IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Web APIs -Uses, Roll, Challenges, Design points, In applications

Satya prakash
Assistant Professor
DPBS PG College Anupshahr

Abstract

Web APIs could explore their evolution, impact on software development, security considerations, and emerging trends, focusing on how they facilitate interoperability and data exchange between di web APIs could explore their evolution, impact on software development, security considerations, and emerging trends, focusing on how they facilitate interoperability and data exchange between different systems and applications. fferent systems and applications

Widely adoption of Information Technologies has resulted in the continuous growing of open data available on the Web. However, the lack of suitable mechanisms to understand open data sources hampers its reusability. One way to overcome this limitation is by means of Web Application Programming Interfaces (APIs) with proper documentation, nowadays being the existing very rudimentary, hard to follow, and sometimes incomplete or even inaccurate in most cases. In order to improve the documentation of Web APIs that access open data, this paper proposes a novel approach to automatically generate interactive Web API documentation, both machine and user readable..

Web Application Programming Interfaces (web APIs) provide programmatic, network-based access to remote data or functionalities. Applications, for example, use the Google Places API to learn about nearby establishments, use the Twitter, Instagram, or Facebook API to connect users with friends and family, or use the Stripe API to accept end-user payments. Increasingly, applications themselves consist of micro-services that expose their capabilities to one another using web APIs.

Interfaces play a key role in facilitating the integration of external sources of innovation and structuring ecosystems. They have been conceptualized as design rules that ensure the interoperability of independently produced modules, with important strategic value for lead firms to attract and control access to complementary assets in platform ecosystems. While meaningful, these theorizations do not fully capture the value and structuring role of web APIs in digital innovation ecosystems. Web APIs enable the integration of capabilities from multiple organizations for the co-production of services and products, by interfacing their information systems. Web APIs are important sources of value creation and capture, increasingly being used to offer or sell services, constituting important sources of revenue.

Keywords: Web API , Open API Documentation, Natural language processing ,Natural language generation

1. Introduction

Nowadays, the Web has become an important information platform, thus worldwide governments and organizations have been generating and publishing open data for years producing economic and social benefits, such as innovation and transparency. Indeed, there is a broader audience out there that is really interested in data, which are called data enthusiasts (i.e. data consumers and knowledge-seekers), who are non-technical users (but domain experts) and they do not have extensive programming abilities either . Therefore, data enthusiasts need easy access to open data in order to answer domain specific questions .

A Web API is an application programming interface (API) designed for the web, allowing applications to communicate and exchange data over the internet using HTTP protocols. It enables developers to build web services and applications that can interact with each other, facilitating data access and functionality across different systems.

While the existence of large amounts of open data may be regarded as an advantage for these data enthusiasts, it is actually a pitfall because data can become difficult to consume . Open data portals should offer users the necessary mechanisms to facilitate the discovery, extraction and usage of open data . Therefore, removing barriers for data consumption processes remains a primary concern . In order to do so, the most adopted approach to facilitate the access to open data to users are Web Application Programming Interfaces (APIs) . Actually, they are a key feature of open data platforms, allowing data consumers (such as data enthusiasts) to access data in a convenient manner . However, data enthusiasts may feel overwhelmed when trying to understand Web APIs and use them to access open data. One of the reasons is that using Web APIs require users to be familiar with query languages and a certain level of knowledge about programming.

The Web Api architecture was introduced in the year 2000, by Thomas Fielding, and is based on the principles that support the World Wide Web [5]. In summary, according to the REST principles [5], REST interfaces rely exclusively on Uniform Resource Identifiers (URI) for resource detec tion and interaction, and usually on the Hypertext Trans fer Protocol (HTTP) for message transfer [3], [6], [7]. A REST service URI only provides location and name of the resource, which serves as a unique resource identifier. The predefined HTTP verbs are used to define the type of op eration that should be performed on the selected resource (e.g., GET to retrieve, DELETE to remove a resource).

