Universiteit Utrecht

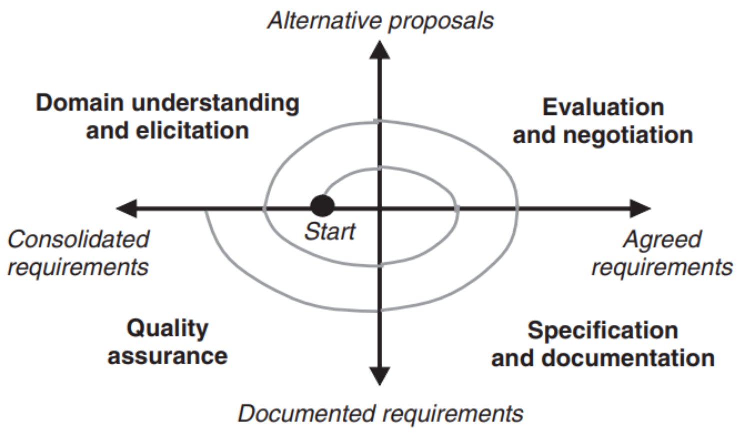
Faculty of Sciences

Designing an approach for highlighting requirements from elicitation interviews

Master Thesis

Xavier de Bondt

Artificial Intelligence


Alternative proposals
Domain understanding and elicitation
Evaluation and negotiation
Consolidated requirements
Start
Agreed requirements
Quality assurance
Specification and documentation
Documented requirements

Daily Supervisor:

T. Spijkman
fizor. business applications

First Supervisor:

Dr. F. Dalpiaz
Intelligent Software Systems

Second Supervisor:

Prof. Dr. S. Brinkkemper
Intelligent Software Systems

August 15, 2022

Contents

1 Introduction

Any software project starts with correctly understanding *what* the problem is, *why* this is a problem and *who* should be involved in solving this problem. This is what the discipline of Requirements Engineering (RE) expands on [1]. In the several phases of requirements engineering, many artefacts are created. The quality of the activities within RE affect the quality of the designed artifacts. As reported by Smith, Bieg, and Cabrey [2], about half of unsuccessful projects fail due to poor requirements management, hence the phrase "Poor requirements = Poor performance"

Requirements elicitation is an important phase of the RE process. As defined by Zowghi and Coulin [3], this is the process of seeking, uncovering, acquiring and elaborating requirements for computer-based systems. This phase comes with a set of problems that are not straightforward to solve.

One of these problems is that it relies on the communicative skills of the requirements engineer. There exist many techniques and approaches for requirements elicitation [1] and the choice for choosing one technique over the other is not always clear [4]. Whatever technique is chosen, it always involves human-to-human interaction, often in conversational form. This can be quite a challenge [5], since it attempts to explore the boundaries of knowledge, who possesses this knowledge and how to acquire that knowledge correctly [6]. Therefore it is important that we make the most of these moments of interaction.

This is where the usage of Natural Language Processing could be very helpful. *Natural language processing* (NLP) is a field that employs computational techniques to learn, understand and produce human language content [7]. In this thesis, we present a concept of an NLP tool that can assist humans analyze the conducted interviews. This is done by showing the important questions asked in a transcript, allowing the user to expand on these questions and see the answers. We can classify our tool according to the tool types for Natural Language Processing for Requirements Engineering (NLP4RE) [8]. The task of showing the user the important questions falls under the 'Search & Retrieval' tasks and after this, we categorize these questions, which falls under the 'Classification' tasks.

This approach provides a direct transfer of knowledge. Whereas in an elicitation interview, the interviewers notes are often not the precise wording of the interviewee. Next to that, it could uncover certain things that would have gone unnoticed. This could lead to better requirements, thus better performance [2].

1.1 Problem statement

While the different accounts on the problem of poor requirements, as reported by Smith, Bieg, and Cabrey [2], created awareness of the problem, it remains a problem. The consequences of such mistakes can be severe, therefore some form of aid is needed for this process. Especially, since this task is so skill-dependent.

There has been a lot of research in the usage of Natural Language Processing for Requirements Engineering (NLP4RE) [8], creating more possibilities to guide the process of requirements engineering. These researches mainly focus on improving artefacts resulting from phases of requirements engineering.

While this research has been important, little to no work has been conducted regarding the application of NLP4RE into processing pre-requirements specification artefacts. In this research we aim to do this by using transcripts of elicitation interviews. By aiding requirements engineers directly after such an interview, the core problem of misunderstanding or missing requirements can be mitigated. This will benefit the entire requirements engineering process and improve all resulting artefacts.

1.2 Research Objective and Questions

Due to COVID, there has been a huge increase in online communication via online platforms. According to the Central Office of Statistics in the Netherlands, in 2019 approximately 20% of people in the ICT sector were working at home, while this increased to 40% in 2020 [9]. Nowadays it seems more and more common to communicate via these online environments, allowing the recordings of these conversations to be more frequent. These recordings gives us the opportunity to aid requirements engineers by showing requirements relevant information in the transcripts of these interviews. Especially since more platforms like Microsoft Teams now allow to record transcripts of the conversations[1]. By considering this highlighted information, it

[1]https://techcommunity.microsoft.com/t5/microsoft-teams-blog/live-transcription-with-speaker-attribution-now-available-in/ba-p/2228817

could improve the quality of the resulting artefacts. Another reason for showing these requirements relevant parts of the transcript can be to validate the resulting artefacts. By reviewing the highlighted parts, a requirements engineer could acquire more context and refresh their memory.

This research aims to design a new tool, to identify this relevant information in a transcript from an elicitation interview. In particular, we aim to compare a NLP based approach to a machine learning approach. This allows us to test respectively a symbolic NLP approach and a statistical NLP approach [8]. In the case of the symbolic NLP approach we might encounter a lack of flexibility, while the statistical NLP approach might have a too large need for annotated natural language. Therefore it is interesting to see which approach works well in which scenario. For this comparison, we use a combination of data from an Master level RE course from Utrecht University and real-world cases provided by fizor.

The main research question that will be discussed in this thesis is the following:

MQ: How can we identify requirements-relevant information in a transcript from a conversation between a business analyst and a stakeholder aiming to elicit requirements?

In order to do this, we will answer a set of sub questions (SQs).

- **SQ1: What are state-of-the-art techniques for identifying requirements-relevant information in conversations?**
  This research question is addressed via a literature research that explores the different possibilities for identifying requirements-relevant information in conversations. This allows us to not re-invent the wheel, but use the knowledge that has already been published, considering the benefits and drawbacks of different approaches.

- **SQ2: How to design an approach for identifying requirements relevant information?**
  The sub question described here tackles the design phase of our approach to identify requirements relevant information. This design will be done in two different ways: a NLP based approach and a machine learning approach. Furthermore, this sub question entails finding out in what cases which approach works best.

- **SQ3: What is the expected effectiveness of the approaches from SQ2?**
  To test the expected effectiveness of the approaches, we make use of interviews made during the Requirements engineering (INFOMRE) course [10], given at Utrecht University by Fabiano Dalpiaz. This will give us a view on how well these approaches work and how useful the approaches are expected to be in practice.

- **SQ4: How can domain information improve our approach?**
  A key problem for NLP4RE tools is domain specificity [8]. There is a shortage of RE specific resources, making it hard to train NLP4RE approaches, since they would have to use general-purpose language, while they need to model this domain specific language.Depending on the case, there is often some form of domain information available, prior to the requirements elicitation interview. This sub question explores how this information could be a part of our approach and how much this will improve the outcome.

This set of sub questions can be visualized in Figure 1.1. It is important to note that the fourth sub question has an overlap in both the second and the third, while the first sub question stand apart from this. Furthermore, the sub questions one, three and four form a cycle. This is connected with the Emperical Cycle of Design Science as described by Wieringa [11], which will be covered in Chapter 2.

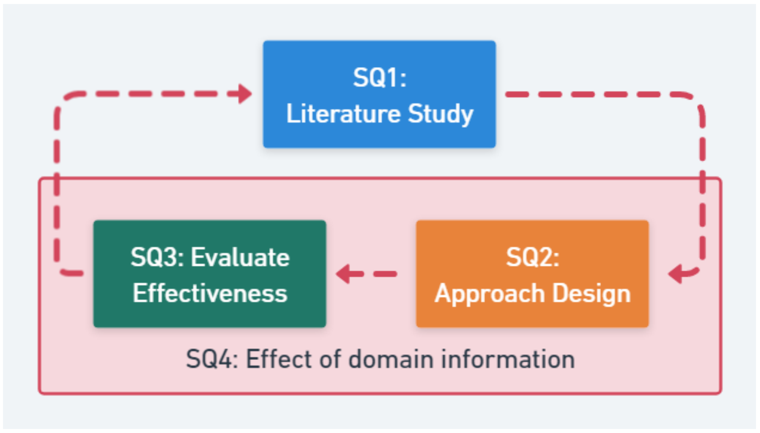


Figure 1.1: A visualisation of the sub questions of this research

1.3 Thesis outline

This thesis is structured in the following fashion: In Chapter 2 we will discuss our structure, goals and foreseen challenges. This will go over the usage of the Empirical Cycle, specifying how we will use it in this research. After that, in Chapter 3, the necessary background knowledge will be explained. This will go over the basics of Requirements Engineering, Natural Language and Natural Language Processing and Machine Learning. Then, in Chapter 4, the related works will be discussed. This reports on the most relevant papers, highlighting the parts that can be useful for this project. Finally, Chapter 5 explains how this tool was developed. We conclude the thesis in Chapter 6 with a discussion of the the research questions.

2 Research

2.1 Structure

As this thesis focuses on developing a tool for showing requirements from elicitation interviews, we will need to design an artefact that is able to do so. Furthermore, we want to advance scientific knowledge beyond current knowledge by doing so. Therefore, we make use of the Emperical Cycle of Design Science as described by Wieringa [11]. This cycle, as shown in figure 2.3, consists of five phases:

1. *Research problem analysis*
   In this phase we clearly state what the research problem is that has to be solved and frame it. In our case we try to answer our different research questions. This first focuses on researching the literature for similar approaches to classify requirements, after which we will start the design of our tool and test it on several real-world cases, taking into consideration that we sometimes have the possibility to use a domain-model in our approach.

2. *Research design and inference design*
   This research will make use of recordings from the Requirements engineering (INFOMRE) course [10], given at Utrecht University by Fabiano Dalpiaz. The recordings from 2021 were only on three different systems, giving us two to four recordings of these conversations per system. Furthermore, they all have a system description, giving us the possibility to create a domain model to use in our approaches. Since these recordings are fairly generic per system, this is a good starting point for designing our tool. This will be done as a *sample-based research*, where we research all the different recordings as a sample of cases, as shown in figure 2.2. This phase will start with creating a NLP based approach, after which it moves on to creating a machine learning approach.

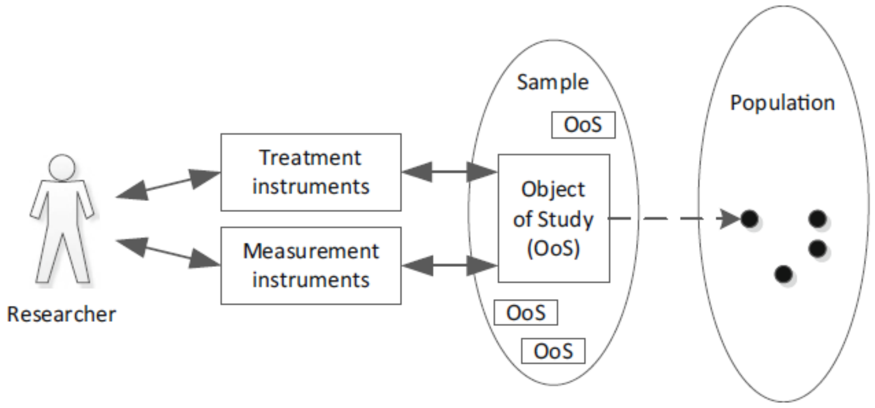


Figure 2.2: Empirical research setup, where sample-based research studies samples of objects of study (OoSs)

3. *Validation of research and inference design*
   To validate our models, we can use the recordings from the INFOMRE course. These refer to three different domains, each having domain information in the form of a system description (see Appendix B). This allows us to compare the algorithm over different domains, but having several recordings for each domain. This would show how well our model generalizes over these different domains.

4. *Research execution*
   This phase will focus on the execution of our model. Does everything work correctly? Are there any critical errors and does each step do what we want it to do?

5. *Data analysis*
   Finally, we analyze the results from our models. Is one model significantly better than the other? Why does this model recognize certain requirements while the other one does not? What requirements did we find that are also found by the business analyst? Did we find any new requirements? How useful is

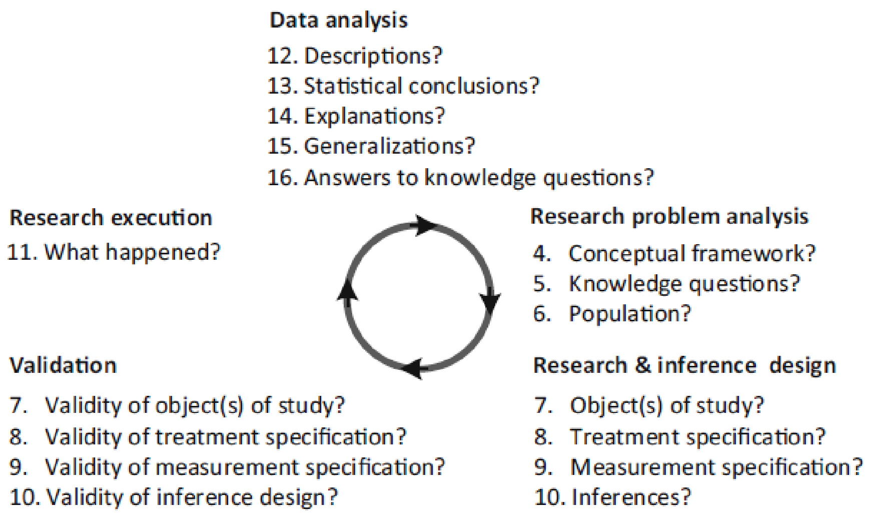

Data analysis
12. Descriptions?
13. Statistical conclusions?
14. Explanations?
15. Generalizations?
16. Answers to knowledge questions?
Research problem analysis
4. Conceptual framework?
5. Knowledge questions?
6. Population?
Research & inference design
7. Object(s) of study?
8. Treatment specification?
9. Measurement specification?
10. Inferences?
Validation
7. Validity of object(s) of study?
8. Validity of treatment specification?
9. Validity of measurement specification?
10. Validity of inference design?
Research execution
11. What happened?

Figure 2.3: The empirical cycle [11]

the output according to a business analyst? What would be the precision and the recall for a certain case?

To start, we will compare the requirements elicitated by the business analyst to the parts that are highlighted by the tool. As the tool develops, we will check the usefulness of the output by reviewing the outputs with practitioners. From there on, we can check for ourselves which outputs are generated correctly or not and validate from there forth. Finally, we could work out a case fully and check what parts are missing and get a sense of precision and recall.

2.2 Goal

The goal of this thesis is to aid the identification of requirements, by conceiving a tool that is able to show requirements relevant transcript segments, to improve the quality of the resulting process. By using two different approaches, a NLP based approach and a machine learning based approach, we can determine in what situation which approach works best. Furthermore, in the process of developing these different approaches, their findings can be of interest for each other. For example, when certain parts of the NLP based approach are very insightful, this might prove to be a good feature for the machine learning approach.

This tool is intended to help requirements elicitation, but this can be done in different scenarios:

- *Custom development*
  When it comes to custom development, there is a lot of requirements relevant information in the transcript of the conversation between the business analyst and the stakeholders. To aid the identification of the requirements, our tool could be useful to distinguish between information that is relevant or irrelevant to the software system. A taxonomy for the scope of this system could prove very useful in making this differentiation.

- *Replacing a legacy system*
  Replacing a legacy system involves differentiating between things that must change in this system and things that must remain the same. The important distinction to make here is what parts of the software system have to change and what must remain the same. Furthermore, we should look out for indications that something is outdated, since these parts must be replaced as well.

- *Extending an existing system*
  Likewise, when adding to an existing system we must differentiate the things that must change in the system and the the things that remain the same.

While this tool can be used in different scenarios, it can also be used in different stage of Requirements Engineering:

- *Before specification documents are written*
  When using the tool in the stage where the specification documents are not written, it can help with writing the specification documents. The difficulty here lies in terms of validation. This could not be validated by comparing it to the requirements mentioned in the specification document, but we must rely on another way of validating the tool. This validation could be done by questioning the business analyst how useful the output is as an aid to create the specification document. Another, less representative way of validating would be to compare how many of the outputs that are generated are classified as correct by an expert. Finally, as in all cases this transcript could be fully worked out and compared to the output generated, giving us insight into precision, recall and other metrics.
  In future research it would be interesting to see a form of action research [12], where the tool is used during the elicitation interview, highlighting relevant parts of the live transcript.
  Another possible experiment would be to test the user interaction with the system via A/B testing [13]. This experiment would entail asking practitioners to perform an interview based on a case study, where the practitioners are split into two groups; A and B. Group A will report on this interview as they would normally, whereas group B will report on the interview, assisted by the tool.

- *After specification documents are written*
  On the other hand, using the tool after the specification documents are written would allow the output to be validated by comparing it to the requirements mentioned in the specification document. Furthermore, this document could provide additional context and traceability. Next to that, the other validation methods mentioned in the other stage could be used as well.

2.3 Challenges

This research, as any research, comes with its own set of challenges. As seen in Section 2.2 there are many different ways to validate our tool. This can be done in four different ways:

- The first way is to compare the requirements elicited by the business analyst and written in the specification document to the output of the tool. This can be a useful first insight, but ultimately defeats our goal of aiding the business analyst as it recreates their findings.

- Another way of validation would be to test the perceived usefulness of the output of the tool. This can be done by creating a questionnaire and asking business analysts how useful they rate the output to be. This can be a good indication for improvement between iterations of the tool.

- A less representative validation method would be to check whether the output of the tool is correct or not, giving a percentage of correct outputs. This can be used as a quick way of validating the tool, but should not be considered to be very representative, since the missing requirements are not mentioned. However, since we do have True Positives and False Positives, we can calculate the precision of the tool.

- Finally, the most representative way of validating would be to go through the entire transcript and manually highlight the relevant information and compare this to the output of the tool. The drawback is that this takes a high amount of time, but this does give us an insight into precision, recall and other useful metrics.

The choice between these different ways of validating are different for each of the scenarios mentioned in Section 2.2.

While other approaches have clear candidate requirements when selecting or demarcating them, we are using transcripts of a conversation, thus not having any clear requirements. They are often not in perfect English, sometimes containing wrong words. The challenge here is to focus on what information could possibly be relevant in the future; what is relevant for our cause.

When creating the machine learning approach, there is a need for labeled data. Labelling a whole case would take a considerable amount of time, but it is necessary to create an approach based on machine learning. Transfer learning [14] can help overcome this issue, but perhaps this will go at the expense of the generalization of such a model. This must be thoroughly researched and thought out during the process of the creation of the machine learning approach.

3 Background

3.1 Requirements Engineering

Creating a software system involves correctly understanding what the problem is that needs to be solved. This comes with the task of discovering *what* the problem is, *why* it is a problem and *who* should be involved in solving that problem. After this discovery, there will be a formulation, analysis and agreement on these problems. This is what the concept of Requirements Engineering (RE) goes into [1].

Requirements engineering for a single software system involves different activities and actors. Each of these actors are so-called *stakeholders*, which are people that are affected by the software system that is discussed. Every stakeholder has a certain influence on this system and play an important role in the requirements engineering process. This process of requirements engineering is guided by a *requirements engineer*. A requirements engineer's aim is to help guide the software design and development of the system as correctly as possible.

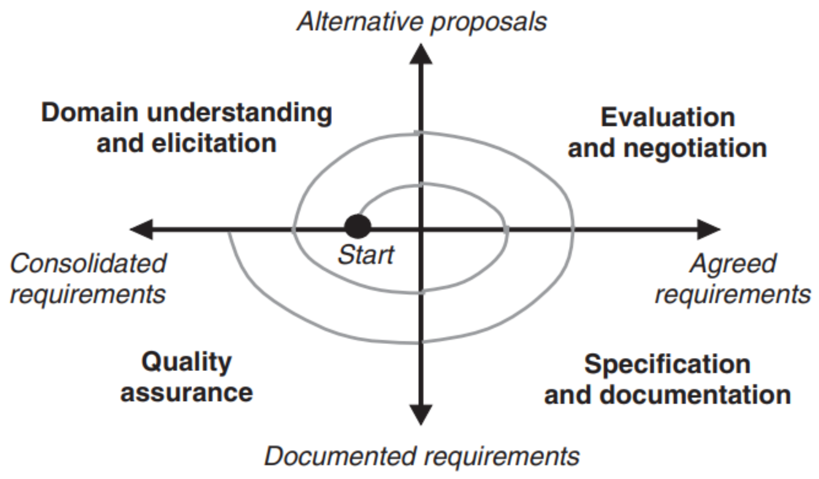

Alternative proposals
Domain understanding and elicitation
Evaluation and negotiation
Consolidated requirements
Start
Agreed requirements
Quality assurance
Specification and documentation
Documented requirements

Figure 3.4: The requirements engineering process [1]

The phases of requirements engineering can be depicted in figure 3.4 following the handbook of Van Lamsweerde [1]. Note that this is one way of describing the several phases of requirements engineering, other handbooks such as the handbook by Glinz et al. [15] may be different. While these phases may be intertwined, overlapping or going back and forth, this model does give a general and flexible view on the process. We will only focus on the first phase of this model. Requirements traceability is another important notion to requirements engineering, essential for developing better quality software systems [16]. Requirements traceability is defined by Gotel and Finkelstein [17] as the ability to describe and follow the life of a requirement in both a forward and a backwards direction. There exist two types of requirements traceability:

- *Pre-requirements specification (pre-RS) traceability*
  This type of requirements traceability is concerned with the aspects of a requirement's life before its inclusion in the requirements specification.
- *Post-requirements specification (post-RS) traceability*
  Post-RS traceability is about the aspects of a requirement's life that result from its inclusion in the requirements specification.

Domain understanding and requirements elicitation

The first phase of the spiral process model is domain understanding. This is the study of the system within the organization and its context, figuring out what the cause of the problem is and where it is situated

[1]. Furthermore, it creates a picture of the involved stakeholders, the scope of the system, the organization and the strengths and weaknesses of the system. Ideally, this phase should result in a proposal that describes these aspects and furthermore describes a *glossary of terms*. This glossary of terms describes the key concepts on which everyone should agree and makes sure nobody uses the same term for a different concept or the same concept for different terms.

Requirements elicitation is the process of seeking, uncovering, acquiring and elaborating requirements for computer-based systems [3]. It involves discovering candidate requirements and assumptions that will create the solution and delves into the symptoms, causes and consequences of the weaknesses of the system. This is done in an incremental fashion, by exploring the entire problem. Requirements elicitation is a so called *cooperative learning process* [1], where the requirements engineer and the stakeholders collaborate closely to elicit the correct requirements. This is a very critical phase, since poor requirements result in poor solutions [2].

Requirements elicitation comes with a set of problems [3]. The *first* being that the requirements are spread across many sources; we have problem owners, stakeholders, documentation of the system and sometimes other existing systems. Therefore, many different techniques are used from different non-computing fields of science such as social sciences, organizational theory, group dynamics and knowledge engineering.

The *second* problem is that of communication. Requirements elicitation depends highly on the communications skills of the requirements engineering. As an example, the requirements engineer has to be a good listener, speak clearly and stay professional. Likewise, the commitment and cooperation of the stakeholders influences the requirements elicitation greatly. Internal politics or an over-scheduled stakeholder can get in the way of the requirements elicitation.

The *third* and final problem is that, there is not one single solution for requirements elicitation. Every customer will be different, some companies are small and some may have very large enterprise systems. Sometimes it involves maintenance of existing systems or legacy systems, but other times it might involve creating a new system or adding new features to an existing system. This makes requirements elicitation a difficult concept that requires choosing the right techniques for the right circumstances and context.

There exist many techniques and approaches for requirements elicitation [1]. While this choice for selecting an elicitation technique is so difficult that Carrizo, Dieste, and Juristo [4] proposed a systematic way of selecting these techniques, we will simply name a few. The most traditional and commonly used technique being interviews [3], which we will elaborate on in Section 3.2. Furthermore, questionnaires are often used during early stages. Task analysis and domain analysis give very good examples. Group work and brainstorming are techniques that serve as extra engagement. Ethnography and observation are about studying the current processes. Prototyping can be very helpful for human-computer interfaces and finally, scenarios can be very useful for understanding and validating requirements.

3.2 Interviews

As we noted in Section 3.1, interviews are one of the most commonly used techniques for requirements elicitation. Furthermore, in this thesis we will be working with transcripts made of elicitation interviews. It is defined by Briggs [18] as follows: "An interview is a communicative event in which the interviewers asks questions to reach the reality of a phenomenon conceived inside the mind of the interviewee." This can be quite a challenge [5] and attempts to explore the boundaries of knowledge, who posses this knowledge and how to acquire that knowledge correctly [6]. These boundaries of knowledge can be expressed using the Tacit Knowledge Framework [19].

The Tacit Knowledge Framework uses four properties. Knowledge is

- *expressible* if it is known knowledge.
- *articulated* if it is documented as known knowledge.
- *accessible* if it is known, but not in the foreground of the stakeholder's mind.
- *relevant* if it is relevant to the project and the domain.

Using these properties, it gives definitions for

- *Known knowns*: expressible, articulated and relevant. This is essentially the information that is passed from the stakeholder to the analyst.
- *Known unknowns*: not expressible or articulated, but accessible and potentially relevant. Here, the stakeholder does not express this information, but the analyst suspects its existence.
- *Unknown knowns*: potentially accessible, but not articulated. Hence this is tacit knowledge: the stakeholder knows, but cannot articulate it.
- *Unknown unknowns*: not expressible, articulated or accessible, but still potentially relevant. This is information that is unknown to both the stakeholder and the analyst.

An example of the Tacit Knowledge Framework can be explained using the following example of an online marketplace.

- A *known known* is the amount of users that are using this marketplace. It is information that is passed from the stakeholder to the analyst.
- A *known unknown* on the other hand is how many of these users are selling fake goods. This is information that the stakeholder did not express, but the analyst does suspect that there will be users that attempt to sell fake goods.
- An *unknown known* is the fact that this marketplace is losing a lot of money, but the stakeholder is not allowed to express this due to the internal politics.
- An *unknown unknown* could be that the fact that the marketplace is losing money is due to internal fraud and money laundering. Neither the stakeholder nor the analyst suspected this or knew this.

When using an interview as a means of requirements elicitation, according to Ferrari et al. [20], three key steps have to be taken. The first step is preparing for the interview. This involves defining the purpose of the meeting, selecting the right person, researching the interviewee, creating the questions and arranging the logistics. When defining the purpose of the meeting, the interviewer should ask questions on whether they need to meet with this person and how it will benefit them both. Furthermore, the interviewer should select the right person. The stakeholder should be relevant, since all stakeholders differ in knowledge and perspective. Next to that, the research on the interviewee has to be done, gaining professional information and information on their relationship with the project. Moreover, the questions and question-types should be decided on. It is customary to start with open-ended questions, not forgetting to aim for obtaining both domain knowledge and process knowledge. Finally, the logistics for this meeting have to be arranged. Selecting a location that is suitable and contacting the interviewee and informing them about the purpose.

The second step is actually conducting the interview. The interviewer should build a rapport, layout the expectations, ask meaningful questions, listen actively, take notes and wrap up the interview. Building rapport can give an insight into the mindset of the interviewee. This rapport can be created on three levels: professional rapport, personal rapport or rapport in terms of the product. Next to this, it is important that the interviewer lays out the expectations and goals by sharing them. This helps to understand the interviewee's attitude towards the interview. During the interview, the interviewer should aim to ask meaningful questions, keeping their goals and expectations in mind. Therefore, the interviewer should also actively listen. This helps turning the stated requirements into actual requirements. Meanwhile, the interviewer should take notes of important information, using the purpose of the meeting as a guideline on how much to write down and what to focus on. Lastly, the interviewer should properly wrap up the interview. This involves taking out some time to conclude the interview, summarizing what was covered, asking for any concerns of the interviewee, planning for more time if needed, reviewing some items to follow up and thanking them for their time.

Finally, once the interview is the interviewer should reinforce what was achieved and build on it. This involves thanking the interviewee again, checking the notes taken and continuing the communication.

While following these three steps might seem very easy, research has shown that there is still a lot of mistakes that can be made [5]. The most common mistakes can be categorized into seven categories:

1. *Question formulation*
   This category of mistakes can be split into three main groups:

   (a) *Asking vague questions*
   One of the most frequent mistakes made is asking vague questions. Ambiguities in a question can result in issues later. The question should be understood by both the interviewee and the interviewer.
   (b) *Asking technical questions*
   Asking technical questions may result in an inadequate response, because the given interviewee does not hold that knowledge. This can also be intimidating and lead to bad rapport.
   (c) *Asking irrelevant or incorrect questions*
   Questions that are out-of-scope are a waste of time. Questions related to the solution are confusing and long questions are often not taken fully or misunderstood.

2. *Question omission*
   The most frequently made mistakes when it comes to omitting questions are the following: Not asking to identify other stakeholders, not asking follow-up or probing questions, not inquiring about existing systems or business processes, not asking about feature priority and not asking about the problem domain.

3. *Order in the interview*
   The order of topics also has to be logical, but often the mistake of an illogical order is made. Furthermore, going back and forth among topics is another bad habit that often occurs.

4. *Communication skills*
   The most frequent mistakes observed can be split into four groups:

   (a) *Unnatural dialog style*
   An interrogatory-style question is uncomfortable. Allowing the customer to create scenarios is more natural.
   (b) *Language-related issues*
   Speaking clearly and correct while minimizing mistakes and minding the pronunciation helps a lot when understanding each other.
   (c) *Low and unclear tone*
   Your tone should be motivating and the interviewee should be able to hear you clearly, otherwise this can be irritating.
   (d) *Poor listening skills*
   Sticking to your agenda is a mistake that leads to bad listening. Actively listening and reacting to the interviewee's words is essential in a good interview.

5. *Analyst behavior*
   These mistakes can be categorized into three categories:

   (a) *Confidence*
   An interviewer could lack confidence, resulting in the interviewee thinking the interviewer was not valuable or doing a poor job. On the other hand an interviewer could be overconfident and come over arrogant, thinking they knew all the answers.
   (b) *Passive attitude*
   Being an active participant ensures the interview does not dominate you and creates a better atmosphere for engaging the interviewee.
   (c) *Unprofessional behaviour*
   The necessity to be professional ensures focus on the interview and getting the most out of it.

6. *Interaction with the interviewee*
   Mistakes that are often made here is not creating rapport or trying to influence the interviewee with certain questions.

7. *Planning*
   Planning is very important. Poor time management, lack of preparation or long pauses during the interview can make it really uncomfortable for the interviewee and result in a bad interview.

3.3 User stories

User stories are a widely used notation for expressing requirements, consisting of three basic components:

- A short piece of text that describes and represents the user story.
- Conversations between stakeholders to exchange perspectives on the user story.
- Acceptance criteria.

This short piece of text that represents the user story captures the essential elements of a requirement:

- *who* it is is for
- *what* is expected from the system
- optionally, *why* it is important

A format that is popularized by Cohn [21] is "As a ⟨type of user⟩, I want ⟨goal⟩, [so that ⟨some reason⟩]. For example, "As a user, I am required to enter a strong password when creating my account, so that my account is secure". Acceptance criteria for this could be that it contains at least 10 characters, 2 digits and 1 uppercase letter.

In the dataset provided by the INFOMRE course, we also have the specification documents that were created after the elicitation interview. These specification documents are a set of user stories, created by the students.

3.3.1 INVEST

In order to assess the quality of user stories and improve them, there are only a limited number of methods. One of these methods is INVEST [22]. INVEST stands for

- ***I****ndependent*
  According to INVEST, the dependencies between user stories should be avoided to the extent that this is possible.
- ***N****egotiable*
  The details of a user story may be changed during the discussion in the iteration planning meetings.
- ***V****aluable*
  The user stories have to be valuable to the customer in some way.
- ***E****stimable*
  The user story is sufficiently detailed so that an estimation of the required effort can be made.
- ***S****calable*
  A user story should be small in effort, not containing any big requirements.
- ***T****estable*
  A user story has to be testable on its acceptance criteria.

3.3.2 QUS

A more advanced method for assessing the quality of user stories is the Quality User Story (QUS) framework [23]. This is a collection of 13 different criteria that determine the quality of the user stories based on their syntactic, semantic and pragmatic qualities.

Figure 3.5 shows the model for user stories as a class diagram. The user story is made up of four different parts, a *role*, a *means*, zero or more *ends* and a *format.* Note that an *epic* is essentially a large user story that can be spit up into multiple smaller, implementable user stories.

The *role* in this model represents the stakeholder that expressed the need in the user story. Alternatively, one could use personas, who have may be defined with more rigor and have clearer goals.

The *means* of a user story in this model consists of three common elements:

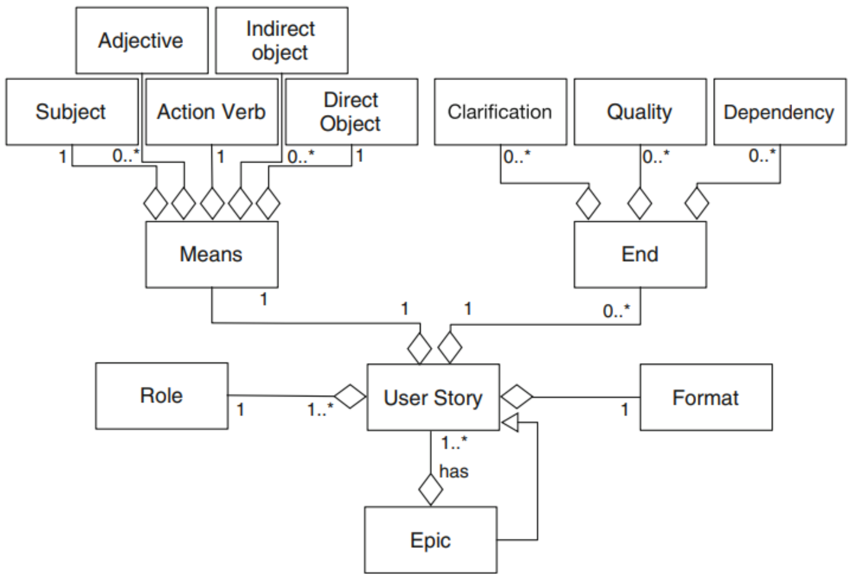


Figure 3.5: Conceptual model of user stories by the QUS framework [23]

- a *subject* with an aim such as 'wanting' something or 'being able' to do something.
- an *action verb* that serves as the action related to the feature being request.
- a *direct object* on which the subject executes the action.

An example could be "I want to open the document". To allow flexibility, this model has two common additions to its structure that are optional. These are an adjective and an indirect object. So extending our example, we get "I want to open a plain text version of the document from my co-workers' shared files". Here the plain text version can be seen as the adjective, and the co-workers' shared files are the indirect object.

The *end* of the user story aims to explain the means that are requested. There are three possible variants of a well-formed end in this model:

- *Clarification of means*
  This variant aims to explain the reason for the means. For example: "As a User, I want to edit my input, so that I can correct any mistakes"
- *Dependency on another functionality*
  Here, the end can implicitly reference to another functionality which is needed for the means to be realized. While we attempt to avoid dependencies, having no dependencies at all is impossible. As an example: "As a Visitor, I want to view the contact page, so that I can contact the business."
- *Quality requirement*
  This end expresses the intended quality of the means. For example: "As an Editor, I want to sort my footage, so that I can more easily select the one I need."

Note that these three types of ends can occur simultaneously, so they are not mutually exclusive.

Finally, the *format* in this model specifies that a user story should follow a certain pre-defined template. Since there are many existing ones, this allows for great flexibility in the QUS model.

In order to check the quality of these user stories, according to the QUS model there are 13 quality criteria it can follow. These are split into how these criteria are made, either syntactically, semantically or pragmatically, as shown in figure 3.6.

