Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Dialogue Management for Conversational AI

by

Ing. Petr Marek

Doctoral Thesis

Supervisor: Ing. Václav Chudáček, Ph.D.
Supervisor-Specialist: Ing. Jan Šedivý, CSc.

Ph.D. programme: Electrical Engineering and Information Technology
Branch of Study: Artificial Intelligence and Biocybernetics

Prague, October 2023

Supervisor:
Ing. Václav Chudáček, Ph.D.
Czech Institute of Informatics, Robotics and Cybernetics
Czech Technical University in Prague
Jugoslávských partyzánů 1580/3
160 00 Prague 6
Czech Republic

Supervisor-Specialist:
Ing. Jan Šedivý, CSc.
Czech Institute of Informatics, Robotics and Cybernetics
Czech Technical University in Prague
Jugoslávských partyzánů 1580/3
160 00 Prague 6
Czech Republic

Abstract

Conversational AI is a computer program that allows us to communicate with computers using natural language. The central component of conversational AI is the dialogue manager, which we can imagine as its brain. Traditional approaches to conversational AI and dialogue management relied primarily on a rule-based systems. The main advantage of rule-based systems is their predictability and controllability. However, the limiting factor of rule-based systems is the need for domain experts to design the rule-based system and limited flexibility to unexpected conversational situations.

Recent years brought significant advancement in language models. Language models approximate probabilities of a series of words based on training text corpora and can be used to generate text. This ability allows us to utilize language models in conversational AI. In conversational AI applications, we also call language models as neural response generators. The advantage of neural response generators is their ability to learn conversation in a data-driven way, which leads to better flexibility and generalization to unexpected conversational situations. However, the limiting factor of neural response generators is their unpredictability and uncontrollability.

This thesis aims to bridge the rule-based and language model-based approaches to conversational AI. The main contribution of the thesis is Hybrid dialogue management. Hybrid dialogue management combines rule-based systems' controllability with the flexibility of neural response generators. The core principle of Hybrid dialogue management is a Pragmatic level of control. The Pragmatic level of control enables us to control the response properties of language models to follow dialogue scenarios specified by a rule-based system.

We propose the PraGPT language model capable of the Pragmatic level of control intended for Hybrid dialogue management. Our aim was to propose an efficient model in the number of parameters with comparable quality on limited domains as recent state-of-the-art language models. We evaluate the model in real world setting by unitizing mobile application operated by conversational AI.

Moreover, we propose innovations for additional components of conversational AI necessary for the practical application of Hybrid dialogue management. We propose a generative adversarial network for out-of-domain data generation. We propose a summarization method based on Named entity density. We also introduce several approaches for topic tracking and flow control dialogue management. Finally, to demonstrate that all introduced innovations are interconnected, we present the architecture of a conversational AI system that combines all the technologies we described.

The motivation to study the combination of rule-based systems with language models arose thanks to the socialbot Alquist. Alquist is a finalist of Amazon Alexa Prize Socialbot Grand Challenges 1, 2 and 3 and winner of Grand Challenge 4. The fact that the Alquist was deployed in real-world conditions and served a large number of users necessitated the proposed solution to be not only academically progressive but practically applicable as well.

Keywords:
conversational AI, dialogue system, dialogue management, out-of-domain recognition, generative adversarial network, summarization, named entity recognition, topic tracking, flow control, hybrid dialogue management, language modelling, neural response generator, dialogue act

Abstrakt

Konverzační umělá inteligence je počítačový program, který umožňuje komunikovat s počítači pomocí přirozeného jazyka. Ústřední složkou konverzační umělé inteligence je dialogový manažer, kterého si můžeme představit jako její mozek. Tradiční přístupy ke konverzační umělé inteligenci a dialogovému managementu byly převážené založeny na pravidlových systémech. Hlavní výhodou pravidlových systémů je jejich předvídatelnost a kontrolovatelnost. Jejich omezujícím faktorem je nutná participace doménových expertů na návrhu pravidel a omezená flexibilita vůči neočekávaným konverzačním situacím.

V posledních letech došlo k významnému pokroku v oblasti jazykových modelů. Jazykové modely odhadují pravděpodobnosti sekvencí slov na základě tréninkových korpusů a mohou být použity ke generování textu. Tato schopnost nám umožňuje využívat jazykové modely v konverzační umělé inteligenci. Jazykové modely se v aplikacích konverzační umělé inteligence také označují jako neuronové generátory odpovědí. Výhodou neuronových generátorů odpovědí je jejich schopnost učit se konverzovat na základě dat, díky čemuž vynikají flexibilitou a adaptací na neočekávané situace v konverzaci. Omezujícím faktorem neuronových generátorů odpovědí je však jejich nepředvídatelnost a nekontrolovatelnost.

Tato práce si klade za cíl propojit přístupy konverzační umělé inteligence založené na pravidlových systémech a jazykových modelech. Hlavním přínosem práce je Hybridní dialogový management. Hybridní dialogový management kombinuje ovladatelnost pravidlových systémů s flexibilitou neuronových generátorů odpovědí. Základním principem Hybridního dialogového managementu je Pragmatická úroveň řízení. Pragmatická úroveň řízení umožňuje určovat vlastnosti odpovědí generovaných jazykovým modelem tak, aby zapadaly do scénářů dialogů pravidlového systému.

V práci navrhujeme jazykový model PraGPT schopný Pragmatické úrovně řízení. Model je navržený pro využití v Hybridním dialogovém managementu. Naším cílem bylo navrhnout efektivní model z pohledu počtu parametrů se srovnatelnou kvalitou na omezených doménách jako mají nejlepší jazykové modely. Model jsme vyhodnotili v reálném nasazení do mobilní aplikace založené na konverzační umělé inteligenci.

Navíc navrhujeme inovace dalších součástí konverzační umělé inteligence, které jsou nezbytné pro praktické použití Hybridního dialogového managementu. Navrhujeme generativní adverzariální síť pro generování mimodoménových dat. Navrhujeme metodu sumarizace založenou na hustotě entit. Představujeme také několik přístupů dialogového managementu pro sledování témat a řízení dialogů. V závěru představujeme architekturu konverzační umělé inteligence, která propojuje všechny v práci popsané technologie s cílem ukázání jejich provazby.

Motivace k výzkumu metod propojující pravidlové systémy s jazykovými modely vyvstala díky socialbotu Alquist. Alquist je finalistou 1., 2. a 3. ročníku soutěže Amazon Alexa Prize Socialbot Grand Challenge a vítězem 4. ročníku soutěže. Skutečnost, že Alquist byl nasazen v reálných podmínkách velkému počtu uživatelů, vyžadovala, aby navržené řešení bylo nejen akademicky progresivní, ale i prakticky použitelné.

Klíčová slova:

konverzační umělá inteligence, dialogový systém, dialogový management, rozpoznání mimodoménových dat, generativní adversariální síť, sumarizace, rozpoznání pojmenovaných entit, sledování tématu, řízení dialogu, hybridní dialogový management, modelování jazyka, neuronový generátor odpovědí, dialogové akty

Acknowledgement

I thank Ing. Jan Šedivý, CSc. and my supervisor Ing. Václav Chudáček, Ph.D., for their advice, support, and guidance.

This research has been partially supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS17/092/OHK3/1T/13, SGS18/103/OHK3/1T/37, SGS19/091/OHK3/1T/37, and SGS20/092/OHK3/1T/37. Amazon also supported the research during the Alexa Prize Socialbot Grand Challenge 1, 2, 3, and 4.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This dissertation thesis aims to propose practical solutions to the problems of dialogue management deployed to the real-world applications of conversational artificial intelligence. In recent years, there has been a boom in chatbots, virtual assistants and voice-enabled systems based on conversational AI. At the core of those systems is a dialogue manager. Most practical applications of conversational AI relied at least partially on rule-based dialogue management systems. While they provide nearly complete control over the flow of conversation, they are costly to develop and are perceived by users as rigid and lacking understanding.

This was the case until the rapid advances of language models based on the transformer architecture of neural networks and derived generative pretrained transformers (GPT) [Radford et al., 2018]. Those language models can conduct a conversation on a level nearly indistinguishable from human communication. Their ability opened the possibility to utilize language models as neural response generators in conversational AI. However, using neural response generators based on language models in conversational AI poses problems connected to dialogue management. Namely, large language models has a limited ability to follow a goal, they tend to hallucinate facts, it is unclear how to combine them with rule-based approaches offering control over the dialogue, and the cost to operate a state-of-the-art language model is exceptionally high. Those facts limit their applicability to real-world applications. Thus, language models used as neural response generators opened new research problems.

The main contribution of this thesis is Hybrid dialogue management. Hybrid dialogue management solves the most pressing issues with large language models in conversational AI. Hybrid dialogue management provides a Pragmatic level of control to large language models. The Pragmatic level of control allows us to specify the properties of the generated response so that the whole flow of the conversation reaches the desired outcome.

We show that thanks to Hybrid dialogue management, smaller and thus cheaper to operate, architectures of large language models achieve similar or better performance than significantly larger architectures. Thus, we propose several methods whose combination and application into dialogue manager addresses all presented limitations of neural response generators based on large language models. Namely, they do not aim for specified conversation goals, hallucinate, cannot be combined with rule-based systems, and have high operational costs.

We introduce several methods of dialogue management. We divide the methods into topic tracking and flow control sub-problems. The topic tracking part of dialogue management decides what will be a large topic that the system should focus on in the following dialogue turns. Flow control makes immediate decisions inside of the dialogue flows.

Moreover, we propose OodGAN, a method to improve the detection of situations when the flow of the rule-based system is too rigid and unable to handle unexpected user input. Consequently, switching to a neural response generator capable of flexible reactions would be advantageous. The proposed method uses a generative adversarial network to generate synthetic examples for an intent recognition classifier.

In order to minimize the hallucination of the neural response generator and insert factual information into conversation while keeping the input size into the language model small, we propose an efficient summarization technique for news articles. The technique is based on entity recognition and the density of named entities in sentences.

Finally, to show that all proposed methods are interconnected, we propose an architecture of conversational AI, which combines all the proposed methods into a single system.

The need for proposed practical solutions arose thanks to open-domain conversational AI Alquist [Pichl et al., 2018, 2020b,a, Konrád et al., 2021]. Alquist is a socialbot competing in the global Alexa Prize Socialbot Grand Challenges organized by Amazon for student teams. We tested the proposed methods as part of Alquist. Thousands of real-world users of Amazon's voice assistant Alexa used Alquist. Thus, it strongly required the proposed methods to be highly reliable and practical. The fact that Alquist was a global winner of the Amazon Alexa Prize Socialbot Grand Challenge 4 [Hu et al., 2021], the second-place winner of Grand Challenge 1 [Ram et al., 2018] and 2 [Khatri et al., 2018], and the third-place winner of Grand Challenge 3 [Gabriel et al., 2020] serves as proof of the usefulness of proposed methods.

1.1 Structure of Thesis

We start the thesis in chapter 2 by introducing various theoretical concepts important for conversational AI, dialogue management and language models. We present a brief introduction to the theory of conversation useful for conversational AI. Next, we present a history of conversational AI. We present the main components of conversational AI and introduce their task. We put particular emphasis on the working principles of language models. Finally, we present evaluation methods of conversational AI, including important distinctions between upstream and downstream evaluation.

Chapter 3 proposes a generative adversarial network for out-of-domain data generation. This method improves the detection of out-of-domain inputs. Out-of-domain detection is an important feature for a dialogue manager, according to which it can decide when to switch out of a rigid but expertly designed conversation flow of a rule-based system to a flexible neural response generator based on a language model. We present state-of-the-art approaches and propose the OodGAN. We describe data and metrics, and we present experiments. Finally, we demonstrate improved performance in out-of-domain detection in the results section.

Chapter 4 presents a method for summarization of articles as an efficient way to provide knowledge into a neural response generator based on a language model to ground its response in the knowledge. Thanks to our proposed method, the dialogue manager can request knowledge in the form of an article relevant to the recent conversation. Our proposed method summarizes the core information of the article and uses it in the conversation to which the neural response generator grounds its response. Grounding limits the neural response generator's tendency to hallucinate. We present the state-of-the-art methods in extractive as well as abstractive summarization. We propose our method together with a new metric based on named entities. We present data and experiments. Lastly, we present results and examples of summarizations.

Chapter 5 describes various methods of dialogue management. It describes the distinction between topic tracking and flow control dialogue management. For the flow control, it describes Structured topic dialogues and Dialogue trees. For topic tracking dialogue management, it describes Monolithic topics, Topic graph and Dialogue selector. Finally, it presents results and their discussion based on experience from their application to socialbot Alquist. Methods of dialogue management described in this chapter are expanded in the chapter 6.

Chapter 6 proposes Hybrid dialogue management, which is the most significant contribution of this thesis. First, we present a motivation to use language models in conversational AI. Next, we introduce a Pragmatic level of control of language models and control mechanisms of neural response generators. We demonstrate the usefulness of Hybrid dialogue management on selected dialogue flows. We propose PraGPT, an efficient large language mode with the Pragmatic level of control applicable in Hybrid dialogue management. We introduce the data with their annotations and experiments we performed. Finally, we present the experiment's results demonstrating the practical usefulness of the proposed Hybrid dialogue management based on human evaluation in a real-world application.

Finally, chapter 7 concludes the thesis by joining all proposed methods into a single conversational AI system. It uses Hybrid dialogue management with an efficient language model PraGPT capable of the Pragmatic level of control as a neural response generator. The dialogue manager selects the dialogue thanks to the Dialogue selector, drives the conversation flow thanks to Dialogue trees and decides to switch from a rule-based dialogue flow to a neural response generator based on the out-of-domain recognition trained on examples generated by OodGAN. Moreover, the dialogue manager can request knowledge in the form of a news article, which is summarized by the efficient method using named entities. The summarization is used to ground the responses of neural response generator in knowledge.

Chapter 2

Theory

Chapter theory introduces theoretical concepts of conversational AI, dialogue management and language models used in this thesis. The following chapters will build on top of the concepts we introduce.

2.1 Conversation

We must understand human conversations before designing conversational artificial intelligence that communicates in natural language. According to the Cambridge Dictionary, a conversation is *a talk between two or more people in which thoughts, feelings, and ideas are expressed, questions are asked and answered, or news and information are exchanged* [Conversation]. We also call conversation a dialogue. Human communication is a complex activity between several players conducted on multiple levels. This complexity makes it an extremely challenging and intertwined task for encompassing it by technology.

We divide communication into visual, vocal and verbal levels. The visual level includes gestures, facial expressions or eye contact. Vocal level comprises prosodic features like accent, tone or voice loudness. And lastly, the verbal level deals with lexis, syntax and semantics. The verbal level is the bare minimum for a conversation. Thus, it is a level on which the conversational AI of today most commonly operates.

The discipline that studies conversations is Conversation Analysis [Sacks, 1992]. Harvey Sacks, Emanuel Schegloff and Gail Jefferson introduced Conversational analysis in the late 1960s and early 1970s [Sacks et al., 1978]. The discipline aims to describe and understand phenomenons and patterns that occur in conversations. The primary method of conversational analysis is a study of recorded conversations in taped or transcribed forms [Sidnell, 2010].

2.2 Dialogue Turn

Dividing dialogue into discrete parts is useful for the analysis and algorithmization of conversation. The dialogue can be thought of as a series of dialogue turns. We can formally describe the dialogue D as a series of turns $T_1, T_2, ..., T_n$. The turns are ordered by time in which they appear in ascending order:

$$D = (T_1, T_2, ..., T_n).$$

The T_1 is the first dialogue turn of the dialogue D, and T_n is the last turn of the dialogue D. The dialogue turn consists of two utterances. The first utterance of dialogue turn is called a message. The second utterance is called response. According to the conventions, a user usually creates the message, and the system creates the response. We can define the dialogue turn formally as:

$$T = (m, r),$$

where m is a message and r is a response. The dialogue turn is closely connected to the turn-taking principle. The turn-taking principle describes that while one actor in the dialogue speaks, the other listens. Then the roles switch and the second actor can react. Thus, the dialogue turn contains exactly one switch of roles.

2.3 Dialogue Utterance

We have already introduced utterances as part of dialogue turn, where they play the role of message and response. We can represent the utterance as a tuple containing two pieces of information. The first part of the tuple contains information about who is a speaker sp of the utterance U. The second part of the tuple contains content u of the utterance U. Informally, the utterance describes who said what. Formally, we define utterance as a tuple:

$$U = (sp, u).$$

2.4 Segment of Utterance

Segments divide utterances into sub-parts based on their role in the dialogue. Informally, we can demonstrate segments on dialogue turn t:

$$\begin{aligned} t = (&(\text{Speaker A}, \textit{“Do you like ice cream?”}),\\ &(\text{Speaker B}, \textit{“Yes, I like the vanilla flavour. What is your favourite flavour?”})). \end{aligned}$$

The message *“Do you like ice cream?”* consists of a single segment in the form of a question. The response *“Yes, I like the vanilla flavour. What is your favourite flavour?”* consists of three segments. It starts with the segment *“Yes”* in the form of agreement. Next, there is *“I like the vanilla flavour”* in the form of opinion. Furthermore, there is *“What is your favourite flavour”* in the form of a question.

Dialogue Act	**Utterance**
Statement-non-opinion	Me, I'm in the legal department.
Acknowledge (Backchannel)	Uh-huh.
Statement-opinion	I think it's great
Agree/Accept	That's exactly it.
Abandoned or Turn-Exit	So, -
Appreciation	I can imagine.
Yes-No-Question	Do you have to have any special training?
Non-verbal	[Laughter], [Throat_clearing]
Yes answers	Yes.
Conventional-closing	Well, it's been nice talking to you.

Table 2.1: Example of the most common dialogue acts in the Switchboard corpus [Jurafsky, 1997]

Formally, content u of utterance U can be divided into segments $u = (s_1, s_2, ..., s_n)$. The division is based on the role of the segment in the dialogue and utterance. A dialogue act can describe the role of the segment.

2.5 Dialogue Acts

The dialogue act of a segment of utterance in the dialogue describes what function the segment serves. Dialogue acts are part of speech acts that appear in dialogues. From the most basic point of view, each utterance segment plays a role in the dialogue. Some segments ask questions, others answer, some command and others propose a choice between several options.

There is no universal set of dialogue acts. Instead, there are multiple sets of dialogue acts, and their selection depends on the application. The most notable set of dialogue acts are Dialog Act Markup in Several Layers (DAMSL) proposed by Allen and Core [1997], set of dialogue acts used in Switchboard dialogue corpus (SWBD) proposed by Jurafsky [1997], and the MIDAS set introduced by Yu and Yu [2019], which is the newest set of the three. We present an example of the most common dialogue acts and corresponding utterances from the Switchboard dialogue corpus in Table 2.1.

2.6 Adjacency Pairs

Alternatively, to dialogue turns, we can describe the conversation as a sequence of adjacency pairs. Adjacency pairs serve as organizational units of conversation.

Adjacency pairs are defined using utterances [Schegloff and Sacks, 1973]. The adjacency pair consists of two utterances. The first utterance starts the adjacency pair, while the second depends on the first. A different speaker speaks each utterance. Both utterances

Speaker	**Utterance**	**Dialogue Act**
Speaker A	Hello!	Conventional-opening
Speaker B	Hi.	Conventional-opening
Speaker A	Do you have to have any special training?	Yes-No-Question
Speaker B	Yes	Yes answers
Speaker A	Well, how old are you?	Wh-Question
Speaker B	I'm 36.	Statement-non-opinion
Speaker A	Who would steal a newspaper?	Rhetorical-Questions
Speaker B	Exactly!	Agree/Accept
Speaker A	Excuse me?	Signal-non-understanding
Speaker B	Oh, fajitas	Repeat-phrase
Speaker A	I'm sorry.	Apology
Speaker B	That's all right.	Downplayer
Speaker A	Well, it's been nice talking to you.	Conventional-closing
Speaker B	Bye!	Conventional-closing

Table 2.2: Example of adjacency pairs annotated by SWDA dialogue acts

have a pragmatic meaning. We can use dialogue act to describe this pragmatic meaning.

Most of the conversations can be described by adjacency pairs, and most of the pragmatic goals of dialogues are reached through their usage. Examples of adjacency pairs can be to ask question and provide response, greet and greet in return or make an offer and accept it or refuse it. We show examples of adjacency pairs in Table 2.2.

Conversational AI must understand and use adjacency pairs correctly. We can calculate probabilities of transitioning between the dialogue act of the first utterance of the adjacency pair and the dialogue act of the second utterance of the adjacency pair as a Markov decision process. The calculated Markov decision process can serve as a guiding element for response selection or generation of conversational AI [Boyer et al., 2009].

Until now, we introduced adjacency pairs, assuming each utterance consists of a single segment. However, adjacency pairs can also be extended to work with segments of utterances. The only difference is that the utterance can have as many pragmatic meanings described by dialogue acts as segments contained in it. We show examples of adjacency pairs extended to segments of utterances in Table 2.3.

2.7 Conversational AI

Conversational artificial intelligence is a computer program that conducts conversation in natural language. Those computer programs are also called dialogue systems, dialogue agents, conversational agents or chatbots. Their application ranges from simple agents booking a flight up to complex systems able to conduct therapeutical sessions. Conversational AI needs a medium through which it communicates. If the medium is text, we speak about text-based conversational AI. If the medium is voice, the conversational AI

Speaker	**Segments of Utterances**	**Dialogue Acts**
Speaker A	Hello!	Conventional-opening
	How are you?	Open Question
Speaker B	I'm good.	Statement-opinion
Speaker A	Would you like to join me?	Yes-No-Question
Speaker B	No,	No answers
	I have to go.	Statement-non-opinion
	Bye!	Conventional-closing
Speaker A	What will the weather be like?	Wh-Question
Speaker B	Sorry,	Apology
	I don't know.	Statement-non-opinion
	Why do you ask?	Wh-Question
Speaker A	Excuse me.	Signal-non-understanding
	Why would I do that?	Rhetorical-Questions
Speaker B	I'm sorry.	Apology
	That's all right.	Downplayer

Table 2.3: Example of adjacency pairs extended to segments of utterances

is voice-based. We speak of multimodal conversational AI if the conversational AI uses a combination of text and voice and possibly other modalities as well.

We can divide the conversational AI into two groups: goal oriented and open domain. The goal oriented conversational AI aims to reach a predetermined goal via conversation. The goal is to solve some task. In traditional settings, it can be restaurant table reservation, providing sightseeing information or booking a flight. Despite many challenges that goal oriented conversational AI faces, goal oriented conversational AI's complexity is smaller than open domain counterparts in several aspects. The main reason is that it operates in a tightly limited domain. Thus, the complete intricacy of communication can be artificially limited for this reason, and many of the more advanced understanding and communication techniques are not necessary. The open-domain conversational AI conducts dialogues without any strict limitations regarding the topics. An example of such dialogue can be chit-chat. The open-domain dialogue systems do not focus on solving particular task. However, it can fulfil user's requests, guide them through multi-step processes or solve some assignments as a part of a conversation.

Conversational artificial intelligence is rarely a monolithic program. It consists of several components. We show the schema of conversational AI in Figure 2.1. We can arrange components according to the order in which they process the input utterance. Automatic speech recognition is the first component of conversational AI that uses voice modality. Components jointly referred to as natural language understanding follow next. They are intent recognition, out-of-domain recognition, entity recognition or dialogue act recognition. Each conversational AI can have a different composition of natural language understanding components, which depends on its task. The output of natural language understanding

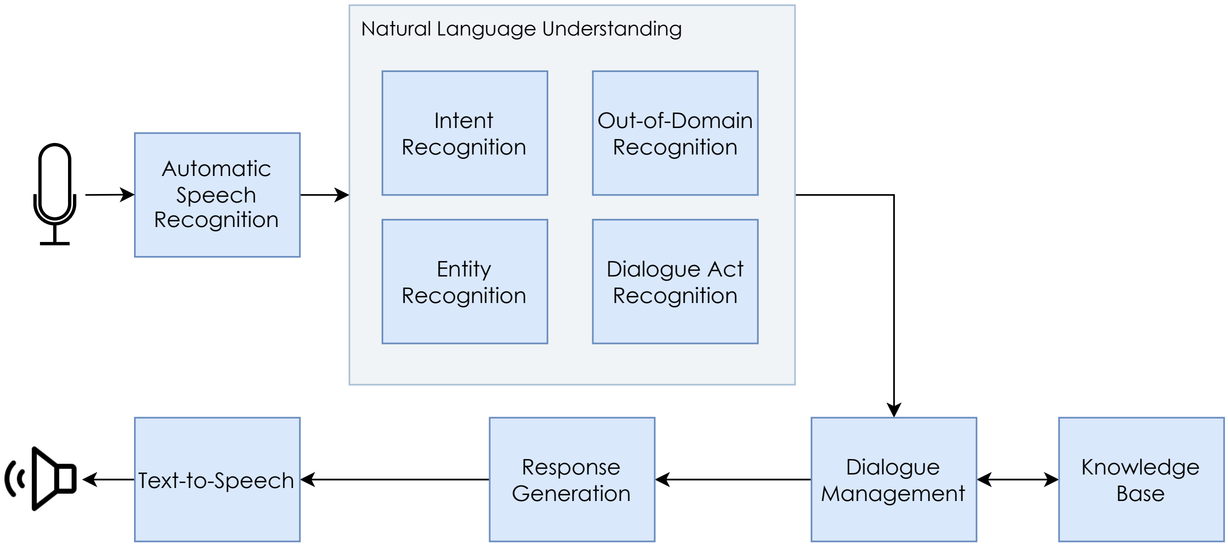


Figure 2.1: Components of conversational AI

is passed into dialogue management. Dialogue management is the only strictly stateful component of conversational AI. It decides the following action of the system based on the results of natural language understanding, the state of the dialogue and information retrieved out of the knowledge base. The information retrieved from the knowledge base might be preprocessed by a summarization algorithm either before they are stored or after they are retrieved from the knowledge base. Next, there are components responsible for response generation. Those components are usually closely tied to dialogue management. Thus, the boundary between dialogue management and response generators might not be strict in practice. Lastly, there is a text-to-speech component for the conversational AIs using voice modality. We will present the components of conversational AI in the following section, which are relevant to the methods we propose. Nevertheless, first, we will present the most notable milestones in the history of conversational AI.

2.8 History of Conversational AI

The foundation work of conversational AI was conducted by Alan Turing in 1950. Turing [1950] proposed the Imitation game, better known as the Turing test nowadays. Turing intended the Turing test as a method to prove or disprove the intelligence of a machine. Player A, player B, and judge C occur in the test. Player A and player B conduct natural conversation using only text. One of the players is a human, and the second is a machine. Judge C is human and observes the conversation without knowing who is human and who is the machine. However, judge C knows that one of the players is the machine. The goal of judge C is to determine which of the two players is the machine. Alan Turing argued that if the judge cannot distinguish whether the machine is player A or B, the machine would pass the test. Passing the test implies that the machine possesses intelligence comparable to humans. An annual Loebner Prize uses the principle of the Turing test to evaluate conversational systems. The competition awards conversational systems that appear to be the most human-like, according to judges. However, there is a criticism of this approach of testing the intelligence of a machine.

Searle [1980] pointed out that even if a machine behaves in a human-like manner, it does not prove its intelligence or understanding. Searle demonstrated his point by using a thought experiment called Chinese Room. Imagine that there is a machine able to pass a modified version of the Turing test. The modified version tests the ability of machine to conduct conversation in Chinese. A judge fluent in Chinese gives input to the machine, and the machine processes input and responses in Chinese. The machine passes the test if the judge cannot distinguish whether they communicates with another Chinese speaker or machine. According to the Turing test, passing the test should imply that the machine understands Chinese.

However, Searle presents follow-up arguments indicating that the machine does not understand Chinese. Imagine that person would take the place of the machine. If we give the person algorithm of the machine, enough paper and plenty of time, the person can follow the steps of the algorithm and produce the same response in Chinese as the machine would, even if the person does not speak or understand Chinese. Thus, the argument of the Chinese room demonstrates that the Turing test does not prove intelligence or understanding of the same principles as humans possess. Nevertheless, the Turing test was the first step towards developing conversational AI.

The next notable milestone in conversational AI was a system called ELIZA proposed by Weizenbaum [1966]. ELIZA is a computer program that enables natural conversation between computer and human. The program uses scripts that describe its communication capabilities. The script contains pattern-transform rules. ELIZA searches for a pattern in each user input and transforms the input into output according to the transformation rule associated with the pattern it finds.

The script for which ELIZA is the most famous is the DOCTOR script. This script simulates a Rogerian psychotherapist. The Rogerian school of psychotherapy utilizes a communication style in which the therapist repeats the patient's words or reformulates the patient's statements to questions. Thus, no fundamental understanding or additional knowledge of the world is needed. This style of communication well suited the initial conversational AIs. Thus, ELIZA was the first system to attempt the Turing test with promising results.

IBM made a subsequent notable step in conversational AI with its system called Watson [Ferrucci et al., 2010] some 40 years after ELIZA. Even though Watson is not a system primarily aimed at conversation, it brought the public's attention to the ability of AI to answer questions in natural language. Watson demonstrated this competence by winning the quiz show Jeopardy. It stood against two human champions of this game.

The Watson was built using IBM's DeepQA and Apache's Unstructured Information Management Architecture frameworks. Frameworks unite several algorithms for natural language understanding, information retrieval, reasoning, and machine learning. Watson used ontological databases like Yago, DBpedia or WordNet as knowledge bases, out of which it constructed answers. The IBM team responsible for creating Watson combined those algorithms and databases into a unified system capable of open-domain question answering.

Siri was the first system that opened up the era of virtual assistants. It was released by Apple in 2011 as a part of its mobile operational system iOS 5 installed on iPhone 4S. Siri enabled users to ask factual questions and control the phone's functions via voice, like setting reminders, dialling calls or setting alarms, and controlling smart home appliances. Next, Amazon introduced Alexa-powered Echo devices in 2013. It transformed mobile smart assistants into smart home speakers. Moreover, Alexa was the first smart assistant allowing developers to extend its capabilities through skill development. Finally, Google released its Google Home smart assistant in 2016. Subsequent generations of Google Home devices included Google Home Hub, the first smart speaker with a screen, making multimodal responses possible.

The next phase of the history of conversational AI began in 2016 when Facebook allowed programmatic access to its messaging platform Messenger. This event started a boom of chatbots. Many companies started to deploy simple, goal-oriented chatbots to their Messenger accounts. Many chatbots were subsequently deployed to websites, too. The main premise was to automate responses to as many customer questions and requests as possible, with the remaining questions and requests directed to human operators. The boom of chatbots also led to the development of NLU and dialogue management tools in the form of online commercial services and open-source libraries. Additionally, new professions like conversational designer or conversational analyst were born as chatbot development became its discipline. In general, chatbot's main contribution was to bringing conversational AI into the mainstream, and they helped create an ecosystem of tools for developing conversational AI. However, chatbots saw success only in narrow and constrained applications. Thus, they never accomplished the promised usefulness, mainly due to limited understating, constrained dialogue scenarios they could handle, and a limited set of prescripted responses.

In 2016, Amazon announced the Alexa Prize competition [Ram et al., 2018]. It aimed to advance the development of conversational AI. Alexa Prize is a worldwide competition for teams of students. The competition's goal is to create a coherent and engaging socialbot capable of conversing about popular topics. The competition's grand challenge is to conduct such a conversation for 20 minutes. Amazon provides research grant for university teams selected to participate and prize money for the top three socialbots. Since 2016, there have been five instalments of Alexa Prize challenges. Each round of competition attracted the attention of the world-leading universities in research and development of conversational AI, like Stanford, University of Washington or Heriot-Watt University. The team of Czech Technical University in Prague was second place winner in the first and second instalments, third place winner in the third instalment and winner of the fourth instalment of the competition. The competition brought numerous innovations in natural language understanding, dialogue management, response generation and knowledge management in open-domain conversations. Moreover, those innovations were tested in practice by users of Amazon Alexa. Thus, competition highlighted many limitations in the practical application of approaches proposed by academia, like inference time or memory requirements, which teams must solve. The grand challenge of 20 minutes long, coherent and engaging conversation remains yet to be broken.

In 2017, Vaswani et al. [2017] proposed a novel architecture of neural network called Transformer, which proved successful in text processing applications. Radford et al. [2018] utilized the novel architecture in 2018 by proposing a Generative Pretrained Transformer (GPT). Their work demonstrated that language models can learn to generate fluent texts and acquire world knowledge by training on a large and diverse corpus of long texts. The proposed approach was scaled up in terms of a number of model parameters in subsequent works proposing GPT-2 [Radford et al., 2019] in 2019 and GPT-3 [Brown et al., 2020] in 2020. The GPT-3 achieved remarkable progress in language generation and numerous zero-shot tasks. A significant proportion of the tasks have an application in conversational AI. The progress in this area gradually brought attention from rule-based dialogue systems to data-driven approaches based on large language models.

The proposed advances in language models inspired research of neural response generators trained on a large corpus of conversations. Zhang et al. [2019] proposed DialoGPT. DialoGPT is based on the architecture of GPT-2. Zhang et al. [2019] demonstrated that DialoGPT initialized by weights of GPT-2 and later finetuned on a corpus of comments extracted from Reddit generate relevant, contentful and context-consistent responses. However, the authors pointed out that responses generated by DialoGPT might be considered unethical, biased or offensive.

Meena [Adiwardana et al., 2020] is another neural response generator proposed by Google Research. The Meena is larger in number of parameters than DialoGPT (176M parameters vs 2.6B parameters) and was trained on 40B words from public domain social media conversations. The authors proposed two human-evaluated metrics: sensibleness and specificity. Sensibleness means that the response makes sense in the given conversation. Specificity means that the response is tailored to the current conversation and can not be used in other contexts. Thus, specific responses are not vague. The experiments with Meena showed that its sensibleness and specificity achieved higher results than other baseline systems, including DialoGPT.

Finally, there is a BlenderBot proposed by Roller et al. [2020]. The first BlenderBot is a neural response generator trained on the Blended Skill Talk dataset. The Blended Skill Talk dataset seamlessly combines skills of using empathy from the EmpatheticDailogues dataset [Rashkin et al., 2018], pretending personality from the PersonaChat dataset [Zhang et al., 2018b] and utilizing knowledge from the Wizard of Wikipedia dataset [Dinan et al., 2018] in the same dialogue. The next breakthrough was BlenderBot 2.0 [Komeili et al., 2021, Xu et al., 2021a]. The system can simultaneously search the internet for knowledge and build long-term memory. Both BlenderBot and BlenderBot 2.0 outperformed DialoGPT and Meena in benchmarking tasks. The latest version is BlenderBot 3.0 [Shuster et al., 2022]. The system is based on the OPT-175B language model [Zhang et al., 2022]. Whereas BlenderBot 2.0 used several distinct models for subtasks like generating a query to the internet, retrieving knowledge from the long-term memory or response generation, BlederBot 3.0 uses OPT-175B for all subtasks. The authors demonstrated significant improvements in BlenderBot 3.0 over BlenderBot 2.0.

The latest progress in conversational AI was achieved thanks to prompting. Prompting is a method in which we specify task for a language model using description in natural language. Optionally, we can add a few solved instances of the problem. We insert the description and solved instances into the model, and the model outputs the solution to our problem. This works in large language models to some extent. However, because large language models are trained only to predict the next probable word, the number of times they fail at task described via prompt is high. Instead, we want to teach the model to perform the specified task. Google researchers made initial attempts with their Text-to-Text Transfer Transformer, simply T5 [Raffel et al., 2020]. They demonstrated that the model performs well on several tasks, like translation, question answering, and classification, if they provide the task description and input to the model as a text and train the model to generate some target text. This allowed them to use the same model, loss function, and hyperparameters for multiple tasks simultaneously. However, the T5's need for task training is not considered a classical prompting.

Ouyang et al. [2022] proposed InstructGPT, the first model capable of prompting. The InstructGPT was trained to be aligned with the prompt through reinforcement learning from human feedback. The authors demonstrated that it significantly improves the alignment with humans on several prompt-described tasks. The next step was ChatGPT [OpenAI, 2023b], which was trained in the same way as InstructGPT, plus it was trained to interact with users via conversation. This allows users to ask follow-up questions or request changes in the produced outputs. Despite its name, the ChatGPT was not trained to be used as a neural response generator for conversational AI. However, it can be directed to behave as such by proper prompting. Finally, GPT-4 [OpenAI, 2023a] scales up the size of any previous GPT model, achieving better results and adding multimodal inputs.

The state-of-the-art language models are currently mainly accessible through API, and the weights or model architectures are scarcely available to the research community. The additional problem is that training state-of-the-art language models capable of following prompts is exceptionally costly. The prices for training reach millions of dollars. For those reasons, there is an active field of research on prompt-following language models, which are significantly cheaper to train and have lower operating costs. The trend started with model LLaMA proposed by Touvron et al. [2023]. LLaMA is a language model with 7B, 13B, 33B and 65B parameter variants. Authors released the weight of this model for the research community. Next, Taori et al. [2023] proposed Alpaca-7B. Alpaca-7B is based on LLaMA 7B. It was finetuned on 52K instruction following examples created by InstructGPT. According to the team, the generation of training examples costs approximately 500 dollars, and the model training costs around 100 dollars. Alternatively, there is a Vicuna [Chiang et al., 2023] whose overall training cost was only approximately 300 dollars. Such models open possibilities for applying them to conversational AI with reasonable costs.

In the following sections, we describe the essential components of conversational AI with a particular focus on components relevant to the thesis.

