
Production Systems and Information Engineering
Volume 9 (2020), pp. 54–66

EVALUATION OF PYTHON BASED NLP FRAMEWORKS FOR TEMPLATE BASED AUTOMATIC QUESTION GENERATION

Walelign Tewabe Sewunetie
University of Miskolc, Hungary
Department of Information Engineering
sewunetie@ait.iit.uni-miskolc.hu

László Kovács
University of Miskolc, Hungary
Department of Information Engineering
kovacs@iit.uni-miskolc.hu

[Received ... and accepted ...]

Abstract. Automatic question generation techniques emerged as a solution to the challenges facing test developers in the development of smart e-tutoring systems. The current challenge in selecting the available developer tools is depend on several aspects, including the kind and source of text, where the level, formal or informal, may influence the performance of such tools. This tool, popular packages for NLP: NLTK, spaCy, TextBlob, and CoreNLP.
Our experiences show that spaCy is several times faster than others in tokenization, tagging and parsing. It has also the best feature set of neural network models and of entity recognition methods. Based on our test results spaCy would be an optimal choice for the implementation of template based automatic question generation. The downside of spaCy is the limited number of supported languages. The choice which NLP package to choose depends on the specific problem you have to solve.

Keywords: Python, NLP Frameworks, Template Based Question Generation, spaCy, NLP

1. Introduction

Natural Language Processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human, in particular how to program computers to process and analyze large amounts of natural language data. It mainly concerns about

teaching machines how to understand human languages and extract meaning from text [1].

Natural language processing is a computer process that requires superior knowledge of mathematics, machine learning, and linguistics. Now, developers can use ready-made tools that simplify text preprocessing so that they can concentrate on building machine learning models [1]. Google, Amazon, or Facebook are pouring millions of dollars into NLP line of research to power their chatbots, virtual assistants, recommendation engines, and other solutions powered by machine learning. NLP relies on advanced computational skills, developers would like to use the best available tools for creating services that can handle natural languages.

There are many things about Python that make it a really good programming language choice for an NLP projects. The simple syntax and transparent semantics of this language make it an excellent choice for complex projects like NLP tasks. Moreover, developers can enjoy excellent support for integration with other languages and tools that come in handy for techniques like machine learning.

Python provides developers with an extensive collection of NLP tools and libraries that enable developers to handle a great number of NLP-related tasks such as document classification, topic modeling, part-of-speech (POS) tagging, word vectors, and sentiment analysis. AQG is characterized as the task of generating syntactically sound, semantically correct, and appropriate questions from multiple input formats such as text, a structured database, or a knowledge base.

Asking assessment questions is an essential feature of advanced learning technologies such as smart tutoring systems, game-based learning environments and inquiry-based environments [2]. A general human technique for generating questions is to thoroughly read the article setting up an internal model of information and then generating questions accordingly. In the case of automated question generation (AQG) the engine generates the questions automatically from the available text documents. Many AQG systems are used in educational applications, such as skill development assessment and knowledge assessment. In the Extended ITS Architecture [3] the question generation module is using intuitionistic logic for evaluation of the generated questions. The field of AQG is an important research area that can be useful in intelligent tutoring systems, dialog systems, educational technology, educational games,

e.t.c [4].

The main goal of the study is to compare, to test and to analyze different available Python based NLP frameworks for template based question generation. Template-based QG is a baseline which utilizes templates created by experts of human extracted from training set and then generates questions by filling the particular templates with certain topic entities.

2. Extensions of the article Evaluation of Python-based NLP Frameworks

2.1. Question Generation from Databases

One approach for AQG is to use a database, like relational database for the input source. The relational database [6] has the benefit that it contains a strict structure to store information and data. This structure enables to determine the meaning, semantic role of the different data items. In this case, the schema may refer to the different semantic components using references to the column names. On the other hand, to generate the NL sentences, the framework requires an NLP module to transform the lemma forms into the corresponding inflected forms.

The formal model of the database oriented AQG system can be given as follows.

- $D : \{T_1, T_1, ..., T_n\}$: input database
- $T_i : T_i(m_{i1}, m_{i2}, ..., m_{im})$: table schema containing columns
- $S : \{(S_i, Q_i)\}$: set of QA schemas
- $S_i = w_1, w_2, ..., w_m$: query schema, where w_i denotes either a NL word or a reference to a table column of the form $T_i.m_j$
- Q_i : *select ...from ...where ...*, the SQL query to yield the answer, where the SQL query can contain references to the symbol parameters used in the query schema.

