computers

MDPI

Review

Agile Development Methodologies and Natural Language Processing: A Mapping Review

Manuel A. Quintana [1,†], Ramón R. Palacio [2,*,†], Gilberto Borrego Soto [3,†] and Samuel González-López [4,†]

[1] Departamento de Eléctrica y Electrónica, Instituto Tecnológico de Sonora, Ciudad Obregón 85000, Mexico
[2] Unidad Navojoa, Instituto Tecnológico de Sonora, Navojoa 85860, Mexico
[3] Departamento de Computación y Diseño, Instituto Tecnológico de Sonora, Ciudad Obregón 85000, Mexico
[4] Departamento de Tecnologías de la Información, Universidad Tecnológica de Nogales, Nogales 84097, Mexico
* Correspondence: ramon.palacio@itson.edu.mx
† These authors contributed equally to this work.

Abstract: Agile software development is one of the most important development paradigms these days. However, there are still some challenges to consider to reduce problems during the documentation process. Some assistive methods have been created to support developers in their documentation activities. In this regard, Natural Language Processing (NLP) can be used to create various related tools (such as assistants) to help with the documentation process. This paper presents the current state-of-the-art NLP techniques used in the agile development documentation process. A mapping review was done to complete the objective, the search strategy is used to obtain relevant studies from ScienceDirect, IEEE Xplore, ACM Digital Library, SpringerLink, and Willey. The search results after inclusion and exclusion criteria application left 47 relevant papers identified. These papers were analyzed to obtain the most used NLP techniques and NLP toolkits. The toolkits were also classified by the kind of techniques that are available in each of them. In addition, the behavior of the research area over time was analyzed using the relevant paper found by year. We found that performance measuring methods are not standardized, and, in consequence, the works are not easily comparable. In general, the number of related works and its distribution per year shows a growing trend of the works related to this topic in recent years; this indicates that the adoption of NLP techniques to improve agile methodologies is increasing.

Keywords: natural language processing; NLP; NLP toolkits; agile software development

Citation: Quintana, M.A.; Palacio, R.R.; Borrego Soto, G.; González-López, S. Agile Development Methodologies and Natural Language Processing: A Mapping Review. *Computers* **2022**, *11*, 179. https://doi.org/10.3390/computers11120179

Academic Editors: Phivos Mylonas, Katia Lida Kermanidis and Manolis Maragoudakis

Received: 15 October 2022
Accepted: 15 November 2022
Published: 7 December 2022

1. Introduction

Agile software development has boomed in the last two decades, being one of today's most important software paradigms. Agile development aims to enable an organization to be faster and flexible [1]; one of the most important bases of agile development is the continuous delivery of software as established in the agile manifesto principles [2]. These principles imply that different methodologies have common bases, such as the iterative approach of small deliveries of functional software. One of the principles establishes that "the software working is the main measure of progress" [2]; constant interaction with clients is another fundamental principle maintained throughout the development.

Software development teams currently use agile methodologies depending on what they want to prioritize. For example, Scrum [3] is an agile methodology that reduces the documentation artifacts compared to traditional development methodologies. Iconix [4] is a semi-agile methodology that also reduces artifacts, but retains some important ones, including the robustness diagram [5]; eXtreme Programming (XP) [6] is intended to improve software quality and responsiveness to changing customer requirements; and Crystal [7] is considered as a lightweight or agile methodology that focuses on individuals and the interactions.

According to the agile manifesto, the best architectures, requirements, and designs emerge from self-organizing teams [2]. Self-organization of teams is fundamental to car-

rying out the documentation process in an agile project to accelerate that process. One of the main points is that self-organization implies no rigid standards for accepting documentation, which speeds up the documentation process. However, this documentation flexibility also entails problems such as inadequate documentation of requirements if the team organization fails [8].

The lack of adequate documentation of changes during an agile project leads to a problem called technical debt. This debt is generated by the activities of the development team members when they do not do certain project activities in due time, preventing or hindering future development [9]. The longer one waits to carry out the postponed tasks, the more costly (in terms of time) it will be for the project, and the more difficulties will carry with it.

It is common to receive changes throughout the project's development; in fact, accepting changes is established in a principle of the agile manifesto [2] to provide a competitive advantage to customers by development flexibility. This situation means that documentation must be updated constantly throughout development. However, developers do not update documentation for different reasons such as lack of time caused by high priority delivery [10] or reluctance to documentation [11].

Some methods to reduce technical debt and documentation problems are based on simplifying the requirements gathering [12]. Some others rely on text analysis to obtain information from documentation, such as code comments, to find information about the project's functionalities [13]. A specific method that could be used to develop text analysis tools is natural language processing (NLP) [14], which belongs to the Deep Learning [15] field. NLP consists of a range of theoretically motivated computational techniques to analyze and represent texts to achieve human-like language processing. It can be divided into three broad categories: lexical analysis, semantic analysis, and discourse analysis. An NLP system that contemplates the three categories is called a complete NLP system. This type of system should perform complex tasks such as paraphrasing input texts, translating a text from a source language to a target language, responding to questions about the content of a text, and producing inferences from an input text [15]. Various techniques and toolkits have already been developed and tested in the NLP area that quickly and efficiently works on text analysis. Stanford CoreNLP [16] is a reasonably robust tool that allows you to use various NLP techniques with a single tool. The Natural Language Toolkit (NLTK) [17] is another tool with a wide variety of modules, such as parsing, tagging, and classification, among others. New toolkits are recently under development, as with Stanza [18], a Python-based tool supporting 66 languages. It has many techniques similar to the previous ones, but with some new modules that include newer NLP techniques.

NLP has been used to process requirements, gathering artifacts such as use cases [19] and user stories [20]; in fact, there is a preference for using NLP to process user stories [21]. This preference may be due to the user stories' characteristics [22], i.e., simple texts which are structured so that the most important elements of each requirement can be identified and their quality can be checked [23]. User stories are a method of representing requirements using a simple structure such as "Who?", "What?", and "for what?" [24]. In agile development, it is common to use user stories as an artifact for requirements gathering [21]. User stories allow the users to express requirements simply; this helps to improve the requirements gathering process and makes the process faster [23]. A quality standard is established to write user stories; that is, the quality of what is expressed is controlled to avoid ambiguity and problems that may lead to a misunderstanding of the requirements [25].

Agile requirements engineering (ARE) is a flexible and fast requirement gathering process that uses user stories as artifacts to collect requirements [26]; this process is iterative and incremental. Unlike traditional requirements engineering, where requirements gathering takes place only in the early phases of the project, agile requirements engineering allows clients to interact and communicate with the development team throughout the entire project development cycle [27,28]. According to [29], software requirements are written in natural language to improve communication with other members of the development teams in such a way that requirements are understandable for all the stakeholders [30]. The

capacity of NLP for data processing leads to exploring solutions to requirements-related problems [31].

This paper presents a Systematic Mapping Review (SMR) to show the existing research on applying NLP techniques and toolkits in the documentation process of agile development methodologies. The mapping was carried out following the method proposed by Petersen et al. [32]. The rest of the document is structured as follows. Section 2 presents background concepts about NLP and related works. Section 3 presents the materials and methods including the mapping planning, research questions, search strategy, and selection strategy. Section 4 presents the results found in the mapping. A discussion of findings is presented in Section 5. Finally, in Section 6, the conclusions of the work are presented. The articles considered relevant for this mapping are presented in Appendix. A

2. Background and Related Work

NLP is the ability of computer programs to understand human language as it is spoken and written [15]. NLP's more important applications include such as (1) machine translation (the learner must read a sentence in one human language and emit an equivalent sentence in another human language), (2) parsing (the process of determining the syntactic structure of a text), and (3) text classification (categorization of the elements of a sentence as nouns, verbs, adjectives, and adverbs) [14,15].

Different NLP techniques enable researchers to solve problems using one or many mixed techniques. Additionally, there exist NLP toolkits [33] that bring researchers a variety of NLP techniques in a simple way to use and implement, which eases the development of NLP systems. On the other hand, various evaluation methods have been established for measuring the performance of an NLP system [34].

To better understand the different NLP techniques, we describe some of the most common below, grouped by categories based on their purpose.

- **Processing Techniques:** This category encompasses those techniques that process the information of the text to find and extract specific data to be analyzed.
  - **Part-of-Speech (POS) Tagging.** This technique categorizes the elements of a sentence as nouns, verbs, adjectives, and adverbs. The labeling grants that differentiation of the elements of a sentence or a complete text inclusive [35].
  - **Tokenization.** The tokenization process consists of converting some text into a correspondent representative token; this token is an object that could be an array, an alphanumeric string, a number, or a customized object. Tokenizing text elements can be from a word-to-word conversion or a group conversion of sentences that share characteristics of interest. Before applying another technique, this technique is commonly used as a pre-processing or first step in NLP pipelines. Even POS Tagging has an implicit tokenization step to carry out the tagging [33].
  - **Parsing.** It consists of identifying the syntactic structure of a text by analyzing its words based on the language's grammar of the given text. One important element of this technique is the parsing tree, which represents the structure of the text and can be represented visually [36,37].
  - **Chunking.** This technique can be defined as a process used to identify parts of speech and even short phrases present in a sentence. The identification goes hand in hand with grouping these for later analysis by identifying all the nouns, verbs, adverbs, etc. [38].
  - **Lemmatization.** The goal of this technique is to reduce the inflectional and derived forms of a word to a common base form for the entire text. A word's base or simple form is a lemma, hence the technique's name. Unlike a simpler technique called stemming, which heuristically clips a word in the hope that clipping will obtain the base form of that word, lemmatization is carried out through morphological analysis and the use of a vocabulary dictionary for proper and efficient conversion [39].

