Subproject B1: Dialogue-Based Requirement Compensation and Style-Adjusted Data-To-Text Generation

Frederik S. Bäumer[1], Wei-Fan Chen[2], Michaela Geierhos[3], Joschka Kersting[2], Henning Wachsmuth[4]

1 Applied AI Group, Bielefeld University of Applied Sciences, Bielefeld, Germany
2 Department of Computer Science, Paderborn University, Paderborn, Germany
3 Research Institute CODE, University of the Bundeswehr Munich, Neubiberg, Germany
4 Institute of Artificial Intelligence, Leibniz University Hannover, Hannover, Germany

1 Introduction

OTF computing is exploring ways to provide people with more customized, on-demand software services that meet their needs. Similar to using a search engine, users should be able to express their needs in natural language without any special technical background. For this reason, processing and interpreting natural language requirements is an essential part of the OTF vision. Since formal specifications are not very intuitive, natural language tends to be the only format for service descriptions that most users will consider.

Subproject B1 deals with different types of service requirement specifications that enable a successful search, composition, and analysis of services. In the sense of agile, collaborative software development, the idea is to involve users in an interactive composition process of software services to be created on-the-fly. This setting implies a dialogical situation between users and systems, suggesting the use of a domain-specific chatbot for a specific query on the one hand, and the resolution of ambiguities on the other. For such a composition process to be successful, it should be transparent to users. In particular, it must be clear which initial requirements were taken into account in the creation and which had to be dropped.

Service descriptions can be considered a less formal form of requirement specifications because they describe a service in natural language. Moreover, the accuracy of a description depends on a number of factors, such as the proficiency of the requirement's author. For this reason, service descriptions are sometimes referred to as user-generated informal documents [MLC14]. Formal and semi-formal description languages are one way to avoid the shortcomings of natural language in OTF computing. The software specification language

frederik.baeumer@fh-bielefeld.de (Frederik S. Bäumer), cwf@mail.upb.de (Wei-Fan Chen), michaela.geierhos@unibw.de (Michaela Geierhos), jkers@mail.upb.de (Joschka Kersting), h.wachsmuth@ai.uni-hannover.de (Henning Wachsmuth)

is an example of providing comprehensive service specifications and is applicable to both non-functional and functional software requirements [PJR+16]. But even in a simplified form, users cannot apply it as they lack the necessary technical expertise [FdSG14; GSB15]. Therefore, developers must accept free-form, natural language requirement descriptions from users. In doing so, they have to deal with difficulties that are typical for free-form text. These include, for example, a lack of structure and correctness, grammatical and spelling errors, and ambiguity in syntax and semantics. In addition, there will be missing information that is essential for development, but that a user does not have in mind. Thus, callbacks are unavoidable.

Initially, this subproject focused on the development of a parameterized core language for services [Pla13; BBP15] to process requirements automatically, accurately, and efficiently. Due to a lack of acceptance by the target group, user-friendly requirement specifications were proposed as an alternative [Bäu17; BG18]. These specifications are intuitively understandable and mainly used by customers in the OTF market (i.e., end users or domain experts). To improve transparency and human-machine collaboration, dialogue-based requirements compensation was later introduced [KAG22], which provides explanations of services in natural language [CASW21]. Comprehensible explanations are needed because the deficiencies in requirement specifications cannot always be automatically compensated and because users do not know which of their requirements have been fulfilled or not, why this is the case, and what other services need to be included. Figure 10 illustrates the outlined evolution of the process over three funding periods.

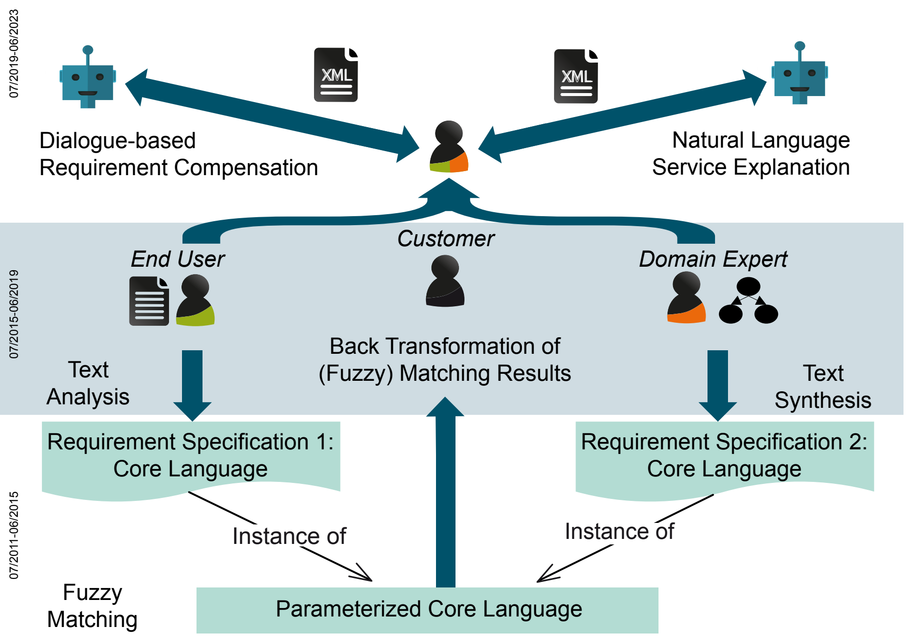

07/2019-06/2023
XML
XML
Dialogue-based Requirement Compensation
Natural Language Service Explanation
07/2015-06/2019
Customer
End User
Domain Expert
Text Analysis
Back Transformation of (Fuzzy) Matching Results
Text Synthesis
07/2011-06/2015
Requirement Specification 1: Core Language
Requirement Specification 2: Core Language
Instance of
Instance of
Fuzzy Matching
Parameterized Core Language

Figure 10: *Overview of the development phases of subproject B1 (bottom to top).*

In the following, we present and discuss four highlights of Subproject B1, the first two of which are related to the *automated optimization* of specifications, the others to the *generation of explanations* of services.

1. **Compensation of linguistic deficiencies:** The automatic selection of techniques that fit the user's description (cf. Sec. 2.1).

2. **Chatbot-enhanced deficiencies resolution:** The interactive resolution of remaining deficiencies in a dialogical manner (cf. Sec. 2.1).
3. **Data-to-text explanation generation:** The computational generation of a text explaining the main features of a created service (cf. Sec. 2.2).
4. **Style adjustment of explanations:** The computational transfer of the text's style to the language of a specific user group (cf. Sec. 2.2).

2 Highlights and Lessons Learned

The research highlights presented in the following contribute to the broad field of natural language processing (NLP). They include novel fundamental computational methods as well as new applications of NLP in real-world technologies for the given scenarios.

2.1 Automated Optimization of Natural Language Specifications

There have been few attempts to address the variety and deficiencies of natural language software specifications as they arise in the software specification process by users without excessive back-checking. On the one hand, we have shown that it is possible to involve the non-specialist requirement creators in the software specification process without limiting their ability to express themselves and to support the software developers by automatically clarifying software specifications. The latter can avoid minor callbacks and thus reach the implementation faster. On the other hand, we addressed the detection of inaccuracies and incompleteness in requirements descriptions that still leave too much room for interpretation for a concrete software implementation. For this purpose, we developed a procedure that compensates ambiguous and partially incomplete statements of the requirement creator by using intelligent (request-driven) knowledge queries.

Compensation of Linguistic Deficiencies in Service Descriptions

Our goal was to develop a parameterized model that automatically chooses the right strategy to compensate for human shortcomings in natural language specification [Bäu17].