For the tests, we have implemented a background database in sqlite. The database schema contains the following tables.

In Table 1, we can find examples for the generated QA schema with references to the implemented database. To implement this system we have used SQLite tools to create a relational database and python for template-based question answering systems.

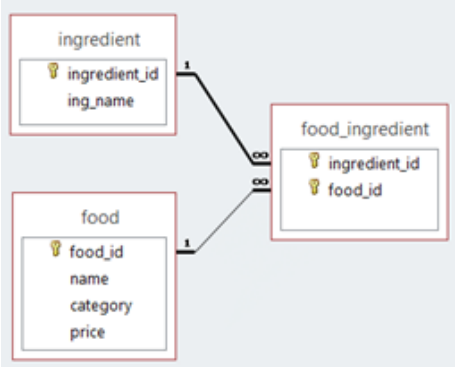

Figure 1. Restaurant food ingredient database schema

No	Question/Query Template	Sample
1	S: What is the active ingredient in @food.name? Q: Select i.ing_name From ingredient i inner join food_ingredient g on i.ingredient_id = g.ingredient_id inner join food f on f.food.id = g.food_id where f.name = @food.name	S: What is the active ingredient in potato? A: vitamin C, potassium, phosphorus and magnesium
2	S: What is the price of @food.name? Q: select price from food where name = @food.name	S: What is the price of potato? A: 20
3	S: What is the category of @food.name? Q: select category from food where name = @food.name	S: What is the category of orange? A: fruit
4	S: What kind of food have ingredient @ing.name? Q: Select f.name From ingredient i inner join food_ingredient g on i.ingredient_id = g.ingredient_id inner join food f on f.food.id = g.food_id where i.name = @ing.name	S: What vegetable have more protein? A: Edamame, Lentils

Table 1. Sample question templates with answers

As the examples show the accuracy of database oriented question generation is very high but it needs to create a template for all possible ways of questions. Thus, in case of large domain it is more challenging and time taking to

construct all the required templates. In future, we will extend this work for automatized schema generation for databases. Here below is a sample python code that we have used for a template-based question answering system with database support.

```
Rules = [("What is the active ingredient in @food.name",
          "Select i.ing_name From ingredient i inner join
          food_ingredient g on i.ingredient_id = g.ingredient_id
          inner join food f on f.food.id = g.food_id where f.name
          = @food.name"),
          ("What is the price of @food.name","select price from
          food where name = '@food.name'")

        ]


def qa_process (qid):
    conn = sqlite3.connect('../Python_Cube/aqgtest')
    cur = conn.cursor()

    qid = 1
    qry = Rules[qid][0]
    qrys = qry.split("@")
    qrys2 = qrys[1].split(" ")[0]
    qryss = qrys2.split(".")

    sql = "select " + qryss[1] + " from " + qryss[0]
    result = conn.execute(sql);
    for rec in result:
        break
    qtext = qry.replace("@"+qrys2,rec[0])+"?"
    #print ("Q:", qtext)
    qry = Rules[qid][1]
    sql = qry.replace ("@"+qrys2,rec[0])
    #print (sql)
    result = conn.execute(sql);
    for rec in result:
        break
    atext = str(rec[0])
    #print ("A:", atext)
    return (qtext,atext)
```

2.2. Question Generation from Free Text

NLP plays a critical role in many intelligent applications such as automated chat bots, article summarizers, multi-lingual translation and opinion identification from data. Every industry which exploits NLP to make sense of unstructured text data, not just demands accuracy, but also swiftness in obtaining results [11]. Some of the tasks in NLP are text classification, entity detection, machine translation, question answering, and concept identification. Python is a top developing software that can handle natural languages in the context of artificial intelligence. For the implementation of Template Based Question Generation the researchers' analyzed different python based NLP frameworks.

In a research work Nguyen-thinh le et al [6] use extracted key concepts to generate questions and determine the types of questions to be generated. They use the domains of energy and economy topic to select the sentences and question. The author presents nouns and noun phrases first extracted from a discussion topic and replace by X placeholder. The template displayed in Table 1 shows that question templates are filled with the noun phrase "nuclear energy" and result in some questions. As we have seen on the template below questions are more dependent on domain and topic.