- **Text representation:** This category encompasses those techniques that transform the input text to obtain a new representation of the same information on a different format.
  - **Word2Vec.** This technique focuses on word-centered processing, which can be approached as a bag of words to see the importance of words in a text, the frequency and weight that each one implies, and to seek to perform certain operations to obtain information as the main topic of a text [40].
  - **Bag-of-Words.** The bag-of-words (BoW) model is one of the simplest feature extraction techniques used in many natural language processing applications, such as text classification, sentiment analysis, and topic modeling. Bag-of-words is built by counting the number of occurrences of unique features such as words and symbols in a document. The bag-of-words model is a representation that turns arbitrary text into fixed-length vectors by counting how many times each word appears. This process is often referred to as vectorization [41].
- **Extraction techniques:** This category encompasses those techniques that process the input to extract desired data. The input could be text or the output of processing or text representation techniques.
  - **Semantic Analysis.** Semantic analysis is the process of understanding a text written in natural language based on context and meaning. In general, semantic analysis processes the whole text to identify the real meaning of a text using the identification of elements and to assign logical and grammar roles to each based on the complete analysis [42].
  - **Correlation and Dependencies Analysis.** A dependency analysis and a correlation analysis are closely related to parsing. Both seek to identify the elements that make up the text structure and the relationships between them, and their impact on the functionality of the wanted text message. This process can be seen as relevant to POS Tagging [43].
  - **Sequence Analysis.** This technique is based on the use of number sequencing generated by tokenizing elements of a text corpus. By assigning a number to each word present in the text, different sequences found in the text can be generated and analyzed and, in this way, locate the importance of the order of the words or the effect it has on the text when analyzing other elements of it [44].
- **Grouping techniques:** This category encompasses those techniques that process the input to create groups based on some specified characteristic of the data.
  - **Clustering.** It is based on locating different words or sentences and grouping them by some common characteristic; the greater the similarity of the text, the closer they will be. The topics of different sentences could define similarity, the type of language used, or even the text grouping depending on which person is referred to by each sentence. The purpose of clustering is to create a dynamic number of groups (compound of similar words or sentences) without the need to analyze all the text manually [45].
- **Neural Networks.** Although artificial neural networks are not an NLP technique, it was decided to group all those approaches that use different types of neural networks in this category. Some types are recurrent networks (RNN), convolutional networks (CNN), or Long Short Term Memory (LSTM). This type of approach uses neural networks for the processing of text data, either for classification or obtaining information [15].

Related Work

To our knowledge, no secondary studies focused on using NLP to improve processes within agile development methodologies. However, some studies address similar issues, focusing on specific aspects, such as analyzing user stories using NLP [46]. Other studies focus on the general use of intelligent tools in agile methodologies [47]. Wagner et al. [48]

discussed the importance and the potential advantages of documentation written in natural language in agile methodologies.

The above-cited works show NLP techniques used in various ways to work directly with user stories [46]. Additionally, they show the challenges encountered in applying NLP techniques, such as the need to improve validations of performance during the training of NLP applications. Another challenge is to find adequate data sets for testing and training; these data sets must contain enough data to ensure the efficient learning of the tool during its training process. Further, the cited works identified a challenge in creating tools independent of the context so that they can be generalized to new data that was not presented in the training phase, such as incorporating human intervention in performance evaluation to complement the effectiveness of the tools [15,34].

3. Materials and Methods

The method was based on the guidelines proposed by Petersen et al. [32] to carry out systematic mapping reviews. The method includes objectives, research questions, search strategy, and selection criteria. The study focuses on finding the applications of NLP in agile methodologies to identify, characterize, and summarize the research of the evidence in the literature on the use of NLP focused on improving the documentation process of agile development methodologies.

3.1. Definition of Research Questions

The main question that we want to answer with this review is the following:

What approaches based on natural language processing techniques exist in the literature focused on improving the documentation process of agile development methodologies?

Due to the main question being too broad, we divided it into four more specific questions, as shown below.

- **RQ1**. What approaches have been made to improve the agile documentation process using NLP techniques?
- **RQ2**. What NLP techniques and toolkits have been used to improve the documentation process of agile development?
- **RQ3**. How is the performance measurement of the NLP techniques when applied to support the documentation process of an agile project?

3.2. Search Strategy

Databases selection was made based on the access to them as follows:

- ACM Digital Library;
- ELSEVIER—ScienceDirect;
- IEEE Digital Library;
- Springer;
- Wiley.

We selected general terms taking into account the general topics of the research:

- NLP;
  - Natural Language Processing;
  - Natural Language;
  - NLP.
- Agile Methodologies;
  - User Story;
  - User Stories;
  - Agile;
  - Documentation.

Depending on the database, a general search string was defined and adapted to each search engine. Below, we present the search string.

("Natural Language Processing" OR "Natural Language" OR NLP) AND ("User Story" OR "User Stories") AND (Agile OR Documentation).

3.3. Selection Strategy

Inclusion and exclusion criteria were established according to Petersen guidelines [49]. The inclusion criteria were defined as follows.

- Primary studies published in journals and conference proceedings
- Studies in which the main topics were NLP, Agile methodologies, and agile documentation;
- Studies published in the last ten years to consider the recent research works in the field.

Now, the exclusion criteria were the following.

- Articles which not published in English;
- Secondary or tertiary studies.

Quality Assessment

An essential part of the selection strategy is the quality assurance of the papers. The quality assessment was made along with the full text review, see Figure 1 for details. We considered the results presented of each paper to decide whether an article has an acceptable quality level.

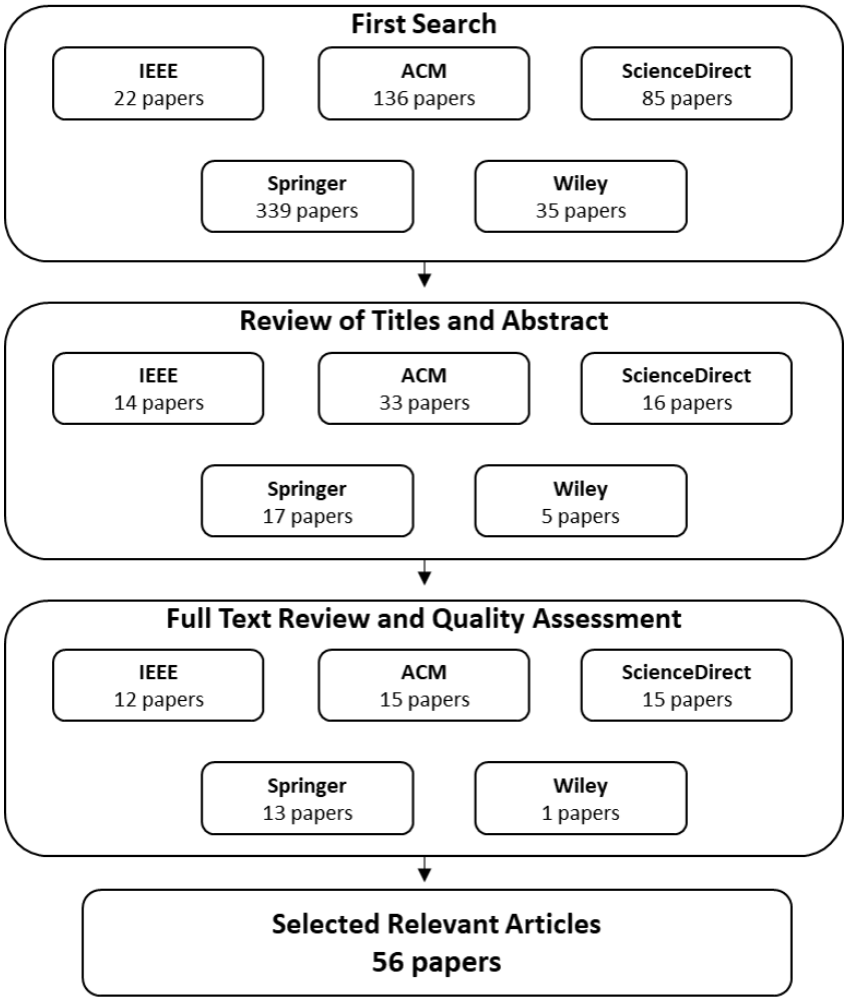

Figure 1. Paper selection process.