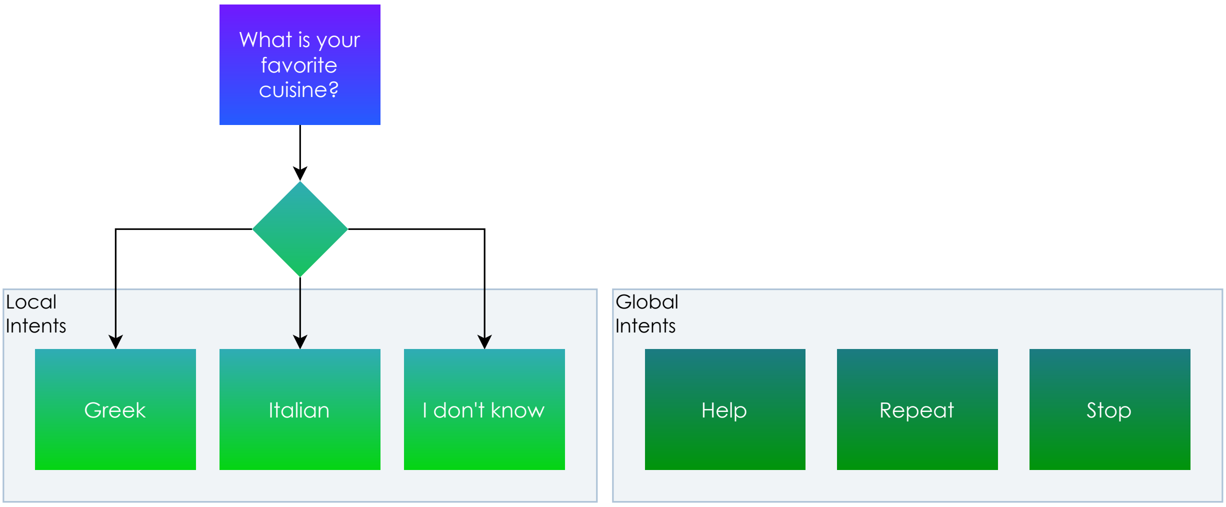


Figure 2.2: Intent recognition using local and global intents

2.9 Intent Recognition

The intent, in general, is an aim to perform some task. In natural language understanding and conversational AI, the intent represents a user's action in the conversation. The action is represented in the user's input. If a user says *"I want to chat about music,"* for example, the intent of this input is to have a conversation about music. Intents are an important feature for a dialogue manager based on which it decides what action the system takes next. We can divide intents into two groups based on their dependency on the state of the dialogue and priority. There are local intents and global intents.

Local intents are dependent on the state of the dialogue. Specific local intents can be recognized only in certain states of the dialogue. For example, the system might ask the user for his favourite type of cuisine. In this state of the conversation, the system can recognize local intents corresponding to the type of cuisine the user likes, like *Like-greek*, *Like-italian* or *Indifferent* intents. Local intents are crucial for the system's ability to continue in the flow of the conversation. They are also more specific for the given dialogue state than global intents. Thus, they have priority over global intents.

Global intents are much less dependent on the state of the conversation. Global intents can be recognized in user inputs in several states of conversation. For example, there can be global intent *Help* that explains what the conversational system is capable of, *Repeat* that repeats the last system's message, or *Stop* that ends the conversation. Global intents are primarily intended to direct and influence the state of the whole system. We present an example of local and global intents in Figure 2.2.

From a technical point of view, intent recognition is a text classification task based on semantic similarity. The task of intent recognition is to classify the user's input into one of the local or global intent classes. The set of intent classes depends on the state of the conversation. Formally, the intent recognition component is trained to maximize the conditional probability $p_\theta(i|x)$ through maximal likelihood estimation, where i is the intent of the input message x. Moreover, $i \in I_L \cup I_G$, where I_L is a set of local intents and I_G is a set of global intents. During the inference, the intent recognition component selects the intent i_n from a set of supported intents $I = I_L \cup I_G$, maximizing the conditional probability

$$i_n = \arg\max_I p_\theta(I|x).$$

We make a strong assumption in the formal definition of the task. The assumption is that we can classify all user inputs to the set of intents our system can handle. However, there are inputs for which this assumption does not hold. Thus, we introduce out-of-domain recognition.

2.10 Out-of-domain Recognition

Out-of-domain (OOD) recognition is an essential task in natural language understanding. The task is to recognize whether or not a given user utterance belongs to the in-domain (IND) distribution. Users usually do not know the limitations of a voice application and assign requests that the system can not act upon. These requests are referred to as OOD since these do not belong to the application's domain. Dialogue managers of conversational AI should be able to handle OOD utterances robustly by not taking unintended action or giving wrong or nonsensical responses, leading to a poor user experience. The problem of OOD handling is divided into two steps. The first is to robustly recognize that the given input is out of domain. The second step is to select an action that handles the out-of-domain input.

OOD recognition is a task closely related to the intent recognition task. Thus, we can approach OOD recognition as an extension of intent recognition. There are three techniques we can use. The first technique adds OOD as another intent to the intent recognition component. Thus, if the intent recognition component initially recognized n intent classes, we train the model to recognize $n+1$ intent classes. The $(n+1)^{th}$ class represents the out-of-domain class. We can describe the problem formally as

$$i = \arg\max_{y \in I \cup ood} p_\theta(y|x),$$

where $p_\theta(y|x)$ is the conditional probability of user message x having an intent I or being classified into out-of-domain class *ood*. In other words, the intent classification selects the most probable intent class or out-of-domain class for user message during inference. The set of all possible intents can be a combination of local and global intents as described in section 2.9.

The advantage of classifying OOD as $(n+1)^{th}$ intent class is its simplicity because we can apply standard methods of intent recognition for this task. The downside of this method is that it requires us to collect OOD data for training before we can deploy the combined intent and out-of-domain classifier to conversational AI.

The second approach is a slight variation of the first. Foremost, we conduct binary classification, determining whether the input is in- or out-of-domain. We classify the intent class in the second step. The second step follows only if the input is classified as in-domain. Formally, we perform this two-step procedure:

$$z = \arg \max_{y \in (ind, ood)} p_\phi(y|x)$$

$$i = \begin{cases} \arg\max_{y \in I} p_\kappa(y|x) & \text{if} \quad z = ind \\ ood & \text{if} \quad z = ood \end{cases},$$

where in the first equation $p_\phi(y|x)$ is the conditional probability of user message x being in in-domain class *ind* or out-of-domain class *ood*. The second equation determines the intent i using conditional probability $p_\kappa(y|x)$ of user message x having an intent I if the result of the first binary classification is *ind*. It classifies user message x as *ood* otherwise. This method shares advantages and disadvantages with the first method. The only difference is slightly higher complexity stemming from a two-step procedure.

The third method uses a threshold on the classifier's output probability distribution. In this approach, we train the classifier to recognize intent classes. Next, we specify the threshold on the confidence of the most probable class that the intent classifier outputs. If the confidence is below the threshold, we classify the input message as out-of-domain. Formally, we apply the decision process:

$$i = \begin{cases} \arg\max_y p_{IC}(y|x) & \text{if} \quad \max_y p_{IC}(y|x) > \lambda \\ ood & \text{if} \quad \max_y p_{IC}(y|x) \leq \lambda \end{cases},$$

where λ is the threshold. We use a validation set to select the best threshold based on the requirements of the final application. Applying a threshold to out-of-domain recognition does not need training OOD data. The OOD data are needed only as a part of a validation set for selecting the optimal threshold.

We present more information regarding out-of-domain recognition in Chapter 3, which proposes a novel approach for generating OOD data from in-domain intent training examples.

Once the system recognizes an out-of-domain input message, it has to handle it by producing a response. We can select different strategies for goal-oriented and open-domain dialogue systems. Recognizing the OOD input in goal-oriented dialogue systems is much more critical. The reason is that misclassified out-of-domain messages can cause the system to take unexpected action and fail. System failure would be especially problematic in critical applications like voice banking. However, the system does not need to produce specific answer. A simple fallback response like *"Sorry, I can't do that."* or *"I don't understand you."* usually works well. Despite this fact, more advanced response generation is possible.

The situation in the open-domain is reversed. As such voice applications are intended to conduct a conversation in a non-critical fashion, individual and rare errors of out-of-domain recognition are not considered futile for the whole conversation. However, the simple handling response can be applied only a few times before users notice the repetitiveness. Consequently, users rate the conversational experience poorly. Our proposed solution to this problem is Hybrid dialogue management, which uses a neural response generator with a Pragmatic level of control. We describe the solution in detail in Chapter 6.

2.11 Entity Recognition

Entity recognition is a task of information extraction. Its goal is to extract entities from a given text and classify them into predefined categories. *Person*, *Location*, or *Organization* can be examples of entity categories. More specifically, entities that refer to physical objects are called Named Entities. However, entities generally do not have to refer to physical objects. They can also refer to activities, concepts, and temporal or numerical expressions.

There were different methods proposed for entity recognition. A classical approach is to use linguistic grammar-based techniques. Such techniques usually achieve good precision. However, those techniques lead to low recall because those methods rely on extensive engineering of linguistics rules. The state-of-the-art approaches utilize neural network models. Open source NLP frameworks that support NER include Stanford CoreNLP [Manning et al., 2014], which implements linear-chain conditional random fields. SpaCy [Honnibal and Montani, 2017] implements a combination of convolutional neural networks and long short-term memory neural networks. Lastly, there is Flair [Akbik et al., 2019], which uses a bidirectional-LSTM neural network. BERT [Devlin et al., 2018] architectures based on Transformers [Vaswani et al., 2017] have gained popularity over the last year despite high requirements on computational resources. The disadvantage of these approaches is the necessity to collect a large number of annotated data.

The classical benchmark dataset for entity recognition is CoNNL-2003 [Sang and De Meulder, 2003]. The CoNNL dataset distinguishes four types of named entities: persons, organizations, locations, and miscellaneous names. The named entities are annotated using the IOB format, which assumes that entities are nonrecursive and non-overlapping. IOB format consists of three types of tags. Tags I-ENTITY mark words that are named entities. Tags B-ENTITY indicate words that are named entities but are not part of the previous entity. B-ENTITY tags are used only if the previous token was I-ENTITY of the same type. Thus, the B-ENTITY tag works as a boundary between two entities of the same type. Moreover, the O tag indicates that a word is not a part of an entity.

Metrics measuring the performance of entity recognition systems are precision, recall, and F1 score. Precision measures how many selected items are relevant. Recall measures how many relevant items are selected. The F1 score is the harmonic mean between Precision and Recall. However, there are problems with the calculation of these metrics in the case of entity recognition. The main problem arises for partially correct entities. Such entities are not a complete failure nor a total success. Examples can be an entity with more tokens selected than desired, fewer tokens selected than desired, or an entity with a wrong classification label. One approach is to measure a fraction of tokens which were classified correctly. However, this approach makes a system's performance seem disproportionately high because most tokens are not part of the entity. Thus, the system predicting *not an entity* for all tokens achieves an accuracy of more than 90% under usual conditions. Also, such a measurement does not penalize the wrong span of an entity. If the entity is a name and the system selects only the first name without a family name, the system achieves 50% accuracy.

Sang and De Meulder [2003] describe CoNNL benchmark metrics as follows: Precision is the number of predicted entity name spans that line up precisely with spans in the gold standard evaluation data. The recall is the number of names in the gold standard that appear simultaneously in the predictions. Moreover, the F1 score is the harmonic mean of these two. It means that if the model makes a classification error on a single token of an entity, then the whole entity is considered an error. Thus, this measure is pessimistic, given that many errors are close to correct prediction and might be enough for a given application. For these reasons, examining the selected system's errors and deciding how important they are for application is beneficial.

The entity recognition helps conversational AI system to recognize entities users are mentioning. Dialogue manager can request additional information about recognized entities, which can be presented to the user as part of the response. Moreover, it can change the state of the dialogue according to it. Lastly, it can store the recognized entities in the corresponding user profile fields. We also propose a summarization method which utilizes named entities in Chapter 4.

2.12 Dialogue Management

Dialogue management is a task conducted by the dialogue manager. Dialogue manager is a central component of conversational AI, which we can imagine as the brain of the whole system. It selects the optimal actions of the whole system. Dialogue manager is the only stateful component of conversational AI. The dialogue manager builds and maintains the state of the conversation based on the input of the user processed by NLU components. It selects the next action according to the state.

We can divide actions into several categories. Action most commonly represents a response generation. Action can also represent a query to the knowledge base, thanks to which the dialogue manager obtains the information it needs for the following action decision or response generation. Finally, the action can also represent an executable code.

The decision process of dialogue management can be divided into two layers. The first is topic tracking, and the second is flow control. At the topic tracking level, the dialogue manager keeps track of the recent topic of the dialogue, updates it based on the user's messages and changes it according to its strategy. Topic tracking is a high-level decision process that helps dialogue manager select appropriate conversation areas. The flow control follows. In most cases, the dialogues which the conversational system can conduct with users are represented as state space with transitions between states. We call the states with transitions the dialogue flow. Thus, in flow control, the goal of the dialogue manager is to keep track of and update the recent state. We can implement both flow control and topic tracking as a rule-based system, as well as machine learning-based systems. We propose topic tracking and flow control methods in Chapter 5.

The dialogue manager is also responsible for handling problematic states of the dialogue. There can be silence in voice-based conversational AI, automatic speech recognition might transcribe voice into text with low confidence, computation of any other component of conversational AI can end in error, or the out-of-domain detection recognizes unexpected user messages into the system. The dialogue manager has to have tactics for those events. We propose a method for out-of-domain recognition in Chapter 3 and subsequent handling in Chapter 6.

In many systems, the response generation depends on the state of the dialogue. Because the state of the dialogue is represented by the dialogue manager, the dialogue manager participates in response generation. Thus, the border between response generation and dialogue manager is not strictly defined in all systems. In our proposed work, we understand response generation as part of dialogue management.

2.13 Summarization

Automatic text summarization is an essential task of natural language processing. The goal is to describe a text accurately, be it a news article, a web page, or a paragraph of a book, using shorter text. The shorter text can be a paragraph, sentence, or even a few words. Automatic text summarization is a challenging problem for automatic systems because they must excel in multiple areas simultaneously. They have to understand the original text's meaning, understand which passages are important and which can be excluded, and generate meaningful and grammatically correct summarizations.

We generally divide text summarization algorithms into two categories, *extractive* and *abstractive*. Extractive summarization algorithms choose pieces from the original text, usually sentences, and combine them to form a summary. From a high-level perspective, most extractive summarizers follow the same two steps: Score all sentences first. Pick N sentences with the highest score next. The main difference between individual extractive methods is how they score sentences. The advantage of extractive methods is that no matter how simple the method is, it always produces syntactically correct sentences, even though they may not be useful summaries. On the other hand, there is a disadvantage too. Extractive summarizers are limited in what they can predict by the sentences of the source text. Thus, more elaborate summaries are out of their reach.

Abstractive summarizers generate summaries consisting of novel sentences not part of the original text. Abstractive summarization algorithms are usually more complex because they have to understand the input text, find the most relevant passages, and generate syntactically correct sentences as summarization. Such a task is nearly impossible for hand-written rules. However, recent machine learning advances, particularly neural networks, make abstractive summarization possible. Moreover, neural networks represent the current state-of-the-art in abstractive summarization.

Summarization is a well-suited method for incorporating knowledge from longer texts into conversation. The algorithms can shorten a long text into a passage, which can be better utilized in conversation. Shorter texts that grasp the original article's main message are generally better to keep the user engaged. Moreover, summarizations are more advantageous than whole articles for the limited input size of large language models. Summarization can serve to ground text generated by a large language model in knowledge. Also, summarization generally does not make any assumption on the article's structure. Thus, even when the headline or initial passages are not helpful, the summarization can extract or construct sentences representing the meaning of a whole article. We propose a method of summarization based on named entity recognition in Chapter 4. Furthermore, we utilize the summarizations of articles in Hybrid dialogue management, which we propose in Chapter 6.

2.14 Language Models

The language model is a model that assigns probabilities to sequences of words. [Jurafsky and Martin, 2022] Language models are helpful in many problems of conversational AI, like automatic speech recognition, machine translation or information retrieval. The language model's ability to generate text is the most important for our work. While a language model is not considered to be a traditional component of conversational AI systems, we present them for their usefulness and their crucial role in the work we propose in Hybrid dialogue management described in Chapter 6.

The language model can be realized using n-grams or neural networks. Neural network-based language models utilize feed-forward neural networks, original recurrent neural networks, long-short-term-memory neural networks or transformer-based architectures of neural networks. We use text corpora to learn the parameters of the language model.

2.14.1 N-gram Language Models

The n-gram language model assumes that a word's probability in a sequence can be determined using a limited size window of previous words. [Markov, 1913, Shannon, 1948] We use the window size to name the n-gram model. If we use a window of one word, we call the model bigram model. If we use a window of two words, we call the model trigram model. We call models using longer windows n-gram models. The general rule is that the n-gram model uses a window size of $n-1$ words. The larger the context we use, the better the language model represents the corpora we used for training.

The n-gram model calculates the joint probability of a sequence of words as a product of conditional probabilities of words in a sequence given their window of previous words. In other words, the n-gram model uses the chain rule of probability. Formally, the n-gram model calculates the probability $P(w_1, w_2, ..., w_n)$ of sequence of words $w_1, w_2, ..., w_n$ using the formula

$$P(w_1, w_2, ..., w_k) = \prod_{i=1}^{k} P(w_k | w_{k-(n-1)}, ..., w_{k-1}).$$

We can calculate the conditional probability of a word given a window of previous words using the formula

$$P(w_k | w_{k-(n-1)}, ..., w_{k-1}) = \frac{|w_{k-(n-1)}, ..., w_{k-1}, w_k|}{|w_{k-(n-1)}, ..., w_{k-1}|},$$

where $|w_{k-(n-1)}, ..., w_{k-1}, w_k|$ denotes the number of times we observed a sequence of words $w_{k-(n-1)}, ..., w_{k-1}, w_k$ in the training corpora.

With the growing size of n-grams, the problem of never-observed sequences of words becomes more severe. Thus, we use smoothing strategies to mitigate the problem. One such strategy is adding 1 to calculated occurrences of all n-grams. This strategy is called the add-one strategy or Laplace strategy. [Jurafsky and Martin, 2022] Instead of 1 we can also add constant k which is between 0 and 1. This method is called add-k smoothing. The last smoothing methods we will mention are the backoff and interpolation. Both methods use a combination of several n-gram models to calculate the probability of word sequence.

The backoff method works with the assumption that there are cases in which it is better to use less information. The backoff method uses the n-gram model if it contains a probability of a word given a context window. If n-gram does not contain the probability, it backoffs to using the probability of (n-1)-gram. If there is no probability in (n-1)-gram, it can go down to (n-2)-gram and possibly down to the uni-gram model.

The interpolation method uses a weighted mix of several n-gram models. To illustrate the interpolation method, we show how to estimate the probability of the tri-gram model $P(w_k | w_{k-2}, w_{k-1})$. We use the formula

$$P(w_k | w_{k-2}, w_{k-1}) \approx \lambda_1 P(w_k) + \lambda_2 P(w_k | w_{k-1}) + \lambda_3 P(w_k | w_{k-2}, w_{k-1}).$$

Moreover, the λ_1, λ_2, and λ_3 must sum to 1

$$\lambda_1 + \lambda_2 + \lambda_3 = 1.$$

Both in the add-k smoothing and interpolation, there is the problem of selecting the values of k and λ_n parameters. We select the best-performing parameters by evaluating the model using the development corpora. We use perplexity to measure the quality of the language model on the corpora.

Generating sentences out of language model is crucial in question answering, machine translation or response generation. To generate a sentence from the n-gram language model, we first select the initial n-gram based on its probability. If we select the most probable n-gram, we call this method greedy. We call the method sampling if we select the n-gram based on its probability, with more probable n-grams having a higher chance of selection than the less probable n-grams. Next, we use the selected n-gram and use it as a context word for selecting the next word using greedy selection or sampling. We iterate this process moving forward with the context window at each step until we select a special word representing the end of the generation.

The problem with n-gram language models is that with the larger n, the number of parameters of a model increases exponentially. Also, n-gram language models cannot transfer knowledge learned on the training set to the testing set. For those reasons, approaches to language modelling utilizing neural networks were proposed.

2.14.2 Feed-forward Neural Network Language Models

Bengio et al. [2000] proposed using neural networks as language models. The method utilized a feed-forward neural network. The feed-forward neural network approximated the probability of the word in the sentence using the window of several previous words. This approach is similar to n-gram models except that words are encoded using embedding into continuous space. The main advantage of embedding is that words with similar semantics share similar features in the embedding space. This fact allows the neural network-based language model to generalize knowledge from the training set to the unobserved sequences of the testing set.

We train the neural network using the back-propagation algorithm by minimizing the cross-entropy loss. During the prediction of the probability of the next word using the trained language model, we first encode the words out of the window as embedding vectors

$$e_n = Ex_n,$$

where x_n is a one-hot encoding of context word, E is embedding matrix and e_n is embedding vector of context word. Next, we combine the embedding vectors. Two popular approaches are to concatenate and average the embedding vectors

$$e = e_{n-2} \oplus e_{n-1} \oplus e_n,$$

where $\oplus$ represents average or concatenation operation and e is resulting embedding representation of window of context words. We apply a fully connected layer of neural network to the embedding vector. The fully connected layer consists of matrix W, bias vector b, and non-linear function σ,

$$h = \sigma(We + b),$$

where h is a resulting vector. Popular choices of non-linear functions include sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$, ReLU function $ReLU(x) = \max(0, x)$ or hyperbolic tangent $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. We apply the next fully connected layer with its matrix W_1 and bias vector b_1 in the following step. We apply several fully connected layers in a row. Finally, the last fully connected layer of the neural network uses the softmax function instead of the non-linear function σ

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{K} e^{x_j}},$$

$$y = \text{softmax}(W_i h + b_i),$$

where y is a vector whose dimension is equal to the size of the vocabulary of the language model. Each element of vector represents probability $P(w_t = j|w_{t-3}, w_{t-2}, w_{t-1})$ of jth word in vocabulary appearing after the words $w_{t-3}, w_{t-2}, w_{t-1}$ present in context window.

The generation in the case of neural networks works analogously to n-gram models. First, we generate the first word of a sentence by passing the special start of sentence words as input to the feed-forward neural network. Again, we can use the greedy method or sampling to select a word out of the probability distribution produced by the softmax function. Next, we use the generated word as context for generating the next word. We iterate the process while moving the context window until the model generates a special end-of-sentence word.

Neural networks are better in language modelling than n-gram models because they generalize better. Their better generalization is achieved thanks to embedding and the smaller number of parameters. However, the explainability of neural networks is significantly smaller than that of n-gram models. Also, like n-gram models, neural networks approximate the probability of a word in a sentence using only a limited window of previous words. Thus, neural networks can not model longer dependencies between words. For this, we have to introduce recurrent neural networks.

2.14.3 Recurrent Neural Network Language Models

Recurrent neural networks [Rumelhart et al., 1985] are designed to represent a prior context of an arbitrary number of words. The recurrent neural network uses recurrent connections, allowing prediction to be conditioned by all previous predictions directly or indirectly. While the rationale behind a specific prediction of recurrent neural networks is even harder to explain than in the case of feed-forward neural network-based language models, it was shown that recurrent neural networks are very effective in language modelling. [Mikolov et al., 2010]

N-gram models and language models based on feed-forward neural networks approximate the probability of a word in a sentence using the probability of a word given the context of several preceding words. A recurrent neural network models the probability of a word in a sentence using all previous words. Formally, while n-gram and neural network-based models approximate the probability of a word using $P(w_t|w_{t-(n-1)}, ..., w_{t-1})$, the recurrent neural networks model the probability $P(w_t|w_{t-1}, ..., w_0)$, where $w_{t-1}, ..., w_0$ are all words in front of word w_t. Moreover, we can calculate the probability of a whole sentence as

$$P(w_1, ...w_n) = \prod_{i=1}^{n} P(w_i|w_{i-1}, ..., w_0).$$

The recurrent neural network works similarly to a feed-forward neural network. First, we create an embedding vector for input word w_t

$$e_t = Ew_t,$$

where E is embedding matrix and e_t is resulting embedding vector. Next, we calculate the hidden state h_t

$$h_t = \sigma((We_t + b) + (Uh_{t-1} + c)),$$

where h_{t-1} is hidden state calculated for word w_{t-1}, W and V are matrixes, b and c are bias vectors and σ is a non-linear function. We use a hidden state calculated for the preceding word, as the formula shows. A recurrent neural network uses hidden states h_t to store information about the sequence. It can use the information in calculations for subsequent words. Finally, we pass the hidden state h_t through the output layer of the recurrent neural network with a softmax function

$$y_t = \text{softmax}(Vh_t + d),$$

where V is a matrix, d is bias vector and y_t is a vector in which element j corresponds to the probability of jth word in the vocabulary of model. We repeat those calculations until we reach the end of the sequence while storing the hidden states for calculating the following output.

We can also use stacked recurrent neural networks. Stacked recurrent neural networks use multiple layers of recurrent neural networks, in which the output of one recurrent neural network is passed as input to the next layer of the recurrent neural network. Theoretically, the more layers we use, the better the language model we can train using the backpropagation algorithm. The rationale behind this phenomenon is that the first layer extracts the basic representation and patterns of a sentence, and the following layers create more advanced abstractions and patterns. This phenomenon allows the recurrent neural network to more precisely model the probability of a word following in a sentence. However, the number of layers working best in practice depends on our training corpora. Also, the cost of training increases with the growing number of layers because we introduce more parameters. Thus, more stacked layers do not automatically make the language model better.

In practice, the recurrent neural networks face two problems. First, hidden states have to encode two sets of information - information that is carried to the calculation for the following words and information for the current output. This fact makes it a more complex task for parameters of recurrent neural network. The second problem is error propagation during training using a backpropagation algorithm. This phenomenon is called the vanishing gradient problem and causes recurrent neural networks to be notoriously hard to train. Those two problems cause recurrent neural networks to encapsulate only local relationships between words in sequence while mostly missing the relationships between words further apart. Thus, in practice, recurrent neural networks are rarely used in their most basic form. Instead, Hochreiter and Schmidhuber [1997] proposed recurrent neural networks using Long short-term memory units, shortly called LSTM. LSTM addresses two limiting properties of basic recurrent neural networks.

2.14.4 LSTM Language Models

There are slight differences in the architecture of LSTM. We will present the standard LSTM. The LSMT uses context vector C_t, sometimes called cell state vector, to carry information along the sequence that the LSTM processes. The context does not pass through any neural network layer during calculations of the following context vector c_{t+1}. Thus, it avoids the vanishing gradient problem. Otherwise, LSMT uses the input x_t and hidden state h_t to calculate the next hidden state h_{t+1} as the recurrent neural network does.

LSTM uses so-called gates to store, retrieve and delete information out of context. The gate allows us to modify the content of the vector thanks to the sigmoid function, which outputs numbers between 0 and 1. Thus, the gate can specify how much information it wants to remove from or add to a vector.

The first gate of LSTM is the forget gate. The forget gate removes information out of the previous context based on the previous hidden state and recent input. Formally, the forget gate calculates the vector

$$f_t = \sigma(W_f(h_t \oplus x_t) + b_f),$$

where $\oplus$ represents concatenation of vectors. The following calculations will apply the forget gate to the context C_t.

The second gate is the add gate. The add gate adds new information into context. The process works in two steps. First, we calculate a new context vector C'

$$C' = \tanh(W_{C'}(h_t \oplus x_t) + b_{C'}).$$

Next, we apply a gate to the vector C', which selects which parts of the new context vector we want to add to the previous context vector

$$a_t = \sigma(W_a(h_t \oplus x_t) + b_a) \odot C'.$$

We apply vectors f_t and a_t produced by forget and add gates to context C_t

$$C_{t+1} = f_t \odot C_t + a_t,$$

where C_{t+1} is the new context vector.

Finally, we calculate a new hidden state h_{t+1} using the new context vector C_{t+1} and output gate based on previous hidden state h_t and recent input x_t. Formally, we use the formula

$$o_t = \sigma(W_o(h_t \oplus x_t) + b_o)$$

to calculate the output gate. We calculate a new hidden state by applying the $tanh$ function to the new context vector C_{t+1} and the result of the output gate. Formally, we calculate

$$h_{t+1} = o_t \odot \tanh(C_{t+1}).$$

In order to use LSTM as a language model, input vector x_t represents the embedding vector of tth word in a sentence, and we pass hidden state h_{t+1} through linear layer followed by softmax function to calculate probabilities of next word out of language model's vocabulary

$$y_t = softmax(Wh_{t+1} + b).$$

A simpler alternative to LSTM is called Gated Recurrent Unit or GRU [Cho et al., 2014]. Similarities between those two architectures of recurrent neural networks are the usage of gates and direct connection between subsequent recurrent units. However, instead of separate hidden state h_t and context c_t vectors of LSTM, GRU uses only hidden state h_t that serves as a direct connection. In practical applications, there are only minor differences in performance between LSTM and GRU.

However, neither LSTM nor GRU does not solve the most limiting factor of recurrent architectures. That is the problem that subsequent calculations have to be performed in sequence. Thus, the recurrent neural networks are hard to scale. This fact limits their ability to learn complex relationships of language modelling. For this reason, the Transformer architecture was proposed.

2.14.5 Transformer Language Models

Transformer architecture [Vaswani et al., 2017] builds on top of two core concepts: multi-headed self-attention and layer normalization. Both innovations allow us to radically scale the number of parameters and achieve new state-of-the-art results on many problems, including language modelling.

The multi-headed self-attention consists of several self-attention layers. Self-attention layer takes as an input sequence of vectors $x_1, x_2, ..., x_n$ and produces the sequence of vectors $y_1, y_2, ..., y_n$. The length of input and output sequences and the dimensions of elements of sequences are the same to allow for the stacking of layers. The usual sequence length equals 1024 or 2048, with the largest models reaching sizes of 4096 or 8192. Those sizes represent the text length in terms of tokens, which we can insert into the model. In many cases, Transformers models do not use word tokenization but use Byte-Pair encoding instead. Thus, in practice, transformer-based models are limited to processing a single paragraph or short article in one forward pass.

First, the self-attention layer calculates each input element's query, key, and value vectors. We can perform those calculations in parallel thanks to matrix operations

$$Q = W_q X$$

$$K = W_k X$$

$$V = W_v X,$$

where X is a matrix consisting of inputs, W_q, W_k, and W_v are matrices transforming the inputs into matrices consisting of query vectors Q, key vectors K, and value vectors V. Finally, we calculate the value of self-attention by calculating the dot product between the query matrix and key matrix, which we normalize, apply the softmax function, and multiply it with the value matrix. Formally, we calculate

$$sa(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d^q}})V,$$

where $sa(Q, K, V)$ is self-attention and $\sqrt{d^q}$ is the square root of the dimension of the query vector used for normalization. However, if we only multiply query matrix Q with transposed key matrix K, we would include the information about the following words in calculations of preceding words. Thus, the resulting model would be useless for predicting the next word in the sequence given the preceding words. For this reason, we have to set the upper triangle of matrix multiplication product QK^T to $-\infty$. Formally, we set the element of the matrix according to

$$q_{ij} = \begin{cases} q_{ij} & \text{if} \quad i \geq j \\ -\infty & \text{if} \quad i < j \end{cases},$$

where q_{ij} is element in matrix at position (i, j).

Multi-headed attention uses multiple self-attentions, which we call heads. Each head has its own matrices W_q^k, W_k^k, and W_v^k, which each produces query, key and value matrices Q^k, K^k, and V^k. The resulting vectors of each self-attention are concatenated into a single vector. We apply a linear transformation to convert the dimension back to the original dimension of input vectors. Formally, we first calculate

$$Q^k = W_q^k X$$

$$K^k = W_k^k X$$

$$V^k = W_v^k X.$$

Next, for each k we calculate $sa(Q^k, K^k, V^k)$. Finally, we calculate multi-headed self-attention by formula

$$mhsa(X) = W_o(sa(Q^1, K^1, V^1) \oplus sa(Q^2, K^2, V^2) \oplus ... \oplus sa(Q^K, K^K, V^K)),$$

where $\oplus$ represents the concatenation operation applied to resulting vectors of individual self-attentions and W_o is the linear transformation matrix. The linear transformation decreases the vector's dimension back to the dimension of single self-attention. Thanks to the reduction of dimensionality, we can stack layers of multi-head self-attention.

The transformer architecture makes use of layer normalization. The purpose of layer normalization is to improve the training performance of the model. The layer normalization moves the values of hidden layers to a reasonable range for the back-propagation algorithm based on gradients. First, we calculate mean μ and standard variation σ of elements in vector X

$$\mu(X) = \frac{1}{d_x} \sum_{i=0}^{d_x} x_i$$

$$\sigma(X) = \sqrt{\frac{1}{d_x} \sum_{i=0}^{d_x} (x_i - \mu)^2},$$

where d_x is dimension of vector X.

We normalize the vector by using the formula

$$X' = \frac{X - \mu(x)}{\sigma(X)}.$$

Finally, we apply gain σ and bias β, which are learnable parameters

$$layNorm(X) = \sigma X' + \beta$$

where $layNorm(X)$ denotes the layer normalization.

We combine multi-headed self-attention and layer normalization into a transformer block. Transformer models consist of multiple transformer blocks. The transformer block first applies multi-headed self-attention to the input matrix X. We apply layer normalization to the result of multi-headed self-attention combined with residual connection. Next, we apply a linear layer. Finally, we apply the second layer normalization to the result of a linear layer combined with a residual connection that produces final Y. Formally, a transformer block can be described as

$$Z = layNorm(mhsa(X) + X)$$

$$Y = layNorm(ZW + b + Z).$$

The last important concept of transformer architecture is positional encoding. Due to the nature of calculations which transformer blocks perform, the position of words in the input sequence is lost. For this reason, we add positional encoding to input embeddings that we pass to the model. Thus, the model knows at which position of the sentence the word is. Positional encoding is created as combination of sin and cos functions

$$PE(pos, 2i) = \sin(\frac{pos}{10000^{\frac{2i}{d}}})$$
$$PE(pos, 2i + 1) = \cos(\frac{pos}{10000^{\frac{2i}{d}}}),$$

where pos is position of word in the input sequence, i is dimension of positional encoding vector PE, and d is dimension of input embedding. Finally, we sum input embedding with positional encoding

$$X = X_{emb} + PE.$$

We pass the resulting matrix X into the transformer model.

The transformer model consists of multiple stacked transformer blocks. The final model includes a feed-forward neural layer with a softmax function, which assigns probabilities to individual tokens of vocabulary. The transformer language model is trained on large corpora like Common Crawl using the backpropagation algorithm, minimizing the cross-entropy loss. In this case, minimizing the cross entropy loss is equivalent to minimizing the negative log probability of the correct word.

During the inference of the transformer-based language model, we insert the sequence of words and generate the following word. Formally, we select the next word with the maximal probability according to the formula

$$w_{t+1} = argmax_{w \in V} P(w|w_1, w_2, ...w_t).$$

Nevertheless, there are other strategies for predicting the next word based on the output probabilities of any model we have described. If we use the most probable word, we call this method greedy. However, there is a problem with this approach. Although the model selects the most probable word in each step, the whole sequence's probability can be suboptimal. In other words, the method can get stuck in local minima. An alternative possibility is to use sampling.

In the sampling, we select the next word based on its probability. We select the more probable words more often in proportion to their probability and probabilities of other words. This approach leads to the generation of more diverse sentences. We can also use alternatives like top-k or top-p sampling. In top-k sampling, we sample only out of the k most probable words, and the probability mass of all words is redistributed only to the k most probable words. Alternatively, the top-p sampling samples only out of the smallest possible set of words whose cumulative probability reaches the probability p. Both top-k and top-p sampling address the problem of ordinary sampling: the unlikely words can be sampled with non-zero probability.

Another decoding method is called beam search. The beam search is a balance between two problems. A greedy search considers only the most probable word, which might not lead to the most probable sentence. However, to find the most probable sentence, we must search the whole search tree representing all possible generated sentences. Search in the whole search tree would be too complex. Beam search considers the k most probable sentence hypothesis. The parameter k is called beam size. It balances greedy and exhaustive search. With k equal to 1, beam search equals greedy search. With k equal to ∞, beam search becomes an exhaustive search. In each decoding step, we take the k most probable sequences, for which we select the next word. We sort the probabilities of sequences and select only the k most probable sequences for the next step of beam search until the model finishes the generation of k sequences. We call the generated sequences hypothesis. The probability of sequence of words $x_1, ..., x_n$ is defined by chain rule of probabilities

$$P(x_1, ..., x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)...P(x_n|x_1, x_2, ..., x_{n-1}).$$

Probabilities in the formula above can cause numerical instabilities due to the multiplication of small numbers. Thus, we can alternatively score the sequence by the sum of logarithms

$$\begin{aligned} score(x_1, ..., x_n) = {} & log(P(x_1)) + log(P(x_2|x_1)) \\ & + log(P(x_3|x_1, x_2)) + ... + log(P(x_n|x_1, x_2, ..., x_{n-1})). \end{aligned}$$

Moreover, we can add length normalization because models tend to score longer sequences as less probable. Thus, we can define the score of sequence as

$$\begin{aligned} score(x_1, ..., x_n) = {} & \frac{1}{n}(log(P(x_1)) + log(P(x_2|x_1)) \\ & + log(P(x_3|x_1, x_2)) + ... + log(P(x_n|x_1, x_2, ..., x_{n-1}))), \end{aligned}$$

where n is the length of the sequence.

Beam search is one of the most popular decoding strategies of language models. Typical beam size is between 5 and 10. However, we also consider smaller beam sizes like 3 in real-world applications of conversational AI. The reason for the smaller size is that the language model produces the response faster with the smaller beam size. Faster reaction time leads to more fluent conversation.

Moreover, we can also combine the beam search with sampling and its derivatives in practice. Instead of selecting the most probable word as a continuation of the sequence, we sample according to the probability of words. Thus, the decoding strategy and its hyperparameters are one of the properties of the system that we have to select. We must select them according to our application because each decoding strategy produces slightly different outputs.

2.14.6 Comparison of Language Models

We present a general comparison of language models in Table 2.4 for an illustration. The Table 2.4 compares perplexity on the Penn Treebank dataset testing set [Marcus et al., 1994]. Compared models are the n-gram language model, feed-forward neural network

Language Model	**Perplexity**
Trigram	367.79 [Takahashi and Tanaka-Ishii, 2019]
FF NN	140.20 [Mikolov and Zweig, 2012]
RNN	114.50 [Bai et al., 2018]
GRU	92.48 [Bai et al., 2018]
LSTM	78.93 [Bai et al., 2018]
Transformer-XL	54.52 [Dai et al., 2019]
GPT-2	35.76 [Radford et al., 2019]
GPT-3	**20.50** [Brown et al., 2020]

Table 2.4: Comparison of perplexity achieved by language models on a testing set of Penn Treebank dataset [Marcus et al., 1994]

language model, recurrent neural network language model, GRU language model, LSTM language model and three variants of Transformer language models. The three variants of Transformer language models namely are Transformer-XL [Dai et al., 2019], GPT-2 [Radford et al., 2019] and GPT-3 [Brown et al., 2020]. We can observe that the perplexity of subsequent methods decreases with increasing complexity of the method. The state-of-the-art performance is achieved by GPT-3.