The authors used an improved NER spaCy which is capable of labeling more entity types, including money, dates/times, etc to generate questions containing the question word Why, How much, to what extent etc, [8].

In the work [8] the template creation focuses on the events (actions, happenings) and existents (characters, settings). The questions in the templates ask about the subject, the predicate, and the object of the events and existents. After removal the errors, they created 19 improved templates under 6 categories that are included in the system [8].

In David's [9] thesis report he explored semantics-based templates that uses Semantic Role Labeling (SRL) in conjunction with generic and domain-specific scope for self-directed learning. According to his report the questions that are generated are not answerable from the original sentence, they were judged answerable from the source document in our evaluation. The ability to generate questions that require the learner to consult other parts of the text is due to the flexibility of the templates.

In this study the authors have used spaCy libraries for POS tagging and this information is used to identify the potential content for the template of

Type	Question
Definition	What is @X? What do you have in mind when you think about @X? What does @X remind you of?
Feature/Property	What are the properties of @X? What are the (opposite)-problems of @X? What features does @X have?
Example	What is an example of @X
Verification	Is there any problem with the arguments about @X?
Judgment	What do you like when you think of or hear about @X
Interpretation	How can @X be used today?
Expectation	How will @X be in the future, based on the way it is now?
Quantification	How many sub-topics did you partners talk about? Which sub-topics do you partners focus on?
Concept Comparison	What is the difference or relations between these sub-topics?

Table 2. Question Templates Proposed for AQG [6]

As recently as 12,500 years ago, the Earth was in the midst of a glacial age referred to as the Last Ice Age.
T: How would you describe [A2-Ipp misc]? Q: How would you describe the Last Ice Age?
T: Summarize the influence of [A1-lp !comma !nv] on the environment. Q: Summarize the influence of a glacial age on the environment.
T: What caused [A2-Ipp nv misc]? ## [A0 null] Q: What caused the Last Ice Age?

Table 3. Sample templates and questions [9]

questions [10]. A part of the speech tagger are used to encode necessary information. In order to decide the type of questions that can be produced from this sentence, verb, object and preposition will be categorized on the basis of the subject.

3. Comparative analysis of NLP Libraries in Python for Template Based Question Generation

The most common tools and libraries that created to solve NLP problems are Natural Language Toolkit (NLTK), spaCy, TextBlob, and CoreNLP. The NLTK , for English written in the Python programming language, is a suite of libraries and programs for symbolic and statistical NLP [13]. It can include various datasets in multiple languages that can be deployed depending on the features you need. Stanford CoreNLP is a community that created core NLP components such as Tokenization, Sentence Recognition, POS Tagging, NER, Entity Linking and Training Annotation, etc [14]. The most distinctive characteristic of the Stanford NLP Group is its successful integration of advanced and deep linguistic modeling and data processing with innovative probabilistic approaches to NLP, machine learning, and deep learning. The comparison of the features offered by spaCy, NLTK, TextBlob and Stanford CoreNLP in table 4 shows that spaCy is an advanced modern NLP library. spaCy have pre-trained NLP models capable of performing the most common NLP tasks, such as tokenization, POS tagging, NER recognition, lemmatization and word vector transformation [15]. TextBlob is a Python library which offers a simple API for accessing its methods and carrying out basic NLP tasks. It offers a simple API for diving into specific NLP tasks such as part-of - speech tagging, extraction of the noun phrase, interpretation of emotions, classification, translation and more [16].

Table 5 shows the comparison of per-document processing time of various spaCy functionalities against other NLP libraries. We show both absolute timings (ms) and relative performance (normalized) to spaCy [17].

Reviewed papers confirmed that spaCy offers the fastest syntactic parser in the world and that its accuracy is within 1% of the best available. The few systems that are more accurate are 20x slower or more [17].

Spacy is very powerful and industrial strength package for almost all natural language processing tasks. The following figure shows the comparison of spaCy with CoreNLP and NLTK based on accuracy for entity extraction.
Spacy consists of a fast entity recognition model which is capable of identifying entity phrases from the document. Entities can be of different types, such as person, location, organization, dates, numerals, etc. These entities can be accessed through ".ents" property. According to the research work [18] and we have also observe that spaCy is easy to use, provides the best overall performance compared to Stanford CoreNLP Suite, Google's SyntaxNet, and NLTK Python library.