Since software descriptions are sensitive to linguistic deficiencies such as ambiguities and elisions, which can delay and thus hinder the specification process, a workaround had to be found. However, there are numerous software programs that can detect and partially correct deficiencies in requirement descriptions. Unfortunately, however, they are often difficult to use and not suitable for end users. In addition, they usually do not cover the full range of flaws and inaccuracies that can occur in natural language [BG18]. Examples include methods that can detect and correct multiple deficiencies in natural language requirements and tools that can be classified as expert solutions. These focus on a single phenomenon of linguistic imprecision, such as lexical ambiguity. However, there are also solutions that can detect ambiguity and incompleteness together or that can find different forms of ambiguity [TB13; Kör14]. Some studies [HB15; SJ15; Bäu17] show that many software solutions focus only on finding deficiencies. Therefore, users are still in charge of compensation. Furthermore, these methods have not been combined to compensate for service descriptions. End users expect their software requirements to be fully implemented, but are unable to identify and correct linguistic deficiencies in requirements themselves. In addition, there is no guarantee that end users will notice ambiguities or incompleteness, although this can be very frustrating. These problems can be solved by implementing an interactive, computer-aided compensation process [BG18].

With our approach, users do not have to select the necessary techniques to compensate for the deficiencies in their service descriptions; this is done automatically. This minimizes the number of callbacks and thus streamlines the process. As a result, outputs are improved and users are freed from persistent, manual tasks. Finally, our approach improves the state of the art in OTF computing [BG18]. Our research methodology, while following the principles of design science, results in a software tool called CORDULA [Bäu17]. Figure 11 shows the modules used in our text analysis pipeline and their interaction.

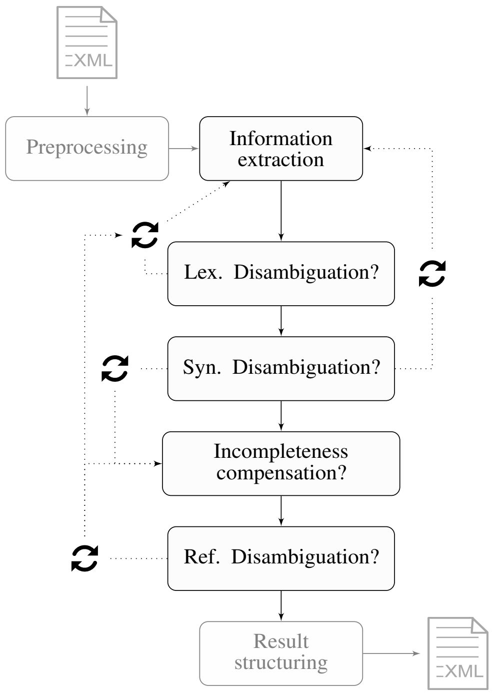
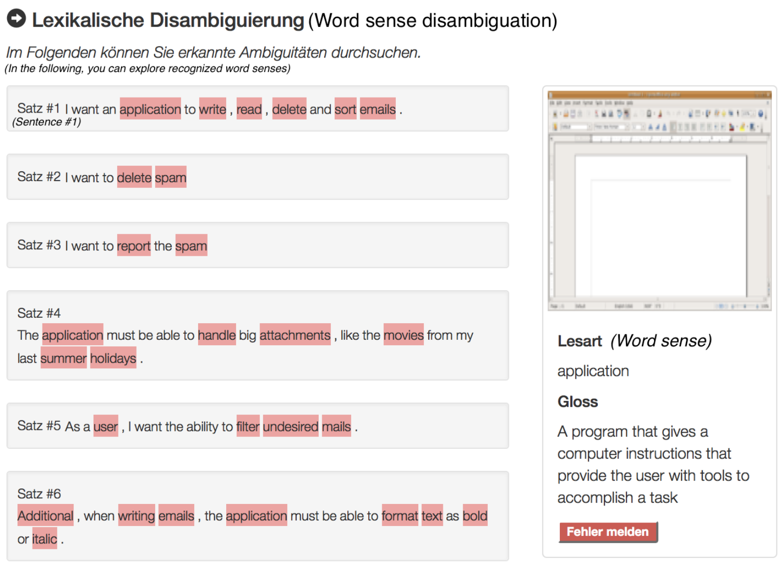


Figure 11: *Indicator-based text analysis pipeline for deficiency compensation [BG18].*

As can be seen in Figure 11, we process the user input step by step, starting at the top. After preprocessing, we extract information. We can deal with syntactic, lexical, and referential ambiguity, as well as incompleteness. The configuration and execution, as well as the monitoring of the processing pipeline, are then performed by an indicator-based compensation strategy [BG18]. The strategy allows to exploit the synergies between the techniques used. However, the most important question here is whether the input is a software description. This pre-step is called on-off-topic classification, and the tool developed for this purpose is called REaCT [DG16]. It finds process words, i.e., semantic information such as the role or the action in a service description. This is done to ensure that the input is indeed a software requirement description. Every other module, including referential disambiguation, is triggered by the corresponding requirement quality indicator [BG18]. Finally, the improved software description is then presented to the user.



Lexikalische Disambiguierung (Word sense disambiguation)

Im Folgenden können Sie erkannte Ambiguitäten durchsuchen.
(In the following, you can explore recognized word senses)

Satz #1 I want an application to write , read , delete and sort emails .
(Sentence #1)

Satz #2 I want to delete spam

Satz #3 I want to report the spam

Satz #4
The application must be able to handle big attachments , like the movies from my last summer holidays .

Satz #5 As a user , I want the ability to filter undesired mails .

Satz #6
Additional , when writing emails , the application must be able to format text as bold or italic .

Lesart *(Word sense)*
application
Gloss
A program that gives a computer instructions that provide the user with tools to accomplish a task

Figure 12: *Lexical disambiguation results as presented by CORDULA [Bäu17].*

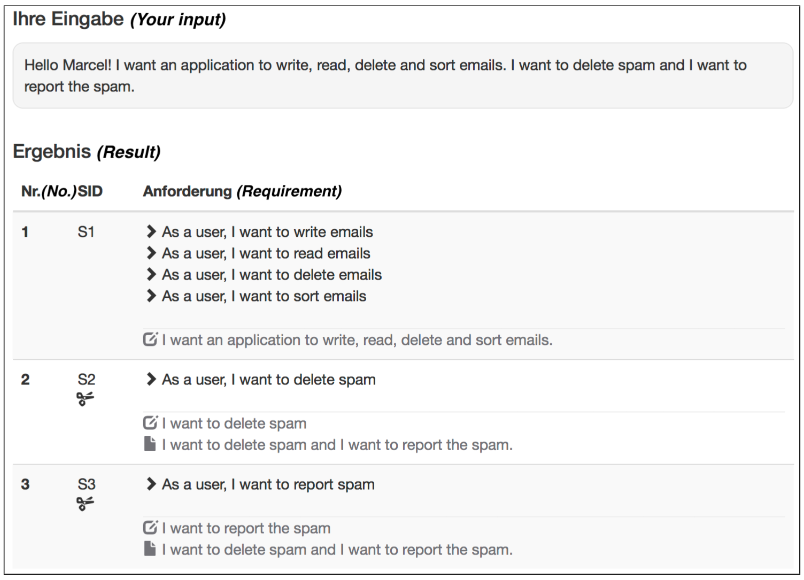
Looking more closely at the individual modules of the pipeline, lexical disambiguation, for example, aims to find the correct meaning of a word. After contextualizing a word, the application finds several possible readings. It is triggered when more than one reading is possible. The process involves the REaCT tool to semantically classify the tokens (i.e., to recognize actions, objects, etc.). In this way, readings can be eliminated, and finally, only the disambiguation candidates remain (see Figure 12).