2.15 Evaluation of Conversational AI

Evaluating conversational AI is necessary for improving the whole research field, yet it is a complicated discipline. We can divide the evaluation methods into upstream evaluation and downstream evaluation.

The downstream evaluation methods measure the quality of individual components and sub-tasks of conversational AI. Downstream evaluation methods are possible thanks to the fact that conversational AI systems consist of multiple components. We are talking about the accuracy of intent recognition, F1 score of named entity recognition or BLEU [Papineni et al., 2002] and ROUGE [Lin, 2004] scores of summarization and machine translation. Those evaluation methods are generally automatic, repeatable and explainable. Thus, they are relatively cheap and reliable approaches to evaluate individual components. Upstream evaluation seems perfect for evaluating conversational AI because optimizing those metrics for individual components should improve a whole system's performance in theory. However, this is not guaranteed in practice. The upstream evaluation does not tell us anything about the system's performance. Error in one component can have a ripple effect on the rest of the components. Moreover, the conversational system might employ a wrong conversational strategy, leading to poor user experience, which would not be observable through upstream evaluation. In conclusion, upstream evaluation is helpful during the development of individual components, but we have to use downstream evaluation metrics to evaluate the quality of a system as a whole.

The downstream evaluation measures the quality of a conversation the system is capable of conducting. The proper downstream evaluation metric is automatic, repeatable, correlated to human judges, explainable and differentiates between dialogue systems. [Deriu et al., 2021] The exact metric is different for each application of conversational AI and its goal. The main problem with downstream evaluation is that an automated evaluation procedure that satisfies the above mentioned requirements has yet to be developed. [Deriu et al., 2021] Thus, we have to use human evaluation for downstream metrics.

Human evaluation of downstream metric is typically conducted by lab experiments, in-field experiments or crowdsourcing. While human evaluation is the only option for downstream evaluation, it is problematic for several reasons. The most pressing issue is that collecting those metrics is time-consuming and costly because humans have to perform the evaluation. Next, most downstream metrics usually rely on rating some desired property of conversation, like answering *appropriately* or having a *good* conversation. The problem with those specifications is that such properties are poorly defined. Moreover, those qualities are highly subjective. This fact leads to low annotation agreement.

Users are asked to rate or annotate some part of a conversation for any downstream metrics. There is a trade-off between the frequency of asking for feedback and the usefulness of the rating. On one extreme, the metric can be designed to ask for feedback only at the end of the conversation. This requires minimal human labour but provides information only for the conversation as a whole. Thus, information about the quality of individual parts of the conversation is lost. On the other extreme is asking for feedback after each conversational turn. This approach provides complete information about the dialogue but is labour-intensive and might unnecessarily interrupt the flow of the conversation.

What is more, there is also a question of how much metrics to use and what is their scale. We can ask for several metrics at once, rating how the conversation is engaging, how it is natural, or how knowledgeable the system appears, but again, we can overwhelm the human annotator and the cost of evaluation increases. Also, it is crucial to select an appropriate scale of the metric. The metrics scale can be simply binary or be more fine-grained and provide more distinction. However, more options make correct and objective rating difficult for human evaluators. This is because differences between individual levels tend to blur with more fine-grained scales. In conclusion, selecting the proper downstream metric is a complicated task, and the final choice depends on the application of the conversational system and resources we have at our disposal. Nevertheless, downstream metrics are the only approach to evaluate conversational AI systems appropriately and provide us with the most valuable feedback.

Chapter 3

Generative Adversarial Network for Out-of-Domain Data Generation

The section proposes a Generative Adversarial Network for Out-of-Domain Data Generation. [Marek et al., 2021b] The proposed approach improves the recognition of out-of-domain inputs by generating out-of-domain synthetic training examples using in-domain training examples for intent recognition. Out-of-domain recognition produces an important signal for dialogue manager. The signal means the input is unexpected and can not be classified into one of the conversational AI's intent classes. Consequently, the dialogue flow is not prepared to handle an unexpected input, and processing the out-of-domain input might lead the dialogue manager to select erroneous action. Thus, an alternative method of handling the unexpected input has to be employed, like Hybrid dialogue management.

The proposed method is closely connected to Hybrid dialogue management that we propose in Chapter 6. Moreover, the proposed method is more computationally efficient regarding the number of trainable parameters than other baseline methods. This fact makes it suitable for real-world applications.

3.1 State of the art

We divide this section into two parts. The first part mentions state-of-the-art intent recognition because it is a closely connected problem to out-of-domain recognition. The second part mentions notable works in out-of-domain recognition on which we build our proposed method.

3.1.1 Intent Recognition

In intent recognition, Kim [2014] proposes convolutional neural networks trained on top of pre-trained word vectors for sentence-level classification tasks. They show that a simple CNN with little hyperparameter tuning and static vectors achieves excellent results on multiple benchmarks. Learning task-specific vectors through fine-tuning offers further gains in performance. They additionally propose a simple modification to the architecture to allow for the use of both task-specific and static vectors.

Zhao and Wu [2016] propose a convolutional neural network with an attention mechanism. They found out that it is not easy to encode long-term contextual information and correlation between non-consecutive words in a traditional convolutional neural network. They demonstrate that the proposed attention-based convolutional neural network can capture these kinds of information for each word without any external features.

Zhang et al. [2018a] propose a capsule-based neural network model that accomplishes slot filling and intent detection via a dynamic routing-by-agreement schema. They propose to model the hierarchical relationship among each word, the slot it belongs to, and the whole utterance's intent label by a hierarchical capsule neural network structure called CAPSULE-NLU. The proposed architecture consists of three types of capsules. WordCaps learns context-aware word representations. SlotCaps categorizes words by their slot types via dynamic routing and constructs a representation for each type of slot by aggregating words that belong to the slot. IntentCaps determines the intent label of the utterance based on the slot representation and the utterance contexts. Once IntentCaps have determined the intent label, the inferred utterance-level intent helps re-recognize slots from the utterance by a rerouting schema.

Niu et al. [2019] propose a bi-directional interrelated model for joint intent detection and slot filling. They introduce an SF-ID network to establish direct connections for the two tasks to promote each other mutually. They also designed an entirely new iteration mechanism inside the SF-ID network to enhance the bi-directional interrelated connections.

Obuchowski and Lew [2020] propose a novel approach to intent recognition, combining transformer architecture with capsule networks. They used the encoder part of transformer architecture combined with capsule networks and dynamic routing to construct an accurate embedding for the queried sentence.

3.1.2 Out-of-domain Recognition

In out-of-domain recognition, Larson et al. [2019] proposes three baseline approaches for predicting whether a query is out-of-domain. The first approach trains additional intent on a limited set of out-of-domain training data. The second uses a threshold on the classifier's probability estimate. Furthermore, the third uses a two-stage process where it first classifies a query as in- or out-of-domain, then classifies it into one of the intents if classified as in-domain.

Zheng et al. [2020] propose a model to generate high-quality pseudo-out-of-domain samples akin to in-domain input utterances, thereby improving the performance of out-of-domain detection. An autoencoder is trained to map an input utterance into a latent code. Moreover, the codes of in-domain and out-of-domain samples are trained to be indistinguishable by utilizing a generative adversarial network. An auxiliary classifier is introduced to regularize the generated out-of-domain samples to have indistinguishable intent labels to provide more supervision signals.

Lee and Shalyminov [2019] proposed the AE-HCN-CNN model, consisting of an autoencoder and a reconstruction score-aware HCN model. They also introduce a counterfeit data augmentation method for training the proposed model.

Ryu et al. [2018] propose to use only in–domain sentences to build a generative adversarial network in which the discriminator generates low scores for out-of-domain sentences. To improve basic GANs, they apply feature matching loss in the discriminator, use domain–category analysis as an additional task in the discriminator, and remove the generator's biases.

3.2 Method

The state-of-the-art intent recognition algorithms are trained using neural networks to produce probability distribution over output classes and use cross-entropy loss. However, Lakshminarayanan et al. [2017], and Guo et al. [2017] pointed out that the neural network classifier tends to be overconfident in its classification. This means that the classifier tends to assign a high probability for one class, even when the example was not seen in the training phase. Thus, such a classifier cannot correctly recognize if an example belongs to an in- or out-of-domain distribution during runtime with any reasonable threshold value. In this method, we focus on improving the performance of the out-of-domain detection method with the help of generated out-of-domain data.

Zheng et al. [2020] proposed to use negative entropy as an additional loss for the classification task in a neural network. The negative entropy loss trains the network to flatten the produced probability distribution as opposed to cross-entropy, which teaches the network to maximize the correct class probability. Thus, the idea is to apply cross-entropy loss on in-domain data and negative entropy loss on out-of-domain data. The result is that in-domain data receives a high probability for the correct class, and out-of-domain data receives low probabilities for all classes. Thanks to this fact, we can select a reasonable threshold on the output probability to classify in- and out-of-domain data correctly. We need out-of-domain data to train models in this way. However, the collection of out-of-domain data is a manual and expensive process.

The in-domain data forms a small distribution cluster in the space of vector text representation. In principle, the rest of that space is covered by out-of-domain data. Also, in real-world scenarios, most out-of-domain data share patterns with in-domain data. Nevertheless, Zheng et al. [2020] demonstrated that training intent recognition model with out-of-domain data just outside in-domain distribution should be sufficient to handle most out-of-domain inputs during runtime.

We propose a novel out-of-domain data generation model OodGAN, which is an extension of SeqGAN [Yu et al., 2017]. We use GAN to generate out-of-domain data that share the same patterns as in-domain data and are very close to in-domain data distribution. Unlike the previously proposed models, OodGAN works with a sequence of words directly. Previously proposed models work on latent space represented by auto-encoder. Our model eliminates the need for the auto-encoder, which reduces the model's overall size.

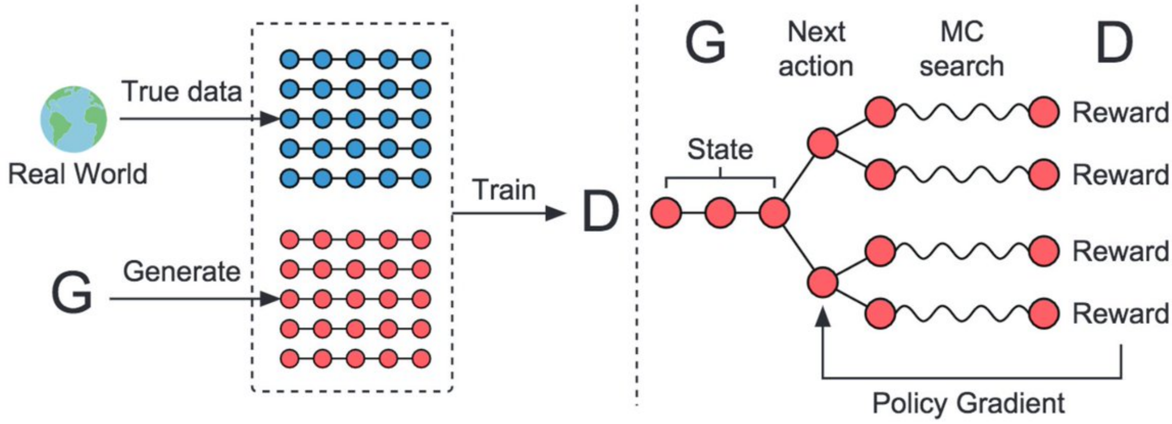


Figure 3.1: The illustration of SeqGAN [Yu et al., 2017]. Left: Discriminator D is trained over the real data and the data generated by generator G. Right: The generator is trained by policy gradient, where the discriminator provides the final reward signal and is passed back to the intermediate action value via Monte Carlo search.

Dialogue designers can define any number of in-domain intents in dialogues and provide sample utterances for each to build voice applications. These voice applications should recognize out-of-domain inputs during run time without additional dialogue designer effort to provide out-of-domain training data. The dialogue manager can then react to out-of-domain utterances and improve user experience.

3.2.1 SeqGAN

The SeqGAN model proposed by Yu et al. [2017] is a starting point for the proposed OodGAN. SeqGAN is a sequence generation framework illustrated in Figure 3.1. Yu et al. [2017] denote the problem of sequence generation as follows. Given a dataset of real-world structured sequences, train a θ-parameterized generative model G_θ to produce a sequence

$$Y_{1:T} = (y_1, ..., y_t, ...y_T), y_t \in Y,$$

where Y is the vocabulary of candidate tokens. They apply reinforcement learning to this problem. In timestep t, the state s is the current produced tokens $(y_1, ..., y_{t-1})$ and the action a is the next token y_t to select.

They propose to additionally train a ϕ-parameterized discriminative model D_ϕ that provides guidance for improving generator G_θ. D_ϕ produces a probability $D_\phi(Y_{1:T})$ representing the probability of $Y_{1:T}$ being a real sequence versus a generated one. The discriminative model D_ϕ is trained with real sequence data, labelled as positive examples, and synthetic sequences from the generative model G_θ, labelled as negative examples.

SeqGAN uses the REINFORCE algorithm [Williams, 1992] to train generative model G_θ. Parameters of generative model G_θ are updated simultaneously by a policy gradient

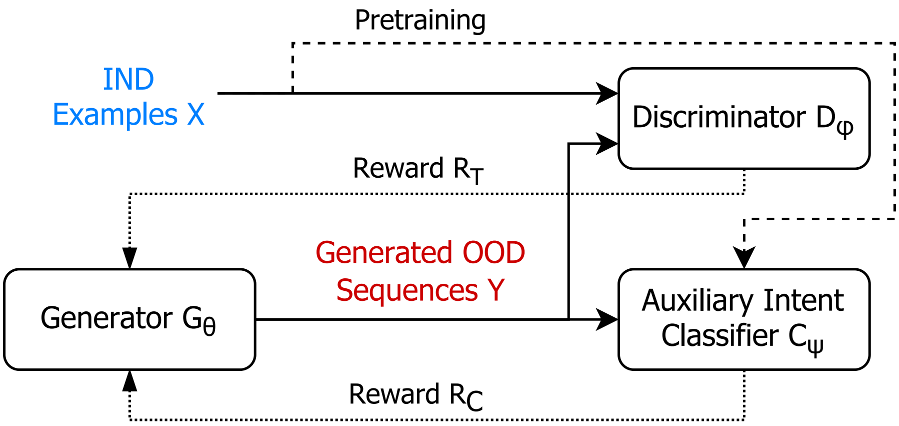


Figure 3.2: The overall architecture of the OodGAN. C_ψ is pre-trained to recognize intent classes for IND examples. D_ϕ is trained to distinguish between IND and generated OOD examples during adversarial training. G_θ is trained by the REINFORCE algorithm during adversarial training to generate OOD sequences. The training is guided by rewards R_C and R_T originating in C_ψ and D_ϕ.

and Monte Carlo search based on the expected end reward received from the discriminative model D_ϕ for the generated sequence. The reward is represented by the likelihood that the generated sequence will fool the discriminative model D_ϕ.

3.2.2 OodGAN

We propose OodGAN based on SeqGAN. There are two benefits of SeqGAN for our task of out-of-domain data generation. SeqGAN produces sequences similar to the training data, and it works directly on input sequence, unlike earlier model [Zheng et al., 2020], which works on latent space. Eliminating the auto-encoder responsible for converting a sequence of words into latent space reduces the overall model size. Also, our experiments with the work of Zheng et al. [2020] show a degradation in the overall performance due to the auto-encoder component. We present this finding in section 3.5.

Since our task is to generate out-of-domain data, we have the additional criterion that generated sequences should be close to the training in-domain sequences. However, we also want them not to belong to any in-domain intent class. We propose the OodGAN to achieve the two criteria.

The main difference between SeqGAN and OodGAN is the introduction of an auxiliary intent classifier. The auxiliary intent classifier C_ψ estimates the probability $C_\psi(z_i|Y)$ of example Y belonging into intent class z_i.

The task of the auxiliary intent classifier is to produce an additional reward signal. The reward signal guides the generator to produce a sequence not belonging to any in-domain intent class. The reward R_{C_ψ} coming from the auxiliary intent classifier for each generated example is defined as Shannon's Entropy

$$R_{C_\psi} = -\sum_{i=1}^{m} C_\psi(z_i|Y) \cdot log(C_\psi(z_i|Y)),$$

where m is the number of in-domain intent classes.

The intuition for using Shannon's Entropy is to reward a generator for producing examples for which the auxiliary intent classifier cannot clearly assign one of in-domain classes. In other words, the auxiliary classifier should assign a nearly uniform distribution across all intent classes for a well-generated example. The generator obtains a high reward for such examples because the uniform distribution has the highest Shannon's Entropy.

We train the auxiliary intent classifier to predict one of the classes $z_{1...m}$ for each training in-domain example $X_{1...n}$ during the pre-training step. We do not have to retrain it during adversarial training because in-domain intent classes' distribution does not change.

The goal of the generator is to generate a sequence that maximizes the expected sum of rewards from discriminator D_ϕ (the estimated probability of the sequence being real) and auxiliary intent classifier C_ψ (Shannon's Entropy calculated using estimated probabilities of sequence belonging to in-domain intent classes by auxiliary intent classifier).

Empirically, we evaluated different training strategies. We found that optimizing generator G using only the discriminator's reward first, followed by using only the auxiliary intent classifier reward and then repeating the process for each training batch produced the most stable results. This worked better than summing up the rewards from the discriminator and auxiliary intent classifier. When we tried summing up the two rewards, we noticed that the generator tended to collapse into a state in which it generated a single sequence highly rewarded by the auxiliary intent classifier, even though this did not happen for all training runs. We observed this situation even when we normalized rewards to a value between 0 and 1.

We also observed that part of the examples generated by OodGAN is semantically similar to some in-domain training examples or is generated multiple times. Examples that are identical or too close to in-domain examples are problematic and confuse the out-of-domain classifier. Duplicated examples do not represent the out-of-domain distribution effectively. For those reasons, we removed generated out-of-domain examples by an automatic filter that are identical or similar to in-domain examples or that are generated repeatedly.

To summarize, OodGAN's training procedure has the following steps:

1. **Train Auxiliary intent classifier:** First train auxiliary intent classifier to predict the classes $z_{1...m}$ for in-domain data $X_{1...n}$ until convergence.

2. **Train Generator as Language Model:** Next, train the generator on the in-domain data $X_{1...n}$ as a language model until it converges. This step makes it easier for the generator to fool the discriminator from the start of the adversarial training.

3. **Train Discriminator:** Generate adversarial examples from the generator. This training step helps the discriminator provide a useful reward signal from the start of adversarial training.

4. **Adversarial Training:** Perform adversarial training of generator and discriminator. There are three optimization steps for each training batch. First, optimize the generator using reward from discriminator as proposed by Yu et al. [2017]. Next, optimize the generator using a reward from the auxiliary classifier. Lastly, optimize the discriminator.

3.2.3 Filtering Mechanism

During experiments, we observed that part of the examples generated by OodGAN is semantically similar to in-domain training examples or is generated multiple times. Examples equal or too close to in-domain examples are problematic and confuse the out-of-domain classifier. Duplicated examples do not represent the out-of-domain distribution effectively. For those reasons, we propose the Filtering Mechanism, which removes generated out-of-domain examples that are equal or similar to in-domain examples and are generated repeatedly. We designed two variants of the Filtering Mechanism:

- **Hard Filter** removes duplicated occurrences of a generated out-of-domain example and removes examples equal to any training in-domain example.
- **Soft Filter** removes duplicated occurrences of a generated out-of-domain example and examples for which auxiliary intent classifier predicts in-domain intent class with a probability higher than the selected threshold. We experimented with different thresholds.

3.3 Data

We conducted experiments on ROSTD [Gangal et al., 2019] and OSQ [Larson et al., 2019] datasets.

- **ROSTD** contains three categories (alarm, reminder, and weather), each consisting of four intents. The dataset consists of 30,000 training, 4,000 validation and 8,000 testing in-domain examples. Out-of-domain examples were selected so that they do not belong to any category and do not share patterns with any in-domain examples. There are also no out-of-domain examples in the training set of the dataset. The testing set contains 4,500 out-of-domain examples. We present selected in- and out-of-domain examples from the ROSTD dataset in Table 3.5.
- **OSQ** consists of 150 intents. The dataset comprises 15,000 training, 3,000 validation and 4,500 testing in-domain examples. The dataset was created using Mechanical Turk. The turkers were given the name of the intent, and they were supposed to write intent examples fitting into the intent. The dataset authors manually went through examples and moved examples not fitting into the given intent class to the out-of-domain class. Thanks to this procedure, out-of-domain examples share the

same patterns as in-domain examples. The OSQ dataset contains 100 training out-of-domain examples. However, we do not use them for training due to the nature of our experiments. There are also 100 validation and 1,000 testing out-of-domain examples. We present selected in- and out-of-domain examples from the OSQ dataset in Table 3.6.

3.4 Experiments

This section explains the experiments we performed and the metric we used to evaluate the quality of our proposed method.

3.4.1 Evaluation Process

The evaluation process of OodGAN is complicated because we can not use upstream evaluation. In theory, we could compare generated out-of-domain data with a pool of testing out-of-domain data to measure the quality of generated out-of-domain data. However, because the distribution of out-of-domain data is limitless, our pool of testing out-of-domain data would capture only a part of the properties of complete out-of-domain data distribution. There is a possibility that generated out-of-domain data would be out of a different part of the entire distribution of out-of-domain data than testing out-of-domain data. This difference would unnecessarily penalize the measured quality of out-of-domain data. Hence, comparing testing out-of-domain data with generated out-of-domain data in upstream evaluation is not objective.

Instead, we evaluate the model on the downstream task of out-of-domain data detection and measure the change in out-of-domain data detection metrics. We designed experiments in the following way. We train the OodGAN on in-domain training examples as a first step. Next, we generate the out-of-domain examples using the trained model of OodGAN. We generate the same number of out-of-domain examples as the number of in-domain examples in the training set. In the third step, we train the threshold-based out-of-domain detection model using cross-entropy loss on training in-domain examples and negative entropy loss on generated out-of-domain examples. In the last step, we evaluate both in-domain and out-of-domain metrics.

3.4.2 Metrics

We evaluate the OodGAN by measuring metrics on the downstream task of out-of-domain detection. We measure AUROC, AUPR, and FPRN metrics [Ren et al., 2019, Hendrycks and Gimpel, 2017, Hendrycks et al., 2019] to evaluate the ability of OodGAN to generate out-of-domain data that helps intent classifier to distinguish in-domain and out-of-domain input utterances. We treat out-of-domain examples as the positive class. We also measure in-domain accuracy that evaluates the influence of generated out-of-domain data on the ability of the intent classifier to recognize the intents of in-domain data correctly.

- **AUROC** The area under the receiver operating characteristic (ROC) curve. The score says the probability that a randomly selected out-of-domain example will have a higher predicted probability of being an out-of-domain than a randomly selected in-domain example. A higher AUROC score is better.

- **AUPR** The area under the precision-recall curve when out-of-domain inputs are treated as positive samples. AUPR calculates the average precision score for all recall values. Intuitively, the higher the classification threshold we select, the more out-of-domain examples will be classified as out-of-domain. However, we risk that more in-domain examples will be classified as out-of-domain. AUPR expresses this risk. A higher AUPR score is better.

- **FPR*N*** The false-positive rate (FPR) when the true positive rate (TPR) is N%. FPRN metric is a practical value in real-world application since it evaluates an out-of-domain detection performance at a particular threshold. Lower FPRN means there is a smaller chance of in-domain examples triggering false alarm (in-domain example getting classified as out-of-domain) when the model correctly recognizes N% of out-of-domain examples. We report FPR when TPR is 0.95 and 0.90. A lower FPRN score is better.

We consider the FPRN metric as the most practical value in real-world application since it evaluates an out-of-domain detection performance at a particular threshold. We also measure in-domain accuracy that evaluates the influence of generated out-of-domain data on the ability of the intent classifier to recognize the intents of in-domain data correctly.

- **IND accuracy** The percentage of in-domain examples that have assigned correct intent label. We expect that generated out-of-domain examples can not improve the ability of the intent classifier to recognize intent labels for in-domain examples. However, generated out-of-domain examples can degrade the ability of the intent classifier to recognize the intents of in-domain examples. Thus, we measure the in-domain accuracy to evaluate whether generated out-of-domain examples negatively impact the intent classifier. Higher IND accuracy is better.

3.5 Results

We present the results of a proposed out-of-domain data generation model OodGAN, which is an extension of SeqGAN Yu et al. [2017]. We use OodGAN to generate out-of-domain data that share the same patterns as in-domain data and are very close to in-domain data distribution.

First, we conducted experiments to replicate results reported by Zheng et al. [2020] on the OSQ dataset. We created our implementation according to the paper's description because there is no publicly accessible implementation of their proposed model. We report results in Table 3.1.

OSQ [Larson et al., 2019]	**AUROC** ↑	**AUPR** ↑	**FPR 0.95** ↓	**FPR 0.90** ↓	**IND Acc.** ↑
Results reported by Zheng et al. [2020]	**95.4**	**98.9**	**25.0**	**10.1**	**93.3**
Our implementation of Zheng et al. [2020]	88.79	58.22	36.49	26.87	88.00

Table 3.1: OOD detection performance on the OSQ dataset with the model proposed by Zheng et al. [2020]

We could not reproduce the results reported by Zheng et al. [2020] even though we implemented the model as was described in the paper. The experiments showed that the denoising auto-encoder is a weak part of the architecture. Its token accuracy of text reconstruction on the validation set was only 0.37%. Thus, the auto-encoder's low performance is why the generator generates poor-quality examples.

Next, we compare OodGAN with baselines. We selected two baselines to evaluate improvements of our proposed OodGAN. Baselines for the ROSTD dataset are our implementation of Zheng et al. [2020] and the work of Gangal et al. [2019]. The baseline for the OSQ dataset is our implementation of Zheng et al. [2020]. To compare OodGAN with baselines, we generated out-of-domain data using OodGAN for both ROSTD and OSQ datasets and evaluated the performance of the out-of-domain detection model. We generated out-of-domain data for both datasets with various filtering mechanism settings. In the main results, we present only the best results out of all the filtering mechanism settings we used for OodGAN. We used a Soft Filter with a maximal intent probability threshold set to 0.5 for the ROSTD dataset and a Soft Filter with a maximal intent probability threshold set to 0.25 for the OSQ dataset. We present the experimental results and the impact of each filtering mechanism at the end of this section.

Table 3.2 shows results on ROSTD dataset and Table 3.3 shows results on OSQ dataset. Results on ROSTD data are promising. They show around 65% relative improvement in FPR 0.95 compared to the baseline of our implementation of Zheng et al. [2020] and around 5% absolute improvement in FPR 0.95 compared to baseline of Gangal et al. [2019]. For the more challenging OSQ dataset, there is around 28% relative improvement in FPR 0.95 and FPR 0.90 compared to the baseline.

Improvement in the OSQ dataset is smaller due to a large number of intents. There are 150 intents. The generator must learn their distribution to generate examples outside their distributions. Also, the generator depends on rewards from the auxiliary classifier. Thus, the auxiliary classifier's low performance can lead to a situation in which the generator generates in-domain data. Our analysis found that auxiliary classifier performance on the OSQ dataset is weak on several intents, causing smaller improvements.

OSQ dataset also contains training out-of-domain examples. Thus, we evaluated the performance of an out-of-domain classifier trained on them as well in order to compare generated out-of-domain examples with out-of-domain examples from the training set. Ta-

ROSTD [Gangal et al., 2019]	AUROC ↑	AUPR ↑	FPR 0.95 ↓	FPR 0.90 ↓	IND Acc. ↑
w.o. OOD	97.64	93.86	8.10	5.56	**99.05**
Our implementation of Zheng et al. [2020]	88.67	54.84	37.82	26.04	88.00
Gangal et al. [2019]	98.22	**96.47**	7.41	-	-
OodGAN	**98.99**	96.26	**2.59**	**1.37**	98.31

Table 3.2: Performance of out-of-domain detection on the ROSTD dataset

OSQ [Larson et al., 2019]	AUROC ↑	AUPR ↑	FPR 0.95 ↓	FPR 0.90 ↓	IND Acc. ↑
w.o. OOD	90.89	97.99	28.11	20.98	89.04
Training OOD	**91.72**	**98.29**	31.78	21.87	86.18
Our implementation of Zheng et al. [2020]	88.79	58.22	36.49	26.87	88.00
OodGAN	91.24	97.79	**26.07**	**19.29**	**90.11**

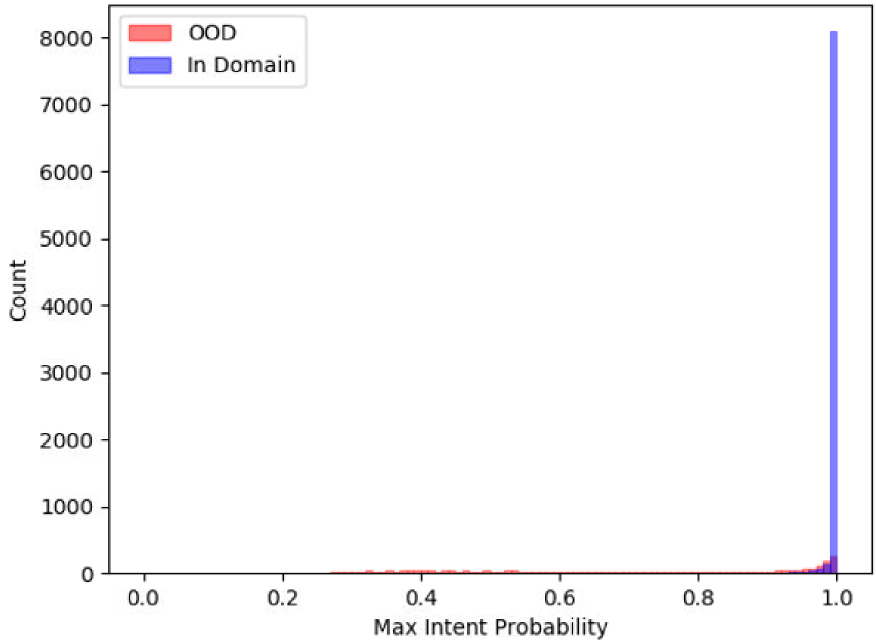
Table 3.3: Performance of out-of-domain detection on the OSQ dataset

ble 3.3 shows that using training out-of-domain data yields better AUROC and AUPR numbers than the model trained with generated data. However, our goal was to propose a system that can only work with in-domain training examples. Hence, the comparison between classifier using training and generated out-of-domain examples is only for illustration.

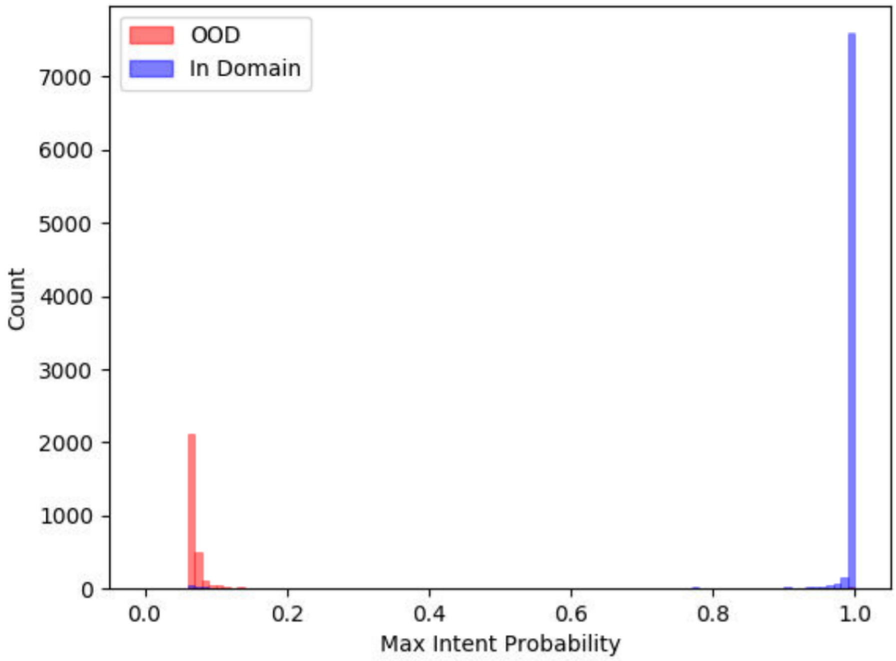
To evaluate whether OodGAN helps the threshold-based out-of-domain recognition model to discriminate between out-of-domain and in-domain examples, we plotted the histogram of the maximum intent probability of test data. We also plotted a histogram for the system trained without any out-of-domain examples to see the improvement we gain by using OodGAN. Figures 3.3 and 3.4 show the histogram for ROSTD and OSQ datasets respectively.

Probability scores for in-domain (blue) and out-of-domain (red) data are spread out over all probability values when no out-of-domain data is used for model training. Thus, it is hard to select a well-discriminating threshold. The result of the model trained with out-of-domain data is significantly better. The ROSTD dataset clearly separates in-domain and out-of-domain data, with in-domain data receiving high intent scores and out-of-domain data receiving low intent scores. The graph for the OSQ dataset is more spread out, but there is still a clear separation between in-domain and out-of-domain data. Thus, OodGAN generates out-of-domain examples that improve the ability of the out-of-domain recognition model to discriminate between out-of-domain and in-domain examples.

The out-of-domain detection model is combined with an intent classifier in many real-world applications. For this reason, the joint accuracy of out-of-domain detection and in-domain intent recognition is an important metric. We show how the joint accuracy

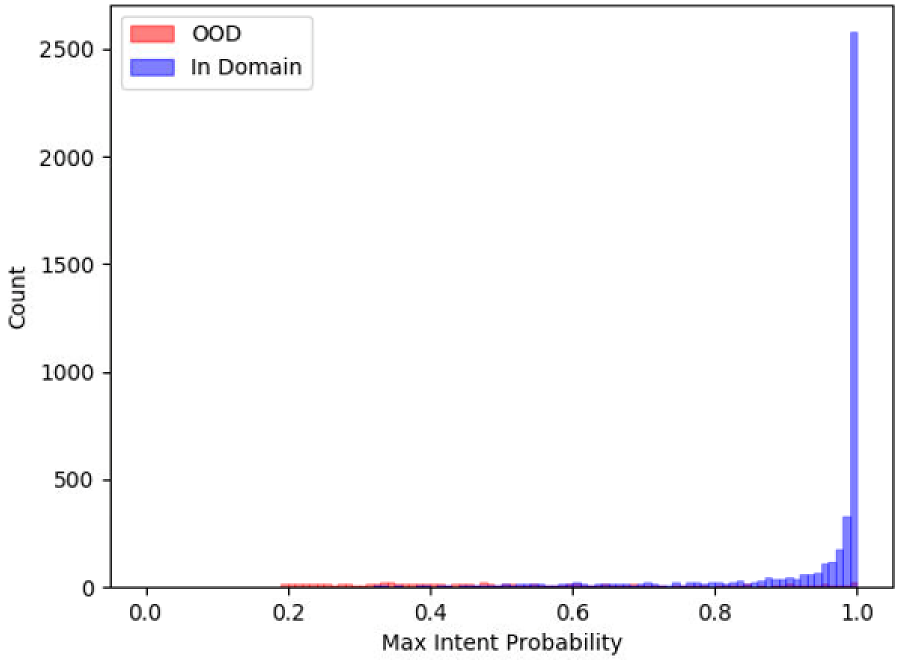


(a) Model trained with no out-of-domain examples

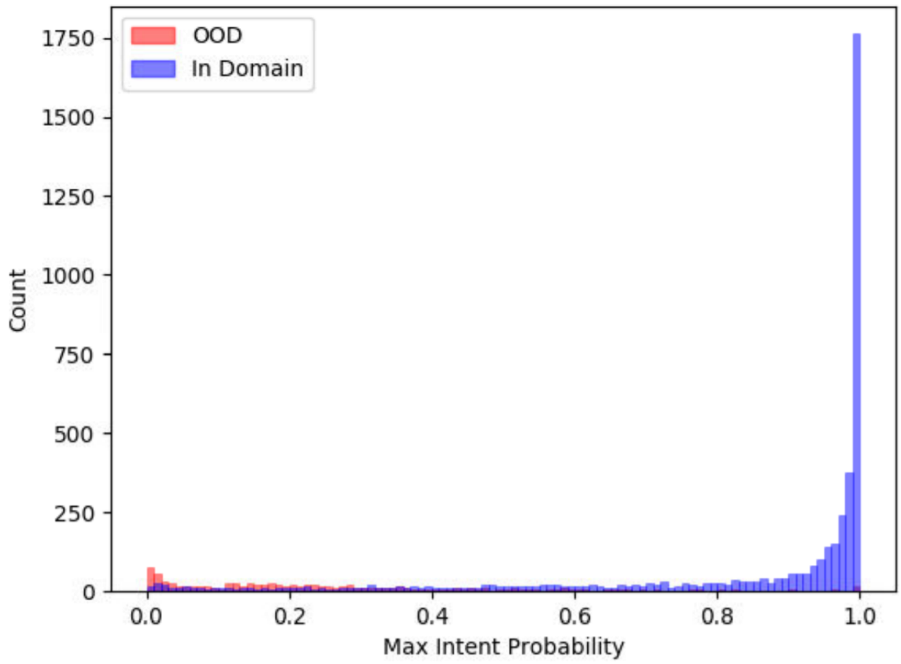


(b) Model trained with generated out-of-domain examples

Figure 3.3: Distributions of intent scores corresponding to the in-domain and out-of-domain examples of the ROSTD dataset



(a) Model trained with no out-of-domain examples



(b) Model trained with generated out-of-domain examples

Figure 3.4: Distributions of intent scores corresponding to the in-domain and out-of-domain examples of the OSQ dataset

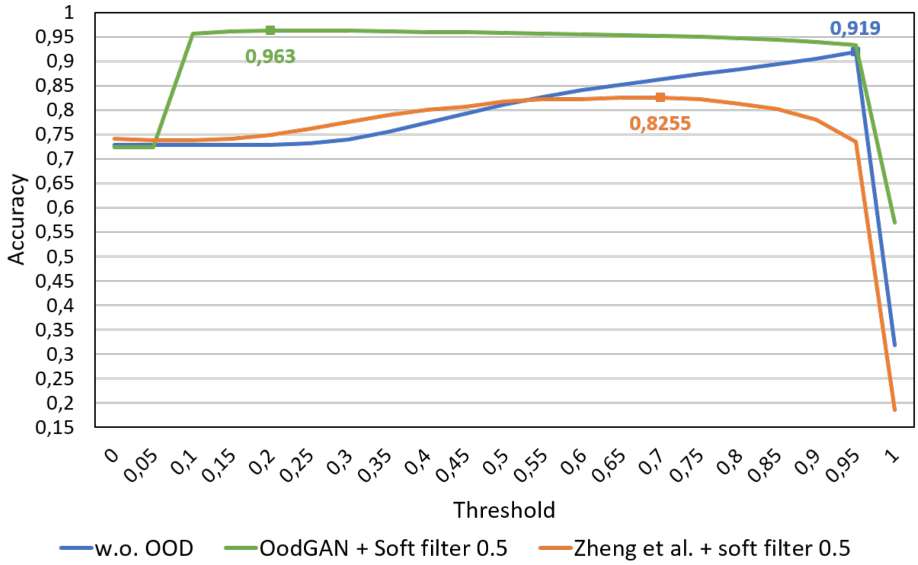


Figure 3.5: Joint accuracy for ROSTD data across different threshold values. Points mark the highest joint accuracy of out-of-domain detection and in-domain intent recognition.