Feature	spaCy	NLTK	Stanford CoreNLP	TextBlob
Programming Language	✓			
Easy Installation	✓	✓	✓	
Neural Network Models	✓	✓		
Integrated word vectors	✓			
Multi language support	✓	✓	✓	
Tokenization	✓	✓	✓	
Part of Speech Tagging	✓	✓	✓	
Sentence segmentation	✓	✓		
Dependency parsing	✓	✓	✓	
Entity recognition	✓	✓		
Stemming	✓	✓	✓	
Lemmatization	✓	✓	✓	

Table 4. Comparison of the functionalities offered by spaCy, NLTK, TextBlob and Stanford CoreNLP

System	Tokenize	Tagging	Parsing	Tokenize	Tag	Parse
spaCy	0.2 ms	1 ms	19 ms	1x	1x	1x
coreNLP	018 ms	10 ms	49 ms	0.9x	10x	2.6x
ZPar	1 ms	8 ms	850 ms	5x	8x	44.7x
NLTK	4 ms	443 ms	n/a	20x	443x	n/a

Table 5. Compare the per-document processing time of various spaCy functionalities against other NLP libraries

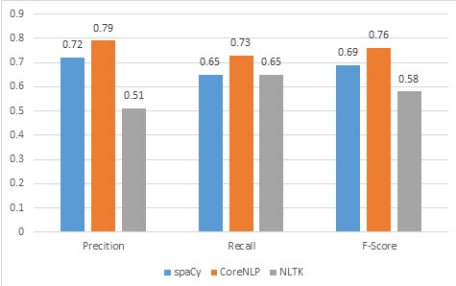

Figure 2. Accuracy for entity extraction

One of the most powerful feature of spacy is the extremely fast and accurate syntactic dependency parser which can be accessed via lightweight API. The parser can also be used for sentence boundary detection and phrase chunking.

The relations can be accessed by the properties ".children", ".root", ".ancestor" etc[12].

4. Feature Level Comparison

The two significant libraries used in NLP are NLTK and spaCy. There are substantial differences between them, which are as follows: NLTK provides a plethora of algorithms to choose from for a particular problem which is boon for a researcher but a bane for a developer. Whereas, spaCy keeps the best algorithm for a problem in its toolkit and keep it updated as state of the art improves.

NLTK is a string processing library and it takes strings as input and returns strings or lists of strings as output. Whereas, spaCy uses object-oriented approach. When we parse a text, spaCy returns document object whose words and sentences are objects themselves. spaCy has support for word vectors whereas NLTK does not. As spaCy uses the latest and best algorithms, its performance is usually good as compared to NLTK. As we can see below, in word tokenization and POS-tagging spaCy performs better, but in sentence tokenization, NLTK outperforms spaCy. Its poor performance in sentence tokenization is a result of differing approaches: NLTK attempts to split the text into sentences. In contrast, spaCy constructs a syntactic tree for each sentence, a more robust method that yields much more information about the text.

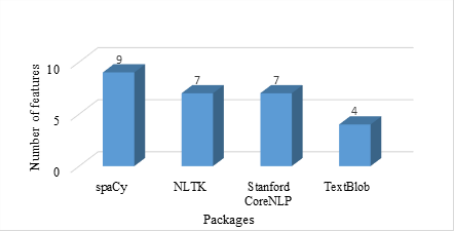

Figure 3. Number of features offered by spaCy, NLTK, Stanford CoreNLP and TextBlob

The most popular NLP tools available in Python, spaCy supports 9 features out of 10. In our observation spaCy have fast processing speed in 3 major key functionalities Tokenization, POS Tagging, Entity Extraction.

SpaCy, on the other hand, is the way to go for app developers. While NLTK provides access to many algorithms to get something done, spaCy provides

the best way to do it. It provides the fastest and most accurate syntactic analysis of any NLP library released to date. It also offers access to larger word vectors that are easier to customize. For an app builder mindset that prioritizes getting features done, spaCy would be the better choice. Both NLTK and spaCy offer great options when you need to build an NLP system. As we have seen, however, spaCy is the right tool to use in a production environment.

5. Conclusion

In this article, we compared some features of several popular NLP libraries. While most of them provide tools for overlapping tasks, some use unique approaches for specific problems. Definitely, the most popular packages for NLP today are NLTK and spaCy. In our opinion, the difference between them lies in the general philosophy of the approach to solving problems.