A subset of these criteria has a potential to create 100% recall, marking user stories that abide to these criteria linguistically 'good'. The following quality criteria are in this first subset:

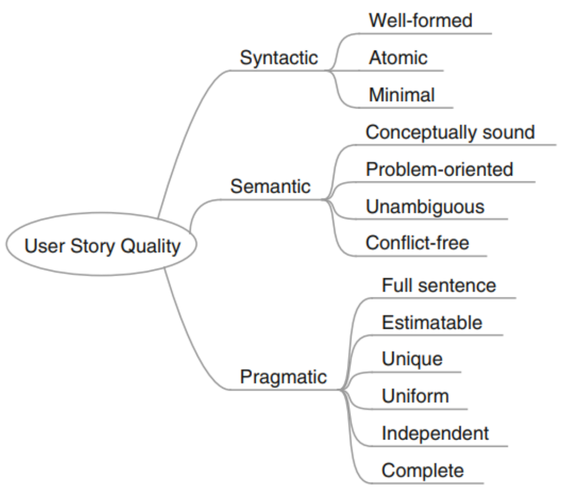


Figure 3.6: The Quality User Story framework defines 13 criteria for user story quality. [23]

- *Well-formed*
  A user story should at least include a role and expected functionality.
- *Atomic*
  A user story should express a requirement for exactly one feature or problem, no more.
- *Minimal*
  A user story contains no more than a role, an action and a benefit.
- *Conceptually Sound*
  The action only expresses a feature, while the benefit only expresses a rationale. These two should not be intertwined.
- *Problem Oriented*
  A user story does not go into the solution, but only specifies the problem.
- *Full sentence*
  A user story should read like a full sentence.
- *Uniform*
  All user stories should (roughly) use the same template.

The other criteria become more relevant after the stories are marked to be linguistically ‘good’ by abiding the first set of criteria. These second criteria are:

- *Unambiguous*
  A user story should avoid using words that have multiple interpretations.
- *Conflict free*
  A user story should not be inconsistent with other stories.
- *Estimable*
  A user story should be easy to plan. They should not denote an unrefined requirement that is difficult to plan and/or prioritize.
- *Unique*
  A user story should be unique; any duplicates are removed.

- *Independent*
  A user story should be self-contained and should have no inherent dependencies on other stories.

- *Completeness*
  A set of user stories should create a feature-complete application when implemented correctly, where no steps are missing.

3.4 Acceptance criteria

As mentioned in Section 3.3, a user story also contains certain acceptance criteria. While the user story itself may be very broad, the acceptance criteria specify more precisely on what to make. These acceptance criteria aim to explain when a solution is acceptable to the customer. An example of an acceptance criteria would be "A customer should not be able to use invalid credit card details". This description leaves a lot of room for ambiguity and misunderstanding. By making it more specific and clear, we ensure that the solution will always be the same. In our example, it would look something like this: "Given that a customer enters their credit card number and it isn't exactly 16 digits long, when trying to submit the form, it should be re-displayed with an error message and point them to the correct number of digits". Furthermore by showing this back to the customer you have more specific information to talk about.

Gherkin [24] is a lightweight structure for documenting examples of the behavior our stakeholders want, in a way that can be easily understood by both the stakeholders and the developers. Its primary goal is to increase human readability, which is achieved by giving us a strict format on our acceptance criteria. Every acceptance criterion can be written in the form "Given ⟨some context⟩, When ⟨some action is carried out⟩, Then ⟨some set of consequences⟩" [25].

This standard form of Gherkin can be extended using the keywords 'And' and 'But', which would make our example look something like this: "*Given* that a customer enters their credit card number, *But* it isn't exactly 16 digits long, *When* trying to submit the form, *Then* it should be re-displayed with an error message *And* point them to the correct number of digits".

3.5 Ambiguity

As we have seen in the previous sections, most software requirements are written in natural language. This has also been shown in previous research on documents for requirement analysis by Mich, Franch, and Novi Inverardi [26], giving the following statistics:

- 71.8 % of these documents were in *common* natural language.
- 15.9 % of these documents were in *structured* natural language.
- 5.3 % of these documents were written in *formalized* language.

This much natural language is a problem, since it introduces *unintended ambiguity.* As defined by Berry et al. [27], unrecognized or *unconscious* disambiguation is that process by which a reader, totally oblivious to other meanings of some text that he has read, understands the first meaning that comes to mind and takes it as the only meaning of the text. That unconsciously assumed meaning may entirely be wrong. This loosely translates to there being multiple interpretations of a software requirements specification (SRS) document, possibly resulting in different implementations. There exist different kinds of ambiguity, as investigated in different fields, under which linguistics [28], computational linguistics [29, 30] and philosophy [31]. These different kinds of ambiguity as defined by Berry et al. [27] are:

- *Lexical ambiguity*
  This kind of ambiguity occurs when a word has several meanings. It can be subdivided into two types of ambiguity:
  - *Homonymy*
    Two words are written the same way and pronounced the same way, but have unrelated meanings and different etymologies. Here etymology means the history of development. An example could be the word bank, it can refer to the establishment for custody, loans, exchange or issue of money. Another interpretation of the word bank could be the rising ground before a lake, river or sea.

  - *Polysemy*
    A word has several meanings, but only one etymology. An example of this is the word green. This has one etymology, but can mean the colour green, something that is not ripened or matured, or the way power is produced.

- *Syntactic ambiguity*
  Syntactic ambiguity, or otherwise known as structural ambiguity, occurs when a given sequence of words can be assigned to more than one grammatical structure, where each structure has a different meaning. There are four forms of syntactic ambiguity:

  - *Analytical ambiguity*
    The role of the constituents within a certain phrase are ambiguous. An example of this type of ambiguity can be found in the sentence "The Dutch history teacher". This can be read as "The (Dutch history) teacher" or "The Dutch (history teacher)". The first means that the teacher teaches Dutch history, while the second means that this is a Dutch teacher that teaches history.
  - *Attachment ambiguity*
    This type of ambiguity occurs when a particular syntactic constituent of a sentence, can be legally attached to two parts of a sentence. For instance, "The man shouted at the boy with a megaphone". This phrase could either mean that the man shouted at the boy using a megaphone, or that the man shouted at a certain boy who had a megaphone on him.
  - *Coordination ambiguity*
    Coordination ambiguity occurs when either

    1. more than one conjunction ('and' or 'or') is used in a sentence.
    2. one conjunction is used with a modifier.

    This first type of coordination ambiguity can be seen in the following sentence: "I saw Peter and Paul and Mary saw me". This could mean that I saw Peter, while Paul and Mary saw me. On the other hand, it could also mean that I saw both Peter and Paul, while Mary saw me.

    An example of this second type would be "Your work will be tracked and saved automatically". Does this mean that your work will be automatically tracked and automatically saved, or does it mean that your work will be tracked (manually) and automatically saved?
  - *Elliptical ambiguity*
    An *ellipsis* is a gap in a sentence caused by omission of a lexically or syntactically necessary constituent. Elliptical ambiguity happens when it is not certain whether or not a sentence contains an ellipsis. Shown by Hakobyan [32] is that having an ellipsis in a sentence does not always mean ambiguity, but it can happen. An example often used for explaining elliptical ambiguity is the following sentence: "Perot knows a man richer than Trump". This could mean, assuming there is no ellipsis, that Perot knows a man who is richer than Trump. On the other hand, assuming there is an ellipsis, that implies a missing 'knows' coming after Trump, we could interpret the sentence to mean that Perot knows a man who is richer than any man Trump knows.

- *Semantic ambiguity*
  Semantic ambiguity occurs when a sentence has more than one way of reading it within its context. According to Berry et al. [27] there are three causes for semantic ambiguity:

  1. Coordination ambiguity
  2. Scope ambiguity
  3. Referential ambiguity

  Furthermore, there is one main form of Semantic ambiguity:

  - *Scope ambiguity*
    Scope ambiguity occurs when quantifier operators (such as every, all and each) and negation operators can enter into different scoping relations with other sentence constituents. This can be seen in the following example: "All developers prefer a workplace". The quantifiers 'all' and 'a'

can interact in two ways. When the scope of ‘a’ includes the scope of ‘all’, it would mean that all developers prefer the same one workplace. On the other hand when the scope of ‘all’ includes the scope of ‘a’ it would mean that each developer prefers a, perhaps different, workplace.

- *Pragmatic ambiguity*
  Pragmatic ambiguity is a type of ambiguity that occurs when a sentence has several meanings in the context in which it is situated. This context can be
  - The sentences before and after, or
  - The situation, background knowledge or expectations from one of the two parties involved in the conversation

  There are two distinct types of pragmatic ambiguity:
  - *Referential ambiguity*
    An *anaphor* is an element of a sentence that depends for its reference on the reference of another element, possibly a part of another sentence. When such an *anaphor* can take its reference from more than one element, that can each play the role of the antecedent, we speak of so called referential ambiguity. An example can be the following: “The police will arrest the thieves before they go home”. This could either mean that the police will arrest the thieves, before the thieves go home or before the police go home. Here ‘they’ is the anaphor and it could refer to either the thieves or the police.
  - *Deictic ambiguity*
    Deictic ambiguity occurs when pronouns, time and adverbs (such as now and here) and other grammatical features have more than one point of reference in the context. This can be seen in the following conversation between Peter and Hank. Hank says “Are you coming here this weekend?” where Peter replied with “No, I thought you were coming here!” Anderson [33]. In this example it is unclear where ‘here’ refers to; it could either refer to the place Hank is thinking about or the place Peter is thinking about.

Two other concepts that are closely related to ambiguity are *vagueness* and *generality*. In the sentence “Sue is visiting her cousin”, its is *general* with respect to gender, since cousin could either refer to a male or a female. Furthermore, a sentence using the word ‘tall’ can be considered vague, since we have no clear definition of tall.

Finally, Berry et al. [27] has experienced an extra category of pragmatic ambiguity; *language error*. This type of ambiguity occurs when grammatical, punctuation, word choice or other mistakes in the language of discourse lead to text that is interpreted by a receiver as having a meaning other than that intended by the sender. An example of this can be seen in the sentence “Every light has their switch”. Here ‘their’ is plural while it should be singular, thus one could interpret this sentence as “Every light has its switch” or “All lights share their switch” if the error was made in the verb.

3.6 Natural Language Processing

The importance of natural language for requirements engineering has been known for a long time [34, 35]. As noted in Section 3.5, most documents based on requirements are written in natural language. Next to that, we have seen in Section 3.2 that interviews are a communicative event, thus also based on natural language. Abbott and Moorhead [36] found that “the best language for requirements is natural language”, explaining that often the issue at hand is too hard to be formalized, thus leaving natural language as the better option to describe that issue. While natural language is easy to write and comprehend, as we saw in Section 3.5, it is inherently ambiguous [27] and large collections of natural language requirements are hard to examine manually to obtain an overview, find inconsistencies, duplicates or missing requirements [37].

Inspired by the close relationship between natural language and requirements, there have been many attempts in developing natural language processing tools and methods for processing requirements texts since the early nineties [35]. The research in Natural Language Processing for Requirements Engineering or NLP4RE in short, has grown into an active research area [38], resulting in annual workshops such as

NLP4RE [37], NLPaSE [39] and NLP-SEA [40]. Recent developments in this research area, as mapped by Zhao et al. [8], include tools for requirements classification [41], detection of requirement defects [42], smells [43] and equivalence [44], glossary term extraction [45], requirements tracing [46], extraction of conceptual models from user stories [47], analyzing requirements-relevant legal texts [48], demarcation of requirements [49], and concept extraction [50].

Natural language processing (NLP) is a field that employs computational techniques to learn, understand and produce human language content [7]. A definition by Liddy [51] is the following:

Natural Language Processing is a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of *linguistic analysis* for the propose of achieving human-like language processing for a range of tasks or applications [51]

In this definition, the levels of *linguistic analysis* refer to *phonetic, lexical, syntactic, discourse* and *pragmatic analysis* of language. It is assumed that humans use all these levels of linguistic analysis to produce or comprehend language [52].

The approaches to NLP can be broadly categorized into *symbolic NLP*, *statistical NLP* and *connectionist NLP* [51].

- *Symbolic NLP*
  This category of NLP goes into performing a deep analysis of linguistic phenomena. This analysis is based on explicit representations of facts about language [51]. While this work very well in rule-based systems or semantic networks by using facts and production rules, it does come with some drawbacks. First off, it lacks flexibility to adapt to new language phenomena. Next to that, the amount of rules may become too big to be able to manage [53]. Finally the symbolic NLP approach may be fail when they are represented with unrecognized grammatical input. [51].

- *Statistical NLP*
  Statistical NLP approaches employs various machine learning methods and large quantities of linguistic data to create approximate and probabilistic models of language [8]. These statistical models are fairly simple, but still robust, because they are based on examples from the linguistic data, rather than analyzed language phenomena as in symbolic NLP. However, some drawbacks for statistical NLP include that the models can also degrade with unfamiliar inputs or inputs with a lot of errors [53]. Next to that, statistical NLP is mostly used for low-level NLP tasks, such as parsing or POS tagging [51].

- *Connectionist NLP*
  Like statistical NLP approaches, connectionist NLP approaches also develop generalized models of linguistic phenomena [51]. What seperates this category of NLP is the usage of various representations, allowing transformations, inferences and manipulations of logic formulas. This is where the usage of neural networks lie, leading to high-level, but less observable models.

Natural Language Processing for Requirements Engineering (NLP4RE) is defined by Zhao et al. [8] as follows:

Natural language processing for requirements engineering is an area of research and development that seeks to apply NLP technologies to different types of requirements documents to support a range of linguistic analysis tasks performed at various RE phases [8]

Where the *NLP techonologies* in this definition can refer to:

- *NLP techniques*
  NLP techniques are practical method, approach, process or procedure for performing a certain NLP task [8]. Examples of this are POS tagging, parsing or tokenizing.

- *NLP tools*
  NLP tools are software systems or libraries that support one or more NLP techniques [8]. Examples are Spacy[2], Stanford CoreNLP[3], NLTK[4] or OpenNLP[5].

[2]https://spacy.io
[3]https://stanfordnlp.github.io/CoreNLP/
[4]https://www.nltk.org
[5]https://opennlp.apache.org

- *NLP resources*
  An NLP resource is a linguistic data resource for supporting NLP techniques or tools (Zhao et al. [8]). This can be a *lexicon* (i.e. dictionary) or a *corpus* (i.e. a collection of texts). Existing lexicons include WordNet[6] and FrameNet[7], where examples of a corpus are the Brown Corpus[8] or the British National Corpus[9]

Not all of these NLP technologies will be useful for every purpose. For our purpose, we will go over a few NLP techniques as described by Jurafsky and Martin [54].

3.6.1 Part of Speech (POS) tagging

Part of Speech tagging is a sequence labeling task that assigns a part-of-speech to each word in a text. Given a certain input sequence $x_1, x_2, \ldots, x_n$ of words and a tagset, the output is a sequence $y_1, y_2, \ldots, y_n$. Each output y_i corresponds to its input x_i, giving a tag to that word as show in figure 3.7.

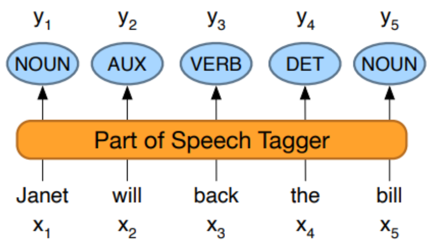
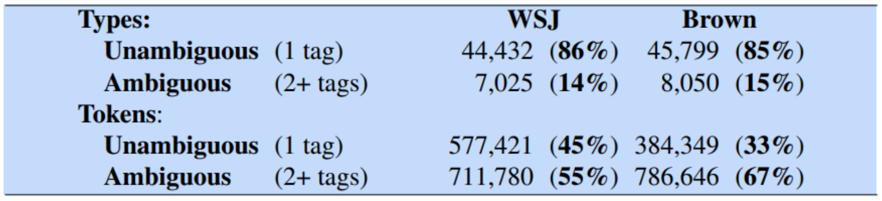


Figure 3.7: Mapping from inputs $x_1, x_2, \ldots, x_n$ to output POS tags $y_1, y_2, \ldots, y_n$ [54]

It is important to note that tagging is a disambiguation task; words are ambigious and therefore have more than one possible part-of-speech as we can see in figure 3.8. The goal here is to find the correct tag for this situation. The accuracy for part-of-speech tagging algorithms are very high.



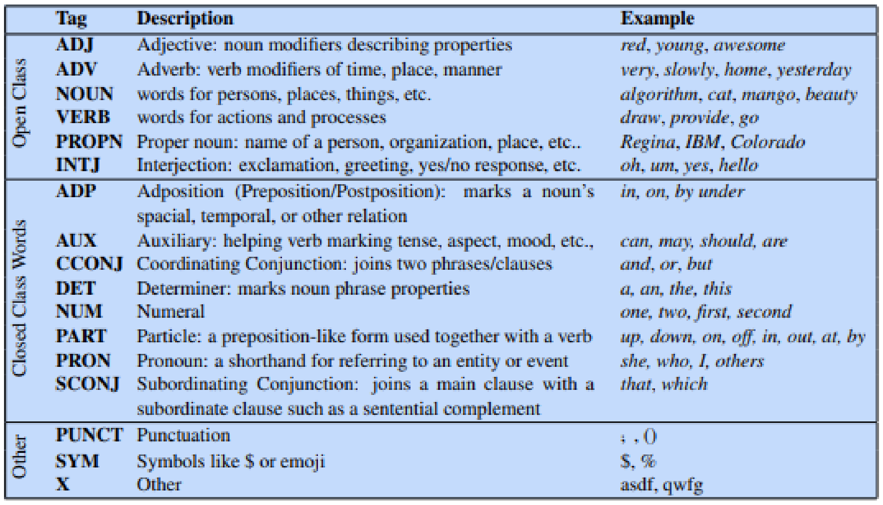
		WSJ		Brown	
Types:					
Unambiguous	(1 tag)	44,432	(**86%**)	45,799	(**85%**)
Ambiguous	(2+ tags)	7,025	(**14%**)	8,050	(**15%**)
Tokens:					
Unambiguous	(1 tag)	577,421	(**45%**)	384,349	(**33%**)
Ambiguous	(2+ tags)	711,780	(**55%**)	786,646	(**67%**)

Figure 3.8: Tag ambiguity in two different corpora [54]

These parts of speech fall into two categories: *open class* and *closed class*. Closed class parts of speech generally stay the same and rarely new closed class parts of speech get created. These closed class words are generally *function words*, which occur frequently and give structure to sentences. On the other hand, open class parts of speech are continually being created and borrowed. As we can see in figure 3.9, English has five major open classes, namely *nouns* (including proper nouns), *verbs*, *adjectives*, *adverbs* and a smaller open class of *interjections*. Most languages have four open classes.

- *Nouns*
  Nouns are used to describe people, places or things. We can divide these nouns into *count nouns* and *mass nouns*. Count nouns are things that can be singular or plural. For example ‘speaker’/‘speakers’, ‘interview’/’interviews’. While mass nouns do not have this property and are used to describe something that is conceptualized as a homogeneous group [54]. For example ‘snow’ or ’salt’. Finally, we have *proper nouns*, which describe specific persons or entities such as ‘Utrecht’ or ‘Regina’.

[6]https://wordnet.princeton.edu
[7]http://www.icsi.berkeley.edu/icsi/projects/ai/framenet
[8]http://korpus.uib.no/icame/manuals/
[9]http://www.natcorp.ox.ac.uk



	Tag	Description	Example
Open Class	**ADJ**	Adjective: noun modifiers describing properties	*red, young, awesome*
	ADV	Adverb: verb modifiers of time, place, manner	*very, slowly, home, yesterday*
	NOUN	words for persons, places, things, etc.	*algorithm, cat, mango, beauty*
	VERB	words for actions and processes	*draw, provide, go*
	PROPN	Proper noun: name of a person, organization, place, etc..	*Regina, IBM, Colorado*
	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	*oh, um, yes, hello*
Closed Class Words	**ADP**	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	*in, on, by under*
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	*can, may, should, are*
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	*and, or, but*
	DET	Determiner: marks noun phrase properties	*a, an, the, this*
	NUM	Numeral	*one, two, first, second*
	PART	Particle: a preposition-like form used together with a verb	*up, down, on, off, in, out, at, by*
	PRON	Pronoun: a shorthand for referring to an entity or event	*she, who, I, others*
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	*that, which*
Other	**PUNCT**	Punctuation	; , ()
	SYM	Symbols like $ or emoji	$, %
	X	Other	asdf, qwfg

Figure 3.9: The 17 parts of speech in the Universal Dependencies tagset (Nivre et al. [55]), from Jurafsky and Martin [54]

- *Verbs*
  The *verbs* are actions or processes, for example 'walk', 'jump' and 'run'. In English there are certain inflections for a word, for the word 'walk' this would be 'walk'-'walks'-'walking'-'walked'.

- *Adjectives*
  To describe properties or qualities of our nouns, we use *adjectives*, such as age ('young', 'old') and price ('cheap', 'expensive').

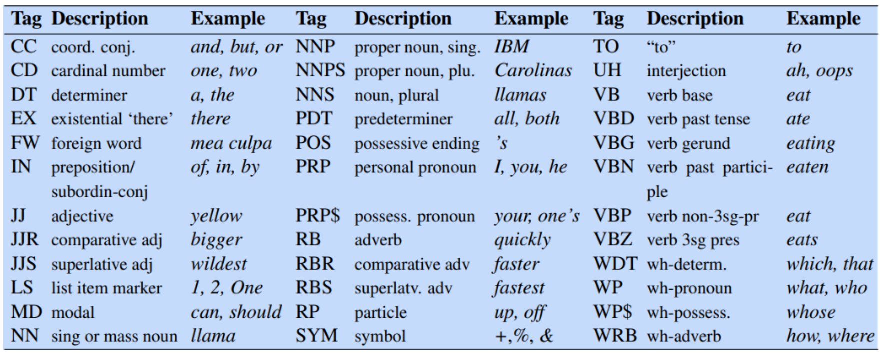
- *Adverbs*
  The usage of *adverbs* is very broad, but generally they modify something. According to Jurafsky and Martin [54] there are a lot of adverb types we can define. We can distinct *directional* or *locative adverbs*, that specify the direction or location of some action, such as 'here', 'there' and 'home'. On the other hand we have *degree adverbs* such as 'extremely', 'very' or 'somewhat'. Degree adverbs specify the extent of some action, process or property. To describe the manner of some action, *manner adverbs* are used, like 'slowly', 'delicately' or 'sneakily'. Finally, to tell when some action or event took place, we can use *temporal adverbs* such as 'yesterday', 'Monday' or 'tomorrow'.

- *Interjections*
  The *interjections* are small sudden utterances ('oh', 'um'), including greetings ('hey', 'bye') and question responses ('yes', 'no').

A another tagset that is more English-specific is the Penn Treebank tagset [56], which has been used to make many different annotated corpora. The part-of-speech tags can be seen in figure 3.10. Note that this tagset is more specific when it comes to tense and participles on verbs.

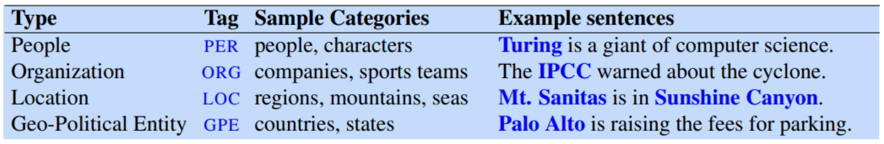
3.6.2 Named Entity Recognition (NER)

As we have seen in Section 3.6.1, POS tagging can recognize proper nouns. These proper nouns can be different kinds of entities, for example 'Utrecht' is a location while 'Regina' could be a person. Therefore, we have *named entities*, which are things that can be referred to with a proper name: a location, a person, an organization. The task of named-entity recognition (NER) is to find parts of text that have proper names and tag the correct type of entity [54]. Four of these types of entity tags are most common: *PER* for person, *LOC* for location, *ORG* for organization and *GPE* for a geo-political entity. There are many more named entities, but these main types can be seen in figure 3.11.



Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	*and, but, or*	NNP	proper noun, sing.	*IBM*	TO	"to"	*to*
CD	cardinal number	*one, two*	NNPS	proper noun, plu.	*Carolinas*	UH	interjection	*ah, oops*
DT	determiner	*a, the*	NNS	noun, plural	*llamas*	VB	verb base	*eat*
EX	existential 'there'	*there*	PDT	predeterminer	*all, both*	VBD	verb past tense	*ate*
FW	foreign word	*mea culpa*	POS	possessive ending	*'s*	VBG	verb gerund	*eating*
IN	preposition/ subordin-conj	*of, in, by*	PRP	personal pronoun	*I, you, he*	VBN	verb past participle	*eaten*
JJ	adjective	*yellow*	PRP$	possess. pronoun	*your, one's*	VBP	verb non-3sg-pr	*eat*
JJR	comparative adj	*bigger*	RB	adverb	*quickly*	VBZ	verb 3sg pres	*eats*
JJS	superlative adj	*wildest*	RBR	comparative adv	*faster*	WDT	wh-determ.	*which, that*
LS	list item marker	*1, 2, One*	RBS	superlatv. adv	*fastest*	WP	wh-pronoun	*what, who*
MD	modal	*can, should*	RP	particle	*up, off*	WP$	wh-possess.	*whose*
NN	sing or mass noun	*llama*	SYM	symbol	+,%, &	WRB	wh-adverb	*how, where*

Figure 3.10: Penn Treebank part-of-speech tags (Marcus, Santorini, and Marcinkiewicz [56]), from Jurafsky and Martin [54]



Type	Tag	Sample Categories	Example sentences
People	PER	people, characters	**Turing** is a giant of computer science.
Organization	ORG	companies, sports teams	The **IPCC** warned about the cyclone.
Location	LOC	regions, mountains, seas	**Mt. Sanitas** is in **Sunshine Canyon**.
Geo-Political Entity	GPE	countries, states	**Palo Alto** is raising the fees for parking.

Figure 3.11: The four most common named entity types, from Jurafsky and Martin [54]

3.7 Machine Learning

Machine Learning aims to allow computer programs that can "learn" from input available to them. As said by Shalev-Shwartz and Ben-David [57], learning is the process of converting experience into expertise or knowledge. The input of such a learning algorithm is training data, representing experience. The output is some expertise.

A useful definition for machine learning, by [58], is:

A computer program is said to *learn* from experience **E** with respect to some class of task **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improved with experience **E**.

An example if this would be:

- Task **T**:
  Highlighting requirements relevant information from transcripts.
- Performance **P**:
  The fraction of correctly highlighted information.
- Experience **E**:
  A set of labelled transcript sections (relevant or not relevant).

Often, we turn to machine learning for tasks are that difficult to write a provably correct algorithm for. By allowing our program to learn from experience, we can achieve satisfactory results once trained on enough examples. This could also mean going beyond human capabilities, such as predicting the weather or detecting meaningful patterns in data. There are three main categories of Machine Learning, namely:

1. Supervised learning
   Supervised learning starts with reading in training data and computing a learned function, after which this can be used to predict other input [59]. An example of this could be learning to classify e-mails as spam. By learning on a training set of e-mails that are already classified as spam or not-spam, the learner can effectively figure out whether a new e-mail is spam or not.
2. Unsupervised learning
   Unsupervised learning can be done with a set of statistical tools, discovering things about the measurements on our data and using this to predict our desired output [60]. This could be used to detect

"unusual" messages within e-mails, without training on a set of labeled e-mails but by training on a large body of e-mail messages.

3. Reinforcement learning
   This type of machine learning deals with learning from interaction with an environment to maximize a long-term objective [61]. This is mainly used in games such as chess, by giving a reward to the moves played.

Next to that, we have some types of Machine Learning that do not fall within the bounds of these categories:

- Semi-Supervised learning
  Semi-supervised learning is a mix between supervised and unsupervised learning. Here, next to unlabelled data, there is some supervision information provided, but not for all examples [62].
- Self-Supervised learning
  In self-supervised learning, the training data is automatically labeled by leveraging the relations between different input sensor signals [63]. Compared to unsupervised learning, we do not detect specific patterns but aims to recover. Figure 3.12 tries to explain this, comparing it to supervised and unsupervised learning.

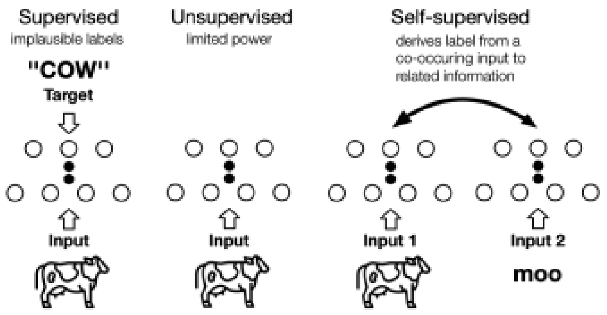


Figure 3.12: The difference between supervised, unsupervised and self-supervised learning [63].

A specific type of machine learning that is often mentioned is *deep learning*. Deep learning effectively represents the world as a nested hierarchy of concepts, where each concept is defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract representations [64]. To show the difference of this type of machine learning, figure 3.13 shows a high-level schematic of how different types of AI systems work compared to deep learning. More specifically, when looking at artificial neural networks, a deep neural network is an artificial neural network with two or more layers. We speak of a neural network that is two or more layers deep (hence the name "deep" learning).

3.7.1 Model selection

When selecting a model, we are interested in how well it *generalizes*, in other words, how does it perform on data that it has not seen before. This comes with the notions of *underfitting* and *overfitting*. For this we will use the definitions given by Daumé [59]. Underfitting supposes the model had an opportunity to learn something, but it did not; the model is too simple. On the other hand, overfitting is when you pay too much attention to the idiosyncrasies of the training data; the model is too detailed.

After checking how well a model generalizes, we need to evaluate how well it performs compared to other models. A commonly used evaluation metric is accuracy; the fraction of correct answers of the total answers. While this may seem like a very good metric, it is not suitable when the label distributions are skewed. Therefore, when evaluating machine learning models, *precision* and *recall* are more appropriate metrics.

In order to talk about the precision and recall, we must first introduce the notion of a *confusion matrix*. A confusion matrix compares the output of a model, or the predicted truth, to the actual truth. Therefore

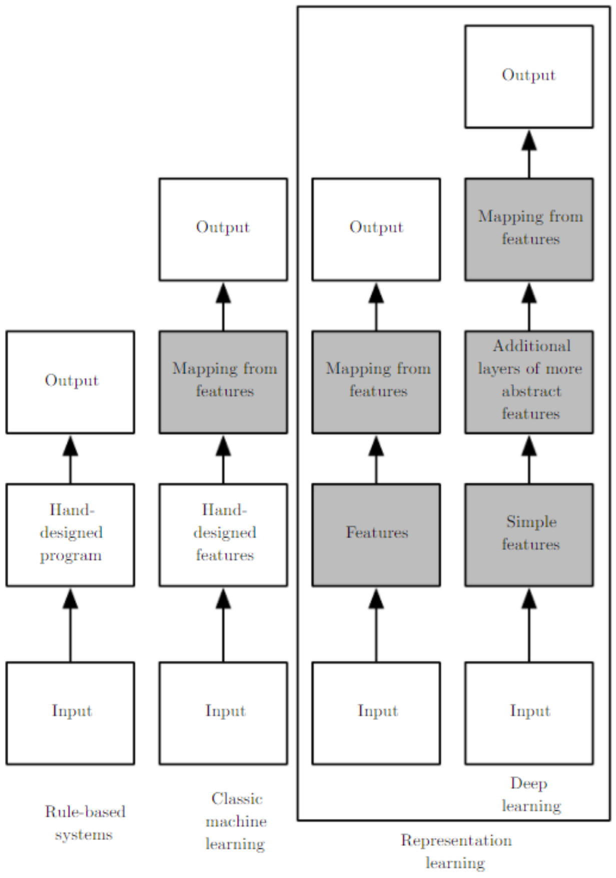


Figure 3.13: A flowchart of how different parts of an AI system relate to each other in different AI disciplines. The gray boxes indicate components that are able to learn from data. [64]

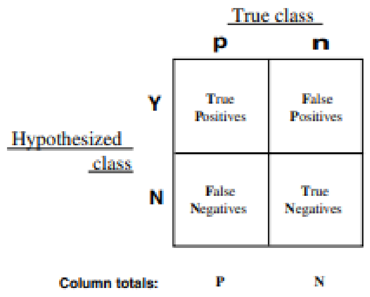


Figure 3.14: A confusion matrix [65]

not simply checking if the answer is correct, but which prediction it made to be correct or be incorrect. An theoretical example of such a confusion matrix can be seen in figure 3.14.

In this case, there are two predictions: something is positive or something is negative. Therefore, when a prediction of hypothesized class is positive while the actual truth is positive, we count this as a *true positive*. On the other hand, when it would have been hypothesized to be negative, this would be a *false negative*. Likewise, when the truth is negative, but the prediction is positive, we count it as a *false positive*. On the other hand, when the prediction would have been negative, this is seen as a *true negative*.

Precision describes what fraction of the ones we have identified to belong to a class, actually belong to that class. For example, what fraction of messages labeled as spam was actually spam. This is calculated as follows:

$$\textbf{Precision} = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

On the other hand, *recall* describes what fraction of the ones that belong to a certain class, has our model identified. In the same example, this would be checking what fraction of the messages that are actually spam have we identified and labeled as spam. The calculation for this would be the following:

$$\textbf{Recall} = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Often, these two metrics are combined into the F_1 score, which can be computed in the following fashion:

$$F_1 = \frac{2}{\frac{1}{\textbf{Recall}} + \frac{1}{\textbf{Precision}}}$$

3.7.2 Transfer Learning

The effectiveness of supervised learning breaks down when we do not have sufficient labeled data to train and validate on [14, 38, 8]. Transfer learning allows us to deal with this by using already existing labeled data of some related task or domain [66]. By storing the knowledge gained in solving the source task in the source domain, we can apply it to our problem of interest, on our target task. Howard and Ruder [67] have shown that with transfer learning, but using 100 times less data to fine-tune on, it still matches the performance of a model trained from scratch.

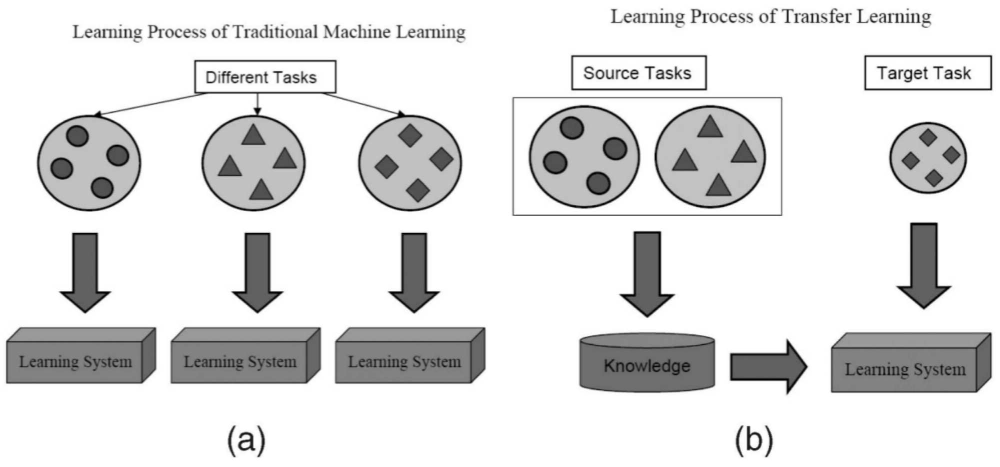


Figure 3.15: Difference in learning process between (a) traditional machine learning and (b) transfer learning. here it is clear that transfer learning aims to extract the knowledge from one or more source tasks and applies the knowledge to a target task [68]

Usage of transfer learning can be seen in different domains. For example, an article by Kermany et al. [69] shows the implementation of transfer learning to support clinical decisions with clinical image classification. By using a convolutional neural network trained on the ImageNet image database [70], which is proven to be very effective [71], this knowledge could be transferred to the training on the clinical image dataset. Figure 3.16 shows the schematic of how this is done.