An article was considered of low quality when it had two or more of the following characteristics.

- It presents results that do not correspond to the information expressed in the paper.

- It presents results without any performance measurement.
- The paper does not present the information necessary to replicate the work.
- The paper results are based on assumptions or decisions that could be biased.

3.4. Conducting the Review

We planned this review between July and September of 2021; we executed the plan in two search periods. The first began in October of the same year, and the second in May of 2022. The selection process was carried out in partnership with a second researcher and the author to ensure the veracity and consistency of the process. The review process results were similar, but with low variations in the number of papers found due to the time process. We carried out the original search in October 2021; the pair review was carried out in November 2021, resulting in two more articles found on average in each database. The original and pair search was carried out for the second search in May 2022 with the same results.

Study Selection

Figure 1 shows the article selection process. We conducted the first search process using the search string and databases defined in Section 3.2. We found 617 articles in this first search; then, we selected 165 papers in the title review; as a result of the abstract review, we selected 85 papers; and finally, we accepted 56 papers when we reviewed the full-text of the papers, which includes the quality assessment process. For example, we discarded some papers because they did not present enough data to replicate their work and did not present their results. Other papers were discarded because they did not present any performance measurement, and the information presented was confusing or did not correspond with their results. We applied the inclusion and exclusion criteria among all the studies' selection processes.

4. Results

In general, we analyzed the full texts of 56 articles. We present the results below organized by research questions.

4.1. Summary of Studies

Below, we present a summary of the accepted papers to explain the characteristics of the complete set of works that we analyzed to answer the research questions.

A total of 56 relevant articles were identified from the review, of which 17 (30%) are journal publications, while 39 (70%) are conference publications. The relevant studies were distributed in different databases, indicating no predominant database where authors prefer to publish on the subject. Around half of the articles analyzed are preliminary studies (27 articles). The other 25 articles present at least one proof of concept of the expressed idea, while 26 presented at least laboratory experiments with their datasets or were based on real industry open source projects. None were applied in a real-world environment or tested in a company environment.

Figure 2 shows the distribution of the relevant articles from 2010 to 2022. It can be noted that the years with the most relevant articles are 2020 and 2019. In addition, a clear upward trend can be observed until 2020, before falling abruptly in 2021. However, this behavior could change in 2022, when almost the same papers are published. There is also a more significant number of articles published in conferences compared to those published in journals.

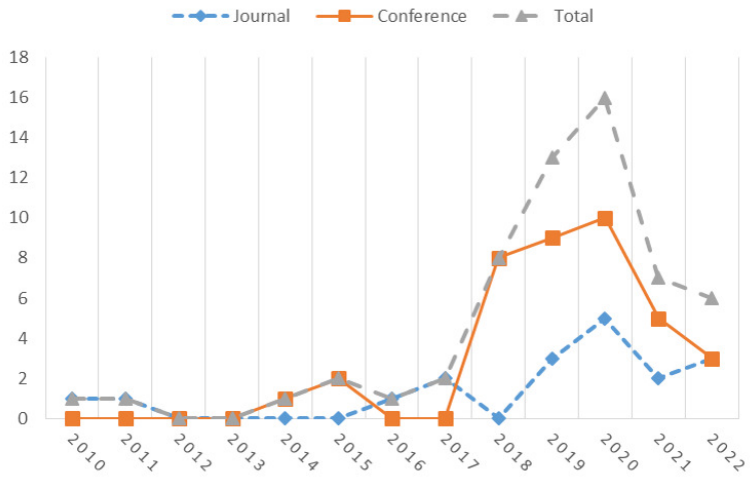

Figure 2. Publications per year at journals, conferences, and total.

4.2. (RQ1) What Approaches Have Been Made to Improve the Agile Documentation Process Using NLP?

During the analysis of complete texts, we found that a few works (three papers) focused on working on a specific agile methodology: two focused on Scrum and one on Extreme Programming. On the contrary, it is preferred to focus on common problems without distinguishing which method is performed when presented. The use of user stories as a fundamental artifact was found in 27 papers; other artifacts used in the selected papers are use case diagrams, use case scenarios, goal models, and class diagrams. Additionally, the findings suggest that it is preferred to focus on documentation artifacts used in various methodologies.

We identified four main categories in which NLP is applied in the agile documentation process:

- **Improve Requirements Engineering**: This category is about improving all the requirements engineering that includes recollection, analysis, validation, and management of requirements. Some works are talking about being a tool for the recollection of requirements **[P19, P24, P43]**, these approaches focus on making more accessible the recollection process. Some approaches in this sense intend to improve the recollection using NLP tools to analyze the clients' requirements on audio or text to produce a clear requirement or artifact with the information. Other approaches focus on enhancing the quality of the requirements on the artifacts **[P5, P6, P7, P47]**; some papers did this by verifying the quality of user stories and detecting redaction problems like redundancy on other artifacts.
- **Artifacts Transformation**: This category is about transforming an artifact, such as a user story, into another artifact corresponding with the same information, such as a use case scenario. The approaches center their efforts on specific artifact conversion, taking the original as the basis of the process. The transformations that we found in the relevant papers include the following: user stories to use case diagrams **[P15]**, user stories to use case scenario **[P10, P33]**, user stories to conceptual models **[P8, P28, P54]**, use cases to class model **[P2]**, user stories to BPM processes **[P32]**, user stories to goal model **[P30]**, user stories to class diagram **[P40]**, and user stories to behavioral UML models **[P50]**. The methods used to transform artifacts are implemented using distinct NLP techniques.
- **Agile Software Development Process**: The papers grouped in this category intend to improve one or many of the elements of the agile development process. The generation of artifacts like estimation of time and cost **[P13, P51]**, acceptance tests **[P1]**; and code generation of unit tests **[P44]**. This category includes works focused on project management **[P22]**, prototype generation **[P23]** and product release **[P17]**.

- **Team Communication**: This category focuses on improving team members' communication. Some approaches focus on processing information with artificial intelligence (AI) to improve the project information and share it with the team **[P31, P41]**. Others focus on designing a domain language to facilitate communication among the team members **[P39, P51]**.

Table 1 shows the classification results by categories of the applications shown in the relevant papers. The most common category found was improving requirements engineering with 24 papers. Figure 3 shows the distribution of the frequency of papers found on each category by year.

Table 1. NLP application categories.

Category	Freq.	Studies
Improve Requirements Engineering	24	[P3], [P4], [P5], [P6], [P7], [P11], [P16], [P18], [P19], [P20], [P24], [P25], [P27], [P29], [P36], [P37], [P43], [P46], [P47], [P48], [P49], [P52], [P55], [P56]
Transformation of Artifacts	11	[P2], [P8], [P10], [P15], [P28], [P30], [P32], [P33], [P40], [P50], [P54]
Agile Software Development Process	15	[P1], [P9], [P12], [P13], [P17], [P21], [P22], [P23], [P26], [P34], [P38], [P42], [P44], [P45], [P53]
Team Communication	6	[P14], [P31], [P35], [P39], [P41], [P51]

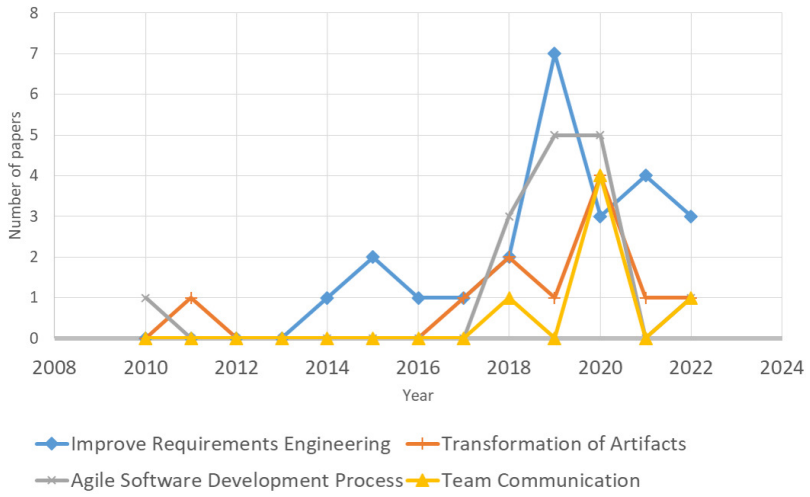

Figure 3. Results of categories by year.

4.3. (RQ2) What NLP Techniques and Toolkits Have Been Used to Improve Any Agile Methodology?

To answer question RQ2 in more detail, we divided the findings into two categories: NLP Techniques and NLP Tools.

