ARTIFICIAL INTELLIGENCE FOR COST REDUCTION IN SPACE ENGINEERING: EXTRACTION OF SOFTWARE-REQUIREMENTS FROM NATURAL LANGUAGE REQUIREMENT DOCUMENTS

M. Ehresmann*, J. Beyer*, S. Fasoulas*
* University of Stuttgart, Institute of Space Systems, Pfaffenwaldring 29, 70569 Stuttgart, Germany

Abstract
Requirement documents are the basis for software developments in space engineering projects. They contain specifications on properties of the result or product. Precision of the stated requirements drives the probability that the final result is viable for the commissioner. In the work reality, space projects - already on software level - consider thousands of requirements. The majority of requirements are stated in natural language.
During most projects individuals transfer requirements to individuals with subsequent manual and to a degree subjective interpretation. Due to project complexity catching misinterpretations early on is challenging, possibly leading to an increase in cost and development duration. Manual transfer and transformation to software tools is a significant work load. Processing requirements readable by machines is therefore a potential for optimizing the design process, reducing cost, effort, and error rate.
ExANT (Extraction of requirements from natural language texts) is a project that evaluates the feasibility of using artifical intelligence natural language processing for processing software specifications to enhance the digitization of the software development process. The consortium consists of space software development experts from Gerlich System and Software Engineering (GSSE), software engineers from OHB System AG and OHB Digital Services, as well as the Institute of Space Systems University of Stuttgart (IRS) to combine academic and industry knowledge and expertise.
The scope of this project considers additionally to being a feasibility study, potential for cost and effort reduction or at least the possibility to enhance requirement quality in order to increase clarity and precision. For computerized natural langauge processing open-source tools as well as commercially available tools are evaluated. Prior to training the respective neural network, the preparation of data in natural language texts and its annotation is required. This is necessary to be able to later correctly derive contents, structure and logical contexts of given requirements. Preliminary considerations for natural language processing of software specifications and the respective annotation process with the help of IBM Watson are given in this paper.

Keywords
Artificial Intelligence; Natural Language Processing; Requirements Engineering

Abbreviations

AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
DLR	German Aerospace Center
ExANT	Extraktion von Anforderungen aus natürlichensprachlichem Text
GUI	Graphical User Interface
IBM	International Business Machines Corp.
INCOSE	International Council on Systems Engineering
NLP	Natural Language Processing
UML	Unified Modelling Language
QA	Question and Answering

1. MOTIVATION

Requirements engineering is one of the first steps of developing a product. The quality of the given requirements is a driver of total project costs and the required time [1]. In literature [2] it is stated that 50% of errors have been introduced during the requirements engineering phase as well as the fact that 80% of design iterations are based on incorrect requirements.
Such errors can be classified as mainly human caused. Thus, removing the human element by automated processing or minimizing the human impact by automated checking and correction has significant potential for improving the requirements engineering phase in terms of reliability, while providing overall project acceleration.
The INCOSE (International Council on Systems Engineering) Guide for Writing Requirements [3] provides a number of properties a good requirement

should have: necessary, appropriate, unambiguous, complete, singular, feasible/realistic, verifiable, correct, conforming, consistent, able to be validated, accurate, concise.

In an ideal world, every single requirement provided to the developer team by a customer would be evaluated with regards to all the previously stated properties. These validated requirements are then transferred via lossless communication to experts of identical knowledge level. In the real world neither sufficient time nor full competence in breadth and depth for all requirements and underlying complexities are allocatable resources. Especially, as most requirements are given in natural language, subjective text interpretation is a likely cause for errors and can never achieve full consistency.
Thus, an entity for consistent and reproducible natural language processing (NLP) is a necessity for being able to automate and accelerate the process. Such entities exist in the form of methods in the artificial intelligence (AI) subfield NLP.

2. FUNDAMENTALS

In this section the necessary basics for requirements engineering as well as basics on NLP as well as the commerical NLP tools available through IBM Watson are explained.

2.1. Requirements Engineering

Requirements engineering is the process of defining, documenting and maintaining requirements [4]. The full process of requirements engineering includes: Requirements inception/elicitation, analysis, system modelling, specification, validation and management [5].
Within the scope of the ExANT project requirement analysis is the main focus. Requirement specification, validation and management are assessed for possible support when applying NLP to the problem. System modelling based on NLP, like for example via UML, is not in scope of the project.

