Extending PythonQA with knowledge from StackOverflow

Renato Preigschadt de Azevedo, Pedro Rangel Henriques[1], and Maria João Varanda Pereira[2]

[1] Centro Algoritmi (CAlg-CTC), Dep. Informática, Universidade do Minho, Braga, Portugal,
renato@redes.ufsm.br, prh@di.uminho.pt,
[2] Centro Algoritmi (CAlg-CTC), Dep. Informática e Comunicações Instituto Politécnico de Bragança Portugal
mjoao@ipb.pt.

Abstract. Question and Answering (QA) Systems provide a platform where users can ask questions in natural language to a system and get answers retrieved from a knowledge base. The work proposed in PythonQA create a Question and Answer System for the Python Programming Language. The knowledge is built from the Python Frequent Answered Questions (PyFAQ). In this paper, we extend the PythonQA system by enhancing the Knowledge Base with Question-Answer pairs from the StackExchange Python Question Answering Community Site. Some tests were performed to analyze the impact of a richer Knowledge Base on the PythonQA system, increasing the number of answer candidates.

Keywords: question and answering systems, natural language processing, stackexchange

1 Introduction

The wideness of information available associated with the demand for direct answers from the users requires a different approach from standard search engines.

Question and Answering (QA) Systems provide a way to process natural language inputs from a user extracting the meaning which enables better and direct answers from a computer system. These systems allow a user to make questions in a more natural way and get concise and straightforward answers, thus decreasing the effort necessary to find a good answer. Unlike standard search engines that retrieve documents based on keywords for the input, QA systems aim to recognize the input as a high-level natural language enabling the retrieval of concise answers instead of a set of possible related documents.

Many Python users are not Software Engineer since Python is more popular in Data Science and other areas. PythonQA is a QA system proposed by Ramos [9] and is a closed-domain QA that addresses the Python programming language [10]. It was developed aiming to be useful for students and professionals that are working or learning Python.

Question and Answering sites like StackExchange (SE) provide a platform where users can ask questions on specialized topics and get feedback provided by users that have knowledge on the topic. This paper aims to extend PythonQA [9], adding knowledge from the SE Python site to the original Python Frequent Asked Questions (PyFAQ) Knowledge Base.

A review of the literature is presented in Section 2 including a discussion on Question and answering systems; In Section 3 an overview of PythonQA is presented; The extension of PythonQA, among details about the StackExchange and tests and results, is shown in Section 4, and Section 5 closes the paper with conclusions and directions for future work.

2 Related Work

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

A simple QA System is composed of several processes: question typing, query construction and text retrieval, and processing answer candidates [5]. Question typing analyses the input from the user to extract meaning from the phrases entered and can be done with Natural Language Processing (NLP) techniques. The query construction and the text retrieval allow to recover the information about relevant document and data from the Knowledge Base (KB). Using that information, a list of answer candidates are retrieved creating a ranking of the best answers to present to the user.

QA systems can be divided into two categories: closed or open domain. Closed-domain QA systems aim to address a specific area of knowledge, providing more accurate answers and being easier to fine tune the system. Some examples of Closed-domain QA are Question Answer System on Education Acts [8], Python Question Answer System (PythonQA) [9], and K-Extractor [2]. Open-domain QA systems attempt to work with any domain of knowledge, having a broader knowledge base than the closed-domain. Examples of Open-domain QA are Intelligent Question Answering System based on Artificial Neural Network [1], Automatic Question-Answering Based on Wikipedia Data Extraction [7], and SEMCORE [6].

In MEANS [3] the authors propose a semantic approach to a medical QA system. They apply NLP to process the corpora and the user questions. The sources documents are annotated with RDF, based on an ontology. The authors propose ten question types. In the work proposed by [8], a QA system to handle education acts is presented. The knowledge base is created from the data publicly available from the UK parliament using NLP techniques. Only keywords are extracted from the user question, ignoring the question type and possible actions present in the input from the user. The authors in [4] created

AskHERMES, a QA 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). To process the possible answers, the authors developed a question summarization and answers presentation based on a clustering technique.

3 PythonQA: an overview

PythonQA is a closed-domain Question and Answering system that answer questions about the Python programming language. As a closed-domain QA system, PythonQA can provide concise answers rather 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 for the QA system. Nonetheless, other languages such as Java, Haskell or Julia could serve as the domain of the PythonQA without the need for structural changes.

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.

The PythonQA system was developed using the Python programming language, together with some libraries such as Natural Language ToolKit (NLTK), Django, among others. 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. The Figure 1 describe the significant phases of the Phrase analysis. Firstly the question is processed with the NLTK library to replace contractions, converting them to their full form. The next 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.