4.3.1. NLP Techniques

The most used NLP techniques were classified as shown in Table 2. Note that each article can present more than one technique and could be counted in many categories. Part of Speech Tagging (POS) is the most used technique with 23 papers. After that, eleven papers used the Tokenizer technique. Additionally, Parsing techniques are used in ten papers. Six papers use Word Analysis like Bag-of-Words, Keyword extraction, and Word embedding. The rest of the techniques are Semantic Analysis (5 papers), Correlation and Dependencies analysis (6 papers), Sequence analysis (4 papers), Clustering (3 papers), Lemmatization (4 papers), and Others (9 papers). Finally, the Not Defined category includes ten articles that did not specify the applied technique or articles that proposed some techniques but they were not implemented. In Figure 4, the frequency distribution with which the different NLP techniques are used through the years can be observed. It can be seen that POS Tagging is the technique that he has mastered for the last three years.

Table 2. NLP techniques applied in studies.

Category	NLP Techniques	Freq.	Studies
Processing Techniques	POS Tagging	23	[P2], [P3], [P5], [P7], [P10], [P11], [P15], [P18], [P19], [P27], [P28], [P29], [P31], [P33], [P37], [P39], [P40], [P42], [P44], [P46], [P47], [P50], [P52]
	Tokenizer	13	[P2], [P5], [P6], [P11], [P14], [P19], [P27], [P39], [P42], [P44], [P46], [P50], [P52]
	Parsing	11	[P2], [P5], [P6], [P7], [P10], [P11], [P13], [P14], [P30], [P34], [P50]
	Lemmatization	4	[P7], [P19], [P27], [P50]
Text Representation	Word2Vec	7	[P21], [P36], [P37], [P42], [P44], [P45], [P53]
Extraction techniques	Semantic Analysis	5	[P16], [P25], [P41], [P47]
	Correlation and Dependencies Analysis	6	[P10], [P27], [P32], [P33], [P50], [P52]
	Sequence Analysis	4	[P4], [P20], [P24], [P55]
Grouping techniques	Clustering	3	[P4], [P24], [P25]
Uncategorized	Neural Networks	3	[P22], [P36], [P37]
	Other	9	[P7], [P14], [P17], [P35], [P48], [P49], [P51], [P54], [P56]
	Not Defined	10	[P1], [P8], [P9], [P12], [P23], [P26], [P35] , [P38], [P41], [P43]

Figure 4. NLP techniques frequency organized by NLP technique and year.

4.3.2. NLP Tools

The most used tools presented in the selected articles are Stanford CoreNLP, with five papers, followed by Natural Language Toolkit (NLTK), with three papers. SpaCy was used in two papers, and Stanford POS Tagger with one paper. Additionally, some papers show their own tools derived from existing ones or completely new tools such as Cortical.io, IdeaSy, and Gherking20AS. Although some new tools were found, there were authors' approaches and need more tests to be considered and published as a toolkit. Figure 5 shows the frequency of the toolkits found in the reviewed papers. A brief description of these main tools is presented below.

Most of the tools have the POS Tagging function, which is the most used in the selected articles, as seen in Figure 3. The usefulness of this technique goes hand in hand with the chunking technique (see Section 2 for background concepts). Through the fragmentation of a phrase, it is possible to obtain smaller elements that can be labeled to identify to which part of a certain sentence it belongs. Table 3 shows the principal tools found in the selected studies.

Table 3. NLP tools.

NLP Tool	Features	Freq.	Studies
Stanford CoreNLP	Tokenization, Part-Of-Speech (POS) Tagging, Lemmatization	5	[P6], [P9], [P11], [P40], [P42]
Natural Language Toolkit (NLTK)	Tokenization, Part-Of-Speech (POS) Tagging, Dependency Parsing, Lemmatization	3	[P5], [P6], [P35]
SpaCy NLP	Tokenization, Part-Of-Speech (POS) Tagging, Dependency Parsing, Lemmatization, Similarity Analysis	3	[P10], [P18], [P52]
Stanford POS Tagger	Part-Of-Speech (POS) Tagging	1	[P22]

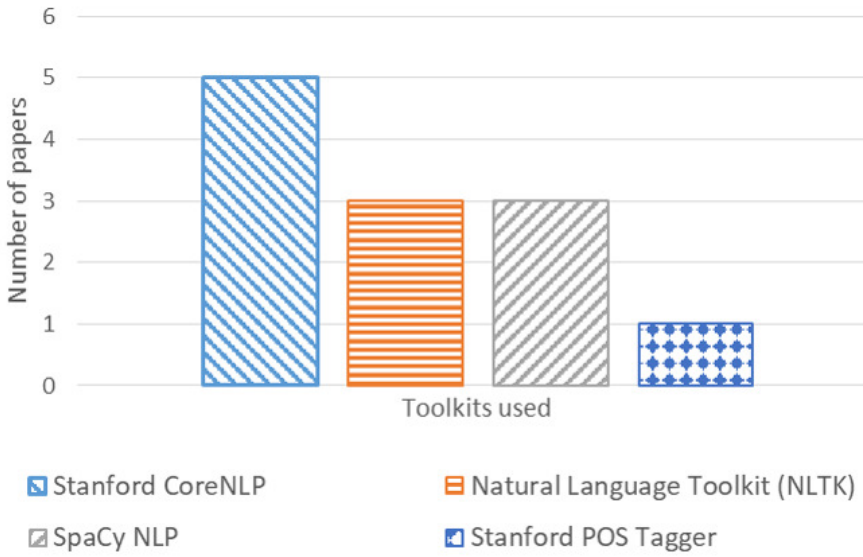

Figure 5. Used toolkit frequency.

- **Stanford CoreNLP** [50]**:** This is a complete tool to perform different operations such as tokenization, POS Tagging, and Lemmatization. It is a toolkit written in Java that can be used freely, having data dictionaries available for various languages such as English, Chinese, Spanish, French, German, Italian, and Arabic.
- **Natural Language Toolkit (NLTK)** [51]**:** NLTK is a platform written in Python that allows it to work easily and quickly with language processing. It includes various elements and libraries that allow the user to carry out various operations. This tool is free to use and has some fairly complete tutorials that allow starting from the use of language to perform complex text analysis operations.
- **SpaCy** [52]**:** This toolkit features a variety of possible operations and claims to be designed for real-world jobs. It can be used in various languages as APIs and has some precompiled features. It also seeks to have a speed of operations at the current state-of-the-art level. In the same way as the previous ones, this is a free-to-use toolkit.
- **Stanford POS Tagger** [53]**:** This software is a Java implementation of the log-linear part-of-speech taggers described in [54]. Current downloads contain three trained tagger models for English, two for Chinese and Arabic, and one for French, German, and Spanish. The tagger can be retrained in any language, given POS-annotated training text for the language.

Figure 6 shows the available NLP techniques in each toolkit found. SpaCy NLP is the tool which includes more techniques, followed by NLTK and Stanford CoreNLP. Stanford POS Tagger is a specialized toolkit focused on POS Tagging and tokenization.

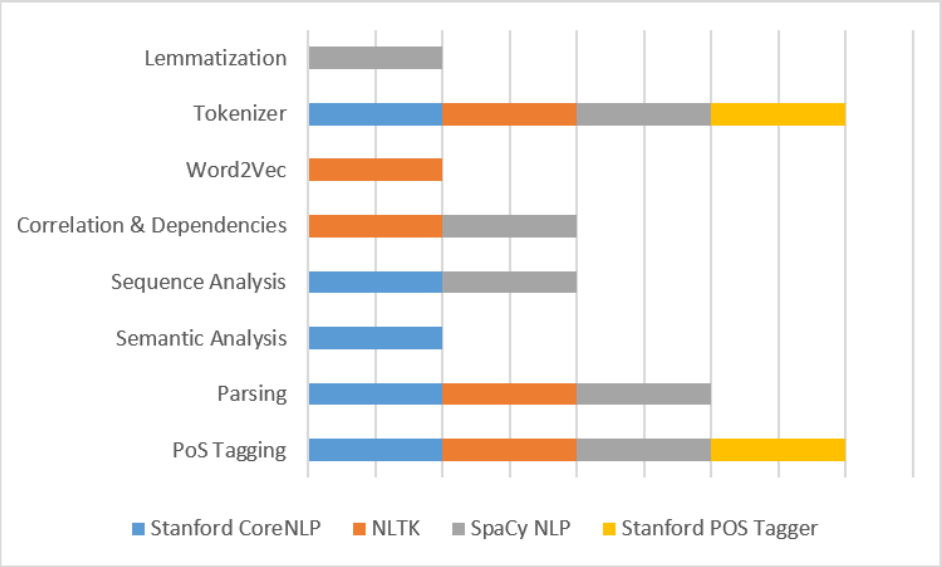

Figure 6. NLP techniques available by the toolkits.

4.4. (RQ3) How Is the Performance Measurement of the NLP Techniques When Applied to Support the Documentation Process of an Agile Project?