2.1.1. Shall method

The ECSS Standard ECSS-E-ST-10-06C [6] describes the "shall method", which is the de facto requirements standard for academic space projects. The standard describes the characteristics of technical requirements divided into: performance, justification, configuration management, traceability, ambiguity, uniqueness, identifiability, singularity, completeness, verification and tolerance.
The standard goes further into hard definition of key words and their meaning, giving the "shall method" its name.
a) The verbal form "shall" shall be used whenever a provision is a requirement.
b) The verbal form "should" shall be used whenever a provision is a recommendation.
c) The verbal form "may" shall be used whenever a provision is a permission.
d) The verbal form "can" shall be used to indicate possibility or capability.
Via explicit wording it is directly evident that a sentence including "shall" is a requirement. Additionally, the standard restricts the use of certain terms usually being insufficiently precise.
A few examples are: "and/or" - which is it? ; "user-friendly" - depends on type of user ; "small/large" - compared to what? ; "sufficient" - not quantified ; "best possible" - unclear metric. For the full list see Reference [6] and similar documents.

Here, the ECSS standard for requirements already attempts to establish a rule-based set to enhance the quality of requirements. Such a rule-based set is fairly straightforward to implement for automation as this can be achieved with direct string comparison and highlighting. As this standard exists for many years and good requirements are still a challenge, additional care is required to achieve better requirements.

2.1.2. Use-case method

An alternative approach for requirements formulation which is more established in industry is the so called "use-case" method developed by Alistair Cockburn [7] [8]. Here a requirement is not a single sentence containing the keyword "shall" but a structured prose text detailing the use case at hand.
*"The problem is that writing use cases is fundamentally an exercise in writing prose essays, with all the difficulties in articulating **good** that comes with prose writing in general."* The book continues further that it is hard to identify a well written use case, but even harder to write one.
A use case describes a system behaviour under various conditions in response to a primary actor. The *primary actor* interacts with the system to achieve a goal. The use case is a collection of responses, event sequences and scenarios for this interaction. The *main success scenario* describes the system behaviour in case of nominal behaviour. While the use case can only be executed when described *preconditions* are met. In most cases a *trigger* is then required to start the use-case behaviour. Additionally *extensions* are permitted to describe alternative scenarios to the main success scenario. Use cases may reference other use cases. Further use-case clarification can be achieved when stating at what *level* the use-case is described as well as who are the *stakeholders and interests*.

While the use-case method yields a description of complex system behaviour and the shall method generates a long list of comparatively short statements both methods are challenging to execute well in a dynamic real world environment. Automated processing and analysis of requirements in either form would increase overall productivity. Such a tool might be found within the AI subfield natural language processing (NLP).

2.2. Natural Language Processing (NLP)

In general NLP is the scientific field of computer-human language interaction, i.e. the processing, analysing, "understanding" of contents and contexts of natural language. While recent advances in speech recognition and generation are impressive, the context of this work is language in text form.
In this section a very brief overview of the tools applicable to the ExANT project is given. To produce an AI model that has a degree of understanding language the neural network needs to be trained on relevant data, here natural text. The challenge here is that natural language processing for specific use cases suffers from a shortage of training data, where human labelled training data on the order of hundreds or thousands is available [9, 10] - which is insufficient for training an NLP AI for more general applications.
In the ExANT project a similar order of requirements is expected to be manually annotated. Thus, pre-training on a sufficiently large data set is required to obtain an NLP AI with basic language understanding.

2.2.1. Open-source tool: Google BERT

This task is accomplished with the example of the open-source tool Google BERT (Bidirectional Encoder Representations from Transformers), which has been trained on the BooksCorpus (800 million words) and the text-only variant of English Wikipedia (2.5 billion words) [12]. The method is a representant of the AI class of transformers, which determines patterns to transform one string of symbols into another string of symbols. This class is part of deep-learning methods, where a large number of hidden layers between inputs and outputs are used. Each layer might be seen as a representation of an abstraction of the input symbol string.
The novelty of the BERT approach is the bidirectionality, which considers language context in both directions before and after a word to extract its meaning. For example the word "bank" has different meanings when "bank account" or "bank of river" is considered within the wider sentence context [9].
The pre-trained BERT model can then be further fine-tuned to specific use cases, where significantly less additional training data is present. A context relevant example can be found in Ref. [13], where a fine tuned BERT model is applied as QA machine to successfully determine relevant elements of given requirement texts.
The application and evaluation of open-source NLP tools is performed by the consortium partner OHB Digital Services GmbH. For a more detailed description of the open-source tools considered within the ExANT project please refer to Ref. [1].

