*. Vol. 1, pp. 160–167. isbn: 9781905593446. doi: 10.3115/1075096.1075117. url: http://dl.acm.org/citation.cfm?id=1075117 (cit. on p. 21).

Packowski, Sarah and Lakhana, A. (2017). "Using IBM Watson Cloud Services to Build Natural Language Processing Solutions to Leverage Chat Tools". In: *Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering*. November. Markham, Ontario, Canada: IBM Corp., pp. 211–218. url: http://dl.acm.org/citation.cfm?id=3172795.3172819 (cit. on p. 6).

Parr, T. (2013). *The Definitive ANTLR 4 Reference*. 2nd. Pragmatic Bookshelf. isbn: 1934356999, 9781934356999 (cit. on pp. 32, 38).

Pasca, M. (2007). "Lightweight Web-based fact repositories for textual question answering". In: *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. ACM, pp. 87–96 (cit. on p. 6).

Plath, W. J. (1976). "REQUEST: A natural language question-answering system". In: *IBM Journal of Research and Development* 20.4, pp. 326–335 (cit. on p. 1).

Rajendran, P. S. and R. Sharon (Dec. 2017). "Dynamic question answering system based on ontology". In: *2017 International Conference on Soft Computing and its Engineering Applications (icSoftComp)*.

IEEE, pp. 1–6. isbn: 978-1-5386-2053-3. doi: 10.1109/ICSOFTCOMP.2017.8280094. url: http://ieeexplore.ieee.org/document/8280094/ (cit. on pp. 8, 10, 12, 13, 15, 16, 18, 19, 21).

Rajpurkar, P. et al. (2016). "Squad: 100,000+ questions for machine comprehension of text". In: *arXiv preprint arXiv:1606.05250* (cit. on p. 22).

Ramos, M., M. J. V. Pereira, and P. R. Henriques (2017). "A {QA} System for learning Python". In: *Communication Papers of the 2017 Federated Conference on Computer Science and Information Systems, FedCSIS 2017, Prague, Czech Republic, September 3-6, 2017.* Pp. 157–164. doi: 10.15439/2017F157. url: https://doi.org/10.15439/2017F157 (cit. on pp. 8, 10, 12, 13, 15, 16, 18, 19, 22, 35, 56).

Robinson, J. (2014). "Likert Scale". In: *Encyclopedia of Quality of Life and Well-Being Research*. Ed. by A. C. Michalos. Dordrecht: Springer Netherlands, pp. 3620–3621. isbn: 978-94-007-0753-5. doi: 10.1007/978-94-007-0753-5_1654. url: https://doi.org/10.1007/978-94-007-0753-5_1654 (cit. on p. 68).

Ronacher, A. (2008). *Welcome to Jinja2 — Jinja2 Documentation (2.9)*. url: http://jinja.pocoo.org/docs/2.9/ (cit. on p. 45).

Sasikumar, U. and L. Sindhu (2014). "A Survey of Natural Language Question Answering System". In: *International Journal of Computer Applications* 108.15 (cit. on p. 5).

Shang, S., J. Liu, and Y. Yang (2020). "Multi-Layer Transformer Aggregation Encoder for Answer Generation". In: *IEEE Access* 8, pp. 90410–90419. issn: 21693536. doi: 10.1109/ACCESS.2020.2993875 (cit. on p. 22).

Shen, Y. et al. (2017). "Modeling Large-Scale Structured Relationships with Shared Memory for Knowledge Base Completion". In: *Proceedings of the 2nd Workshop on Representation Learning for NLP*. arXiv: 1611.04642. url: http://arxiv.org/abs/1611.04642 (cit. on p. 22).

Sloane, A. (2002). "Post-design domain-specific language embedding: a case study in the software engineering domain". In: *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pp. 3647–3655. doi: 10.1109/HICSS.2002.994492 (cit. on p. 28).

Smaragdakis, Y. and D. Batory (Oct. 1997). "DiSTiL: A Transformation Library for Data Structures". In: *Conference on Domain-Specific Languages (DSL 97)*. Santa Barbara, CA: USENIX Association. url: https://www.usenix.org/conference/dsl-97/distil-transformation-library-data-structures (cit. on p. 28).

Sreelakshmi, A. S. et al. (Nov. 2019). "A Question Answering and Quiz Generation Chatbot for Education". In: *2019 Grace Hopper Celebration India, GHCI 2019*. Institute of Electrical and Electronics Engineers Inc. isbn: 9781728142647. doi: 10.1109/GHCI47972.2019.9071832 (cit. on p. 20).

Stallings, W. (2017). *Cryptography and network security: principles and practice*. Pearson Upper Saddle River (cit. on p. 45).

Thibault, S., R. Marlet, and C. Consel (1999). "Domain-specific languages: from design to implementation application to video device drivers generation". In: *IEEE Transactions on Software Engineering* 25.3, pp. 363–377. doi: 10.1109/32.798325 (cit. on p. 28).

Tomassetti, F. (2020). *How to create pragmatic , lightweight languages Learn the process to create DSLs and GPLs*. Ed. by Leanpub. 1st ed. Leanpub (cit. on p. 28).

Vargas-Vera, M. and M. D. Lytras (2010). "Aqua: A closed-domain question answering system". In: *Information Systems Management* 27.3, pp. 217–225 (cit. on p. 5).

Waltz, D. L. (1978). "An English language question answering system for a large relational database". In: *Communications of the ACM* 21.7, pp. 526–539 (cit. on p. 1).

Weissenborn, D., G. Wiese, and L. Seiffe (2017). "FastQA: A simple and efficient neural architecture for question answering". In: *arXiv preprint arXiv:1703.04816* (cit. on pp. 8, 10, 12, 13, 15, 19, 20, 76).

Yan, G. and J. Li (July 2018). "Mobile medical question and answer system with auto domain lexicon extraction and question auto annotation". In: *Proceedings - 2018 33rd Youth Academic Annual Conference of Chinese Association of Automation, YAC 2018*. Institute of Electrical and Electronics Engineers Inc., pp. 637–641. isbn: 9781538672556. doi: 10.1109/YAC.2018.8406451 (cit. on p. 22).

Ylonen, T. (1996). "SSH–secure login connections over the Internet". In: *Proceedings of the 6th USENIX Security Symposium*. Vol. 37 (cit. on pp. 40, 52).

Zou, Y., Y. He, and Y. Liu (July 2020). "Research and implementation of intelligent question answering system based on knowledge Graph of traditional Chinese medicine". In: *Chinese Control Conference, CCC*. Vol. 2020-July. IEEE Computer Society, pp. 4266–4272. isbn: 9789881563903. doi: 10.23919/CCC50068.2020.9189518 (cit. on p. 20).