With the analysis of the studies, we found different approaches in the use of diverse NLP techniques. Depending on the approach in which each technique is explored, the performance of the technique varies. Due to the mentioned above, a fair comparison is difficult since the techniques are not performing the same activity. Still, some methods are used to assess performance. Most of the studies report evaluation methods based on study cases or using prototypes to demonstrate the feasibility of their proposal. Some studies show at least a theoretical proof of concept to validate their proposals. Some other studies use precision, recall, and F-measure measures as indicators. The precision is about "What proportion of identifications was correct?" and it is calculated as it is shown in Equation (1). Recall refers to "What proportion of actual positives was identified correctly?" and it is calculated as it is shown in Equation (2). Finally, F-measure unifies the precision and the recall calculating as it is shown in Equation (3).

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{1}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{2}$$

$$F - Measure = 2x\frac{Precision \times Recall}{Precision + Recall} \tag{3}$$

where $TruePositive$ = the correct labeled instances $FalsePositive$ = incorrectly labeled instances, $FalseNegative$ = the missed-out instances by the system.

Mainly, performance evaluations on the selected papers were made by comparing the results obtained by humans with those obtained by a system. Usually, a group of volunteers is used as annotators who can be students or software developers, depending on the authors.

From the total 56 articles, only 12 use Precision/Recall/F1 as methods to validate the proposal's performance.

5. Discussion

This work shows the use of NLP in the agile documentation process. The capacity of NLP for data processing, and the characteristic of requirements written in natural language to be understandable by all the project team stakeholders, make it possible to use NLP to analyze and process requirements artifacts. This process produces valuable information

obtained from the artifacts to avoid human errors and making easier the analysis of the extracted information. Some related studies address the analysis of user stories using NLP [46], but they are not focused on improving the documentation process of an agile project. Other studies are focused on the general use of intelligent tools in agile methodologies [47] without emphasizing the possible use of NLP to bring to that tools the capability of extracting meaningful information from artifacts to improve performance. On the other hand, in [48], the importance of documentation in natural language in agile methodologies is discussed, and they concluded that using natural language in documentation is widely used by organizations. However, they did not consider or analyze how NLP tools could take advantage of it to extract relevant information from projects' documentation.

Below, we present discussion corresponding to each research questions divided by subsections. Section 5.1 refers to RQ1, Section 5.2 refers to RQ2, and Section 5.3 refers to RQ3.

5.1. Approaches to Improve the Agile Documentation Process Using NLP Techniques

The selected articles are focused on improving the documentation process; most of them prefer to focus on processing requirements artifacts or requirements representations without distinguishing in which agile methodology (see Section 4.2). This situation allows us to adapt the tools to any methodology by not mixing something specific of some methodology in the tool. For example, user stories were the most common artifacts in the works, which can be applied in various agile methodologies. On the other hand, in some cases, better performance could be achieved by taking into account the context of an artifact within the internal process of a specific methodology. Other artifacts related to a particular methodology could also enhance the performance of the developed tool. For example, if the Scrum context is considered, we could consider the sprints and the backlog information [22] to enhance the quality of the data used to generate or transform artifacts. More studies may be needed to explore the impact of context with NLP. We found several ways in which NLP is used within the agile project documentation process. Some of them focus on processing user stories [46], while others choose to use other artifacts [48] such as use cases, use case scenarios, goal models, and class diagrams.

5.2. NLP Techniques and Toolkits Used to Improve the Documentation Process of Agile Development

During the analysis of the use of NLP techniques in the selected articles, we found that POS Tagging was the most commonly used technique. This finding indicates that tagging elements in a text is an important task when applying NLP to agile requirements. POS Tagging helps to identify each part of a text and allows applying other types of semantic analysis techniques, dependencies, and correlations, among others. NLP techniques, such as Tokenizer (the second most used technique) are commonly used as a preprocessing step; this indicates that using a preprocessing step might deliver better final results.

We also found that using NLP toolkits is entirely accepted within the studied area. It allows faster progress with research ideas to turn them into tools by making the development process easier and providing practical tools when working with NLP. These toolkits eliminate the need to recreate from scratch a process that effectively uses the desired NLP technique and provides a field-proven and accepted technique. Thus, researchers can move forward more confidently, knowing that what they use to develop a tool has been successfully tested multiple times. Although toolkits are widely used, in some cases, it might be better to develop the necessary NLP technique from scratch to combine it efficiently with other deep learning methods such as convolutional networks [55]. Some studies focus on comparing different toolkits applied to the same dataset of words [33]. These studies allow researchers to identify the appropriate toolkit depending on the task they want to address.

5.3. Performance Measurement of NLP Techniques Applied to Support the Documentation Process of Agile Development

Regarding the answer to question RQ3, we noticed that there is no standardized method to evaluate the performance of an NLP tool in agile applications. We found that each reviewed article presented an evaluation at their convenience. Either they are qualitative, based on the perception of those who used the tool, or quantitatively, using some method such as precision, recall, or F-measure indexes. The lack of a standardized method causes many of the results to be not comparable. These comparison difficulties can be a problem in determining the best-performing tool or evaluating a new tool with existing ones in the literature. Suppose some authors establish and require a standardized method for performance evaluation. In that case, the new approaches to NLP applications could be easily compared with the other existing approaches to show their characteristics and the best performance. In ref. [34] are described some evaluation methods recommended for NLP systems; the authors established that, although humans do the evaluation, it is essential to condense it into a standard measurement and not leave it only in data given by human evaluators. This lack of standardized methods for performance evaluation may indicate that the research area is still in progress and is on its way to obtaining a better maturity level in future years.

5.4. Threats to Validity

This study has some threats to validity that could affect the results or conclusions. We present them in the following four categories established in [56] and recommendations explained in [57].

5.4.1. Internal Validity

This category includes threats that may have affected the results and need to be properly considered.

- **Research Method:** One threat to validity is the restricted access to some databases that were excluded from the selection. To overcome this threat, we collected papers from databases used frequently in software engineering research [46]. Another threat could be improper research methods, which we overcome using the guidelines proposed by Petersen et al. [32]. The different search methods of each database were also a threat mitigated with the design of a base search string adapted to each database.
- **Research Questions:** The set of defined research questions for this study might not cover all the possible interest areas in the field, but we defined them in a way that covers the main fundamental points of interest.

5.4.2. Conclusion Validity

This category includes the threats that may affect drawing inaccurate conclusions from observations.

- **Selected articles:** The article selection might be improper because of the researcher's bias. This threat was mitigated by doing a pair search with the help of another researcher. The results were similar not only on the first search (with the search string), but also in the reviews of the title, abstract and full text.

5.4.3. Construction Validity

This category includes threats to the relationship between theory and observation.

- **Analysis:** Validity of construction is the threat that experimenters can influence the results of a study based on what they expect from their experience in the field. This threat is not present in our work because we only report data founded on the selected papers without changing them.

5.4.4. External Validity

This category includes threats that affect the generalizability of the results.

- **Generalization:** As Wohlin et al. said in [57], the generalization of the results to the search field is very important, this could be a threat to our study depending on which studies were selected. To mitigate this threat, we defined all the search processes in a general form that could bring us a variety of studies; this is, using distinct databases with the same search string and analyzing each result by abstract, title, and full text.

6. Conclusions and Future Work

This work presented a Systematic Mapping Review to show the existing research on applying NLP techniques and toolkits in the documentation process of agile development methodologies. Many NLP techniques have been used to improve agile development processes; Tokenizer, POS Tagging, Parsing, Word, and Semantic Analysis are the most used in the founded articles. In addition to specific techniques, toolkits that facilitate using these techniques have also been found, and new ones are currently being published. These findings show that the field is under continuous study, which indicates that there are still points of interest and areas of opportunity to research.

Although we do not find related works on the same subject, some similar reviews address documentation and NLP. Next, we present two examples of related papers with similar subjects. In [46], they review the research of NLP with user stories; they report how NLP has addressed user stories to write them, analyze them and check their quality. On the other hand, in [47], the use of intelligent software in the documentation process of agile projects is addressed by obtaining the most frequent areas of the software development process like software management, software design, testing, and quality assurance, in which intelligent software was used. The main differences with these works are that our work is not limited to a single artifact like user stories, even if they are widely used, and also is not limited to a particular way to apply NLP. Moreover, our work focuses on using NLP rather than other intelligence techniques.

Below, we present the conclusions and future work of this research.

6.1. Conclusions

Regarding NLP technique applications to improve documentation process focusing on artifact conversion, something that we have not found and that could be interesting is a complete tool that can process an artifact as input to obtain a user story as output. Moreover, at the same time, to be able to provide the developer with a code template serving as a base for developing the code fragment related to that user story. The closest examples to this approach were works that convert user stories into use case scenarios and use cases into goal models.

There are two main ways when working with NLP techniques: developing it from scratch or using a toolkit. Using personalized NLP techniques is an important option when wanting to work in the area. It must be considered that when a technique is developed from scratch, some essential points can be customized, not only the parametrization. For example, a better method can be established to combine the NLP technique with other deep learning techniques, such as CNN or LSTM neural networks. The preprocessing, processing, and post-processing methods can form a complete tool that, when developed from scratch, can be self-sufficient and a complete tool for a specific purpose. The problem is the time required to sufficiently test the custom tool in the desired language. This problem is avoided in toolkits because they already have support for various languages, and if required, it is easier to change the desired language. The number of toolkits available and their characteristics allow us to start working with NLP easily and quickly. Each toolkit has a variety of languages that can be processed with it. This variety of toolkits allows various valuable systems to be developed in languages like Spanish, English, Portuguese, French, and many others.