Application Programming Interfaces (APIs) are constructs made available in programming languages to allow developers to create complex functionality more easily. They abstract more complex code away from you, providing some easier syntax to use in its place.

As a real-world example, think about the electricity supply in your house, apartment, or other dwellings. If you want to use an appliance in your house, you plug it into a plug socket and it works. You don't try to wire it directly into the power supply — to do so would be really inefficient and, if you are not an electrician, difficult and dangerous to attempt.

The context of our research is aligned with the previous work of Abell et al. and Robillard et al. in the sense that a comprehensive and useful documentation is key to facilitate the wide use of APIs. This documentation will promote the reuse of open data because the value of this data is limited by our ability to interpret and comprehend it . Moreover, the importance of this documentation is even more relevant regarding Web APIs that provide direct data access (query-level Web APIs) in order to facilitate the creation of third-party solutions that reuse this data . Therefore, an important factor to attract users and increase the value of APIs consists of facilitating its use by an accurate, complete and interactive documentation . Unfortunately, the available APIs to access open data are generally incomplete and difficult to use because they lack adequate documentation so users must dedicate more time and effort to learn how to correctly use these APIs.

In order to overcome the lack of understandable Web API documentation, Natural Language Processing (NLP) research area could help to generate at the same time readable text by users and process able by machines. This NLP area is a computational subfield of Artificial Intelligence whose main purpose relies on the analysis, processing, generation and representation of natural language . Within the areas enclosed in NLP, when considering the automatic generation of documentation descriptions, the area of Natural Language Generation (NLG) is essential. The main aim of this area is to develop techniques capable of generating human utterances, whether the input to these techniques is text or non-linguistic data . Due to the great functionality and adaptability of NLG, our hypothesis is that the use of its techniques can be beneficial in the present research work about generation of API documentation, providing both interactive documentation and natural language descriptions at the same time.

As the main goal of this research is to provide data enthusiasts with comprehensive APIs to better access open data, we propose an approach that integrates NLP and NLG techniques. These techniques help us to generate machine and user readable documentation of open data Web APIs by including natural language descriptions, which are then easy to read and understand by data enthusiasts' users. As far as the authors are concerned, our proposal is novel for the state of the art, since the main efforts of existing related work are on the generation of documentation for many different purposes in particular scenarios, but they do not generally focus on making the documentation interactive and do not take into account the positive impact of generating automatically natural language descriptions. As new approaches about the generation of API documentation are needed to improve the comprehension of APIs and the reusability of data, we first proposed a preliminary version , and now, we propose a next step in the improvement of open data Web APIs documentation.

In order to verify that our approach provides easier to use Web APIs documentation, we perform a complete evaluation with a case study and an experiment with data enthusiasts. Indeed, in this experiment the interactive and natural language documentation generated by our approach is compared to a basic machine-readable documentation of the same APIs, so that data enthusiasts give their opinions about each documentation. With this experiment we aim to confirm that the problem associated to the difficulty of using current Web APIs by data enthusiasts because of their documentation, can be alleviated by the documentation improvement we propose.

Our contributions to this improvement of Web APIs documentation are:

- A novel freestanding approach for tackling the challenge of data enthusiasts to understand data available through Web APIs.

- A complete documentation which in addition to including easy to process specifications for machines, also adds easy to read descriptions for users, facilitating the reuse of data.

- The natural language descriptions included are generated automatically using a NLG template-based approach in conjunction with semantic knowledge, rather than documenting manually each API. It reduces the cost and effort of documenting APIs, thus enhancing the data reuse process.

- Interactive documentation which facilitates the use of Web APIs to easily obtain the desired data, reducing the human-error factor.

- An evaluation with 20 data enthusiasts, each of them accessing a Web API with a basic machine-readable documentation or an interactive natural language documentation generated by our approach, so that we can compare the results.

2. Related work

In this section, existing related work is investigated. First of all, we review existing types of Web API documentation in order to know the current situation of Web APIs to access open data. Then, we explore current solutions to support Web APIs undersandability, such as those regarding automation and NLP.