(5)				$\boxed{\text{Action}}$	$\boxed{\text{Object}}$			$\boxed{\text{Refinement}}$
(4)				VB	NNS		~~PRP$~~	NN
(3)	~~I~~	~~want~~	~~to~~	send	emails	~~to~~	my	family
(2)	$\boxed{\text{Role}}$	$\boxed{\text{Priority}}$		$\boxed{\text{Action}}$	$\boxed{\text{Object}}$		Refinement	Refinement
(1)	I	want	to	send	emails	to	my	family

Figure 13: *Natural language processing with semantic role labeling [BG18].*

In addition, incompleteness compensation addresses service descriptions that lack information. REaCT's semantic role labeling (see Figure 13) is used to detect incomplete sentence constructs and to identify the missing information. Another ambiguity in language is referential ambiguity, i.e., which pronouns refer to which nouns. This is critical for the system to understand the query as a whole. We used part-of-speech tagging in combination with discourse analysis to solve this problem [BG18].

Based on this concept, we developed a system that uses automatic compensation strategies to assist end users in creating clear and comprehensive natural language requirements. For this purpose, linguistic indicators were developed to identify the need for each compensation method in the descriptions. Based on these indicators, the entire text analysis pipeline we have built is configured ad-hoc and then tailored to the unique details of a service description [BG18]. The technical implementation was done using CORDULA, a tool for the *Compensation of Requirements Descriptions Using Linguistic Analysis* [Bäu17], whose original frontend can be seen in Figure 14.


Ihre Eingabe *(Your input)*

Hello Marcel! I want an application to write, read, delete and sort emails. I want to delete spam and I want to report the spam.

Ergebnis *(Result)*

Nr.*(No.)*	SID	Anforderung *(Requirement)*
1	S1	› As a user, I want to write emails › As a user, I want to read emails › As a user, I want to delete emails › As a user, I want to sort emails I want an application to write, read, delete and sort emails.
2	S2	› As a user, I want to delete spam I want to delete spam I want to delete spam and I want to report the spam.
3	S3	› As a user, I want to report spam I want to report the spam I want to delete spam and I want to report the spam.

Figure 14: *Frontend for tracing the processing steps of the text analysis pipeline [Bäu17].*

CORDULA allows users to express their software requirements in natural language. It checks the descriptions for weaknesses such as incompleteness and ambiguity, but also for flaws, based on the pipeline shown in Figure 11. The tool compensates for these deficiencies in an understandable way via a web interface. Figure 14 shows that the output is in a simple language so that users can easily understand the information provided and check its consistency. CORDULA is based on two views: the end-user (and administrator) view and the system view. The frontend shown here is the (simplified) user view. For special information, e.g., disambiguation, additional interfaces are available to help debug the output. After compensation, the prototype transforms functional requirements into structured output for further use in the OTF computing scenario. Thus, others can build on our output, for example, by configuring software services ad-hoc and providing the software requested by the end user [FBG18].

In this highlight, we showed that inaccuracies in software descriptions can be automatically detected and compensated without user interaction. Our processing pipeline is optimized by linguistic indicators, minimizing runtime and user interaction. We also demonstrated that rule-based indicators accurately handle most linguistic deficiencies [BG18].

Chatbot-Enhanced Requirement Deficiencies Resolution

In order to eliminate requirement deficiencies that cannot be compensated by existing methods, dialogues are required that help to specify and complete derived requirement specifications. Existing extraction and compensation methods generate preliminary service templates from which software services can be selected and composed. The goal of dialogue control is to iteratively revise deficient templates in a guided manner in order to provide as much information as possible for service composition. To achieve this, we have divided the dialogue control into a requirement interpretation and a chat interpretation. While the latter is responsible for the dialogue control and the interpretation of all user input, the requirement interpretation is responsible for the classification, interpretation, and validation of detected requirements. This separation makes it possible to integrate existing chatbot techniques, while the extraction and compensation components developed in previous work can be used for processing software requirements. Especially for end users with little prior technical knowledge, it is necessary to perform the iterative clarification processes not only in dialogue but also with the help of explanations and examples. For example, when compensating for incompleteness, this may mean not only pointing out the missing information and asking for it to be filled in, but also providing a similar example that fits the context [BKG19].

In [FBG18], CORDULA has been further developed to take requirements deficiency resolution to a new level. The goal here is to maximize the automation of the process. It has been shown that domain-specific resources are needed to produce high-quality results. In addition, we have found that it is necessary to involve users in the process instead of presenting them with the results and having them go through the process again if they are not satisfied [BKG19]. Consequently, we developed a new approach for CORDULA that includes a chatbot and a knowledge base. Both support the given process by clarifying the possibilities and limitations of the software configuration process to the user, and the chatbot asks if the information is incomplete or ambiguous [Ahm22; KAG22].

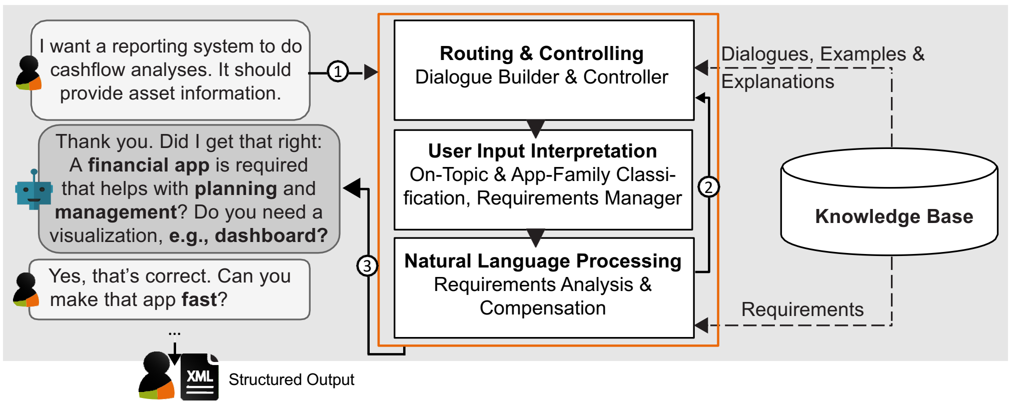

I want a reporting system to do cashflow analyses. It should provide asset information.
Thank you. Did I get that right: A **financial app** is required that helps with **planning** and **management**? Do you need a visualization, **e.g., dashboard?**
Yes, that's correct. Can you make that app **fast**?
...
XML
Structured Output
1
2
3
Routing & Controlling
Dialogue Builder & Controller
User Input Interpretation
On-Topic & App-Family Classification, Requirements Manager
Natural Language Processing
Requirements Analysis & Compensation
Dialogues, Examples & Explanations
Knowledge Base
Requirements

Figure 15: *Bidirectional information flow in the enhanced CORDULA version.*

Figure 15 shows our evolved solution, which, unlike its predecessor [Bäu17] (cf. Figure 11), is bidirectional. As you can see on the left side, we have a dialogue flow where the user interacts with an automated chatbot system. In the middle are the relevant elements of the processing pipeline. On the right is the knowledge base. The natural language processing pipeline shown in Figure 11 has been further developed and integrated into our solution, which includes on-off topic classification and incompleteness detection, but also new features.

Based on the use of domain knowledge, we also created a deep learning model that recognizes app families, which are stored in the knowledge base [Ahm22]. Figure 16 shows a snippet of the structure of this knowledge base. There is a hierarchical order describing app families, corresponding apps, services, functions, and templates. By using the knowledge base, the chatbot knows exactly what information it needs from the user to fill out a service template. In addition, the chatbot uses the app family to decide which path options are possible, which templates, and which service descriptions are possible.