The need for standardized measuring methods shows that the need to establish a standardized method for evaluating performance must be addressed in future work to improve the comparison of approaches; moreover, to make more visible to researchers the strengths and weakness of each approach to applying NLP to agile documentation.

Making this visible allows researchers to proceed more confidently by making appropriate comparisons with existing work in the literature without the need to seek to convert others' data into a standard comparable measure for all. Even though there are previously established methods in the area of NLP for performance evaluation, only 12 used said methods in the relevant papers. The other works focus on an evaluation made by people and presented in tables without performing the calculations of measures such as Precision, Recall, and F-Measure.

6.2. Future Work

Our results show a trend in using NLP in agile methodologies in the previous years. This trend suggests that the field will continue to grow and is worth contributing to NLP in agile methods. The papers have focused more on working in the category of requirements engineering improvement, which is a fundamental point of the agile development process. These findings and the need for more work focused on mixing several of these categories inspire searching for ways to combine at least two categories to create more complete tools that are helpful in agile development. In future work, we will combine requirements improvement, agile software process, and artifacts transformation to develop helpful tools for developers to create, organize, and update documentation artifacts during the agile project process.

Author Contributions: Conceptualization, methodology, formal analysis, investigation, writing—original draft, writing—review and editing, visualization, M.A.Q.; conceptualization, methodology, writing—original draft, writing—review and editing, supervision, project administration, G.B.S.; validation, data curation, writing—original draft, writing—review and editing, R.R.P.; conceptualization, methodology, software, investigation, data curation, writing—original draft, writing—review and editing, S.G.-L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the "Programa de Fortalecimiento a la Investigación 2022" (project numbers: PROFAPI 2022_0080 and PROFAPI 2022_0035) of the Instituto Tecnológico de Sonora.

Data Availability Statement: Not applicable.

Acknowledgments: This work has been supported by the CONACYT scholarship 787750.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

NLP	Natural Language Processing
AI	Artificial Intelligence
UML	Unified Modeling Language

Appendix A. Included Studies

The following list is compounded of the relevant selected papers for the mapping review.
[P1] El-Attar, M., & Miller, J. (2010). Developing comprehensive acceptance tests from use cases and robustness diagrams. Requirements engineering, 15(3), 285–306.
[P2] Elbendak, M., Vickers, P., & Rossiter, N. (2011). Parsed use case descriptions as a basis for object-oriented class model generation. Journal of Systems and Software, 84(7), 1209–1223.
[P3] Fuckner, M., Barthès, J. P., & Scalabrin, E. E. (2014, May). Using a personal assistant for exploiting service interfaces. In Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD) (pp. 89–94). IEEE.
[P4] Barbosa, R., Januario, D., Silva, A. E., Moraes, R., & Martins, P. (2015, June). An approach to clustering and sequencing of textual requirements. In 2015 IEEE International Conference on Dependable Systems and Networks Workshops (pp. 39–44). IEEE.

[P5] Lucassen, G., Dalpiaz, F., Van Der Werf, J. M. E., & Brinkkemper, S. (2015, August). Forging high-quality user stories: towards a discipline for agile requirements. In 2015 IEEE 23rd international requirements engineering conference (RE) (pp. 126–135). IEEE.
[P6] Lucassen, G., Dalpiaz, F., van der Werf, J. M. E., & Brinkkemper, S. (2016). Improving agile requirements: the quality user story framework and tool. Requirements engineering, 21(3), 383–403.
[P7] Femmer, H., Fernández, D. M., Wagner, S., & Eder, S. (2017). Rapid quality assurance with requirements smells. Journal of Systems and Software, 123, 190–213.
[P8] Lucassen, G., Robeer, M., Dalpiaz, F., Van Der Werf, J. M. E., & Brinkkemper, S. (2017). Extracting conceptual models from user stories with Visual Narrator. Requirements Engineering, 22(3), 339–358.
[P9] Athiththan, K., Rovinsan, S., Sathveegan, S., Gunasekaran, N., Gunawardena, K. S., & Kasthurirathna, D. (2018, December). An ontology-based approach to automate the software development process. In 2018 IEEE International Conference on Information and Automation for Sustainability (ICIAfS) (pp. 1–6). IEEE.
[P10] Gilson, F., & Irwin, C. (2018, November). From user stories to use case scenarios towards a generative approach. In 2018 25th Australasian Software Engineering Conference (ASWEC) (pp. 61–65). IEEE.
[P11] Sachdeva, S., Arya, A., Paygude, P., Chaudhary, S., & Idate, S. (2018, February). Prioritizing User Requirements for Agile Software Development. In 2018 International Conference On Advances in Communication and Computing Technology (ICACCT) (pp. 495–498). IEEE.
[P12] Dimanidis, A., Chatzidimitriou, K. C., & Symeonidis, A. L. (2018, April). A Natural Language Driven Approach for Automated Web API Development: Gherkin2OAS. In Companion Proceedings of the The Web Conference 2018 (pp. 1869–1874).
[P13] Ecar, M., Kepler, F. N., & da Silva, J. P. S. (2018, June). AutoCosmic: COSMIC automated estimation and management tool. In Proceedings of the XIV Brazilian Symposium on Information Systems (pp. 1–8).
[P14] Sermet, Y., & Demir, I. (2018). An intelligent system on knowledge generation and communication about flooding. Environmental modelling & software, 108, 51–60.
[P15] Elallaoui, M., Nafil, K., & Touahni, R. (2018). Automatic transformation of user stories into UML use case diagrams using NLP techniques. Procedia computer science, 130, 42–49.
[P16] Dalpiaz, F., Van der Schalk, I., & Lucassen, G. (2018, March). Pinpointing ambiguity and incompleteness in requirements engineering via information visualization and NLP. In International Working Conference on Requirements Engineering: Foundation for Software Quality (pp. 119–135). Springer, Cham.
[P17] Sharma, S., & Kumar, D. (2019, February). Agile release planning using natural language processing algorithm. In 2019 Amity International Conference on Artificial Intelligence (AICAI) (pp. 934–938). IEEE.
[P18] Gilson, F., Galster, M., & Georis, F. (2019, March). Extracting quality attributes from user stories for early architecture decision making. In 2019 IEEE International Conference on Software Architecture Companion (ICSA-C) (pp. 129–136). IEEE.
[P19] Raharjana, I. K., Siahaan, D., & Fatichah, C. (2019, July). User story extraction from online news for software requirements elicitation: A conceptual model. In 2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE) (pp. 342–347). IEEE.
[P20] Madala, K. (2019, May). An artificial intelligence-based model-driven approach for exposing off-nominal behaviors. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) (pp. 214–217). IEEE.
[P21] De Bortoli Fávero, E. M., Casanova, D., & Pimentel, A. R. (2019, September). Embse: A word embeddings model oriented towards software engineering domain. In Proceedings of the XXXIII Brazilian Symposium on Software Engineering (pp. 172–180).
[P22] Dam, H. K., Tran, T., Grundy, J., Ghose, A., & Kamei, Y. (2019, May). Towards effective AI-powered agile project management. In 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER) (pp. 41–44). IEEE.