You can use it to try different methods and algorithms, combine them, etc. spaCy, instead, provides one out-of-box solution for each problem. Also, spaCy is several times faster than NLTK. Despite the popularity of these two libraries, there are many different options, and the choice which NLP package to choose depends on the specific problem you have to solve. spaCy would be an optimal choice for template based question generation.

Acknowledgements

The research reported here was carried out as part of the EFOP-3.6.1-16-2016-00011 "Younger and Renewing University – Innovative Knowledge City – Institutional development of the University of Miskolc aiming at intelligent specialization" project implemented in the framework of the Széchenyi 2020 program. The realization of this project is supported by the European Union, co-financed by the European Social Fund.

REFERENCES

[1] Dominik Kozaczko, *Best python natural language processing nlp libraries*, [Online]. Available: https://sunscrapers.com/blog/8-best-python-natural-language-processing-nlp-libraries/,2018.

[2] K. E. a. P. P. e. Boyer: Proceedings of QG2010. *The Third Workshop on Question Generation, in Pittsburgh*, questiongeneration.org, 2010.

[3] Walelign T.S et al: *The development and analysis of extended of architecture model for intelligent tutoring systems*, Gradus ISSN 2064-8014 , vol. 6, no. 4, pp. 128-138, 2019.

[4] J. P. a. I. S. Dhaval Swali:, *Automatic Question Generation from Paragraph*, International Journal of Advance Engineering and Research Development, vol. 3, no. 12, pp. 73-78, 2016.

[5] M. H. N. A. Smith:, *Good Question! Statistical Ranking for Question Generation, in Human Language Technologies:* Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, , Los Angeles, California, USA, June 2-4, 2010.

[6] N.-T. Le and N. Pinkwart, *Evaluation of a question generation approach using semantic web for supporting argumentation*, Research and Practice in Technology Enhanced Learning , vol. 10, no. 3, p. 19, June 2015.

[7] , G. Keswani,*AutoQuest (An Intelligent Automatic Question Paper Generator System)*, Abdul Kalam Technology University, Lucknow, 2018-19.

[8] K. Mhartre, *Question Generation using NLP*, International Journal of Scientific Research & Engineering Trends, vol. 5, no. 2, pp. 394-397, 2019.

[9] E. L. Fasya, *Automatic question generation for virtual humans*,Enschede, The Netherlands, August 2017.

[10] D. Lindberg, *Automatic question generation from text for self directed learning*,Simon Fraser university, Canada, 2013.

[11] Mandasari Yani: *Follow-up question generation*, University of Twente M.Sc Thesis, 2019.

[12] S BANSAL, *Natural Language Processing Made Easy using spaCy (in Python)*,[Online]. Available: https: //www.analyticsvidhya. com/blog/2017/04/ natural-language-processing-made-easy-using-spacy- [Accessed 19 10 2020].

[13] Awesome Python, *Natural Language Processing packages and projects*, https: //python.libhunt.com/categories/169-natural-language-processing,[Accessed 19 10 2020].

[14] Manning and Christopher D. at al, *The Stanford CoreNLP Natural Language Processing Toolkit*, In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60, 2014.

[15] Online,*Python PoS Tagging and Lemmatization using spaCy*, Available: https://www.geeksforgeeks.org/python-pos-tagging-and-lemmatization-using-spacy/,[Accessed 19 10 2020].

[16] S. Loria, *TextBlob: Simplified Text Processing*, [Online]. Available: https://textblob.readthedocs.io/en/dev/. [Accessed 25 07 2020].

[17] spaCy, *Facts & Figures*, [Online]. Available: https://spacy.io/usage/facts-figures. [Accessed 16 4 2020].

[18] F. N. A. Al Omran and C. Treude, *Choosing an NLP Library for Analyzing Software Documentation A Systematic Literature Review and a Series of Experiments*,IEEE/ACM 14th International Conference on Mining Software Repositories (MSR) DOI: 10.1109/MSR.2017.42, pp. 187 - 197, 20-21 May 2017.

[19] M. H. N. A. Smith, *Good Question! Statistical Ranking for Question Generation, in Human Language Technologies*: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, , Los Angeles, California, USA, June 2-4, 2010.