2.3. Commercial tool: IBM Watson

The NLP AI system IBM Watson came to initial fame by winning the game "Jeopardy" in 2011 against the standing champions [14]. The challenge of the game is to determine the questions for a given answer or set of answers. At this time IBM Watson was part of the system of Question and Answering (QA) machines, which was able to outperform human players by combining many language processing techniques of the time in a highly time efficient manner.
Since then applications and capabilities of IBM Watson have been vastly expanded and are commercially available on the IBM cloud[1]. The relevant tools (Watson Knowledge Studio, Watson NLU, Watson Discovery) are introduced here.

2.3.1. Watson Knowledge Studio

Utilizing a pre-trained Watson language model is a de facto default use-case. Pre-training on large bodies of text, e.g. Wikipedia, and more specialized bodies of knowledge, e.g. contract /insurance texts are a possibility. Watson Knowledge Studio is the tool where a customized entity and relationship model is generated [15].
Manual and automated annotation of reference texts can be performed here. The tool offers a no code environment to allow domain experts to teach the NLP system the specific domain language required to solve a specific problem. Domain experts aid in describing field unique relationships and entities on the basis of a so called corpus of knowledge.

It is vital to have more than one human annotator in this process to achieve a more objective model. If a single annotator would process the full body of reference text, his/her subjective way of annotation would then be transferred to the language model. This should be avoided to de-bias the model. For this purpose the tool supports collaborative annotation, as well as comparison on performance of annotators and showing discrepancies. This can also be utilized to further fine-tune according to the language nuances of the domain. Rules for semi-automatic annotation can be defined here.
After annotation the model is trained. Respective language model performance metrics such as precision, recall, inter-annotator agreements are directly presented in a model performance dashboard. These quantities are of special interest for the ExANT project to estimate the usability of the tool. Initial annotation of the system is an additional effort. Resulting performance of the trained language model should have high precision and recall, while minimizing human effort.

2.3.2. Watson Natural Language Understanding

Watson NLU is able to analyse text and to extract meta-data from content, such as concepts, entities,

[1]https://www.ibm.com/cloud

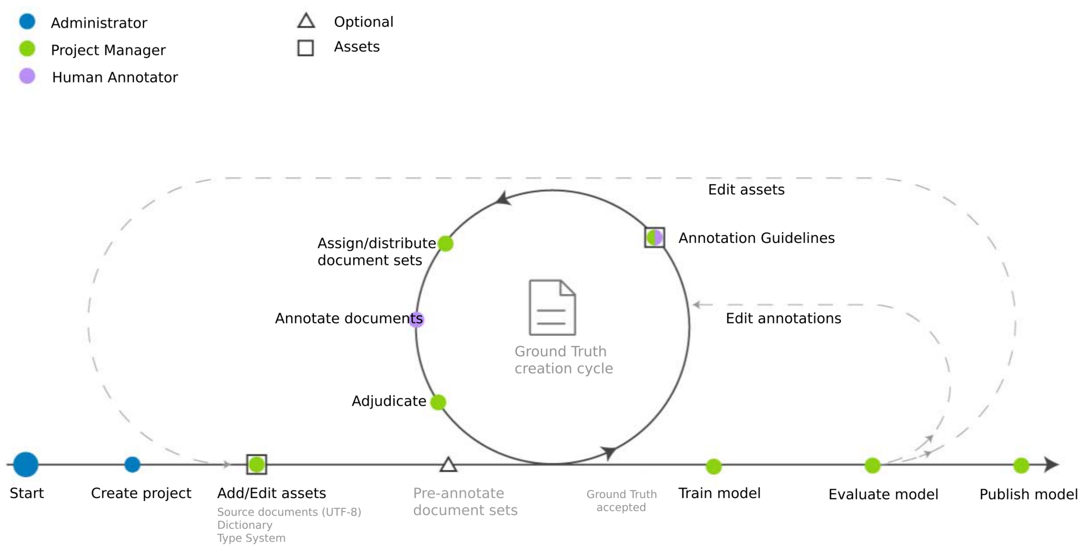


FIG 1. Workflow of developing a machine learning model within IBM Watson Knowledge Studio for annotating documents, training and evaluating a model with indication of the roles of administrator (blue), project manager (green) and annotator (purple) [11]

keywords, categories, relations and semantic roles [16].
It is currently capable to process text in eleven languages, in addition to the language models defined with Watson Knowledge Studio.
An ability of Watson NLU is to derive sentiment and emotion of texts. This is unlikely to be viable for the ExANT project as requirements should be written in an objective manner. Nonetheless, it might be possible to exploit the feature of detecting emotional language to request a user to rewrite into a more objective style.