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 identified in the previous steps. To identify keywords firstly the following information is removed: stopwords, verbs, unwanted characters, ”Python”. After the removal of unwanted information, the keyword candidates

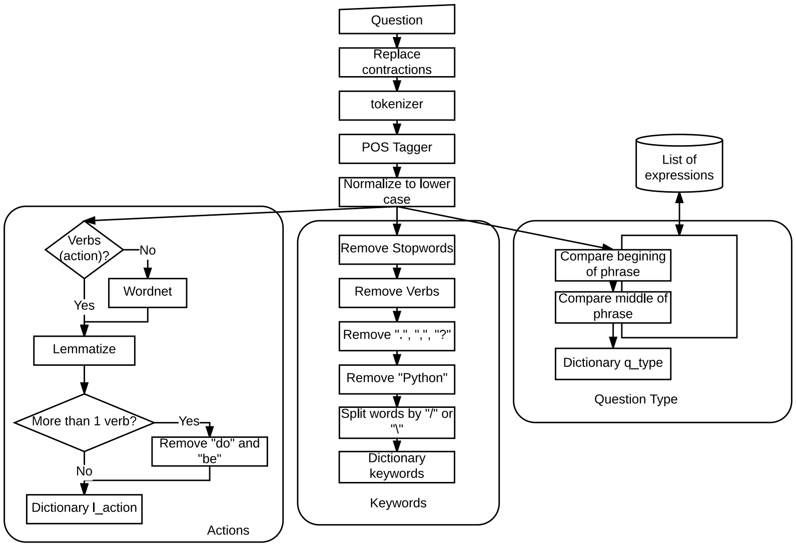

Fig. 1. Phrase Analysis

are processed to split words that may have a slash (”/” or ”̈) between them. To finish a dictionary is created with the keywords found in the previous steps, along with a value of assertiveness. The PythonQA contains a list of expressions that was extracted with the manual analysis of the PythonFAQ [10]. This list has expressions like ”How”, ”When”, ”Where”. This expressions 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 value for the question type.

The Knowledge Base was constructed with the entries from the Python Frequently Asked Questions (PyFAQ [10]). All the questions are processed by the Phrase Analysis module of PythonQA. For each pair Questions → Answer, the KB is populated with the raw data, along with the dictionaries actions, keywords, and question type. The information stored in the KB is crucial for the information module be able to extract and present concise answers to users.

The Answer retrieval is the module that is responsible for processing the information gathered in the Phrase Analysis module and present an answer to the user. The Figure 2 depicts the steps necessary to find and process the answer candidates. The analysis of Actions and Keywords are made looking for a direct match with the KB. After the previous phase, the PythonQA system uses an NLTK Stemmer package to get the base word. With the base word, the system tries to find synonyms that are used 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 to question type is done firstly with a direct match, and then with

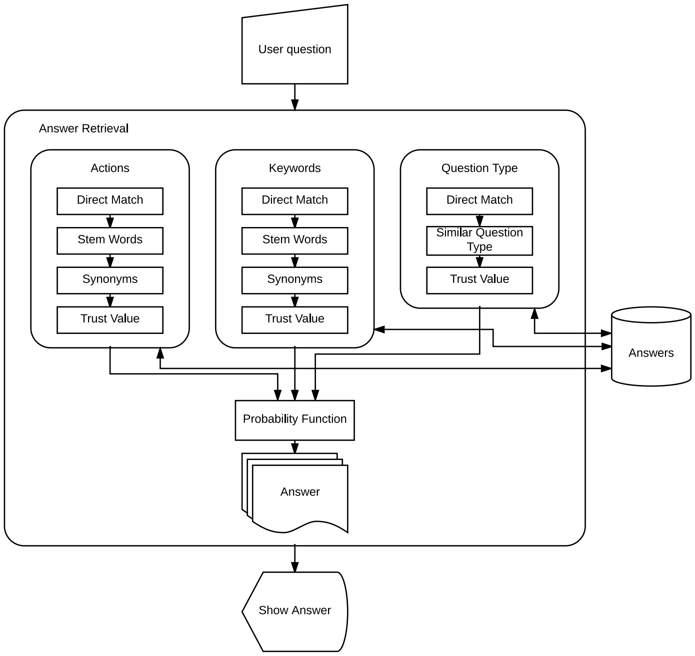

Fig. 2. Answer Retrieval

similar question type. A trust value is then assigned for each answer retrieved from the KB. All these steps are made to retrieve more answers candidates that match actions, keywords, and question type. With all these candidates 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.