[P23] Pinto, T. D., Gonçalves, W. I., & Costa, P. V. (2019, October). User interface prototype generation from agile requirements specifications written in concordia. In Proceedings of the 25th Brazillian Symposium on Multimedia and the Web (pp. 61–64).
[P24] Reddivari, S., Bhowmik, T., & Hollis, C. (2019). Automated support to capture verbal just-in-time requirements via audio mining and cluster-based visualization. Journal of Industrial Information Integration, 14, 41–49.
[P25] Dalpiaz, F., Van Der Schalk, I., Brinkkemper, S., Aydemir, F. B., & Lucassen, G. (2019). Detecting terminological ambiguity in user stories: Tool and experimentation. Information and Software Technology, 110, 3–16.
[P26] Hotomski, S., & Glinz, M. (2019). GuideGen: An approach for keeping requirements and acceptance tests aligned via automatically generated guidance. Information and Software Technology, 110, 17–38.
[P27] Nazaruka, E. (2019, May). An overview of ways of discovering cause-effect relations in text by using natural language processing. In International Conference on Evaluation of Novel Approaches to Software Engineering (pp. 22–38). Springer, Cham.
[P28] Gupta, A., Poels, G., & Bera, P. (2019, November). Creation of Multiple Conceptual Models from User Stories–A Natural Language Processing Approach. In International Conference on Conceptual Modeling (pp. 47–57). Springer, Cham.
[P29] Müter, L., Deoskar, T., Mathijssen, M., Brinkkemper, S., & Dalpiaz, F. (2019, March). Refinement of user stories into backlog items: Linguistic structure and action verbs. In International Working Conference on Requirements Engineering: Foundation for Software Quality (pp. 109–116). Springer, Cham.
[P30] Güneş, T., & Aydemir, F. B. (2020, August). Automated goal model extraction from user stories using NLP. In 2020 IEEE 28th International Requirements Engineering Conference (RE) (pp. 382–387). IEEE.
[P31] T. Wolf, C., & L. Blomberg, J. (2020, June). Ambitions and Ambivalences in Participatory Design: Lessons from a Smart Workplace Project. In Proceedings of the 16th Participatory Design Conference 2020-Participation (s) Otherwise-Volume 1 (pp. 193–202).
[P32] Baïna, K., El Hamlaoui, M., & Kabbaj, H. (2020, September). Business Process Modelling Augmented: Model Driven transformation of User Stories to Processes. In Proceedings of the 13th International Conference on Intelligent Systems: Theories and Applications (pp. 1–6).
[P33] Gilson, F., Galster, M., & Georis, F. (2020, June). Generating use case scenarios from user stories. In Proceedings of the International Conference on Software and System Processes (pp. 31–40).
[P34] Moran, K., Palacio, D. N., Bernal-Cárdenas, C., McCrystal, D., Poshyvanyk, D., Shenefiel, C., & Johnson, J. (2020, June). Improving the effectiveness of traceability link recovery using hierarchical bayesian networks. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (pp. 873–885).
[P35] Pinto, R., Silva, L., & Valentim, R. (2020, March). Managing sessions of creative requirements elicitation and assessment. In Proceedings of the 35th Annual ACM Symposium on Applied Computing (pp. 1355–1362).
[P36] Wang, Y., Zheng, L., & Li, N. (2020, January). ROM: A Requirement Opinions Mining Method Preliminary Try Based on Software Review Data. In Proceedings of the 2020 4th International Conference on Management Engineering, Software Engineering and Service Sciences (pp. 26–33).
[P37] Ochodek, M., Kopczyńska, S., & Staron, M. (2020). Deep learning model for end-to-end approximation of COSMIC functional size based on use-case names. Information and Software Technology, 123, 106310.
[P38] Zampetti, F., Di Sorbo, A., Visaggio, C. A., Canfora, G., & Di Penta, M. (2020). Demystifying the adoption of behavior-driven development in open source projects. Information and Software Technology, 123, 106311.
[P39] Urbieta, M., Antonelli, L., Rossi, G., & do Prado Leite, J. C. S. (2020). The impact of using a domain language for an agile requirements management. Information and Software Technology, 127, 106375.

[P40] Nasiri, S., Rhazali, Y., & Lahmer, M. (2021). Towards a generation of class diagram from user stories in agile methods. In Advancements in Model-Driven Architecture in Software Engineering (pp. 135–159). IGI Global. **[P41]** Schuh, G., Hicking, J., Stroh, M. F., & Benning, J. (2020). Using AI to Facilitate Technology Management–Designing an Automated Technology Radar. Procedia CIRP, 93, 419–424.
[P42] Villamizar, H., Kalinowski, M., Garcia, A., & Mendez, D. (2020). An efficient approach for reviewing security-related aspects in agile requirements specifications of web applications. Requirements Engineering, 25(4), 439–468.
[P43] Dalpiaz, F., & Sturm, A. (2020, March). Conceptualizing Requirements Using User Stories and Use Cases: A Controlled Experiment. In REFSQ (pp. 221–238).
[P44] Mulla, N., & Jayakumar, N. (2020). The potent combo of software testing and NLP. In ICDSMLA 2019 (pp. 1623-1632). Springer, Singapore.
[P45] Resketi, M. R., Motameni, H., Nematzadeh, H., & Akbari, E. (2020). Automatic summarising of user stories in order to be reused in future similar projects. IET Software, 14(6), 711–723.
[P46] Rahmi Dewi, M., Kharisma Raharjana, I., Siahaan, D., & Fatichah, C. (2021, February). Software Requirement-Related Information Extraction from Online News using Domain Specificity for Requirements Elicitation: How the system analyst can get software requirements without constrained by time and stakeholder availability. In 2021 10th International Conference on Software and Computer Applications (pp. 81–87).
[P47] Kifetew, F. M., Perini, A., Susi, A., Siena, A., Muñante, D., & Morales-Ramirez, I. (2021). Automating user-feedback driven requirements prioritization. Information and Software Technology, 138, 106635.
[P48] Dalpiaz, F., & Brinkkemper, S. (2021, September). Agile Requirements Engineering: from User Stories to Software Architectures. In 2021 IEEE 29th International Requirements Engineering Conference (RE) (pp. 504–505). IEEE.
[P49] Günes, T., Öz, C. A., & Aydemir, F. B. (2021, September). ArTu: A Tool for Generating Goal Models from User Stories. In 2021 IEEE 29th International Requirements Engineering Conference (RE) (pp. 436–437). IEEE.
[P50] A. Abdelnabi, E., M. Maatuk, A., & M. Abdelaziz, T. (2021, October). An Algorithmic Approach for Generating Behavioral UML Models Using Natural Language Processing. In The 7th International Conference on Engineering & MIS 2021 (pp. 1–6).
[P51] Wambsganss, T., Söllner, M., Koedinger, K. R., & Leimeister, J. M. (2022, April). Adaptive Empathy Learning Support in Peer Review Scenarios. In CHI Conference on Human Factors in Computing Systems (pp. 1–17).
[P52] Casillo, F., Deufemia, V., & Gravino, C. (2022). Detecting privacy requirements from User Stories with NLP transfer learning models. Information and Software Technology, 146, 106853.
[P53] Fávero, E. M. D. B., Casanova, D., & Pimentel, A. R. (2022). SE3M: A model for software effort estimation using pre-trained embedding models. Information and Software Technology, 147, 106886.
[P54] Bragilovski, M., Dalpiaz, F., & Sturm, A. (2022, March). Guided Derivation of Conceptual Models from User Stories: A Controlled Experiment. In International Working Conference on Requirements Engineering: Foundation for Software Quality (pp. 131–147). Springer, Cham.
[P55] Urbieta, M. (2022). Tracing User Stories and Source Code Using the Language Extended Lexicon. Information Systems and Technologies: WorldCIST 2022, Volume 2, 469, 413.
[P56] Liscio, E., van der Meer, M., Siebert, L. C., Jonker, C. M., & Murukannaiah, P. K. (2022). What values should an agent align with?. Autonomous agents and multi-agent systems, 36(1), 1–32.