2.1. Human vs machine readable API documentation

The documentation of Web APIs is generally very rudimentary, hard to follow, and sometimes incomplete or even inaccurate . Moreover, this documentation is mostly manually written and provided as plain text, preventing users to take advantage of having a machine-readable specification automatically generated . Data enthusiasts that deal with Web APIs have to face these documentation obstacles. Therefore, they need to deeply evaluate APIs in order to understand and use them correctly , requiring human effort to create them and excessive time to understand that documentation.

Regarding documentation types, with the information gathered from existing research, . The existing documentation of Web APIs can be: (i) presented as plain text without following any kind of documentation standard, which promotes the readability for users but is difficult to process by machines; (ii) created manually following the Open API standard, which is machine-readable but because of the human factor it is error prone and has a high cost; (iii) generated automatically in Open API, which is general sable (can be applied for any API) and machine-readable, but it is difficult to read by users; and finally, (iv) interactive Open API documentation, which can be automatically generated or not, which is easy to process by computers and may be easier to use by users. Although this interactive documentation helps users to query Web APIs, the information about what the API offers may still be difficult to read by users, hindering the use of this API despite its interactivity. This kind of documentation is better compared to those in Open API without interactivity, but is not generally available on current Web APIs. Therefore, there could be a further step in the way of providing documentation of an API: interactive Open API documentation with NL descriptions generated automatically. This interactivity means that users would be able to query the data from the API just by clicking the documentation, which works as a Web interface. In this case, in addition to being easy to process by computers, it would also be easy for users to use the API by this kind of interactive documentation.

Table 1. Advantages and disadvantages of different API documentation types.

Documentation type	Advantages	Disadvantages
Plain text (not following a standard)	User readable	Difficult to process by machines
OpenAPI manually created	Machine processable	Error prone and higher cost
OpenAPI automatically generated	Processable and generalisable	Difficult to read by users
Interactive OpenAPI	Processable and easy to use	Still difficult to read

As seen in Table 1, the different types of Web API documentation not only offer advantages such as being easy to use or to process, but also disadvantages such as difficulty in their use. There may seem that there is no adequate documentation that offers all the benefits but without the drawbacks. However, we propose to offer a documentation easy to read for users because it includes natural language descriptions, easy to

process by machines as long as it follows the OpenAPI standard, easy to obtain since it is generated automatically and not by hand, and easy to use without errors due to its interactivity.

2.2. Supporting web APIs understandability

In this section, related work regarding the comprehension of Web APIs is reviewed: efforts regarding API documentation are first discussed, and then, the generation of text from the NLP perspective is analysed.

From the point of view of the API documentation generation, the main efforts have been focused on the creation of basic documentation (i.e., a machine-readable specification), leaving the interactivity in a secondary plane. In general, the documentation is presented as plain text or following an open standard, but they are not presented as interactive (i.e., a Web interface to interact with the documentation itself). One example that considers the interactivity of API documentation addresses the generation and design of Web frontends based on OpenAPI documentation. However, the tool presented in needs to be handled manually by developers in order to improve the documentation for end users, but the enhancement of the documentation for non-technical users is not addressed and thus they may still have difficulties in understanding and using APIs.