3.7.3 Zero-shot learning

Compared to Transfer Learning, Zero-shot learning (ZSL) improves the idea of training with little labeled data, by reasoning only on the embedding of sentences and tags, thus requiring no labeled data. Originally, ZSL was used in image processing to predict unseen images [72]. After that it was adapted to the use of text classification to predict unseen classes, by Pushp and Srivastava [73]. These models predict whether a sentence is related to a certain tag or not. It sees this problem as finding relatedness between sentences and

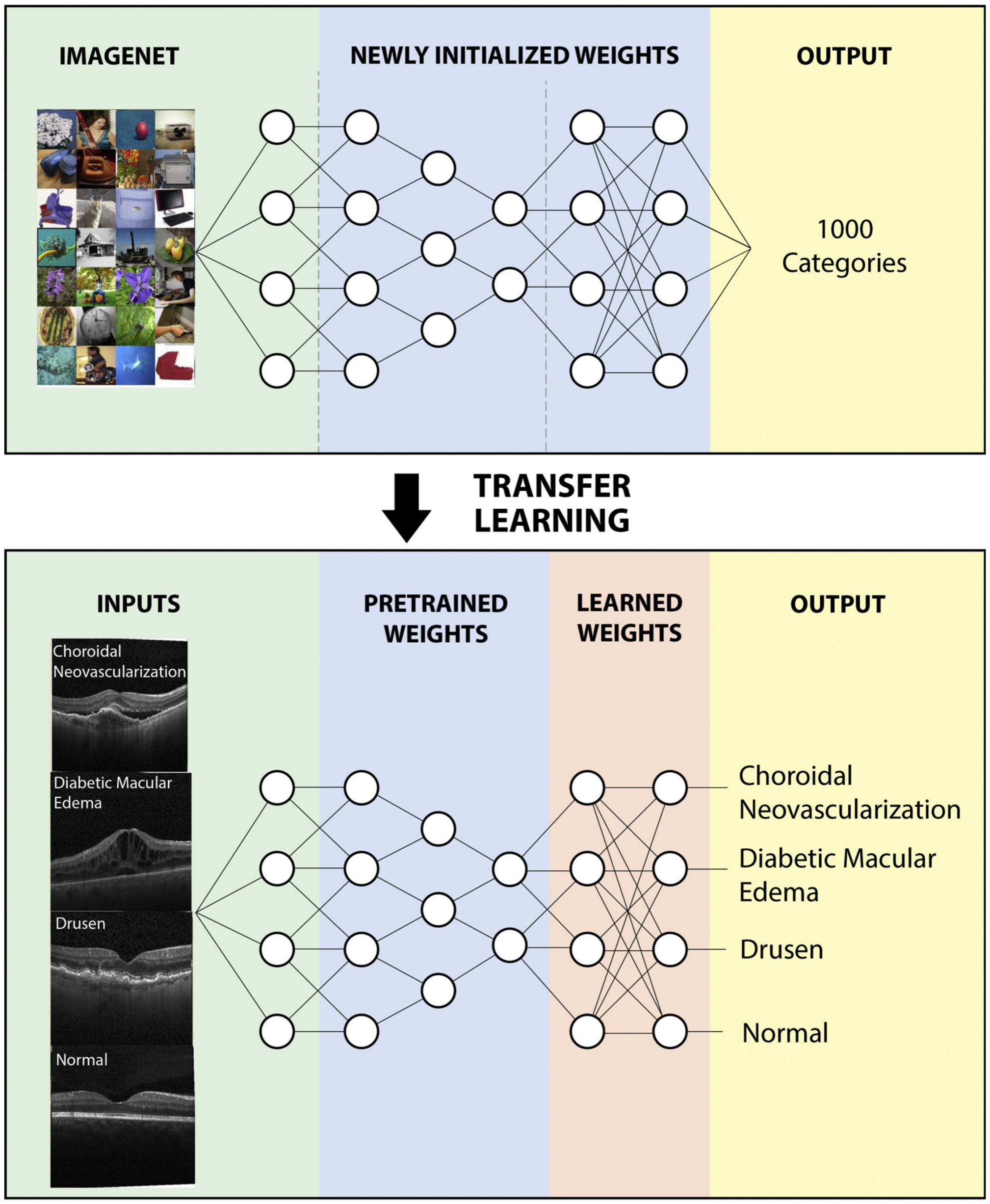


Figure 3.16: Schematic of the solution for classifying clinical images by Kermany et al. [69]

classes.
There are two main methods for training a ZSL model:

- *embedding-based method*
  This method integrates two layers: the text embedding layer and the tag embedding layer. After that it measures the probabilities for their relatedness, using a similarity function [73].

- *entailment-based method*
  Here, the input text sequence is treated as a premise and the candidate tags as an hypothesis. By inferring if the input text is an entailment of any of the tags, gives the output whether the tag fits with the sequence or not [74].

By using large pre-trained language models such as BERT, a ZSL model can perform NLP tasks without fine-tuning (zero-shot) or with only some labelled examples (few-shot). This similarly tackles the issue of the expensive need for data-labelling [14, 38, 8].

3.7.4 Ensemble learning

Ensemble learning is a term for methods that combine multiple models, so that the errors of a single model will likely be compensated by others, so that the overall prediction performance of the ensemble would be better [75]. There are several factors that the usage of ensemble methods could improve [76, 77]:

- *Avoidance of overfitting*
  Taking the average of different hypothesis reduces the risk of choosing a wrong hypothesis, therefore avoiding the overfitting.
- *Computational*
  A single model could get stuck in a local optimum, but by combining different different models this can be overcome.
- *Representational*
  By combining different models, the search space may be extended to better fit to the data space.

To families of ensemble methods can be distinguished [78]:

- *Averaging methods*
  Here, several models are built build independently after which their predictions are averaged.
- *Boosting methods*
  In boosting methods, the models are built sequentially and one tries to reduce the bias of the combined model.

4 Related work

When it comes to the related works on state-of-the-art techniques for identifying requirements-relevant information in conversations, we can differentiate between two types of works. In Section 4.1, we discuss research efforts that provide a concept that is relevant to our research. These works provide a concept or concept tool that might show us a possible direction to take in our approach. Furthermore, in Section 4.2 we go through different related works that show techniques that might prove useful in our use case. This can also be the challenges that were faced in these works, or limitations in their approach.

4.1 Conceptual works

For the related works that discuss a concept or concept tool that is relevant to our research, we discuss the works of Spijkman, Dalpiaz, and Brinkkemper [79] and Spijkman et al. [50]. In Section 4.1.1, we look at a work that discusses a Fit-Gap analysis, done on transcripts of elicitation sessions. Next, in Section 4.1.2 we follow the design of a Natural Language Processing concept tool that extracts the known and unknown concepts using a domain ontology.

4.1.1 Fit-Gap Analysis

As we have seen in Section 3.1, most requirements elicitation is done through conversational scenarios. Which allows for opportunities in *speech-driven RE*: the analysis of conversations aimed at detecting and extracting requirements-relevant information [79]. One of the opportunities is the so called *Fit-gap analysis*, which is a commonly used elicitation method, mainly used for enterprise applications [80, 81, 82, 83, 84]. *Fit-gap analysis* (FGA) compares the abilities of a software product and certain characteristics of the target organization [82].

When software products get extra requirements or requirements change, this can often lead to customization of the product. In order to be more specific, we define *customization*, based on the work of Light [84] as:

Any customer specific change or addition to the functionality available in the the standard software product.

A method for managing this mass customization is the creation of a product line, which consists of a set of products that share similarities and are created from different reusable parts [85]. On the other hand, many products can be adjust to the specific needs of a customer through configuration. To make a distinction between customization and *configuration*, we use the following definition for *configuration* from Apel et al. [85]:

Set-up of a software product concerning a predefined set of options used to tailor the software to the customer.

Fit-gap analysis as defined by Spijkman, Dalpiaz, and Brinkkemper [79] is the following:

A requirements elicitation technique that based on a customer's needs with the functionality of a software product, identifies needs that are supported by the current functionality as fits, and needs that are not as gaps.

As the name suggests, the outputs from a fit-gap analysis can be a *fit* between the customer needs and the software products, or a *gap* in the functionality required by the customer [79]. Here the fits can be a part of the functionality that is already implemented or some configuration. The gaps on the other hand indicate that some customization has to be done. Figure 4.17 visualizes this input, process and output.

The concept of Fit-gap analysis could be very useful for recognizing important configuration or customization requirements. However, this does requirement some form of vocabulary or ontology of the software product. This is not always available, but when this is available the usage of Fit-gap analysis could be very useful.

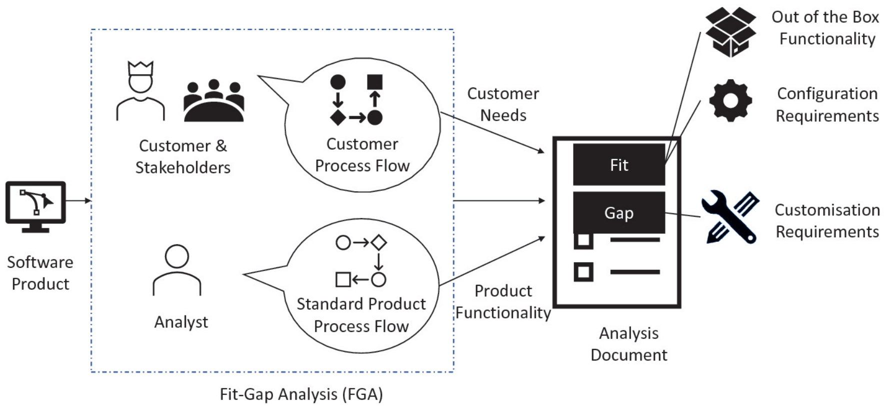


Figure 4.17: Fit-gap analysis in the context of requirements elicitation for software products [79]

4.1.2 Concept Extraction

As we noted in Section 3.6, a recent development in the usage of natural language processing in requirements engineering is concept extraction [50]. The aim of Spijkman et al. [50] is to discuss and design an aid for RE practitioners, that extracts key concepts from RE transcripts and compares them to a *software product ontology*. The definition for software product ontology by Spijkman et al. [50] is as follows:

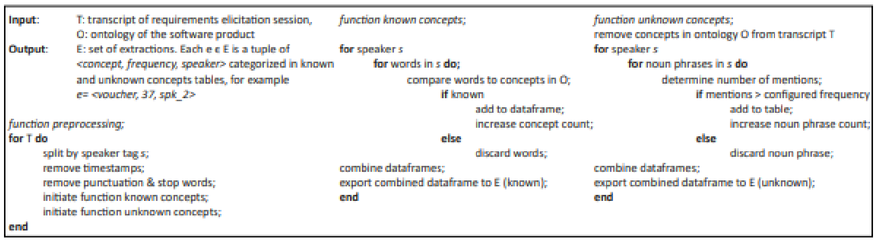
A set of concepts and relationships in a software product domain, which describe functionalities, artifacts and related software systems

They designed a prototype key abstraction extraction tool [86], aiming to detect both unknown and known concepts in a requirements elicitation session. Here, unknown concepts will indicate the need for customization of the product (with customization as defined in Section 4.1.1 by Light [84]). On the other hand, known concepts will show the need for configurations of existing features of the product (as defined in Section 4.1.1 by Apel et al. [85]). For example:

"Our company provides shipping of goods on multiple types of routes. We utilize our trucks for national shipping, and our *Cargo ships* to ship goods internationally. For all our shipping we need to ensure efficient **fuel** use." [50]

Here the known concept, highlighted in bold, *fuel*, indicates that efficiency as a key factor of the configuration. While *Cargo ships*, the unknown concept, could indicate the support for naval route planning as a potential customization.

A description of the early stage[10] of this tool's process in pseudocode is shown in figure 4.18.



```
Input:   T: transcript of requirements elicitation session,
         O: ontology of the software product
Output:  E: set of extractions. Each e ∈ E is a tuple of
         <concept, frequency, speaker> categorized in known
         and unknown concepts tables, for example
         e= <voucher, 37, spk_2>

function preprocessing;
for T do
    split by speaker tag s;
    remove timestamps;
    remove punctuation & stop words;
    initiate function known concepts;
    initiate function unknown concepts;
end

function known concepts;

for speaker s
    for words in s do;
        compare words to concepts in O;
            if known
                add to dataframe;
                increase concept count;
            else
                discard words;
combine dataframes;
export combined dataframe to E (known);
end

function unknown concepts;
remove concepts in ontology O from transcript T
for speaker s
    for noun phrases in s do
        determine number of mentions;
            if mentions > configured frequency
                add to table;
                increase noun phrase count;
            else
                discard noun phrase;
combine dataframes;
export combined dataframe to E (unknown);
end
```

Figure 4.18: Pseudocode of the key abstraction extraction tool prototype [50]

It starts with some preprocessing, splitting the text by speaker, removing timestamps, punctuation and certain stop words. After that it starts with extracting the known and unknown concepts. For the known concepts it is simply iteration through each word in the text matching with the ontology, adding it to the

[10]The work on this tool is still in progress and can be found on `https://bowis.github.io/keyextractor/`

table when there is a match. For the unknown concepts, the noun phrases that are not known concepts are selected and added to the table when occurring more than a certain frequency. At the end, two tables are generated; one with known concepts and one with (potentially) unknown concepts.

What this work shows us, is that with the use of an ontology and relatively easy usage of NLP tooling can result in a good tool. Even though our tool has to be a lot more complex, we can still use this as a part of our tooling.

4.2 Related techniques

When it comes to techniques related to our thesis project, we first look at a prototype tool by Abualhaija et al. [49] in Section 4.2.1, that demarcates requirements from free-form text. Then, we walk through different approaches to categorize requirements [87, 88, 89, 90, 91, 92] in Section 4.2.2. Finally, in Section 4.2.3 we look at related work that shows how to extract declarative process models from natural language [93]. All of these works used different NLP techniques, showing how they can be utilized and what works well. Furthermore, they reveal challenges that might reoccur in our project.

4.2.1 Automated demarcation

Another prototype tool, developed by Abualhaija et al. [49], aims to demarcate requirements in free-form requirements specifications. By using machine learning, this prototype is flexible to be used in different domains and used with different writing styles. By training and evaluating on labeled datasets, their average precision and recall are very encouraging.

The tool was made using supervised machine learning (as explained in Section 3.7) and recognizing the problem of distinguishing between requirements and non-requirements as a binary classification problem. While most machine learning classification algorithms attempt to minimize misclassification, the rationale here is that, as long as false positives are not too many, the effort of manually discarding them is a compelling trade-off for a better recall.

By using three main NLP technologies [49], the parsing is done:

- *Constituency parsing*
  Delineating the structural units of sentences, mostly Noun Phrases (NPs) and Verb Phrases (VPs).
- *Dependency parsing*
  To infer grammatical dependencies between the words in sentences.
- *Semantic parsing*
  To get a representation of the meaning of the sentence based on the meaning of the sentence's constituents.

For the preprocessing, three modules are used. The first being the *Tokenizer*, splitting the input text into different tokens. The next is the *Sentence Splitter*, which splits the text into sentences based on the delimiters. The third and final module is the *POS tagger*, which assigns a POS tag to each token (as described in Section 3.6.1.

This whole NLP process can be seen in figure 4.19, as part of the whole process seen in figure 4.20. After the completion of the NLP process, frequency-related metadata is computed based on:

- The most frequent *modal verb* in the text. A *modal verb* is, according to the Cambridge Academic Content Dictionary, a verb that is used with another verb to express an idea such as possibility that is not expressed by the main verb.
- The top 1% of the most frequent noun phrases.
- Frequency levels for the different identifier patterns used within the text.

The second phase is building the feature matrix for training our model and classifying our input. In order to keep these features generic, Abualhaija et al. [49] oriented their features around structural and semantic properties. By following the practices and of designing linguistic and stylistic features in NLP applications, as described by Stamatatos [94], the features were designed in an iterative manner. This resulted in a table of 20 features, split into four categories:

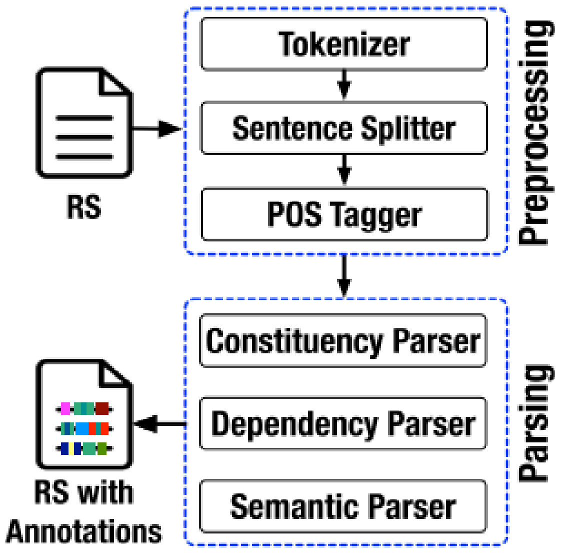


Figure 4.19: The NLP pipeline of the tool by Abualhaija et al. [49]

- *Token-based features*
  The six features are based on the token-level information, such as the amount of tokens, number of alphabetic words or the number of one-character tokens.

- *Syntactic features*
  The syntactic features are a set of eight features that are derived from the syntax-related information, such as POS tags, grammatical dependencies and phrasal structure. Examples of these features are possession of a verb, a modal verb or conditional clauses.

- *Semantic features*
  These three features are derived from semantic categories of the verbs, therefore checking if it contains a cognition verb, action verb or stative verb.

- *Frequency-based features*
  The document-wide frequency metadata, as created in the second phase, are represented in these three features.

The full table can be found in Appendix A in tables 17 and 18

The third phase uses the train machine learning model and predicts for each requirement candidate whether it will be a requirement or non-requirement. These are seen as intermediary results, as some post-processing and refining is needed.

In the fourth and last step a *list detection* refinement is done. This parses the text and looks for any environments containing a header and sub-items. These components of such an environment are considered separately in the approach, therefore in this list detection phase they are considered together. If the header of the list has been marked as a requirement, then the sub-items are also marked as requirements. This can possibly increase the number of false positives, but can reduce the number of false negatives. This is a consideration made by Abualhaija et al. [49], preferring a reduction of false negatives over and increase of false positives.

Relating back to our project, the NLP pipeline seems very interesting. In this project, the usage of a Sentence Splitter, POS tagger and Tokenizer is most likely very useful. Furthermore, the frequency metadata focus on the modal verbs and most frequent noun phrases is something that can be used in the tool for this project. On the other hand, the work by Abualhaija et al. [49] uses supervised learning, while we do not have the option to do this. While this paper goes to talk about candidate requirements, these are not present in the transcripts our tool will work with. Therefore our approach will look very different from what we see here.

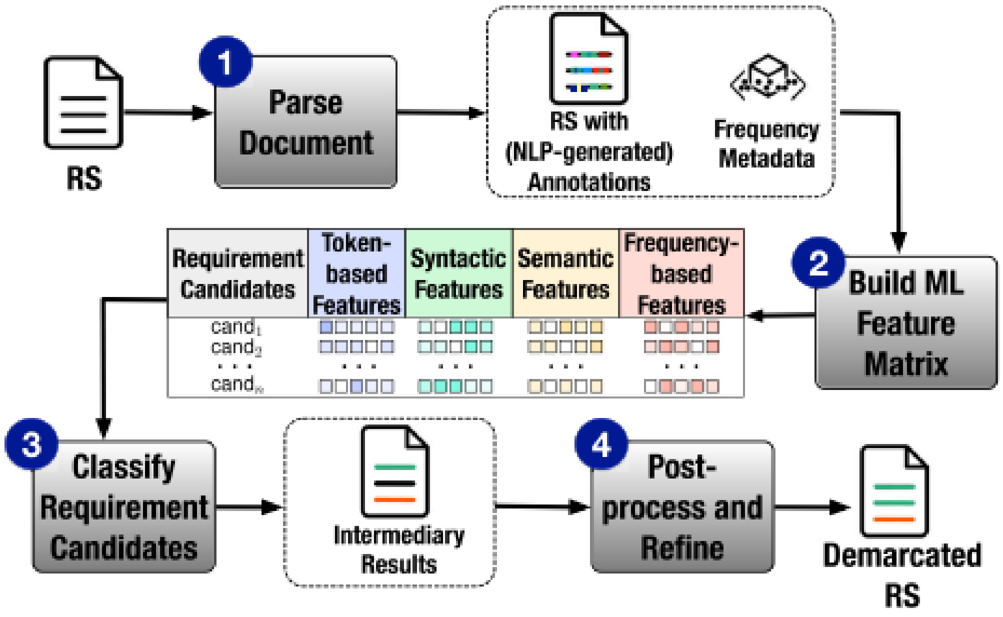


Figure 4.20: Overview of the approach by Abualhaija et al. [49]

4.2.2 Requirement Classification

Many papers in the NLP4RE area of research focus on classification of requirements. One of the RE17 data challenges was the identification of requirement types using the "Quality attributes (NFR)" dataset provided. These papers mainly distinguishing functional and non-functional requirements [87, 88, 89, 90, 91, 92]. This automatic categorization of functional and non-functional requirements is successfully done using Machine Learning techniques, where comparisons between these techniques are made by Dias Canedo and Cordeiro Mendes [91] and Abad et al. [92].

As defined by Van Lamsweerde [1]:

- *functional requirements*
  - Address what services the software-to-be should provide
  - Capture the intended software effects on the environment
- *non-functional requirements* Constrain how the functional requirements should be, given:
  - Quality requirements: safety, security, accuracy, ...
  - Other requirements; compliance, architectural, development, ...

An approach taken by Kurtanović and Maalej [87] is to use supervised machine learning. By using a Support Vector Machine, they were able to classify functional and non-functional requirements with high precision and recall. Furthermore, they experimented with oversampling the dataset to handle the class imbalance within the non-functional requirements. This indeed demonstrated to be a useful approach to handle class imbalances. Next to that, they analyzed the most informative features. This showed that POS tags, word n-grams, modal verbs and POS tag cardinal numbers were very informative. More work on this has been done by Dalpiaz et al. [37], showing that certain higher-level linguistic dependencies can be very useful as well. An important observation by Dalpiaz et al. [37] is that the performance of the classifiers degrade considerably when using other datasets than they were trained on, especially when identifying quality aspects.

Whereas the other approaches focus on one machine learning algorithm to classify the requirements, Dias Canedo and Cordeiro Mendes [91] and Abad et al. [92] compare different machine learning algorithms. These researches concluded with Binarized Naïve Bayes performing the best [92] in one research and Logistic

Regression with Term Frequency–Inverse Document Frequency (TF-IDF) as feature selection [91] performing the best in the other research.

Hey et al. [88] describe the increased usage of transfer learning approaches in natural language processing. They point out that in cases such as requirements engineering, where only a limited amount of (labeled) data exists, this might prove advantageous. This is done by fine-tuning the Bidirectional Encoder Representations from Transformers (BERT) [95], a language model based on deep learning. BERT is pre-trained on a large text corpus, but then fine-tuned on the specific task of requirement classification provided with only a small amount of data. Hey et al. [88] call this approach NoRBERT, which stands for Non-functional and functional Requirements classification using BERT.

Originally, BERT is a language model. This aims to estimate the probabilities of sequences of words, thus predicting the probability of a word to follow a given sentence [88]. Though this is the main task for such a language model, they are capable of transfer learning, thus allowing them to be used for other tasks than they were trained on with little fine-tuning effort. As shown in figure 4.21, the input is tokenized and fed into BERT, after which the pooled output can be fed into a classifier that assigns probabilities to the different classes.

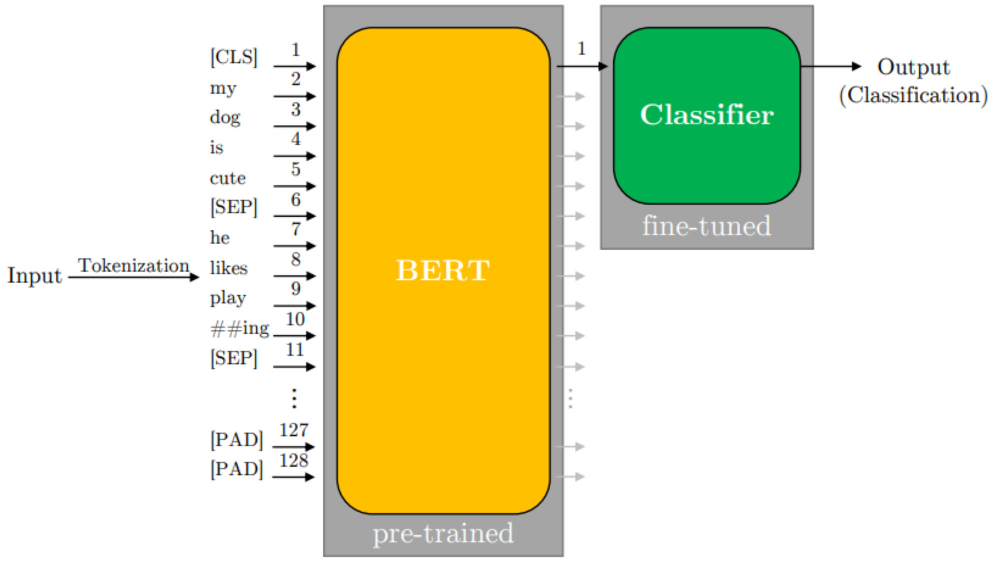


Figure 4.21: Architecture of BERT fine-tuning for classification [88]

Since the subclasses of non-functional requirements are very imbalanced, Hey et al. [88] experimented with *oversampling* and *undersampling* strategies. For *undersampling*, they randomly sampled a number of of the majority class representatives until equal to the number of minority class examples. On the other hand, *oversampling* repeatedly adds the whole training set minority class population until the number of minority class representatives would exceed one of the majority classes. Furthermore, they investigate the effect of *early stopping*, to avoid overfitting.

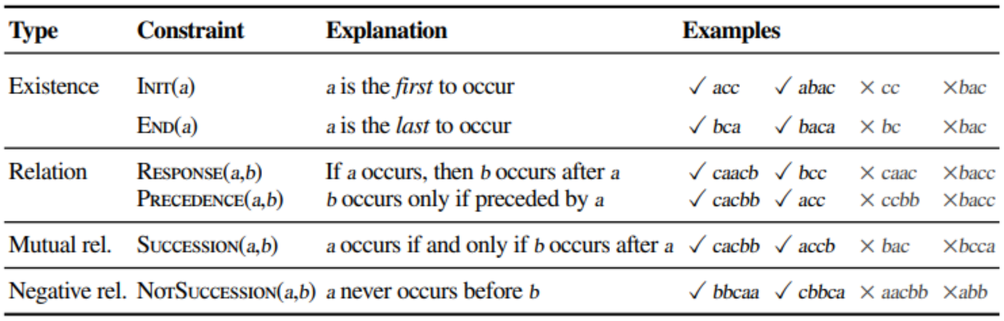
While the goal of classifying requirements to be functional or non-functional has a more narrow focus than this project, we can still relate these works back to this project. First and foremost, the usage of transfer learning could be very helpful in our project. As noted by Hey et al. [88], in cases where only limited amount of (labeled) data exists, this might prove advantageous. By using something similar like their approach using BERT as shown in figure 4.21, we could create something similar. Furthermore, the practices of over- and undersampling as seen in some of these papers should be taken into account. Likewise, the effect of other practices such as early stopping should also be investigated when creating the machine learning approach. Next to that, the most prominent features as shown by Kurtanović and Maalej [87] and Dalpiaz et al. [89] should be noted and used in the approach. Finally, the best performing supervised machine learning models shown by Abad et al. [92] and Dias Canedo and Cordeiro Mendes [91] should be taken into consideration when attempting to create a supervised machine learning model. Furthermore, the observation mate by Dalpiaz et al. [37] that the performance considerably degrades when testing on a different dataset should be a lesson for our approach.

4.2.3 Extracting Declarative Process Models from Natural Language

In the extraction of declarative process models from natural language, similar challenges were tackled that we will have in this thesis, while also using interesting techniques for our issue at hand.

For business processes that have activities executed in a typical order without any deviations or exceptions, *imperative* business process modelling can be a good way of capturing these processes [93]. For example the Business Process Modeling Notation is a notation that is readily understandable by all business users, from the business analyst, to the technical developers and the business people [96]. On the other hand, other business processes are often more complex, and their orders cannot be fully specified in advance [97]. These processes are better captured using *declarative* process models. The paper by Aa et al. [93] introduces an approach for automatic extraction of declarative process models from textual constraint descriptions.

This paper uses the DECLARE process modeling language [98], which provides a standard library of templates [99]. For this research, they used the five most occurring templates [100], as explained with some examples in figure 4.22.



Type	Constraint	Explanation	Examples			
Existence	INIT(*a*)	*a* is the *first* to occur	✓ *acc*	✓ *abac*	× *cc*	×*bac*
	END(*a*)	*a* is the *last* to occur	✓ *bca*	✓ *baca*	× *bc*	×*bac*
Relation	RESPONSE(*a*,*b*)	If *a* occurs, then *b* occurs after *a*	✓ *caacb*	✓ *bcc*	× *caac*	×*bacc*
	PRECEDENCE(*a*,*b*)	*b* occurs only if preceded by *a*	✓ *cacbb*	✓ *acc*	× *ccbb*	×*bacc*
Mutual rel.	SUCCESSION(*a*,*b*)	*a* occurs if and only if *b* occurs after *a*	✓ *cacbb*	✓ *accb*	× *bac*	×*bcca*
Negative rel.	NOTSUCCESSION(*a*,*b*)	*a* never occurs before *b*	✓ *bbcaa*	✓ *cbbca*	× *aacbb*	×*abb*

Figure 4.22: Description and notation of the considered DECLARE constraints [**Aa.e tal/CAiSE2019:DeclarativeProcessModelsfromNaturalLanguage**]

Some challenges addressed by Aa et al. [93] can be explained using their exemplary constraint descriptions in figure 4.23.

C1: Synonymous terms and phrases
In the constraint descriptions, there will be synonymous terms. For example in $S_1$1 there could be many words that indicate that this is the final step of a process. Furthermore, entire phrases could be used to describe the same context, giving us synonymous phrasing as well.

C2: Description order
There can be a different order in each sentence. For example S_1 and S_4 use a chronological order, while S_2 and S_4 use a reverse order.

C3: Noun-based actions
Another challenge is the identification and extraction of noun-based actions. While other approaches [101] have tackled identifying verb-based activities such as “a claim must be created” in S_1, noun-based activities such as “creation of a claim” in S_3 present a challenge.

C4: Constraint restrictiveness
The textual differences between RESPONSE(a, b), PRECEDENCE(a, b) and SUCCESSION(a, b) can be minor. For example the difference between S_6 and S_7 is the word “can” or “must”, but it creates the distinction between PRECEDENCE and RESPONSE.

C5: Negation
Recognizing negation will be a crucial step in this approach. For example, in S_9 the word “cannot” is used, which changes it meaning completely compared to $S_1$0.

C6: Multi-constraint descriptions
Single sentences may describe more than one declarative constraint, by using terms such as “and” or “or”. For example S_1 is identical to S_5 apart from the “or” in the end, which gives an additional constraint.

ID	Description	Constraint
S_0	The process starts when a claim is received.	INIT(*receive claim*)
S_1	A claim must be created before it can be approved	PREC.(*create claim*, *approve claim*)
S_2	Before a claim is approved, it must be created	PREC.(*create claim*, *approve claim*)
S_3	Creation of a claim should precede its approval	PREC.(*create claim*, *approve claim*)
S_4	If a claim is approved, then it must have been created first	PREC.(*create claim*, *approve claim*)
S_5	A claim must be created before it can be approved	PREC.(*create claim*, *approve claim*)
	or rejected	PREC.(*create claim*, *reject claim*)
S_6	When an order is shipped, an invoice can be sent	PREC.(*ship order*, *send invoice*)
S_7	When an order is shipped, an invoice must be sent	RESP.(*ship order*, *send invoice*)
S_8	When an order is shipped, an invoice must be sent first	PREC.(*send invoice*, *ship order*)
S_9	An invoice cannot be paid before it is received	NOTSCSN.(*pay invoice*, *receive invoice*)
S_{10}	An invoice can be paid before it is received	SCSN.(*pay invoice*, *receive invoice*)
S_{11}	The process ends when the invoice has been paid	END(*pay invoice*)

Figure 4.23: Example natural language descriptions and their constraints [93]

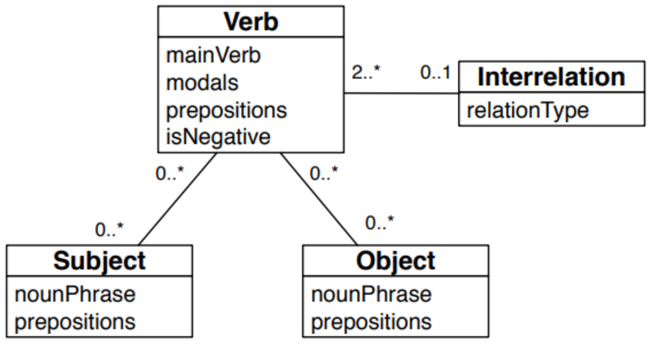


Figure 4.24: The main semantic components extracted in the first step by [93]

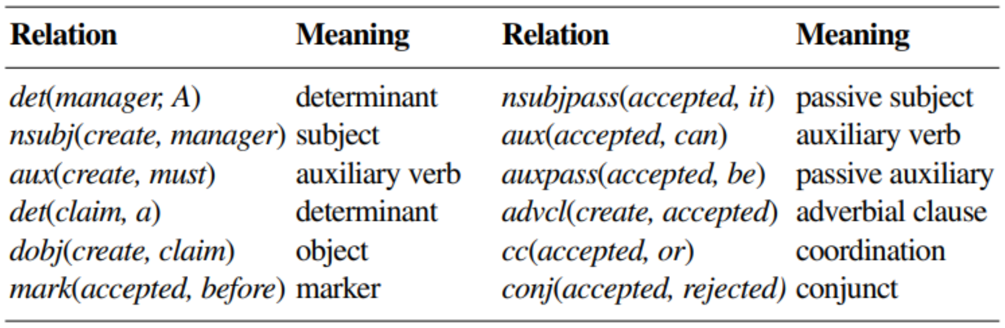
The approach by Aa et al. [93] starts with linguistic processing. This tackles challenges C1 and C2. It splits up sentences into the main semantic components shown in figure 4.24. These are identified in the following way:

- **Verbs**
  By using part-of-speech tagging, as explained in Section 3.6.1, the verbs are easily extracted from a sentence.

- **Subjects and objects**
  To identify the subjects and objects in a sentence, *dependency grammars* are used. These grammars capture the grammatical relationships in the sentence, using dependency relations, such as the *Stanford relations* [102]. By looking at S_5, we can see some helpful examples, therefore the grammatical dependencies for S_5 are shown in figure 4.25. As an example, the relation *nsubj(create, manager)* shows that “manager” is the subject performing the verb “create”. Furthermore, *dobj(create, claim)* tells us that this is the direct object of a verb. On the other hand the *nsubjpass* relation refers to a *synthetic subject* in a passive clause. As an example, the small sentence “A claim is created” would contain the relation *nsubjpass(create, claim)* which is equivalent to the *dobj* relation for active phrases.

- **Specifiers**
  The particular specifiers Aa et al. [93] focussed on were the following:
  - *Modal verbs*
    Modal verbs indicate if something is certain, probable or possible. This helps distinguish the various declarative constraints (challenge C4). By extracting these modal verbs, they can be associated with the related main verbs. For example in S_5, the *aux* relation relates "must" to the verb "created".
  - *Negation*
    By using the *neg* dependency relation, the negated verbs can be identified.
  - *Prepositions*
    Prepositions are terms to specify the relationship between a noun and another part of the sentence, commonly used to indicated ordering relations. By using *mark* and *advmod* dependencies, we can identify these prepositions. For example in S_5, which describes that "must create a claim" "before" "a claim is approved", which is found in the relation *mark(accepted, before)*.
- **Interrelations**
  There are two types of interrelations that are considered to exist among verbs in these constraint descriptions:
  - *Adverbial clauses*
    Adverbial clauses are dependent clauses that modify other entities in a text, for example in S_5 we have the relation *advcl(create, accepted)* which indicates that the term "A manager must create a claim" is a specifier for the latter term "it can be accepted".
  - *Coordinating conjunctions*
    Conjunctions are often indicated by "and" or "or", therefore in S_5, *conj(accepted, rejected)* indicates such a conjunctive relation.

After the linguistic processing, the semantic components are analyzed to identify the activities named in the description, covering challenge C3. Finally, the constraints are generated, taking on challenges C4, C5 and C6.



Relation	Meaning	Relation	Meaning
det(*manager, A*)	determinant	*nsubjpass*(*accepted, it*)	passive subject
nsubj(*create, manager*)	subject	*aux*(*accepted, can*)	auxiliary verb
aux(*create, must*)	auxiliary verb	*auxpass*(*accepted, be*)	passive auxiliary
det(*claim, a*)	determinant	*advcl*(*create, accepted*)	adverbial clause
dobj(*create, claim*)	object	*cc*(*accepted, or*)	coordination
mark(*accepted, before*)	marker	*conj*(*accepted, rejected*)	conjunct

Figure 4.25: Grammatical dependencies for constraint S_5 [93]

In the case of this thesis, we do not have access to such nice textual constraint descriptions, since we are working with transcripts from a conversation. This does not take away the fact that the linguistic processing shows us how useful part-of-speech tagging and dependency grammars can prove to be. By identifying our own challenges when it comes to identifying requirements relevant information, we can tackle our problems in a similar fashion to the approach taken by Aa et al. [93]. Furthermore, challenges C1, C3 and C5 will definitely be part of this research as well.