References

1. Highsmith, J.; Cockburn, A. Agile software development: The business of innovation. *Computer* **2001**, *34*, 120–127. [CrossRef]
2. Beck, K.; Beedle, M.; Van Bennekum, A.; Cockburn, A.; Cunningham, W.; Fowler, M.; Grenning, J.; Highsmith, J.; Hunt, A.; Jeffries, R.; et al. Manifesto for agile software development twelve principles of agile software. *Zugriff* **2001**, *5*, 2020.
3. Ken, S.; Beedle, M. *Agile Software Development with Scrum*; Prentice Hall: Upper Saddle River, NJ, USA, 2002; Volume 1.
4. Collins-Cope, M.; Stephens, M.; Rosenberg, D. *Agile Development with the ICONIX Process: People, Process and Pragmatism*; Springer: Berlin/Heidelberg, Germany, 2005.
5. Ambler, S.W. *The Elements of UML(TM) 2.0 Style*; Cambridge University Press: Cambridge, UK, 2005.
6. Lindstrom, L.; Jeffries, R. Extreme programming and agile software development methodologies. In *IS Management Handbook*; Auerbach Publications: New York, NY, USA, 2003; pp. 531–550.
7. Cockburn, A. *Crystal Clear: A Human-Powered Methodology for Small Teams: A Human-Powered Methodology for Small Teams*; Pearson Education: Upper Saddle River, NJ, USA, 2004.
8. Tom, E.; Aurum, A.; Vidgen, R. An exploration of technical debt. *J. Syst. Softw.* **2013**, *86*, 1498–1516. [CrossRef]
9. Cunningham, W. The WyCash portfolio management system. *ACM SIGPLAN OOPS Messenger* **1992**, *4*, 29–30. [CrossRef]
10. Bosch, J. Software architecture: The next step. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3047, pp. 194–199. [CrossRef]
11. Clear, T. Documentation and agile methods: Striking a balance. *ACM SIGCSE Bull.* **2003**, *35*, 12–13. [CrossRef]
12. Codabux, Z.; Williams, B. Managing Technical Debt: An Industrial Case Study. In Proceedings of the 2013 4th International Workshop on Managing Technical Debt (MTD), San Francisco, CA, USA, 20–20 May 2013.
13. Casamayor, A.; Godoy, D.; Campo, M. Mining textual requirements to assist architectural software design: A state of the art review. *Artif. Intell. Rev.* **2012**, *38*, 173–191. [CrossRef]
14. Deng, L.; Yu, D. Deep learning: Methods and applications. *Found. Trends Signal Process.* **2014**, *7*, 197–387. [CrossRef]
15. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
16. Manning, C.D.; Surdeanu, M.; Bauer, J.; Finkel, J.R.; Bethard, S.; McClosky, D. The Stanford CoreNLP natural language processing toolkit. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Baltimore, MD, USA, 22–27 June 2014; pp. 55–60.
17. Loper, E.; Bird, S. Nltk: The natural language toolkit. *arXiv* **2002**, arXiv:cs/0205028.
18. Qi, P.; Zhang, Y.; Zhang, Y.; Bolton, J.; Manning, C.D. Stanza: A Python natural language processing toolkit for many human languages. *arXiv* **2020**, arXiv:2003.07082.
19. Wang, C.; Pastore, F.; Goknil, A.; Briand, L. Automatic generation of acceptance test cases from use case specifications: An nlp-based approach. *IEEE Trans. Softw. Eng.* **2020**, *48*, 585–616. [CrossRef]
20. Elallaoui, M.; Nafil, K.; Touahni, R. Automatic transformation of user stories into UML use case diagrams using NLP techniques. *Procedia Comput. Sci.* **2018**, *130*, 42–49. [CrossRef]
21. Wang, X.; Zhao, L.; Wang, Y.; Sun, J. The role of requirements engineering practices in agile development: An empirical study. In *Requirements Engineering*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 195–209.
22. Plank, B.; Sauer, T.; Schaefer, I. Supporting agile software development by natural language processing. In *International Workshop on Eternal Systems*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 91–102.
23. Lucassen, G.; Dalpiaz, F.; Van Der Werf, J.M.E.M.; Brinkkemper, S. Forging high-quality user stories: Towards a discipline for agile requirements. In Proceedings of the 2015 IEEE 23rd International Requirements Engineering Conference (RE), Ottawa, ON, Canada, 24–28 August 2015; pp. 126–135.
24. Cohn, M. *User Stories Applied: For Agile Software Development*; Addison-Wesley Professional: Boston, MA, USA, 2004.
25. Cohn, M. Advantages of User Stories for Requirements. InformIT Network. 2004. Available online: http://www.informit.com/articles (accessed on 6 September 2021).
26. Cao, L.; Ramesh, B. Agile requirements engineering practices: An empirical study. *IEEE Softw.* **2008**, *25*, 60–67. [CrossRef]
27. Paetsch, F.; Eberlein, A.; Maurer, F. Requirements engineering and agile software development. In Proceedings of the WET ICE 2003, Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Linz, Austria, 9–11 June 2003; pp. 308–313.
28. Batool, A.; Motla, Y.H.; Hamid, B.; Asghar, S.; Riaz, M.; Mukhtar, M.; Ahmed, M. Comparative study of traditional requirement engineering and agile requirement engineering. In Proceedings of the 2013 15th International Conference on Advanced Communications Technology (ICACT), PyeongChang, Republic of Korea, 27–30 January 2013; pp. 1006–1014.
29. Ambriola, V.; Gervasi, V. Processing natural language requirements. In Proceedings of the 12th IEEE International Conference Automated Software Engineering, Incline Village, NV, USA, 1–5 November 1997; pp. 36–45.
30. Jackson, M. Problems and requirements [software development]. In Proceedings of the 1995 IEEE International Symposium on Requirements Engineering (RE'95), York, UK, 27–29 March 1995; pp. 2–8.
31. Liddy, E.D. Natural language processing. In *Encyclopedia of Library and Information Science*, 2nd ed.; Marcel Decker, Inc.: New York, NY, USA, 2001.
32. Petersen, K.; Feldt, R.; Mujtaba, S.; Mattsson, M. Systematic mapping studies in software engineering. In Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12, Bari, Italy, 26–27 June 2008; pp. 1–10.

33. Pinto, A.; Gonçalo Oliveira, H.; Oliveira Alves, A. Comparing the performance of different NLP toolkits in formal and social media text. In Proceedings of the 5th Symposium on Languages, Applications and Technologies (SLATE'16), Maribor, Slovenia, 20–21 June 2016; Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik: Dagstuhl, Germany, 2016.
34. Nakache, D.; Metais, E.; Timsit, J.F. Evaluation and NLP. In Proceedings of the International Conference on Database and Expert Systems Applications, Valencia, Spain, 1–4 September 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 626–632.
35. Brill, E. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Comput. Linguist.* **1995**, *21*, 543–565.
36. Mitchell, D.C. Sentence parsing. In *Handbook of Psycholinguistics*; Elsevier: Amsterdam, The Netherlands, 1994; pp. 375–409.
37. Klein, D.; Manning, C.D. Accurate unlexicalized parsing. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, Sapporo Japan, 7–12 July 2003; pp. 423–430.
38. Kudo, T.; Matsumoto, Y. Chunking with support vector machines. In Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics, Pittsburgh, PA, USA, 2 June 2001.
39. Halácsy, P.; Trón, V. Benefits of deep NLP-based Lemmatization for Information Retrieval. In Proceedings of the CLEF (Working Notes), Alicante, Spain, 20–22 September 2006.
40. Tshitoyan, V.; Dagdelen, J.; Weston, L.; Dunn, A.; Rong, Z.; Kononova, O.; Persson, K.A.; Ceder, G.; Jain, A. Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature* **2019**, *571*, 95–98. [CrossRef] [PubMed]
41. Zhang, Y.; Jin, R.; Zhou, Z.H. Understanding bag-of-words model: A statistical framework. *Int. J. Mach. Learn. Cybern.* **2010**, *1*, 43–52. [CrossRef]
42. Maulud, D.H.; Zeebaree, S.R.; Jacksi, K.; Sadeeq, M.A.M.; Sharif, K.H. State of art for semantic analysis of natural language processing. *Qubahan Acad. J.* **2021**, *1*, 21–28. [CrossRef]
43. Choi, J.D.; Tetreault, J.; Stent, A. It depends: Dependency parser comparison using a web-based evaluation tool. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Beijing, China, 26–31 July 2015; Volume 1: Long Papers, pp. 387–396.
44. Lin, J.C.W.; Shao, Y.; Djenouri, Y.; Yun, U. ASRNN: A recurrent neural network with an attention model for sequence labeling. *Knowl.-Based Syst.* **2021**, *212*, 106548. [CrossRef]
45. Xu, J.; Wang, P.; Tian, G.; Xu, B.; Zhao, J.; Wang, F.; Hao, H. Short text clustering via convolutional neural networks. In Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing, Denver, CO, USA, 5 June 2015; pp. 62–69.
46. Raharjana, I.K.; Siahaan, D.; Fatichah, C. User Stories and Natural Language Processing: A Systematic Literature Review. *IEEE Access* **2021**, *9*, 53811–53826. [CrossRef]
47. Perkusich, M.; Chaves e Silva, L.; Costa, A.; Ramos, F.; Saraiva, R.; Freire, A.; Dilorenzo, E.; Dantas, E.; Santos, D.; Gorgônio, K.; et al. Intelligent software engineering in the context of agile software development: A systematic literature review. *Inf. Softw. Technol.* **2020**, *119*, 106241. [CrossRef]
48. Wagner, S.; Fernández, D.M.; Felderer, M.; Vetrò, A.; Kalinowski, M.; Wieringa, R.; Pfahl, D.; Conte, T.; Christiansson, M.T.; Greer, D.; et al. Status quo in requirements engineering: A theory and a global family of surveys. *ACM Trans. Softw. Eng. Methodol.* **2019**, *28*, 1–48. [CrossRef]
49. Petersen, K.; Vakkalanka, S.; Kuzniarz, L. Guidelines for conducting systematic mapping studies in software engineering: An update. *Inf. Softw. Technol.* **2015**, *64*, 1–18. [CrossRef]
50. Stanford CoreNLP. Available online: https://stanfordnlp.github.io/CoreNLP/ (accessed on 21 January 2022).
51. Natural Language Toolkit. Available online: https://www.nltk.org/ (accessed on 21 January 2022).
52. spaCy. Available online: https://spacy.io/ (accessed on 21 January 2022).
53. Stanford Log-Linear Part-of-Speech Tagger. Available online: https://nlp.stanford.edu/software/tagger.shtml (accessed on 21 January 2022).
54. Toutanvoa, K.; Manning, C.D. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, Hong Kong, China, 7–8 October 2000; pp. 63–70.
55. Park, S.; Song, J.H.; Kim, Y. A neural language model for multi-dimensional textual data based on CNN-LSTM network. In Proceedings of the 2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Busan, Republic of Korea, 27–29 June 2018; pp. 212–217.
56. Cook, T.D.; Campbell, D.T.; Day, A. *Quasi-Experimentation: Design & Analysis Issues for Field Settings*; Houghton Mifflin Boston: Boston, MA, USA, 1979; Volume 351.
57. Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M.C.; Regnell, B.; Wesslén, A. *Experimentation in Software Engineering*; Springer Science & Business Media: Norwell, MA, USA, 2012.