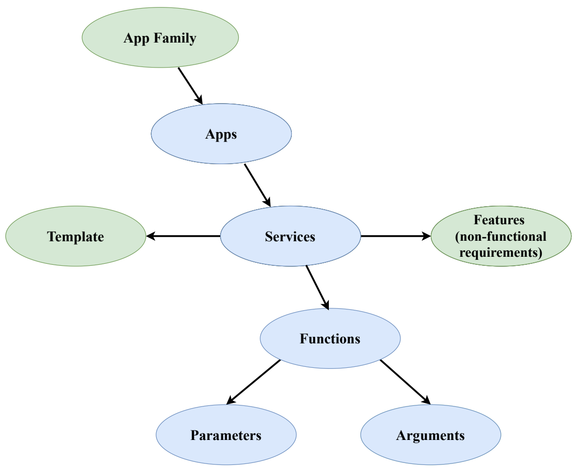


Figure 16: *Graph representation of a substructure of the knowledge base [Ahm22].*

For the dialogue management of the chatbot, we used Rasa[5] after testing different solutions such as DialoGPT [ZSG+20] or ChatterBot[6] (see Table 1).

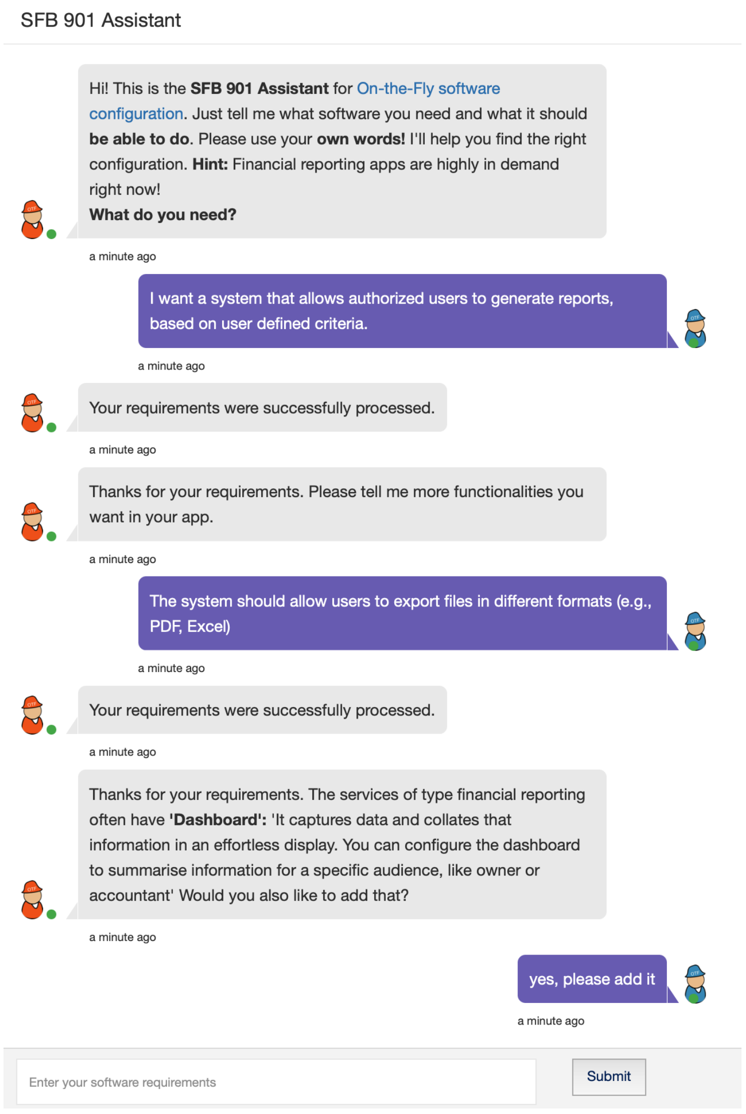
Models	Effectiveness	Efficiency	Satisfaction	User Approval
Rasa & knowledge base	0.84	0.78	0.73	0.73
ChatterBot	0.26	0.21	0.31	0.36
DialoGPT	0.47	0.79	0.52	0.63

Table 1: *Performance ratings for tested chatbot systems [Ahm22].*

Table 1 presents the effectiveness results in terms of the conversational intelligence and performance shown by the chatbot with respect to the user's goal. Efficiency refers to whether the chatbot seemed human-like. Satisfaction captures whether the user is satisfied with the chatbot's responses and behavior (e.g., politeness). User approval deals with the overall satisfaction of the user. DialoGPT cannot be directly combined with a knowledge base; the knowledge should be incorporated into the parameters of the chatbot. The chatbot achieves partially convincing results. However, ChatterBot is not convincing at all. We conclude that Rasa is the best solution based on an evaluation of user interactions. The intent classification works well, while the dialogue management allows us to add features from CORDULA and integrate the chatbot into our knowledge base. When users tested all three implementations, Rasa was by far the best in terms of overall performance, but also in terms of fine-grained evaluation results [Ahm22; KAG22].

Figure 17 shows an example of a user-chatbot interaction. Here, the resolution of the requirement description is done in a bidirectional manner. The chatbot greets the user and asks for the user's requirement descriptions. Each time the response is successfully processed by the system, the chatbot provides this as feedback ("Your requirements were successfully processed."). Some of the responses follow a rule-based approach to ensure that the requested information is always provided to each user. After the chatbot receives and processes the initial user input, it asks for more information. In the backend, the system queries the knowledge base and performs matching tasks using deep learning models to

[5] `https://rasa.com/`, last accessed 2023-02-06.
[6] `https://chatterbot.readthedocs.io/en/stable/`, last accessed 2023-02-06.


SFB 901 Assistant

Hi! This is the **SFB 901 Assistant** for On-the-Fly software configuration. Just tell me what software you need and what it should **be able to do**. Please use your **own words!** I'll help you find the right configuration. **Hint:** Financial reporting apps are highly in demand right now!
What do you need?

a minute ago

I want a system that allows authorized users to generate reports, based on user defined criteria.

a minute ago

Your requirements were successfully processed.

a minute ago

Thanks for your requirements. Please tell me more functionalities you want in your app.

a minute ago

The system should allow users to export files in different formats (e.g., PDF, Excel)

a minute ago

Your requirements were successfully processed.

a minute ago

Thanks for your requirements. The services of type financial reporting often have **'Dashboard':** 'It captures data and collates that information in an effortless display. You can configure the dashboard to summarise information for a specific audience, like owner or accountant' Would you also like to add that?

a minute ago

yes, please add it

a minute ago

Figure 17: *Example of a conversation with the enhanced CORDULA version [Ahm22].*

look for relevant service configurations or features that already exist in the OTF computing market. The user is free to decide what features to include or not include in the app. When there is enough information about the app that the chatbot has collected together with a user, the chatbot gives the user a summary of the requirements that have been met and those that have not. This way, the user knows what to expect from the configured app provided by the OTF system [Ahm22; KAG22].

In addition, the chatbot deals with parameters that are initially unknown to the user. This means that software configurations sometimes have requirement templates, for example, because some services cannot be combined with others, or because some apps require certain services. End users cannot know this, but they also do not have to: The system tries to fill in the templates in the background during the conversation and controls the chat accordingly, e.g. by asking appropriate questions and suggesting certain services. The aforementioned summary, which is communicated before the app is deployed, also includes services that have been added based on a template.

In summary, our research has solved the following three main problems: (1) Deficiencies in requirement descriptions cannot always be automatically fixed, which we address through our bidirectional chatbot solution with a knowledge base. (2) Configuring parameters that are initially unknown to users leads to errors in the service templates. We integrated this into the conversation, and we provided explanatory summaries. (3) Previously, users did not know which of their requirements were met (and which were not) in a created service, they had to start from scratch. This is also solved by our summary and explanation at the end of the chat. The user sees a list of all requirements, whether they were met, and where they came from (from the user, from a template, or from an app/service suggestion accepted by the user).