5 Approach development

This Chapter explains how the tool was developed and what decisions were made during this process. As noted in section 2.1, the Empirical Design Cycle was used for this project. This allowed for many adjustments and alterations in the design and structure of the approaches. An overview of the resulting tool can be seen in Figure 5.26. This tool consists of three different stages. The first, explained in Section 5.1 detects the questions in the transcript. Then, in Section 5.3 the second step in the tool is explained. This takes the identified questions from the first step, and filters them based on relevance. Finally, as a last stage, these relevant questions are categorized based on our tagged data. This data tagging is explained in Section 5.2 and the last step of this tool is explained in 5.4.

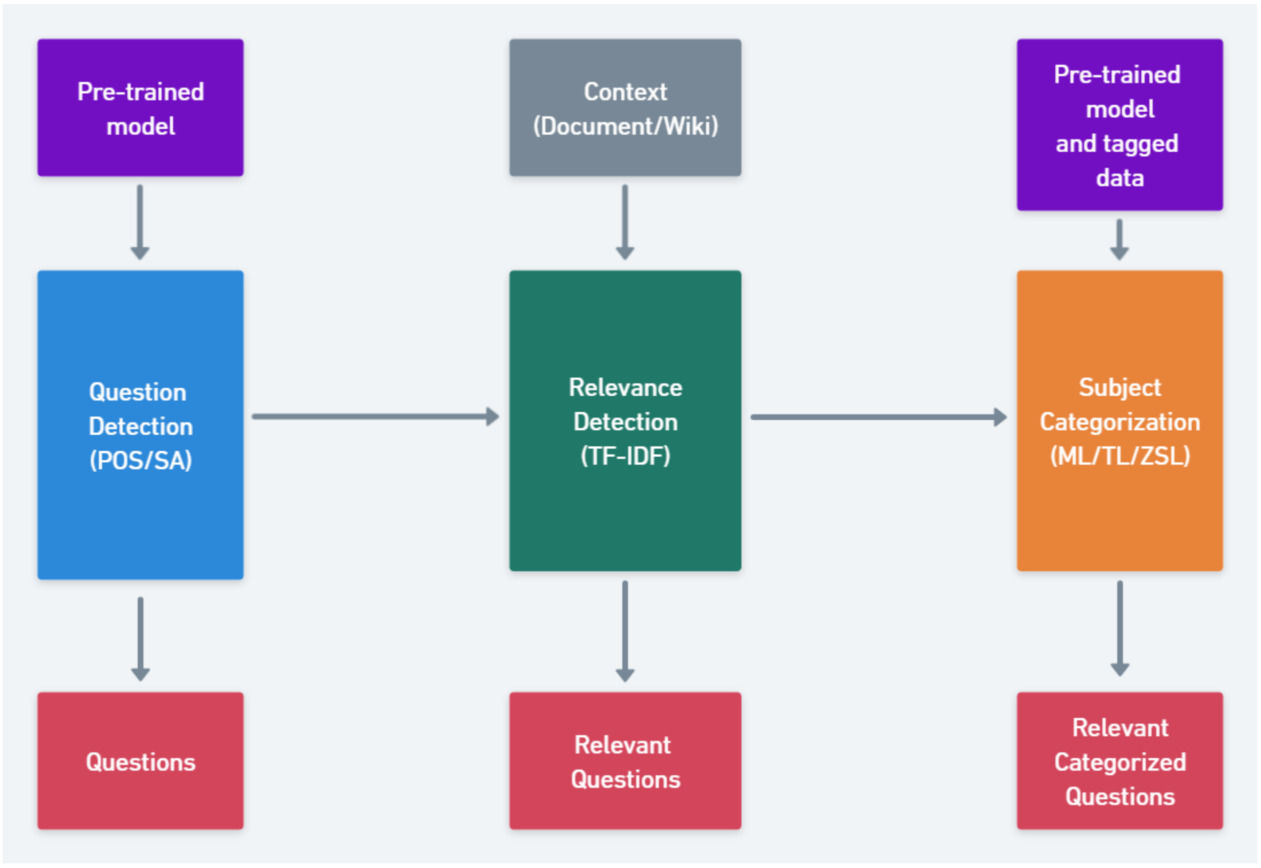

Pre-trained model
Context (Document/Wiki)
Pre-trained model and tagged data
Question Detection (POS/SA)
Relevance Detection (TF-IDF)
Subject Categorization (ML/TL/ZSL)
Questions
Relevant Questions
Relevant Categorized Questions

Figure 5.26: An overview of the resulting tool from this research, containing three different steps. One finds questions, the next filters these questions on relevance and the last categorizes these relevant questions.

5.1 Questions and answers

Our main goal for this thesis is to identify requirements-relevant information in a transcript. While we have seen in the related works in Chapter 4, that the demarcation of requirements-relevant information is an option [49]. However, this would still make it hard to traverse a transcript, since the number of speakerturns still remains the same. Therefore, we turned to summarizing the transcript of a conversation by showing the relevant speakerturns. When analyzing conversations from the RE-course, it became clear that these interviews revolve around questions and answers. This aligns with the definition from Chapter 3.2, where Briggs [18] defines an interview as “A communicative event in which the interviewers asks questions to reach the reality of a phenomenon conceived inside the mind of the interviewee.”. Often, the interviewer asks a specific questions after which the interviewee answers that questions. By filtering out the relevant questions, we could find the relevant parts of the conversation. Depending on the type of interview structure, there can be a consistent back-and-forth between questions and answers, as meant in structured interviews. While, unstructured or semi-structured interviews leave more room for flexibility and open-ended questions [103]. These open-ended questions can lead to eliciting a narrative from the interviewee [104], instead of giving a more concise answer.

By showing only the relevant questions, rather than all speakerturns, we can create a more concise view of the transcript for the user to go through. This would allow for the user to only look at the question to see if it interests them, otherwise moving on to reading the next question. When a question is of interest, they can expand on that question showing the answer to the question and more context if needed.

In most written texts, it is easy to determine whether a sentence is a question or not. Often this is indicated by a question mark at the end of a sentence. This is not always the case in our transcripts, since these were automatically generated from a recorded conversation. These transcripts can contain a question mark where no question was asked, and miss a question mark at the end of a question. Therefore, we turned to two different ways of finding questions in our transcripts.

As a starting point, we used the interviews made during the Requirements engineering (INFOMRE) course [10], given at Utrecht University by Fabiano Dalpiaz. These are 9 different interviews, where the students from this course are the interviewers, while they interview an employee from Utrecht University whom has more knowledge about the given domain. There are three different domains:

- A - The International Football Association (IFA) Portal
  The IFA wants to create a portal for which the interviewers have to figure out the requirements. This portal has to be able to manage various leagues, schedule games and referees, auditing budget of the football teams, notifying the stakeholders on events, and providing statistics on things such as games, players, teams and coaches.

- B - The Urban Mobility Simulator
  This department from a municipality has to investigate and improve the traffic situation using an urban mobility simulator.

- C - Hospital Management System
  This hospital needs a new hospital management system to integrate a majority of components that are provided over 32 different systems.

The full system descriptions can be found in Appendix B. An overview of the different conversations with their identifiers can be found in Table 1.

Identifier	Domain	Duration
2	B	00:50:23
4	A	00:49:15
5	B	00:41:29
6	C	00:23:05
7	A	00:58:06
9	C	00:38:25
10	A	00:47:12
12	C	00:39:24
16	C	00:30:31
Total		06:17:50
Average		00:41:59

Table 1: The INFOMRE conversations used in this research, with the domain it belongs to and the duration of the conversation.

The first task at hand is to identify the questions in the transcript. Since these transcripts are automatically generated, it is not as trivial as searching for a question mark. For finding questions, we have used two different approaches. The first based on Part of Speech tags and the second based on Speech Acts, as described in sections 5.1.1 and 5.1.2. This allowed us to take any automatically generated transcript and find the questions without any further input, as shown in Figure 5.27.

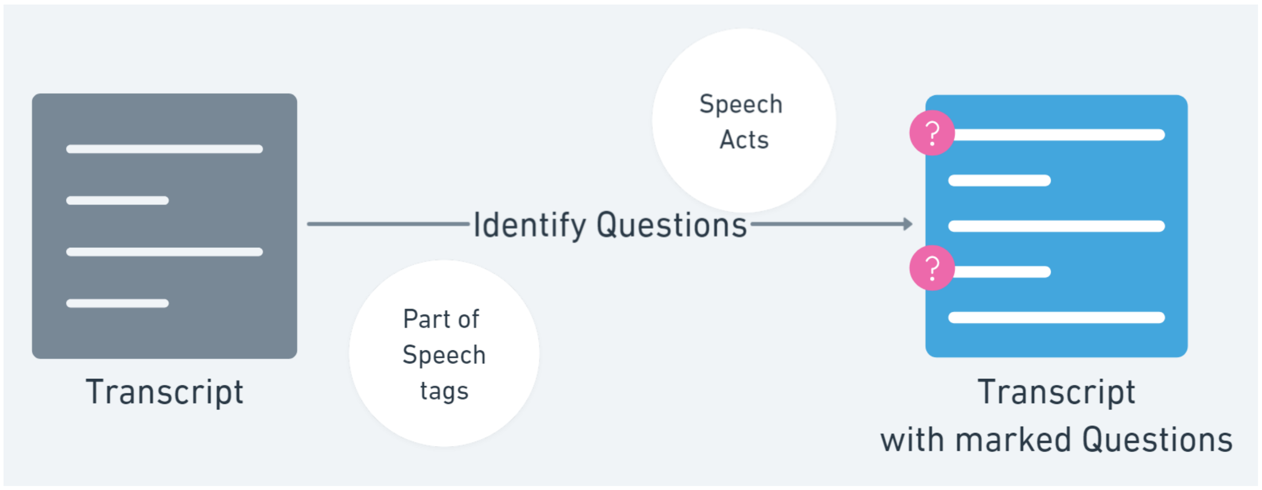


Figure 5.27: Given a transcript, we can use either Part of Speech tagging or Speech Act classification to get a transcript in which it is clear what speakerturns are asking a question.

5.1.1 Iteration 1 - POS tagging for finding questions

The first treatment design for finding questions is based on Part of Speech (POS) tagging, as explained in section 3.6.1. By using the Penn Treebank POS Tagset, we were able to determine whether a sentence is a question without relying on the question mark[11]. This Penn Treebank POS Tagset contains two clause level tags that can indicate questions [105], namely the **SBARQ** and **SQ** tag. These tags can find three different types of questions, which can be found in Figure 5.28.

[11]With inspiration taken from `https://github.com/garcia2015/NLP_QuestionDetector`

A *wh*-question, such as

```
(SBARQ (WHNP−1 Who)
    (SQ (NP−SBJ ∗T∗−1)
        (VP threw
            (NP the ball)))
    ?)
```

A *yes-no*-question, which can be answered with yes or no, for example:

```
(SQ Did
    (NP−SBJ Casey)
    (VP throw
        (NP the ball))
    ?)
```

A *tag*-question, which consists of a statement, after which a confirmation is asked. For example:

```
(SQ (S (NP−SBJ That)
    (VP 's
        (NP−PRD the problem)))
    ,
    (SQ is
        n't
        (NP−SBJ it)
        (NP−PRD ∗?∗))
    ?)
```

Figure 5.28: Some examples for each different type of question that can be found using Part of Speech tags.

By using these tags, we were able to identify questions such as the one in Figure 5.29. It would be impossible to find this situation by using only question marks, as seen in Figure 5.29a. While many questions like these were identified, there were also identified questions that were questions to the speaker themselves, for example the ones in Figure 5.29b.

INTERVIEWER : **Okay, and could you described how you would like the homepage to look like or what you would like to know from the system in Yeah, just an observation.**

INTERVIEWER : **And then in what form it is the the record itself like should the system be able to read multiple formats multiple different file types.**

(a) Examples of a question the POS tagging approach was able to find. Note that there is no question mark that could indicate a question.

INTERVIEWEE : (...) **Or what did he call?** Open street map, something like that. And you can take the data, (...)

INTERVIEWEE : (...) **How do I specify certain season for a league?** I would I sign referees to that league in specific season. (...)

INTERVIEWEE : (...) **Okay what did the doctor order, what kind of test did they order? Um Is it urgent or is it not?** So they should be able to view all of this information and then (...)

(b) Examples of questions that the speaker asked themselves, but were identified as questions by the POS tagging approach

Figure 5.29: Some examples of questions that were identified with the POS tagging approach.

In general, we are more focused on questions asked by the Interviewer than by the Interviewee. Overall, the usage of the SQ and SBARQ tags is quite low compared to the other tags, as shown in Figure 5.30.

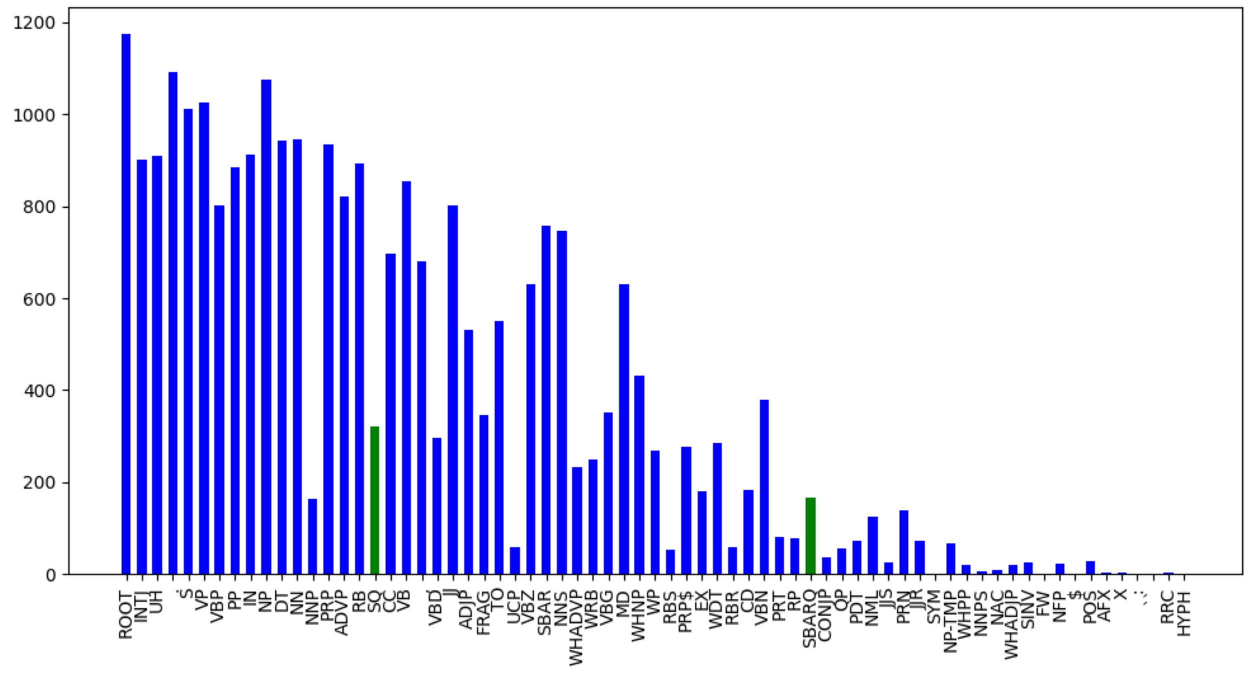


Figure 5.30: The amount of Part of Speech tags used for all of the conversations in our INFOMRE dataset.

5.1.2 Iteration 2 - Speech Act Classification for finding questions

For our second iteration, we turn to another technique for identifying our questions, and we employ classifier trained on Speech Acts. The Speech Act classifier that we used is trained on the Switchboard-1 corpus [106]. This corpus was created using 2,400 telephone conversations, with conversations among 543 speakers on 70 topics with 1,155 conversations in total. This data was tagged on their Speech Acts, of which 38 out of 42 are available using the Speech Act classifier found at `https://github.com/bhavitvyamalik/DialogTag`. The list of the Speech Acts that indicate a question and examples of their usage can be found in Table 2.

TAG	EXAMPLE
Yes-No-Question	**Is there is there already some data that can be gathered from the existing systems that can already be put in The new one or is there no,**
Wh-Question	**I'm gonna ask you, how long does it take for that person to analyze the situation and uh monitor a certain road or urban traffic situations.**
Declarative Yes-No-Question	**So it so it would be a manual change, not a new iteration of the automated schedule.**
Backchannel in question form	**Um, this should also be made available I imagine during a match for instance, the score of the match should be updated immediately once it's changed. Right?**
Open-Question	**What do you mean with local I. F. A. ?**
Rhetorical-Questions	(...) **So what kind of privileges should the coaches have**
Or-Clause	**So you you think there should be the same rights for every user of the system? Or do you think that one user should have less rights capable?**
Tag-Question	***Right?***
Declarative Wh-Question	***You are what kind of buff?***

Table 2: Example usage of the Speech Acts that indicate questions, where the **Tag-Question** and **Delcarative Wh-Question** are the default examples from `https://github.com/bhavitvyamalik/DialogTag`, since these were not identified in any of our transcripts.

By using the different Speech Acts that indicate a question being asked, we can identify the questions in our transcripts. Again, questions without a question mark could be easily identified, as shown in Table 2. We were unable to find the **Declarative Wh-question** and **Tag-Question** in our dataset. The amount of questions identified per tag varied, but most of them came from the **Yes-No-Question** tag, as can be seen in Figure 5.31. This is comparable to the amount of identified questions in the previous treatment design described in section 5.1.1. However, for this approach a lot more tags were used to identify these questions. Furthermore, we were able to use 28 out of 38 of the Speech Acts that were available. A full overview of examples of the Speech Acts found in the transcript can be found in Appendix C.

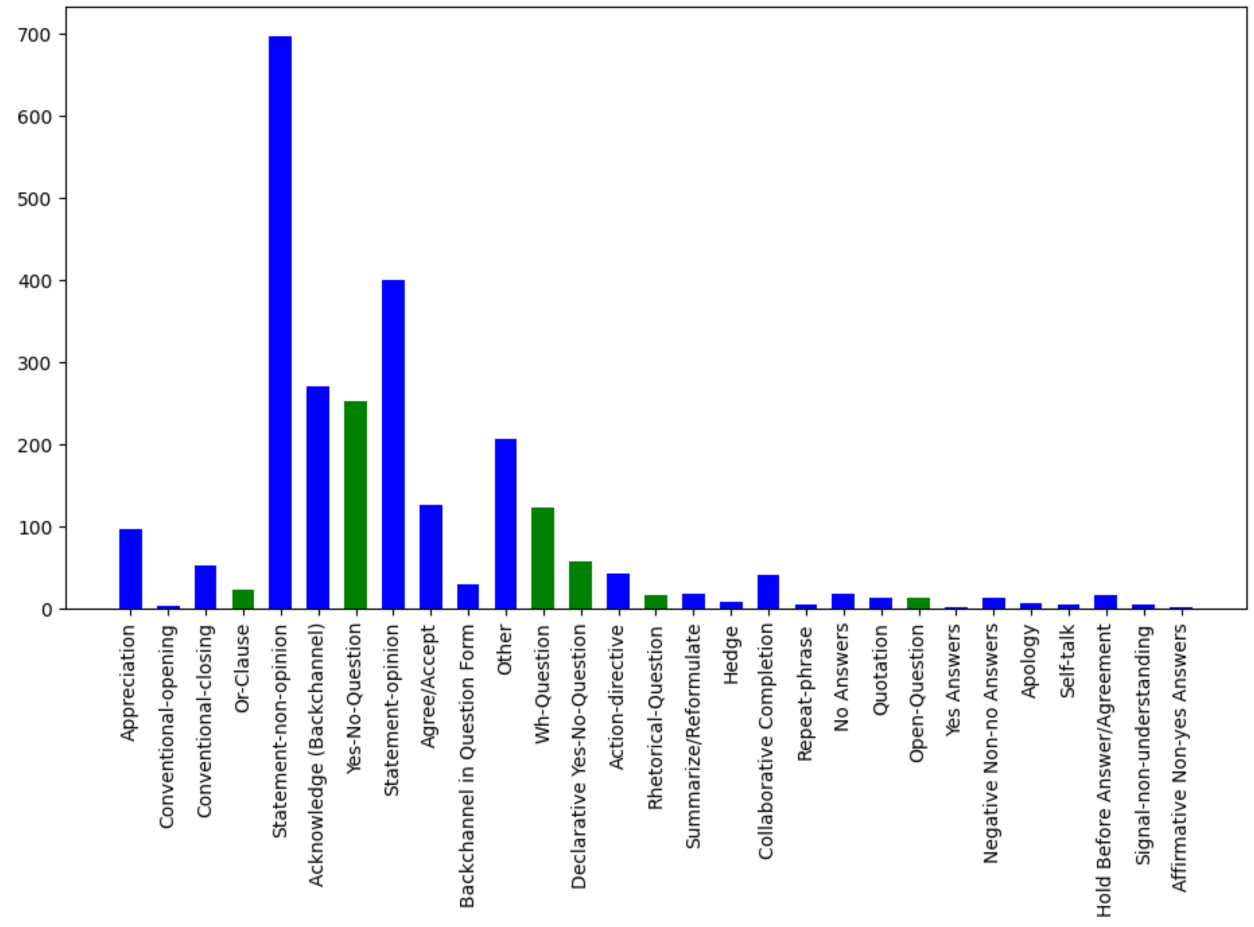


Figure 5.31: The amount of Speech Act tags used for all of the conversations in our INFOMRE dataset.

In order to test the performance of our approaches, we need to compare this to tagged data, which we will focus on in section 5.2. This will give insight into how many questions were identified correctly.

5.2 Relevance

One of the main hurdles in the development of our approach is to define for ourselves what requirements-relevant information is and how this relevance is defined. The definition of requirements-relevance has changed many times during the course of the project. Many times the question "what does it mean to be relevant?" remained an open question. This can be attributed to the fact that there are no strict rules as to when something is relevant to a requirement. At the stage that this tool can be used, these requirements are not created yet. This makes it depend on the domain knowledge and requirements engineering expertise of the person checking the sentence whether something is relevant to a requirement or not.

In the early phases of the research, we thought of checking the relevance of a sentence based on it being close to As-is situation, close to the To-be situation or not close to either, comparable to the Fit Gap Analysis [79]. While this idea still stands, it proved to be a difficult starting point. Especially when working with the automatically generated transcripts and different domains, there was no way of automated classification of our transcript based on these classes with the information that we have at our disposal.

While this first approach was a difficult starting point, another starting point was a binary classification of the transcript. Either a speakerturn is relevant or irrelevant. This allowed us to create different approaches and see how these can be improved or adjusted easily. In general, it was easy to look at and interpret the output of the binary classification. We also considered that when using machine learning for binary classification it is easier to fix overfitting or underfitting, than when using it with more than two classes.

In order to get to our definition of requirements-relevant, we went through four different iterations of Wieringa's design science research methodology [11], as Sections 5.2.1, 5.2.3, 5.2.4 and 5.2.5. These iterations allowed us to design our own tagging scheme for finding requirements-relevant information. Furthermore, in Section 5.2.6 it allowed us to create a Golden Standard, to compare our different approaches from 5.1 to.

5.2.1 Iteration 1 - Relevant questions

To start with the problem investigation, by looking at the results of our binary classification in Chapter 5.1, it became clear to us that the interviews from the INFOMRE course were revolving around questions and answers. Often, when a question is asked that is relevant to a requirement, the answers after that question are also relevant to that requirement. This allowed us to narrow our focus on relevant questions, scoping down the project. This means there are more than several categories in the set of questions, therefore introducing our treatment design for this problem:

- *Q:* Questions
  The set of questions in the transcript. Note that this set is not trivial, as question marks are often omitted or placed on a wrong place in the process of automatically transcribing a conversation.
- *RelQ:* Relevant questions
  These questions are directly relevant to a requirement. They are to the point and use the related domain terms to that requirement.
- *IdQ:* Identified questions
  The questions that are identified by the tool. As the tool is automated, we will have a set of questions that are identified by the tool as a question.
- *IdRelQ:* Identified relevant questions
  Combining the two previous categories, these are the questions that are identified by the tool and are marked as relevant by the tool.

When discussing whether a question is relevant or not, we also noted that there are multiple reasons on why a question can be relevant.

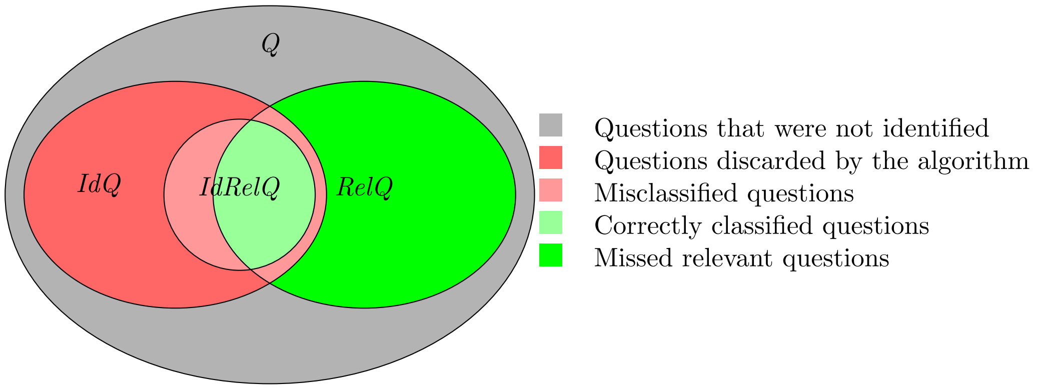

Q
IdQ
IdRelQ
RelQ
Questions that were not identified
Questions discarded by the algorithm
Misclassified questions
Correctly classified questions
Missed relevant questions

Figure 5.32: Venn diagram on possibilities on relevant questions

In order to validate this treatment, we attempted to tag a conversation ourselves. This tagging was done by two of the supervisors of this project, Fabiano Dalpiaz and Tjerk Spijkman. For this, we took conversation 2, belonging to domain B (see Table 1. The tagging was done by showing the speakerturn that contains a question, with the option to show more context. Using this option, the taggers would see the previous speakerturn and the next speakerturn. Giving this option allowed us to see whether we relied on the surrounding speakerturns or not. Several options were given for categorizing the relevance:

- *A question about functionality*
- *A question about a domain concept*
- *Another relevant question*
  This option included an option to fill in text.
- *Relevant, but not a question*
  We already noted that some questions are not relevant, but the text around the question can be relevant.

On the other hand, we had some options for distinguishing between speakerturns that are not relevant:

- *An unclear question*
- *An off-topic question*
- *Not a question*

	Tagger 1	**Tagger 2**
Total relevant	46	41
Total not relevant	11	16
Percentage relevant	81%	72%

Table 3: Tagging results on INFOMRE interview recording 2, on domain B, containing 57 questions.

Explanations tagger 1	**Tagger 2**
About the faced problems [also, more than one question here!]	Context for the project
Problem/process understanding	Can be about current process or functionality
Boundaries of the system	(missed part of the next turn) - Relevant for building the business case for the app
Goals of the system?	Goals for the app, not necessarily functionality nor domain
It's about who will use the system, not the what	Constrains / environmental question. (non-functional?)
Confirmation of the system idea presented before	Question about users / stakeholders not functionality nor domain
Mostly about qualities, not functionality	question about stakeholders again
Stakeholder/user identification	multiple questions, and most questions are more conversational than actually a question. (thinking process of the speaker)
About users	Current process is not the same as functionality.
User identification	constraint question
Budget constraints	constraint / project feasibility
Project boundaries (could be irrelevant)	non-functional question
Qualities / NFRs	validation of previous utterance
Qualities/NFRs	project approach related question
Identifying problems	planning / project management
	questions on expertise of one party

Table 4: Other reasons why the taggers marked a question as relevant

Conversation 2 is a 50 minute conversation, in which we identified 57 questions. Between the two taggers, the average percentage of relevant questions was $76,5\%$. The full statistics can be found in table 3. There were some disagreements, but most of these were attributed to the current tagging scheme, vision on the use case or were resolved in the discussion afterwards.

(I) A lot of times the options *Another relevant question* was used and several different explanations were given for that, which can be found in table 4. In our next iteration, these should be translated into the tagging scheme.

(II) Next to that, it was noted that the context should always be shown. The tagging activity became quite tedious when having to select the option to show the next and previous speakerturn.

(III) The question we are looking at in the current speakerturn should be highlighted, to prevent any confusion about this. Therefore, in the next iteration all questions will be marked in **bold**.

(IV) Furthermore, we found that not only a question can be relevant, but the question can also induce relevance. For example the question itself was not very meaningful, but the context around it was.

(V) A final remark was made on the quality of the transcripts. Often, errors in the transcription were confusing and led to different answers based on the guesses the taggers made some words were supposed to be. Also, the quality of these transcripts from the INFOMRE course may differ from reality. These interviews were straight to the point, not containing much small talk and asking very direct questions.

5.2.2 Fixing the transcripts

From Observation (V), it was clear that some errors had to be fixed in the transcript, in order for our taggers to be able to correctly tag the conversation better. Since these transcripts were automatically generated from an audio recording, these transcripts were not perfect. These fixes were needed to improve the readability of the transcript and the understanding of the concept that was discussed in the transcript. This involved different types of fixes, which varies from moving information from one speakerturn to the other, adding information from the audio recording and adjusting information that is written in the transcript. For example, in Figure 5.33 we see the question that we want is separated over three different speakerturns. For our algorithms to correctly identify the questions, we need the questions to be in one single speakerturn and not overlap with different speakerturns as is the case here. When this is not the case, we would miss out on this question and not have this information available to the requirements engineer. For further examples of all the different types of transcription errors, see Appendix D.

Interviewee : I think it's fine if it is just on a mobile version in terms of you access it via the browser. **Okay,**

Interviewer : **so web base is the**

Interviewee : **reference.** Yeah. We're not putting the focus yet on Absolutely perfect. But maybe if that's possible. I mean if there's an option for it. So in the future we might consider developing an app that will be great if we can somehow make it possible that this is a possible extension. Okay. Yeah.

(a) An example of a part of a transcript where the speakerturns are not ending properly.

Interviewee : I think it's fine if it is just on a mobile version in terms of you access it via the browser.

Interviewer : **Okay, so web base is the preference?**

Interviewee : Yeah. We're not putting the focus yet on apps. But maybe if that's possible. I mean if there's an option for it. So in the future we might consider developing an app that will be great if we can somehow make it possible that this is a possible extension.

(b) After manually adjusting the endings of the speakerturn, the transcript looks like this.

Figure 5.33: Example of a transcription error, where we adjusting the ending of a speakerturn,

5.2.3 Iteration 2 - Relevant parts of the speakerturn

The conclusions drawn in section 5.2.1 triggered a new design science iteration. Because of Observation (II), we focused more on parts of the speakerturn and the next speakerturn, shifting our focus from only the question in the current speakerturn to also the next speakerturn. To do so, we present the following sequence of three speakerturns, chronological in order:

- **The previous speakerturn**
  This speakerturn serves as context. It is there for the understanding of the conversation, but this should not be tagged or analyzed.
- **The current speakerturn**
  This is the speakerturn that we focus on and contains a question.
- **The next speakerturn**
  The following speakerturn, that possibly includes an answer to the question asked.

Due to Observation (IV), we shift our focus from finding relevant questions to asking two different questions, in order:
First, we aim to identify where the relevance is located by asking "How would you describe the relevance of the following speakerturn?". The different speakerturns allowed us to distinguish the type of relevance this set of speakerturns into the following answers:

- *A relevant question*
  The question asked in the current speakerturn is relevant and uses the related domain terms to a requirement. For example:

  CURRENT SPEAKERTURN : **What kind of platform would you like the application to support?**

- *An irrelevant question, but the speakerturn itself is relevant*
  While this question in the current speakerturn may be irrelevant, the current speakerturn itself could be relevant. So, while the question goes on about something else or is abstract and doesn't use any domain terms and is not related to any requirement, the speakerturn could be using domain terms and could be related to a certain requirement. As an example:

  CURRENT SPEAKERTURN : You mentioned over e-mail that the system should be able to support users with multiple accounts. **How does that work?**

  Here, the information that the system should be able to support users with multiple account is relevant to a requirement, but the question does not contain requirements-relevant information. It simply tries to elicit more information from the interviewee on this requirement.

- *An irrelevant question and speakerturn, but the next speakerturn is relevant*
  In this case, the speakerturn we are looking at and its question are both not relevant. Due to the question asked in this speakerturn, the next speakerturn is indeed relevant. We argue that the initial speakerturn induced the relevance of this next speakerturn, thus we should mark is as relevant. The following speakerturns show that such relevance.

  CURRENT SPEAKERTURN : **And how does that work?**

  NEXT SPEAKERTURN : Explanation..

  Here, the current speakerturn of the interviewer does not contain any relevant information, but induces the relevance in the next speakerturn. It elicited this information from the interviewee, whom gave an explanation as to how it works.

- *Not relevant*
  Neither the speakerturn, question or next speakerturn are relevant. We could say, they are talking about something abstract or off-topic subject matter. For example:

  CURRENT SPEAKERTURN : **How are you doing?**

  NEXT SPEAKERTURN : I'm fine, thanks. **How are you?**

Then, we aim to find the reason why this relevance was chosen. This scheme changed according to Observation (I). From the explanations given in the previous tagging exercise, as found in table 4, we came up with the following categories of the relevance, as shown in Figure 5.34.

What

- *A goal*
- *A functional requirement*
- *A non-functional requirement*

Who

- *System users*
- *Stakeholders*

Other

- *Faced problems or Current process understanding*
- *Project management*
- *Scope of the system or boundaries*
- *Other*
  Again, with the option to insert a textual answer

Figure 5.34: The categories of relevance for the second iteration of the approach.

Using these categories, we aim to reduce the amount of disagreements between the taggers and provide the ability to discuss their tagging.

To validate this iteration of our design, we used interview 10, which is 47 minutes long, containing 49 questions (see Table 1. These questions have been checked manually and the entire transcript was also improved manually. This manual improvement of the transcript involved fixing words that were supposed to be different words and making sure the speakerturn was cut-off correctly according to section 5.2.2. Furthermore, the questions were highlighted by putting them in bold, according to Observation (III).

The statistics on this tagging exercise can be found in table 5a. We see that we have a much higher percentage of marking a set of speakerturns as relevant. The distinction between question, current speakerturn or next speakerturn can be found in 5b. It can be noted that this dataset contained a lot of relevant questions. There were 7 disagreements on the type on what part is relevant, which turned out to be not always as intuitive as intended. This gave the insight that sometimes finding the relevant questions can be an identity function from the identified questions, making all questions relevant. What is more interesting is the reasons why these questions were marked as relevant, which can be seen in table 5c. There were 23 instances in which the two taggers tagged completely different, so they did not use any of the same tags. These disagreements were discussed and most can be attributed to the list of categories being non-exhaustive, having no crisp distinction between the categories, difference in interpretation, bias on the usage of the tagging and bias on the domain.

From this tagging exercise, we can draw the following conclusions:

(VI) Most disagreements are due to the fact that the list of reasons why something is requirements-relevant is non-exhaustive. Furthermore, for others to be able to understand what falls under each category, it would need a better description of that category.

(VII) The idea that a question can be requirements-relevant, but the speakerturn is not can be confusing. Either the question can ask something that can be answered with requirements-relevant information or not. This is a distinction that needs to be made.

(VIII) The bias of the tagger plays a big role in how they tag certain speakerturns. Once a tagger has more knowledge about the domain, they are likely to use that to fill in the gaps.

5.2.4 Iteration 3 - Questions that can be answered with requirements-relevant information

For iteration 3, from Observation (VII) can be drawn that the current forms of relevance became difficult to explain. Next to that, a focus can be put on the answer to the question as well. We are often looking for the answer to the question that is asked, hence we should look into this as well and consider both the speakerturn that contains the question and the next speakerturn that might contain the answer to that question. In order to tag these questions, we first focus on the question "Where is the **requirements-relevant** information located?". Here we introduce the notion of requirements relevance, which which we define as follows:

	Tagger 1	Tagger 2
Total relevant	49	46
Not relevant	0	3
Percentage relevant	100%	94%
Number of disagreements	7	

(a) Second tagging results on INFOMRE interview 10, which is 47 minutes long and contains 49 questions.

	Tagger 1	Tagger 2
Question relevant	48	41
Current speakerturn relevant	0	1
Next speakerturn relevant	1	4

(b) Choice of relevant tagging answers of the second tagging.

	Tagger 1	Tagger 2
Goal	0	0
Functional requirement	9	12
Non-functional requirement	14	1
System Users	0	0
Stakeholders	2	1
Faced problems or Current process understanding	10	2
Project management	0	0
Scope of the system or Boundaries	2	7
Number of disagreements	23	

(c) Reasons why the parts were marked as relevant in the second tagging.

Table 5: Results of the tagging on the second iteration of the tagging scheme.

Definition 5.2.1. Requirements-relevant information is information that is relevant to requirements engineers for setting up the requirements correctly.

To the question asked, we give the following possible answers:

- *The question in the* ***current speakerturn*** *can be answered with requirements-relevant information*
  For example, the interviewer asks a question that that is most likely answered with requirements-relevant information, since it is prompting information about the current situation.