2.3.3. Watson Discovery

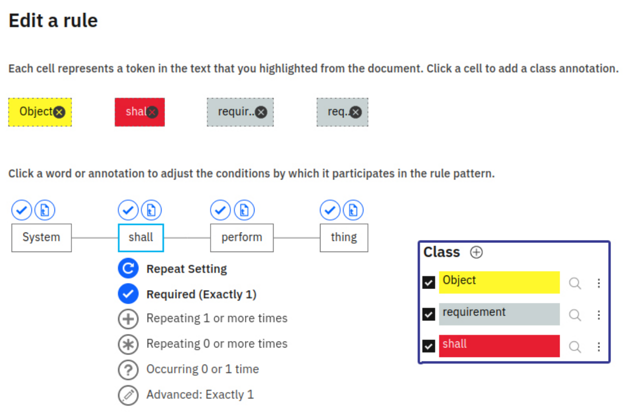
With a pre-trained model enhanced by Watson Knowledge Studio language model in combination with the derived natural language understanding Watson Discovery is empowered to operate as question and answering machine on unknown text [17]. The Discovery tool has been shown as capable to consistently yield productive answers to questions of similar context but different wording. Discovery is likely to be able for an application similar to the BERT QA machine demonstrated on technical requirements in Ref. [13].

3. UTILIZATION

3.1. Overview

The workflow in Fig. 1 shows how an NLP project is set up. First an assigned IBM cloud administrator creates the project, then the project manager begins with adding and editing assets. These are mainly documents to be analysed and processed. Next is the big iteration loop of annotating documents, based on annotation guidelines, distribution documents and performing the annotation. A final manual step is the adjudication, here discrepancies between human and machine annotation need to be resolved to establish the ground truth of the data. After that the machine learning model can be trained and evaluated.

The annotation process requires a type system. A suitable type system is needed for the annotation process, as it controls how the content can be annotated, i.e. the available categories for annotation. The type system contains the type of label-able entities and how relations are labelled. An entity type can be a technical *requirement*, with associated subtypes of *functional/performance/design requirement*. A type relation example might be a *performanceby*, describing an entity that is performing a certain performance for another entity.
For a productive type system dedicated care and experience is necessary. A domain expert can bring know-how from the domain side, while an NLP expert may aid in creating most suitable types. It has to be noted that the number of types, and thereby categories for annotation, should be low [18]. Both annotation and subsequent training of the NLP model would require significant effort, if too many types are defined. It is recommended by IBM, that a type system should be "as close to final as possible" before beginning large scale annotation to create a reliable statistical model. Too many rarely occurring types create a low-reliability model [18].



Edit a rule

Each cell represents a token in the text that you highlighted from the document. Click a cell to add a class annotation.

Click a word or annotation to adjust the conditions by which it participates in the rule pattern.

FIG 2. A simple example of defining the basic shall rule within Watson Knowledge Studio with three defined classes: object, shall and requirement. The class requirement here is a verb and a noun

3.1.1. Pre-annotation

To accelerate the annotation process it is possible to use four types of pre-annotation. These are in increasing complexity dictionary, rule-based, machine learning model and natural language understanding.
Dictionary pre-annotation should only be used when documents contain unique and specialized terminology. The tool will search through provided documents, when adding new definitions and suggest new possible entries. A useful example for such a rule are proper nouns. For example "DLRK" with the surface forms "DLRK 2022" and "Deutscher Luft- und Raumfahrtkongress" being defined as a noun can be a suitable dictionary pre-annotation definition. This is only true, if provided texts do not contain alternate mentions of "DLRK", a quick online search resulted for example in the "Dansk Land-Rover Klub" or "Door Lock Replacement Kit" when using abbreviations naively to pre-annotate.
Rule-based pre-annotation is the next level of abstraction. Here patterns of tokens/classes can be defined. These pattern rules that are then found and automatically annotated in text. For the classic "shall" requirement a very basic rule may consist of the tokens/classes:

object + "shall" + requirement.