2.2 Natural Language Explanation Generation

One of our key goals was to make the service configuration process more comprehensible for end users. In this way, users should not have to find out by trial and error whether their service configuration expectations were met. Instead, they should get an early insight into the feasibility of their requirements. For this purpose, we investigated the extent to which typical user formulations based on the requirement descriptions are suitable for training a text generation approach and which techniques can be used for text generation in a dialogical question-answer setting.

To improve user understanding and service transparency, we provide explanations in natural language that describe the created service while being adjusted to the user's language. Human-like explanations are still understudied in natural language processing (NLP) research [WA22], particularly their generation. To obtain correct but understandable explanations, we combine the results of two complementary efforts: On the one hand, we developed a data-to-text generator for natural language explanations and evaluated its performance on a data set of explanations. On the other hand, we investigated how to adapt the linguistic style of the generated explanations to the user's proficiency. For the latter, we developed novel text style transfer methods and evaluated them on a corresponding data set. Both are described in detail below.

Data-To-Text Stylized Generation of Service Explanations

In the following, we summarize the contributions that we have made to data-to-text explanation generation with a touch on style adjustment [Bül21; ACGW21]. Specifically, we have attempted to generate natural language sentences from the output of the services. Here, the output is given as a set of tuples of the $< name, value >$. For example, $< F_1, 0.89 >$ declares the F_1-score of a computational model used in the service to be 0.89. In NLP, the underlying task is known as a data-to-text generation problem.

Since we did not have a sufficient number of service outputs with the target sentence data, we decided to study the given problem on data from another domain that shares similar properties with service explanations at an abstract level. In particular, we make use of transcriptions of politicians' speeches. These speeches naturally contain explanations of policies, actions, and the like. From OTF computing perspectives, such explanations provide the data we are interested in by using a distant-supervision manner.

A specific aspect of interest within the given task is the consideration of different linguistic styles. In our study, we collected speeches from various politicians and conducted experiments on them. For a controlled setting, we focused on the styles of two former U.S. presidents, Barack Obama and Donald Trump. Both have clearly recognizable speaking styles. Learning to generate texts that match the styles of the two presidents is challenging. Specifically, it involves at least the following types of style adjustment:

1. **Formality:** The complexity of the words used by the two presidents also differs significantly. In line with expectations, we observed that Obama uses more professional and educated words, while Trump prefers simple language. Analogous to our service explanation task, formality also serves as the generated explanations' formality.
2. **Political bias** and **subjectivity:** The two presidents are from various parties. As a result, they support or oppose different policies and have opposing views on several issues. For example, Obama is seen as being open to immigrants, unlike Trump. Our analysis also revealed that the subjectivity of the two presidents' language plays a role in distinguishing their styles. For example, Trump uses many emotional words in his speeches, while Obama avoids them. These two style features are specific to political speeches. In our service explanation task, it helps to specify word usage preferences in the explanation, such as preferring F-score instead of F-measure.

In our research, we investigated how different models learn and adapt these different aspects. For this purpose, we built a data set with a total of 962 transcribed public speeches, press conferences, and interviews of the two presidents, 434 of Obama and 528 of Trump. Since the two presidents' speeches have limited overlap in time, the data set is naturally non-parallel, meaning that we had to learn style adjustment without having training pairs from which to infer style correspondences directly.

Given the data set, we focused on *sentence-level* style adaptation as a first substantial step towards full document generation. First, we analyzed the training set to understand the style of each politician. In line with our previous work [CWAS18], we calculated the most discriminative words of each style to analyze each former president's word usage preference, and we determined the proportion of emotional words in the speeches. The results highlighted that Trump frequently used emotional words (e.g., disgrace or lied), while Obama did not (e.g., resolved or urgency). This informed us about what to look for when developing and evaluating style transfer approaches. For example, after a style transfer, the text should have a similar distribution of word usage as in the desired style. We also found that not all sentences in the speeches had significant style indicators. Therefore, we used only those sentences that contained at least one discriminative word (words used twice more often in one style) found by the analysis.

On this basis, we evaluated two approaches to the explanation generation task:

1. **NER + Data-to-text:** This is the approach we developed. It first extracts named entities in a sentence before applying one of two (one for Obama and one for Trump) fine-tuned neural language models (BART) to generate a sentence from the given named entities. The NER step is designed to capture the main content of the sentence, while the data-to-text part is designed to generate a sentence about the named entities using a specific speaker style.
2. **Cross-aligned autoencoder:** For comparison, we used a cross-aligned autoencoder for this task [SLBJ17] as a baseline. This model learns to optimize two neural

	Style Adaptation		**Explainability**	
	Automatic	Manual	Automatic	Manual
Cross-aligned autoencoder	68%	2.75	44%	2.24
Data-to-text generation (our approach)	86%	3.55	13%	1.62

Table 2: *Evaluation of data-to-text generation in terms of style adaptation and explainability: automatically computed percentage of success and manually assigned scores from 1 (worst) to 5 (best) for the cross-aligned autoencoder and for our approach.*

autoencoders to separate content and style in a latent space. An important feature of the model is that it can preserve the text structure and change the style of the text. The cross-aligned autoencoder shows how we can generate the explanations and adapt the style using a model.

The approach we developed for the explanation generation task first extracts named entities from the political speeches as the targets of the explanation. These named entities include the names of policies, organizations, and politicians. After extracting the named entities, the second step is to train a data-to-text model to generate texts from the extracted named entities, as shown in Figure 18.

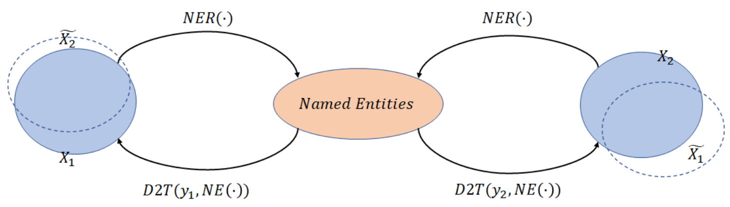


Figure 18: *The data-to-text explanation generator by [Bül21]. The approach is a combination of named entity recognition (NER) and a data-to-text generator (D2T).*

The idea of our approach is to express the content of a sentence in political speaker transfer only on the basis of the named entities. All other words are considered as style words by the speaker. The three key elements of our approach are as follows.

First, the *encoding* is the reduction of a sentence to its textual keywords. This is done using a widely used information extraction algorithm, namely the NER system of the popular Python library spaCy. Second, instead of a latent *representation*, the input is textual. It consists of a number of instances, each composed of the named entities of a proposition and their respective types. This has the advantage of being explicable, which is often not the case with neural network models. And third, *decoding* is the process of transforming the named entities into an explanation using natural language, relying on the outlined data-to-text transformer model. The model, trained on a corpus of either political speaker, should implicitly learn to generate an explanation that expresses the speaker's style and thus resembles the language usage.

We automatically and manually evaluated our approach and the baseline. Both evaluations were aimed at the effectiveness of style adaptation and explainability. The results are shown in Table 2. The automatic results are given as a percentage of success, while the manual results are scores ranging from 1 to 5, with 5 being the best. Table 2 suggests that both approaches have pros and cons, namely that the cross-aligned autoencoder has better explainability, while our approach succeeds in style adaptation.

The following sentences are examples generated by our approach to describe how the two politicians talk about Europe and China:

> Trump style: *They'll compare us to Europe and we did very well.*
> Obama style: *That's the spirit that binds us to Europe.*
>
> Trump style: *If Biden is elected, China will own America.*
> Obama style: *China is firmly committed to the path of peaceful development.*

In these two examples, the Trump-style sentences convey negative opinions about Europe and China. The Obama-style sentences describe the two entities from a more neutral point of view. This phenomenon suggests that our approach learns the language use of the two politicians in terms of the usual way they describe other countries, where Trump typically emphasizes the competition while Obama focuses on the cooperation between the countries.