On the other hand, as the documentation can be in natural language, or at least the descriptions of each method or parameter of the API, the use of natural language generation techniques is needed. Indeed, NLG has been used in many applications, such as text summarization, dialog systems or the generation of simplified texts [. Moreover, it has been also widely employed and integrated in different research areas, such as in computer vision, for the generation of textual descriptions for human activities in videos or in business intelligence, for the generation of reports and real time notifications about the state of a company's information technology services . However, the automatic generation of natural language descriptions is left in a secondary plane by related work. In general, related research approaches provide the documentation of Web APIs with manually written descriptions, including methods and parameters. In other cases, these descriptions do not include natural language as they are generated as a rough draft containing a few keywords related to the overall behaviour of the method, or they are not generated at all. Some examples of this type of approaches can be found in , where a set of techniques for generating structured documentation of Web APIs from usage examples are detailed. The authors propose a first step towards automatically learning complete service descriptions. However, the generation of methods' descriptions is not tackled. Also, the authors of this paper proposed an automatic API generation process which also generates API documentation, but it includes very simple descriptions with keywords extracted from API methods, without exploiting the potentials that the integration of NLP techniques could provide. In , a new framework for generating titles for Web tables is presented. This is accomplished by extracting keywords that are relevant to the table. The proposed technique is the first to consider text generation methods for table titles. However, NLG is not applied to generate documentation and it is based on existing table descriptions in plain text. Another paper presents a technique to automatically generate human readable summaries for Java classes. The proposed tool determines the class and method stereotypes and uses them to select the information to be included in the documentation. However, this text generation approach is only valid for programming code documentation and it does not address the description of APIs in natural language. Moreover, the documentation generated is only about a general vision of the Java class, but the explanation of their methods is missing in contrast to our proposal. There are also some research works that deal with automatically inferring API specifications from manually written documentation.

The importance and usefulness of API documentation, especially in OpenAPI format is emphasised . They deal with generating models from API documentations, but the problem of API documentation descriptions' quality is not addressed and they do not address the generation of these descriptions in natural language. Moreover, the research presented is focused on improving existing data-intensive APIs and their

maintenance through the analysis of their usage by users. From this analysis, the documentation of the API can then be improved. Related to the readability of Web Services.

From the NLP point of view, the NLG task is essential since it allows the automatic generation of text regardless of the type of input (e.g., text or non-linguistic data). This task has been commonly addressed through the use of knowledge-based approaches. This type of approaches commonly rely on the use of templates and rules for generating text . For example, PASS is a system that describes non-linguistic data, which generates soccer reports. This system creates a summary of a specific match employing a template-based approach. The input to this system are the match statistics, as well as heterogeneous data such as the league, the date, the match events, the players, the total number of shots or the accuracy of the passes. Although this research line could As seen in the existing related work, dealing with the automatic generation of both machine and human documentation for Web APIs is barely addressed. There are some research that deals with similar problems, but in some cases the problem arises in other scenarios, such as Web services rather than APIs. Therefore, to the best of our knowledge, the solution proposed in this paper is novel and goes beyond the state of the art in providing the suitable and interactive documentation for Web APIs, in a fully automatic manner, by integrating NLP techniques into the API documentation generation process.

3. NLP For improving API documentation

In this section, our approach[1] for the documentation generation process is described. This constitutes a further step on the documentation of data Web APIs, by improving and enriching current documentation with suitable descriptions in natural language (NL). These descriptions are automatically generated and included in an interactive adaptation of the OpenAPI documentation. Specifically, the process of improving the documentation is able to add NL descriptions to any existing OpenAPI documentation of a Web API. With this incorporation of descriptions, the documentation of Web APIs will be not only machine-readable but also understandable by users. Moreover, this documentation of the API is made interactive, which means that it is presented through a Web interface so that users can directly query the underlying API to easily get the data. Therefore, the enhancement of the comprehension and use of Web APIs will promote as a result the reuse of data.

The process of improving open data Web API documentation starts with an existing OpenAPI documentation of an open API. After that, the generation of natural language descriptions are automatically generated and integrated in the existing initial documentation. In this manner, a more detailed and comprehensive documentation is obtained. Finally, this documentation is made interactive by creating a Web interface version with the help of Swagger[2] open source specification and tools. This whole process is now explained step by step in the following subsections.