Model	# Parameters
Zheng et al. [2020]	7M
SeqGAN [Yu et al., 2017]	800K
OodGAN	2M

Table 3.4: Number of parameters

depends on the selected threshold in Figures 3.5 and 3.6.

To draw this graph, we selected different thresholds and classified examples with an intent score below the threshold as out-of-domain. We classify intent for the rest. Our proposed approach leads to high joint accuracy of out-of-domain detection and in-domain intent recognition with low threshold values. That confirms that models trained with generated out-of-domain data assign low scores to out-of-domain examples and high scores to in-domain examples.

The separation between generated out-of-domain and in-domain examples is also visible in t-SNE [Hinton and Roweis, 2002] visualization. Figures 3.7 and 3.8 show the t-SNE visualization of in-domain and generated out-of-domain data. We can notice that generated out-of-domain data create recognizable clusters close to in-domain data but do not mix with it.

We also compare the number of parameters between OodGAN, SeqGAN, and Zheng et al. [2020] in Table 3.4. The table shows that OodGAN is more frugal than Zheng et al. [2020] in the model size.

Finally, we list out-of-domain examples generated by OodGAN in tables 3.5 and 3.6. Examples demonstrate that OodGAN can generate grammatically correct sentences, and it

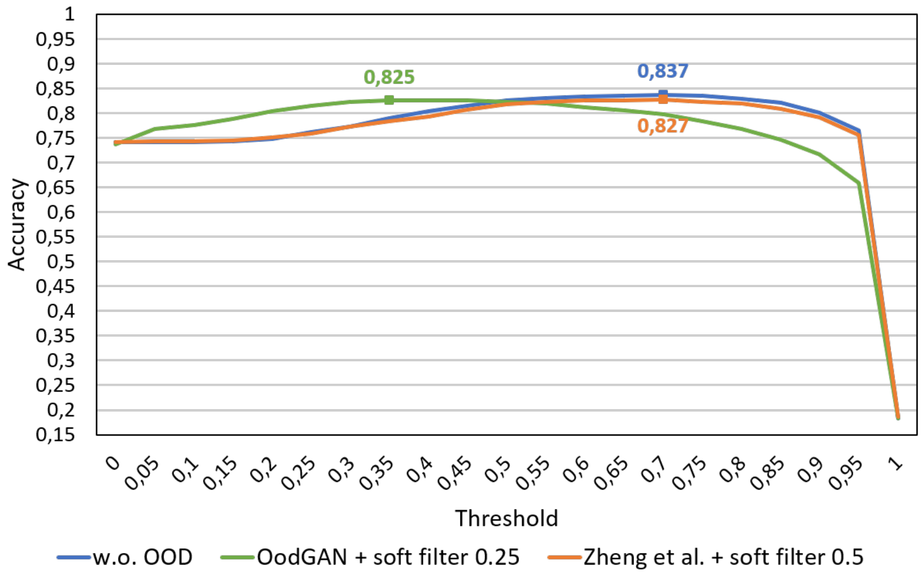


Figure 3.6: Joint accuracy for OSQ data across different threshold values. Points mark the highest joint accuracy of out-of-domain detection and in-domain intent recognition.

can also generate entities (such as *"Game of Thrones"* or *"Galway"*). This is confirmation that examples generated by OodGAN are close to in-domain data. Results of t-SNE visualization and observation of generated out-of-domain examples confirm the findings of Zheng et al. [2020] that training out-of-domain recognition model with out-of-domain examples that are just outside in-domain data distribution should be sufficient to handle most of the out-of-domain inputs during runtime.

Table 3.7 and Table 3.8 shows filtering influence on ROSTD and OSQ dataset respectively. We evaluated OodGAN without the Filtering Mechanism, with Hard Filter and Soft Filter using several thresholds. We can see that the Filtering Mechanism positively influences the performance of out-of-domain detection. The reason is that OodGAN, without any filtering mechanism, tends to generate some examples that are similar or too close to in-domain examples. Such examples confuse the out-of-domain classifier that part of the in-domain data distribution belongs to the out-of-domain class. The Hard Filter removes generated examples that are part of training in-domain data. Thus, it improves the performance of the out-of-domain classifier. Part of the generated out-of-domain examples is a paraphrase of training in-domain examples. Such generated out-of-domain examples are highly similar to training in-domain examples and are removed by the Soft Filter for this reason. Results show that lowering the similarity threshold on Soft Filter yields better results.

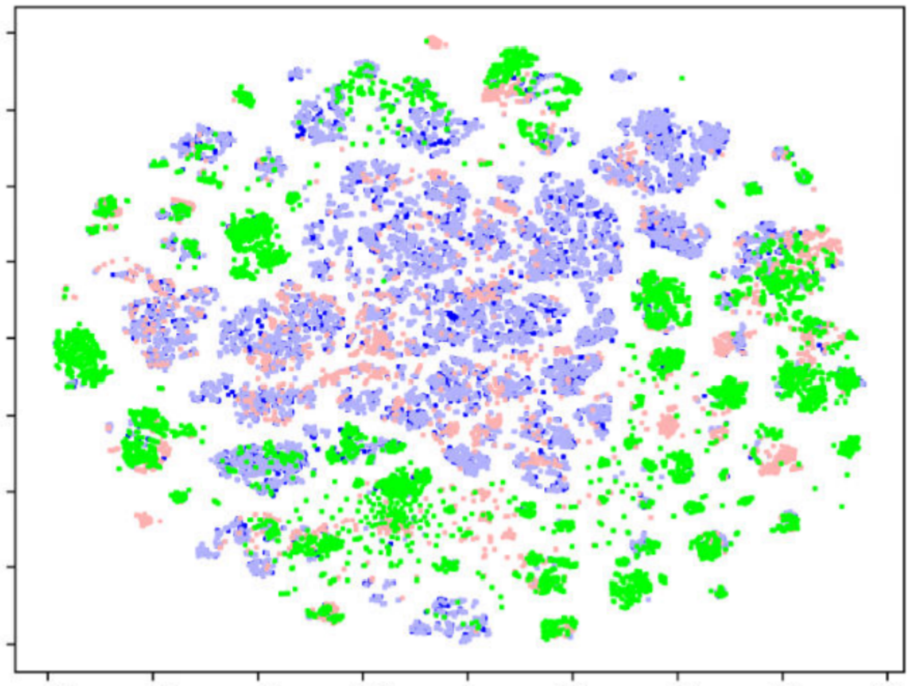


Figure 3.7: t-SNE visualization of the BERT feature vectors associated with the examples from the ROSTD dataset. The blue colour represents IND examples, and the red colour represents OOD examples. A lighter shade of the colour represents training examples, and a darker shade represents validation and testing examples. The green colour represents examples generated by OodGAN.

IND Examples	Should I be expecting rain today I need a new alarm for 8:30 am Show my reminders Show me the extended forecast please Snooze alarm for 5 more minutes
OOD Examples	Why do people watch television Where do pineapples grow Should I go to the mall today or tomorrow Tell me how to install a pool Transfer my PayPal balance to my bank
Generated by OodGAN	Remind me of my 4pm and Game of Thrones alarm When should I unpack Add day at workout please Give me my Sarasota appointment Do I need to pack to Galway this umbrella

Table 3.5: Examples sampled from the IND and OOD test set of the ROSTD dataset and OOD utterances generated using the OodGAN model.

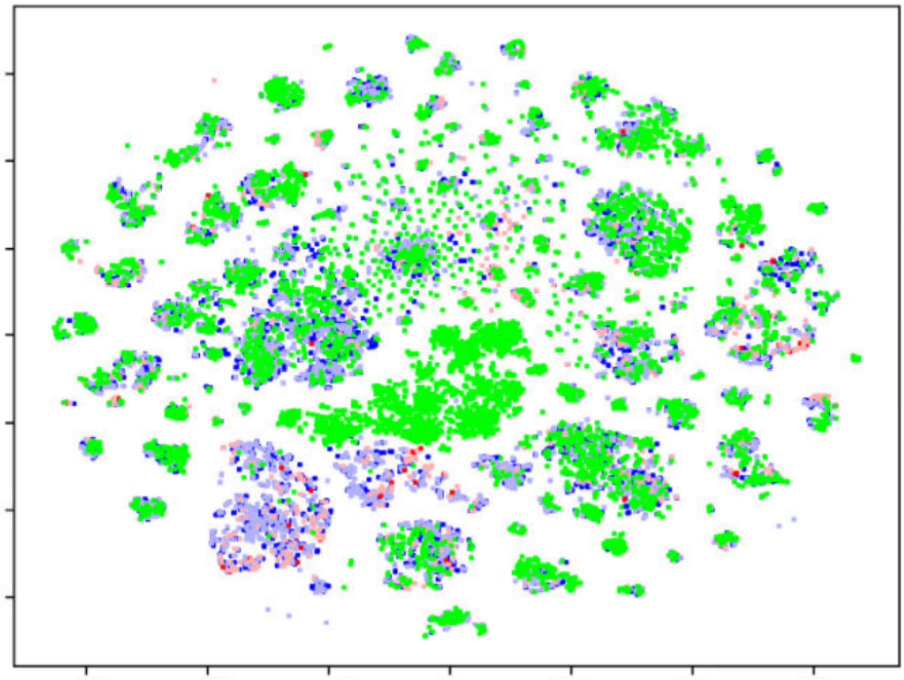


Figure 3.8: t-SNE visualization of the BERT feature vectors associated with the examples from the OSQ dataset. The blue colour represents IND examples, and the red colour represents OOD examples. The lighter shade of the colour represents training examples, and the darker shade represents validation and testing examples. The green colour represents examples generated by OodGAN.

IND Examples	How do you say please in French Transfer $800 from my checking to savings account Set a 5 minute timer Can I edit my insurance policy Can you call me a different name
OOD Examples	How is my driving Can you tell me who sells Dixie paper plates Which stocks have lost the least today Close all internet tabs When is the next full moon
Generated by Zheng et al. [2020]	Change the the to of the to do list How do i get to my my my card card How many calories are in a cup What is the next time to get to the oil Make sure to the to for a meeting
Generated by OodGAN	Can you find an option Please address me softer How do I show Noel Can I hurt the company by that song Can you assure me if I lost the exchange for bank

Table 3.6: Examples sampled from the IND and OOD test set of the OSQ dataset and OOD examples generated using Zheng et al. [2020] and OodGAN models.

3.6 Conclusion of Chapter

This chapter proposed an out-of-domain data generation model, OodGAN, that generates out-of-domain examples. The generated-of-domain examples improve out-of-domain detection, an important signal for Hybrid dialogue management (see chapter 6). The model does not require any out-of-domain training examples. Moreover, the model uses fewer parameters than other approaches thanks to not relying on the auto-encoder to map utterances into latent space. It models the data generator as a stochastic policy in reinforcement learning instead. The fact that it uses a smaller number of parameters makes the OodGAN

ROSTD [Gangal et al., 2019]	**AUROC** ↑	**AUPR** ↑	**FPR 0.95** ↓	**FPR 0.90** ↓	**IND Accuracy** ↑
w.o. OOD	97.64	93.86	8.10	5.56	**99.05**
OodGAN w.o. Filter	75.52	47.28	73.20	59.70	98.47
OodGAN+Hard Filter	87.78	64.13	36.91	27.55	97.87
OodGAN+Soft Filter 0.75	98.71	95.74	3.35	2.08	97.85
OodGAN+Soft Filter 0.50	**98.99**	**96.26**	**2.59**	**1.37**	98.31

Table 3.7: Influence of Filtering Mechanism on ROSTD dataset

OSQ [Larson et al., 2019]	**AUROC** ↑	**AUPR** ↑	**FPR 0.95** ↓	**FPR 0.90** ↓	**IND Accuracy** ↑
w.o. OOD	90.89	97.99	28.11	20.98	89.04
OodGAN w.o. Filter	86.39	96.80	35.84	28.27	90.64
OodGAN+Hard Filter	87.03	97.00	34.84	27.60	91.38
OodGAN+Soft Filter 0.75	88.96	97.46	29.60	21.87	**91.42**
OodGAN+Soft Filter 0.50	91.11	**97.91**	26.71	**17.91**	91.02
OodGAN+Soft Filter 0.25	**91.24**	97.79	**26.07**	19.29	90.11

Table 3.8: Influence of Filtering Mechanism on OSQ dataset

more practical for real-world applications from the point of view of resource requirements. The model uses two rewards for the generator. The discriminator's reward guides the generator to generate examples as close to the IND data as possible. The auxiliary intent classifier's reward guides the generator to generate examples with low probabilities for all intent classes. Our experiments show that OOD examples generated by OodGAN improve the performance of the OOD detection problem, which is critical in conversational AI.

Chapter 4

Named Entity Density for Summarization

This chapter proposes a method for text summarization utilizing entities as features. [Marek et al., 2021a] We propose an extractive summarization approach, Named Entity Density, that selects a sentence with the highest ratio between a number of entities and the length of the sentence as the summary of the article. We demonstrate that the selected sentence reflects the style of reports concisely identifying to whom, when, where, and what happened. We show that such a summary is beneficial in combination with the first sentence of an article in voice applications presenting news articles. We use summaries of articles as an efficient way to ground responses of neural response generators in knowledge, as we demonstrate in chapter 5. Using summaries instead of whole articles is desirable for language models with a limited number of input tokens.

4.1 State of the art

Allahyari et al. [2017] briefly survey extractive and abstractive methods for text summarization. We introduce methods for both approaches in the following subsections.

4.1.1 Extractive Summarization

Mihalcea and Tarau [2004] introduce a Textrank algorithm, a graph-based ranking model. It creates a graph of sentences based on their overlap. It chooses the most important sentences according to the created graph. Pal and Saha [2014] propose a summarization algorithm that derives the relevance of the sentences within the text using the Simplified Lesk algorithm and the WordNet online database. Kågebäck et al. [2014] propose using continuous vector representations for semantically aware representations of sentences for summarization. Zhang et al. [2016] develop convolutional neural networks that learn sentence features and perform sentence ranking. The latest results are achieved by Liu [2019]. They apply the BERT model [Devlin et al., 2018] to extractive summarization.

Especially relevant works for our research are those working with named entities. Nobata et al. [2002] introduce named entity tagging and pattern discovery to a summarization system based on a sentence extraction technique. Hassel [2003] integrates a Named Entity tagger into the SweSum summarizer for Swedish newspaper texts. Filatova and Hatzivassiloglou [2004] propose a summarization technique using features based on low-level atomic events that describe the relationships between important actors in a document or a set of documents. The extraction of atomic events relies on a noun phrase and named entity recognition [Hatzivassiloglou and Filatova, 2003]. Jabeen et al. [2013] apply named entity recognition for summarization of tweets. Schulze and Neves [2016] present EntityRank, a multi-document graph-based summarization algorithm solely based on named entities. They apply it to texts from the medical domain. Khademi and Fakhredanesh [2020] propose an unsupervised method for summarizing Persian texts that use a named entity recognition system. Their method consists of three phases: training a supervised NER model, recognizing named entities in the text, and generating a summary.

4.1.2 Abstractive Summarization

Nallapati et al. [2016] models abstractive text summarization using Attentional Encoder-Decoder Recurrent Neural Networks. They propose several novel models that address critical problems in summarization that are not adequately modelled by the basic architecture, such as modelling keywords, capturing the hierarchy of sentence-to-word structure, and emitting rare or unseen words during the training time. Liu et al. [2017] propose an adversarial process for abstractive text summarization. Yao et al. [2018] propose a recurrent neural network-based Seq2Seq attentional model with a dual encoder including the primary and the secondary encoders. Song et al. [2019] propose an LSTM-CNN-based approach that can construct new sentences by exploring more fine-grained fragments than sentences, namely, semantic phrases. The proposed approach is composed of two main stages. The first stage extracts phrases from source sentences. The second stage generates text summaries using deep learning. Liu and Lapata [2019] apply BERT in text summarization and propose a general framework for extractive and abstractive models. For abstractive summarization, they propose a new fine-tuning schedule that adopts different optimizers for the encoder and the decoder to alleviate the mismatch between the two, as the former is pretrained while the latter is not.

4.2 Method

In our experiments with summarization, we focus on the summarization of Czech news articles by a one-sentence summary. We can also describe this task as the automatic creation of a headline for a given text. We use the SumeCzech dataset [Straka et al., 2018] for our experiments.

Additionally, we explore the influence of the named entities on text summarization. We use SpaCy's named entity recognition (NER) model, trained on a CoNLL-based extended

CNEC 2.0 dataset [Ševčíková et al., 2014], to label SumeCzech with named entities to create additional features.

We use the annotations as a foundation for our newly proposed method, which we call *Named Entity Density.* The method selects the sentence with the highest ratio of the number of entities to the sentence length as the summary of the article. We show that our proposed method achieves nearly as good results in the automatic evaluation as the hard-to-beat baseline in the news domain that selects the first sentence. Nenkova [2005] shows that the baseline selecting the first sentence is strong because authors tend to summarize the main points of an article in the first sentence, especially in the news domain. We also show that sentences selected by *Named Entity Density* possess a high information value mentioning *to whom*, *where*, *when*, and *what* happened. This structure resembles the style of reports that concisely identifies and examines issues, events, or findings that have happened. We propose that such a summary is helpful in voice applications presenting news. Voice application can present the summary formed out of the first sentence of a news article and continue with the sentence selected by *Named Entity Density* if a user requests additional information. Moreover, both methods can be used as input to neural response generators to ground their response in knowledge. Input constructed by summarization, instead of an entire article, is advantageous because it is efficient in terms of length. The short length is important for transformer-based neural response generators with limited input sizes.

We also propose two abstractive methods that can construct a novel sentence as a summary. They are based on the Seq2Seq architecture, initially used for machine translation. The first method uses the text of the article only. The second method uses additional annotations created by the named entity recognition system as input features. Our experiments show that both models achieved state-of-the-art results in SumeCzech's task to summarize the headline from the text of the article. We also show that the named entities added as an additional input feature improve the ability of the model to generalize to the out-of-domain data.

Finally, we propose a new metric, ROUGE_{NE}, which measures the overlap of named entities in the target and generated summaries. Poor results of the experiments in ROUGE_{NE} show that summarization of entities still poses many challenges, and this task has not been fully solved yet.

4.2.1 Named Entity Density

Named Entity Density is our proposed unsupervised extractive method. It calculates each sentence's named entity density score and selects the sentence with the highest score. The score is calculated in two steps. First, we apply a named entity recognition algorithm to all sentences. Next, we calculate the named entity density as a ratio of the number of tokens marked as a named entity to the total number of tokens in the sentence.

Formally, let us denote the article A, for which we want to create a summary, as a set of sentences $s_0 \dots s_n$:

$$A = \{s_0, s_1, s_2, \ldots, s_n\}.$$

Each sentence s_i contains word tokens $x_0 \ldots x_m$:

$$s_i = \{x_0, x_1, x_2, \ldots, x_m\}.$$

We apply the named entity recognition algorithm NER on each sentence s_i of the article A:

$$NER(s_i) = \{e_0, e_1, e_2, \ldots, e_m\},$$

that produces NER labels $e_0 \ldots e_m$ for each token $x_0 \ldots x_m$ of the sentence s_i. We can divide the NER tokens into two sets: E_{NE} and $E_{\neg NE}$. Each NER token belongs to exactly one of those two sets. E_{NE} contains all NER tokens representing some entity type. $E_{\neg NE}$ contains all NER tokens that do not represent any entity type. In the analogy of the IOB format, E_{NE} contains I and B tokens, and $E_{\neg NE}$ contains O tokens. Next, we calculate the named entity density NED for each sentence $s_0 \ldots s_n$ of the article A. The NED is defined as:

$$NED(s_i) = \frac{|E_{NE}|}{|s_i|},$$

where $|E_{NE}|$ denotes the number of tokens in the sentence s_i which NER algorithm marked as named entities and $|s_i|$ is the number of all tokens $x_0 \ldots x_m$ forming the sentence s_i. We select the sentence s_i with the highest NED score as a summary of article A.

The intuition of the *Named Entity Density* is that the sentence with the high NED score mentions the highest number of entities within the smallest text fragment. Such a sentence corresponds to the form of a report that is structured around concisely identifying and examining issues, events, or findings that have happened.

4.2.2 Seq2Seq

Seq2Seq is a supervised abstractive summarization method that uses a Seq2Seq neural network with global attention. Formally, a Seq2Seq neural network models the conditional probability $p(y|x)$ of translating a source text $x = \{x_0, x_1, x_2, ..., x_n\}$ into a target text $y = \{y_0, y_1, y_2, ..., y_m\}$ [Luong et al., 2015]. The source text x is an article, y is a summary, and $m < n$ in our case. The Seq2Seq neural network consists of an encoder and a decoder. The encoder creates a fixed-length vector representation r of the source text x:

$$r = ENC(x).$$

The encoder is usually a recurrent neural network with hidden states h_s^{ENC}:

$$h_s^{ENC} = f^{ENC}(h_{s-1}^{ENC}, x_s),$$

and the output of the encoder is its last hidden state:

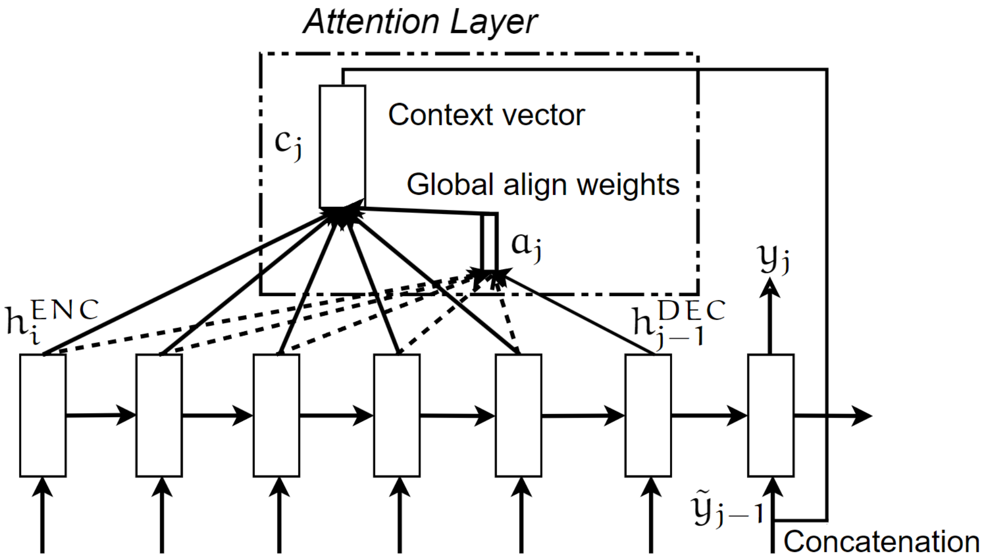


Figure 4.1: Seq2Seq with a global attention mechanism. The figure is inspired by Luong et al. [2015]

$$ENC(x) = h_n^{ENC}$$

The f^{ENC} can be a vanilla RNN [Rumelhart et al., 1985], LSTM [Hochreiter and Schmidhuber, 1997], or GRU [Cho et al., 2014] unit.

The decoder takes r as an input and generates a target text, one token at a time:

$$\log p(y|x) = \sum_{j=1}^{m} \log p(y_j|y_{<j}, r).$$

We can also represent the probability of generating a target word y_j as:

$$p(y_j|y_{<j}, r) = softmax(g(h_j^{DEC})),$$

where g is a transformation function that generates a vocabulary-sized vector. The h_j^{DEC} is the output of recurrent neural network unit:

$$h_j^{DEC} = f^{DEC}(h_{j-1}^{DEC}, y_{j-1}).$$

The function f^{DEC} can be a vanilla RNN, LSTM, or GRU unit, like in the case of the encoder.

We add a global attention mechanism to the Seq2Seq neural network. The attention allows the network to selectively focus on parts of the source text during the target text generation. We illustrate the global attention mechanism in Figure 4.1.

The idea is to concatenate a source-side context vector c_j with the input vector of the encoder y_{j-1}:

$$\tilde{y}_{j-1} = [c_j, y_{j-1}].$$

The vector $\tilde{y}_{j-1}$ is fed into f^{DEC}:

$$h_j^{DEC} = f^{DEC}(h_{j-1}^{DEC}, \tilde{y}_{j-1}).$$

The context vector c_j is computed as a weighted average over all vectors of hidden states of the encoder h_s^{ENC}:

$$c_j = \sum_{i=0}^{n} a_{j_i} \cdot h_i^{ENC},$$

where a_{j_i} is the ith element of the weight vector a_j.

The a_j is calculated by the softmax function comparing each hidden state of the encoder h_s^{ENC} with the current hidden state of the decoder h_j^{ENC}:

$$a_{j_i} = \frac{exp(score(h_{j-1}^{DEC}, h_i^{ENC}))}{\sum_{s'} exp(score(h_{j-1}^{DEC}, h_{s'}^{ENC}))}.$$

There are multiple definitions of the *score* function. We selected the *general score* function, defined as:

$$score(h_t, h_s) = h_t^{\top} W h_s.$$

4.2.3 Seq2Seq–NER

Seq2Seq–NER is a supervised abstractive method that uses a Seq2Seq model with global attention and adds the NER feature encoded as a one-hot encoded vector appended to the input embedding vector. Formally, the Seq2Seq neural network models the conditional probability $p(y|x_{NER})$, where y is a target text $y = \{y_0, y_1, y_2, ..., y_m\}$ and x_{NER} is a source sequence. The source sequence is a concatenation of a vector representation of the source token x_i and one-hot vector representation of entity type e_i:

$$x_{NER} = \{[x_0, e_0], [x_1, e_1], [x_2, e_2], \ldots, [x_n, e_n]\}.$$

The rest of the model works similarly to the Seq2Seq model we have already described. To summarize, the difference between *Seq2Seq* and *Seq2Seq–NER* models is that the latter has access to the named entity labels of the source words produced by the NER algorithm.

4.2.4 ROUGE$_{\text{NE}}$ Summarization Metric

Since we focus on the role of named entities in summarization, we propose a novel metric ROUGE$_{\text{NE}}$. ROUGE$_{\text{NE}}$ measures the overlap of named entities between the reference and the generated summaries. This metric evaluates the model's ability to transfer named entities to the summary.

Formally, let us denote the tokens of a true summary X:

$$X = \{x_0, x_1, x_2, \dots, x_n\},$$

where x_i are individual tokens of the summary. Let us denote the generated summary Y in a similar fashion:

$$Y = \{y_0, y_1, y_2, \dots, y_m\}.$$

Next, we apply the named entity recognition algorithm on X and Y. The result is entity annotations xe_i and ye_i for all tokens x_i and y_i:

$$\{xe_0, xe_1, xe_2, \dots, xe_n\},$$

$$\{ye_0, ye_1, ye_2, \dots, ye_m\}.$$

The annotations can be divided into a set of annotations E_{NE} that mark entities and annotations $E_{\neg NE}$ that do not mark entities. In the analogy of IOB format, the former are I and B annotations, and the latter are O annotations. For the calculation of ROUGE_{NE}, we select only tokens that are marked as entities. Formally, we select only the tokens of summaries X and Y, for which its entity label is an element of E_{NE}. The results are the X_e and Y_e:

$$X_e = \{x_i\} \text{ for } i = 0 \dots n \text{ if } xe_i \in E_{NE},$$

$$Y_e = \{y_i\} \text{ for } i = 0 \dots m \text{ if } ye_i \in E_{NE}.$$

Next, we calculate the ROUGE precision and recall scores using the tokens of X_e and Y_e as follows:

$$precision = \frac{|X_e \cap Y_e|}{|X_e|},$$

$$recall = \frac{|X_e \cap Y_e|}{|Y_e|},$$

where $|X_e|$ and $|Y_e|$ denote the sizes of X_e and Y_e. $|X_e \cap Y_e|$ denotes the number of overlapping tokens between X_e and Y_e. The resulting values are the precision and recall of ROUGE_{NE}. We define the metrics to be equal to zero for summaries without any named entity.

4.3 Data

We use SumeCzech for experiments. SumeCzech is a Czech news-based summarization dataset created by Straka et al. [2018]. It contains over a million documents, consisting of

Website	**Number**	**Percentage**
ceskenoviny.cz	4,854	0.5%
denik.cz	157,581	15.7%
idnes.cz	463,192	46.2%
lidovky.cz	136,899	13.7%
novinky.cz	239,067	23.9%
Total	1,001,593	100.0%

Table 4.1: Number of documents in SumeCzech from individual news websites

a headline, several sentences of abstract, and a full text. The dataset was collected from various Czech news websites. We show the distribution of the websites in Table 4.1.

SumeCzech is split into four parts. Three of them are the train, development, and test sets. Additionally, to simulate a real-life situation where a model is trained on data from one domain and used on real data from other domains, Straka et al. [2018] created an out-of-domain (OOD) test set. OOD test set evaluates how models cope with news articles from domain never seen during training. They clustered the whole dataset into 25 clusters using K-Means on abstracts of the articles and selected one cluster as the OOD test set. The OOD test set contains approximately 4.5% of all articles. The OOD testing set seems to contain news articles about concerts and festivals. The remaining articles were divided into train, development, and test sets in 86.5 : 4.5 : 4.5 ratio.

4.3.1 Named Entity Annotations

We train a model for named entity recognition in the Czech language to annotate SumeCzech by named entities. We selected the CoNLL-based extended CNEC 2.0 [Konkol et al., 2014] as the training dataset, as it is the largest and most up-to-date Czech named entity recognition dataset. The advantage is that the dataset contains no nested entities, making the outputs easier to use for summarizers.

We selected SpaCy's NER model[1] [Honnibal et al., 2020] because previous experiments by Müller [2020a] showed that SpaCy's NER model offers a good trade-off between performance, speed, and memory requirements. Speed and memory requirements might seem unimportant for our experiments because we can precompute the annotations. However, for practical usage, in which the labels have to be created as soon as possible once a new document for summarization arrives, we also decided to take those properties into account. The SpaCy's NER model achieved a 78.45 F-Score on the CoNLL-based extended CNEC 2.0 testing set. For comparison, the current state-of-the-art result on this dataset is 86.39 F-Score [Straková et al., 2019, Müller, 2020b].

We applied the trained SpaCy's NER model to the text of SumeCzech's articles. The result was annotations in IOB2 format, with one label for each word token. The NER found

[1] `https://spacy.io/api/entityrecognizer`

Entity Type	**Train**	**Dev**	**Test**	**Test OOD**
Numbers in addresses	116,990	5,052	5,129	1,827
Geographical names	5,271,938	282,440	285,307	212,637
Institutions	4,488,357	222,524	234,147	250,555
Media names	534,340	24,379	27,966	22,360
Artifact names	2,367,532	118,938	108,811	196,009
Personal names	7,991,790	406,938	395,867	646,556
Time expressions	1,684,152	87,096	86,866	121,357
Total	22,455,099	1,147,367	1,144,093	1,451,301

Table 4.2: Number of named entities in texts of SumeCzech's articles

Entity Type	**Train**	**Dev**	**Test**	**Test OOD**
Numbers in addresses	331	18	12	26
Geographical names	285,148	15,903	14,697	13,502
Institutions	161,809	7,578	8,472	12,806
Media names	9,088	371	420	718
Artifact names	62,124	3,344	2,837	7,748
Personal names	302,276	15,117	15,856	31,266
Time expressions	14,400	760	838	1,127
Total	835,176	43,091	43,132	67,193

Table 4.3: Number of named entities in headlines of SumeCzech's articles

Entity Type	**Train**	**Dev**	**Test**	**Test OOD**
Numbers in addresses	1,686	105	85	83
Geographical names	773,901	41,759	38,903	33,001
Institutions	601,129	28,380	33,119	52,938
Media names	77,591	3,744	4,320	3,946
Artifact names	159,122	7,550	7,174	25,204
Personal names	747,686	36,783	37,712	65,950
Time expressions	132,276	7,214	7,272	23,544
Total	2,493,391	125,535	128,585	204,666

Table 4.4: Number of named entities in abstracts of SumeCzech's articles

Split	Percentage
Train	36.1%
Dev	35.4%
Test	35.7%
Test OOD	14.1%

Table 4.5: Percentage of headlines containing no named entity

approximately 26M named entities in texts, 1M in headlines, and 3M in abstracts. (We do not use abstracts in our experiments. We present the numbers of named entities in the abstracts for completeness only.) We show the detailed statistics in Table 4.2, Table 4.3, and Table 4.4. We also counted the number of headlines without any named entity. We show the statistic in Table 4.5. We published the annotations[2] to promote replication of results and to enable further research [Marek and Müller, 2021].

4.4 Experiments

The task we study is to create a one-sentence summary from the text of the article. The one-sentence summarization can be seen as the task of creating a headline for the article. We use five baselines introduced by Straka et al. [2018]. Moreover, we propose one extractive method – *Named Entity Density* and two abstractive approaches, *Seq2Seq* and *Seq2Seq–NER*, for text summarization.

4.4.1 Metrics

We used the $\text{ROUGE}_{\text{RAW}}$ metric for evaluation. $\text{ROUGE}_{\text{RAW}}$ was proposed by Straka et al. [2018] as a language-agnostic variant of ROUGE [Lin, 2004]. The original ROUGE metric automatically determines the quality of the generated summary by comparing it to a reference summary created by humans. There are two variants. ROUGE-N measures the overlap of N-grams between the generated and reference summaries. ROUGE-L looks at the longest common subsequence between the reference and the generated summaries. ROUGE calculates recall and is English-specific. It employs English stemmer, stop words, and synonyms.

$\text{ROUGE}_{\text{RAW}}$ does not need any stemmer, stop words, or synonyms, which makes it language-independent. It measures recall, precision and F1-score. It also has two variants, $\text{ROUGE}_{\text{RAW}}$-N and $\text{ROUGE}_{\text{RAW}}$-L, corresponding to the variants of the original ROUGE metric. We selected $\text{ROUGE}_{\text{RAW}}$-1, $\text{ROUGE}_{\text{RAW}}$-2, and $\text{ROUGE}_{\text{RAW}}$-L to evaluate approaches that we propose.

Additionally, we used the proposed metric ROUGE_{NE}. ROUGE_{NE} measures the overlap of named entities between the reference and the generated summaries. This metric

[2] `http://hdl.handle.net/11234/1-3505`

evaluates the model's ability to transfer named entities to the summary.

4.4.2 Baselines

We adopt the methods proposed by Straka et al. [2018] as a baseline. They propose four extractive methods and one abstractive method for SumeCzech's task to create a headline out of the text of the article:

- *First*: unsupervised extractive method. It returns the first sentence of the article.
- *Random*: unsupervised extractive method. It returns a random sentence from the article.
- *TextRank*: unsupervised extractive method. It selects the most important sentence of the article using the TextRank [Mihalcea and Tarau, 2004] algorithm.
- *clf-rf*: supervised extractive method. It selects the sentence that receives the highest score produced by the Random forest classifier. The classifier performs classification using vector representation of sentences. The vector representation consists of the sum of TF-IDF for each word normalized by the sentence length, length of the sentence, cohesion (distance from other sentences), the count of capitalized words in the sentence, the count of tokens that consist of digits, and the count of non-essential words that suggests the sentence relates to some other sentence.
- *t2t*: supervised abstractive method. It uses a neural machine translation model proposed by Vaswani et al. [2017] to generate a summary consisting of a novel sentence.

4.5 Results

We show the evaluation results in Table 4.6. First, we replicated the results of *First* and *Random* baselines reported in Straka et al. [2018]. Our results were on par with the reported Precision, Recall, and F1-Score of $\text{ROUGE}_{\text{Raw}}$-1, $\text{ROUGE}_{\text{Raw}}$-2, $\text{ROUGE}_{\text{Raw}}$-L. We additionally evaluated our proposed metric ROUGE_{NE} for comparison with other methods.

Next, we evaluated the proposed extractive method *Named Entity Density* (NE Density). Results of *Named Entity Density* compared to the *First*, a solid summarization baseline, especially in the news articles domain, are encouraging. *Named Entity Density* achieves only slightly worse results. Moreover, the results are consistent between the test and the OOD test set. Additionally, as we will show in subsection 4.5.1, *Named Entity Density* produces summaries resembling the style of informationally concise reports.