  CURRENT SPEAKERTURN : Okay. We're not during, during the season. You should not change any policies. Yeah. Okay. That's good for the scheduling. And then the third value is better support for the fans. **And that was um, can you, first of all in the as-is situation, can you explain some of the how are the fans able to, what kind of support is there for the fans at the moment?**

- *The* ***next speakerturn*** *(after the question) contains requirements-relevant information*
  In this example, the interviewer asks a question that will be answered with requirements-relevant information, after which the interviewee gives that requirements-relevant information in their answer.

  CURRENT SPEAKERTURN : Okay. **And then if we go to the referees, what kind of things should the referees be able to manage in the system?**

  NEXT SPEAKERTURN : The referees mainly can look at the game they are scheduled to. Uh huh. Uh Can report on the events during the game.

  The next speakerturn should be marked to contain requirements-relevant information, since it explicitly describes what the referees can do. Next to that, the current speakerturn also contains a question that will be answered with requirements-relevant information, so actually both options should be taken here.

After this question, we ask a question to categorize this requirements-relevance. For this, we have the following categories, expanded with a description due to Observation (VI):

- **Functional requirement**
  Functionality. The speakerturn refers to functionality that the software system has to exhibit. For example, register users, schedule events, calculate something or allow messaging.

- **Non functional requirement**
  Software quality or non-functional requirement. The speakerturn refers to qualities that the system should provide while delivering its functionalities, e.g. speed, security, capacity, compatibility, reliability, usability, portability.

- **System Users**
  This talks about the users of the system, also include other stakeholders here that do not use the system.

- **Current Process understanding**
  This is information about the current process or system as-is, this can be about the current problems they are facing.

- **Project Management**
  For this category, anything about the management of the project should be tagged. For example, deadlines for the system to-be, colleagues, planning and expertise of people.

- **Scope of the system**
  When tagging information about the scope of the system, this is any information that talks about what the boundaries are of the system. What should be implemented in the system and what is to be forgotten or not implemented in this system. What we also tag in this are limits to the system.

- **Other**
  When you find that there is requirements-relevant information in one of the parts that you have tagged to contain requirements-relevant information, but you cannot find a category that it might fit into, please select this category and describe what you find to be requirements-relevant about this information.

For the next validation, with Observation (VIII) we noted that the actual tagging had to be done by individuals that did not have the bias that we have. Working with this data, already having more knowledge that provided about the domain, being part of the interview that was conducted or having a view on how it should be tagged for a certain purpose gave us different biases that influence the way that we would tag the data. Therefore, we introduced the idea of a group of candidates that would tag this data for us. This required the process of tagging to be robust and reliable. Our candidates consist of other academics that work for their PhD, MSc, or prospective PhD studies with dr. Fabiano Dalpiaz. We will refer to this group of academics as the *RE-Lab*.

By providing a presentation on the way that the interviews should be tagged, a written tagging guide and a system description as domain knowledge, two pairs of people from the RE-lab tagged another INFOMRE interview recording, which contained 25 questions. The tagging was done by discussing the answers and coming to an agreement on how the tagging should be done for each tagging. These discussions were monitored to see what difficulties the participants encountered and how to improve on this. When putting the tagging from the two pairs together, we come to a percentage of 92% requirements-relevant tagged speakerturns, as seen in table 6a. The trend of a high percentage of relevant questions in the INFOMRE interview recordings stays the same. Furthermore, all tags that were marked as relevant, marked that the question in the current can be answered with requirements-relevant information, while it was not always the case that the next speakerturn contained a requirements-relevant answer, see table 6b. Two categories that were often used when choosing a reason as to why this information is requirements-relevant, are functional requirement and system users, as shown in table 6c. Often in these interviews it is very clear that they are discussing some functionality or talking about the users of the system. Less frequently, but also at different points in the conversation, the current process is discussed, hence the usage of the current process understanding is used more frequent than others.

	Count	Percentage
Total relevant	23	92%
Not relevant	2	8%

(a) Pilot tagging results on an INFOMRE interview recording containing 25 questions.

	Count	Percentage
Answer relevant	20	87%
Question relevant	23	100%

(b) Pilot relevant tagging results, separated into relevant questions and answers.

	Count	Percentage
Functional requirement	17	74%
Non-functional requirement	4	17%
System Users	18	78%
Current process understanding	11	48%
Project Management	2	9%
Scope of the system	5	22%

(c) Pilot tagging results, categories for relevance.

Table 6: Results of the tagging pilot on the third iteration of the tagging scheme.

The RE-Lab participants of this pilot study tagging gave extensive feedback on their tagging experience, which led to the following observations:

(IX) They noted that the transcriptions of the interviews were at first very confusing to read, it is completely different than reading normal text as it is a conversation that is directly transcribed, thus often containing 'um' or adjustments to what a speaker was trying to say. Next to that, it would be nice to be able to see which speaker belongs to the speakerturn. This would tell a lot about the conversation and the requirements-relevance of the question or answer.

(X) They also noted that they missed the possibility to go back to the previous question, since they were still getting used to how to tag these parts of the conversation.

(XI) More specifically about the requirements-relevance categories, the participants found the scope of the system category to be confusing. If it includes both what should and should not be included in the system, most questions and answers belong to this category.

(XII) Next to that, it was unclear of how strict this categorization should be done. If the name of a system user is mentioned, does it belong in the system users category? If 'real time' is mentioned, does that indicate a non-functional requirement? Also, some categorizations can be found indirectly. For example, when the interviewee says:

Interviewee : The referee should be able to register events to their game.

We can argue that this implies that a referee should only be able to register events to the game they are scheduled to, which would be a non-functional requirement (since this is a security constraint).

(XIII) When surveying the pairs of participants that were tagging, it became clear that showing the requirements-relevance categories first, before asking where the requirements-relevant information is would be easier for our participants. However, it would restrict the tagging to only those categories, while this list may be non-exhaustive.

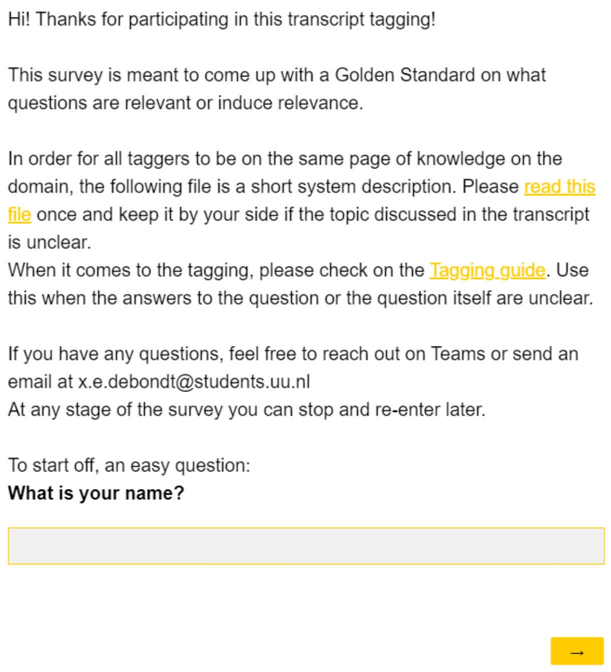
(XIV) Next to that, the term 'requirements-relevant' can be misleading, since it refers to requirements engineering as defined in definition 5.2.1, not to requirements themselves. Therefore it includes the category of project management. For a more intuitive understanding for our participants, we should instead focus on speakerturns that are relevant to requirements, not requirements engineering, and remove the project management category. This would focus our tagging on what is relevant for requirements elicitation, instead of a broader focus on the entire requirements engineering process.

5.2.5 Iteration 4 - Tagging the requirements-relevant data

This iteration, will be the final tagging scheme that will tag the entire dataset of recordings of interviews made during the INFOMRE course [10]. We improved the process of tagging according to the observations of the pilot, as described in Chapter 5.2.4. In order to gather more participants, we created some broader inclusion requirements for the tagging. These participants must:

- Have knowledge of the process of requirements elicitation, or have taken part in requirements elicitation. Once our taggers know this, they understand what kind of conversation this is and what kind of questions can be expected.
- Understand the relevance categories presented.
  It is important that our taggers agree on what information fits what category. Therefore, the categories were elaborated thoroughly in the tagging guide, but also the option itself was extended with a more extensive description of what would fit this category, some with examples. Furthermore, if anything was unclear the participants all had the option to contact me.

This allowed us to gather participants from fizor., other students at Utrecht University, more people from the RE-Lab and others. For distributing the survey among our participants, we used the Utrecht University Qualtrics survey tool[12]. By using Qualtrics, we were able to create a survey and distribute it among our participants. Furthermore, it allows the downloading of the survey data in a format of choice that can be processed afterwards. This resulted in the following order of events when a tagger uses the link distributed to them. Before any tagger can start, they are presented with the screen shown in figure 5.35. This links to a system description, as provided by the INFOMRE course that provides the taggers with sufficient domain knowledge for the system. This can be seen as sufficient, since this was the only information the interviewers had as well when they started the interview. These system descriptions can be found in Appendix B. Furthermore, there is another link that goes to the tagging guide. This is a document that provides context to the tagging exercise, explains the purpose, possible answers and gives some small examples. The tagging guide can be found in Appendix E. Finally, they are asked to enter their name. This is solely for the purpose of differentiating the different taggers.


Hi! Thanks for participating in this transcript tagging!

This survey is meant to come up with a Golden Standard on what questions are relevant or induce relevance.

In order for all taggers to be on the same page of knowledge on the domain, the following file is a short system description. Please read this file once and keep it by your side if the topic discussed in the transcript is unclear.
When it comes to the tagging, please check on the Tagging guide. Use this when the answers to the question or the question itself are unclear.

If you have any questions, feel free to reach out on Teams or send an email at x.e.debondt@students.uu.nl
At any stage of the survey you can stop and re-enter later.

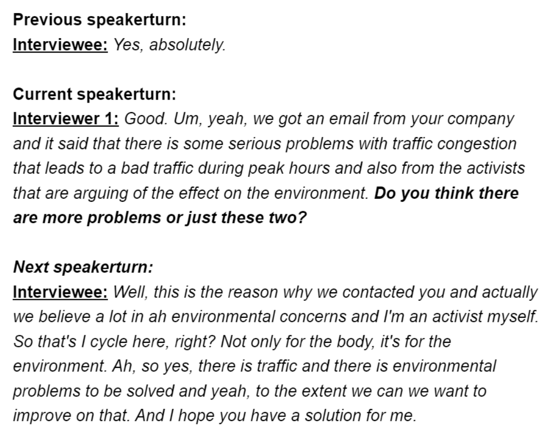
To start off, an easy question:
What is your name?

Figure 5.35: The starting screen of the survey as shown to the taggers

[12]For further reference, see `https://students.uu.nl/en/node/6/qualtrics-a-survey-tool`

In the next screens they will be shown three speakerturns, as seen in figure 5.36a. The previous speakerturn, current speakerturn and next speakerturn. The previous speakerturn is meant to provide context to the part of the conversation that is shown. The current speakerturn contains a question that interests us and the next speakerturn could contain an answer to that question. Because of Observation (IX), for each of the speakerturns, it is specified what type of speaker says this. For this, there are two types, namely Interviewer and Interviewee. In the case of these recordings from the INFOMRE course there are always two interviewers and one interviewee. Below this part of the conversation is the question "What type of requirements-relevant information can be found here?" as shown in figure 5.36b. The title of these categories was extended to make it more clear what belongs to each category, as was found necessary in Observation (XII). Furthermore, due to Observation (XIV) the project management category was removed and due to Observation (XI) we changed the scoping category to be more clear. Therefore, the taggers are presented with the following options, of which they can select multiple:

- A functional requirement (functionalities that the system should exhibit, e.g. registering users, scheduling events, calculating something, ...) Functionality. The speakerturn refers to functionality that the software system has to exhibit. For example, register users, schedule events, calculate something or allow messaging.
- A non-functional requirement (a quality that should be there given certain functionality, e.g. speed, security, capacity, compatibility, usability, ...) Software quality or non-functional requirement. The speakerturn refers to qualities that the system should provide while delivering its functionalities, e.g. speed, security, capacity, compatibility, reliability, usability, portability.
- System users (directly discusses the users of the system, or stakeholders) This talks about the users of the system, also include other stakeholders here that do not use the system.
- Current process understanding (talks about the system as-is, problems that are faced or things that have to improve) This is information about the current process or system as-is, this can be about the current problems they are facing.
- Within or outside of the scope (directly talking about certain things that are inside the scope of the system to-be or not, boundaries discussed) Any direct discussion of elements that should be in the system to-be or not. This discussed boundaries to the scope of the system.
- There is no requirements-relevant information Some questions asked in the transcript and answers to that question do not contain any requirements-relevant information. In other words, none of the other categories apply to this part of the transcript.

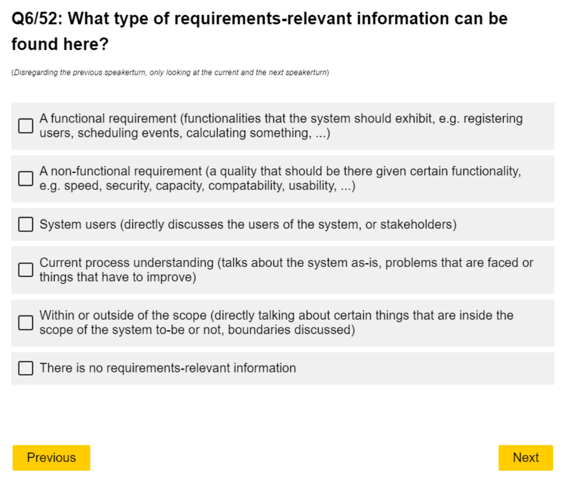


Previous speakerturn:
Interviewee: *Yes, absolutely.*

Current speakerturn:
Interviewer 1: *Good. Um, yeah, we got an email from your company and it said that there is some serious problems with traffic congestion that leads to a bad traffic during peak hours and also from the activists that are arguing of the effect on the environment.* ***Do you think there are more problems or just these two?***

Next speakerturn:
Interviewee: *Well, this is the reason why we contacted you and actually we believe a lot in ah environmental concerns and I'm an activist myself. So that's I cycle here, right? Not only for the body, it's for the environment. Ah, so yes, there is traffic and there is environmental problems to be solved and yeah, to the extent we can we want to improve on that. And I hope you have a solution for me.*

(a) An example of the part of the conversation that is shown to the tagger



Q6/52: What type of requirements-relevant information can be found here?

(Disregarding the previous speakerturn, only looking at the current and the next speakerturn)

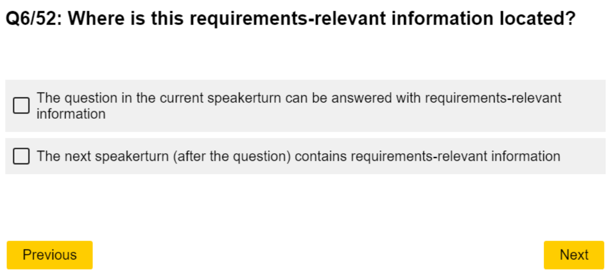
- ☐ A functional requirement (functionalities that the system should exhibit, e.g. registering users, scheduling events, calculating something, ...)
- ☐ A non-functional requirement (a quality that should be there given certain functionality, e.g. speed, security, capacity, compatability, usability, ...)
- ☐ System users (directly discusses the users of the system, or stakeholders)
- ☐ Current process understanding (talks about the system as-is, problems that are faced or things that have to improve)
- ☐ Within or outside of the scope (directly talking about certain things that are inside the scope of the system to-be or not, boundaries discussed)
- ☐ There is no requirements-relevant information

(b) The first question for every part of the conversation that is shown.

Figure 5.36: First part of the final tagging scheme

When the option 'There is no requirements-relevant information' is not selected, another question in shown, as shown in figure 5.37. Note that the order of questions is now different from the other iterations, as we changed this order according to Observation (XIII). Here the taggers are asked the question 'Where is this requirements-relevant information located?' with the following options:

- The question in the current speakerturn can be answered with requirements-relevant information
- The next speakerturn (after the question) contains requirements-relevant information



Q6/52: Where is this requirements-relevant information located?

☐ The question in the current speakerturn can be answered with requirements-relevant information

☐ The next speakerturn (after the question) contains requirements-relevant information

Figure 5.37: The second question shown for every part of the conversation.

At any point, the participants were able to go back to the previous question, as was noted in Observation (X).

To validate this iteration of the design, it is important to note that the number of questions and the length of the conversations in the dataset varied. Table 7a shows that the amount of speakerturns vary from 69 to 179, averaging at 130. The amount of questions range from 21 to 56, with an average of 42. By looking at the amount of speakerturns that were shown and that the participants were able to tag, we can look at table 7b to see what percentage of the conversation is shown in terms of speakerturns when it comes to questions, shown speakerturns and taggable speakerturns. Furthermore, when looking at the length of the conversation we can do a similar analysis. The duration of the conversation in which speakerturns ask a question can be found in table 8a. Next to that, this table shows the duration of the conversation that is shown to the tagger and that is taggable for the tagger. This translates to table 8b, where we can see the percentage of the conversation in terms of duration that is shown, taggable or asks a question.

We can say that in terms of speakerturns, we summarize the conversation down to 30% of the conversation when we only show the speakerturns that ask a question. When also showing the possible answer to this question, this turns into about 60% of the conversation. Finally, once we add the speakerturn before the question as well, this is about 70%. Likewise, when looking at the duration of the conversation, we can say that about 30% of the conversation is made of speakerturns that ask or formulate a question. Including the answer to this question, about 70% of the conversation's duration is shown. With the context of the speakerturn before the question, 80% of the duration of the conversation is shown to the tagger.

Set	Total	Shown	Taggable	Questions
2	167	117	95	52
4	148	108	87	48
5	98	69	56	30
6	69	50	41	21
7	179	132	105	56
9	116	78	64	31
10	162	110	91	49
12	155	115	98	56
16	80	62	55	31
Average	130	93	77	42
Total	1174	841	692	374

(a) Speakerturn quantities, devided in shown, Taggable and questioning.

Set	Shown	Taggable	Questions
2	70.060%	56.886%	31.138%
4	72.973%	58.784%	32.432%
5	70.408%	57.143%	30.612%
6	72.464%	59.420%	30.435%
7	73.743%	58.659%	31.285%
9	67.241%	55.172%	26.724%
10	67.901%	56.173%	30.247%
12	74.194%	63.226%	36.129%
16	77.500%	68.750%	38.750%
Total	71.635%	58.944%	31.857%

(b) Percentages on the speakerturn quantities from table 7a.

Table 7: Statistics on the speakerturns of the conversation.

Set	Total	Shown	Taggable	Question
2	00:50:23	00:42:13	00:34:37	00:09:01
4	00:49:15	00:38:53	00:34:23	00:18:21
5	00:41:29	00:30:10	00:23:30	00:06:42
6	00:23:05	00:17:30	00:16:57	00:06:00
7	00:58:06	00:51:19	00:48:49	00:25:05
9	00:38:25	00:30:57	00:23:41	00:09:05
10	00:47:12	00:37:19	00:33:13	00:17:37
12	00:39:24	00:31:03	00:28:11	00:13:34
16	00:30:31	00:26:50	00:23:21	00:08:46
Average	00:41:59	00:34:02	00:29:38	00:12:41
Total	06:17:50	05:06:14	04:26:42	01:54:11

(a) Duration of the conversation, divided in shown, taggable and questions.

Set	Shown	Taggable	Question
2	83.791%	68.707%	17.896%
4	78.951%	69.814%	37.259%
5	72.720%	56.649%	16.151%
6	75.812%	73.430%	25.993%
7	88.325%	84.022%	43.173%
9	80.564%	61.649%	23.644%
10	79.061%	70.374%	37.323%
12	78.807%	71.531%	34.433%
16	87.930%	76.516%	28.727%
Total	81.050%	70.587%	30.221%

(b) Percentages of the duration of the conversation from table 8a.

Table 8: Statistics on the duration of the conversation.

The amount of disagreements varied per pair of taggers, as can be seen in table 9a. Once a tagger chose that something contained requirements-relevant information, they were given the option whether the question and/or the answer contained this information. This allowed for the disagreement to be on the question or the answer containing requirements-relevant information. These disagreements can be found in tables 9b and 9c. Overall, there was a disagreement percentage of about 30%, where the majority of the disagreements was among the questions that were asked (about 70% of the disagreements).

Set	Amount	Percentage
2	44	46.31%
4	44	50.58%
5	23	41.07%
6	5	12.20%
7	27	25.71%
9	28	43.75%
10	44	48.35%
12	15	15.31%
16	8	14.55%
Total	238	34.40%

(a) Amount of disagreements in the final tagging, compared to the percentage of speakerturns that were tagged.

Set	Question	Answer
2	34	12
4	32	13
5	16	7
6	3	2
7	17	10
9	15	13
10	36	8
12	10	5
16	4	4
Total	167	74

(b) Separated disagreements in question and answer.

Set	Question	Answer
2	77.27%	27.27%
4	72.73%	29.55%
5	69.57%	30.43%
6	60.00%	40.00%
7	62.96%	37.04%
9	53.57%	46.43%
10	81.82%	18.18%
12	66.67%	33.33%
16	50.00%	50.00%
Total	70.17%	31.09%

(c) Percentages on the disagreement separated in question and answer.

Table 9: Statistics on the disagreements among the participants per conversation.

For the validation of iteration 4, the following observations can be made:

(XV) After and during the tagging by our participants, there was positive feedback on the differentiation between the speakers and the fact that the taggers were tagging one conversation in chronological order. This gave them the benefit of having one coherent story, allowing for sufficient focus on the exercise. Next to that, it gave them more knowledge about the domain as they progressed through the questions.

(XVI) It was noted that there was no clear category for questions that ask to clarify something further. Often it occurred that the interviewee was explaining something and the interviewer asked to explain this further, but there was no clear category for these types of questions.

(XVII) For some of the participants it would have been easier to highlight the location of the requirements-relevant information instead of ticking whether the question and/or the answer contains that information. This goes along with the feedback that there was no way to differentiate between the reason why the question contains requirements-relevant information and why the answer contains requirements-relevant information.

(XVIII) It was said that it could be beneficial that the speakerturns of the different interviewers can be combined. In some conversations, the two interviewers were discussing whether they should ask a certain question or not.

(XIX) Finally, some participants missed the initial links to the system description and the tagging guide. This led to the questions being unclear and having difficulty understanding what to do.

5.2.6 Creating a golden standard

From the final tagging in Section 5.2.5, the goal was to create a golden standard, so we have data to compare our different approaches for finding requirements-relevant information to. In order to do so, we had to solve the disagreements among participants. We focus on the disagreements of the second question, indicating where the requirements-relevant information is located. This was done by going through these agreements together with the initial taggers of the first two iterations of the design, the supervisors of this project, Fabiano Dalpiaz and Tjerk Spijkman. This allowed us to identify three different types of disagreements:

- **Missed nuance of the question.**
  The answer "The question in the current speakerturn can be answered with requirements-relevant information" was misinterpreted as "The question in the current speakerturn contains requirements-relevant information". This could be seen by certain participants selecting this answer only when there

is obvious requirements-relevant information in the question. The following example shows a question that one participant did not mark the question as one that can be answered with requirements-relevant information, however the other participant did mark it as such. We solved such disagreements, by marking the question as one that can be answered with requirements-relevant information.

INTERVIEWER 2 : **How do you typically solve such a problem? What is the way to solve a congestion like that?**

INTERVIEWEE : Yeah, so we went through many different ways. So The typical one is, especially when it is urgent, I meet a few people here in the department. These are very smart technical people, they know some, we recently hired to some mathematicians, they do great models. So what they try to do is the they kind of create a small map of roads around that problem and they tell us well, if you redirect the cars there, it's best for this and that reason. So that's something we do. Sometimes we try if it is not so urgent, we have a bigger kind of meeting uh in that case which I have to involve uh more stakeholders, more people, so we go maybe beyond the micro management. Still, it's kind of saying in the first case, we focus here. In the second case, we have a bigger area that we can consider because we vote for people.

- **The answer was a yes- or no-answer.** Sometimes, the question was answered with a yes or no. In the tagging instructions it was not made clear whether when this answer should be considered requirements-relevant, when the question asked can be answered with requirements-relevant information. For example:

INTERVIEWER 1 : **So no digital versions being sent?**

INTERVIEWEE : No. So right now you only get information when you come to an appointment from the doctor verbally um and you have to call or walk into schedule an appointment and that's basically it. Um Only on request you can request to be sent like your your file but then in a printed version other than that they are not included so far.

Therefore, here both the answer and question were marked to respectively ask for requirements-relevant information and provide it.

- **Interviewer summarizing** In some cases, the interviewer was asking a question by summarizing what has been talked about through the conversation and then asking the interviewee if they have anything to add. This is a technique which can be used when taking interviews, as noted in Section 3.2. In our case, we deem these types of questions as not relevant to our use-case. These are questions which do not clearly indicate what they are asking, therefore a user could not see what this question is about, or expect a certain answer to that question. Below is an example of such a summarizing question:

INTERVIEWER 1 : If I could just clarify this, Excuse me. Uh In this case, you want to stakeholder as the administrator to have whole um so they can manage to have all access to the each team's uh finance. So in this, in this case you want to connect both stakeholders, teams and the I. F. A Administrators. **Am I right in this case?**

These types of questions were marked to not ask for requirements-relevant information, since they do not ask for new information, they simply ask for confirmation.

By solving these disagreements and tagging them accordingly, we were able to create a golden standard dataset, consisting of all the conversations from Table 1, indicating whether a question was requirements-relevant or not.

Tables 10 and 11 show that the percentage of the conversations is reduced from about 30% to 20% when only looking at the relevant questions. This confirms our idea of giving a concise overview of the transcript by showing only the relevant questions to the requirements engineer, as this would show only 20% of the transcript.

Set	**Shown**	**Taggable**	**Question**	**Relevant Question**
2	83.791%	68.707%	17.896%	11.148%
4	78.951%	69.814%	37.259%	10.152%
5	72.720%	56.649%	16.151%	12.053%
6	75.812%	73.430%	25.993%	18.412%
7	88.325%	84.022%	43.173%	17.814%
9	80.564%	61.649%	23.644%	13.449%
10	79.061%	70.374%	37.323%	32.698%
12	78.807%	71.531%	34.433%	23.266%
16	87.930%	76.516%	28.727%	16.603%
Total	81.050%	70.587%	30.221%	20.004%

Table 10: Final percentages of the duration of the conversation, adjusted from table 8b, now showing how long the relevant questions take.

Set	**Shown**	**Taggable**	**Questions**	**Relevant Questions**
2	70.060%	56.886%	31.138%	17.365%
4	72.973%	58.784%	32.432%	22.297%
5	70.408%	57.143%	30.612%	20.408%
6	72.464%	59.420%	30.435%	21.739%
7	73.743%	58.659%	31.285%	10.615%
9	67.241%	55.172%	26.724%	13.793%
10	67.901%	56.173%	30.247%	25.309%
12	74.194%	63.226%	36.129%	28.871%
16	77.500%	68.750%	38.750%	23.750%
Total	71.635%	58.944%	31.857%	19.506%

Table 11: Percentages on the amount of speakerturns of the conversation, adjusted from table 7b, now including the relevant questions

5.3 Identifying requirements-relevant question

In order to find the requirements-relevant questions, as the taggers did when creating the golden standard in Section 5.2, we have created two different treatment designs using Wieringa's design science research methodology [11]. The first iteration takes the System Description that belongs to each of the different domains that our interviews belong to (see Appendix B). We take this as our Context Document, as it provides context to our classifier as to what words are expected to be used. The second iteration compares our conversation to the words used in a large corpus of Wikipedia pages. Then, in Section 5.3.3, we see how our approaches in combination with the approaches in Section 5.1 compare to our golden standard created in Section 5.2. Finally, we also test how well Artificial Intelligence works on our tagged dataset, in Section 5.3.6.

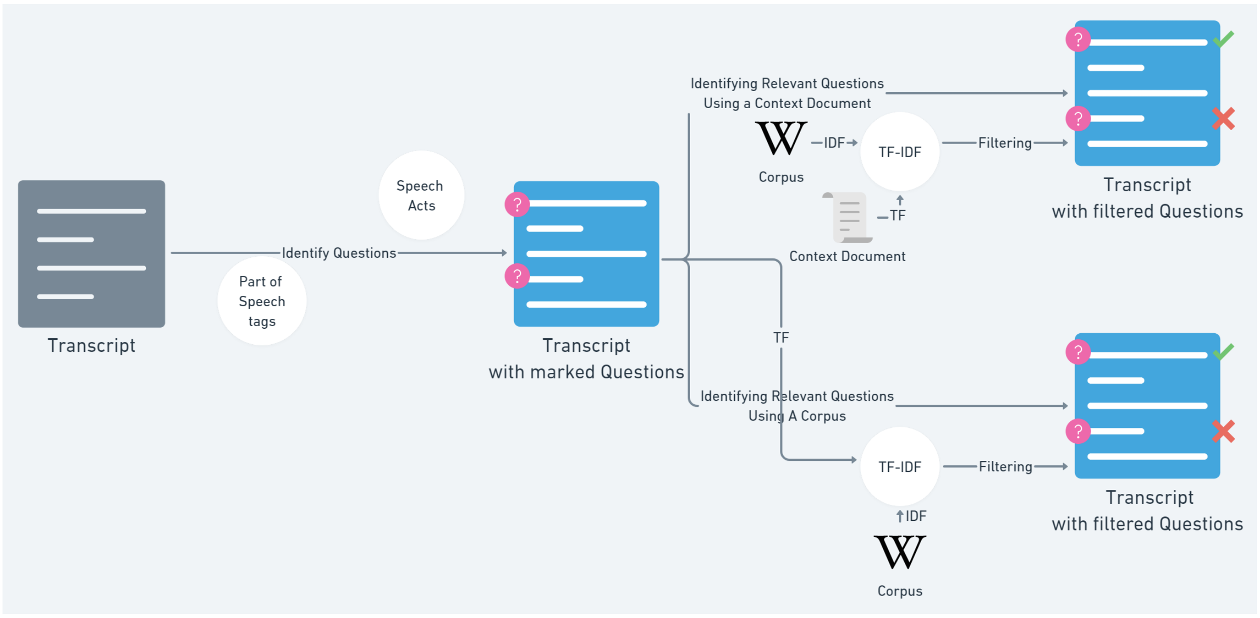

Transcript
Identify Questions
Part of Speech tags
Speech Acts
Transcript with marked Questions
Identifying Relevant Questions Using a Context Document
W
IDF
Corpus
TF-IDF
TF
Context Document
Filtering
Transcript with filtered Questions
TF
Identifying Relevant Questions Using A Corpus
TF-IDF
Filtering
IDF
W
Corpus
Transcript with filtered Questions

Figure 5.38: A flowchart to explain the flow of the combination of our treatment designs. First the relevant questions are identified, using either Part of Speech Tags or Speech Act classification. After that, the relevance of these questions is classified. This is done by using TF-IDF on the Context Document or on the Transcript itself. As a final output, this gives us a set of relevant questions according to our algorithms.

5.3.1 Iteration 1 - Context Document

To tackle the problem of finding requirements-relevant questions, the INFOMRE dataset provided us with system descriptions, which can be found in Appendix B. These system descriptions introduce the context and the project aims that the interviews will focus on. Therefore, in this first treatment design we will focus on these system descriptions, as these documents are the starting points for our interviewers to ask their questions. The terminology used in these documents will most likely match the terminology that will be used in the conversations and therefore this terminology could determine whether a question is about something that is relevant to the requirements of the system or not. To determine what specific terminology is used in the context document, we used the stems and terms from a dataset containing a lot of Wikipedia pages [107]. By using these terms in combination with Term Frequency-Inverse Document Frequency (TF-IDF), we can determine what words in this system description are interesting. In combination with using this on the questions can determine with the algorithms in section 5.1, we can find the questions that are interesting to us, and most likely contain requirements-relevant information. This can be done with any document that has this sort of context on the domain, therefore we shall refer to such a document as a Context Document, as shown in Figure 5.39.

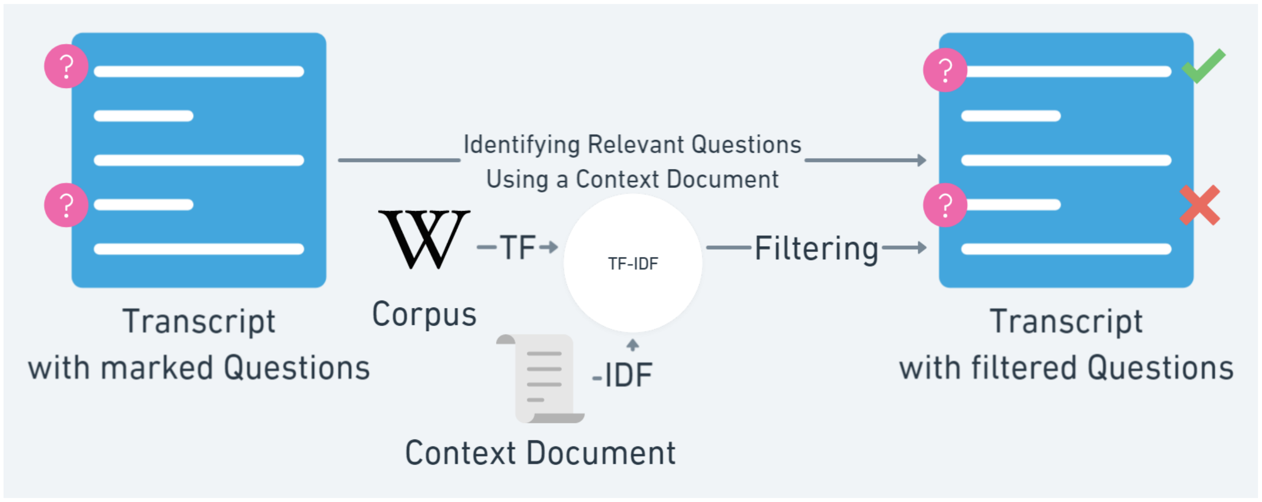


Figure 5.39: By taking a Transcript that already has the questions marked, we can use TF-IDF with a Wikipedia corpus and our Context Document in order to find the relevant questions.

After filtering based on common stop words, this approach gave us a set of words per context document that most likely indicate relevance in our questions, of which Table 12 shows a subset.

Domain	**Relevance-indicating words**
A	IFA, stakeholders, auditing, infrastructure, notify, scheduling, cancellation, fans, operational, teams, rescheduling, budget, real-time, cio, systems, leagues, coaches, system, information, restructure, analytics, stadiums, unauthorized, automation, support, transparency, monitored, transaction, updates, games, administration
B	mow, congestions, mobility, cloudbased, simulator, urban, microlevel, traffic, urgencies, simulation, pollution, replicating, planner, mathematical, citizens
C	medz, transferals, scheduling, hospital, patient, management, healthcare, departments, digitalization, privacy, janitors, data, interoperable, presciptions, laboratory, referral, appointment, analytics, nurses

Table 12: A selection of the words extracted using TF-IDF on our System Descriptions from Appendix B.

5.3.2 Iteration 2 - Wikipedia

In order for us to create a second iteration, we can also apply Term Frequency-Inverse Document Frequency (TF-IDF) on the transcript of the conversation itself. This would allow us to focus on the words that are actually used in the conversation, instead of hoping that the Interviewer and Interviewee actually use the terminology used in the Context Document. By using the stems and term from the Wikipedia dataset [107] and using this in combination with our transcript with TF-IDF, we were able to identify different sets of words that can indicate our relevant questions, as shown in Figure 5.40.