4 Extending the 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 [9], restricting the knowledge of the system.

To increase the KB, we have to choose between CQAS such as StackExchange[3] or Yahoo Answers[4]. 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. StackExchange is an Online Social Question and

[3] www.stackexchange.com

[4] answers.yahoo.com

Answering site which allows users to post questions and answers to questions already asked. StackOverflow is one of the 166 Stack Exchange Community and provide information about programming languages, like Python.

The data is available as a direct download through the Archive.org Site[5]. 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 into 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 body of the question. The Comments file contains comments produced by users of SE about the questions and answer that is inside the Posts file. We downloaded the StackExchange programming data from StackOverflow.

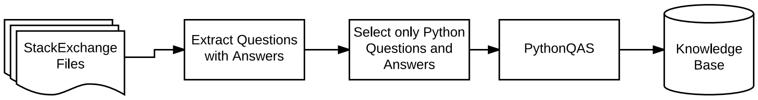
StackExchange Files
Extract Questions with Answers
Select only Python Questions and Answers
PythonQAS
Knowledge Base

Fig. 3. Extending PythonQA

The Figure 3 detail the steps necessary to process the data from the StackExchange and insert in the Knowledge Base of PythonQA. Firstly we extract the Questions that have answers from the Posts file. Next, we select only questions and answers that has a Python tag associated with the pair Question → Answer. After we have extracted all Question → Answer pair, we process them into PythonQA in the Phrase Analysis module to insert into the Knowledge Base of the system.

Some improvements have to be made in the PythonQA system, to be possible to process more than 480 thousand Question → Answer pairs. To add information in the PythonQA system, we developed a module that handles the data in an unattended way.

Some preliminary tests were made, with ten random questions extracted from StackOverflow that were not imported to KB. The original KB was only able to correctly answer 20% of the analyzed questions, while the extended KB fulfilled successfully 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. This was possible because the information available in the StackOverflow are 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 a XML against an DTD in Python". The q1 is correctly answered in PythonQA with original and extended KB. But

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

with the question q2 only with the extended KB, a relevant answer is presented. Because of page limitations, more details about the tests and PythonQA extended version is available at *http://pythonqas2.epl.di.uminho.pt*.

5 Conclusions

We presented our improvements made in the PythonQA system, which aims to extend the knowledge base and provide better answers to the users. The KB of the PythonQA system was improved with knowledge from the community QA site StackExchange, which provides relevant Questions and Answers from the Python topic. The changes made to the system code allowed the inclusion of a larger knowledge base. The PythonQA system has also benefited from the user's tailored information about the quality of answers present in the dataset. The information present in the answers usually has examples of Python code, turning the answer more relevant to the user.

As a future work, we can enhance the PythonQA system by using different CQAS, extending the knowledge base.

References

1. Ansari, A., Maknojia, M., Shaikh, A.: Intelligent question answering system based on artificial neural network. In: 2016 IEEE International Conference on Engineering and Technology (ICETECH). pp. 758–763 (2016)
2. Balakrishna, M., Werner, S., Tatu, M., Erekhinskaya, T., Moldovan, D.: K-Extractor: Automatic Knowledge Extraction for Hybrid Question Answering. In: Proceedings - 2016 IEEE 10th International Conference on Semantic Computing, ICSC 2016 (2016)
3. Ben Abacha, A., Zweigenbaum, P.: MEANS: A medical question-answering system combining NLP techniques and semantic Web technologies. Information Processing and Management 51(5), 570–594 (2015)
4. Cao, Y.G., Liu, F., Simpson, P., Antieau, L., Bennett, A., Cimino, J.J., Ely, J., Yu, H.: AskHERMES: An online question answering system for complex clinical questions. Journal of Biomedical Informatics 44(2), 277–288 (2011)
5. Clark, A., Fox, C., Lappin, S.: The Handbook of Computational Linguistics and Natural Language Processing. Wiley-Blackwell (2010)
6. Hoque, M.M., Quaresma, P.: A Content-Aware Hybrid Architecture for Answering Questions from Open-domain Texts. 2016 19th International Conference on Computer and Information Technology (ICCIT) pp. 293–298 (2016)
7. Huang, X., Wei, B., Zhang, Y.: Automatic question-answering based on wikipedia data extraction. In: 10th International Conference on Intelligent Systems and Knowledge Engineering, ISKE 2015, Taipei, Taiwan. pp. 314–317 (2015)
8. Lende, S.P., Raghuwanshi, M.M.: Question answering system on education acts using NLP techniques. In: IEEE WCTFTR - Proceedings of 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (2016)
9. Ramos, M., Pereira, M.J.V., Henriques, P.R.: A QA system for learning python. In: Communication Papers of the 2017 FedCSIS, Prague, Czech Republic. (2017)
10. Rossum, G.: Python reference manual. Tech. rep., Amsterdam, The Netherlands, The Netherlands (1995)