3.1. Starting point: An existing openAPI documentation

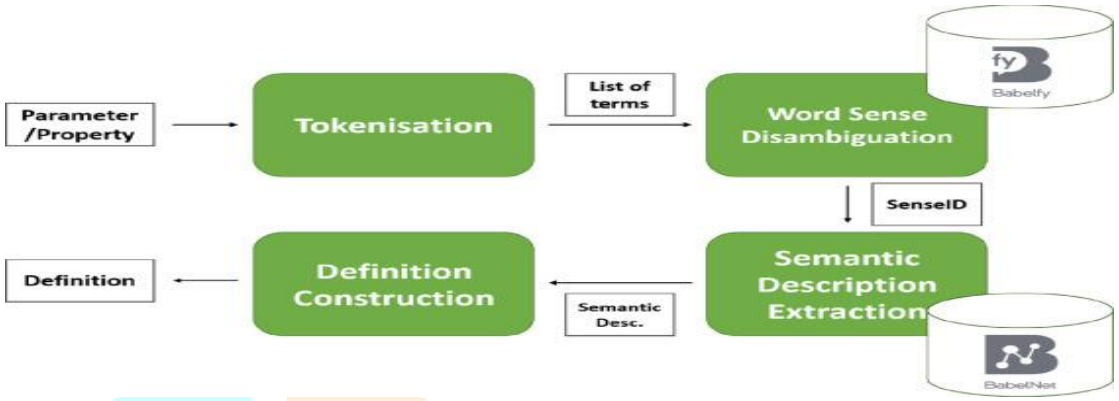
The process starts with a basic machine-readable API documentation following the OpenAPI standard in JSON format. Then, using the proposed NL descriptions generator approach, a set of descriptions in natural language is created and later appended to the documentation of the API, facilitating the understanding of the Web API.

Web API including an OpenAPI documentation which is only machine-readable. This documentation is not easy to read by human users, so generating NL descriptions of the API is essential to better understand and access it to get the underlying data.

In order to generate these NL descriptions, the automatic generator process first gathers the following information about the API from this starting OpenAPI documentation: the title given to the API; the name of each method; all the parameters from each method, with name, type and example value; and all the properties given as a result of the API, also with name, type and example data. This information is extracted from the input documentation by analysing the JSON object that contains all the components of the API

documentation. In order to do so, the process iterates through the JSON objects and arrays that the OpenAPI standard specifies, such as API “title”, “paths”, “components”, “parameters” and “properties”.

Once the aforementioned information is gathered, the process continues with the next step in order to automatically generate the general description of the API as well as a description for each of the API methods, parameters and properties. This generation process will be further explained in the next subsection.



3.2. Natural language descriptions generator

Taking as input the information extracted in the first step of our approach’s architecture, the second step is the generation of descriptions using NLP techniques. From this information, the generation of NL descriptions is performed using and integrating different NLP techniques. Specifically, tokenization, word sense disambiguation and a NLG template-based approach are employed to generate the descriptions that will be added to the OpenAPI

The templates to be used in this proposed approach are designed considering as reference the generic CKAN API[3], which is used by many open data platforms such as Data.gov (the U.S. Government’s open data platform) and Data.gov.uk (the U.K. Government’s open data platform). Since the text needed to describe the different parts of the API must be different, a variety of 3 different generic templates were hand-crafted : (i) general API description; (ii) API method description; and (iii) parameter/property description. Within these templates, the most important information is shown in bold in order to highlight it to the readers. In addition, there are different variables used along the templates to include the information coming from the API: ${fileName} (name of the dataset to access through the API), ${params} (list of available parameters to filter data), ${methodName} and ${methodName2} (names of two methods of the API as example), ${methodEx} and ${methodEx2} (example values for two methods of the API), ${recordExample} (API response example in JSON format), ${ptype} (indicates if it corresponds to a parameter or a property of the API), ${pname} (name of the parameter or property), ${datatype} (parameter or property data type such as string), ${pexample} (example of parameter or property value), ${pdefinition} (definition of the parameter or property name), ${required} (if the parameter is required in the method call) and ${pin} (if the parameter must be in query or in the path of the API request).