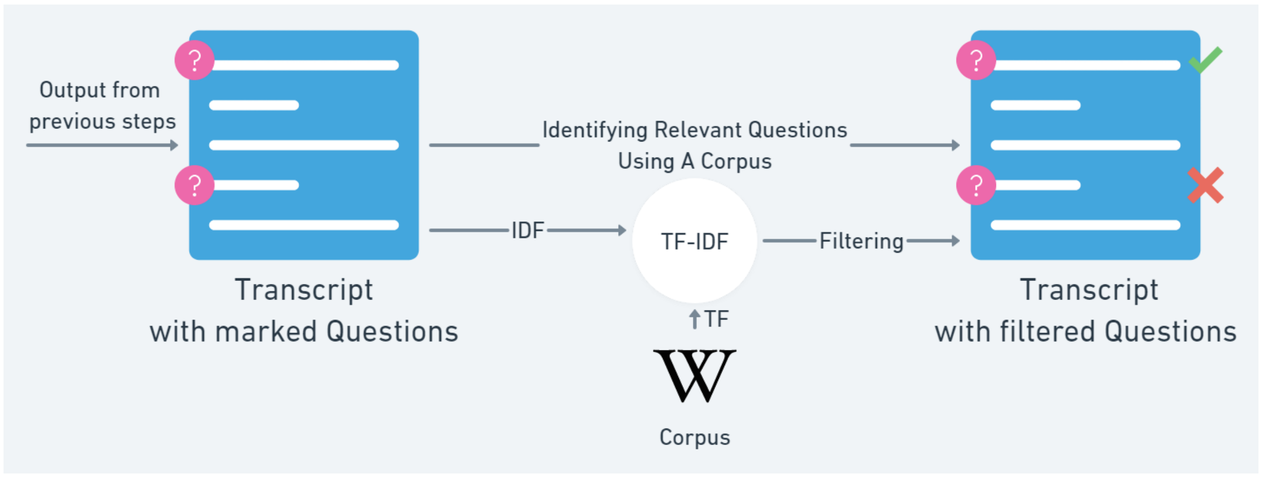

Output from previous steps
Identifying Relevant Questions Using A Corpus
IDF
TF-IDF
Filtering
TF
Corpus
Transcript with marked Questions
Transcript with filtered Questions

Figure 5.40: By taking a Transcript that already has the questions marked, we can use TF-IDF with a Wikipedia corpus and our Transcript in order to find the relevant questions.

While these words did need a lot more filtering, as there are some special words being used in the conversations nevertheless (such as ’Uh’, ’uhh’ and ’Um’), it did provide an interesting set of words for each of the conversations, as can be seen in Table 13

Identifier	Domain	Relevance-indicating words
2	B	sensors, wonderland, CO2, simulator, maps, traffic, prototype
4	A	referees, scheduling, cancellation, notification, stakeholder, transparency, granularity, discussed, automation, fans, teams, leagues, budget
5	B	roundabout, sensors, pollution, traffic, congestion, functionalities, reversible, decisions, simulation
6	C	patient, privacy, scheduling, prescription, mobile, data, analytics, appointment, database, pharmacy, devices, medical, system, electronic
7	A	referees, stakeholder, notification, scheduling, fans, offline, schedule, cancelling
9	C	patient, DigiD, devices, patients, doctors, nurses, medical, privacy, receptionists, solution, data, laboratory, administrator
10	A	scheduling, referees, notifications, biases, notify, pages, fans, budget, privileges, coaches, manage, teams,
12	C	patient, janitors, pharmacies, scope, scheduling, receptionist, nurses, patients, laboratory, doctors, data, prescriptions, authentication
16	C	scheduling, patient, appointment, nurses, mobile, receptionists, inventory, device, privacy, outdated, doctors, alert, safety

Table 13: A selection of the words extracted using TF-IDF on our Transcripts.

5.3.3 Automated requirements-relevant questions

In order to validate our treatment design for automatically finding requirements-relevant information, we will combine the approaches used in section 5.1 and 5.3. This means combining the Part Of Speech Tagging and Speech Act Tagging approaches from Section 5.1.1 and 5.1.2 with the Context Document TF-IDF and Wikipedia TF-IDF approaches from Section 5.3.1 and 5.3.2. By combining these approaches as shown in Figure 5.38 we can compare our results to the Golden Standard created in Section 5.2 and see how well our treatment design works.

5.3.4 Comparing approaches for finding questions

First, we can compare our different approaches created for finding relevant questions. Here, we can also take the combination of both approaches, which simply takes something to be a question if either of the algorithms indicates this. For the results, we look at the confusion matrices of our approaches. These matrices show the performance of a classifier with respect to some data it is tested on [108]. These matrices show the true class of an object in the rows, while the predicted class of the object assigned by the classifier can be found in the column.

For example in Figure 5.41 we see a confusion matrix where we show the performance of a classifier with respect to class A. Here, the cases where both the classifier and the actual class are A is designated as True Positives (TP). Furthermore, the same is done for where both the classifier and the actual class are not A, which are called True Negatives (TN). Next to that, in the cases where the classification goes wrong, we have two different designations. When the classifier predicts an object to

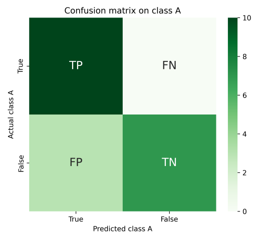


Figure 5.41: An example confusion matrix, on a classifier for class A.

belong to class A, while in fact it does not, we speak of a False Positive (FP). Likewise, when the classifier predicts an object not to belong to class while in truth it does belong to class A, we speak of a False Negative (FN). Looking at the matrix in Figure 5.41 we see from the color bar that there are a lot of True Positives, namely about 10. There are a bit less True Negatives, about 7 or so. There are no False Negatives but there are approximately 3 False Positives. This is what we can quickly draw from this color bar, but in our case the next confusion matrices in this thesis will have the exact numbers in the cells, so no estimation is needed, but the color bar can give a quick overview.

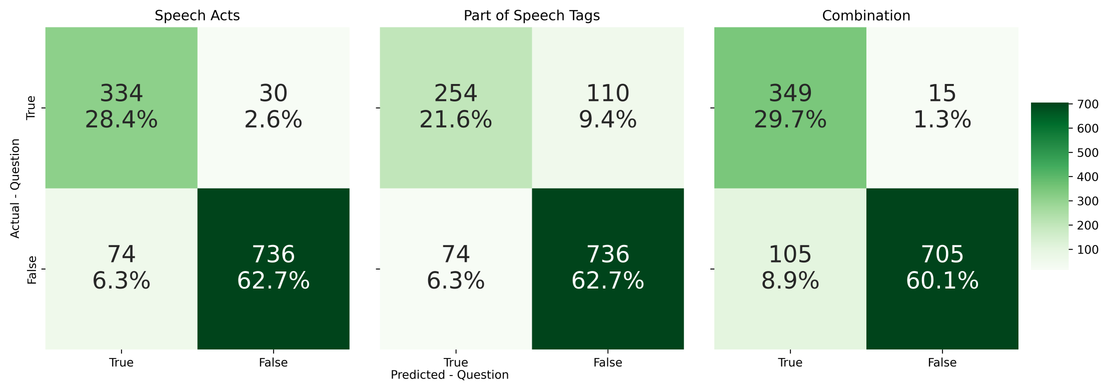


Figure 5.42: Confusion Matrices for our different approaches for identifying questions.

Figure 5.42 show the confusion matrices for our two different approaches. It can be noted that both algorithms work quite well, none a high number of False Positives or False Negatives. It must be said that Speech Acts have a lot more True Positives, whereas the POS tags approach made a lot of False Negatives. This can also be seen in the Precision, Recall, F1-Score and Accuracy of these solutions, as shown in Table 14. It can be said that in our case, Speech Acts have proven to be more useful.

The combination of both the algorithms if able to find more True Positives, at the compromise of finding fewer True Negatives. It finds a lot more questions, since the False Positives have gone up as well. In terms of statistics, it is better than Part of Speech Tags alone, but cannot improve over Speech Acts. However, we are interested in finding all the questions, so this Combination approach is still a viable option, since it returns the most True Positives.

Approach	**Precision**	**Recall**	**F1-score**	**Accuracy**
Speech Acts	81.8%	91.7%	86.5%	91.1%
Part of Speech Tags	69.7%	77.4%	73.4%	84.3%
Combination	76.8%	95.8%	85.3%	89.7%

Table 14: Precision, recall, f1 and accuracy on different approaches for finding questions.

5.3.5 Comparing approaches for finding relevant questions

By combining our approaches for identifying questions and filtering relevant questions, we can create several confusion matrices. First, comparing our Context Document and Wikipedia TF-IDF approaches on the questions identified by Speech Act classification. This comparison is done by looking at the confusion matrices of the two approaches, similar to the comparison done in Section 5.3.4. The confusion matrices for our approaches can be seen in Figure 5.43.

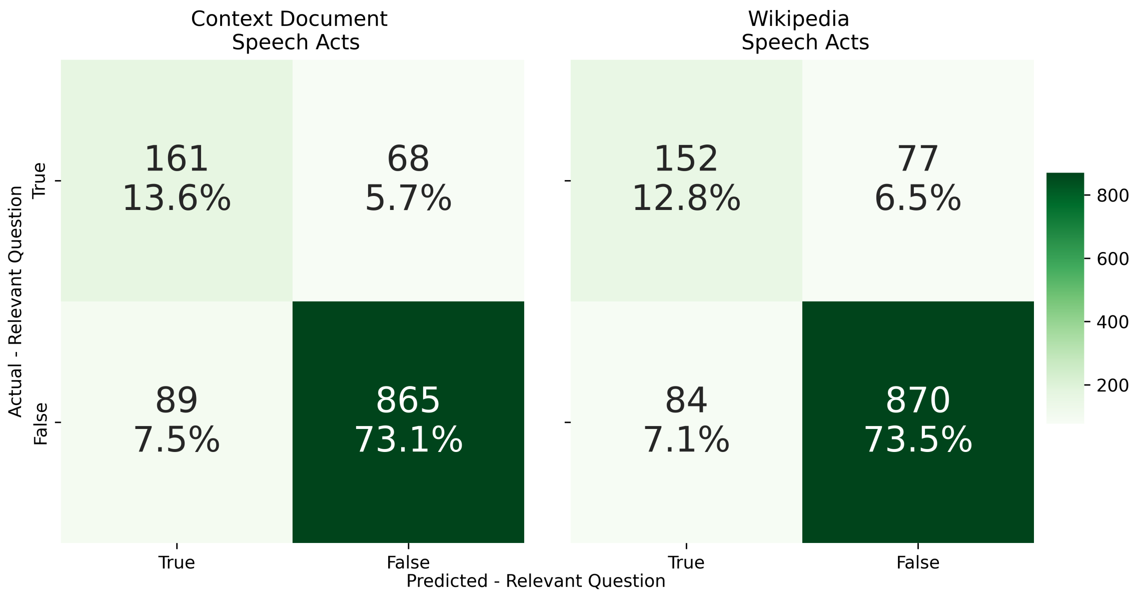


Figure 5.43: The confusion matrices of our Context Document and Wikipedia TF-IDF approaches on the questions identified by the Speech Act classification approach compared to the Golden Standard found in section 5.2.

Again, both of the algorithms hold up quite well, although there are a lot of False Positives. While these approaches seem to identify a lot of the relevant questions, in the process they also take a long some questions that were not deemed relevant by the taggers.

Similarly, we can compare our approaches using Part of Speech Tagging, resulting in the confusion matrix from Figure 5.44.

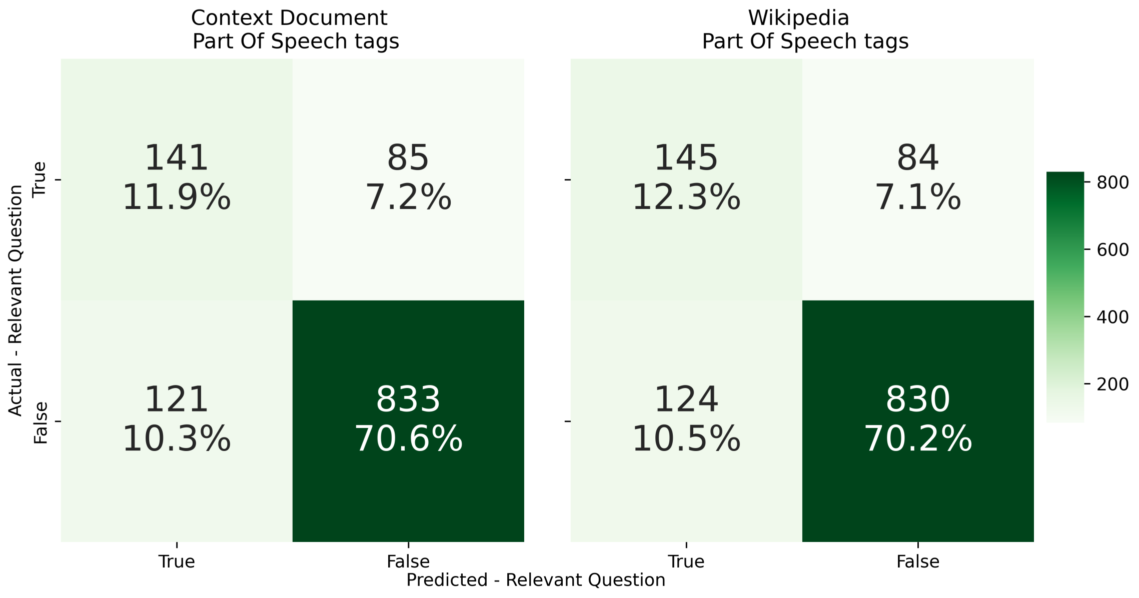


Figure 5.44: The confusion matrices of our Context Document and Wikipedia TF-IDF approaches on the questions identified by the Part of Speech tags approach compared to the Golden Standard found in section 5.2.

Approach	**Precision**	**Recall**	**F1-score**	**Accuracy**
Context Doc. - SA	64.4%	70.3%	67.2%	86.7%
Wikipedia - SA	64.4%	66.4%	65.4%	86.4%
Context Doc. - POS	53.8%	62.4%	57.8%	82.5%
Wikipedia - POS	53.9%	63.3%	58.2%	82.4%
Context Doc. - COMB	55.0%	81.7%	65.7%	83.5%
Wikipedia - COMB	55.7%	81.2%	66.1%	83.9%

Table 15: Precision, recall, f1 and accuracy on different approaches for finding relevant questions.

The results from this combination of approaches is similar to the previous. It must be noted that there are more False Positives in this case, which can be explained by the performance of our Part of Speech tagging approach. The Part of Speech tagging approach allowed a higher number of mistakes for finding questions to come through, leading to more False Positives when it comes to finding relevant questions.

Finally, we can compare our approaches using the Combination of the Speech Act classification and Part of Speech tags approach. These results are shown in Figure 5.45.

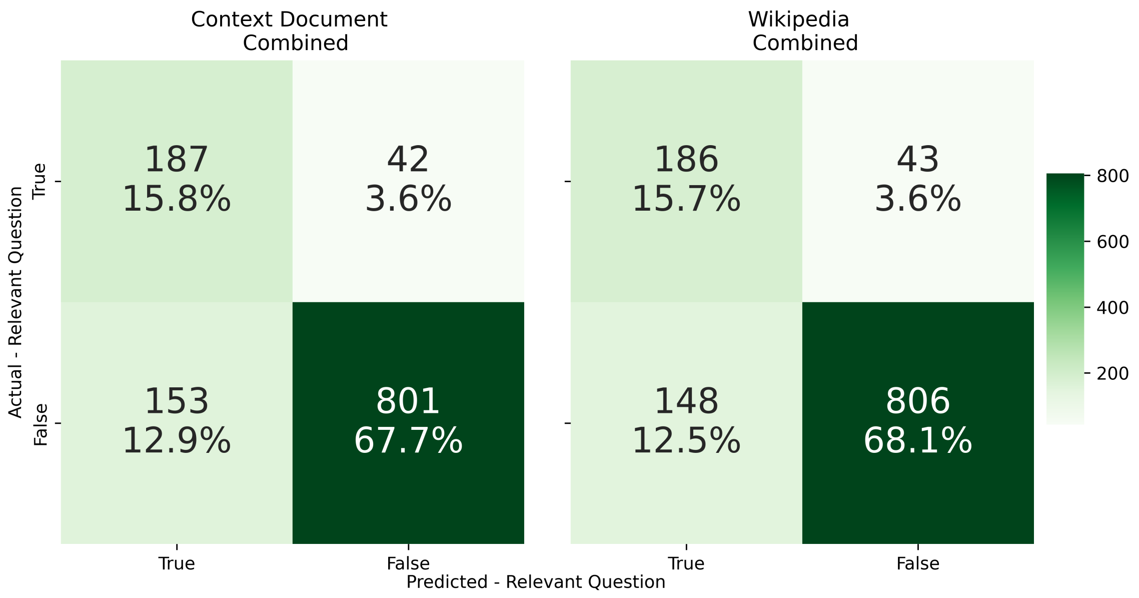


Figure 5.45: The confusion matrices of our Context Document and Wikipedia TF-IDF approaches on the questions identified by the Combination of the Speech Act classification and the Part of Speech tagging approach compared to the Golden Standard found in section 5.2.

These results are very similar to the results from using Part of Speech tagging approach, however they include more True Positives due to the questions identified by the Speech Act classification approach.

In order to compare all these approaches at the same time, we can look at the statistics found in Table 15.

We see that the Context Document and Wikipedia TF-IDF approaches perform in a comparable way. In all of the different approaches for finding questions, there are only minor differences in the amounts. Furthermore, it can be said that the Speech Act classification approach outperforms Part of Speech tagging approach, where a compromise can be made by taking the combination of both approaches, leading to more identified questions. This can give a higher recall, since there are more True Positives, however the precision will be lower since the False Positives will also increase.

5.3.6 Comparing 'learning' approaches for finding relevant questions

Since our Golden Standard dataset available from section 5.2, we test how well Machine Learning or Transfer Learning approaches work on this data. This would involve taking the Transcript and then finding the relevant questions, replacing the full solutions we compared earlier, as shown in Figure 5.46.

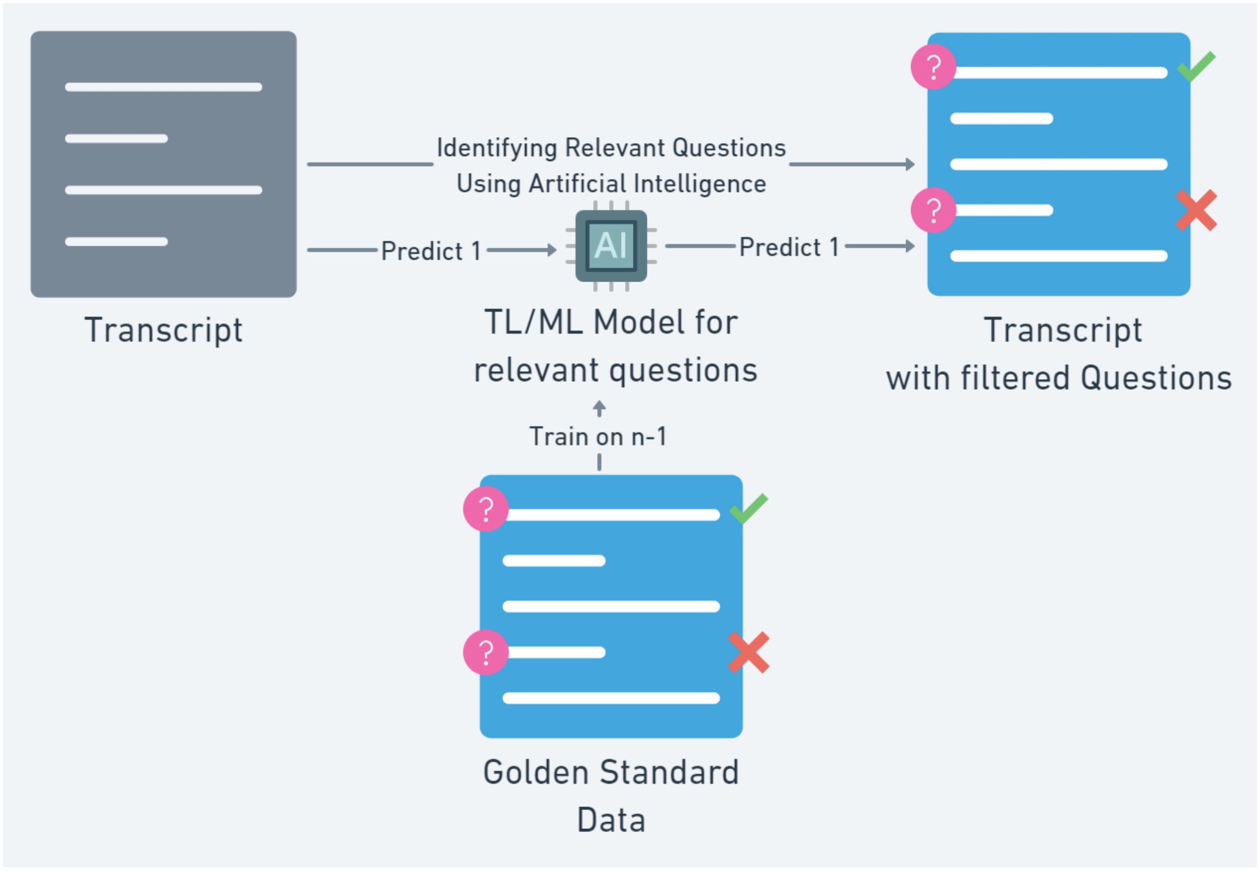

Transcript
Identifying Relevant Questions Using Artificial Intelligence
Predict 1
AI
Predict 1
TL/ML Model for relevant questions
Transcript with filtered Questions
Train on n-1
Golden Standard Data

Figure 5.46: A flowchart to explain the flow of the combination of our next treatment designs. In one step, the the relevant questions are identified, using either a Machine Learning or Transfer Learning model. This is done by using Golden Standard as training data. We train on all but one dataset, which we predict. Predicting all of the data with different models.

For Transfer Learning, we used a BERT-model, which took both the speakerturn it was attempting to classify and the speakerturn after it. This was done, because when tagging the data, it was often needed to look at the answer given to the question in order to determine whether it is relevant or not. The model used for Machine Learning was a Random Forest classifier. This Random Forest classifier was also given some extra Boolean features, inspired by Abualhaija et al. [49]:

- **hasModal**
  This checks if a Modal verb is used in the questioning sentence.
- **hasNPModalVP**
  This checks if a Noun Phrase is followed by a Verb Phrase that uses a Modal verb.
- **hasHFNP**
  This checks if a high frequency (top 1%) Noun Phrase is used.

Note that the features presented by Abualhaija et al. [49] are based on requirements specifications, therefore using this on our automatically transcribed text is very ambitious, but worth a shot. To research these classifiers thoroughly, we follow the train-validate-test methodology.

To validate these models, we use a stratified k-fold comparison, using $k = 5$ and $k = 10$ to see what kind of result we can expect. For this, we look at the confusion matrices of these models.

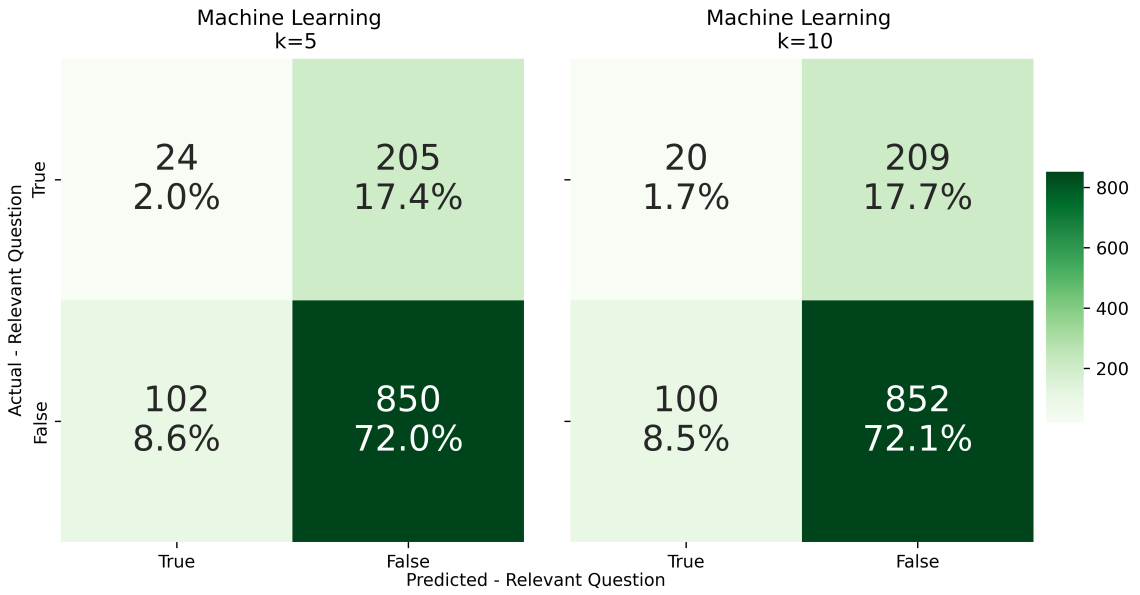


Figure 5.47: The confusion matrices of the machine learning approach, based on the tagging done in section 5.2.5, evaluated using stratified k-fold.

In Figure 5.47 can see that, the Machine Learning model has a strong bias towards predicting speakerturns to not be a relevant question. Furthermore, it does predict some to be relevant questions, but the majority of those predictions are false.

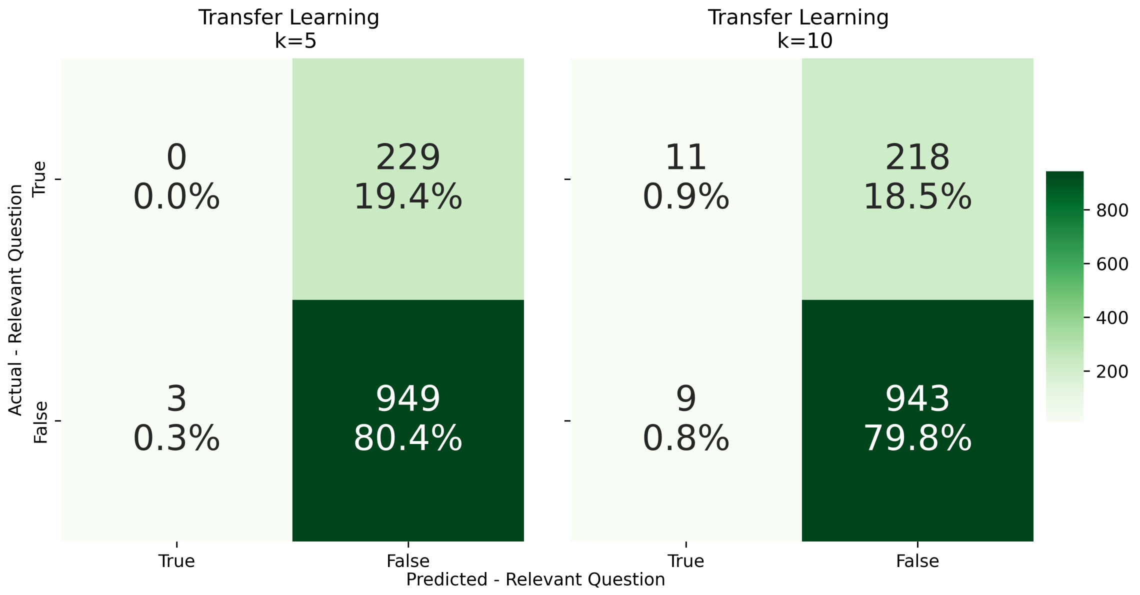


Figure 5.48: The confusion matrices of the transfer learning approach, based on the tagging done in section 5.2.5, evaluated using stratified k-fold.

The Transfer Learning model seems to be predicting almost everything to not be a relevant question, as can be seen in the confusion matrices in Figure 5.48. This does not improve significantly by taking a higher value for k. Therefore, we can expect this behaviour to be similar when testing this model.

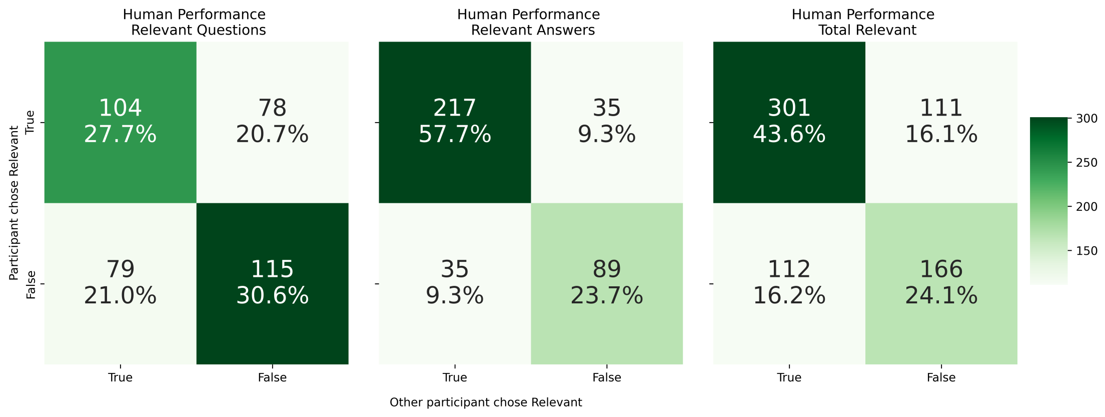


Figure 5.49: The confusion matrices based on the tagging done in section 5.2.5. This is separated on the disagreements on the relevance of questions, answer and the total.

When comparing these final results of these models, we should also compare this to the human performance on tagging the dataset. How much did the participants (dis)agree amongst the pairs? Therefore, we can look at the confusion matrices in Figure 5.49.

We can see, that in general about half of the questions presented, the participants agreed on to either be relevant or irrelevant. Furthermore, it was easier to agree on the answers, than it was to agree on the questions being relevant. Comparing this to the performance of our algorithms, we see the following row of confusion matrices in Figure 5.50.

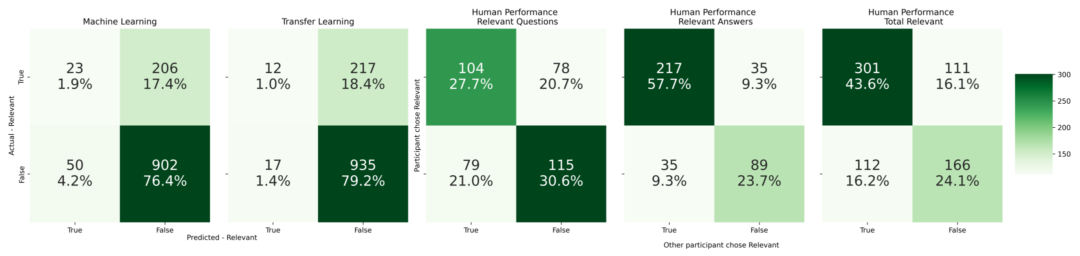


Figure 5.50: The confusion matrices based on the tagging done in section 5.2.5. This is separated on the disagreements on the relevance of questions, answer and the total.

It can be noted that both classifiers try to classify all of the speakerturns as not relevant. Their predictions on something to be relevant are very poor, as for Transfer Learning about half are false, while for Machine Learning this is more than half. Next to that, it is incomparable to the human performance. This poor performance can be accredited due to the hard task at hand. For a classifier to identify questions that are relevant in one step, whilst not giving any information on what is relevant other than training that is very hard. Next to that, the amount of questions that were relevant in our dataset was 229 while there were 1181 speakerturns.

5.4 Classifying the reason of requirements relevance

When creating the Golden Standard in section 5.2.5, the participants were selecting a reason as to why this information is requirements-relevant. This can be seen as a labeling on the data. By using this training a classifier on this labeling, we can add this labeling to the output of our tool. This would quickly indicate whether something talks about a functional requirements, non-functional requirements, system user, current process or scope. Figure 5.51 shows how our classifier would take a transcript where the questions are already filtered and then add a label to that transcript, for the end-user to see.

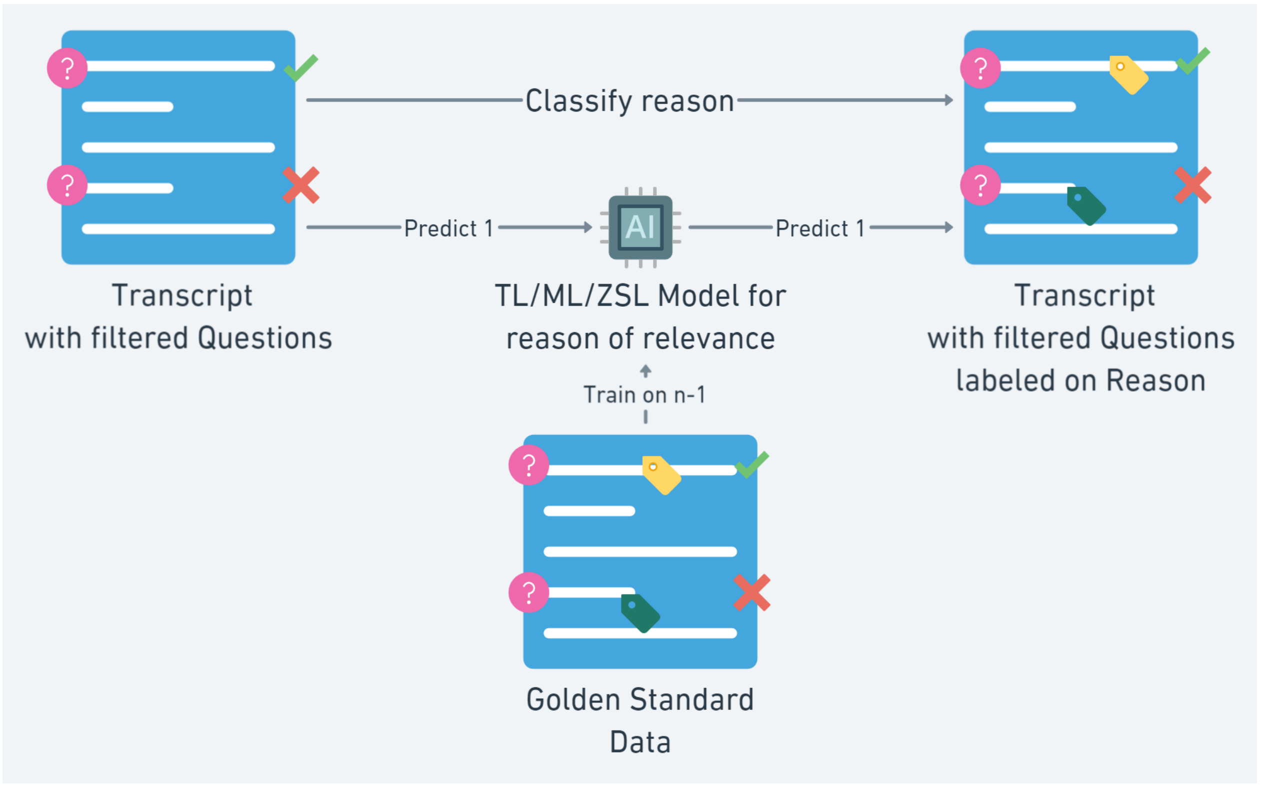

Classify reason
Predict 1
AI
Predict 1
Transcript with filtered Questions
TL/ML/ZSL Model for reason of relevance
Transcript with filtered Questions labeled on Reason
Train on n-1
Golden Standard Data

Figure 5.51: A flowchart to explain the flow of the continuation of our treatment designs. After the requirements-relevant questions are found, we can use a Machine Learning, Transfer Learning or Zero-Shot Learning model to classify the reason of requirements-relevance of these questions. This is done by training on n-1 conversations of the Golden Standard dataset and predicting on the one conversation we did not train on.

For Machine Learning, the same features were used when attempting to classify requirements-relevant questions (see Section 5.3.6). Similarly, a BERT-model was used for Transfer Learning. Finally, for Zero-Shot Learning, we take a large BERT model [109], trained on a Multi-Genre Natural Language Inference dataset [110]. To validate our treatment designs, we will compare these approaches to the human performance, as also done in Section 5.3.6. In this way, we can compare how well these classifiers perform, compared to the agreement among the participants on the reason. Similarly, we first validate the machine learning and transfer learning approaches using a stratified k-fold, after which we test the approach as shown in Figure 5.51.

Functional

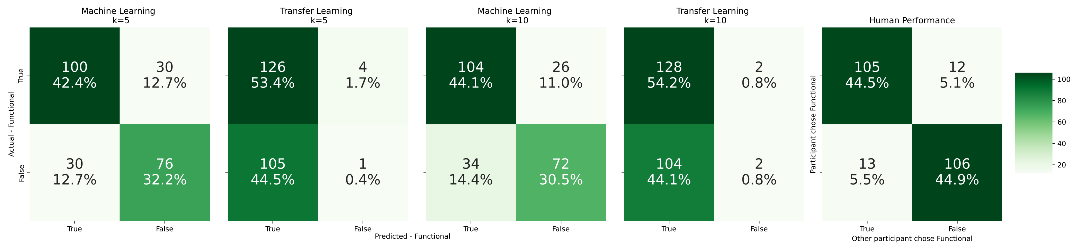


Figure 5.52: Confusion matrices for the Machine Learning and Transfer Learning with $k = 5$ and $k = 10$, compared to the human performance as tested when creating the golden standard. This confusion matrix focuses on the reason ‘Functional’

We can see in Figure 5.52, that the Machine Learning approach performs quite similar to the Human Performance, either with $k = 5$ or $k = 10$. Most likely, this performance will also be seen in the final test. For Transfer Learning it is obvious that it mainly predicts questions to belong to the ‘Functional’ category, as the amount of True Negatives and False Negatives is very low.