We evaluated the *Seq2Seq* and *Seq2Seq–NER* on the test set to compare those methods with the approaches proposed by Straka et al. [2018]. We can see that our proposed *Seq2Seq* and *Seq2Seq–NER* methods achieve better results on average by 80% relatively in Precision and F1-score compared to the best methods proposed by Straka et al. [2018]. Only *Textrank*

Dataset	Method	ROUGE$_{\text{Raw}}$-1			ROUGE$_{\text{Raw}}$-2			ROUGE$_{\text{Raw}}$-L			ROUGE$_{\text{NE}}$		
		P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Test	First	7.8	14.6	9.4	1.1	***2.3***	1.5	6.7	12.6	8.1	2.4	2.7	2.4
	Random	6.2	11.0	7.3	0.5	0.9	0.6	5.4	9.5	6.3	1.8	2.1	1.8
	Textrank	6.0	***16.5***	8.3	0.8	0.6	0.7	5.0	***13.8***	6.9	-	-	-
	Tensor2Tensor	8.8	7.0	7.5	0.8	0.6	0.7	8.1	6.5	7.0	-	-	-
	NE Density	6.6	10.7	7.3	0.8	1.4	0.9	5.9	9.4	6.4	1.5	2.2	1.6
	Seq2Seq	16.1	14.1	14.6	***2.5***	2.1	***2.2***	14.6	12.8	13.2	***5.3***	***6.5***	***5.6***
	Seq2Seq–NER	***16.2***	14.1	***14.7***	***2.5***	2.1	***2.2***	***14.7***	12.8	***13.3***	4.7	6.0	5.0
OOD	First	7.0	14.7	8.7	1.4	***2.9***	1.7	6.1	***12.8***	7.6	***1.4***	1.7	***1.4***
	Random	5.5	10.9	6.6	0.7	1.4	0.8	4.8	9.5	5.8	0.9	1.3	1.0
	Textrank	5.8	***16.9***	8.1	1.1	3.4	1.5	5.0	14.5	6.9	-	-	-
	Tensor2Tensor	6.3	5.1	5.5	0.5	0.4	0.4	5.9	4.8	4.8	-	-	-
	NE Density	6.3	11.4	7.1	1.3	2.3	1.4	5.7	10.2	6.3	1.0	***1.9***	1.1
	Seq2Seq	13.1	11.8	12.0	***2.0***	1.7	***1.8***	12.1	11.0	11.2	1.0	1.0	1.0
	Seq2Seq–NER	***13.7***	11.9	***12.4***	***2.0***	1.7	***1.8***	***12.6***	11.1	***11.4***	0.9	0.9	0.9

Table 4.6: Results of automatic evaluation

and *First* achieve better results in Recall. The *Seq2Seq–NER* achieved slightly better results than *Seq2Seq*, which proves NER labels' usefulness for summarization. However, it seems from the results of ROUGE$_{\text{NE}}$ that the better score is not caused by the improved performance of using entities in the summaries.

We evaluated the *Seq2Seq* and *Seq2Seq–NER* on the OOD test set to learn how models cope with news articles from a domain never seen during training. The results are encouraging. Even though they show a drop in absolute values of metrics between Test and OOD test sets, the trend is the same. *Seq2Seq* and *Seq2Seq–NER* methods achieve the best results of all compared methods in Precision and F1-score, and *Seq2Seq–NER* has slightly better results than *Seq2Seq*.

Finally, we look at the results in ROUGE$_{\text{NE}}$. We do not have results for *Textrank* and *Tensor2Tensor* because they were not reported in the work of Straka et al. [2018], and we did not implement the methods ourselves. However, it is clear from the rest of the results that even the recent state-of-the-art methods are struggling with the named entities in the summarization.

4.5.1 Examples

We choose a few representative examples from the test and OOD test sets to show how different methods summarize. We also provide English translation for convenience. Only very simple automatic post-processing was done on the output of the proposed *Seq2Seq* and *Seq2Seq–NER* models. We filtered the start of the sentence and end of the sentence symbols, removed spaces before punctuation, stripped the text of any starting or ending space, and capitalized the first letter.

First, we present examples of summarization created by *Named Entity Density* in Ta-

Method	Headline
Gold	Maloobchod v srpnu výrazně rostl *Retail trade grew significantly in August*
First	Po očištění o sezónní a kalendářní vlivy rostl maloobchod meziročně o 4,2 procenta. *After adjusting for seasonal and calendar effects, retail trade grew by 4.2 percent year on year.*
NED	Podle Eurostatu vzrostly meziročně kalendářně očištěné maloobchodní tržby v celé Evropské unii o 2,2 procenta. *According to Eurostat, calendar-adjusted retail sales rose by 2.2 percent year on year across the European Union.*
Gold	Snoubenci zestárli, přibývá levnějších obřadů bez svatebčanů *The couple is getting old, there are more and more cheaper ceremonies without wedding guests*
First	Stoupá počet sňatků bez svatebčanů, ve všední den, jen za přítomnosti svědků. *The number of marriages without wedding guests is increasing, on weekdays, only in the presence of witnesses.*
NED	Centrum metropole bude stále patřit k nejžádanějším místům pro oddávání, potvrdila Právu mluvčí Prahy 1 Veronika Blažková. *The center of the metropolis will still be one of the most sought-after places for wedding, "Veronika Blažková, spokeswoman for Prague 1, confirmed to Právo.*
Gold	Vranovskou přehradu znovu znečistila ropa, unikala ze sudů na dně *The Vranov dam was again polluted by oil, escaping from barrels at the bottom*
First	Likvidace probíhá za odborné spolupráce pracovníků povodí Moravy a odboru životního prostředí. *The liquidation takes place with the professional cooperation of the employees of the Moravia River Basin and the Department of the Environment.*
NED	Starosta Vranova nad Dyjí se o ropě dozvěděl z tisku, což jej rozlítilo. *The mayor of Vranov nad Dyjí learned about the oil from the press, which angered him.*
Gold	Z Fondové bude Reaganova žena, doplní ji Oprah a Poslední skotský král *The Fond will be Reagan's wife, complemented by Oprah and the Last King of Scotland*
First	Ve snímku s názvem The Butler (Majordomus) o správci v Bílém domě pracujícím pro několik amerických prezidentů by se v hlavní roli mohl podle časopsisu (the word *časopsisu* is misspelled in the dataset) Variety objevit americký herec Forest Whitaker. *According to Variety magazine, American actor Forest Whitaker could star in the film The Butler about a White House caretaker working for several US presidents.*
NED	Amerického prezidenta Richarda Nixona si zřejmě zahraje John Cusak. *US President Richard Nixon is likely to be played by John Cusack.*
Gold	Zlatého ledňáčka na festivalu Finále Plzeň získal snímek Jako nikdy *Movie Jako nikdy won Golden Kingfisher at the Finale Pilsen festival*
First	Letošní ročník festivalu Finále byl výjimečný tím, že poprvé soutěžily kromě českých také slovenské snímky. *This year's Finale festival was exceptional in that, for the first time, in addition to Czech, Slovak films also competed*
NED	Letošní ročník festivalu Finále Plzeň navštívilo od 27. dubna do 3. května 10 853 diváků. *This year's Finale Plzeň festival was visited by 10,853 spectators from April 27 to May 3.*

Table 4.7: Examples of summarizations created by *Named Entity Density*

ble 4.7. We do not divide the examples into test and OOD test sets because the generated summaries of both sets achieve comparative quality thanks to the fact that *Named Entity Density* is an unsupervised method.

We can see that the selected sentences contain named entities. Those sentences are comprised of factual information. The sentences are always grammatically correct, thanks to the fact that *Named Entity Density* is an extractive approach. Even though the summaries created by *First* can contain more entities in general, the summaries created by *Named Entity Density* have a higher density of entities. We can see that the summaries created by *Named Entity Density* revolve around *to whom*, *when*, *where*, and *what* happened. It closely resembles the style of reports that concisely identify and examine issues, events, or findings that have happened.

Notice also that the sentences selected by *Named Entity Density* are not the first sentences of the articles. We measured that the sentence selected as a summary by *Named Entity Density* differs from the sentence selected by *First* in 93% of SumeCzech's articles. Thus, we can use the summaries created by *Named Entity Density* as an alternative version or reformulation of summaries created by the method selecting the *First* sentence of an article. This property is highly praised by voice applications like Alquist [Pichl et al., 2020b] or Emora [Finch et al., 2020]. Voice applications present news articles in a summary because users quickly lose focus as news articles are not intended to be read by synthetic voices. Initially, the voice application can present the first sentence of the article. Additionally, if the user requests to learn more, it can present the summary produced by *Named Entity Density*.

We show the results of *Seq2Seq* and *Seq2Seq–NER* models for test and OOD test sets separately in Table 4.8 and Table 4.9. Both models generate novel sentences and incorporate entities into the generated summarizations successfully. Despite the promising results of the automatic evaluation, we can see that some of the outputs are not grammatically correct and contain repeated words.

4.6 Conclusion of Chapter

We proposed an extractive approach *Named Entity Density* that selects a sentence with the highest ratio between the number of entities and length of the sentence as the summary of the article. We selected the SumeCzech dataset Straka et al. [2018] for our experiments, which we annotated by named entities by the SpaCy's NER. We proposed a new metric, $\mathrm{ROUGE}_{\mathrm{NE}}$, that measures the overlap of named entities between the true and generated summaries. The experiments showed that *Named Entity Density* achieved nearly as good results as baseline selecting the article's first sentence, which is a very hard baseline to beat, especially in news articles. Nevertheless, the summaries generated by *Named Entity Density* demonstrated that the selected sentences reflect the style of reports concisely identifying *to whom*, *when*, *where*, and *what* happened. We proposed combining *Named Entity Density* and *First* summaries in voice applications. The voice application can initially present the first sentence of the article and continue by follow-up created by

Method	Headline
Gold	Nejznámější Albánec může o stavbě mešity přemýšlet ve vězení *The most famous Albanian can think about building a mosque in prison*
Seq2Seq	Soud potrestal za únos s lidmi *The court punished for kidnapping with people*
Seq2Seq–NER	Soud potvrdil tresty za pašování drog *The court upheld the penalties for drug smuggling*
Gold	Kriminalisté dopadli násilníka, který v lednu zneužil školáky z Orlové *Criminal investigators caught a rapist who abused schoolchildren from Orlová in January*
Seq2Seq	Policie hledá muže, který v Ostravě znásilnil děti *Police are looking for a man who raped children in Ostrava*
Seq2Seq–NER	Policie hledá muže, který se v Ostravě, který se na něj *Police are looking for a man in Ostrava who at him*
Gold	Do Valtického Podzemí za divadlem místo vína *To the Valtice Underground for the theater instead of wine*
Seq2Seq	Divadlo se v Brně otevře v Brně *The theater in Brno will open in Brno*
Seq2Seq–NER	V Brně otevřeli novou sezonu, divadlo se otevře návštěvníkům *They have opened a new season in Brno, the theater will be open to visitors*

Table 4.8: Examples of summarizations from the Test set

Named Entity Density if a user requests additional information.

Next, we proposed two abstractive approaches based on the Seq2Seq architecture. The first approach, *Seq2Seq*, generates novel summaries using only tokens of the article's text. The second approach, *Seq2Seq–NER*, additionally uses the named entity labels of each word as its input. Experiments showed that both proposed methods achieve better results than the methods proposed previously by Straka et al. [2018]. *Seq2Seq–NER* improved the results over *Seq2Seq* in automatic evaluation. This result demonstrated the usefulness of named entity labels for summarization. Furthermore, the results of the methods showed similar trends even on the out-of-domain test set.

The rationale for including summarization techniques in this work is their benefit for Hybrid dialogue management. The techniques serve as a connection between dialogue management and knowledge base. They provide dense and informationally rich summaries of knowledge stored as potentially long articles. Due to the limited size of neural response generators based on transformer architecture, it is advantageous to create summarizations of those long articles. Summarizations ground the response of neural response generators in knowledge, as described in chapter 6 proposing Hybrid dialogue management.

Method	**Headline**
Gold	Havlova Asanace by sama asanaci potřebovala *Havel's Asanace itself would need sanitation*
Seq2Seq	Havel se vrátil do divadla *Havel returned to the theater*
Seq2Seq–NER	Havel se s s Havlem Na zábradlí. Na hradě *Havel with with Havel at Na zábradlí. On Castle*
Gold	Hrad Bouzov nadchne cyklisty i zájemce o mučení a draky *Bouzov Castle will delight cyclists and those interested in torture and dragons*
Seq2Seq	Hrady a zámky na hrad. Kde se můžete vidět i na hrad *Castles and chateaux for the castle. Where you can yourself and castle*
SeqSseq–NER	Na kole na hrad *By bike to the castle*
Gold	Filmy z Indie opět v Praze *Films from India again in Prague*
Seq2Seq	V Indii se chystá na film o lásce *There are preparations for a movie about love in India*
Seq2Seq–NER	V Indii se vrací do Indie *In India, he returns to India*

Table 4.9: Examples of summarizations from the OOD test set

Chapter 5

Dialogue Management

This chapter introduces approaches to dialogue management. We can divide the task of dialogue management into two levels: topic tracking and flow control. The topic tracking and flow control differ in the frequency with which they make decisions and the size of the dialogue unit they select. The former is responsible for the strategic decision process, while the latter makes tactical decisions. Topic tracking is responsible for selecting the topic of the conversation that would take place in the following several turns of the dialogue. In practice, the topic tracking usually changes the topic if the user requests a change to a different topic or some application-dependent condition is met. The infrequency of the topic tracking decision process also implies that topic tracking makes the selection between large conversation units that take place for several turns. Thus, the topic tracking part of dialogue management decides with a frequency of several turns and selects a large unit of conversation. The immediate actions in each turn are selected by flow control.

The flow control part of dialogue management is responsible for action selection in every dialogue turn. It has to represent the immediate state of the dialogue in its internal state. Flow control selects the action and updates its internal state based on the user's message and previous state. The set of actions is usually determined by a dialogue script, which we call dialogue in this thesis. The system's response is constructed or selected based on the flow control output. Thus, the flow control level of dialogue management makes frequent decisions based on which immediate response of the system is dependent.

In this chapter, we present flow control and topic tracking methods. Important information for a reader is that this chapter serves as a prelude to the Hybrid dialogue management proposed in chapter 6.

5.1 State of the art

The following sections introduce state-of-the-art methods in topic tracking and flow control of dialogue management.

5.1.1 Topic Tracking

We can divide works in topic tracking into two groups. The first group of works focuses on the topic discovery and representation of the topic state. Those works are related to problems of natural language understanding. The second group focuses on the policy and topic selection by which the dialogue manager determines the optimal topic to discuss in a mixed- or system-initiative setting. The second group is relevant to our proposed methods. First, we will present relevant works on the topic discovery and representation of the topic state.

Kim et al. [2014] propose a composite kernel approach for dialogue topic tracking to utilize various types of domain knowledge from Wikipedia. Two kernels are defined based on history sequences, and context trees are constructed based on the extracted features. The proposed work is improved by a subsequent work of Kim et al. [2015] proposing to utilize Wikification-based features for providing mention-level correspondences to Wikipedia concepts for dialogue topic tracking.

Kim et al. [2016a] present various artificial neural network models for dialogue topic tracking, including convolutional neural networks to account for semantics at each utterance and recurrent neural networks to account for conversational contexts along multiple turns in the dialogue history.

Xu et al. [2021b] presents a topic-aware solution for multi-turn dialogue modelling, which segments and extracts topic-aware utterances in an unsupervised way so that the resulting model is capable of capturing salient topic shift at the discourse level in need and thus effectively tracks topic flow during multi-turn conversation. Topic-aware modelling is implemented by a newly proposed unsupervised topic-aware segmentation algorithm and Topic-Aware Dual-Attention Matching (TADAM) Network, which matches each topic segment with the response in a dual cross-attention way.

Williams et al. [2014] introduced the first Dialogue State Tracking Challenge (DSTC) focused on the dialogue state tracking of human-computer dialogues in the bus timetable domain. Next, there were DSTC 2 [Henderson et al., 2014a] focused on dialogue state tracking on human-computer dialogues in the restaurant information domain, DSTC 3 [Henderson et al., 2014b] focused on domain adaptation for human-computer dialogue state tracking from the restaurant to the tourist information domain, DSTC 4 [Kim et al., 2017] focused on dialogue state tracking on human-human dialogues in the tourist information domain and DSTC 5 [Kim et al., 2016b] focused on cross-language dialogue state tracking on human-human dialogues in the tourist information domain. With the sixth instalment of DSTC, the challenge's focus changed to different topics. DSTC challenges were substantial testing grounds fostering innovation in topic tracking.

Next, we will present related works in policy and topic selection that are especially relevant to our work. We start with the systems that won the Alexa Prize Socialbot Grand Challenge 1 [Ram et al., 2018], 2 [Khatri et al., 2018] and 3 [Gabriel et al., 2020] as they are the most closely influenced methods we propose.

Fang et al. [2017] proposed a Sounding Board. Sounding Board uses a hierarchically structured, state-based dialogue model, where the state includes a discrete set of interaction types, the result of the last personality quiz, and a memory of previously discussed content. Within the DM module is a master processing sequence that manages the conversation as a whole and a collection of miniskills that manage conversation segments for specific types of interactions, which they refer to as conversation modes. The hierarchical architecture simplifies the process of updating or adding new capabilities, and it is useful for handling the frequent high-level conversation mode changes that they observed in user interactions with the Sounding Board. At each turn, the DM module executes a sequence of processing steps to identify a response strategy that addresses the user intent and meets any constraints on the conversation topic. At the master processing level, the goal is to identify the conversation mode and the appropriate miniskill to respond to it. First, a state-independent processing step tries to identify cases that clearly initiate a new conversation segment, such as for an explicit topic request or other command types. If such cases are not found in the state-independent process, a second processing stage is used where state-dependent dialogue policies are executed. These processing stages poll mini skills to identify which can satisfy user intent and topic constraints. Miniskills with the most detailed topic match are prioritized, but otherwise, miniskills are selected randomly, trying to avoid the same miniskill for consecutive turns.

Chen et al. [2018] proposed Gunrock. Gunrock uses a two-level hierarchical dialogue manager to handle users' conversations. The high-level system selects the best topic dialogue module for each user request, leveraging the output from NLU. After that, the low-level system would activate this topic dialogue module to generate a response. The system identifies user intent based on NLU output and combines it with each sub-module feedback. The high-level dialogue manager decides which sub-module should handle the user utterance. The dialogue module selector first picks a topic dialogue module responsible for the topic intent detected by our intent classifier. In order to keep the consistency of responses, the selected module would provide a signal called propose_continue to the system after they generate a response. The system selects this module for the user's next utterance if it is set as CONTINUE. If it is set as UNCLEAR, the system selects this module only when it cannot detect other topics. When it is set as STOP, which means it cannot handle the user's further requests, the system does not select this module for the next turn. In this case, the module should propose a module that may better handle the request. Otherwise, the system would get a special template that proposes one topic dialogue module that has not been proposed or discussed and concatenates it to the module's response. The authors proposed tuning the modules' priority based on their performance. The selector selects the proposed module once the user adopts our proposal in the next turn. The dialogue module selector would deal with some explicit strong user intents specifically. Authors designed regular expression patterns to catch utterances with such intents and send them to the module responsible for the current topic. For example, if the utterance is *"Let's talk about movies,"* the dialogue module selector selects the movie module immediately.

Paranjape et al. [2020] propose a Chirpy Cardinal, an open-domain dialogue agent. The dialogue manager of the agent handles the high-level logic of tracking which topics it is discussing with the user and which responses or prompts should be used to form the bot's utterances. User has navigational intent when indicating they do or do not want to discuss a particular topic. The entity tracker tracks the current entity in the form of a Wikipedia article. The system relies on multiple response generators. The authors propose to use a priority system to decide which response generator's response should be selected on each turn. When generating responses, each response generator provides one of the response priorities. A hierarchy of priorities supports preserving conversational continuity while remaining responsive to the user's initiative. By avoiding a centralized response-choosing module, the proposed design allows response generators to decide whether they should respond and whether their response is high quality.

5.1.2 Flow Control

The first notable work which concerns flow control was proposed by Weizenbaum [1966] in the ELIZA chatbot. ELIZA uses scripts to describe the conversation. The script contains rules based on string matching. The system outputs a corresponding response if the rule is found in the input. There is also memory, which allows ELIZA to store found keywords and refer to them later in the conversation. Except for memory, ELIZA's flow control is concerned with producing immediate responses. Thus, its flow control is rather limited. Despite this fact, it was a pioneering work.

The subsequent two works are in the field of goal-based dialogue management. We include them in the related work because they propose innovation in flow control.

Rudnicky and Xu [1999] proposed Agenda dialogue management. The agenda-based approach addresses the problem of dialogue management in complex problem-solving tasks. It treats the task at hand as one of cooperatively constructing a complex data structure, a product, and uses this structure to guide the task. The product consists of a tree of handlers. Each handler encapsulates processing relevant to a particular schema.

Bohus and Rudnicky [2003] proposed RavenClaw. RavenClaw introduces a clear separation between task and discourse behaviour specification and allows rapid development of dialogue management components for spoken dialogue systems operating in complex, goal-oriented domains. The system development effort is focused entirely on the specification of the dialogue task, while the dialogue engine transparently generates a rich set of domain-independent conversational behaviours.

The next step in the evolution of flow control proposed Wallace [2003]. It was Artificial Intelligence Markup Language (AIML). AIML enables designers to input knowledge into chatbots based on the A.L.I.C.E free software technology. AIML describes a class of data objects called AIML objects and partially describes the behaviour of computer programs that process them. AIML objects are made up of units called topics and categories, which contain either parsed or unparsed data. Several dialogue systems were developed using AIML, like Mitsuku proposed by Worswick [2017]. Mitsuku is a five-times winner of the Loebner Prize [Mauldin, 1994].

Williams et al. [2017] proposed Hybrid Code Networks (HCNs), which combine an RNN with domain-specific knowledge encoded as software and system action templates. Compared to existing end-to-end approaches, HCNs considerably reduce the amount of training data required while retaining the key benefit of inferring a latent representation of the dialogue state. In addition, HCNs can be optimized with supervised learning, reinforcement learning, or a mixture of both. HCNs attain state-of-the-art performance on the bAbI dialogue dataset [Bordes et al., 2016] and outperform two commercially deployed customer-facing dialogue systems.

Vlasov et al. [2019] proposed Dialogue Transformers. It is a dialogue policy based on a transformer architecture, where the self-attention mechanism operates over the sequence of dialogue turns. Authors argue that this is a more appropriate architecture than an RNN due to the presence of interleaved topics in real-life conversations.

Bocklisch et al. [2017] proposed Rasa. Rasa is an open-source tool for building conversational software. It aims to make machine-learning-based dialogue management and language understanding accessible to non-specialist software developers. Rasa works with human-readable training data formats. Rasa requires a list of utterances annotated with intents and entities. These can be specified either as a JSON structure or in markdown format. The utterances and responses are connected into linear training dialogues, which Rasa calls stories. Rasa can also visualise a graph of training dialogues. A story graph is a directed graph with actions as nodes. Edges are labelled with the user utterances that occur in between the execution of two actions. The edge label is omitted if there is no user interaction between two consecutive actions. Each graph has an initial node called START and a terminal node called END. The graph does not capture the entire dialogue state, and not all possible walks along the edges necessarily occur in the training set. To simplify the visualisation, a heuristic is used to merge similar nodes.

Burtsev et al. [2018] proposed DeepPavlov tailored for developing conversational agents. The library prioritises efficiency, modularity, and extensibility to make it easier to develop dialogue systems from scratch and with limited data available. It supports modular as well as end-to-end approaches to the implementation of conversational agents. A conversational agent consists of skills, and every skill can be decomposed into components. It allows assembling a dialogue system from building blocks that implement models for required NLP functionality. These blocks can be recombined and reused in agents for different dialogue tasks. Such modularity opens possibilities for fast prototyping and knowledge transfer. The library supports the creation of multi-purpose agents with diverse skills. This is important for real-life application scenarios because skills can be added, upgraded or removed independently when a dialogue system is already deployed.

Similar systems for the creation of conversational AI on the level of flow control are Voiceflow[1], Alexa Skills Kit[2] and Google Dialogflow[3]. All three are commercial software as a service tools. They allow developers to create, test, modify and deploy dialogue flows using a visual designer tool. The flows are created by placing dialogue building blocks, representing messages and responses, and connecting them via transitions.

[1] `https://www.voiceflow.com/`

[2] `https://developer.amazon.com/en-US/alexa/alexa-skills-kit`

[3] `https://cloud.google.com/dialogflow`

5.2 Method

In this section, we will introduce methods for both levels of dialogue management, topic tracking and flow control. The methods were developed for, utilized and tested in subsequent versions of socialbot Alquist [Pichl et al., 2018, 2020b,a, Konrád et al., 2021]. Alquist competed in the Alexa Prize Socialbot Grandchallenges, achieving significant successes.

Division of dialogue management into two parts proved highly advantageous in open domain socialbot. It allowed us to develop a topic tracking component responsible for the strategic planning of the dialogue independently of the individual dialogue units executed by the flow control component. The independent development opened doors for the rapid development of conversational topics, which turned out to be a powerful strategy, especially in initial instalments of Alexa Prize Socialbot Grandchallenges.

5.2.1 Topic Tracking

Topic tracking is responsible for the strategic planning of the dialogue. Based on the context of the conversation and the current state of the dialogue manager, it selects the optimal topic of a conversation. The topic is usually represented as one or several units of dialogue realized by flow control dialogue management.

Throughout Alexa Prize Socialbot Grand challenges, we developed three methods of topic tracking for Alquist. We sort methods in the order in which they were proposed, the growing complexity of the decision process and the number of dialogue units and topics it allowed to handle. We call the first method Monolithic topics dialogue management. The name already suggests that this method relied on a small number of topics realized as large dialogues. The following proposed method, Topic graph dialogue management, introduced an ordered graph of topics, which the dialogue manager uses to traverse from specific to increasingly more general topics of conversation. Finally, we propose Dialogue selector. Dialogue selector assigns topic tags to dialogues between which it transitions with the objective of minimizing the distance between topic tags. The following subsections introduce our three methods of topic tracking in greater detail.

5.2.1.1 Monolithic Topic Dialogue Management

The Monolithic topic dialogue management is the baseline method for topic tracking we proposed. This method can, in theory, support any number of topics. However, the reasonable limit is units or tens of topics in practice. Consequently, most of the agency in the conversation is left to the flow control dialogue management.

Monolithic topic dialogue manager selects topic t based on the intent of recent user input i and inner state of dialogue manager s

$$t = MTDM(i, s),$$

where t is an element of a set of topics T and $MTDM$ is a function of the dialogue manager according to which it selects the topic. The function $MTDM$ relies on $1 : m$

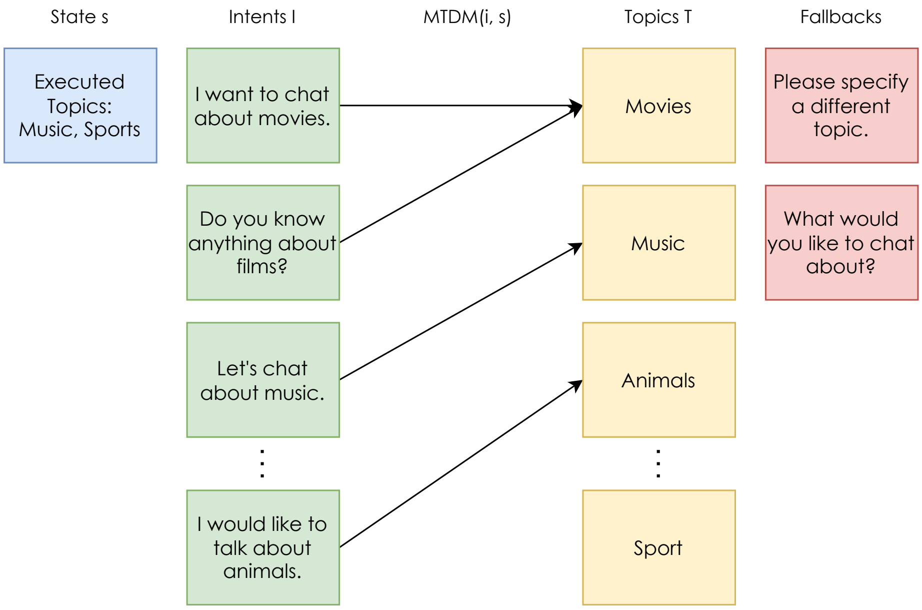


Figure 5.1: Schema of Monolithic topic dialogue manager

mapping, in which m intents are mapped to a topic. We illustrate the Monolithic topic dialogue management using Figure 5.1.

Demonstrated by practical example, the intents recognised in inputs *I want to chat about movies* and *Do you know anything about films?* are mapped to topic *Movies*. The execution of conversation about movies is directed to a corresponding dialogue unit of flow control. Additionally, the dialogue manager keeps track of topics that have been selected using its inner state s. Suppose the topic t would be selected for the second time. In that case, the dialogue manager selects a fallback dialogue, which asks the user to select a different topic to avoid repetition in the conversation. Moreover, the system can select dialogue directly, asking what topic the user prefers next if no intent with mapping to any topic is detected.

We tested Monolithic topic dialogue manager for topic tracking in Alquist 1.0 [Pichl et al., 2018] during Alexa Prize Socialbot Grand Challenge 1. We applied it with Structured topic dialogues (see section 5.2.2.1) for flow control. The main advantage of this approach is its simplicity. The reason is that the dialogue manager makes infrequent decisions and selects out of a small pool of possible topics. However, this approach in topic tracking moves most of the conversation's decision-making to flow control. Consequently, the dialogues executed by flow control must be complex, limiting the modularity of the whole system. Moreover, the limited modularity can lead to a rigidity of dialogue flows. Those weaknesses are addressed in Topic graph dialogue manager.

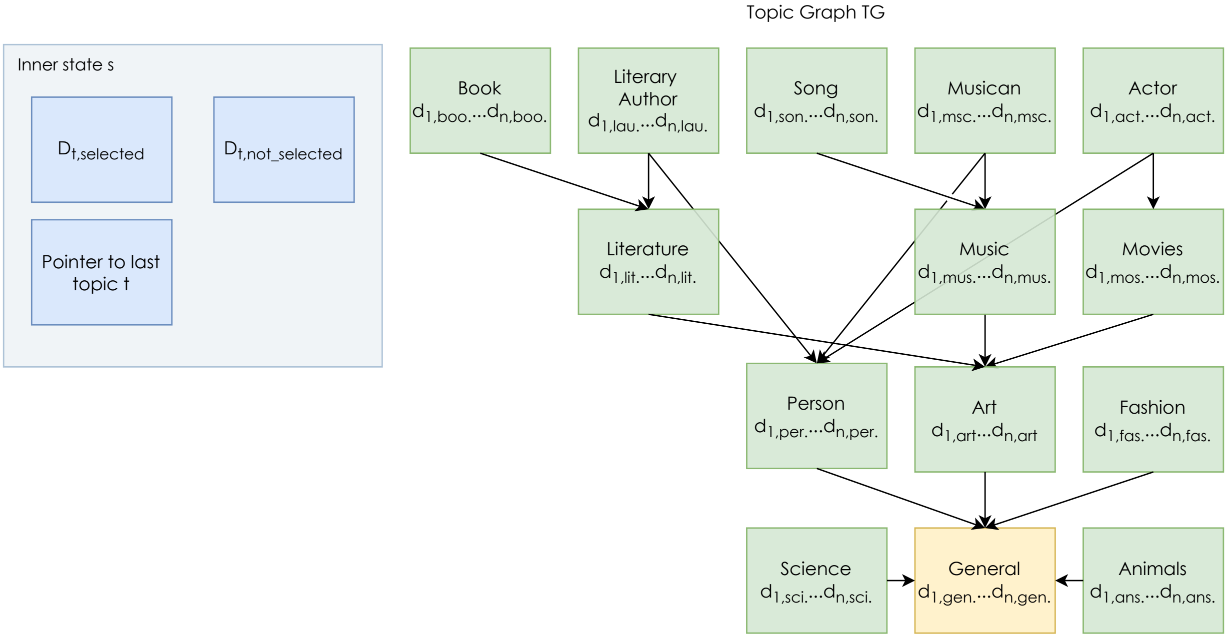


Figure 5.2: Schema of Topic graph dialogue manager. The Topic graph dialogue manager consists of the inner state s and Topic graph TG. The Topic graph consists of topic nodes. Each topic node contains a set of dialogues $d_{1,t}...d_{n,t}$. There is a root topic node we call General. The root node contains dialogues which apply to entities of unrecognizable topics. We simplified the structure of the Topic graph in the figure for clarity.

5.2.1.2 Topic Graph Dialogue Manager

Topic graph dialogue manager addresses the rigidity of dialogue flows of Monolithic topic dialogue manager. Topic graph is the core principle of Topic graph dialogue manager. The Topic graph is a graph structure that groups dialogues into topics and connects topics into broader topics. This structure allows the Topic graph dialogue manager to chain dialogues of the same topics. Moreover, the dialogue manager can transition to dialogues from more general but still related topics once the dialogues of a certain topic are depleted. The approach based on the Topic graph allows for a modular development of dialogues executed by a flow control dialogue management as a consequence.

Formally, the Topic graph is an oriented graph TG. TG consists of nodes V and edges E

$$TG = (V, E).$$

Each node $v \in V$ represents topic t out of a set of all supported topics T. The topic t contains dialogues D_t of the topic t.

$$D_t = d_{1,t}, d_{2,t}, ..., d_{n,t}$$

The edges E are oriented. Edge connects two topic nodes according to the relative generality of the nodes. If the edge $e_i \in E$ starts in node $v_j \in V$ and ends in node $v_k \in V$

it means that topic t_j of node v_j is less general than topic t_k of node v_k. To demonstrate the relation by an example, the topic t_j of node v_j can be *literary author* and topic t_k of node v_k can be *literature*.

The Topic graph dialogue manager is a function $TGDM(i, e, s, TG)$, where i and e are the intent and entity extracted out of a user's message, s is the inner state of the dialogue manager, and TG is the Topic graph. The inner state s contains two non-overlapping sets of dialogues. One set contains dialogues that have been selected already $D_{t,selected}$ and dialogues that have not been selected yet $D_{t,not\_selected}$. Moreover, it contains a pointer to the node v of topic t in the Topic graph TG, out of which the last dialogue was selected. The result of the function $TGDM(i, e, s, TG)$ is one dialogue which a flow control dialogue manager will execute.

The procedure in which the function $TGDM(i, e, s, TG)$ selects the dialogue consists of the following steps. First, the Topic graph dialogue manager tries to find topic node v according to its inner intent mapping. If there is topic node v to which the intent i is mapped, the Topic graph dialogue manager selects one of the dialogues d connected to node v and present in $D_{t,not\_selected}$. Also, the pointer in the inner state is updated to point to v and selected dialogue d is moved out of $D_{t,not\_selected}$ to $D_{t,selected}$. Alternatively, the topic node v can also be selected similarly by mapping entity types to topic nodes. Next, if the last selected dialogue ends and Topic graph dialogue manager has to select the following dialogue, it first tries to select one of the dialogues out of topic node v to which points the pointer stored in the inner state s. The selection is limited to set $D_{t,not\_selected}$. If the set $D_{t,not\_selected}$ is empty, the dialogue manager takes a transition out of topic node v through oriented edge of Topic graph TG to topic node v_k with more general topic t_k. The Dialogue selector tries to select dialogue connected to v_k and not present in $D_{t,selected}$. Without such dialogue, the dialogue manager transitions to an even more general topic. The procedure repeats until some dialogue is selected. If no edge leads to a more general topic, the dialogue manager has to execute a procedure to select a new topic node. Such a procedure can be a simple random function, explicitly asking users for their preferences or some recommendation algorithm.

We illustrate the Topic graph dialogue manager in Figure 5.2. Moreover, to demonstrate Topic graph dialogue manager using an example, imagine that the user explicitly asks to talk about literary author J. R. R. Tolkien, who is recognized by the system as an entity of type *literary author*. The dialogue manager finds mapping from entity type literary author to topic node *literary author*. This topic node contains dialogues that can lead a conversation about any literary author. Those dialogues can be, for example: *"How do you like the work of this author?"*, *"What is your favourite book of this author?"* or *"How many books written by this author have you read?"* One of the dialogues which has not been selected yet is selected. Once the user goes through all the dialogues from *literary author* topic, the Dialogue selector can transition to a more general topic *literature*. This topic contains dialogues that talk about literature in general: *"What genre is your favourite?"*, *"Where do you read books?"*, *"How do you pick a book to read,"* etc. Once those dialogues are all selected, the Dialogue selector can move to a more general topic of *art* or *entertainment*.

We used Topic graph dialogue manager for topic tracking in Alquist 2.0 [Pichl et al., 2020b] competing in Alexa Prize Socialbot Grand Challenge 2. We used it in Alquist 2.0 together with Dialogue trees (see section 5.2.2.2). We observed two main benefits of Topic graph dialogue manager. The first is that thanks to the ability of the dialogue manager to transition from more to less specific topic nodes, the conversation can naturally progress for a longer time without significant or abrupt changes to the topic. Second, the dialogue management system is more modular due to separating topics and dialogues. This fact is advantageous because it eases the development of dialogues and can introduce more variable conversation flows. The variability stems from the selection process of individual dialogues connected to topic nodes because there is no strict order in which the dialogues are selected. However, the Topic graph still induces one topic for one dialogue. The system does not allow multiple topics to be assigned to any dialogue. This property of the Topic graph dialogue manager leaves room for improvement.

5.2.1.3 Dialogue Selector

Dialogue selector is a dialogue manager for topic tracking inspired by content-based filtering recommendation systems [Pazzani and Billsus, 2007]. The Dialogue selector improves the idea behind the Topic graph dialogue manager. Instead of fixed mapping of dialogue to a single topic enforced by a structure of Topic graph, the Dialogue selector uses tags. There can be multiple tags assigned to each dialogue. The tag represents the topic of the dialogue. Tags can have several layers of granularity. There can be tags describing a whole topic like *movies*, *music* or *animals*. Next, there can be a more specific layer of tags like *action movies*, *romantic movies* or *western movies*. Next, there can be even more specific layer of tags like *plot of action movies*, *actors in action movies* or *directors of action movies*. Thus, there can be a dialogue asking users *"Do you eat popcorn when you watch action movies?"* with assigned tags *food*, *snacks*, *popcorn*, *movies* and *action movies*.

In addition to tags, each dialogue has a starting condition. The starting condition specifies what conditions must be met for the dialogue to be considered for selection. The starting condition is a logical expression with either true or false result. If the result is true, dialogue can be selected by a Dialogue selector. If the result of a logical expression is false, the dialogue can not be selected. The input to the starting condition is a user profile and the inner state of the Dialogue selector. The starting condition can require a certain value stored in the user profile or state. The starting condition is crucial in ensuring that the Dialogue selector will avoid selecting a dialogue for which execution any information is missing. For example, for the dialogue *"Do you eat popcorn when you watch action movies?"* the starting condition can require the fact *"user likes action movies"* to be true and the fact that the dialogue was not executed in the conversation's history to be true.