3.3. Generating an interactive openapi documentation

Finally, the last step consists of creating an interactive documentation which is easy to process by machines and easy to read by data enthusiasts at the same time . First, the generated descriptions in natural language of the API are integrated in the existing (machine-readable) documentation of the API. Then, this documentation is going to be made interactive by automatically creating a Web interface. In order to do so, the necessary Web programming code (in HTML, CSS and Javascript languages) is appended to the documentation, which is incorporated in a Web server in the NodeJS environment to allow interactivity. Therefore, this documentation of the API will be now both machine and user readable, all in a single and interactive OpenAPI documentation

the OpenAPI documentation including descriptions in natural language is included in an automatically generated server to offer the documentation as a Web interface. It consists in a NodeJS server that is created with the help of the Swagger Codegen tool[7], which creates the structure of the server and manages the calls to the underlying API. One of the main benefits of having an interactive documentation is the consistency of the results, which means that the possibility of introducing errors such as writing mistakes is severely reduced, and therefore there is less chance of obtaining erroneous data. For example, if users want to filter the data by a parameter called "Myarea_Type", they have to write the parameter exactly taking into account symbols and uppercase. However, with our interactive approach users do not need to write down the parameter and they can just click on the specific method for filtering.

4. Evaluation

In this section the proposed approach is first evaluated with different examples. Then, a specific case study is introduced, and finally, an experiment with real users is presented. To ensure the correct performance of the proposed approach, a set of 5 different OpenAPI documentations have been used. Each documentation is related to one Web API that provides direct access to specific data about different topics.

time spent by the proposed approach for generating the OpenAPI documentation with natural language descriptions. As can be seen, the time required to generate the API documentation by our proposed approach is affected by the number of API methods'. In this sense, the disambiguation process would introduce a delay in generation time for each method. Taking into account that the Web APIs contain between 7 and 46 methods, the time to generate the related API documentation including natural language descriptions is between 24 and 215 seconds.

4.1. Case study

A case study is now introduced to illustrate the whole process and show the feasibility and usefulness of our proposal. It consists of applying the proposed approach described to an existing data Web API. With this example we attempt to demonstrate that data enthusiasts need to make a great effort to interpret and understand the available third-party Web APIs because of the lack of suitable human-readable API documentation, which also hampers the reusability of data.

Our scenario includes open data from a Web API providing employment statistics, from which we are going to create an interactive OpenAPI documentation with natural language descriptions.

4.1.1. Generating documentation for employment data

An API providing employment statistics has been chosen to illustrate this process. The information provided by the API is about employment in a wide variety of professions in New York. an extract of employment data from the dataset, which contains information such as occupation, area name or wage. This data originally comes from the U.S. Government's open data website (data.gov).

4.2. Experiment with users

To evaluate our approach with real users, an experiment comparing the use of OpenAPI documentation with the use of interactive OpenAPI documentation with NL descriptions created by our approach was conducted. This experiment was carried out with data enthusiasts using the original basic documentation and the generated by our approach for 3 Web APIs selected which makes a total of 6 API documentations. For each documentation, we supplied a different survey. In this experiment with users, 20 participants from Alicante (Spain) accessed specific data using the related Web API and its documentation. These participants were data enthusiasts without programming skills which attended a big data seminar oriented to take advantage of data in different scenarios, which was conducted by the Employment Centre of the University of Alicante

4.2.1. Experiment setup

First of all, we provide an explanation of the experiment for the participants in a website[13] we created specifically for the experiment. This website includes a link to one of the 6 available surveys, randomly for each participant to take a different survey (only one). These 6 surveys are different because there is 3 available Web APIs (their structure is the same but they are about different data), and each one has the option to use the original documentation or the documentation generated using our NL documentation approach. However, the difficulty of the surveys remains the same and also the questions for participants are equal despite the documentation they are using, the only changes affect the data asked depending on the API they have to query (because the data offered by the APIs is different).

A preliminary training phase was conducted in order to help data enthusiasts to know about APIs and OpenAPI documentation. Therefore, we provided in the experiment website[13] an explanatory video about what is a Web API, how to access data and how to read an OpenAPI documentation.

Then, users were asked to fill in an online survey where they have to perform the following tasks: (i) read the provided documentation; (ii) query the specified API to get specific data; (iii) answer some questions about the retrieved data; and finally, (iv) answer general questions about the documentation used and their satisfaction about the process of accessing data using the API and its documentation.