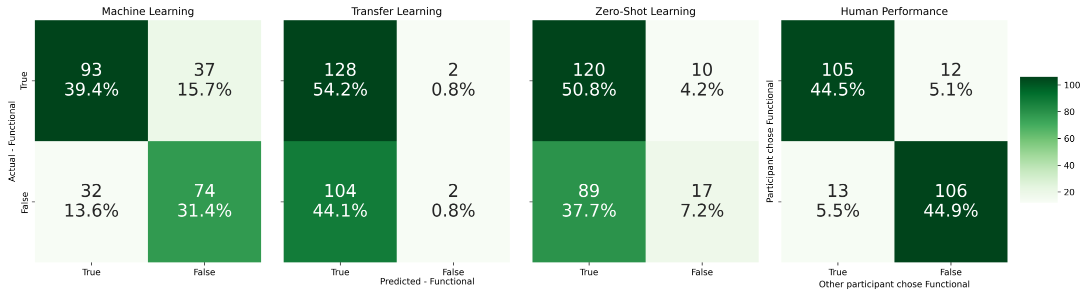


Figure 5.53: Confusion matrices for the Machine Learning, Transfer Learning and Zero-Shot Learning approach for finding the reason of relevance, compared to the human performance as tested when creating the golden standard. This confusion matrix focuses on the reason ‘Functional’

In Figure 5.53 we see the different confusion matrices for classifying whether a relevant question is relevant, because it’s discussing something functional or not. Transfer Learning and Zero-Shot Learning seem to simply classify most questions to be talking about something Functional. On the other hand, Machine Learning does quite a decent job here, which is comparable to the Human Performance.

Non-Functional

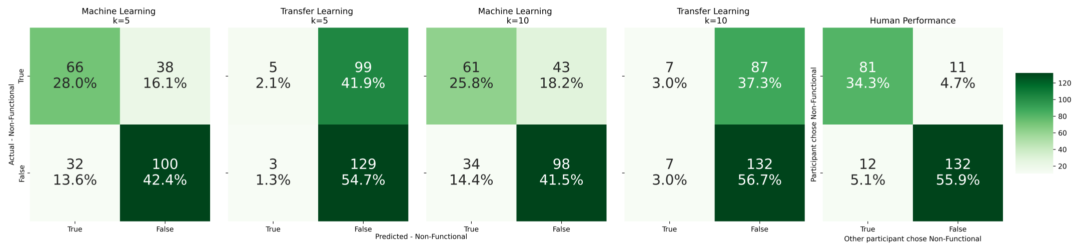


Figure 5.54: Confusion matrices for the Machine Learning and Transfer Learning with $k = 5$ and $k = 10$, compared to the human performance as tested when creating the golden standard. This confusion matrix focuses on the reason 'Non-Functional'

For the 'Non-Functional' category, we can see in Figure 5.54, that Machine Learning again performs quite similar to the Human Performance. Here, its performance worsens slightly with the increase from $k = 5$ to $k = 10$, but it is still very comparable. On the other hand, Transfer Learning predicts most questions to not belong to this category.

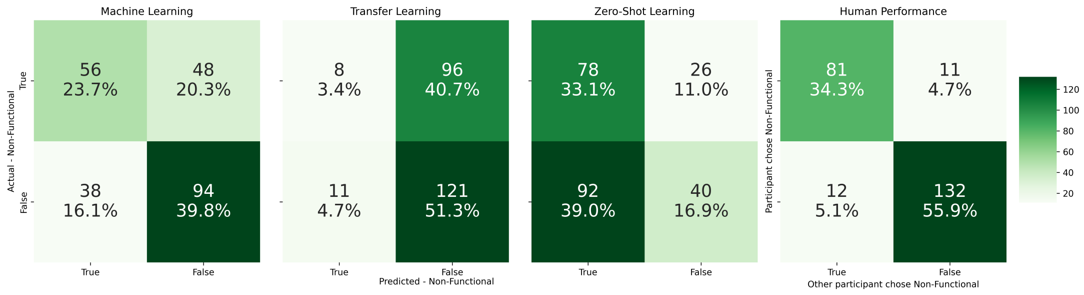


Figure 5.55: Confusion matrices for the Machine Learning, Transfer Learning and Zero-Shot Learning approach for finding the reason of relevance, compared to the human performance as tested when creating the golden standard. This confusion matrix focuses on the reason 'Non-Functional'

Figure 5.55 shows the confusion matrices for the category 'Non-Functional'. Here, again Transfer Learning and Zero-Shot Learning seem to classify the questions respectively only as not belonging to 'Non-Functional' and mostly belonging to 'Non-functional'. Machine Learning does a better job at this, whilst also creating a lot of False Negatives. Again, Machine Learning is most comparable to the Human Performance.

System Users

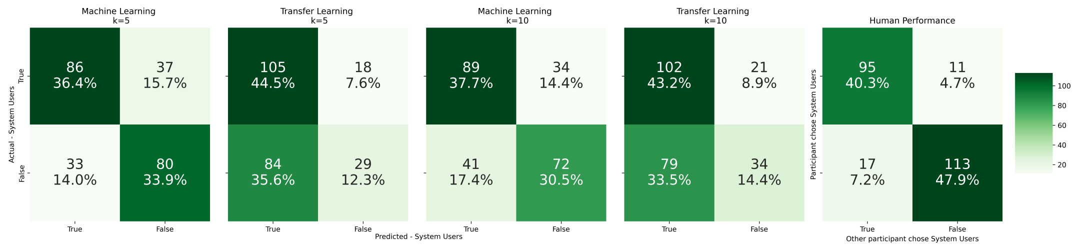


Figure 5.56: Confusion matrices for the Machine Learning and Transfer Learning with $k = 5$ and $k = 10$, compared to the human performance as tested when creating the golden standard. This confusion matrix focuses on the reason ‘System Users’

To validate the classification of questions about system users, Machine Learning performs quite well again. It worsens slightly when changing $k = 5$ to $k = 10$, but it is still comparable to the Human Performance. On the other hand, Transfer Learning predicts most questions to belong to this category, however its performance improves slightly when increasing $k = 5$ to $k = 10$.

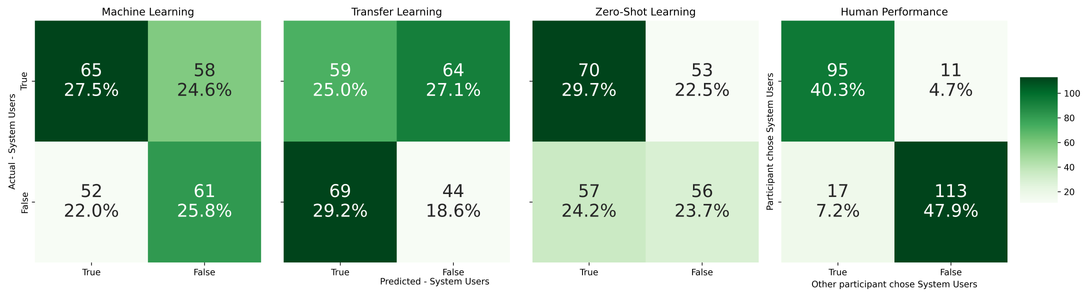


Figure 5.57: Confusion matrices for the Machine Learning, Transfer Learning and Zero-Shot Learning approach for finding the reason of relevance, compared to the human performance as tested when creating the golden standard. This confusion matrix focuses on the reason ‘System Users’

When it comes to classifying whether a question belong to ‘System Users’, Machine Learning, Transfer Learning and Zero-Shot Learning are comparable, as can be seen in Figure 5.57. They all classify a lot of the questions as belonging to ‘System Users’, whilst also creating a lot of False Positives and False Negatives. None are very comparable to the Human Performance.

Current Process

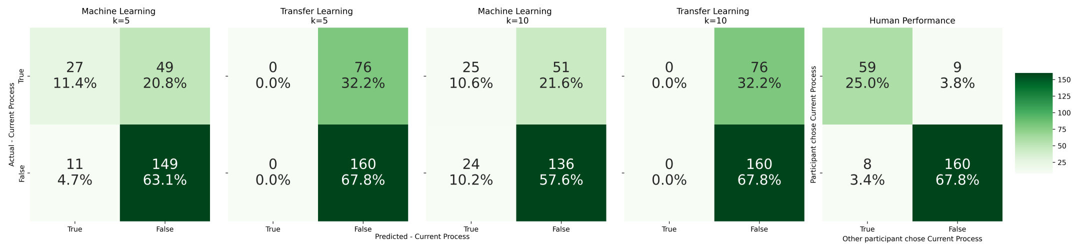


Figure 5.58: Confusion matrices for the Machine Learning and Transfer Learning with $k = 5$ and $k = 10$, compared to the human performance as tested when creating the golden standard. This confusion matrix focuses on the reason 'Current Process'

To validate the Machine Learning and Transfer Learning approaches for classifying whether questions belong to the 'Current Process' tag, we see in Figure 5.58 that none of these approaches work well. They all classify most questions not to belong to this class, which is not comparable to the Human Performance.

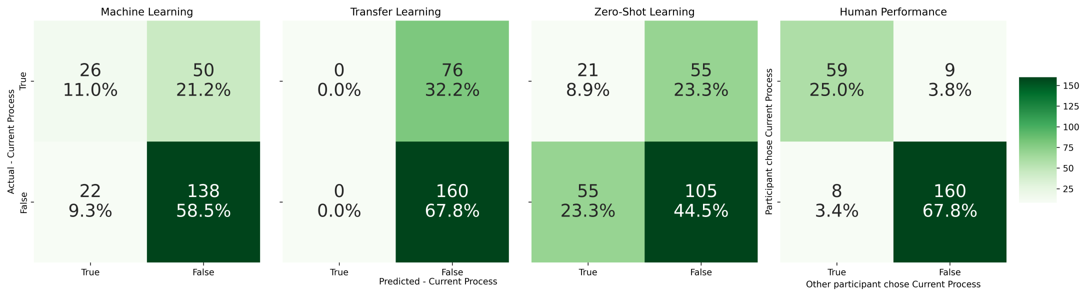


Figure 5.59: Confusion matrices for the Machine Learning, Transfer Learning and Zero-Shot Learning approach for finding the reason of relevance, compared to the human performance as tested when creating the golden standard. This confusion matrix focuses on the reason 'Current Process'

The confusion matrices in Figure 5.59 show how Machine Learning and Transfer Learning predict most questions to not belong to the 'Current Process' tag. This can be accredited to the low usage of the tag. Although Zero-Shot Learning does a more decent job at this tag, it is not comparable to the Human Performance.

Scope

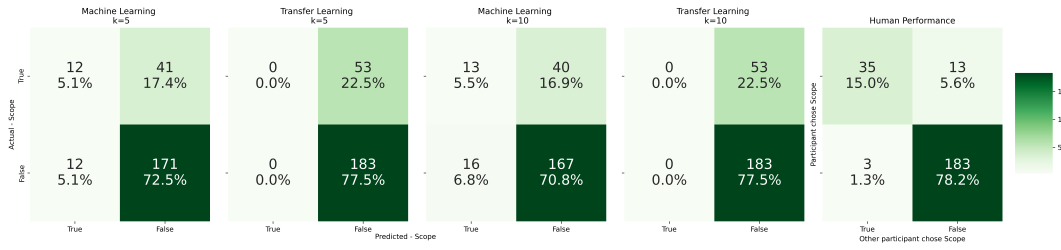


Figure 5.60: Confusion matrices for the Machine Learning and Transfer Learning with $k = 5$ and $k = 10$, compared to the human performance as tested when creating the golden standard. This confusion matrix focuses on the reason 'Scope'

Validating the final Machine Learning and Transfer Learning approaches, we see in Figure 5.60 that the approaches for classifying 'Scope' questions works very poorly. They all classify most questions not to belong to this category, and Transfer Learning even classifies all questions to not belong to this category. While the usage of this tag is low in the Human Performance, this is not comparable.

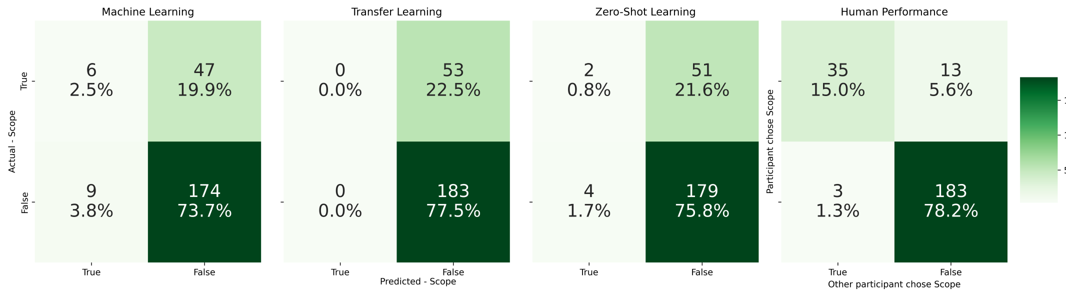


Figure 5.61: Confusion matrices for the Machine Learning, Transfer Learning and Zero-Shot Learning approach for finding the reason of relevance, compared to the human performance as tested when creating the golden standard. This confusion matrix focuses on the reason 'Scope'

Finally, Figure 5.61 shows the confusion matrices for classifying whether a question belongs to the 'Scope' tag. This tag is used very infrequently in the Human Performance, because of which all the classifiers predict most questions not to belong to this tag.

Overall

Because of the distributions of the tags and the limited amount of relevant questions these algorithms had to train on, the statistics found in Table 16 are not very impressive. In order to perform better, more tagged data is required. However, the Machine Learning classifier for predicting if a question belonged to the 'Functional' was quite comparable to the Human Performance, which is impressive for the amount of data.

	Precision			**Recall**			**F1-score**			**Support**		
Approach	*ML*	*TL*	*ZSL*	*ML*	*TL*	*ZSL*	*ML*	*TL*	*ZSL*	*ML*	*TL*	*ZSL*
Functional	71.5%	98.5%	92.3%	67.4%	55.2%	57.4%	69.4%	70.7%	70.8%	138	232	209
Non-functional	58.7%	6.7%	75.0%	56.0%	38.9%	45.9%	57.3%	11.5%	56.9%	109	18	170
System users	52.8%	48.0%	56.9%	56.0%	46.1%	55.1%	54.4%	47.0%	56.0%	116	128	127
Current process	28.9%	0.0%	27.6%	55.0%	0.0%	27.6%	37.9%	0.0%	27.6%	40	0	76
Scope	13.2%	0.0%	3.8%	30.4%	0.0%	33.3%	18.4%	0.0%	6.8%	23	0	6
Weighted avg.	56.0%	77.0%	70.4%	58.2%	51.3%	49.5%	56.5%	59.9%	57.4%	426	387	588

Table 16: Precision, recall, f1 and support on different approaches for classifying the reason of relevance.

6 Discussion

In this research, we went through different iterations of Wieringa's design science research methodology [11], in order to design an approach for finding requirements-relevant information in requirements elicitation interviews. This started with two iterations for finding relevant questions in Sections 5.1.1 and 5.1.2. After which we also had two iterations for filtering requirements-relevant questions in Sections 5.3.1 and 5.3.2. In order to evaluate this treatment design, we created a Golden Standard on our dataset of transcripts, which can be found in Section 5.2.6. We enabled different participants to label the data for us. This showed us how well our tagging scheme performed, since this taught us what the tagger perceived. It removed our biased view, giving an unbiased understanding of relevance. Furthermore, the tagging allowed us to not spend a considerable amount of time labeling, while making our goal more clear and opening a discussion about the disagreements among these participants.This required to finely line out what the scope of our project is and what requirements relevance means and took four iterations of design, found in Sections 5.2.1, 5.2.3, 5.2.4 and 5.2.5. We analyzed how the use of different Natural Language Processing (NLP) techniques compared, in Sections 5.3.4 and 5.3.5. Next to that, we attempted to do this identification of relevant questions using Artificial Intelligence in Section 5.3.6. Finally, we attempted to classify the reason of relevance, using this data as a potential way of categorizing our relevant questions in Section 5.4. This chapter discusses our research questions, the limitations and possibilities for future work.

6.1 Answering the sub research questions

SQ1: What are state of the art techniques for identifying requirements-relevant information in conversations?

In Chapter 4 we have discussed several related works. On a conceptual level, there were a couple of different works. The first being a study on the categorization of information in transcripts of Fit-Gap Analysis elicitation sessions [79]. This empirical investigation provided a list of keywords that could identify these different categories in a conversation. Furthermore, another study was done on requirements elicitation sessions where the concepts were extracted, with the ability to identify known and unknown concepts based on an ontology [50]. This provided insight into the design of a prototype NLP tool on transcripts.

When it comes to techniques, other related works show different approaches to categorize requirements [87, 88, 89, 90, 91, 92], demarcate requirements from free-form text [49] or extract declarative process models from natural language [93]. These works gave an idea of how to create our NLP approach and what challenges are to be expect.

While we were able to identify these different techniques and concepts, new progress had to be made on the identification of requirements-relevant information in transcripts and the classification of that information.

SQ2: How to design an approach for identifying requirements relevant information?

As the literature suggested [18], by focusing on the questions asked by the Interviewer, we were able to identify questions that ask for requirements-relevant information from the Interviewee. This could be done automatically using several Natural Language Processing (NLP) techniques. For the detection of questions, given the limitations of the language created from an automatic transcription, we were able to test both Speech Act classification and Part of Speech tags. Furthermore, when combining these approaches with the most important words found using Term Frequency-Inverse Document Frequency on either the transcript or a context document, we were able to identify the requirements-relevant questions. Finally, with the help of a dataset of tagged conversations we were able to train models that could categorize these requirements-relevant questions into five categories.

SQ3: What is the expected effectiveness of the approaches from SQ2?

In order to estimate the expected effectiveness of these approaches, we created a tagged dataset, by having pairs of taggers tag our questions and answers to be requirements-relevant or not, given a certain reason.

These conversations were taken from a requirements engineering course, given at Utrecht University. This tagged dataset allowed us to get an insight into the performance of our approaches. For the detection of questions, it became clear that Speech Act classification worked better than Part of Speech tags for detecting the questions in our Transcripts, as can be seen in Section 5.3.4. Using Speech Act classification we were able to identify the questions with a precision 80%, recall of 90% and accuracy of 90%. Furthermore, when adding the identification of requirements-relevant information to this, there was no difference in performance when it came to using Term Frequency-Inverse Document Frequency on the transcript or the context document. In Section 5.3.5 it can be seen that with the use of Speech Act classification and TF-IDF on a context document, were able to identify the requirements-relevant questions with a precision of 70%, recall of 80% and accuracy of 66%. Finally, for the classification of the requirements-relevant information we tested Machine Learning, Transfer Learning and Zero-Shot Learning models for this task. However, none of these approaches turned out to be great classifiers, with the weighted average F1-score under 60%, most likely due to the lack of quality data to train on. While some categories seemed to have a decent classifier, this was mostly due to the high imbalance in the use of this category. In our results, the Machine Learning approach performed most comparable to the human performance.

SQ4: How can domain information improve our approach?

By using a context document in our approach for determining whether a question is relevant or not, we tested the usage of domain information. For our approaches, there was no significant difference in the outcome, as show in the answer to SQ3. However, during the creation of the tagged dataset, participants noted that the domain knowledge they gathered throughout the tagging exercise helped their understanding of the transcript. This allows us to believe that our testing is inconclusive, assuming that our approach tries to mimic this human behaviour of gathering domain knowledge in order to classify the requirements-relevance better.

6.2 Answering the main research question

Given the answers to the sub research questions above, we can now answer the main research question of this thesis:

MQ: How can we identify requirements relevant information in a transcript from a conversation between a business analyst and a stakeholder aiming to elicit requirements?

Literature has shown that there is limited work on transcriptions of requirements elicitation interviews, as seen in SQ1. Therefore, it was necessary to create a new approach to identify the requirements-relevant information in these transcripts. Given the fact that these interviews are “A communicative event in which the interviewers asks questions to reach the reality of a phenomenon conceived inside the mind of the interviewee.” [18], we have focused on the questions asked by the Interviewer, especially the ones concerning requirements-relevant topics, as discussed in SQ2. These questions were identified best using Speech Act classification, since this technique was able to identify many of the questions asked, compared to Part of Speech tags. Furthermore, by using Term Frequency-Inverse Document Frequency (TF-IDF) we were able to identify the words that could indicate the requirements-relevant topics of information. Here, the use of a context document did not perform better than using the transcript itself, as shown in SQ4. By creating a Golden Standard on what questions are relevant, we were able to evaluate these approaches, giving us a precision of 70%, recall of 80% and accuracy of 66% for identifying requirements-relevant questions in an automatically generated transcript using Speech Act classification and TF-IDF on our context document, as seen in SQ3. Finally, by training on the reason of relevance, we were able to train models to classify these reasons, using Machine Learning, Transfer Learning and Zero-Shot Learning. None of these classifiers showed significant results, due to the lack of data, but Machine Learning performed best.

By using showing the requirements-relevant questions asked in the transcript, we can give a quick and concise overview of the transcript, since this is only 20% of the conversation. By allowing the user to have a drop-down view on the question, revealing an answer would enable the user to be able to quickly traverse the transcript and get the information they need.

6.3 Limitations

This research has a number of limitations, which we present according to four categories: conclusion validity, internal validity, construct validity and external validity [111].

6.3.1 Conclusion validity

Conclusion validity is concerned with the relationship between the treatment and the outcome, according to Wohlin et al. [111]. In order to draw conclusion correctly, we have compared our different approaches to our golden standard. This golden standard was created by taggers, one pair of taggers per conversation. However, this came along with some disagreements among these participants. Therefore, we did not only compare our approaches to this golden standard, but also to the human performance on the tagging of the conversations. For our learning approaches, we used a stratified k-fold for validating after which we tested on each conversation. This made sure we did not draw any ungrounded conclusions on the results. Furthermore, when comparing our approaches for classifying the reason of relevance, the human performance was able to show us the over- or underpopulation of certain categories. This allowed us to get an insight into why our classifiers took a certain decision. Next to that, for the evaluation of our approaches while we looked at the human performance, we also turned to metrics such as precision, recall and f1-score. However, in order to measure how useful this tool can be in practice, other experiments have to be done.

6.3.2 Internal validity

Threats to the relationship between the treatment and the outcome fall under internal validity [111]. In the final version of the tagging scheme that was used for the golden standard tagging, it was known that the list of reasons for why something is considered requirements-relevant was non-exhaustive. Therefore, some questions and answers will be misclassified, as the taggers could not find a fitting category according to the list of reasons. The results showed that the taggers often did not agree on the reason of relevance. It can be said that the relevance of a question was often determined by different factors and perhaps the participants determined the relevance because of different factors.

Furthermore, the location of the requirements-relevant information could be chosen for different reasons; either/both the question could be answered with requirements-relevant information or/and the answer contained requirements-relevant information. This allows the taggers to agree on the requirements-relevance for different reasons, as they do not indicate where in this question or answer they can find the requirements-relevant information indicators.

Next to that, the quality of the automatic transcription played a role in the output of our approaches and the outcome of this research. By creating the golden standard, a lot of manual adjustments have been made in order to make the transcript understandable for the taggers. This understandability of the transcript played a big role in the outcome of the tagging. These adjustments most likely also played a big role in the ability for our approaches to classify the questions and answers, requirements-relevance and reason correctly.

Finally, the taggers were shown different questions in chronological order, allowing them to build up knowledge on the domain as they progressed through the questions. This allowed for external reasoning, next to the context that is provided. This external reasoning would influence our outcome without a visible cause.

6.3.3 Construct validity

When it comes to the relationship between the theory and observations, we talk about construct validity [111]. As our literature research showed, there is not a lot of research that of requirements engineering that focuses on transcripts of conversations. Therefore, it is unclear how the theory translated to our observations.

More specifically, we focused on requirements elicitation interviews that have a system description. This was one very specific type of conversation, where the interviewers had nothing but the knowledge that could be gathered from the system description.

The amount of data that was used for our learning approaches and the validation of our heuristics was limited. This makes the reflection of the outcome on the effect debatable.

However, the conclusion still holds that we can summarize the transcript using our approach for finding relevant questions.

6.3.4 External validity

The external validity concerns the generalization of the treatment [111]. When investing the usage of Natural Language Processing for identifying questions, it became clear that the quality of the transcript can determine how well the approach performs. We were able to cope with the missing question marks, but some of the manual adjustments made to the transcript an algorithm would not have been able to tackle. These transcription errors varied from severity to the outcome, but more of these errors will worsen the outcome.

Furthermore, we have tested our algorithms on a dataset containing recordings of requirements elicitation interviews from a Requirements engineering course , given at Utrecht University. These interviews were conducted by students, who were taught according to the guidelines of the course to conduct this interview. How this translates into real-world cases is to be seen.

The generalization of the reason categorization is debatable as well. On one hand, the categories are very road and we train our classifier on different domains, but this training data influences the outcome. Therefore, a completely different domain could be classified completely wrong. Hopefully, with an enlarged set of tagged conversations on different domains, the classifier will instead of recognizing specific words, start recognizing patterns in the text. Such as 'the system should' could refer to a functionality, and 'privileges' could refer to system users. Words like 'currently' could refer to the current process and 'speed' or 'security' could mean it has something to do with non-functionalities. However, in the current stage of the tool, with the current set of conversations to train on, the generalization of this categorization is debatable.

6.4 Future work

While this thesis presents an approach for highlighting requirements from elicitation interviews, our scope was fairly narrow. It would be interesting to investigate other ways of exploring such a transcript, as not much work has been done on transcriptions. Challenges rely on the quality of the transcription, the availability of data and the quality of the conversation.

Furthermore, real-world scenario's should be tested and evaluated. Applying the tool in the process of Requirements Engineering and evaluating the effectiveness by for example A/B-testing or surveying the usefulness after usage of the tool.

Next to that, there is much to be investigated about this artifact of transcriptions. As we turn into a digital era, more and more effective communication will be held online, which will be recorded and can be transcribed. Even outside of the scope of Requirements Engineering, there are most likely a lot of useful tools to be designed.

When it comes to the use of Artificial Intelligence for categorizing speakerturns in transcripts, based on tagged data, we have seen that more data is required. It would be interesting to see if with more data these approaches can outperform the heuristic approaches and at what amount of data this would happen.

Work can be done on highlighting where miscommunication takes place. Perhaps focus sing on detecting ambiguity or misunderstandings can help requirements engineers improve their interviewing techniques, as there is often no timely feedback to the elicitation interviews.

A Feature matrices for automated demarcation

ID	Feature (Type)	Description (D) & Intuition (I)
(a) Token-based features		
Tok1	numTokens (Numeric)	(D) Number of tokens in a requirement candidate. (I) A requirement candidate that is too long or too short could indicate a non-requirement.
Tok2	numAlphabetics (Numeric)	(D) Number of alphabetic words in a requirement candidate. (I) Few alphabetic words in a requirement candidate could indicate a non-requirement.
Tok3	numOneCharTokens (Numeric)	(D) Number of one-character tokens in a requirement candidate. (I) Too many one-character tokens in a requirement candidate could indicate a non-requirement, e.g., section headings.
Tok4	startsWithId (Boolean)	(D) TRUE if a requirement candidate starts with an alphanumeric segment containing special characters such as periods and hyphens, otherwise FALSE. (I) Alphanumeric segments with special characters could indicate identifiers for requirements.
Tok5	startsWithTriggerWord (Boolean)	(D) TRUE if a requirement candidate begins with a trigger word (“Note”, “Rationale”, “Comment”), otherwise FALSE. (I) A trigger word at the beginning of a requirement candidate is a strong indicator for a non-requirement.
Tok6	hasUnits (Boolean)	(D) TRUE if a requirement candidate contains some measurement unit, otherwise FALSE. (I) According to several domain experts consulted throughout our work, the presence of measurement units increases the likelihood of a requirement candidate being a requirement.
(b) Syntactic Features		
Syn1	hasVerb (Boolean)	(D) TRUE if a requirement candidate has a verb per POS tags, otherwise FALSE. (I) A requirement candidate without a verb is unlikely to be a requirement.
Syn2	hasModalVerb (Boolean)	(D) TRUE if a requirement candidate has a modal verb, otherwise FALSE. (I) The presence of a modal verb is a good indicator for a requirement candidate being a requirement.
Syn3	hasNPModalVP (Boolean)	(D) TRUE if a requirement candidate contains a sequence composed of a Noun Phrase (NP) followed by a Verb Phase (VP) that includes a modal verb, otherwise FALSE. (I) The intuition is the same as that for Syn2. Syn3 goes beyond Syn2 by capturing the presence of the NP preceding a modal VP. This NP typically acts as a subject for the VP.
Syn4	startsWithDetVerb (Boolean)	(D) TRUE if a requirement candidate, excluding head alphanumeric patterns / trigger words, begins with a pronoun or determiner followed by a verb, otherwise FALSE. (I) This is a common natural-language construct for justification and explanation, and thus could indicate a non-requirement.

Table 17: Features for learning in the paper by Abualhaija et al. [49]

ID	Feature (Type)	Description (D) & Intuition (I)
(b) Syntactic Features		
Syn5	hasConditionals (Boolean)	(D) TRUE if a requirement candidate has a conditional clause, otherwise FALSE. (I) Conditional clauses are more likely to appear in requirements than non-requirements.
Syn6	hasPassiveVoice (Boolean)	(D) TRUE if a requirement candidate has passive voice through some dependency relation, otherwise FALSE. (I) Requirements not containing modal verbs may be specified in passive voice.
Syn7	hasVBToBeAdj (Boolean)	(D) TRUE if, in requirement candidate, there is some form of the verb "to be" appearing as root verb followed by an adjective, otherwise FALSE. (I) The pattern described is more likely to appear in requirements.
Syn8	isPresentTense (Boolean)	(D) TRUE if a requirement candidate has some root verb which is in present tense, otherwise FALSE. (I) Sometimes, requirements are written in present tense rather than with modal verbs.
(c) Semantic Features		
Sem1	hasCognitionVerb (Boolean)	(D) TRUE if a requirement candidate has some verb conveying reasoning or intention, otherwise FALSE. (I) Reasoning and intention are a common characteristic for non-requirements.
Sem2	hasActionVerb (Boolean)	(D) TRUE if a requirement candidate has some verb conveying motion or change of status, otherwise FALSE. (I) Action verbs are common in requirements for describing behaviors and state changes.
Sem3	hasStativeVerb (Boolean)	(D) TRUE if a requirement candidate has some stative verb, otherwise FALSE. (I) Stative verbs are common in requirements for describing system properties.
(d) Frequency-based Features		
Frq1	idPatternFrequency (Enumeration)	(D) Maximum frequency level (high, medium, low) associated with the identifier pattern with which a given requirement candidate starts. If a requirement candidate does not start with an alphanumeric pattern, the returned value is NA (not applicable). (I) A frequent id pattern in a requirement candidate is likely to signify a requirement. This is because alphanumeric are prevalently used for marking requirements.
Frq2	hasMFModalVerb (Boolean)	(D) TRUE if a requirement candidate contains the most frequent modal verb of the RS, otherwise FALSE. (I) While a consistent application of modal verbs cannot be guaranteed, the most frequent modal verb is a strong indicator for requirements.
Frq3	hasHFNP (Boolean)	(D) TRUE if a requirement candidate contains a highly frequent (top 1%) NP in the RS, otherwise FALSE. (I) Highly frequent NPs (after stopword removal) often signify core concepts, e.g., the system and its main components. These concepts are more likely to appear in requirements.

Table 18: Features for learning in the paper by Abualhaija et al. [49]

B System descriptions

REQUIREMENTS ENGINEERING 2019/2020: SYSTEM A

INTERNATIONAL FOOTBALL ASSOCIATION (IFA) PORTAL

ORGANIZATION

The International Football Association (IFA) aims at managing the various leagues, scheduling the games and the referees, controlling and auditing the teams' budget, notifying the many stakeholders including fans, IFA administration, teams, and referees about important events, and providing statistics on games, players, teams, coaches, etc. To increase the level of service with respect to the issues mentioned above, the IFA decided to re-structure its core information systems.

The IFA has decided to develop a new operational and analytics system to better serve the stakeholders. Your company is asked to consider the development of such a system. You need to go and talk to Mr. Sturm, the chief information officer of the IFA who is in charge of leading the entire reform.

AS-IS SITUATION

The current operational systems do not well support many of the values the IFA would like to promote. These include the following: (1) transparency, mainly when referring to budget auditing; (2) automation in scheduling, mainly to avoid biases; (3) better support for the fans that are actually the driving force beyond the IFA; (4) unauthorized data collection of the events occur in the games; and (5) communication with the various stakeholders is inefficient as it is done using irrelevant media (e.g., notify referees about the cancelation of the game over a phone call and notify the same for the fans over radio broadcasts).

Indeed, the IFA manages its operation by various means, including several systems that are isolated and thus caused inconsistencies and redundant data, which lead to problems in achieving the IFA goals. For example, the teams manage their own systems and deliver written reports to the IFA, thus limiting the auditing capabilities of the IFA. Currently, the IFA does not support real time notifications and updates that are required for the fans. In addition, referees are notified about their schedule without considering their constraints and thus re-scheduling occur regularly. Nowadays, the entire operation is done by the IFA administration and consume much of their time, limiting the IFA expansion.

VISION

It is clear to the IFA management and the CIO that there is a need for a complete change to the operational systems and infrastructure of the IFA, so to move its entire operation into the digital era. The goal of this infrastructure is to better engage the stakeholders into the IFA activities. For that purpose, the user experience using the new facilities for each of the stakeholders should tremendously be improved. It is expected that 50,000 people could access the system at any time and the response time should be fast. Furthermore, the infrastructure should serve people around the globe. Furthermore, the infrastructure should allow the support for variability in many aspects such as rules for budget management and policies for determine the teams' placement within the league. The IFA is fully aware on its low ability to maintain and operate that infrastructure, as it is not a software company, and thus requires the infrastructure to cope with that constraint.

The system should delegate responsibilities for each of the stakeholders. For example, teams will need to manage their own resources: stadiums, players, coaches, budget, transaction, etc. This will be, of course, monitored by the IFA administration. Referees will write the game report using the system, preferably in real-time, using the appropriate technologies. Fans are usually information consumers, yet they will be able to add comments, photos, and articles about the teams, players, etc. The leagues status should be available at any time.

Mr. Sturm is willing to provide you with additional information through a skype or hangouts interview. He has some ideas himself that he would like to share too; he is certain that his ideas can help you shape the system to be built.

REQUIREMENTS ENGINEERING 2019/2020: SYSTEM B

URBAN MOBILITY SIMULATOR

ORGANIZATION

The Municipality of Wonderland (MoW) is facing serious problems with traffic congestions within the city, both in the city center and in the city ring. This is leading to increasing dissatisfied citizens, who are confronted with endless lines to reach their destination, especially during peak hours. Furthermore, to make things more difficult, activists started arguing about the adverse effects of traffic congestions on the environment. MoW's urban mobility department has decided to purchase an urban mobility simulator to investigate and improve the situation, and the company you are working has been contracted to build the system. First things first: you need to go talk to Mr. Dalpiaz first, the head of their urban planning department.

AS-IS SITUATION

Mr. Dalpiaz has already provided you with a high-level overview of how the department is currently deciding on the traffic planning. He wrote that they continuously experiment with micro-level management: whenever a problem is encountered, the department opens a project that focuses on improving that problem. For example, recently, the Northern section of the ring, which surrounds the entire city, had a serious issue during the peak hours, and they have decided to introduce adaptive traffic lights to reduce the inflow of cars from the North West entrances of the ring. This decision was taken via a focus group with three citizens, two police officers, a professor from a technical university, an urban mobility planner from a nearby city of approximately the same size of Wonderland, the vice-mayor, and four members of the department itself. During the workshop, they have evaluated different solutions that were proposed by the various people who were involved in the focus group. In case of urgencies, instead, the decisions are typically taken internally by the department, and the major source of inspiration consists of replicating solutions that the people in the department have seen in other towns or cities.

Unfortunately, it seems like the effectiveness of these techniques is low and there seems to be no easy way out. The department, however, has equipped the city with a whole number of sensors, and there is now a knowledge base that represents traffic flows for around three weeks. The department has recruited two smart mathematical modeling people. They created some models that seem to be able to reduce traffic congestions, at least in theory. However, there are serious doubts on the environmental friendliness of the proposals that were made.

VISION

After a thorough discussion with these mathematical modelers, it became clear that their models work well to improve flows, but they are not geared toward reducing pollution, noise, and they do not provide a fair treatment to all the areas of the city. A more interactive solution is necessary. Mr. Dalpiaz was told to look into urban traffic simulators, which seem to be much more powerful, customizable, and easier to use by the urban mobility planners. He said he would like a reliable system; he can't afford, financially, to pay for the construction of a system from scratch. However, he has heard that several platforms exist, although they do not readily support the creation of alternative scenarios and the identification of solutions that minimize pollution levels and keep noise levels under control. He would like a platform that can be operated independently by the people in the department, without the necessity of asking an expert to execute a simulation. Furthermore, he wants this simulator to execute on premise, for cloud-based solutions are not allowed by the municipality regulations. Finally, the simulator should have a simple way to embed real-time data to make sure the executed simulation regards the current situation in the city, and not an old one. However, it should also be possible to take snapshots of specific days, or parts of the day, and to execute simulations against those scenarios.