The principle behind the selection process of Dialogue selector is to minimize the change of topics between two selected dialogues while keeping the flexibility of possible selections as high as possible. Thus, the Dialogue selector selects one of the dialogue candidates based on the overlap between the tags of the last selected dialogue and the dialogue candidates. This principle ensures a high degree of freedom in selection with little change in the dialogue

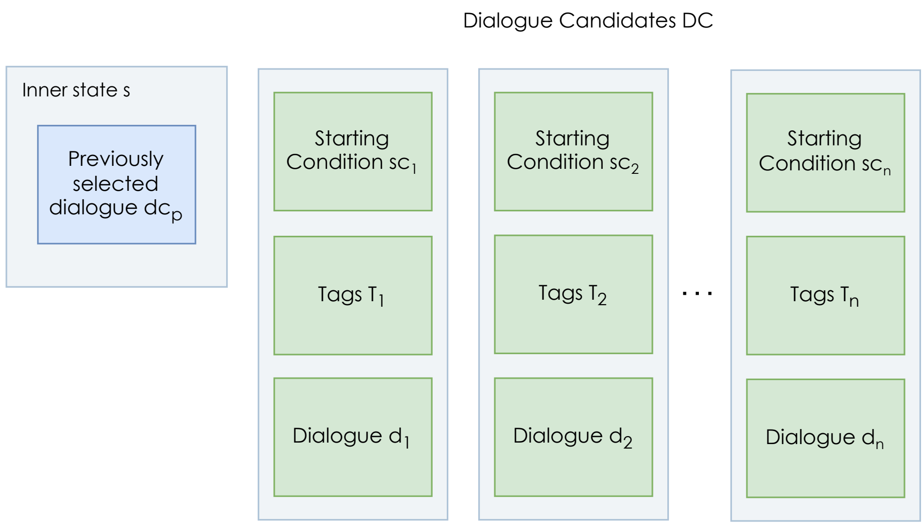


Figure 5.3: Schema of Dialogue selector

topic. Thanks to this fact, the topic of the dialogue evolves smoothly. Moreover, only dialogue candidates for which their starting condition is satisfied are considered candidates to be selected. This fact assures that only dialogues meeting the necessary conditions can be executed, and the dialogue does not fail.

Formally, the Dialogue selector is a function $DS(i, e, s, DC)$, where i is the intent of the user message, e are entity types recognized in user message, s is the inner state and DC is the set of dialogues that are candidates for selection. The dialogue candidate is formally a triple

$$dc_n = (sc_n, T_n, d_n),$$

where sc_n is a starting condition in the form of logical expression, T_n is a set of tags $(t_{n1}, t_{n2}, ..., t_{nm})$ assigned to the dialogue candidate, and d_n is the dialogue itself which flow control dialogue manager will execute in case of selection of dialogue candidate dc_n. We illustrate the Dialogue selector in Figure 5.3.

The selection process of dialogue by Dialogue selector goes as follows. First, the Dialogue selector evaluates the starting condition of all dialogue candidates $dc_1, dc_2, ..., dc_n$. Only the dialogue candidates whose starting condition holds are considered for the next selection step. We create a set of dc' that consists of all dc_n for which the starting condition holds.

Next, we compare the tag overlap between tags of previously selected dialogue candidate $dc_p = (sc_p, T_p, d_p)$ kept in the inner state s and tags of all dialogue candidates of dc'_n. For each dc'_n, we calculate the tag overlap score os

$$os_n = \frac{|T'_n \cap T_p|}{\max(|T'_n|, |T_p|)},$$

where $|T'_n \cap T_p|$ is a number of identical tags that are assigned both to dc'_n and dc_p, and $\max(|T'_n|, |T_p|)$ is a normalization factor equal to the number of tags in the larger of two sets of tags T'_n and $T_p|$. Finally, the Dialogue selector selects the candidate dialogue dc'_s with maximal overlap score os_s

$$dc'_s = \arg\max_s(os_1, os_2, ..., os_n).$$

The selection process requires the previous dialogue with assigned tags in order to calculate the overlap score and select the following dialogue. However, the previously selected dialogue is not present at the start of the conversation. In this case, the baseline approach would be randomly selecting some dialogue. However, a better approach is to start a dialogue by asking users what topic they are interested in discussing. The topic tags are retrieved from the user's response either through entity recognition or mapping intent to the tags. Those retrieved tags are then used for the initial selection of dialogue by the Dialogue selector.

The user initiative to change the topic of the dialogue is similarly handled in Dialogue selector as in the Topic graph with a slight change. There is also mapping using intents and entities extracted from user messages. However, instead of topics, they are mapped to the sets of tags. The tags from mapping are used to calculate topic overlap scores instead of topic tags of previously selected dialogue.

We tested the Dialogue selector in a practical setting in Alquist 4.0 [Konrád et al., 2021] competing in Alexa Prize Socialbot Chrand Challenge 4 together with Dialogue trees (see section 5.2.2.2). The Alquist 4.0 conducted more dynamic conversation thanks to Dialogue selector.

5.2.2 Flow Control

Flow control dialogue management starts once the topic tracking dialogue manager selects the dialogue unit which should continue in the conversation. The flow control dialogue manager is responsible for tactical decisions in the dialogue. This fact means that the flow control dialogue manager is responsible for selecting actions based on the immediate user input, the context of the dialogue and its inner state. In most cases, the action is an immediate response presented to the user. However, it can also be a call to an external service or knowledge base. Flow control dialogue manager uses their response to select the next action. Thus, the flow control dialogue manager decides what will happen in the dialogue on a turn-by-turn basis.

We developed two approaches to flow control for socialbot Alquist throughout Alexa Prize competitions. The first approach uses Structured topic dialogues [Pichl et al., 2018]. The second approach utilizes Dialogue trees [Pichl et al., 2020b].

5.2.2.1 Structured Topic Dialogues

Structured topic dialogue is a rule-based system inspired by state automata with an extension of intents. The dialogue is represented as a graph structure consisting of nodes called states and edges called transitions. The state consists of a function represented by a programmatic code, which can access user input, modify the inner state of the dialogue manager, make requests to external services and knowledge base and construct a response. The mandatory result of the state's function is to select a transition to following state. There are two types of transition: the immediate transition and the transition with user input. The first type of transition executes the programmatic code of the following state immediately. The second type of transition presents the constructed response to the user first, lets the user input a message and only after this executes the programmatic code of the following state. Thus, the second type of transition ends the dialogue turn. The Structured topic dialogue handles user initiative through intents. If the supported intent is recognized, the Structured topic dialogue switch the execution of the conversation to the state according to its inner mapping. There is one starting state in which the Structured topic dialogue begins and multiple ending states. Once any ending state is reached, the dialogue processing is transferred from the flow control level of the dialogue manager back to the topic tracking level of the dialogue manager, which selects the following Structured topic dialogue.

Formally, the Structured topic dialogue S is a graph consisting of nodes $N = n_1, n_2, ..., n_i$, oriented edges $E = e_1, e_2, ..., e_k$, intents $I = i_1, i_2, ..., i_j$, one starting node $n_s \in N$ and set of terminating nodes $T \subseteq N$. Each node n_x is implemented as a function taking the inner state of dialogue manager s_t and user input i_t. There is partially finished response r_t in the inner state s_t. The function modifies inner state s_t including the response r_t forming inner state s_{t+1} with response r_{t+1}, and outputs one of the edges $e_n \in E$ adjacent with the node n_x.

$$s_{t+1}, e_n = n_x(c_t, r_t)$$

The edge e is a tuple containing reference to the next node and an indication of whether the transition is immediate or whether the turn ends

$$e = (n_y, (\text{immediate/turn ends})).$$

The intent i_n contains the reference to the node

$$i_n = n_i.$$

If the intent i_n is recognized in the user's input, the Structured topic dialogue gives control to the node n_i.

There is a starting node n_s, in which the Structured topic dialogue starts, and a set of nodes $T = (n_{t1}, n_{t1}, ..., n_{tn})$ in which the Structured topic dialogue terminates.

In practice, the nodes in Structured topic dialogue are functions implemented using arbitrary programming language. We present a pseudocode in Listing 5.1. The function takes the inner state of the dialogue manager and user input as its arguments. The code can access NLU annotations like recognized intent, detected entities or sentiment analysis results. It can make API calls to external services, access knowledge bases, or do any other

```
class Edge:
    def __init__(self, next_node, transition):
        self.next_node = next_node
        self.transition = trasition

class Node1(State):
    def execute(context, user_input):
        context.addResponse("Hello, how are you?")
        return Edge(Node2, Transition.TURN_ENDS)

class Node2(Node):
    def execute(context, user_input):
        if "good" in user_input:
            return Edge(Node3, Transition.IMMEDIATE)
        if "bad" in user_input:
            return Edge(Node4, Transition.IMMEDIATE)

class StructuredTopicDialogue:
    def __init__(self):
        self.state = State()
        self.starting_node = Node1()
        self.terminating_nodes = [Node3, Node4]
        self.intents = {intent1: Node1, intent2: Node2}
        self.current_node = starting_node

    def process(self, state, user_input):
        intent = IntentRecognition().recognize(user_input)
        if intent in self.intents:
            self.current_node = self.intents[intent]
        while True:
            if current_node in terminating_nodes:
                exit()
            edge, state = self.current_node.execute(state,
                                 user_input)
            self.current_node = edge.next_node
            transition = edge.transition
            if transition == Transition.TURN_ENDS:
                break
```

Listing 5.1: Pseudocode of Structured topic dialogue in Python computation or data manipulation. Based on the previous results, it can store new data in the inner state and modify the response. The functions have to return a transition to the next node and indicate whether the turn ends. The intent extension in Structured topic dialogue is represented as a map with keys containing the intent and values containing node references. First, the Structured topic dialogue checks whether recognized intent is in the map at the beginning of a turn. If the intent is present in the map, the node from the transition is overwritten by the node to which the map points.

The advantage of Structured topic dialogues is its flexibility, thanks to the fact that nodes are represented by the programming code. This fact allows the node to do arbitrary computations and data manipulation. Thus, even the most complicated logic is possible in this approach. Moreover, there is a possibility to make a library of commonly used nodes for reusability. Also, the intent extension allows Structured topic dialogues to handle users with initiative.

However, the main disadvantage of this method for dialogue management is the requirement to code. Thus, the complicated logic possible in theory is feasible only for experienced programmers who can implement it. Moreover, the demand for proper code design increases with the complexity of the dialogue logic and the number of nodes. The possibility for errors and difficulty of maintenance increases, too. For those reasons, the Structured topic dialogues proved infeasible for typical dialogue designers with little programming experience, and it was challenging for programmers who needed significant time for development. We made this observation during the development of Alquist 1.0 [Pichl et al., 2018] for Alexa Prize Socialbot Grand Challenge 1, in which Structured topic dialogues were utilized for flow control dialogue management together with Monolithic topic dialogue management (see section 5.2.1.1) for topic tracking.

5.2.2.2 Dialogue Trees

Dialogue trees is a method of flow control dialogue management whose aim is to ease and standardise the development of dialogues compared to Structured topic dialogues. Dialogue trees model the conversation as a graph structure. The graph consists of nodes and edges. The nodes represent actions the dialogue manager performs. Instead of arbitrary code in states, Dialogue trees uses standardised nodes, which represent the dialogue's start and end, conversational AI's response, user inputs, local intents, global intents, and functions. Edges of the graph describe the dialogue flow. The flow of the dialogue is branched in a standardised way through intent nodes instead of return values of states in the form of transitions. Also, the dialogue turn ends once the flow reaches the user input node, thanks to which the dialogue designer does not have to specify a type of transition as in Structured topic dialogues. Thus, the Dialogue trees bring the development of dialogue flows using standardised building blocks, making the development easier and faster.

We demonstrate an example of a Dialogue tree in Figure 5.4. Naming the Dialogue tree a tree is misleading because the flow of the dialogue can merge two branches, or it can loop back to any previous node of the flow. Thus, it is a graph. However, its name was inspired by a gameplay mechanic used in video games for interaction with non-player

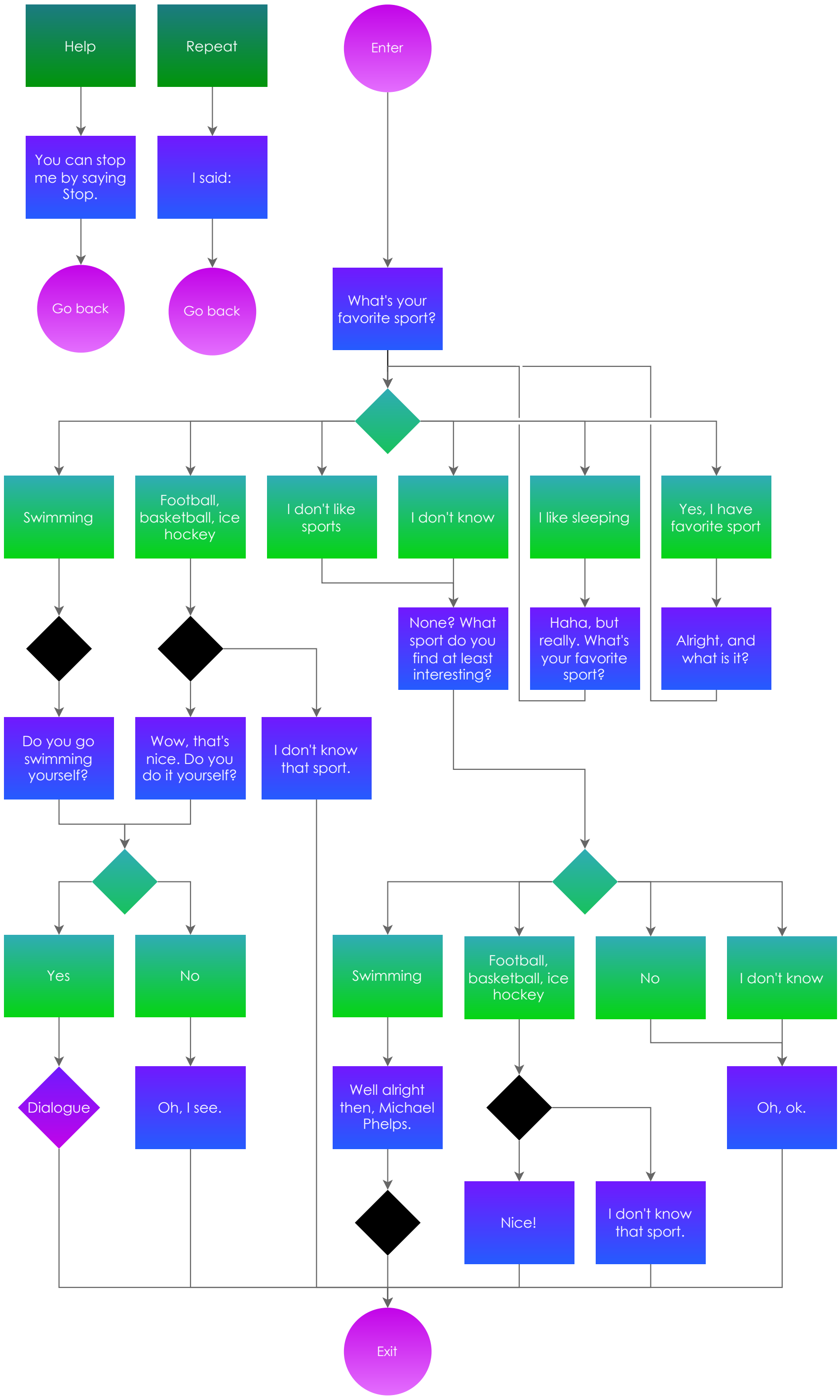


Figure 5.4: Example of Dialogue tree. Blue nodes represent system responses. Green rhombuses represent user inputs. Light green nodes represent local user intents. Black nodes represent function nodes. In this particular example, function nodes try to detect and save sports mentioned by users into their user profiles and branch dialogue according to the result. Finally, there is a purple rhombus representing subdialogue.

characters. [Adams, 2014]

The flow of the conversation starts in the node representing the start of the dialogue. Next, it follows the progression of edges through nodes. Every time the node is reached, the action it represents is performed. The response nodes add part of the response. Code in the function node is executed. User input starts the new turn by prompting the user to provide input. Corresponding local intents connected via edge to user input and global intents are recognized by the intent recognition component in the user's input.

The flow of the dialogue can be branched by user input based on the connected local intent nodes and all global intent nodes. Whereas local intents branch the dialogue on the local level, the global intents allow the dialogue flow to switch to a completely different part of the conversation (see section 2.9). Additionally, flow can be branched by function nodes based on the results of functions they contain. The flow is ended in one of the end nodes of the Dialogue tree.

Formally, the Dialogue tree consists of set of nodes $N = (n_1, n_2, ..., n_i)$ and oriented edges $E = (e_1, e_2, ..., e_j)$. Nodes are divided into types. There is a set of response nodes $N_r \in N$, user input nodes $N_{ui} \in N$, local intent nodes $N_{li} \in N$, global intent nodes $N_{gi} \in N$ and function nodes $N_f \in N$. There is also a set of ending nodes $N_t \in N$ and one starting node $n_s \in N$.

Edges have the following constrains: Edge starting in n_s has to end in $n_j \in (N_r \cup N_f)$. Edges starting in $n_{ri} \in N_r$ have to end in $n_j \in (N_r \cup N_{ui} \cup N_f \cup N_t)$. Only one edge can start in every $n_{ri} \in N_r$. Edges starting in $n_{uii} \in N_{ui}$ have to end in $n_j \in N_i$. Multiple edges can start in n_{uii}. Multiple edges can start in n_{uii}. Edges starting in $n_{lii} \in N_{li}$ have to end in $n_j \in (N_r \cup N_f \cup N_t)$. Only one edge can start in every $n_{lii} \in N_l i$. Edges starting in $N_{gii} \in N_r$ have to end in $n_j \in (N_r \cup N_f \cup N_t)$. There can be only one edge starting in every N_{gii}, and no edge ends in any N_{gii}. Edges starting in $n_{fi} \in N_f$ have to end in $n_j \in (N_r \cup N_f \cup N_u i \cup N_t)$. Multiple edges can start in n_{fi}. Moreover, there are no edges leading to n_s or originating in any $n_j \in N_t$.

Furthermore, we can introduce nesting, modularity and reusability into Dialogue trees through additional node representing other Dialogue trees. Thus, during the execution of the Dialogue tree, we can reach the node that directs the conversation flow to another Dialogue tree. We can call this nested Dialogue tree a subdialogue. The rules which apply to Dialogue trees apply to subdialogues as well. The flow of the subdialogue starts in the starting node and follows the dialogue flow according to the edges until it reaches one of its end nodes. At that time, the flow is returned to the original Dialogue tree, and the flow continues from the node representing the subdialogue. The described method allows us to create Dialogue trees out of standardized building blocks that can be reused in many situations. This modularity significantly eases the development of Dialogue trees.

The main advantage of Dialogue trees is that they can be designed using a visual editor (see Figure 5.5). Dialogue designers use visual editor in a drag-and-drop fashion. They drag nodes out of the editor's palette to the working canvas and configure the node for its purpose in the conversation. They write several intent examples into intent nodes, write responses into response nodes or write code into function nodes. Next, they connect the nodes by edges into the flow of the dialogue. The visual representation of the dialogue

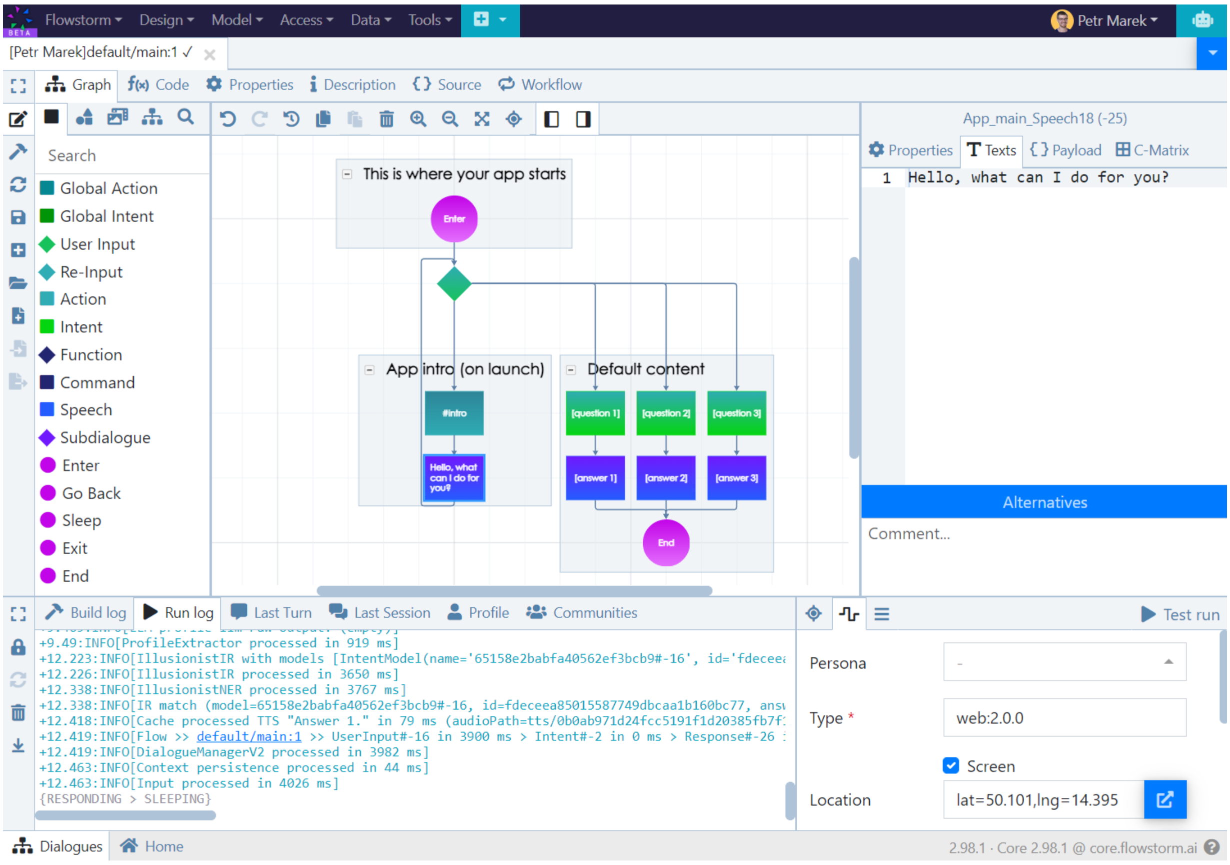


Figure 5.5: Visual drag-and-drop editor for Dialogue trees

makes for a straightforward interpretation of the dialogue. Although we can also develop a visual editor for Structured topic dialogues in theory, the requirement to code makes it far less ideal for a drag-and-drop visual editor than a pallet of standardized nodes of Dialogue trees. The Dialogue trees, together with drag and drop visual editor, were tested in the practice in Alexa Prize Grand Challenge 2, 3 and 4, where they proved highly effective.

5.3 Discussion and Results

This section compares proposed methods of topic tracking and flow control dialogue management. Nevertheless, as described in section 2.15, objective evaluation of conversational AI is problematic. This fact is amplified in the case of dialogue managers for open domain conversations, for which no suitable upstream evaluation has been proposed yet, according to our best knowledge. Although many other factors play an important role, the performance of the dialogue manager can be approximated by the performance of the whole conversational system using the downstream evaluation. This is because the dialogue manager plays a crucial part in the system's perceived quality. The quality of the conversation can be determined via human evaluation. Nevertheless, such evaluation tends to be highly subjective. Thus, collecting many ratings via crowd-sourcing is necessary for objectively evaluating conversational AI. Thus, we can use it as an approximation for the quality of dialogue managers.

For the comparison, we take advantage of the fact that all proposed systems for topic tracking and flow control were applied in subsequent versions of Alquist. Alquist competed in the Alexa Prize Socialbot Grand Challenges. The Alexa Prize Socialbot Grand Challenge used implicit and explicit metrics to evaluate socialbots collected via crowd-sourcing. The socialbots were made available to users of Amazon Alexa based in the United States. Users activated one randomly selected socialbot competing in the competition by the phrase *"Alexa, let's chat."* Users had a conversation, which they terminated as they pleased. The system measured the duration of the conversation and asked for the rating of the conversation on a scale from one to five stars. Hundreds of thousands of ratings were collected for each socialbot in the span of one instalment of grand challenge in this fashion. Average rating, median and 90th percentile of time were used as metrics. Moreover, during the finals, Amazon conducted a small-scale qualitative comparison of systems through human experts, which awarded the three best socialbots. We present the results in Table 5.1 and discuss them in the following paragraphs.

The first version of Alquist socialbot utilized Structured topic dialogues for flow control dialogue management and Monolithic topic for topic tracking dialogue management. While the combination of those two systems allowed for an independent development of conversational topics, the development of the topic itself was complicated. The reason was that the topic tracking dialogue management did not allow for any more fine-grained division of topics due to its limitation to traverse similar topics. Thus, the dialogue manager selected the Structured topic dialogue based on the intent recognized in user phrases like *"I would like to chat about movies"*, *"I want to chat about video games"*, or *"Let's chat about books"*. Next, the Structured topic dialogues handled the rest of the conversation. Consequently, the Structured topic dialogues had to be large and structurally intricate, which complicated development. Also, the Structured topic dialogues flows were rigid because they relied on the underlying graph of states with low flexibility. Nevertheless, the first version of Alquist was a second-place winner of Alexa Prize Socialbot Grand Challenge 1, with an average rating of 3.44, median time of conversation of 2:06 and 90th percentile of time of conversation equal to 8:24 during the week of Alexa Prize finals.

The second version of Alquist socialbot utilized a Topic graph for topic tracking and Dialogue trees for flow control. The Topic graph significantly increased the number of dialogues the system could handle and move between them smoothly. Dialogue trees amplified this progress. Our philosophy was to create many small dialogue flows, which a topic tracking dialogue manager would flexibly chain. As a consequence of this development, the perceived quality of the conversation improved thanks to a richer content of the conversation, as demonstrated by an improved average rating of 3.6, as well as time spent in the conversation. The median of time reached 2:13, and the 90th percentile of time grew to 12:49. The values were obtained during the week of the Alexa Prize finals. Again, the second version of Alquist was the second-place winner of Alexa Prize Socialbot Grand Challenge 2.

System	Topic Tracking	Flow Control	# Dialogues	Avg. Rating	Median Time	90th Percentile Time	Alexa Prize
Alquist 1.0	Monolithic topics	Str. topic dial.	16	3.44	2:06	8:24	2nd
Alquist 2.0	Topic graph	Dialogue trees	164	**3.6**	**2:13**	**12:49**	2nd
Alquist 3.0	Topic graph	Dialogue trees	164	3.59	2:06	12:31	3rd
Alquist 4.0	Dialogue selector	Dialogue trees	**273**	3.51	1:30	9:11	**1st**

Table 5.1: The final performance of Alquist socialbot in Alexa Prize Socialbot Grand Challenges and their corresponding flow control and topic tracking dialogue managers.

The third version of Alquist socialbot competing in Alexa Prize Grand Challenge 3 utilized the combination of Topic graph and Dialogue trees. The main reason for using the principally same components was the start of the COVID-19 pandemic, which complicated the system's development and slowed innovation due to the necessity for social isolation and remote work. The system achieved an average rating of 3.59, a median time of conversation of 2:06 and a 90th percentile time of conversation of 12:31 during the week of Alexa Prize finals. The Alquist 3.0 was the third-time winner of Alexa Prize Grand Challenge 3.

The last innovation was a combination of Dialogue trees with Dialogue selector introduced in the fourth version of Alquist socialbot. The Dialogue selector improved the flexibility of the Topic graph. While the Topic graph allowed for a transition between dialogues in the same topic, and once the topic was exhausted, it transitioned to dialogues of the parent topic, the Dialogue selector created a web of connections between dialogues. Thus, the number of possible transitions grew, and with it, the flexibility of the dialogue flows. Moreover, the number of dialogues that the system incorporated grew to 273, which can be attributed to the modularity the Dialogue selector enabled.

Despite the progress, the average rating of Alquist 4.0 was only 3.51. The median time was 1:30, and 90th Percentile time of conversation was 9:11. However, the Hu et al. [2021] stated in the overarching report of Alexa Prize Grand Challenge 4 that average rating of all competing socialbots was also 9.7% lower than at the end of Semifinals of Alexa Prize Grand Challenge 3. Hu et al. [2021] noticed the similar trend in 90th percentile of time, which decreased by 8.39%. This was the first instalment of the Alexa Prize Grand Challenge, in which average results fell compared to the preceding instalment. Hu et al. [2021] hypothesise that the drop can be attributed in part to rising expectations from users. Nevertheless, Alquist 4.0 succeeded in the expert jury by winning first place in the Alexa Prize Grand Challenge 4, which was the most significant achievement we reached throughout the Alexa Prize Socialbot Grand Challenges.

Chapter 6

Hybrid Dialogue Management

Rule-based systems and neural response generators based on large language models provide opposite and complementary advantages for conversational AI. Rule-based systems provide controllability, while language models excel in flexibility. However, there was little progress in the research of combining those approaches into a single system so that the advantages of both are utilized. In this chapter, we propose a Hybrid dialogue management, which combines the control of rule-based systems with the flexibility of neural response generators based on large language models.

Rule-based dialogue systems were a norm in conversational AI until recent years. Those systems operate according to the rules describing the system's behaviour. The rules can be in the form of simple mapping of users' messages to the system's responses [Weizenbaum, 1966] up to complex dialogue flows in the form of Dialogue trees that consider the context of the dialogue implicitly through its structure (see chapter 5). Domain expert or dialogue designer creates the rules by which the dialogue system operates. This fact ensures that the rule-based dialogue system's designers have control over its behaviour. They can influence how the system reacts to various user inputs, the style of responses, and the content of the response can be checked for untrue, explicit or otherwise harmful elements in advance. Most importantly, designers explicitly or implicitly specify a goal of the dialogue system in the conversation. Thus, rule-based systems allow designers to control their behaviour and specify its purpose.

A traditional approach to designing rule-based dialogue systems requires a domain expert to create the rules describing the system's behaviour in the form of code or some domain-specific language [Wallace, 2003]. This approach was streamlined thanks to progress in design tools for conversational AI [Bocklisch et al., 2017, Burtsev et al., 2018, Pichl et al., 2022]. However, even nowadays, rule-based dialogue systems still require dialogue designers to design the flow of the dialogue on the level of individual system's utterances and possible users' inputs represented by intents. The design process is a labour-intensive task which requires more effort with the growing complexity of conversations the system has to support.

Moreover, the system's behaviour and reaction must be specified during the design phase. Thus, the designers must anticipate users' behaviour through previous experience or initial system testing. Experiments like Wizard-of-Oz might provide helpful insight. However, those approaches are primarily viable for constrained applications. In open-domain conversational applications, the sheer amount of conversational situations makes it highly unlikely that designers would describe them all and specify satisfactory reactions. As a result, there will always be a golden path of the dialogue, which a system covers well, and out-of-domain inputs, which the system does not handle adequately. To mitigate this problem, the designers usually limit the initiative of users. The system keeps the initiative and direction of conversation through proposals and questions to keep users on the golden path of designed conversation. Consequently, users might perceive such systems as too invasive, directive and non-intelligent.

Over the last several years, there has been a boom of large language models [Radford et al., 2018, 2019, Brown et al., 2020] capable of generating high-quality texts. Those models were quickly adapted to the domain of conversations [Zhang et al., 2019, Roller et al., 2020, Adiwardana et al., 2020]. The large language models in this domain are also called neural response generators. In the domain of conversation, their task is to generate meaningful responses given the context of the dialogue. Generating responses in conversation is more complicated than a general generation of text because conversations bring unique challenges like competing goals of two speakers, multiple possible responses, changing of initiative, grounding or repairs. Nevertheless, neural response generators provide satisfactory results even in such complicated situations. Their most significant advantage is the flexibility with which they react to the never-observed conversation contexts. Moreover, they can adapt without any work needed from dialogue designers because neural response generators learn their behaviour out of training data. Thus, neural response generators can appear as an ideal solution for conversational applications.

However, the neural response generator's conversational style, strategy or goal is mostly unknown in advance. Predicting how neural response generators would react to various conversational situations is complicated. Moreover, neural response generators alone provide very little control over their behaviour on the level of an individual response, let alone the overall strategy and goal of the conversation. It is true that instruction-based models like ChatGPT [OpenAI, 2023b], Vikuna [Chiang et al., 2023] or Falcon [Almazrouei et al., 2023, Penedo et al., 2023] marked a progress in influencing the behaviour of large language models through prompts. However, they enable dialogue designers to influence the properties of individual responses or a few turns long section of a dialogue, which can hardly influence the overall strategy and goal of the whole system. Thus, neural response generators are applied only to open-domain chit-chat applications and are sparsely utilized in applications where some goal is specified.

Moreover, the state-of-the-art large language models are implemented as neural networks with tens or hundreds of billions of parameters. Such architectures demand significant computational resources for training as well as inference. The resource requirements increase the cost of training and operation of large language models, limit the usability of models in client and mobile devices, and hinder academic research due to the high cost of experiments. Also specific to conversational AI is the problem of high latency of large model architectures, which is problematic in real-time communication. Thus, any method limiting the computational requirements of large language models would make more applications viable.

Finally, many works point out that large language models tend to hallucinate [Maynez et al., 2020, Liu et al., 2023, Zhang et al., 2023, Mündler et al., 2023], produce toxic outputs [Gehman et al., 2020, Welbl et al., 2021, Shaikh et al., 2022, Deshpande et al., 2023] and generate otherwise malicious or biased responses [Motoki et al., 2023, Venkit et al., 2023, Dhingra et al., 2023]. Large language models generate facts that are not true [Pan et al., 2023, Zhou et al., 2023], overstate their abilities or mislead users. They also do it very convincingly, which is recognizable only by experts in the said domain [Else, 2023, Alkaissi and McFarlane, 2023]. The general public might be misled to believe them. This misleading can spread misinformation, which might cause harm in the real world. Thus, mitigating the hallucination of large language models is an active field of research. Even though neural response generators based on large language models can be vital in conversational AI, primarily thanks to the robustness and flexibility they introduce compared to rule-based systems, their adoption is limited due to the low controllability, high costs of operation and concerns about safety.

A solution might be a combination of the flexibility of neural response generators with the controllability and predictability of rule-based systems. However, the exact technique allowing us to combine both approaches was unclear. Although neural response generators can produce a response correctly reacting to user input, we do not want to completely give the management of the dialogue to the neural response generator. A neural response generator without any direction might completely miss the goal of the intended dialogue. Thus, we want a neural response generator to handle specific situations in the dialogue but return the flow back to the original flow of the dialogue outlined by a human designer to reach its goal.

Combining responses written by human designers and generated responses in a single conversational AI system opens two problems. First, it is preferable to make the generated response style similar to responses produced by a rule-based part of the system designed by the human designer. Second, suppose the conversation flow deviates from the rule-based system's designed flow. In that case, letting the neural response generator handle the dialogue for several dialogue turns is relatively straightforward. The hard part of this problem is to return the flow back to the original flow, which is designed to reach the intended goal of the dialogue. The ability to influence properties of generated response is an approach to mitigate both issues. With the right tools for influencing the properties of generated response (see section 6.2), we can unify the style of responses. Moreover, we can specify the properties of the response, which will connect back to the original dialogue flow so that it happens as smoothly as possible. The ability to influence dialogue act (see section 2.5) is vital for a smooth transition back to the original flow. Thanks to it, the neural response generator does not generate questions in an inappropriate moment, for example.

Hybrid dialogue management is the method we propose for combining a rule-based system with a neural response generator. Rule base system, preferably in the form of Dialogue trees (see section 5.2.2.2), can represent the flow of the dialogue on the pragmatic level. By pragmatic level, we mean that the flow of the conversation is divided into several steps, and each step describes what should happen during its execution. Each step is described by prompt, linguistic or stylistic properties of responses by tags or guides the model to the desired topic by introducing the appropriate context of a dialogue. Moreover, we can also specify what knowledge the generative model should utilize in each step. The neural response generator creates a response during the execution of the dialogue based on the specified properties of the step. It is also possible to combine generated responses in the flow with responses written by human designers and transition smoothly between both parts. Thus, in Hybrid dialogue management, a rule-based system provides a means of influencing the overall strategy of the conversation, while a neural response generator introduces robustness and flexibility to edge scenarios.

While Hybrid dialogue management can theoretically utilize prompt-based language models [OpenAI, 2023b, Chiang et al., 2023, Almazrouei et al., 2023, Taori et al., 2023], we propose a lightweight language model PraGPT. PraGPT is capable of a Pragmatic level of control. By Pragmatic level of control, we mean mechanism to influence the style and content of the generated response. A rule-based system injects the Pragmatic level of control into the model. PraGPT is several magnitudes smaller than stat-of-the-art neural response generators. Nevertheless, our experiments show that PraGPT achieves comparable results in human evaluation as models with significantly larger architecture in constrained conversational domains where Hybrid dialogue management is intended to be applied.

6.1 State of the art

We present state-of-the-art methods which influenced or are related to Hybrid dialogue management, and we describe the main differences with our proposed approach.

Keskar et al. [2019] propose CTRL, a 1.63 billion-parameter conditional transformer language model trained to condition on control codes that govern style, content, and task-specific behavior. They utilised features that naturally occur in texts, like URLs, domains or tasks, as a control mechanism. This allowed them to train the model in an unsupervised setting. They taught the model probabilities of texts prepended with control codes. They demonstrate that the model outputs change based on the provided control code.

Opposed to the work of Keskar et al. [2019], which proposes a general method for conditional text generation, our proposed method addresses conditional generation specifically in the domain of response generation in dialogue systems.

Hedayatnia et al. [2020] addressed the weakness of neural response generation approaches of not having explicit mechanisms to control the content or style of the generated response. This frequently results in uninformative utterances. They propose using a dialogue policy to plan the content and style of target responses as an action plan, which includes knowledge sentences related to the dialogue context, targeted dialogue acts or topic information. Their work results in a policy-driven neural response generator, which consists of a dialogue policy that determines the action plan based on the dialogue context and a response generation model that takes the action plan and the dialogue context as input to generate a response. They investigate different dialogue policy models to predict an action plan given the dialogue context. They demonstrate that a basic dialogue policy that operates at the sentence level generates better responses than a turn-level generation and baseline models with no action plan.

Similarly, Xu et al. [2018] propose managing the flow of human-machine interactions with the dialogue acts as policies. The policies and response generation are jointly learned from human-human conversations, and the former is further optimized with a reinforcement learning approach. They demonstrate that their proposed method achieves better responses and longer dialogue length than other state-of-the-art methods they were comparing.

Gao et al. [2023] propose CAB, a framework that takes a comprehensive perspective of cognition, affection and behaviour to generate empathetic responses. They propose to use appropriate dialogue acts to guide the dialogue generation to enhance its empathy expression. They propose to use eight categories of dialogue acts.