Our qualitative results suggest that the cross-aligned autoencoder often successfully adapts the style by using stylized words, but fails to preserve the explanation sufficiently. In contrast, our proposed approach often successfully reformulates the content in different words. We also observe that after applying this approach, a pre-trained style classifier predicts that the generated text has the desired style.

In conclusion, this section has summarized our contributions based on our experiments on political speeches as a distance-supervision data to study service explanations: (1) We have studied approximately how to generate natural language explanations of services by considering the problem as a data-to-text task. (2) We demonstrated that our approach can generate explanations for the desired language style.

Text Professionalization as an Example of Style-Adjusted Generation

After generating the explanations, the next goal is to adjust the language style of a given explanation, which is called the *text style transfer* task in NLP. Unlike the previous task, where the input was a tuple of $< name, value >$, here the input is a natural language sentence ready to be style adjusted.

In text-style transfer, the goal is to rewrite a text in a defined style while keeping the content of the text similar. Text-style transfer tasks have become increasingly popular in the era of deep learning. Depending on the goal of the task, the term "style" can refer to different attributes of a text, such as media bias [CWAS18] or the speaking style of politicians as mentioned above.

In the following, we summarize our main findings from two text-style transfer studies on a specific task that exemplifies the adjustment of a text to a specific proficiency level [Mis21; Pal22; CASW20]:

> **Text professionalization:** Given a text, rewrite it so that its style becomes more formal while preserving as much of its original content as possible.

The following two sentences show an example of how a text in "normal style" can be rewritten in a more professional way while still conveying the same information:

> Normal: *Oktoberfest happens every year in Munich and many people take part in it, where they drink beer at the Theresienwiese square.*
>
> Professional: *The Oktoberfest occurs annually in the German city of Munich and is celebrated by a large crowd, where participants gather in beer-drinking festivities in the Theresienwiese square.*

The professional sentence in the example above uses synonyms for "happens" and "every year" that reflect a more sophisticated style ("occurs" and "annually"). It also provides more context by adding the country in which the city is located ("the German city of Munich"). Finally, the professional sentence uses other phrases, such as "is celebrated by a large crowd" and "participants gather in beer-drinking festivities," which enrich the sentence with elaborate details compared to the standard phrases "many people take part in it" and "they drink beer."

For our research, we used SSCORPUS [KK16]. This corpus has 493,000 aligned sentences extracted by pairing simple English Wikipedia[7] with standard English Wikipedia[8]. It contains sentences from Wikipedia articles on various topics, which fits our research goal of text professionalization. Each instance in the corpus is composed of a sentence from standard Wikipedia, the corresponding sentence from simple Wikipedia, and a similarity score between them. The sentence from standard Wikipedia uses formal and professional language that fits the characteristics of how professional language should be. It is followed by a simpler version of the same sentence extracted from the simple Wikipedia, which uses simple and regular words that are easier to understand. Finally, the similarity score, ranging from 0 to 1, describes the similarity in meaning of the two sentences: A score of 1 means that the two sentences are identical, while a score of 0 means no similarity at all.

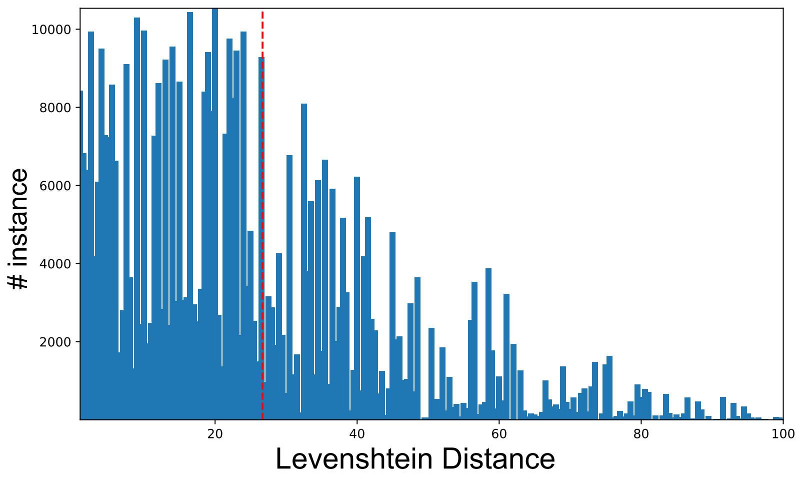


Figure 19: *The Levenshtein distance distribution in SSCORPUS. The red line marks the threshold we used for filtering.*

The first step was to filter the given data set. In particular, the original use of SSCORPUS

[7]`https://simple.wikipedia.org`, last accessed 2023-02-06.

[8]`https://en.wikipedia.org`, last accessed 2023-02-06.

	Automatic Evaluation		Manual Evaluation		
	BLEU	ROUGE	Factuality	Fluency	Professionality
Simple	-	-	4.07	3.94	3.95
Professional	-	-	4.16	4.03	4.01
Generated	0.59	0.64	4.21	4.07	4.07

Table 3: *Evaluation of style-adjusted generation: automatic scores (BLEU, ROUGE) and manually assigned scores from 1 (worst) to 5 (best) in terms of factuality, fluency and professionality for the simple sentence, the professional sentence, and the generated sentence.*

is to transfer the text from its simple version to its professional version. However, we could not simply switch the inputs and outputs of SSCORPUS, because the professional versions of the text not only use more professional words but also add more details. In other words, transferring to simple texts results in removing these details, while transferring to professional texts means adding these details. In practice, it is easier to remove information than to add it. Sometimes it is almost impossible to add extra information out of nothing. As a measure, we used the Levenshtein distance between the simple and professional versions and discarded those pairs that added too much new information (having a too-high Levenshtein distance). Figure 19 shows the distribution of the Levenshtein distance in SSCORPUS and the threshold (25) for selecting the better instances. In order to obtain a professionalized text, we used the context of the text as an additional input in our approach. Specifically, we used entity information to model the context, where the entity information was extracted using the NER tool spaCy. Figure 20 shows the architecture of the model within our approach.

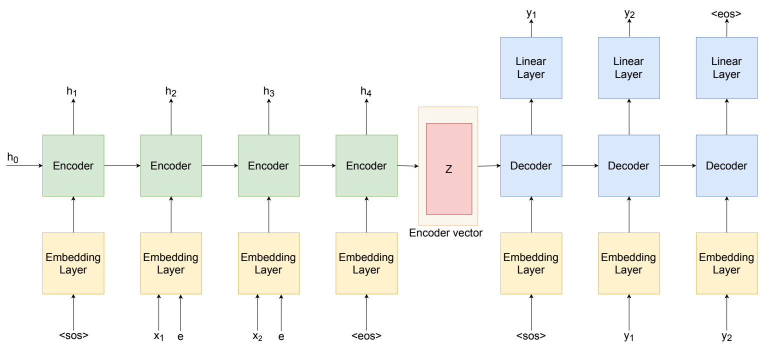


Figure 20: *The architecture of our model adapted from Mishra [Mis21]: x_n and y_n denote the n-th input and output tokens, h_t is the hidden state in the step t, <sos> and <eos> are the start and end of sentence tokens, z is the representation of the input, and e is the entity information as an additional input to the model.*

The final step was to select a base model for fine-tuning on our data set. We chose BART from Facebook AI, which was pre-trained on the CNN/DailyMail data set. At the time of developing our approach, BART showed state-of-the-art performance on many NLP tasks, such as machine translation, question answering, and text simplification.