The requirement token consists of at least one verb followed by at least one noun, see the illustration of the Knowledge Studio GUI in Fig. 2. Many of these rules can be defined for more complex and reoccurring patterns. Such patterns can be extracted from the ECSS standard on technical specifications [6] or the respective use-case method pattern as well as additional common patterns for technical writing, e.g. parameter and unit. The development of custom patterns is best performed while manually annotating the source material.

Both methods: dictionary and rule-based pre-annotation can be used to bootstrap the annotation process. This means that these annotation methods are best applied as early as possible in the annotation process.

Machine learning and NLU pre-annotation on the other hand are methods that enhance the annotation process once a first machine learning model is established.
The NLU pre-annotator might be seen as a "common knowledge" pre-annotation tool. This pre-annotator is recommended for a general knowledge subject. It is not recommended for documents of highly specialized fields. This means that NLU pre-annotator is out of the question for application within ExANT, as spacecraft engineering software requirements are certainly highly specialized.
Thus, the remaining option for productivity enhancing the annotation process may be found in the machine learning method. This requires a trained model. Thus, bootstrapping methods of annotation are a requirement beforehand. After training the model with sufficient data and assuming similarity to fresh documents this pre-annotation tool is recommended as a best choice.
The machine learning model neglects any entries of the dictionary, it does not assume the entity type when it was given an entity type in the dictionary.

3.1.2. Training

Typical for machine learning training requires a set of data to be processed. In our case, multiple documents with many individual requirements should be individually uploaded for creating the data set. This set is then, as is tradition, split into a training set, a test set and a blind set [19].
The training set is human or pre-annotated and utilised to train the model. This teaches correct annotation, including training with and "understanding" non-annotated text.
The test set is a set of annotated documents, which is used for analysis of the performance of the ML model, allowing to give indications for possible improvements of the trained NLP system.
Last, the blind set is distinctly separated from the previous sets. The ML model is forbidden to use this set during the training process. A potential weakness of a machine learning process is over-fitting. This means that it can be able to memorize the full data set, resulting in extremely good performance for the provided data, while producing bad results on fresh data. Thus, the blind set is set aside to serve as fresh data and is periodically utilised to evaluate the trained state of the NLP system.

3.2. Limitations

First limitations on applying NLP to the challenge of requirement definition and analysis became clear during consortial discussions. An NLP system can only evaluate what kind of text it was provided. This makes definition gaps very challenging to locate. Further, implicit expert domain knowledge is not directly textually represented in a requirement but a critical information. A wrong but possibly language wise "correct" requirement could be:

The data rate of the optical lens shall be at least 5 *W.*

The problems of this requirement that a lens can not generate a data rate and that a data rate has never the unit of power are implicit knowledge. The correct unit corresponding data rate might be captured with an extra rule, but the fact that a lens is unable to provide this entity is fully implicit. Additionally, simple ontological truths are not directly represented in any AI system. For example the reader will know that a "GPS" has a relation to navigation; an "AOCS" has a relation to altitude and tracking of a spacecraft. A trained machine learning model might learn that words of such a context occur with increased likelihood with each other, but still lacks a deeper understanding.
This would either require a fully functional and physically correct model and/or a "common sense AI". Both are not the scope of the ExANT project and each a very strong challenge by themselves.

Inversely it is not unlikely that contextually unrelated word pairs with high occurrence frequency appear near each other. This might happen when special unique names for mission, projects, instruments are frequently present in texts. Thus, it is required to monitor the trained NLP model for such behaviour and also evaluate how much human effort is required to minimize such errors.

A general limitation of a trained AI system is the quality of inputs. The provided requirements applied within ExANT might have unconscious biases in form of preferences in terms of structures, wording, semantics or logical relations. Thus, additional testing and validation of the NLP system on fresh unknown data is vital for the success of the use case.

Data preparation and enrichment is usually the most time consuming part of producing an AI application, part of the ExANT project is the evaluation whether this time is a meaningful investment in comparison to the conventional engineering process.

4. CONCLUSION

At the current state of the project conclusions on the feasibility of the utilizing NLP for accelerating and improving the software requirements development and evaluation process would be premature. As discussed the type system and annotation process is vital for any NLP project and requires significant effort to be performed well.

The abilities of the IBM Watson tool set appears very promising in terms of collaborative working, pre-annotation, annotation and NLP performance analysis. Nonetheless, further work is required to clear the picture for productive use as well as to ascertain the limitations of such an application better.