In this evaluation we recorded the time spent to answer the questions and the correctness of answers in order to know the effort they needed to query the API using the provided documentation and their efficiency. Therefore, the data analysis used for this experiment consisted of analysing the time in seconds for the correct answers with ANOVA, using the type of documentation and the specific API as fixed factors. It is also important to consider that the time has been transformed with square root in order to avoid heteroscedasticity (circumstance in which the standard deviations of a predicted variable are non-constant).

4.2.2. Online surveys

First of all, in the introduction of the survey we offer information about the API and we encourage users to carefully read the provided documentation before answering the survey. Then, users are asked to perform 6 different queries to the API using the documentation. One of the queries consists of obtaining the mean wage of financial managers employees from “Long Island Region”. We also asked the user about the query used to obtain this required data.

Finally, the general questions were related to gather information about previous experience in the use of Web APIs, difficulty in the queries performed, usefulness of the documentation, satisfaction with the documentation and possible improvements or changes for the documentation used.

4.3. Discussion

From this evaluation we can state that the inclusion of interactivity and NLP techniques improve the documentation of Web APIs, which can now be easily understood by users and processed by machines at the same time. Compared to not having any documentation, or having only a machine-readable documentation that only includes the names of the API methods, our approach contributes to the existing related work by providing both human and machine-readable documentation, which simplifies the comprehension and reusability of the data. Furthermore, the generated documentation allows users to

query the underlying API through different methods by the documentation Web interface, thus facilitating the access to the data.

These results can be justified as it is normal that with the original basic documentation, users take less time to answer the surveys than with the generated NL documentation because they do not understand machine-readable documentation and thus they have less to read. However, the main problem is that with the basic documentation there are many more errors because data enthusiasts can be confused with invalid results that they believe to be true. Instead, when using our generated documentation it is advantageous against the misunderstanding as well as the misleading use of the Web API.

therefore, we have evaluated with examples and users that the proposal successfully achieves the objective of generating the suitable Web API descriptive documentation in different situations. The importance of including natural language descriptions in OpenAPI documentation is that it actually helps users to reuse existing data and citizens to be able to access the data offered on the Web.

5. Conclusions and future work

In this paper we have presented an approach that integrates NLP techniques to generate interactive documentation of open data Web APIs, easy to read and understand by data enthusiasts users. Our approach starts with a basic machine-readable API documentation in the OpenAPI standard, which is easy to process by machines but difficult to understand by common users, hindering the reuse of data. From this documentation, we propose a natural language description generator to create a set of descriptions in natural language and append them to the existing machine-readable documentation of the API. This process is based in NLP and specifically in NLG techniques that allow us to create NL descriptions from important concepts of the initial documentation. Moreover, this documentation is made interactive by presenting it as a Web interface to facilitate even more the task of querying the API. For evaluating the proposed approach, we tested the automatic generation of documentation process with several machine-readable documentations of datasets from the Data.gov open data portal. After that, we have presented a case study in which our approach is applied within a specific scenario and then evaluated with data enthusiasts in an experiment. In this sense, we illustrated and described a real-based situation where users are interested in obtaining specific data from an API. The proposed approach (publicly available at GitHub[1]) is a key element for improving data management and analysis. This is because enhanced API documentation would lead to a better understanding of the API by users, facilitating the access and handling of open data available online. Regarding this implementation of the approach, the generation of enriched descriptions is independent of the API generation, being this step easily applied to existing APIs from different contexts, as well as the API generation process can be applied from different data sources available online. Moreover, the interactivity of the proposed approach allows users to easily access the documentation's webpage and directly query the underlying API by performing a few clicks. Therefore, the implementation allows to easily use and extrapolate the solution for existing problems regarding documentation of APIs.