We re-evaluated our approach using both automatic and manual measures. In the automatic evaluation, we considered BLEU (Bilingual Evaluation Understudy) and ROUGE (Recall-Oriented Understudy for Gisting Evaluation). For manual evaluation, we used Amazon Mechanical Turk to evaluate the text in terms of fluency, factuality, and professionality. The evaluation results are shown in Table 3. The BLEU and ROUGE scores indicate that the generated sentences are close to professional sentences. In terms of manual evaluation, the judges gave very high scores to the generated sentences in all three aspects, indicating that our approach successfully transforms simple sentences into professional sentences.

The following is an example of a professional sentence generated by our approach:

> Simple: *During pregnancy, the endometrium develops a lot of glands and blood vessels.*
>
> Professional: *During pregnancy, the glands and blood vessels in the endometrium further increase in size and number.*
>
> Generated: *During pregnancy, the endometrium develops a large number of glands and blood vessels, known as endometrial granules.*

Here, the professional sentence changes the part “develops a lot of” into a more precise description that the endometrium “increase in size and number.” The generated sentence also professionalizes the simple text in a similar manner by changing it to “develops a large number.” Furthermore, the generated sentence adds a new concept “endometrial granules.” The model appears to learn such word usage from the pre-trained weights of the BART model. In addition, a judge from Amazon Mechanical Turk commented that *“all sentences are good, but the generated one is the clearest and most descriptive.”*

In summary, this research investigates one of the core topics of Subproject B1: adjusting a text to the user’s language. Depending on the professionality of the users, our approach can transfer the style of the text to a more professional version. In principle, it could also be easily adapted to transfer the texts to a less professional version. From the OTF computing perspective, we study how to generate explanations for different user groups. For example, machine learning beginners would be simple explanations with few technical terms. On the other hand, experts would expect explanations with algorithm details.

3 Impact and Outlook

Our work is embedded in the context of OTF computing where our goal was to allow end users to participate in the specification process by supporting natural language requirements without limiting expressiveness. The richness and ambiguity of natural language, as well as its imprecision, may cause deficient service descriptions and hence posed a challenge for processing, which had to be mastered. By incorporating natural language generation and style transfer techniques, we enabled communication with users in a language they can understand.

3.1 Optimization of Service Specifications

Until now, most of the existing work has focused on the use and development of semi-formal or formal specification languages, which means that there have been few approaches in the area of requirements extraction. In particular, there has been a lack of linguistic resources. To be able to process natural language requirements, we developed our own knowledge-based resources (e.g., requirements corpora and compensation graphs [GB16]) and an approach (REaCT, Requirements Extraction and Classification Tool) that detects on-topic statements in service descriptions. Based on this, the insufficiently specified end-user requirements could be analyzed (i.e., identified, extracted, and formalized) to compensate for ambiguity, vagueness, and incompleteness.

In order to efficiently and reliably compensate for deficient service descriptions, strategies have been developed to select appropriate algorithms [GB17]. The goal was to obtain concrete natural language service specifications depending on certain linguistic inaccuracies (e.g., ambiguity and incompleteness) in order to best specify the user's original requirements. For this purpose, a parameterized model has been developed [BG18] that automatically chooses the most appropriate strategy to compensate for linguistic deficiencies in service descriptions. On the one hand, strategies were developed for demand-oriented and performant control of appropriate compensation procedures. On the other hand, it was shown that the strategy configuration itself can be performed in a data-driven manner in real-time, depending on the situation, and can lead to better results than predefined rule sets. Also, the proven learning effects of the prototype CORDULA (Compensation of Requirements Descriptions Using Linguistic Analysis) [FBG18] through caching during (domain-specific) lexical disambiguation could be achieved in practice. Furthermore, in line with our earlier research [WSE11], we questioned the predefined order of text analysis steps that has been established in natural language processing for years and achieved good results by deviating from the classical NLP pipeline. In this way, the respective input text itself reveals which inaccuracies are present, and the necessary compensation steps as well as their order are determined in a data-driven manner, taking into account the interactions between ambiguous and incomplete linguistic expressions.

Based on this work, we moved to bidirectional requirement compensation, since unidirectional and analytic approaches do not produce fully actionable requirements. Therefore, we used the previously constructed processing pipeline and enriched it with a chat functionality that can access an underlying knowledge graph created specifically for the domain. The chatbot ensures that user requests are processed end-to-end. With access to the knowledge graph, the chatbot suggests application features to the user, fills out templates in the background, and asks questions that help the chatbot determine the type of application being requested. After evaluating this approach with real users, we will be able to generate software requests based on natural language [KAG22].

3.2 Generation of Service Explanations

Learning to explain sophisticated concepts is never an easy task, even for humans. In our research, we faced two major challenges: First, we needed data sets to analyze human explanations and to train explanation generation methods. Second, we wanted to adapt the style of the explanations to the language of the users.

We decided to use data sets from other task domains because no service-related explanation data were available. In the first highlight we presented in Section 2.2, we outlined how we adapted the transcriptions of politicians' speeches into a data set for explanations. In this way, we were able to collect the explanations remotely, rather than having to ask experts to write the explanations for our study. In the second highlight, we wanted to provide users with different professional levels of explanations. Again, no perfectly matched data were available. To solve this problem, we used the existing SSCORPUS from another task and further filtered it to meet our needs for text professionalization. In both highlights, we described how to solve the problem of data sparseness and also demonstrated that we can use the data sets created in the two highlights to properly study the desired research topics.

The second challenge was to adjust the style of the explanations. In the two highlights, we have shown that such text style transfer can be done in two different ways: On the one hand, we trained multiple data-to-text explanation generators, one for each style; in the example of politicians' explanations, we had two generators, one for each politician. On the other hand, we trained a neural style transfer model and applied it to the explanation results. In the task studied, the professionality of the texts was successfully changed after the texts were generated. Both approaches have their advantages and disadvantages. In the first solution, we trained the model to perform explanation generation and style adjustment in one approach, i.e., as an end-to-end model. In the second solution, we treated the two tasks separately, suggesting a cascading architecture. In general, an end-to-end approach may achieve better performance, while a cascade approach may suffer from error propagation. However, the cascade approach may be easier to interpret, and one can optimize each component individually.

In summary, in the two highlights we have discussed how we approached the task of style-adjusted explanation generation. With our approaches, we contribute to Subproject B1 and fulfill one of its main goals: user-adjusted explanation generation.

3.3 Outlook on Explainable Results

Since there are different ways of elaborating software requirements, the question of explainability arises when many improvements have been made. That is, the reasons for correcting certain parts of the requirements and the reasons for the improvements must be clear to the users. Thus, end users should be informed so that they understand the composition of the service and are satisfied with its features. To date, corrections are sometimes highlighted or compared to the input, but there is no explanatory information about the changes. In the OTF context, there are several challenges: Natural language input from end users primarily shapes the software composition, although the nature of the composition (the actual software service selection step) also has an impact. The result is a transparent composition process that includes natural language inaccuracy detection and compensation methods to improve comprehension. To what extent ChatGPT [9] can be useful and adapted for this domain-specific task remains to be investigated.

[9] `https://openai.com/blog/chatgpt/`, last accessed 2023-02-05.