Opposed to works of Hedayatnia et al. [2020], Xu et al. [2018] and Gao et al. [2023], our proposed method targets the problem of combining rule-based Dialogue trees with a controllable generative model, which their works do not address.

6.2 Control Mechanisms of NRGs

Despite their ability to handle conversations on many different topics, neural response generators lack the intent of the conversation. The reason is that neural response generators are language models. Language models train to output the most probable sequence of tokens. Thus, the conversation flow conducted by the neural response generator alone follows the most probable path instead of aiming for a specified goal.

Neural response generators offer four approaches to influence their output. The first approach is finetuning. We can use training data, which contains a specific conversation style, knowledge, or the dialogue flow of the training conversations to achieve the desired goal. Second, there are tags. Tags are special input tokens that describe linguistic or stylistic properties of the response. Third, we can use the context of the dialogue or specific knowledge as an input to the model. The generative model then uses this input to generate the following conversation. Lastly, we can use prompts to direct the neural response generator.

6.2.1 Finetuning

Finetuning is a method that allows us to adapt a model trained to perform one task to perform a different task that is similar to the original task. The finetuning can be done using significantly less data than the original training.

In the domain of neural response generators, the model is usually trained on a large corpus of conversational data on which the model learns to generate texts or operate in the dialogue domain. Examples of such corpora are Reddit comments [Baumgartner et al., 2020] or Common Crawl[1]. The model learns to model basic properties of conversation using this corpus. However, the large corpus is usually moderated only basically. The Reddit comments have many offensive, non-conversational or generally poor-quality responses. The Common Crawl is not a conversational dataset. It is a corpus extracted from internet websites. It has the same problems commonly found on the internet, like falsehoods, grammatically wrong texts, or no longer true facts. Filtering all of those problems is an open field of research. Thus, a model can learn to model undesired properties of conversation.

Additionally, training large language models on large-scale corpora is an expensive process. The time to fully train the model might take several days of computational time. The training requires sizeable computational power and data storage capacity. For those reasons, the training process alone can cost millions of dollars with measurable environmental impact on CO2 emissions [Luccioni et al., 2022, Luccioni and Hernandez-Garcia, 2023]. Thus, training each model from the random weights would be impractical. Thankfully, many research teams share pre-trained language models on large corpora [Wolf et al., 2019]. We can use those models as a starting point for finetuning.

The finetuning is performed using significantly less data specific to the dialogues we would like to model. Since the model is trained on large corpora, it can generate texts or operate in dialogue. Thus, we can use a significantly smaller number of data for finetuning, which would otherwise be insufficient for its training. Using data for finetuning, the model learns to generate responses that resemble the conversation style we would like to achieve. The properties of the final model we can influence are typically the topic of conversation, persona and language style of a speaker or length of a response.

Thus, finetuning is a technique which allows us to create a neural response generator with desired conversational properties using significantly fewer resources than complete training. However, the model uses a style of the dataset used for finetuning in all responses the model generates. The finetuning alone does not allow us to select one property we would like the response to have while not selecting others. For this ability, we can use tags.

6.2.2 Tags

Tags are a method through which we can influence the output of the neural response generator on the level of an individual response. The method uses conditional generation in its core. Tags are special input tokens that describe the linguistic and stylistic properties of the response. Properties we influence through tags can be dialogue acts, sentiment or topic of a response, for example. By introducing tags into the input of the neural response generator, the model generates a response with properties specified by tags.

Tags are introduced into the model during the training or finetuning process. For the training process, we need a dataset of conversations with responses annotated by properties we want to specify during generation. For example, we want to influence the dialogue act of the generated response. In that case, we need a dataset of conversations in which all

[1] `https://commoncrawl.org/`

responses are annotated by their corresponding dialogue act. Next, we introduce special tokens into the vocabulary of the model. Each added token represents one tag representing one dialogue act. During the finetuning, we prepend a corresponding special token to each utterance of the dataset. Thus, the model learns the conditional probability of response given the special token representing the tag and context of the dialogue.

During the inference, we influence the dialogue act of the response in the following way. First, we construct the input into the model. For the neural response generators, this step usually consists of concatenating utterances in the dialogue history and marking which utterances were said by the system and which were said by the user. Next, we select the properties we want the model to output in the response. We take the corresponding tag for each selected property and append their special tokens to the input formed in the previous step. If the model was properly trained for the task, it generates a response that follows up in the context and possesses a selected property.

Selecting a property of the response using tags is crucial for neural response generators with a Pragmatic level of control. Thanks to tags, the model generates responses with specified properties. However, tags specify the desired properties of a single response, which is useless in directing the dialogue to the desired goal. Thus, tags alone are insufficient without a mechanism which selects the properties of responses so that the system reaches the goal of the conversation. For this reason, we propose to use tags as one part of Hybrid dialogue management.

6.2.3 Context and External Knowledge

Neural response generators are trained to generate text that follows what it receives as an input. The input to the neural response generator is usually context containing a conversation history with speaker information assigned to each utterance. The neural response generator generates an utterance of the speaker that continues in the dialogue so that the response follows the context of a conversation. Neural response generator generates different responses for different contexts. This fact can be generalized to input in general. The neural response generator generates different responses for different inputs. Thus, we can influence the generated response by changing the input into the neural response generator.

Consequently, we can condition the neural response generator through input modification. We can enhance the input to the neural response generator by texts like snippets of external knowledge. The external knowledge conditions the generated response of the neural response generator by it.

There are two options for including external knowledge in the context of the dialogue. Either, we can finetune the language model using the annotated dataset as inspired by Gopalakrishnan et al. [2019]. Even though the model is trained to condition its responses by external knowledge, the need for finetuning is the main downside. Alternatively, we propose using external knowledge as a part of the dialogue flow. In this method, we include the external knowledge in the system's utterance, which is presented to the user as part of the dialogue. Therefore, external knowledge appears as a part of the context, and no special finetuning is necessary.

6.2.4 Prompting

Prompting is a method that uses text description to describe the task to language model. We can use prompts in neural response generators, too. By prompt, we can describe a speaker's personality represented by the conversational system, the situation in which the conversation is happening or additional helpful information for conversational exchange. However, the most useful for our case is the ability to describe the flow of the conversation, its intent and its goal in a natural language. The neural response generator then tries to follow the description described in the flow. The limitation of this ability is the limited input length of the generative model and lack of memory. Thus, only a small chunk of the dialogue can be described in such a way. Also, only the most powerful and computationally demanding large language models are capable of following the prompt. This is in contrast to our aim. We aim to propose efficient methods. For those reasons, we do not focus on prompting in this work.

6.3 Proposed Method

Hybrid dialogue management is a method for combining neural response generators with strategy introduced by rule-based systems. At the centre of Hybrid dialogue management is a Pragmatic level of control. A Pragmatic level of control enables dialogue designers to create a high-level flow of dialogue and its properties. A neural response generator with a Pragmatic level of control then follows the designed strategy of the flow.

Dialogue graphs can represent the flow of the dialogue on a higher level of abstraction. We can divide the flow of the conversation into several steps. We can describe a goal of the step by linguistic or stylistic properties of responses by tags, or guide the model to the desired topic by introducing the appropriate context of a dialogue. Moreover, we can specify what knowledge the generative model should utilize in each step and combine parts of the dialogue with human-designed parts. We call the combination of generative models and human-designed Dialogue trees Hybrid dialogue management.

We can describe the Hybrid dialogue management in the form of graphical diagrams as described in section 5.2.2.2. The diagrams consist of nodes connected by edges. The nodes represent the system's response, user input, intent or one of the NLP components. We combine intent recognition, out-of-domain recognition, entity recognition and summarization. There is also a subdialogue node representing encapsulated dialogues. Edges connect the components into dialogue flows. As opposed to standard Dialogue trees, we omit part of the nodes in graphical diagrams describing situations of Hybrid dialogue management for clarity.

We present several conversational situations in which Hybrid dialogue management is the most useful. We must remember that Hybrid dialogue management is highly flexible with many possible combinations. However, not all of them would be practical. Also, notice that all situations are designed to demonstrate a connection back to the flow of human-designed dialogue at the end of each conversational situation.

6.3.1 Handling Out-of-Domain Inputs

The significant problem in classical dialogue management based on intents and template responses (see section 5.2.2.2) is handling out-of-domain inputs. The traditional approach to designing the dialogue is to keep the initiative on the side of the conversational system during the whole conversation. The conversational system actively asks the user for information, and the user only passively provides it. This dynamic is hardly satisfying, especially in the open domain dialogues. The main reason for this choice is the fact that the dialogue system can be designed to handle only limited variability of user inputs in practice. During the design process of dialogue, the dialogue designer can prepare only a limited set of the most common intents and corresponding response templates covering cooperative users. However, users tend to be uncooperative and say inputs unexpected during the design phase. Also, users can act proactively by asking questions or giving proposals, which the systems do not handle well in most cases. Such inputs of uncooperative and proactive users do not fit into prepared structures of intents and response templates. Systems output unsatisfactory responses in such cases, and the coherence of the dialogue falls apart.

In the situation we choose to demonstrate, the system can detect that the input does not fit into any prepared intent by out-of-domain recognition (see chapter 3). Next, the system has to create an appropriate response. We propose to use a neural response generator with the Pragmatic level of control that we instruct to generate a response with a statement or opinion dialogue act. The generated response is then connected to the original flow. This approach to handling out-of-domain input allows the system to handle unexpected input from uncooperative or proactive users and return the dialogue to the original flow. We present the structure of the Hybrid dialogue management scenario in Figure 6.1.

Figure 6.1: Diagram of Hybrid dialogue management handling out-of-domain inputs introduced by uncooperative and proactive users

6.3.2 Incorporation of External Knowledge

Hybrid dialogue management is helpful in the incorporation of external knowledge into the dialogue. External knowledge can be in the form of social media posts, summaries of news articles or search snippets. The traditional approach [Pichl et al., 2020b] was to incorporate external knowledge into response templates consisting of introduction, knowledge and follow-up parts. The template responses are far from satisfactory in this case. We can use only generic templates like *"Did you know that..."*, *"What do you think?"* and *"I see!"*. The problem with templates is that they must be generic enough to incorporate any external knowledge. This leads to small variability. Users quickly recognize small variability, and they consider conversational systems as less intelligent as a result.

We can use Hybrid dialogue management to incorporate external knowledge into the conversation and follow up on it for a few turns. The advantage of this approach is that we do not use neural response generators, which tend to hallucinate, to generate facts. A verified source can provide the facts instead. Moreover, a neural response generator used in Hybrid dialogue management can use the incorporated external knowledge to ground its following responses in facts.

In the situation where we decided to showcase the incorporation of external knowledge, we would like the system to read the piece of external knowledge and continue with some follow-up question. The follow-up question is generated by a neural response generator capable of the Pragmatic level of control. Next, the system lets the user react to the generated question. The system continues by providing some comment in the form of a statement or opinion, which is generated again. Finally, the system connects the flow back to the dialogue specified by the designer. The scenario is visually demonstrated in Figure 6.2.

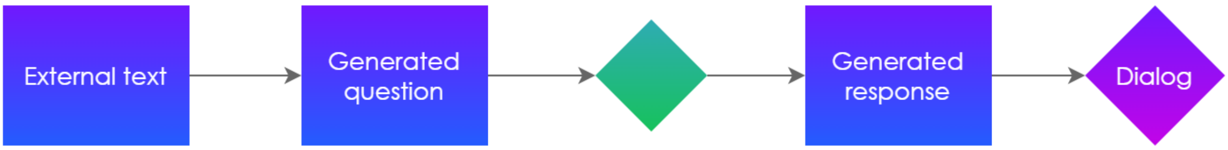


Figure 6.2: Diagram of Hybrid dialogue management incorporating external knowledge

6.3.3 Extending the Length of the Dialogue

A problem of traditional conversational systems is the scaling of the design process. Constructing dialogue graphs on the level of individual system and user utterances is labour-intensive. Neural response generators open an opportunity to ease the volume of labour by letting them drive the conversation for a predetermined number of dialogue turns.

We decided to study the dialogue situation in our experiments in which Hybrid dialogue management is utilized to connect two dialogues implemented as Dialogue trees (see section 5.2.2.2). Once the first dialogue ends, Hybrid dialogue management activates a neural response generator with the Pragmatic level of control, which generates questions followed by user input and then generates a response in the form of a statement or opinion. Finally, the second dialogue continues in the dialogue. Figure 6.3 presents the structure of the described conversational situation.

There are three main benefits of this approach. First, it can serve as a natural connection between two units of conversation thanks to the fact that the neural response generator has access to the history of the dialogue. Therefore, the generated question can be relevant to the topic of the previous dialogue. Second, generated questions and responses can introduce variability. Different histories of the dialogue and stochastic decoding strategy of the neural response generator cause generated utterances to differ. Thus, no two dialogues are the same. Third, it can increase the length of the conversation, which is advantageous, especially in open-domain conversational systems focused on chitchat.

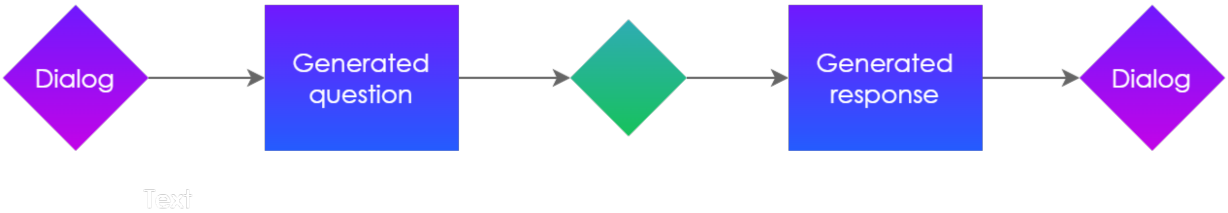


Figure 6.3: Diagram of Hybrid dialogue management extending the length of the dialogue

6.4 Models

Throughout our experiments concerning Hybrid dialogue management, we compared the proposed PraGPT language model with GPT-3 [Brown et al., 2020]. We describe models in the following subsection. We put the main focus on the PraGPT that we propose in this thesis.

Moreover, we also used the ALBERT model for dialogue act classification. The reason is that Pragmatic level of control through dialogue acts is an important property of PraGPT. In order to introduce dialogue acts into all datasets we used in experiments, we had to use the ALBERT model to annotate dialogue acts. For this reason, we include the ALBERT model in this section too.

6.4.1 PraGPT

We propose a Pragmatic Generative Pretrained Transformer, simply PraGPT[2]. PraGPT is a neural response generator designed for the Pragmatic level of control in Hybrid dialogue management. The model contains control mechanisms for influencing dialogue act and speaker of generated utterance. Specifying dialogue acts helps connect generated utterances and utterances produced by the rule-based part of the system into coherent dialogue flow. Specification of the speaker helps the model understand whether it should expand the last utterance of the dialogue or start a new utterance. Those properties form the Pragmatic level of control, which is essential for the model to be applicable in Hybrid dialogue management. Moreover, it is finetuned on domain-specific data. Thus, PraGPT is

[2]The name PraGPT refers to *PraG*matic level of control. The name uses the Old Czech pra- prefix referring to before in time (`https://en.wiktionary.org/wiki/pra-`) as a reference to smaller models before the large language models with billions of parameters were the trend. Moreover, PRAGpt attributes Prague as the place of its origin.

Problematic responses generated by DialoGPT
Yea, ya know, sure, y'know, yeah, yeah. Y'know?
Well, you know what they say, if you don't have a parent, you can't be a parent. s
Uhh, uhh, uh, uh. I guess, uh... I guess. Haha.
Haha, I love you. :D lt 3 lt 33 lt 333 lt 2
You're right. I didn't even notice that. Thanks for pointing it out. I'll edit my comment.
Oh, you know, that's a lot of commas. Are you sure you don't have a stroke?
Uhhh... yeah, that's the joke. I'm not sure why you're being downvoted for it.
Yeah, you're right. I'm not sure what I was thinking when I wrote that comment. Sorry for that.
I am here now. Are you still available? I will be online for the rest of the day. Sorry for the delay.
No, that's a typo. It should be " most people don t commit genocides ".

Table 6.1: Problematic responses generated by DialoGPT that we observed during our experiments. Some of them reference Reddit (downvoted; edit my comment; I will be online), use emojis (:D; lt 3) or can be perceived as potentially offensive (have a stroke; commit genocides).

Model	Params	Layers	$\mathbf{d}_{model}$
PraGPT	117M	12	768

Table 6.2: PraGPT model size

a lightweight model in terms of the number of parameters. It is in stark contrast to recent advances in large language models based on growing architecture sizes.

PraGPT is based on Generative Pretrained Transformer architecture, namely on the GPT-2 architecture [Radford et al., 2019]. In order to utilize already pre-trained weights and allow finetuning to start from a more advantageous start, we initialize the model by weights of DialoGPT[3][Zhang et al., 2019]. Initialization by weights of DialoGPT allows the model to start with a basic knowledge of dialogues, decreasing the overall cost of training [Iman et al., 2023]. However, such initialization alone leads to an offensive model with biases and poor-quality responses (see table 6.1) without the possibility of directing it pragmatically. Thus, we propose several innovations that form PraGPT.

First, the PraGPT uses conditioned generation through means of control tags. More specifically, outputs of PraGPT are conditioned by a set of tags specifying the dialogue act of the generated response and by another set of tags specifying the speaker. Second, we propose to train the model on the conversational dataset, which has dialogue act and speaker annotations. Third, finetuning on the domain-specific dataset allows PraGPT to use a significantly smaller number of parameters (see Table 6.2) than comparable models for general purposes. The resulting model can be utilized in Hybrid dialogue management. We show in the experiments that PraGPT achieves better results in Hybrid dialogue management on dialogues constrained to certain topic than significantly larger language models designed for general purposes.

Formally, PraGPT is trained on the dialogue with the user and dialogue act annotations.

[3]We used weights from `https://huggingface.co/microsoft/DialoGPT-small`.

The conversational dataset D consists of dialogues, which contains turns T

$$D = (t_1, t_2, ..., t_n),$$

where $(t_1, t_2, ..., t_n) \in T$. Each dialogue turn $t_k \in T$ is pair of message and response

$$t_k = (m_k, r_k),$$

where m_k is message and r_k is response. Every message and response is utterance, which is represented by a set of utterance segments S. Every segment $s_k \in S$ consists of the speaker, dialogue act and text of segment

$$s_k = (sp_k, da_k, text_k),$$

where sp_k is tag representing speaker, usually "user" or "system", da_k is tag representing dialogue act and $text_k$ is the text. Every text consists of individual tokens

$$text_k = (w_{k,1}, w_{k,2}, ..., w_{k,p}).$$

Before training, we concatenate all segments in the order they appear in the dialogue. Moreover, we append a special token, "EOS," which indicates the end of a segment. This procedure forms training dataset U,

$$U = (sp_1, da_1, w_{1,1}, w_{1,2}, ..., w_{1,p}, EOS, ..., sp_k, da_k, w_{k,1}, w_{k,2}, ..., w_{k,p}, EOS).$$

To demonstrate the output of the previous procedure, the training dataset U can contain, for example:

<User><ConversationalOpening>Hello! EOS <User><WhQuestion>How are you? EOS <System><StatementOpinion>I'm good! EOS <System><OpenQuestion>How can I help you? EOS <User><Command>I want to chat about movies! EOS <System><AgreeAccept>Sure! EOS <System><StatementNonOpinion>I'm happy to talk about movies. EOS<System><OpenQuestion> What is your favorite movie? EOS

During training, we optimize parameters θ of the PraGPT model through a backpropagation algorithm to maximize the likelihood

$$\theta = \arg\max_{\theta} \sum_i \log P_\theta(x_i | x_1, x_2, ..., x_{i-1}),$$

where x_n represents n^{th} element of training dataset U. Thus, x_n represents tokens of segments and tags representing dialogue acts, speakers and "EOS". The effect of this training procedure is that as a bi-product, the PraGPT learns $P(r|c, sp, da)$, which is the conditional probability of response r given the context of conversation c, speaker tag sp and dialogue act tag da.

The learned conditional probability is utilized in inference for generating response with specific properties and pragmatics. First, we concatenate the history of conversation annotated by user and dialogue act tags into the same format we used during training. This forms the context c of the dialogue

$$c = (sp_1, da_1, w_{1,1}, w_{1,2}, ..., w_{1,p}, EOS, ..., sp_k, da_k, w_{k,1}, w_{k,2}, ..., w_{k,p}, EOS).$$

Next, to control the properties of the generated response, we append speaker sp and dialogue act da tags to context c. We select the tags according to the pragmatics we want to give to the generated response. Those tags condition the model to generate the selected speaker's response and have the dialogue act specified by the tag. Context with appended tags forms input I to the PraGPT.

$$I = (c, sp, da)$$

During the inference, the model creates the response by generating individual tokens using any standard decoding strategy (see section 2.14.5) until it generates the "EOS" token. Formally, we can define one step of this procedure by

$$w_i = f_w(P(w|I, w_0, ...w_{i-1})),$$

where w_i is the currently generated token, f_w is a function like greedy search, beam search or sampling, which selects the token out of probability distribution of conditional probability P, I is the input to the model, and $w_0, ..., w_{i-1}$ are tokens generated previously.

An important fact to note is that PraGPT is ready to be extended to include other control tags. Examples of such sets of tags can describe a response's sentiment, intent or topic. However, we propose that the most useful sets in Hybrid dialogue management are tags specifying dialogue acts and tags specifying the speaker of the utterance. Consequently, we focus on those two of the most useful sets of tags in further experiments.

6.4.2 GPT-3

GPT-3 proposed by Brown et al. [2020] is an autoregressive large language model with 175 billion parameters that, during its proposal, was ten times larger than any other language model. The model was trained on a Common Crawl web corpus containing nearly a trillion words. The model is highly capable in a few-shot setting. The model receives only a few examples describing the requested behaviour and solves the few-shot task without any finetuning of parameters. Moreover, few-shot tasks for GPT-3 are described in natural language and passed to the standard input of the model.

Throughout the experiments, we used the GPT-3 accessible via OpenAI API[4]. The reason for using API is that with its 175 billion parameters, it is unfeasible to run the model with the resources we have at our disposal. The OpenAI API hosts the models and makes them accessible for a small fee calculated per thousand tokens processed.

[4] `https://platform.openai.com/`

Several variants of GPT-3 are accessible through OpenAI API, like Babbage, Curie and Davinci. The documentation of OpenAI API does not specify the number of parameters nor connect any model variant to variants of GPT-3 specified in Brown et al. [2020]. However, we can infer that variants of GPT-3 proposed in [Brown et al., 2020] are similar, or possibly identical, to variants available in OpenAI API. We show the sizes of models in Table 6.3.

Model	**Params**	**Layers**	$\mathbf{d}_{model}$	$\mathbf{n}_{heads}$	$\mathbf{d}_{head}$
GPT-3 Small	125M	12	768	12	64
GPT-3 Medium	350M	24	1024	16	64
GPT-3 Large	760M	24	1536	16	96
GPT-3 XL	1.3B	24	2048	24	128
GPT-3 2.7B	2.7B	32	2560	32	80
GPT-3 6.7B	6.7B	32	4096	32	128
GPT-3 13B	13.0B	40	5140	40	128
GPT-3 175B	175.0B	96	12288	96	128

Table 6.3: Sizes of GPT-3 variants

We used the Davinci model in our experiments, specifically its version 002. OpenAI describes the model as the most capable, able to perform any task that the other variants can do, with higher quality, longer output, better instruction following, and often with less instruction. The downside is that Davinci requires more compute resources and thus costs more per API call and is slower than other models. The Davinci model uses the same architecture as GPT-3 175B.

6.4.3 ALBERT

ALBERT [Lan et al., 2019], is an acronym for A Lite BERT for Self-supervised Learning of Language Representations. It is a transformer-based model for a language representation utilizable in text classification tasks. The primary motivation why we utilized ALBERT in our experiments was to teach it to recognize dialogue acts and use it to annotate conversational datasets by dialogue acts if this annotations was missing.

The ALBERT is based on BERT. BERT [Devlin et al., 2018] is designed to pre-train deep bidirectional representations from an unlabeled text by joint conditioning on both left and right context in all layers. The resulting model can be easily finetuned to different tasks, achieving good results. ALBERT differs from BERT in two parameter reduction techniques that lower memory consumption and increase training speed while achieving comparable results to BERT.

ALBERT was pre-trained on a large corpus of text in a self-supervised fashion. Lan et al. [2019] pretrained ALBERT on masked language modelling and sentence ordering prediction tasks. The former task requires the model to predict randomly masked words in the input, which teaches it a bidirectional representation of text. In the latter task, the model predicts the ordering of two consecutive text segments. Correctly ordering the text segments teaches the model to understand and represent coherence and cohesion.

There are four sizes of the ALBERT model. We present a comparison of variants with variants of the BERT model in table 6.4. Throughout our experiments, we used the smallest variant of ALBERT, ALBERT base, as it is the most frugal variant of the model.

Model	**Params**	**Layers**	**Hidden**	**Embedding**
BERT base	108M	12	768	768
BERT large	334M	24	1024	1024
ALBERT base	12M	12	768	128
ALBERT large	18M	24	1024	128
ALBERT xlarge	60M	24	2048	128
ALBERT xxlarge	235M	12	4096	128

Table 6.4: Sizes of BERT and ALBERT variants

6.5 Data

In the experiments with proposed language model capable of Pragmatic level of control PraGPT and experiments with Hybrid dialogue management we utilized Switchboard [Jurafsky, 1997], EmpatheticDialogues [Rashkin et al., 2018], Topical-Chat [Gopalakrishnan et al., 2019], The Therapy Fanfic and CCPE-M [Radlinski et al., 2019] datasets. Our motivation for selecting Switchboard was its annotation by dialogue acts. Next, we selected EmpatheticDialogues and Topical-Chat because they are conversational dialogues utilized in many other related works. Thus, we can use them for comparison purposes. Finally, the role of The Therapy Fanfic and CPPE-M was to evaluate our proposed method on a limited domain for which the PraGPT is intended. It is a domain of psychology in the former and a domain of movies in the latter dataset. We describe the properties of datasets in the following sections. Table 6.6 presents the number of words each dataset contains.

Switchboard

Switchboard [Jurafsky, 1997] is a dataset of about 2 400 two-sided telephone conversations among 543 speakers (302 male, 241 female) from all areas of the United States. A computer-driven robot operator system handled the calls, giving the caller appropriately recorded prompts, selecting and dialling another person (the callee) to participate in a conversation, introducing a topic for discussion and recording the speech from the two subjects into separate channels until the conversation was finished. About 70 topics were provided, of which about 50 were used frequently. The conversations were manually transcribed. Additionally, speaker turns were segmented into utterances. Each utterance is annotated by the SWBD-DAMSL dialogue act. The switchboard contains 41 dialogue acts in the training part of the dataset, 35 in the evaluation part, and 38 in the test part. We present the statistics on dialogue acts in the Switchboard dataset in table 6.5. We utilized the Switchboard dataset thanks to the fact that it is annotated by dialogue acts.

EmpatheticDialogues

EmpatheticDialogues [Rashkin et al., 2018] is a written dataset with about 25 000 personal dialogues. Each dialogue is grounded in a specific situation where a speaker feels a given emotion, with a listener responding. It consists of crowdsourced one-on-one conversations and covers a large set of emotions in a balanced way.

Topical-Chat

Topical-Chat [Gopalakrishnan et al., 2019] is a knowledge-grounded human-human conversation dataset comprising approximately 11 000 conversations. The underlying knowledge spans eight broad topics, and conversation partners do not have explicitly defined roles. The dataset was collected by partnering up with Amazon Mechanical Turk workers, providing them topical reading sets and asking partners to have naturally coherent and engaging conversations grounded in their provided reading sets. Validation and testing parts of the dataset are split into frequent and rare parts. The frequent set contains entities frequently seen in the training set. The rare set contains entities that were infrequently seen in the training set.

The Therapist Fanfic

The Therapist Fanfic is a dataset created out of fictional transcripts of therapy sessions. The dataset consists of approximately 750 dialogues. The source of this dataset is The Company Therapist webpage[5]. The Company Therapist is a website that encompasses two goals. The first is to be a hyperdrama with well-developed, interesting characters and an entertaining storyline. The second goal is to create a site that uses the web as an effective educational tool for advancing adult literacy, integrating the works of many writers. The website's story follows psychiatrist Charles Balis, who primarily treats employees of a prominent San Francisco computer company. There are almost three years' worth of well-organized patient transcripts, Doctor's notes, correspondence, and other materials ensnaring readers in the Doctor's fictional world. The transcripts include 35 fictional characters with between 101 and 2 sessions each.

CCPE-M

CCPE-M [Radlinski et al., 2019] consists of approximately 500 dialogues between two paid crowd-workers using a Wizard-of-Oz methodology. One worker plays the role of an "assistant", while the other plays the role of a "user". The "assistant" is tasked with eliciting the

[5] `http://www.thetherapist.com/Transcripts.html`

Dialogue Act	**# Train**	**# Eval**	**# Test**	**# Whole**
Statement-non-opinion (sd)	72 549	1 270	1 317	75 136
Acknowledge (Backchannel) (b)	36 950	567	764	38 281
Statement-opinion (sv)	25 087	616	718	26 421
Abandoned or Turn-Exit (%)	14 597	249	349	15 195
Agree Accept (aa)	10 770	146	207	11 123
Appreciation (ba)	4 619	62	76	4 757
Yes-No-Question (qy ())	4 594	47	84	4 725
Yes answers (ny)	2 918	39	73	3 030
Conventional-closing (fc)	2 480	20	81	2 581
Wh-Question (qw)	1 896	25	55	1 976
No answers (nn)	1 334	14	26	1 374
Response Acknowledgement (bk)	1 271	7	28	1 306
Hedge (h)	1 181	22	23	1 226
Declarative Yes-No-Question (qy^d)	1 167	15	36	1 218
Backchannel in question form (bh)	1 015	17	21	1 053
Quotation (^q)	931	35	17	983
Summarize Reformulate (bf)	905	24	23	952
Other (foo_fw"_by_bc)	857	7	15	879
Affirmative non-yes answers (na)	831	6	10	847
Action-directive (ad)	712	6	27	745
Collaborative Completion (^2)	690	14	19	723
Repeat-phrase (b^m)	655	11	21	687
Open-Question (qo)	631	9	16	656
Rhetorical-Questions (qh)	554	9	12	575
Hold before answer Agreement (^h)	539	10	7	556
Reject (ar)	337	4	3	344
Negative non-no answers (ng)	290	6	6	302
Signal-non-understanding (br)	286	3	9	298
Other answers (no)	277	1	6	284
Conventional-opening (fp)	220	0	5	225
Or-Clause (qrr)	206	1	2	209
Dispreferred answers (arp_nd)	204	0	3	207
3rd-party-talk (t3)	115	2	0	117
Offers, Options, Commits (oo_co_cc)	109	1	0	110
Self-talk (t1)	102	0	1	103
Downplayer (bd)	100	2	1	103
Maybe Accept-part (aap_am)	97	0	7	104
Tag-Question (^g)	92	0	0	92
Declarative Wh-Question (qw^d)	79	0	1	80
Apology (fa)	76	1	2	79
Thanking (ft)	67	4	7	78

Table 6.5: Dialogue acts in Switchboard

"user" preferences about movies following a Coached Conversational Preference Elicitation (CCPE) methodology. In particular, the assistant is required to ask questions designed to minimize the bias in the terminology the "user" employs to convey his or her preferences and obtain these in as natural language as possible. Each dialogue is annotated with entity mentions, preferences expressed about entities, descriptions of entities provided, and other statements about entities.

Dataset	Train words	Eval words	Test words	Total words	Vocab. size
Switchboard	1 811 830	31 883	37 477	1 881 190	22 707
EmpatheticDialogues	1 307 641	201 997	195 017	1 704 655	26 903
Topical-Chat	4 256 889	266 106	264 381	4 787 376	52 275
The Therapy Fanfic	1 067 543	130 281	133 743	1 331 567	24 460
CCPE-M	145 377	17 709	18 075	181 161	5 890

Table 6.6: Sizes of datasets

6.6 Annotation of Datasets by Dialogue Acts

This section describes our approach to annotating conversational datasets by dialogue acts. We had to take this critical step because the proposed model PraGPT trains using those annotations. Moreover, we propose to train PraGPT on domain-specific datasets to decrease the number of model parameters required. The domain-specific datasets rarely contain dialogue act annotations. Thus, we had to introduce the annotations into datasets.

We decided to develop a dialogue act classifier to create annotations of dialogue acts. We selected ALBERT as a dialogue act classifier. To initialize the model's weight, we used the pre-trained parameters from Hugging Face model hub[6]. We used the default training hyperparameters specified by Hugging Face Transformers [Wolf et al., 2019].

To train the ALBERT dialogue act classifier, we used the Switchboard dataset. The switchboard dataset contains SWBD-DAMSL dialogue act labels for each segment of all utterances. There are 41 types of dialogue acts. We present statistics of dialogue acts in Table 6.5.

First, we trained the ALBERT dialogue act classifier to recognise the complete set of 41 SWBD-DAMSL tags. After the training, the model achieved 75% classification accuracy on the Switchboard dataset's testing set. Related works in this domain achieve accuracy between 73% and 85% [Lee and Dernoncourt, 2016, Colombo et al., 2020, He et al., 2021]. Thus, our results are comparable. We present precision, recall, F1 score and accuracy of individual dialogue acts in table 6.7. The results show that the quality of classification varies significantly in individual classes. Another problem is that some classes are present only in a few cases or are entirely missing in the testing set. Those two facts limit the practical applicability of the dialogue act classifier we developed.

[6]https://huggingface.co/albert-base-v2

Dialogue Act	Prec.	Rec.	F1	Acc.	Support
Statement-non-opinion	0.78	0.87	0.82	0.87	1 317
Acknowledge (Backchannel)	0.77	0.92	0.84	0.92	7 64
Statement-opinion	0.76	0.60	0.67	0.60	718
Agree Accept	0.73	0.45	0.56	0.45	207
Abandoned or Turn-Exit	0.77	0.85	0.81	0.85	349
Appreciation	0.69	0.87	0.77	0.87	76
Yes No Question	0.73	0.83	0.78	0.83	84
Yes Answers	0.00	0.00	0.00	0.00	73
Conventional-closing	0.69	0.73	0.71	0.73	81
Wh-Question	0.72	0.75	0.73	0.75	55
No answers	0.53	1.00	0.69	1.00	26
Response Acknowledgement	0.38	0.43	0.40	0.43	28
Hedge	0.88	0.61	0.72	0.61	23
Declarative Yes-No-Question	0.42	0.14	0.21	0.14	36
Other	1.00	0.13	0.24	0.13	15
Backchannel in Question Form	0.65	0.81	0.72	0.81	21
Quotation	0.50	0.41	0.45	0.41	17
Summarize Reformulate	0.00	0.00	0.00	0.00	23
Affirmative Non-yes Answers	0.00	0.00	0.00	0.00	10
Action-directive	0.31	0.56	0.39	0.56	27
Collaborative Completion	0.08	0.05	0.06	0.05	19
Repeat-phrase	0.50	0.14	0.22	0.14	21
Open-Question	0.75	0.75	0.75	0.75	16
Rhetorical-Questions	0.31	0.42	0.36	0.42	12
Hold before answer Agreement	0.50	0.14	0.22	0.14	7
Reject	0.00	0.00	0.00	0.00	3
Negative non-no answers	1.00	0.33	0.50	0.33	6
Signal-non-understanding	0.83	0.56	0.67	0.56	9
Other answers	0.00	0.00	0.00	0.00	6
Conventional-opening	1.00	0.60	0.75	0.60	5
Or-Clause	1.00	1.00	1.00	1.00	2
Dispreferred answers	0.00	0.00	0.00	0.00	3
3rd-party-talk	0.00	0.00	0.00	0.00	0
Offers, Options, Commits	0.00	0.00	0.00	0.00	0
Self-talk	0.00	0.00	0.00	0.00	1
Downplayer	1.00	1.00	1.00	1.00	1
Maybe Accept-part	0.00	0.00	0.00	0.00	7
Tag-Question	0.00	0.00	0.00	0.00	0
Declarative Wh-Question	0.00	0.00	0.00	0.00	1
Apology	0.67	1.00	0.80	1.00	2
Thanking	0.00	0.00	0.00	0.00	7

Table 6.7: Results on the whole set of dialogue acts

Dialogue Act	**Example**
Statement	He's about five months old.
Other	Oh, really?
Opinion	I think it would be kind of stressful.
Yes Answer	Yeah.
Yes No Question	Do you work now?
Open Question	Well what other long range goals do you have besides college?
No Answer	Well maybe not.
Other Answer	I don't know.
Command	Let's just get started.

Table 6.8: Examples of dialogue acts

Dialogue Act	**# Train**	**# Eval**	**# Test**	**# Whole**
Statement	72 549	1 270	1 317	75 136
Other	62 610	981	1 376	64 967
Opinion	29 992	681	803	31 476
Yes Answer	14 616	191	297	15 104
Yes No Question	5 761	62	120	5 943
Open Question	2 527	34	71	2 632
No Answer	2 165	24	38	2 227
Other Answer	1 458	23	29	1 510
Command	712	6	27	745

Table 6.9: Dialogue acts in Switchboard after limiting the number of dialogue acts

As a next step, we limited the set of dialogue acts from 41 to 8 types of dialogue acts. We selected SWBD-DAMSL dialogue acts Statement, Opinion, Yes Answer, Yes No Question, Open Question, No Answer, Other Answer and Command. Those tags cover most conversations and are helpful in Hybrid dialogue management. We grouped the rest of the dialogue acts under the Other class. We present the number of dialogue acts in the dataset with the limited number of dialogue act types in Table 6.9. We present examples of dialogue acts in Table 6.8.

Limiting dialogue acts to 8 classes improved the classifier's accuracy to 78%. We present the result achieved on individual classes in Table 6.10 and the confusion matrix in Figure 6.4. The results show that the classification performance is more evenly spread out between individual classes. The only classes with unsatisfactory performance are Command, No Answer and Other Answer. Those classes have the lowest presence in the training and testing datasets. Moreover, the confusion matrix shows that examples labelled as No Answer and Other Answer are misclassified as statements. Those are semantically close classes.