Limitations of our approach mainly come from the challenges that NLG embraces. As stated in [, it is necessary not only to describe how the NLG is used for a specific task, but also to discuss its limitations to show the complexity of the task and the challenges that need to be addressed. The first limitation of our approach is related to the unavailability of metadata found in open data sources . This may prevent NLG from generating satisfactory sentences for useful Web API documentation. The use of external knowledge resources could improve this limitation. Another important limitation of NLG is related to hallucination from the data , i.e., the generation of texts that are apparently well written but they are unsubstantiated and not faithful to the provided data. In our approach, templates are used to try to overcome this limitation when documentation of Web APIs is generated from open data sources.

As future work, the generation process will be extended by using semantic Web technologies to apply data integration mechanisms. With regard to the generation of descriptions within the API documentation, the NLP area also provides with techniques that allow the adaptation or customization of the generated descriptions depending on the user needs. In this sense, the descriptions could be simplified according to a specific linguistic level or could also include more technical terms if required. In addition to this, this approach could be easily extended to other languages (i.e., multilingual) since the semantic resources employed are linked to many languages, which would facilitate the reuse of code. Finally, studying how our proposal could impact the software industry is an interesting avenue for future work, as according to the poor documentation of APIs for accessing open data causes two main problems for entrepreneurs and companies: on the one hand, the lack of documentation hinders the effort to design and develop open data-based services and software products; and on the other hand, even if the documentation exists, software developers have difficulties in understanding and seeing what open data the APIs can access.

ACKNOWLEDGEMENTS

the server's response to a client request, indicating whether the request was received and processed successfully, or if there were any issues. This is often conveyed through HTTP status codes and headers.
This work has been partially supported by the project oxo, Cooperation Programme, QBSS 2024 grant agreement

Reference

[1] E. Wittern, A. Cha, and J. A. Laredo. 2018. Generating GraphQL-Wrappers for REST(-like) APIs. In *Web Engineering.* Springer International Publishing..

[2] E. Wittern, A. T. T. Ying, Y. Zheng, J. Dolby, and J. A. Laredo. 2017. Statically Checking Web API Requests in JavaScript.

[3] E. Wittern, A. T. T. Ying, Y. Zheng, J. A. Laredo, J. Dolby, C. C. Young, and A. A. Slominski. 2017. Opportunities in Software Engineering Research for Web API Consumption.
F. Bülthoff and M. Maleshkova, “RESTful or RESTless – Current State of Today’s Top Web APIs,” in The Semantic Web: ESWC 2014 Satellite Events, 2014, pp. 64–74.

[4] J. Kopecký, P. Fremantle, and R. Boakes, “A history and future of Web APIs,” It - Inf. Technol., vol. 56, 2014.

[5] R. T. Fielding, “Architectural Styles and the Design of Network based Software Architectures,” Univ. of California, Irvine, 2000.

[6] R. Battle and E. Benson, “Bridging the semantic Web and Web 2.0 with Representational State Transfer (REST),” Web Semant. Sci. Serv. Agents World Wide Web, vol. 6, no. 1, pp. 61–69, Feb. 2008.

[7] B. Costa, P. F. Pires, F. C. Delicato, and P. Merson, “Evaluating a Representational State Transfer (REST) Architecture: What is the Impact of REST in My Architecture?,” in 2014 IEEE/IFIP Confer ence on Software Architecture, 2014, pp. 105–114.

[8] G. Schermann, J. Cito, and P. Leitner, “All the Services Large and Micro: Revisiting Industrial Practice in Services Computing,” in Service-Oriented Computing – ICSOC 2015 Workshops, 2015, pp. 36–47.

[9] C. Rodríguez et al., “REST APIs: A Large-Scale Analysis of Com pliance with Principles and Best Practices,” in Web Engineering, vol. 9671, A. Bozzon, P. Cudre-Maroux, and C. Pautasso, Eds. Cham: Springer International Publishing, 2016, pp. 21–39.

[10] F. Haupt, F. Leymann, A. Scherer, and K. Vukojevic-Haupt, “A Framework for the Structural Analysis of REST APIs,” in Soft ware Architecture (ICSA), 2017 IEEE International Conference on, 2017, pp. 55–58.
[11] Mark Masse, REST API Design Rulebook. 2011.