Bibliography

[ACGW21] ALSHOMARY, M.; CHEN, W.-F.; GURCKE, T.; WACHSMUTH, H.: Belief-based Generation of Argumentative Claims. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 2021, pp. 224–233

[Ahm22] AHMED, M.: Knowledge Base Enhanced & User-centric Dialogue Design for OTF Computing. MA thesis. Paderborn University, Germany, 2022

[Bäu17] BÄUMER, F. S.: Indikatorbasierte Erkennung und Kompensation von ungenauen und unvollständig beschriebenen Softwareanforderungen. PhD thesis. Paderborn University, Germany, 2017

[BBP15] BÖRDING, P.; BRUNS, M.; PLATENIUS, M. C.: Comprehensive Service Matching with MatchBox. In: *Proceedings of 10th Joint Meeting on Foundations of Software Engineering*. ACM, 2015, pp. 974–977

[BG18] BÄUMER, F. S.; GEIERHOS, M.: Flexible Ambiguity Resolution and Incompleteness Detection in Requirements Descriptions via an Indicator-based Configuration of Text Analysis Pipelines. In: *Proceedings of the 51st Hawaii International Conference on System Sciences*. 2018, pp. 5746–5755

[BKG19] BÄUMER, F. S.; KERSTING, J.; GEIERHOS, M.: Natural Language Processing in OTF Computing: Challenges and the Need for Interactive Approaches. In: *Computers* 8 (2019), no. 1, pp. 1–14

[Bül21] BÜLLING, J.: Political Speaker Transfer–Learning to Generate Text in the Styles of Barack Obama and Donald Trump. MA thesis. Paderborn University, Germany, 2021

[CASW20] CHEN, W.-F.; AL KHATIB, K.; STEIN, B.; WACHSMUTH, H.: Detecting Media Bias in News Articles using Gaussian Bias Distributions. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. 2020, pp. 4290–4300

[CASW21] CHEN, W.-F.; AL KHATIB, K.; STEIN, B.; WACHSMUTH, H.: Controlled Neural Sentence-Level Reframing of News Articles. In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. ACL, Nov. 2021, pp. 2683–2693

[CWAS18] CHEN, W.-F.; WACHSMUTH, H.; AL-KHATIB, K.; STEIN, B.: Learning to Flip the Bias of News Headlines. In: *Proceedings of the 11th International Conference on Natural Language Generation*. ACL, Nov. 2018, pp. 79–88

[DG16] DOLLMANN, M.; GEIERHOS, M.: On- and Off-Topic Classification and Semantic Annotation of User-Generated Software Requirements. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. ACL, Nov. 2016, pp. 1807–1816.

[FBG18] FRIESEN, E.; BÄUMER, F. S.; GEIERHOS, M.: CORDULA: Software Requirements Extraction Utilizing Chatbot as Communication Interface. In: *Joint Proceedings of REFSQ-2018 Workshops, Doctoral Symposium, Live Studies Track, and Poster Track co-located with the 23rd International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2018)*. Vol. 2075. CEUR-WS.org, 2018

[FdSG14] FERRARI, A.; DELL'ORLETTA, F.; SPAGNOLO, G. O.; GNESI, S.: Measuring and Improving the Completeness of Natural Language Requirements. In: *Requirements Engineering: Foundation for Software Quality*. Ed. by SALINESI, C.; WEERD, I. van de. Springer, 2014, pp. 23–38

[GB16] GEIERHOS, M.; BÄUMER, F. S.: How to Complete Customer Requirements: Using Concept Expansion for Requirement Refinement. In: *Proceedings of the 21st Int. Conf. on Applications of NL to Information Systems (NLDB)*. Ed. by MÉTAIS, E.; MEZIANE, F.; SARAEE, M.; SUGUMARAN, V.; VADERA, S. E. Vol. 9612. LNCS. Springer, 2016, pp. 37–47

[GB17] GEIERHOS, M.; BÄUMER, F. S.: Guesswork? Resolving Vagueness in User-Generated Software Requirements. In: *Partiality and Underspecification in Information, Languages, and Knowledge*. Ed. by CHRISTIANSEN, H.; JIMÉNEZ-LÓPEZ, M. D.; LOUKANOVA, R.; MOSS, L. S. Partiality and Underspecification in Information, Languages, and Knowledge. Cambridge Scholars Publishing, 2017. Chap. 3, pp. 65–108

[GSB15] Geierhos, M.; Schulze, S.; Bäumer, F. S.: What did you mean? Facing the Challenges of User-generated Software Requirements. In: *Proceedings of the 7th ICAART*. Ed. by Loiseau, S.; Filipe, J.; Duval, B.; Herik, J. van den. Special Session on PUaNLP 2015. SCITEPRESS, 2015, pp. 277–283

[HB15] Husain, S.; Beg, R.: Advances in Ambiguity less NL SRS: A review. In: *Proceedings of ICETECH 2015*. Mar. 2015, pp. 221–225

[KAG22] Kersting, J.; Ahmed, M.; Geierhos, M.: Chatbot-Enhanced Requirements Resolution for Automated Service Compositions. In: *HCI International 2022 Posters*. Ed. by Stephanidis, C.; Antona, M.; Ntoa, S. Vol. 1580. CCIS. Springer, 2022, pp. 419–426

[KK16] Kajiwara, T.; Komachi, M.: Building a monolingual parallel corpus for text simplification using sentence similarity based on alignment between word embeddings. In: *Proceedings of COLING 2016*. 2016, pp. 1147–1158

[Kör14] Körner, S. J.: RECAA - Werkzeugunterstützung in der Anforderungserhebung. PhD thesis. KIT, Feb. 2014

[Mis21] Mishra, A.: Computational Text Professionalization using Neural Sequence-to-Sequence Models. MA thesis. Paderborn University, Germany, 2021

[MLC14] Moens, M.-F.; Li, J.; Chua, T.-S., eds.: *Mining User Generated Content*. CRC Press, 2014

[Pal22] Palushi, J.: Domain-aware Text Professionalization using Sequence-to-Sequence Neural Networks. BA thesis. Paderborn University, Germany, 2022

[PJR+16] Platenius, M. C.; Josifovska, K.; Rooijen, L. van; Arifulina, S.; Becker, M.; Engels, G.; Schäfer, W.: *An Overview of Service Specification Language and Matching in On-The-Fly Computing (v0.3)*. Technical Report. HNI, Paderborn University, Germany, 2016

[Pla13] Platenius, M. C.: Fuzzy Service Matching in On-the-fly Computing. In: *Proceedings of the 2013 9th Joint Meeting on FSE*. ESEC/FSE'13. ACM, 2013, pp. 715–718

[SJ15] Shah, U. S.; Jinwala, D. C.: Resolving Ambiguities in Natural Language Software Requirements: A Comprehensive Survey. In: *SIGSOFT Software Engineering Notes* 40 (Sept. 2015), no. 5, pp. 1–7

[SLBJ17] Shen, T.; Lei, T.; Barzilay, R.; Jaakkola, T.: Style transfer from non-parallel text by cross-alignment. In: *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017, pp. 6833–6844

[TB13] Tjong, S. F.; Berry, D. M.: The Design of SREE – A Prototype Potential Ambiguity Finder for Requirements Specifications and Lessons Learned. English. In: *Requirements Engineering: Foundation for Software Quality*. Ed. by Doerr, J.; Opdahl, A. L. Vol. 7830. LNCS. Springer, 2013, pp. 80–95

[WA22] Wachsmuth, H.; Alshomary, M.: "Mama Always Had a Way of Explaining Things So I Could Understand": A Dialogue Corpus for Learning to Construct Explanations. In: *Proceedings of the 29th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, Oct. 2022, pp. 344–354

[WSE11] Wachsmuth, H.; Stein, B.; Engels, G.: Constructing Efficient Information Extraction Pipelines. In: *20th ACM International Conference on Information and Knowledge Management*. Ed. by Berendt, B.; Vries, A. de; Fan, W.; Macdonald, C.; Ounis, I.; Ruthven, I. ACM, Oct. 2011, pp. 2237–2240

[ZSG+20] Zhang, Y.; Sun, S.; Galley, M.; Chen, Y.-C.; Brockett, C.; Gao, X.; Gao, J.; Liu, J.; Dolan, B.: DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation. In: *Proceedings of the 58th Annual Meeting of the ACL*. Ed. by Jurafsky, D.; Chai, J.; Schluter, N.; Tetreault, J. ACL. 2020, pp. 270–278