5. OUTLOOK

In the following months of the ExANT project the vital steps are establishing a corpus of knowledge as well as increasing the number of manually annotated documents for the provided software requirements within Watson Knowledge Studio. This data will then applied to train the NLP system, which infuses the Watson Natural Language Processing as well as Watson Discovery tools.
Next step is then to validate that the NLP system is capable to extract relevant entities of requirement natural language texts and identify required rules for further processing.
Application of the trained NLP system on unknown data of non-uniform quality , i.e. new requirements, is the follow-on step. The trained NLP AI will be evaluated to conventional performance parameters like precision and recall, but also stress tested in terms of identification of clearly wrong or malformed requirements.
General conclusions on this type of NLP AI system will be derived from this work as well as the comparison to competing open-source solutions based for example on Google BERT. Total efforts as well as assessing the amount of prior knowledge will be considered.

6. ACKNOWLEDGEMENT

The ExANT study is funded by the German Federal Ministry of Economic Affairs and Climate Action (BMWi), based on a decision of Deutsche Bundestag (The Parliament of the Federal Republic of Germany), FKZ 50RM2104B. We would like to thank the members (Rainer Gerlich System and Software Engineering GSSE, OHB System AG, OHB Digital Services GmbH) of the ExANT project for their support in this ongoing work.

Contact address:

ehresmann@irs.uni-stuttgart.com

References

[1] Rainer Gerlich, Ralf Gerlich, Ingo Schoolmann, and Martin Schorfmann. Searching for the hidden treasure: Formalization of textual requirements by ai. In *Proceedings of Data Systems in Aerospace 2022*, Online, May 2022.

[2] James Martin. *An Information Systems Manifesto*. Prentice Hall PTR, Harlow, England, 1984. ISBN: 9780134647692.

[3] International Council on Systems Engineering INCOSE Requirements Working Group. Guide fir Writing Requirements. Standard, San Diego, California 92111-2222 USA, July 2019. INCOSE-TP-2010-006-03.

[4] Murali Chemuturi. *Requirements engineering and management for software development projects*. Springer New York, New York, NY, 2013. ISBN: 9781461453765.

[5] Ian Sommerville. *Software Engineering: United States Edition*. Pearson, Upper Saddle River, NJ, 9 edition, 2010. ISBN: 9780137035151.

[6] European Cooperation for Space Standardization ESA-ESTEC Requirements and Standards Divison (ECSS). FSpace Engineering - Technical requirements specification. Standard, Nordwijk, The Netherlands, Mar. 2009. ECSS-E-ST-10-06C.

[7] Alistair Cockburn. *Writing Effective Use Cases*. Addison Wesley, Boston, MA, 2000. ISBN: 9780201702255.

[8] Alistair Cockburn Andy Pols Steve Adolph, Paul Bramble. *Patterns for Effective Use Cases*. The Agile Software Development Series. Addison-Wesley Professional, 2002. ISBN: 9780201721843,0201721848.

[9] 2018. https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[11] IBM. Machine learning model creation workflow, 2022. https://cloud.ibm.com/docs/watson-knowledge-studio?topic=watson-knowledge-studio-ml_annotator.

[12] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015. ISBN: 9781467383912.

[13] Haluk Akay and Sang-Gook Kim. Reading functional requirements using machine learning-based language processing. *CIRP Annals*, 70(1):139–142, 2021. ISSN: 0007-8506. DOI: https://doi.org/10.1016/j.cirp.2021.04.021.

[14] Jeopardy. Ibm and 'jeopardy!! relive history with encore presentation of 'jeopardy!: The ibm challenge', 2021. https://web.archive.org/web/20130616092431/http://www.jeopardy.com/news/watson1x7ap4.php.

[15] IBM. Ibm watson knowledge studio, 2022. https://www.ibm.com/cloud/watson-knowledge-studio.

[16] IBM. Ibm watson natural language understanding, 2022. https://www.ibm.com/cloud/watson-natural-language-understanding.

[17] IBM. Ibm watson discovery, 2022. https://www.ibm.com/cloud/watson-discovery.

[18] IBM. Establishing a type system, 2022. https://cloud.ibm.com/docs/watson-knowledge-studio-data?topic=watson-knowledge-studio-data-typesystem.

[19] IBM. Making machine learning model improvements, 2022. https://cloud.ibm.com/docs/watson-knowledge-studio-data?topic=watson-knowledge-studio-data-improve-ml#wks_mamanagedata.