Mr. Dalpiaz is willing to provide you with additional information through a face-to-face interview in his office. He has some ideas himself that he would like to share too; he is certain that his ideas can help you shape the system to be built.

REQUIREMENTS ENGINEERING 2019/2020: SYSTEM C

HOSPITAL MANAGEMENT SYSTEM

ORGANIZATION

The University Hospital of Pediatric Excellence (UHOPE) is the main hospital in the city of Medz and is affiliated with the Medz University. Since the foundation of the university in 1612, an academic hospital has existed in various forms. Today, UHOPE Medz comprises an academic hospital, the faculty of Medicine, and the children's hospital. With approximately 7,600 employees, UHOPE is one of the biggest employers in the region. Patient safety is UHOPES number one priority. To keep track of its day-to-day activities and patient records, the hospital uses more than 32 different management systems that were implemented independently during the last decades, confronting the hospital with various challenges. In addition, rising costs and mounting privacy issues add to these challenges. In order to ensure the highest level of patient safety and to keep up with the ever-evolving digitalization of healthcare, the hospital administration decided to invest in a new hospital management system. The company you are working for has been contracted to implement a new hospital management system to integrate a majority of the components currently provided by over 32 different systems.

AS-IS SITUATION

UHOPE consists of many different departments, including Emergency, Pediatrics, Anesthetics, Laboratory, Neurology, Pathology, Human Resources, etc. Furthermore, a variety of stakeholders are to be considered in a hospital. Next to the doctors there are nurses, receptionists, administration employees, janitors, researchers, and the patients. As described above, the hospital currently uses various different systems for appointment scheduling, room scheduling, administration of hospital staff, patient data management, laboratory, and so on. In addition, a lot of things are still done on paper, i.e., writing prescriptions or transferals. This makes hospital management not only very expensive and discontinuous but also highly prone to errors. For example, the reception is equipped with a simple appointment scheduling system that requires patients to call or physically appear for scheduling their appointments. Furthermore, patient data is stored in a local database for each of the departments. A referral to another department requires to manually transfer the patient's medical record, causing inconsistencies and redundancies that can affect physicians' productivity and, ultimately, patient safety.

Healthcare is a traditionally slow adapting industry, process- and paper-heavy, and resistant to change. Furthermore, most stakeholders are not IT experts. Therefore, the hospital is looking for a user-friendly, integrated solution of a central hospital management system to increase interoperability and efficiency of patient care with the ultimate goal of maintaining the highest level of patient safety.

VISION

The hospital's enterprise architect Ms. Gieske was assigned to take over the project management of implementing the core hospital management system. With her enthusiasm for leveraging ubiquitous technology within healthcare her focus is on 1) providing a continuous, interoperable management system facilitating the electronic exchange of information across departments and beyond, 2) allowing data collection for both prescriptive as well as predictive analytics of clinical data, 3) empowering the patients by including them in their healthcare management, 4) ensuring privacy, data security, and compliance to regulatory processes, and 5) fostering the integration of mobile devices into the core management system.

There is no need to create a system from scratch since there are various hospital management systems available on the market. Instead, the challenge is to find a system that can be tailored to UHOPE's unique requirements and needs. Ms. Gieske is willing to provide you with additional information through a face-to-face interview in her office.

C Speech Acts

For the approach for finding questions using Speech Acts, we were able to use 28 out of 38 available tags. Examples of the identified tags in the transcripts can be found in Table 19. Furthermore, Table 20 contains the example usage of the tags there were not identified in our transcripts.

Tag	Conversational Example
Statement-non-opinion	And it was also mentioned that these sensors, they are not able to or they are not an environmentally friendly solution.
Acknowledge (Backchannel)	Yeah. Indeed.
Statement-opinion	It's not so bad.
Agree/Accept	Yes, absolutely.
Appreciation	Yes. Mhm.
Yes-No-Question	**Is there is there already some data that can be gathered from the existing systems that can already be put in The new one or is there no,**
Yes answers	Um Yes.
Conventional-closing	Thank you. Thank you very much.
Wh-Question	**I'm gonna ask you, how long does it take for that person to analyze the situation and uh monitor a certain road or urban traffic situations.**
No answers	Oh no no no no.
Hedge	Oh yes, I don't know how but that is so important.
Declarative Yes-No-Question	**So it so it would be a manual change, not a new iteration of the automated schedule.**
Other	Okay.
Backchannel in question form	**Um, this should also be made available I imagine during a match for instance, the score of the match should be updated immediately once it's changed. Right?**
Quotation	Okay that's not a priority.
Summarize/reformulate	Okay. Okay. So basically the referees have constraint and these have to be taken as input (...)
Affirmative non-yes answers	Yeah as little as possible.
Action-directive	Go ahead. Go ahead.
Collaborative Completion	Building maintainers. Janitors?
Repeat-phrase	Yeah. Janitors!
Open-Question	**What do you mean with local I. F. A. ?**
Rhetorical-Questions	**Okay, so the idea is, how do I get the first event that occurs during the game, for example?** (...)
Hold before answer/agreement	Yeah I think other than that let me just recap.
Negative non-no answers	Mm Not necessarily, no.
Signal-non-understanding	Sorry?
Conventional-opening	Oh good morning, good morning, good morning, how are you? Hello,
Or-Clause	**So you you think there should be the same rights for every user of the system? Or do you think that one user should have less rights capable?**
Self-talk	Um what else they can do.

Table 19: Example usage of the all of the Speech Act Tags that were found in our transcripts.

TAG	EXAMPLE
Uninterpretable	But, uh, yeah
Response Acknowledgement	Oh, okay.
Dispreferred answers	Well, not so much that.
3rd-party-talk	My goodness, Diane, get down from there.
Offers, Options Commits	I'll have to check that out
Downplayer	That's all right.
Maybe/Accept-part	Something like that
Tag-Question	Right?
Declarative Wh-Question	You are what kind of buff?
Thanking	Hey thanks a lot

Table 20: Example usage of the Speech Act Tags from `https://github.com/bhavitvyamalik/DialogTag` that were not found in our transcripts.

D Transcription Errors

INTERVIEWEE : I think it's fine if it is just on a mobile version in terms of you access it via the browser. **Okay,**

INTERVIEWER : **so web base is the**

INTERVIEWEE : **reference.** Yeah. We're not putting the focus yet on Absolutely perfect. But maybe if that's possible. I mean if there's an option for it. So in the future we might consider developing an app that will be great if we can somehow make it possible that this is a possible extension. Okay. Yeah.

(a) An example of a part of a transcript where the speakerturns are not ending properly.

INTERVIEWEE : I think it's fine if it is just on a mobile version in terms of you access it via the browser.

INTERVIEWER : **Okay, so web base is the preference?**

INTERVIEWEE : Yeah. We're not putting the focus yet on apps. But maybe if that's possible. I mean if there's an option for it. So in the future we might consider developing an app that will be great if we can somehow make it possible that this is a possible extension.

(b) After manually adjusting the endings of the speakerturn, the transcript looks like this.

Figure 4.62: Adjusting the ending of a speakerturn

For our algorithms to correctly identify the questions, we need the questions to be in one single speakerturn and not overlap with different speakerturns as is the case in the example in Figure 4.62. When this is not the case, we would miss out on this question and not have this information available to the requirements engineer.

INTERVIEWEE : Um Are you referring to the **predictive**

INTERVIEWER : **and the analysis**

INTERVIEWEE : **part?** Um I would not put a focus on this.

(a) An example of a part of a transcript the speakerturn is ended, but shouldn't be. 'Um Are you referring to the predictive and the analysis part?' should be one question from the interviewee.

INTERVIEWEE : Um Are you referring to the **predictive and the analysis part?**

INTERVIEWER : Yeah yeah.

INTERVIEWEE : Um I would not put a focus on this.

(b) After manually adjusting the this erroneous change of the speakerturns, the transcript looks like this.

Figure 4.63: Adjusting a erroneous change of speakerturns

Similar to the example in Figure 4.62, the question in the example in Figure 4.63 is separated over three speakerturns as well, making us most likely miss the entire question. In this case, it is due to a change of speakerturns that was not necessary, the interviewee keeps talking during the entire part that was transcribed. This error is probably due to the “Yeah yeah.” that the interviewer said.

INTERVIEWER : **The referees mainly can look at the game. Is scheduling too?** Uh huh. Uh Can report on the events during the game.

(a) An example of a part of a transcript where the sentence is ended at a wrong point. Here the part of the speakerturn 'The referee mainly can look at the the game. Is scheduling too?' should be one sentence, while it is transcribed as two different ones here, even the last being a question.

INTERVIEWER : **The referees mainly can look at the game they are scheduled to.** Uh huh. Uh Can report on the events during the game.

(b) After manually adjusting the sentence being split in the speakerturn, it looks like this.

Figure 4.64: Adjusting a sentence that was erroneously split into two sentences.

In the example in Figure 4.64 a sentence is erroneously split. This can be an issue when a question mark is placed after a sentence that was not meant to be questioning. Therefore this could be identified wrongly as a question, creating a false positive for our algorithm. Furthermore, this makes the sentence understandable to the reader and allows us to tag the sentences accordingly.

INTERVIEWEE : So this is like yeah this brings us to the next question which is what is what do you expect the types of inputs and outputs you expect the system to have like okay a medical record is what type of input you want to be able to add it to have your own vision of the client or the patients record. **So what could also be possibly another input to the system.**

(a) In this speakerturn, the final sentence is a question, but the question mark is omitted in the transcript.

INTERVIEWEE : So this is like yeah this brings us to the next question which is what is what do you expect the types of inputs and outputs you expect the system to have like okay a medical record is what type of input you want to be able to add it to have your own vision of the client or the patients record. **So what could also be possibly another input to the system?**

(b) After adding the question, it would look like this.

Figure 4.65: Adjusting a sentence that omitted its question mark.

More often than not, a question mark can be missing in the automatically generated transcript. Figure 4.65 shows an example of where a question mark is missing. Although our algorithms described in section 5.1 can handle this, it is important for our taggers that it is clear what sentence is a question or not.

INTERVIEWER : and what should be the assignment trigger if you for example should the system automatically know that a patient got assigned to a doctor through a booking. So for example if you call and have a booking, **does that automatically mean that she got assigned to a specific doctor**

INTERVIEWEE : **the patient's file?** You should have doctors that are assigned to that patient. So if it's for example a new patient then um within the booking or scheduling they can either pick a doctor or assigned a doctor or depends on whether they call or whether they do it online themselves, whether they care or they say I I don't care what doctor I'm going to whoever is available and that way you already have the link.

(a) An example of a part of a transcript where in the first speakerturn, the interviewer asks a question, but the next speakerturn takes this question mark.

INTERVIEWER : Okay, and what should be the assignment trigger? If you for example should the system automatically know that a patient got assigned to a doctor through a booking. So for example if you call and have a booking, **does that automatically mean that she got assigned to a specific doctor?**

INTERVIEWEE : **Yeah so in the patient's file,** you should have doctors that are assigned to that patient. So if it's for example a new patient then um within the booking or scheduling they can either pick a doctor or assigned a doctor or depends on whether they call or whether they do it online themselves, whether they care or they say I I don't care what doctor I'm going to whoever is available and that way you already have the link.

(b) When we move the question mark to the interviewer's speakerturn, and fix the sentence, it looks like the following.

Figure 4.66: Adjusting a question mark in that was moved into the wrong speakerturn.

As seen in the previous examples, often a sentence is ended incorrectly and moved to the next speakerturn. This can also be the case with a questioning sentence, but the question gets moved into the next speakerturn. As an example, in Figure 4.66 this is the case. For our tagging this would make us focus on the wrong speakerturn when finding the speakerturn that asks the question.

INTERVIEWER : **it's sort of rescheduling from yesterday?**

(a) An example of a part of a sentence in the transcript that should not be a question.

INTERVIEWER : **Thanks for the rescheduling from yesterday.**

(b) Instead, the sentence should look like this.

Figure 4.67: Adjusting a a sentence that was not a question.

In some cases, such as the example in Figure 4.67, a sentence is not actually a question, but still contains a question mark. This is due to an error in the automated transcription, where the speaker is misheard or misunderstood. In our case, this would make a false positive.

INTERVIEWER : No

INTERVIEWEE : So right now you only get information when you come to an appointment from the doctor verbally um and you have to call or walk into schedule an appointment and that's basically it. Um Only on request you can request to be sent like your your file but then in a printed version other than that **they are not included so**

INTERVIEWER : **far. No.** Okay. Um where do you think they should be involved in the health care management?

(a) Here, the interviewer has a speakerturn that also contains part of a speakerturn by the interviewee. This can be seen because they are saying 'No' to nothing in particular.

INTERVIEWEE : No. So right now you only get information when you come to an appointment from the doctor verbally um and you have to call or walk into schedule an appointment and that's basically it. Um Only on request you can request to be sent like your your file but then in a printed version other than that **they are not included so far.**

INTERVIEWER : *So no digital versions being sent?*

INTERVIEWEE : **No.**

INTERVIEWER : Okay. Um where do you think they should be involved in the health care management?

(b) After manually adjusting the omitted change of speakerturn and transcribing the part that was not included in the automated transcription, the final part of the transcript looks like this.

Figure 4.68: Adjusting an omitted change of speakerturn

Finally, in the example in Figure 4.68, we see that sometimes information can be omitted. Here a change of speakerturn was missed and allowed us to miss out on a question that was asked, in this case asking if there are no digital versions being sent. We would miss out on this information which can be crucial to the requirements engineer.

E Tagging guide

Designing an approach for highlighting requirements from elicitation interviews - Tagging Guide

Xavier de Bondt (6221033)

June 20, 2022

Problem statement

The design of software systems generally start out with a conversation to understand the current situation and achieve a shared understanding. An important aspect of such a conversation between a practitioner and customers or stakeholders is gathering requirements. This so-called requirements elicitation (Zowghi and Coulin, 2005) , is a complex process that can introduce a certain bias (Ferrari et al., 2016) from the practitioner, because of their background knowledge or an ambiguity in the conversation. Next to that, requirements elicitation can introduce risk of missing a requirement or misinterpreting a requirement.

To aid the identification of these requirements, it is possible to conceive a tool that is able to highlight requirements-relevant transcript segments, to improve the quality of the resulting process. Moreover, this could be used to validate the resulting output, based on the conversation.

As we started designing approaches to perform this task, we noted the role questions take in these interviews. To test our approaches we need a golden standard for what questions could bring requirements-relevant information, which is information that is relevant for requirements engineers.

To help us reach this golden standard, you will be asked to tag the speakerturns[1] in our conversations. This entails that you will answer two questions, namely "What type of requirements-relevant information can be found here?" and "Where is this requirements-relevant information located?".

You will be shown a a part of the transcript, which is a sequence of three speakerturns, chronological in order:

- **The previous speakerturn**
  This speakerturn serves as context. It is there for your understanding of the conversation, but this should not be tagged or analyzed.
- **The current speakerturn**
  This is the speakerturn that we focus on and contains a **question.** This question will be marked in bold.
- **The next speakerturn**
  The following speakerturn, that possibly includes an answer to the question asked.

Here, we indicate which speaker says what. There are two roles here, interviewer and interviewee. The interviewer(s) ask questions about the system while the interviewee tries to answer these.

Categorization of the requirements-relevant information

In order to determine if there is any requirements-relevant information in this part of the transcript, we ask you to categorize the requirements-relevant information. This is done by selecting one or more from the following categories:

- **Functional requirement**
  Functionality. The speakerturn refers to functionality that the software system has to exhibit. For example, register users, schedule events, calculate something or allow messaging.
- **Non functional requirement**
  Software quality or non-functional requirement. The speakerturn refers to qualities that the system should provide while delivering its functionalities, e.g. speed, security, capacity, compatibility, reliability, usability, portability.

[1]A speakerturn is the part in which a certain speaker is speaking, thus containing only the parts that are spoken by one single speaker.

- **System Users**
  This talks about the users of the system, also include other stakeholders here that do not use the system.
- **Current Process understanding**
  This is information about the current process or system as-is, this can be about the current problems they are facing.
- **Within or outside of the scope**
  Any direct discussion of elements that should be in the system to-be or not. This discussed boundaries to the scope of the system.
- **There is no requirements-relevant information**
  Some questions asked in the transcript and answers to that question do not contain any requirements-relevant information. In other words, none of the other categories apply to this part of the transcript.

Locating requirements-relevant information

The final question that will be asked for each piece of transcript is "Where is this requirements-relevant information located?", if the previous question was not answered with 'There is no requirements-relevant information. Here, we give you the two options, to be answered yes or no (checked or unchecked):

- *Do you expect the question in the* ***current speakerturn*** *to be answered with requirements-relevant information?*
  For example, the interviewer asks a question that that is most likely answered with requirements-relevant information, since it is prompting information about the current situation.

  CURRENT SPEAKERTURN (INTERVIEWER) : Okay. We're not during, during the season. You should not change any policies. Yeah. Okay. That's good for the scheduling. And then the third value is better support for the fans. **And that was um, can you, first of all in the as-is situation, can you explain some of the how are the fans able to, what kind of support is there for the fans at the moment?**

- *Does the* ***next speakerturn*** *(after the question) contain requirements-relevant information?*
  In this example, the interviewer asks a question that will be answered with requirements-relevant information, after which the interviewee gives that requirements-relevant information in their answer.

  CURRENT SPEAKERTURN (INTERVIEWER) : Okay. **And then if we go to the referees, what kind of things should the referees be able to manage in the system?**

  NEXT SPEAKERTURN (INTERVIEWEE) : The referees mainly can look at the game they are scheduled to. Uh huh. Uh Can report on the events during the game.

  The next speakerturn should be marked to contain requirements-relevant information, since it explicitly describes what the referees can do. Next to that, the current speakerturn also contains a question that will be answered with requirements-relevant information, so actually both options should be taken here.

Tagging instructions

In this activity, you will be asked what type of requirements-relevant information can be found in piece of transcript that is shown. Here we ask you to disregard the previous speakerturn, only focusing on the current and the next speakerturn. If the option 'There is no requirements-relevant information' is not selected, there will be a followup question. This asks 'Where is this requirements-relevant information located?'.

References

Ferrari, A., Spoletini, P., and Gnesi, S. (2016). Ambiguity and tacit knowledge in requirements elicitation interviews. *Requirements Engineering*, 21.

Zowghi, D. and Coulin, C. (2005). Requirements elicitation: A survey of techniques, approaches, and tools. In *Engineering and managing software requirements*, pages 19–46. Springer.

References

[1] Axel Van Lamsweerde. *Requirements engineering: From system goals to UML models to software.* Vol. 10. Chichester, UK: John Wiley & Sons, 2009.

[2] A Smith, D Bieg, and T Cabrey. "PMI's Pulse of the Profession® In-Depth Report: Requirements Management–A Core Competency for Project and Program Success". In: *Project Management Institute, Newtown Square, PA* (2014).

[3] Didar Zowghi and Chad Coulin. "Requirements elicitation: A survey of techniques, approaches, and tools". In: *Engineering and managing software requirements.* Springer, 2005, pp. 19–46.

[4] Dante Carrizo, Oscar Dieste, and Natalia Juristo. "Systematizing requirements elicitation technique selection". In: *Information and Software Technology* 56.6 (2014), pp. 644–669.

[5] Muneera Bano et al. "Learning from mistakes: An empirical study of elicitation interviews performed by novices". In: *2018 IEEE 26th International Requirements Engineering Conference (RE).* IEEE. 2018, pp. 182–193.

[6] Alistair Sutcliffe and Pete Sawyer. "Requirements elicitation: Towards the unknown unknowns". In: *2013 21st IEEE International Requirements Engineering Conference (RE).* IEEE. 2013, pp. 92–104.

[7] Julia Hirschberg and Christopher D Manning. "Advances in natural language processing". In: *Science* 349.6245 (2015), pp. 261–266.

[8] Liping Zhao et al. "Natural language processing (NLP) for requirements engineering: A systematic mapping study". In: *arXiv preprint arXiv:2004.01099* (2020).

[9] Piet den Blanken. *ICT'ers Werken Vaakst Vanuit Huis Tijdens Coronacrisis.* Aug. 2020. URL: `https://www.cbs.nl/nl-nl/nieuws/2020/33/ict-ers-werken-vaakst-vanuit-huis-tijdens-coronacrisis`.

[10] OSIRIS Student Mobile. "Requirements engineering (INFOMRE)". In: (2021). URL: `https://osiris-student.uu.nl/#/onderwijscatalogus/extern/cursus?cursuscode=INFOMRE&taal=nl&collegejaar=2021`.

[11] Roel J Wieringa. *Design science methodology for information systems and software engineering.* Springer, 2014.

[12] David E Avison et al. "Action research". In: *Communications of the ACM* 42.1 (1999), pp. 94–97.

[13] Scott WH Young. "Improving library user experience with A/B testing: Principles and process". In: *Weave: Journal of Library User Experience* 1.1 (2014).

[14] Sebastian Ruder. *Transfer Learning - Machine Learning's Next Frontier.* `http://ruder.io/transfer-learning/`. 2017.

[15] Martin Glinz et al. "Handbook for the CPRE Foundation Level according to the IREB Standard". In: (2020).

[16] Orlena Gotel and Anthony Finkelstein. "Modelling the contribution structure underlying requirements". In: Auflage Aachen: Verlag der Augustinus Buchhandlung. 1994.

[17] O Gotel and ACW Finkelstein. *An Analysis of the Requirements Engineering Traceability Problem.* Tech. rep. Tech. Rep., Imperial College, Department of Computing, TR-93-41, 1993.

[18] Charles L Briggs. *Learning how to ask: A sociolinguistic appraisal of the role of the interview in social science research.* 1. Cambridge university press, 1986.

[19] Vincenzo Gervasi et al. "Unpacking tacit knowledge for requirements engineering". In: *Managing requirements knowledge.* Springer, 2013, pp. 23–47.

[20] Alessio Ferrari et al. *SaPeer and ReverseSaPeer Approaches for Training Students in Requirements Elicitation Interviews— Educational Material.* Version 2.0. Apr. 2020. DOI: `10.5281/zenodo.3765214`. URL: `https://doi.org/10.5281/zenodo.3765214`.

[21] Mike Cohn. *User stories applied: For agile software development.* Addison-Wesley Professional, 2004.

[22] Bill Wake. "INVEST in good stories, and SMART tasks". In: (2003).

[23] Garm Lucassen et al. “Improving agile requirements: the quality user story framework and tool”. In: *Requirements engineering* 21.3 (2016), pp. 383–403.

[24] Matt Wynne, Aslak Hellesoy, and Steve Tooke. *The cucumber book: behaviour-driven development for testers and developers*. Pragmatic Bookshelf, 2017.

[25] *What is ”given - when - then”?* Mar. 2021. URL: `https://www.agilealliance.org/glossary/gwt/`.

[26] L Mich, M Franch, and P Novi Inverardi. “Requirements analysis using linguistic tools: Results of an on-line survey”. In: *Requirements Engineering Journal* 2 (2003).

[27] Daniel M Berry et al. “From contract drafting to software specification: Linguistic sources of ambiguity-a handbook version 1.0”. In: (2000).

[28] John Lyons and Lyons John. *Linguistic semantics: An introduction*. Cambridge University Press, 1995.

[29] Graeme Hirst. *Semantic interpretation and the resolution of ambiguity*. Cambridge University Press, 1992.

[30] James Allen. *Natural language understanding*. Benjamin-Cummings Publishing Co., Inc., 1988.

[31] Douglas Walton. *Fallacies arising from ambiguity*. Vol. 1. Springer Science & Business Media, 2013.

[32] Gayane Hakobyan. “Elliptical Structures in Newspaper Discourse”. In: (June 2016). DOI: `10.13140/RG.2.1.3854.0403`.

[33] Catherine Anderson. *10.4 deixis: Meaning that depends on context*. Mar. 2018. URL: `https://ecampusontario.pressbooks.pub/essentialsoflinguistics/chapter/10-4-deixis-meaning-that-depends-on-context/`.

[34] Colette Rolland and Christophe Proix. “A natural language approach for requirements engineering”. In: *International Conference on Advanced Information Systems Engineering*. Springer. 1992, pp. 257–277.

[35] Kevin Ryan. “The role of natural language in requirements engineering”. In: *[1993] Proceedings of the IEEE International Symposium on Requirements Engineering*. IEEE. 1993, pp. 240–242.

[36] Russell J Abbott and DK Moorhead. “Software requirements and specifications: A survey of needs and languages”. In: *Journal of Systems and Software* 2.4 (1981), pp. 297–316.

[37] Fabiano Dalpiaz et al. “Natural language processing for requirements engineering: The best is yet to come”. In: *IEEE software* 35.5 (2018), pp. 115–119.

[38] Alessio Ferrari et al. “Natural Language Requirements Processing: A 4D Vision.” In: *IEEE Softw.* 34.6 (2017), pp. 28–35.

[39] Saurabh Tiwaria et al. “A Report on the First Workshop on Natural Language Processing Advancements for Software Engineering (NLPaSE) co-located with APSEC 2020”. In: (2020).

[40] “Message from the NLP-SEA 2020 Chairs”. In: *2020 35th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*. Los Alamitos, CA, USA: IEEE Computer Society, Sept. 2020, pp. 14–14. DOI: `10.1109/ASEW50548.2020.00005`. URL: `https://doi.ieeecomputersociety.org/10.1109/ASEW50548.2020.00005`.

[41] Agustin Casamayor, Daniela Godoy, and Marcelo Campo. “Identification of non-functional requirements in textual specifications: A semi-supervised learning approach”. In: *Information and Software Technology* 52.4 (2010), pp. 436–445.

[42] Alessio Ferrari et al. “Detecting requirements defects with NLP patterns: an industrial experience in the railway domain”. In: *Empirical Software Engineering* 23.6 (2018), pp. 3684–3733.

[43] Henning Femmer et al. “Rapid quality assurance with requirements smells”. In: *Journal of Systems and Software* 123 (2017), pp. 190–213.

[44] Davide Falessi, Giovanni Cantone, and Gerardo Canfora. “Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques”. In: *IEEE Transactions on Software Engineering* 39.1 (2011), pp. 18–44.

[45] Chetan Arora et al. “Automated extraction and clustering of requirements glossary terms”. In: *IEEE Transactions on Software Engineering* 43.10 (2016), pp. 918–945.

[46] Jin Guo, Jinghui Cheng, and Jane Cleland-Huang. "Semantically enhanced software traceability using deep learning techniques". In: *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE. 2017, pp. 3–14.

[47] Marcel Robeer et al. "Automated extraction of conceptual models from user stories via NLP". In: *2016 IEEE 24th international requirements engineering conference (RE)*. IEEE. 2016, pp. 196–205.

[48] Travis Breaux and Annie Antón. "Analyzing regulatory rules for privacy and security requirements". In: *IEEE transactions on software engineering* 34.1 (2008), pp. 5–20.

[49] Sallam Abualhaija et al. "Automated demarcation of requirements in textual specifications: a machine learning-based approach". In: *Empirical Software Engineering* 25.6 (2020), pp. 5454–5497.

[50] Tjerk Spijkman et al. "Concept Extraction in Requirements Elicitation Session Recordings: Prototype and Experimentation". In: *Joint Proceedings of REFSQ 2021 Workshops, OpenRE, Poster and Tools Track, and Doctoral Symposium co-located with the 27th International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2021), Essen, Germany, April 12, 2021*. Ed. by Fatma Basak Aydemir et al. Vol. 2857. CEUR Workshop Proceedings. CEUR-WS.org, 2021. URL: `http://ceur-ws.org/Vol-2857/nlp4re5.pdf`.

[51] Elizabeth D Liddy. "Natural language processing". In: (2001).

[52] Gobinda G Chowdhury. "Natural language processing". In: *Annual review of information science and technology* 37.1 (2003), pp. 51–89.

[53] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. "Natural language processing: an introduction". In: *Journal of the American Medical Informatics Association* 18.5 (2011), pp. 544–551.

[54] Dan Jurafsky and James H Martin. "Speech and language processing. Vol. 3". In: *US: Prentice Hall* (2014).

[55] Joakim Nivre et al. "Universal Dependencies v1: A Multilingual Treebank Collection". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 1659–1666. URL: `https://aclanthology.org/L16-1262`.

[56] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. "Building a Large Annotated Corpus of English: The Penn Treebank". In: *Computational Linguistics* 19.2 (1993), pp. 313–330. URL: `https://aclanthology.org/J93-2004`.

[57] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[58] Tom Mitchell. "Machine learning". In: (1997).

[59] Hal Daumé. *A course in machine learning*. Hal Daumé III, 2017.

[60] Gareth James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[61] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[62] Xiaojin Zhu and Andrew B Goldberg. "Introduction to semi-supervised learning". In: *Synthesis lectures on artificial intelligence and machine learning* 3.1 (2009), pp. 1–130.

[63] Xiao Liu et al. "Self-supervised learning: Generative or contrastive". In: *IEEE Transactions on Knowledge and Data Engineering* (2021).

[64] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[65] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern recognition letters* 27.8 (2006), pp. 861–874.

[66] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. "A survey of transfer learning". In: *Journal of Big data* 3.1 (2016), pp. 1–40.

[67] Jeremy Howard and Sebastian Ruder. "Universal language model fine-tuning for text classification". In: *arXiv preprint arXiv:1801.06146* (2018).

[68] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[69] Daniel S Kermany et al. "Identifying medical diagnoses and treatable diseases by image-based deep learning". In: *Cell* 172.5 (2018), pp. 1122–1131.

[70] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition.* Ieee. 2009, pp. 248–255.

[71] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[72] Bernardino Romera-Paredes and Philip Torr. "An embarrassingly simple approach to zero-shot learning". In: *International conference on machine learning.* PMLR. 2015, pp. 2152–2161.

[73] Pushpankar Kumar Pushp and Muktabh Mayank Srivastava. "Train once, test anywhere: Zero-shot learning for text classification". In: *arXiv preprint arXiv:1712.05972* (2017).

[74] Wenpeng Yin, Jamaal Hay, and Dan Roth. "Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach". In: *arXiv preprint arXiv:1909.00161* (2019).

[75] Omer Sagi and Lior Rokach. "Ensemble learning: A survey". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.4 (2018), e1249.

[76] Yann LeCun, Yoshua Bengio, et al. "The handbook of brain theory and neural networks". In: (1998).

[77] Robi Polikar. "Ensemble based systems in decision making". In: *IEEE Circuits and systems magazine* 6.3 (2006), pp. 21–45.

[78] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[79] Tjerk Spijkman, Fabiano Dalpiaz, and Sjaak Brinkkemper. "Requirements Elicitation via Fit-Gap Analysis: A View Through the Grounded Theory Lens". In: *Advanced Information Systems Engineering - 33rd International Conference, CAiSE 2021, Melbourne, VIC, Australia, June 28 - July 2, 2021, Proceedings.* Ed. by Marcello La Rosa, Shazia W. Sadiq, and Ernest Teniente. Vol. 12751. Lecture Notes in Computer Science. Springer, 2021, pp. 363–380. DOI: `10.1007/978-3-030-79382-1\_22`. URL: `https://doi.org/10.1007/978-3-030-79382-1%5C_22`.

[80] George Blick, Thomas Gulledge, and Rainer Sommer. "Defining business process requirements for large scale public sector ERP implementations: A case study". In: *ECIS 2000 Proceedings* (2000), p. 157.

[81] Janis Grabis. "Optimization of Gaps Resolution Strategy in Implementation of ERP Systems." In: *ICEIS (1).* 2019, pp. 84–92.

[82] Thomas R Gulledge. "ERP gap-fit analysis from a business process orientation". In: *International Journal of Services and Standards* 2.4 (2006), pp. 339–348.

[83] Tsai Chi Kuo. "Mass customization and personalization software development: a case study eco-design product service system". In: *Journal of Intelligent Manufacturing* 24.5 (2013), pp. 1019–1031.

[84] Ben Light. "The maintenance implications of the customization of ERP software". In: *Journal of software maintenance and evolution: research and practice* 13.6 (2001), pp. 415–429.

[85] Sven Apel et al. *Feature-oriented software product lines.* Springer, 2016.

[86] Daniel Berry et al. "The case for dumb requirements engineering tools". In: *International Working Conference on Requirements Engineering: Foundation for Software Quality.* Springer. 2012, pp. 211–217.

[87] Zijad Kurtanović and Walid Maalej. "Automatically classifying functional and non-functional requirements using supervised machine learning". In: *2017 IEEE 25th International Requirements Engineering Conference (RE).* Ieee. 2017, pp. 490–495.

[88] Tobias Hey et al. "NoRBERT: Transfer learning for requirements classification". In: *2020 IEEE 28th International Requirements Engineering Conference (RE).* IEEE. 2020, pp. 169–179.

[89] Fabiano Dalpiaz et al. "Requirements classification with interpretable machine learning and dependency parsing". In: *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE. 2019, pp. 142–152.

[90] Ishrar Hussain, Leila Kosseim, and Olga Ormandjieva. "Using linguistic knowledge to classify non-functional requirements in SRS documents". In: *International Conference on Application of Natural Language to Information Systems*. Springer. 2008, pp. 287–298.

[91] Edna Dias Canedo and Bruno Cordeiro Mendes. "Software Requirements Classification Using Machine Learning Algorithms". In: *Entropy* 22.9 (2020), p. 1057.

[92] Zahra Shakeri Hossein Abad et al. "What works better? A study of classifying requirements". In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE. 2017, pp. 496–501.

[93] Han van der Aa et al. "Extracting Declarative Process Models from Natural Language". In: *CAiSE*. Ed. by Paolo Giorgini and Barbara Weber. 2019, pp. 365–382. DOI: `10.1007/978-3-030-21290-2_23`.

[94] Efstathios Stamatatos. "A survey of modern authorship attribution methods". In: *Journal of the American Society for information Science and Technology* 60.3 (2009), pp. 538–556.

[95] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[96] Stephen A White. "Introduction to BPMN". In: *Ibm Cooperation* 2.0 (2004).

[97] Claudio Di Ciccio, Andrea Marrella, and Alessandro Russo. "Knowledge-intensive processes: characteristics, requirements and analysis of contemporary approaches". In: *Journal on Data Semantics* 4.1 (2015), pp. 29–57.

[98] Wil MP van Der Aalst, Maja Pesic, and Helen Schonenberg. "Declarative workflows: Balancing between flexibility and support". In: *Computer Science-Research and Development* 23.2 (2009), pp. 99–113.

[99] Matthew B Dwyer, George S Avrunin, and James C Corbett. "Patterns in property specifications for finite-state verification". In: *Proceedings of the 21st international conference on Software engineering*. 1999, pp. 411–420.

[100] Claudio Di Ciccio et al. "Resolving inconsistencies and redundancies in declarative process models". In: *Information Systems* 64 (2017), pp. 425–446.

[101] Fabian Friedrich, Jan Mendling, and Frank Puhlmann. "Process model generation from natural language text". In: *International Conference on Advanced Information Systems Engineering*. Springer. 2011, pp. 482–496.

[102] Marie-Catherine De Marneffe and Christopher D Manning. "The Stanford typed dependencies representation". In: *Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation*. 2008, pp. 1–8.

[103] Owen Doody and Maria Noonan. "Preparing and conducting interviews to collect data". In: *Nurse researcher* 20.5 (2013).

[104] Wendy Hollway and Tony Jefferson. "Eliciting narrative through the in-depth interview". In: *Qualitative inquiry* 3.1 (1997), pp. 53–70.

[105] Ann Bies et al. "Bracketing guidelines for Treebank II style Penn Treebank project". In: *University of Pennsylvania* 97 (1995), p. 100.

[106] Edward Holliman John J. Godfrey. *Switchboard-1 Release 2*. 1993. DOI: `10.35111/sw3h-rw02`. URL: `https://doi.org/10.35111/sw3h-rw02`.

[107] Mikhail Galkin and Valentin Malykh. *Wikipedia TF-IDF Dataset Release*. Version v1.0. Jan. 2020. DOI: `10.5281/zenodo.3631674`. URL: `https://doi.org/10.5281/zenodo.3631674`.

[108] Kai Ming Ting. "Confusion matrix." In: *Encyclopedia of machine learning and data mining* 260 (2017).

[109] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[110] Adina Williams, Nikita Nangia, and Samuel Bowman. “A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers).* New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 1112–1122. URL: `http://aclweb.org/anthology/N18-1101`.

[111] Claes Wohlin et al. *Experimentation in software engineering.* Springer Science & Business Media, 2012.