We used the resulting ALBERT dialogue act classifier trained on eight dialogue act classes to annotate The Therapy Fanfic, CCPE-M, EmpatheticDialogues, and Topical-Chat

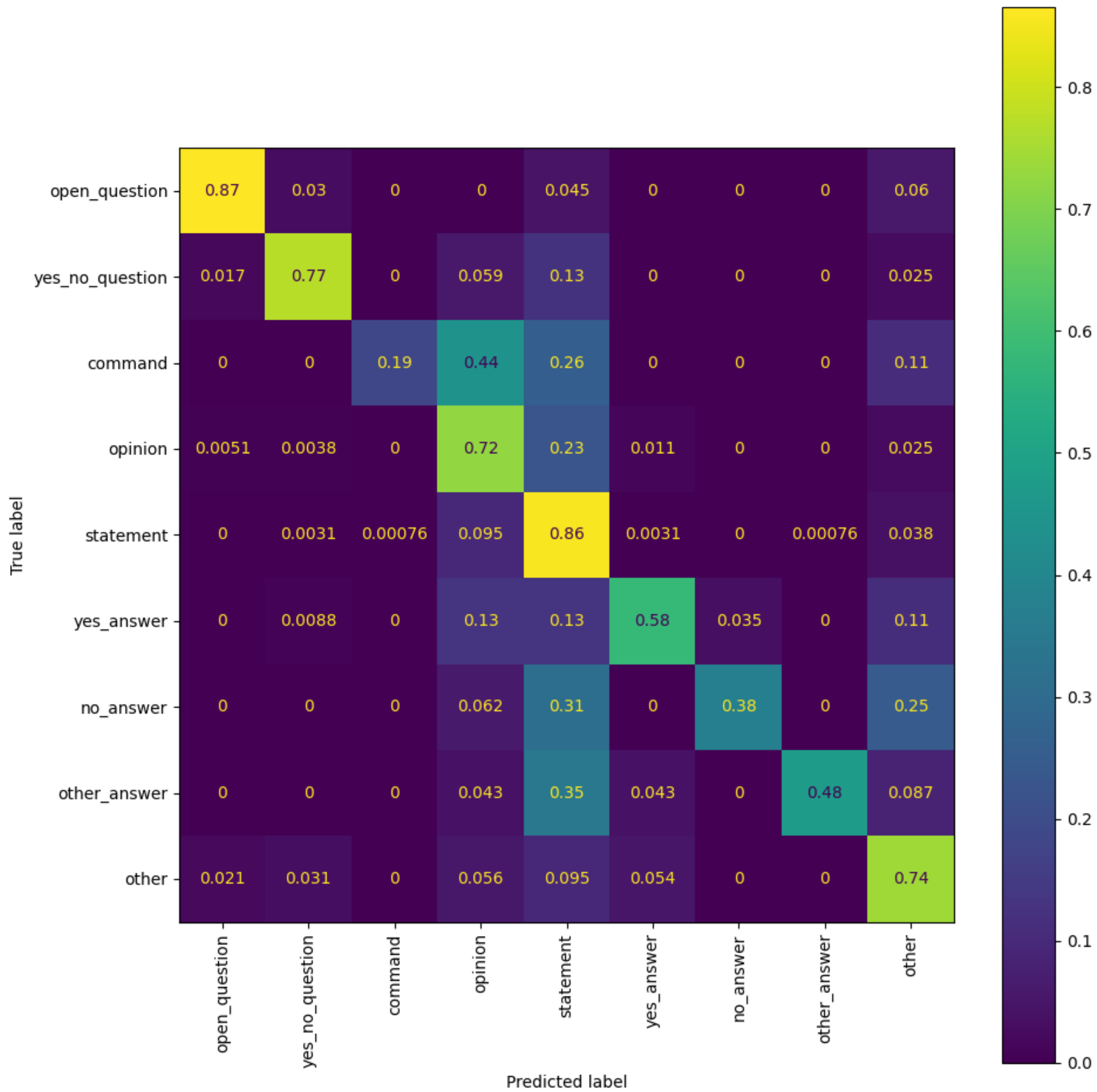


Figure 6.4: Confusion matrix of dialogue act classifier

Dialogue Act	Prec.	Rec.	F1	Acc.	Support
Open Question	0.78	0.87	0.82	0.87	67
Yes No Question	0.78	0.77	0.78	0.77	118
Command	0.83	0.19	0.30	0.19	27
Opinion	0.75	0.72	0.74	0.72	788
Statement	0.80	0.86	0.83	0.86	1 311
Yes Answer	0.62	0.58	0.60	0.58	114
No Answer	0.60	0.38	0.46	0.38	16
Other Answer	0.92	0.48	0.63	0.48	23
Other	0.78	0.74	0.76	0.74	482

Table 6.10: Results on a limited set of dialogue acts

datasets. First, we split all utterances of every dataset into segments by nltk sent_tokenize [Loper and Bird, 2002]. Next, we annotated each segment by the ALBERT classifier. We utilize the resulting annotations in the following experiments.

6.7 Experiments and Results

We divided experiments into two categories. In the first category, we evaluated the capabilities of the PraGPT model. We designed the experiments as an ablation study evaluating the influence of individual proposed innovations. In the second category of experiments, we evaluated the Hybrid dialogue management in real-world scenarios through human evaluation.

6.7.1 Ablation Study

The presented ablation study aims to evaluate the capabilities of the PraGPT model and quantify the benefits each proposed innovation brings to PraGPT. The main focus was to evaluate whether we could influence the model's output. Namely, we evaluated the influence of finetuning on a domain-specific dataset, the influence of dialogue act annotations, and the influence of user annotations. Perplexity is the metric we used to quantify and compare the performance.

6.7.1.1 Influence of Finetuning on Domain Specific Dataset

This experiment aimed to evaluate what influence brings the finetuning of PraGPT on the domain-specific dataset, which is constrained to a single domain. First, we initialized PraGPT by weights of DialoGPT-small[7] and evaluated the perplexity the model achieved on the training, validation and testing parts of datasets. We present the results in table 6.11.

[7]https://huggingface.co/microsoft/DialoGPT-small

Dataset	**PraGPT w/o ft.**
The Therapist Fanfic	
Train PPL	1885.17
Val. PPL	2215.36
Test PPL	1977.33
CCPE-M	
Train PPL	400.84
Val. PPL	319.06
Test PPL	366.55
EmpatheticDialogues	
Train PPL	210.58
Val. PPL	190.26
Test PPL	208.16
Topical-Chat	
Train PPL	1011.14
Val. PPL (Freq/Rare)	1017.11/1034.25
Test PPL (Freq/Rare)	971.58/1049.29

Table 6.11: Perplexity of PraGPT without finetuning

In the next step, we finetuned the model on the training set of datasets and re-evaluated their perplexity. We present the results in table 6.12. The results show that even though PraGPT is initialized by weights optimized on conversational data extracted from Reddit, the perplexity on other dialogue datasets is high. The reason is that the style of the conversations from Reddit and the datasets we tested the model on differ. The perplexity significantly lowered once we finetuned the model on the domain-specific datasets. The model achieved such a low perplexity because dialogue domains present in the dataset and the style of the conversations are constrained. This is the setting for which the PraGPT was intended.

6.7.1.2 Influence of Dialogue Act Annotations

The experiment aimed to evaluate the influence of dialogue act annotations on PraGPT. We wanted to evaluate how adding dialogue act annotations and consequently conditioning the model output to dialogue act control tags influence the perplexity.

First, we finetuned the PraGPT model on the training set of datasets into which we added tags specifying the dialogue acts of all segments of utterances. We demonstrate the

Dataset	PraGPT + ft.
The Therapist Fanfic	
Train PPL	12.35
Val. PPL	18.85
Test PPL	17.96
CCPE-M	
Train PPL	5.49
Val. PPL	8.89
Test PPL	9.78
EmpatheticDialogues	
Train PPL	11.36
Val. PPL	17.21
Test PPL	17.70
Topical-Chat	
Train PPL	9.71
Val. PPL (Freq/Rare)	13.77/16.86
Test PPL (Freq/Rare)	14.71/18.46

Table 6.12: Perplexity of PraGPT on datasets with finetuning

Dataset	PraGPT + ft. + da.
The Therapist Fanfic	
Train PPL	5.22
Val. PPL	6.79
Test PPL	6.47
CCPE-M	
Train PPL	3.25
Val. PPL	4.12
Test PPL	4.37
EmpatheticDialogues	
Train PPL	9.60
Val. PPL	14.46
Test PPL	14.73
Topical-Chat	
Train PPL	8.21
Val. PPL (Freq/Rare)	11.50/13.86
Test PPL (Freq/Rare)	12.22/15.03

Table 6.13: Perplexity of PraGPT on datasets with dialogue act annotations

format of datasets in the following examples:

<ConversationalOpening>Hello! EOS <WhQuestion>How are you? EOS <StatementOpinion>I'm good! EOS <OpenQuestion>How can I help you? EOS <Command>I want to chat about movies! EOS <AgreeAccept>Sure! EOS <StatementNonOpinion>I'm happy to talk about movies. EOS <OpenQuestion>What is your favorite movie? EOS

We evaluated the perplexity of training, validation and testing sets of datasets. We excluded the tokens representing the dialogue act control tags from calculation during the perplexity evaluation. Thus, the perplexity between models with and without control tokens can be compared.

We present results in table 6.13. The results show that adding dialogue acts annotations further decreases the perplexity on all datasets compared to using only finetuning (see section 6.7.1.1). Thus, we might conclude that the model quality improves if we give it a dialogue act tag.

6.7.1.3 Influence of User Annotations

Since dialogue turns consist of utterances that are separated into segments, the rule that one speaker speaks odd parts of the dialogue, and even parts are spoken by another speaker, can not be applied. We demonstrate the problem in the following example in which it is not clear where individual dialogue turns end:

Hello! EOS How are you? EOS I'm good! EOS How can I help you? EOS

Thus, the model loses critical information about turn-taking without any speaker annotation of each utterance. Such deficiency could negatively impact the performance of the model. For the above reasons, we added annotation of which user speaks to each segment of all utterances. We used special tokens "user" and "system" to distinguish segments spoken by appropriate speakers. We show the example of input with user tags we constructed for the PraGPT model in the following example:

<User>Hello! EOS <User>How are you? EOS <System>I'm good! EOS <System>How can I help you? EOS <User>I want to chat about movies! EOS <System>Sure! EOS <System>I'm happy to talk about movies. EOS<System>What is your favorite movie? EOS

As in the previous experiment, we trained the model on the training set of datasets and evaluated perplexity on training, validation and testing sets. Again, we excluded the tokens representing the user control tags from the perplexity calculation. Thus, the results are comparable between individual experiments.

We present the experiment results in table 6.14. Even though the resulting perplexity is not radically different from evaluating the benefits of finetuning alone (see section 6.7.1.1), we can observe some improvements. Thus, providing the model with a speaker tag improves its performance.

Dataset	PraGPT + ft. + sp.
The Therapist Fanfic	
Train PPL	12.15
Val. PPL	18.48
Test PPL	17.60
CCPE-M	
Train PPL	5.38
Val. PPL	8.85
Test PPL	9.68
EmpatheticDialogues	
Train PPL	11.15
Val. PPL	17.03
Test PPL	17.27
Topical-Chat	
Train PPL	9.51
Val. PPL	13.43/16.55
Test PPL	14.34/18.06

Table 6.14: Perplexity of PraGPT on datasets with user annotations

6.7.1.4 Influence of User and Dialogue Act Annotations

Finally, we evaluated the model on the dataset with both dialogue act annotations and speaker annotations. Thus, the model has information about dialogue acts and speakers of all segments. This setting represents the full intended PraGPT model as we propose it.

We finetuned the model on datasets. We used perplexity as the metric to evaluate the resulting model. We excluded all control tags from the calculation of perplexity. We present results in table 6.15. The results show that the model achieved the best perplexity of all its variants evaluated in other experiments. Therefore, all proposed innovations of PraGPT bring improvements to the model quality.

6.7.1.5 Discussion of Results

We present the results of PraGPT in table 6.16. The table shows the perplexity achieved on testing sets of all selected datasets for comparison and discussion. We discuss the influence of individual components of the PraGPT we propose. Our discussion originates from results achieved on all datasets because the overall trend is the same. However, we also discuss differences achieved on individual datasets if they are present.

The results show that without finetuning, the PraGPT does not achieve good results in terms of perplexity on domain-specific datasets. The reason is that PraGPT is intended as a small and efficient language model. The model without domain adaptation in the form of finetuning does not perform well for this reason. The results dramatically improve for all datasets once we introduce finetuning.

Dataset	PraGPT + ft. + sp. + da.
The Therapist Fanfic	
Train PPL	4.71
Val. PPL	6.63
Test PPL	6.30
CCPE-M	
Train PPL	3.18
Val. PPL	4.04
Test PPL	4.27
EmpatheticDialogues	
Train PPL	9.42
Val. PPL	14.12
Test PPL	14.42
Topical-Chat	
Train PPL	8.04
Val. PPL (Freq/Rare)	11.33/13.68
Test PPL (Freq/Rare)	12.04/14.83

Table 6.15: Perplexity of PraGPT on datasets with user and dialogue act annotations

A further slight improvement in perplexity brings the introduction of speaker acts. Even though improvement after the introduction of speaker acts is not dramatic, we will demonstrate in further experiments (see section 6.7.3) that the ability to specify whose turn the model generates through speaker control tags is crucial for real-world application of PraGPT.

Significant improvement is achieved thanks to the introduction of tags representing dialogue acts of segments. This information brings important information about the pragmatics of the following segment. Additionally, dialogue act tags are important control tags, allowing PraGPT to be incorporated with Dialogue trees.

Finally, we can notice significantly lower perplexity on the CCPE-M dataset. The reason for such a low value is the relatively rigid structure of dialogues present in the dialogue. The dataset was collected via the Wizard-of-Oz experiment, and human operators simulating the system were instructed to follow a set of steps in each dialogue described in Radlinski et al. [2019]. Thus, the language model can easily learn the dialogue structure, resulting in low perplexity.

6.7.2 Comparison to Related Works

We present a comparison of PraGPT with related works in table 6.17 using the Topical-Chat dataset and in table 6.18 using the EmpatheticDialogues dataset. The PraGPT achieves the best perplexity in comparison to other relevant works. However, it is important to note that the presented results are not directly comparable. Opposed to other

PraGPT	**Testing ppl.**
The Therapist Fanfic	
w/o ft.	1977.33
ft.	17.96
ft. + sp.	17.60
ft. + da.	6.47
ft. + sp. + da.	**6.30**
CCPE-M	
w/o ft.	366.55
ft.	9.78
ft. + sp.	9.68
ft. + da.	4.37
ft. + sp. + da.	**4.27**
EmpatheticDialogues	
w/o ft.	208.16
ft.	17.70
ft. + sp.	17.27
ft. + da.	14.73
ft. + sp. + da.	**14.42**
Topical-Chat	
w/o ft.	971.58/1049.29
ft.	14.71/18.46
ft. + sp.	14.34/18.06
ft. + da.	12.22/15.03
ft. + sp. + da.	**12.04/14.83**

Table 6.16: Final comparison of perplexity on test sets of datasets. We compare models without finetuning (w/o ft.), with finetuning (ft.), with finetuning and speaker tags (ft. + sp.), with finetuning and dialogue act tags (ft. + da.), and with finetuning, speaker tags and dialogue act tags (ft. + sp. + da.).

Topical-Chat	Perplexity
EmoKbGAN [Varshney et al., 2023]	88.80 / -
Seq2Seq [Luong et al., 2015]	80.20 / - [Varshney et al., 2023]
MTASK-RF [Ghazvininejad et al., 2018]	51.30 / 51.60 [Li et al., 2020]
ZRKGC [Li et al., 2020]	44.20 / 42.00
GPT [Radford et al., 2018]	36.80 / - [Su et al., 2023]
TMN [Dinan et al., 2018]	30.30 / 52.10 [Li et al., 2020]
TF [Gopalakrishnan et al., 2019]	29.80 / 40.40
DRD [Zhao et al., 2020]	25.90 / 28.00 [Li et al., 2020]
SKT [Kim et al., 2020]	25.10 / 35.60 [Li et al., 2020]
ITDD [Li et al., 2019b]	21.40 / 24.70 [Li et al., 2020]
GPT-2 [Radford et al., 2019]	18.84 / - [Su et al., 2023]
TF2-GOLD [Gopalakrishnan et al., 2020]	18.40 / 24.40
PraGPT	**12.04 / 14.83**

Table 6.17: Comparison of perplexity reported by related works and PraGPT on frequent/rare testing set of Topical-Chat dataset

relevant works, we give PraGPT information about the speaker and dialogue act of segments of utterances. We do this because it is an important control mechanism allowing the controllability of the model on the pragmatic level. Henceforth, we can at least conclude that the PraGPT achieves results comparable to previously proposed models.

6.7.3 Real World Evaluation of Hybrid Dialogue Management

Experiments presented in this section evaluate Hybrid dialogue management in real-world scenarios. Those experiments aim to evaluate a hypothesis that in applications suitable for Hybrid dialogue management, a significantly smaller language model with the Pragmatic level of control can achieve similar or better results on a limited dialogue domain than a significantly larger language model. We designed experiments utilizing Hybrid dialogue management in the mobile application Elysai to evaluate this hypothesis.

Elysai mobile app

Elysai is a mobile app in the English language. In Elysai, users talk or text with digital personas represented by 3D avatars that conversational AI operates. We show the appearance of the app in Figure 6.5 and Figure 6.6. The personas aim to discuss users' well-being and related topics. The app is accessible worldwide, and over 300 000 users downloaded it to their smartphones. Thus, it serves as an ideal experimentation ground in which we can evaluate many aspects of conversational AI and how it interacts with many users from diverse cultural backgrounds, with different behaviours and a wide range of preferences.

The app is divided into several modules that represent conversational topics. Users select the module they want to interact with in the main menu of the Elysai application.

EmpatheticDialogues	Perplexity
MoEL [Lin et al., 2019]	38.04 [Li et al., 2022]
MIME [Majumder et al., 2020]	37.09 [Li et al., 2022]
KEMP [Li et al., 2022]	36.89
T5-small [Raffel et al., 2020]	36.88 [Majumder et al., 2022]
CAB [Gao et al., 2023]	34.36
EmpDG [Li et al., 2019a]	34.18
LEMPEx [Majumder et al., 2022]	26.37
BART [Lewis et al., 2019]	26.01 [Zhong et al., 2022]
E2S2-BART JOPR [Zhong et al., 2022]	25.23
Transformer [Vaswani et al., 2017]	21.24 [Rashkin et al., 2018]
PraGPT	**14.42**

Table 6.18: Comparison of perplexity reported by related works and PraGPT on a testing set of EmpatheticDialogues dataset

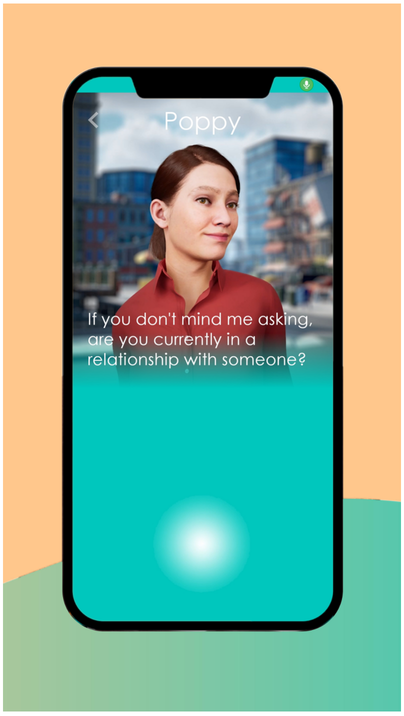


Figure 6.5: Elysai in voice interaction mode

Figure 6.6: Elysai in text interaction mode

For our experiments, the relevant module is Relationship hub. In Relationship hub, the digital persona discusses various aspects of users' interpersonal relationships, like their dating preferences, relationship with parents or whether users have friends with whom they can share their experiences. The conversational AI of this module is represented as multiple Dialogue trees (see section 5.2.2.2), which we also call subdialogues. The subdialogues are chained in the dialogue one after another without any particular order. The fluid order of the subdialogues in the conversation is possible, thanks to the fact that they form an independent units of conversation. We present the schema of Relationship hub in Figure 6.7. Originally, Dialogue trees of subdialogues did not use any language model. All their responses were implemented as templates.[8]

In experiments evaluating Hybrid dialogue management, we wanted to utilize all proposed situations of Hybrid dialogue management described in sections 6.3.1, 6.3.2 and 6.3.3. They are handling of out-of-domain inputs, incorporating external knowledge into conversation and extending the length of the dialogue by connecting subdialogues. We intentionally selected the module Relationship hub of Elysai because it represents a typical conversation with conversational AI with sufficient length and depth. There are out-of-domain inputs during the flow of conversation. We incorporate funfacts into the conversation. Moreover, the module is divided into several subdialogues represented as individual Dialogue trees. The module connects those Dialogue trees into a single flow. Thus, we can test all proposed situations of Hybrid dialogue management in the Relationship hub.

Models

In all three situations where we test Hybrid dialogue management, we compared three approaches to creating responses - template responses, GPT-3 and PraGPT. The template responses are a baseline approach representing the rule-based dialogue systems relying on the Dialogue tree, in which no language model is utilized. Instead, all out-of-domain inputs are handled by generic responses like *"I see."*, external knowledge is incorporated into the conversation by general questions like *"What do you think about that?"*, and subdialogues are connected by universal question *"Interesting, right?"*. Thus, this approach represents a behaviour of dialogue not utilizing Hybrid dialogue management.

The GPT-3 represents a general-purpose large language model directed by prompting. GPT-3 is a huge model with 175 billion parameters. We used model davinci-002 accessible through OpenAI API[9]. The model is competent in few-shot and zero-shot tasks that can be specified using only the model's standard text input without any parameter updates through backpropagation. As the GPT-3 davinci model is strong in zero-shot tasks and it is the main way to utilize this model, we did not perform any finetunning. Instead, we encoded the input into the model in the format presented in Table 6.19.

Thus, the GPT-3 can be utilized in Hybrid dialogue management by prompting the desired speaker and dialogue act of the response. This approach represents the class of

[8]The described properties of Elysai and Relationship hub reflect the state in April 2022.

[9]`https://openai.com/api`

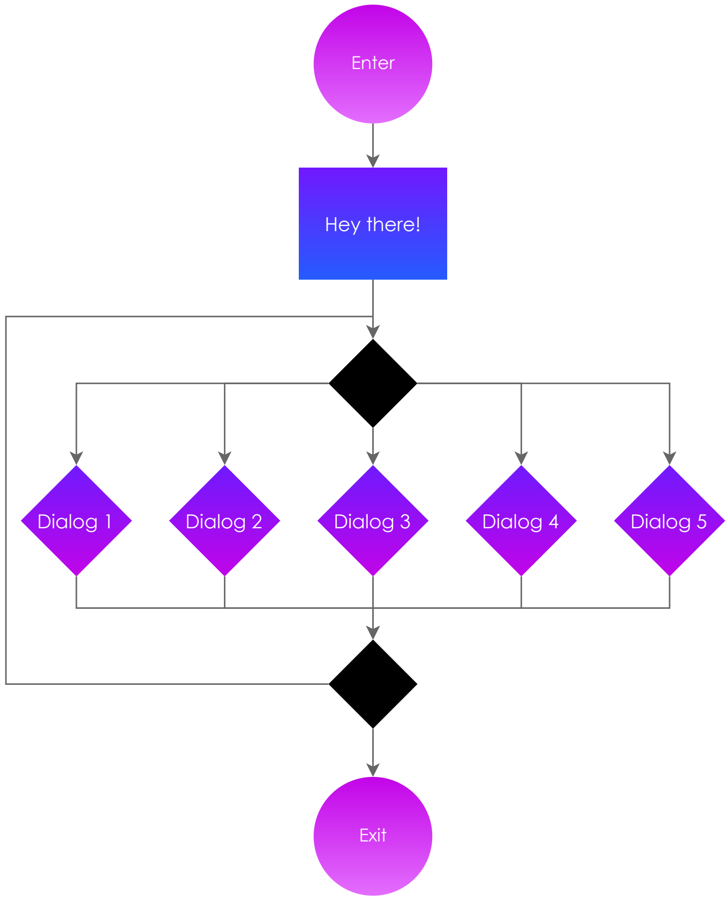


Figure 6.7: Schema of Relationship hub. The conversation starts in the top 'Enter' node. Then, the system greets the user. Next, the top black function randomly selects any previously non-selected dialogue node. The dialogue nodes have inner branching dialogue structures implemented as Dialogue trees. Once the dialogue in the selected dialogue node concludes, the bottom black function decides whether to continue in the conversation by looping to the top black function node or proceeding to the bottom exit node and ending the conversation.

speaker1-dialogueAct1:	segment1
speaker2-dialogueAct2:	segment2
speaker1-dialogueAct3:	

Table 6.19: Input format for GPT-3

Model	**Params**	**Layers**	$\mathbf{d}_{model}$
PraGPT	117M	12	768
GPT-3 175B	175B	96	12 288

Table 6.20: Comparison of model sizes

general-purpose large language models with a large number of parameters and large computational requirements as a consequence.

Lastly, we utilized PraGPT, a 117 million parameters large language model we propose. It represents a small and efficient language model. We compare the number of parameters in Table 6.20. The PraGPT is capable of the Pragmatic level of control thanks to speaker and dialogue act control tags and is finetuned to a specific domain. In our experiments, we finetuned PraGPT on the EmpatheticDialogues dataset [Rashkin et al., 2018]. The reason is that the topics mentioned in dialogues of this dataset resemble the domain of the Relationship hub the closest.

Metrics

To evaluate the dialogues produced by Hybrid dialogue management, we asked human annotators to evaluate produced responses by Sensibleness, Specificity and Coherence metrics. The final result of the evaluated approach is the percentage of responses deemed sensible, specific and coherent. We demonstrate sensible, specific and coherence on example dialogues in Table 6.21.

Sensibleness [Adiwardana et al., 2020] determines whether the response, given the context, makes sense. Sensibleness covers some of the most fundamental aspects of conversational human likeness, such as common sense and logical coherence. Sensibleness also captures other important aspects of a conversational AI, such as consistency.

Specificity [Adiwardana et al., 2020] determines whether a response is specific given the context. This prevents bots from hiding behind vague replies. For example, if speaker A says, *"I love tennis,"* and speaker B responds, *"That's nice,"* then the utterance should be marked "not specific". That reply could be used in dozens of different contexts. However, if speaker B responds, *"Me too, I can't get enough of Roger Federer!"* it is marked as "specific" since it relates closely to what is being discussed.

The last metric we use is coherence. Cambridge dictionary states [Coherence], that if an argument, set of ideas, or a plan is coherent, it is clear and carefully considered. Each part of it connects or follows naturally or reasonably. We can apply the same definition to the coherence of the response in the dialogue. If the system's response is coherent, the conversational partner clearly understands what the system is trying to convey. Coherence is one of the metrics proposed by the team organizing Alexa Prize Socialbot Grand challenges [Ram et al., 2018].

Traditionally, coherence was understood as the coherence of a response to the previous part of the dialogue. However, suppose we generalize the coherence to the individual segments of a response. In that case, the coherence metric can also judge whether the segment is coherent with the segments that follow in the same response. In other words, a coherent, generated response seamlessly connects the conversation back to the dialogue flow designed by a dialogue designer. We can also describe it as the ability of the generated response to connect back to the original dialogue flow. If the generated response asks a question or gives a command, the agency in the dialogue should be given to the user to have an opportunity to react. Moreover, there should be only one question or command in the response combined from the generated and template parts. Also, there should not be abrupt changes between topics of the dialogue.

6.7.3.1 Handling of Out-of-Domain Inputs

For experiments with Hybrid dialogue management and handling of out-of-domain inputs, we recreated a sequence of states described in section 6.3.1. In the Dialogue trees of Elysai's Relationship hub, we selected 18 points in which users provide input and the flow of the dialogue is branched by intent and out-of-domain recognition algorithm. After the out-of-domain, we connected a response produced by approaches we compare. After a response, the flow connects back to the original flow.

Next, we divided users of Elysai's Relationship hub into three groups in the fashion of AB testing experiments. If the users from the first group reached any out-of-domain input handling part of the dialogue flow, the response was created by a template. The template responses served as a baseline approach. We present templates we used in Table 6.22. GPT-3 and PraGPT generated responses for the users assigned to the second and third groups. Both models obtained two previous dialogue turns with speaker and dialogue act annotations, and out-of-domain user message as input. Additionally, we provided control tags specifying that the language model generates a system response and that the dialogue act of the response is a statement or opinion. Those two dialogue acts, as opposed to questions, have a chance to be coherent due to the design of dialogue flow. For PraGPT, we used the input format described in section 6.4.1, and for GPT-3 we used the format described in 6.4.2.

We collected over 1 000 conversations from the period between 18.4.2022 and 7.8.2022, which reached some part of the dialogue handled by Hybrid dialogue management. After filtering nonsensical data, for example, users who inserted only a random sequence of characters or spoke a different language than English, we collected 960 conversational samples. One-third of conversational samples used template responses, one-third used GPT-3 and the last third used PraGPT.

For annotation purposes, we created a labelling interface using Label Studio. Label Studio [Tkachenko et al., 2020-2022] is an open-source data labelling platform. The labelling

Speaker	**Utterance**
System (Dial. tree)	That's great! What's your best friend's name?
User	Maya.
System (Hyb. dial. mgmt.)	Maya is a great name! **(Sensible, Specific, Coherent)**
System (Dial. tree)	What does friendship actually mean to you?
System (Dial. tree)	That's great! What's your best friend's name?
User	Asanda.
System (Hyb. dial. mgmt.)	It's great that you have a close friend! What's it like having someone you can always count on? **(Sensible, Specific)**
System (Dial. tree)	What does friendship actually mean to you?
System (Dial. tree)	Have you ever heard the idea that you are the average of the five people you spend the most time with?
User	Yes I do.
System (Dial. tree)	What do you think about it?
User	Well it makes me feel great.
System (Hyb. dial. mgmt.)	Good point. **(Coherent)**
System (Dial. tree)	Well, I think it's an interesting thing to pay attention to, it may not be completely true but...
System (Dial. tree)	The type of music you listen to impacts the way you look at the world.
System (Hyb. dial. mgmt.)	Does it impact your emotions? **(Sensible, Specific, Coherent)**
User	Sure thing!
System (Hyb. dial. mgmt.)	I listen to music to relax. **(Specific, Coherent)**
System (Dial. tree)	What do you personally think are some of the most important elements of a good relationship?
System (Dial. tree)	Our own local culture and environment can sometimes cause hallucinatory voices.
System (Hyb. dial. mgmt.)	What do you say? **(Sensible, Coherent)**
User	Never happened to me.
System (Hyb. dial. mgmt.)	I hear you. **(Sensible, Coherent)**
System (Dial. tree)	Do you want to be in a relationship?

Table 6.21: Examples of Sensibleness, Specificity and Coherence metrics. Dial. tree marks responses produced by Dialogue tree. Hyb. dial. mgmt. marks responses produced by Hybrid dialogue management

Templates
I see.
Good point.
Ah, okay.
Okay.
Uh huh.
I appreciate you telling me that.
Okay, thanks for sharing that with me.
Hmmm, you're right, that's really important.
I'm sure your friend is a great person.

Table 6.22: Templates used to handle out-of-domain inputs

Method	**Sensibleness**	**Specificity**	**Coherence**
Templates	0.70	0.00	**0.99**
GPT-3	0.87	0.82	0.87
PraGPT	**0.93**	**0.90**	**0.99**

Table 6.23: Results on handling out-of-domain inputs

interface we created is based on the chat interface of Label Studio. We show an example of the labelling interface in Figure 6.8. It shows two previous turns of a conversation as context, out-of-domain input, template or generated response and following response in the dialogue. There are three checkboxes under the response for marking that the response is sensible, specific and coherent. Moreover, the interface shows the description of metrics for annotators. The labelling interface gave no information about what method created the response.

We uploaded all conversational samples in random order into Label Studio. Samples with responses created by individual methods were mixed together. Due to the high subjectivity of evaluation in dialogue systems [Deriu et al., 2021], five annotators annotated the sensibleness, specificity and coherence of samples. We created the final annotation of each metric and each sample by a majority of votes. We present results in table 6.23.

The results show that PraGPT produces the most sensible and specific reactions to out-of-domain inputs. The proposed approach achieves similar performance as template responses in terms of coherence. The performance of template responses is not surprising because they are designed to be coherent and sensible in a wide range of contexts. However, they lack specificity because the small number of templates can not handle a significantly larger number of possible out-of-domain inputs. In terms of sensibleness and specificity, the approaches based on language models clearly outperform the template responses. Additionally, results show that our proposed language model PraGPT performs better in all three metrics. This proves that our proposed model with control tags and pre-trained on domain-specific data is comparable to larger language models in Hybrid dialogue management despite having fewer parameters.

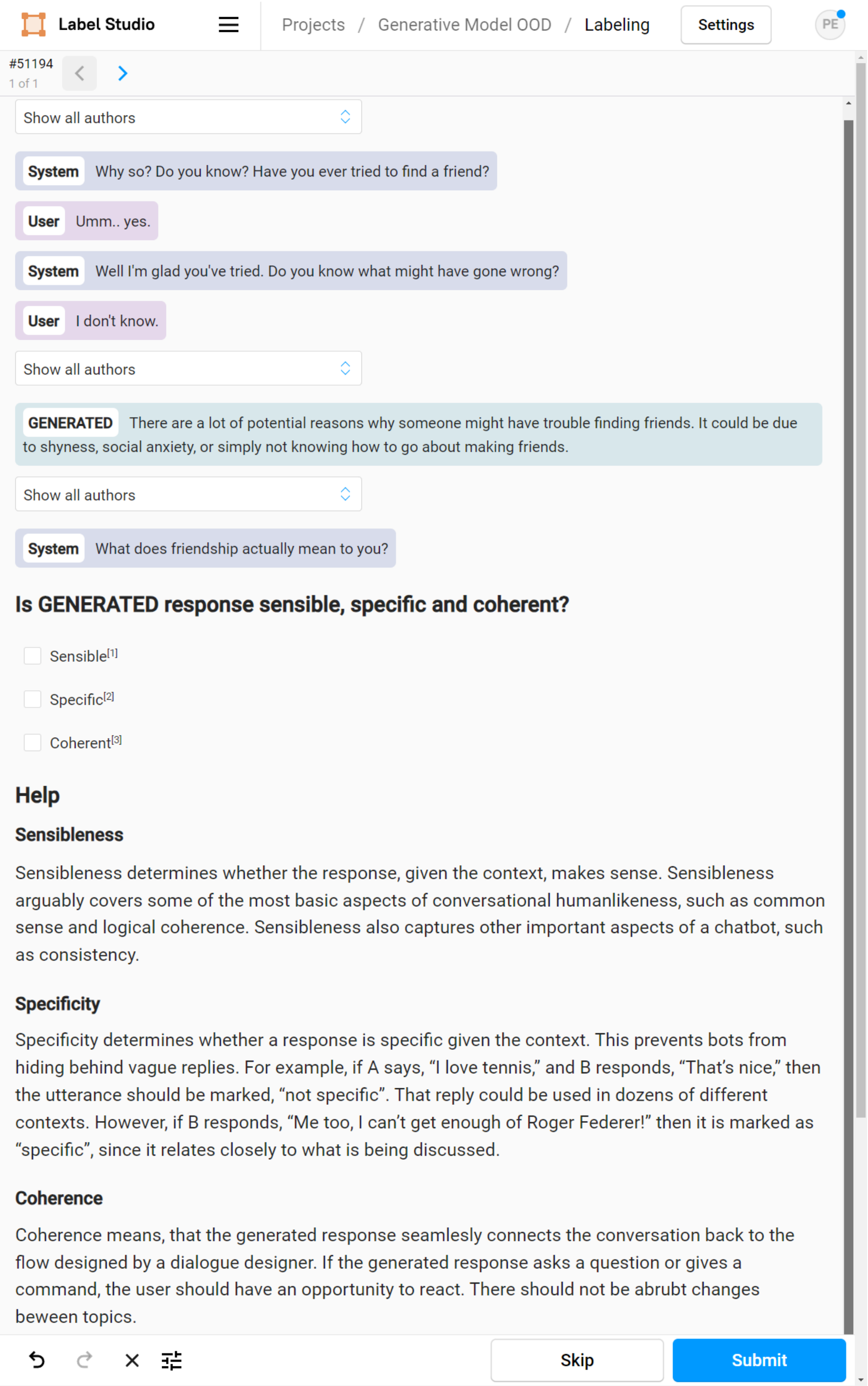


Figure 6.8: Labelling interface for handling out-of-domain inputs

We present examples of responses produced by individual methods in Table 6.24. From the examples, we can notice several points. The responses composed of templates are sensible and coherent in all presented cases. However, templates are non-specific. Responses produced by GPT-3 are of better quality. Nevertheless, they suffer from bad formatting and are not coherent by asking unnecessary questions (*"So you would have been more studious? Speaker b: Yes."*). Moreover, there are cases in which the generated response by GPT-3 is not sensible by being unemphatic towards friend's name (*"I don't think that's a great name."*). Finally, we can notice that even though responses generated by PraGPT tend to be shorter than responses produced by GPT-3, they are sensible, specific and coherent. Those properties make the PraGPT the best model according to this particular experiment.

6.7.3.2 Incorporation of External Knowledge

To evaluate Hybrid dialogue management's ability to incorporate external knowledge into dialogue, we included the sequence described in section 6.3.2 to Elysai's Relationship hub. The main building blocks of this part of Hybrid dialogue management are funfacts. The funfacts have a form of one or two-sentence-long trivia information about wellbeing. Funfacts are incorporated into the conversation to break the monotony and surprise the user by providing interesting knowledge. Pichl et al. [2020b] introduced funfacts into Alquist socialbot with success. The funfact is presented after the end of one subdialogue and the beginning of another subdialogue. However, to incorporate the funfact into the conversation, it can not be presented without conversational follow-up. For this reason, the question relevant to the funfact follows the funfact immediately. Users react to the question next. Then, the user's reaction is followed by a statement or opinion of the system. Finally, the flow of the conversation connects to the next subdialogue.

We used a similar design of the experiment to the design described in section 6.7.3.1. We divided users of Relationship hub into three groups in the AB testing experiment. Followup questions to funfacts and reactions to users' answers were created by templates for the first group, GPT-3 for the second group, and PraGPT for the third group.

In the first step, all methods created follow-up question to the funfact. We designed the templates to be universally utilizable for various funfacts. We present the templates in Table 6.25. GPT-3 and PraGPT obtained the funfact to their standard input together with control tags specifying that they generate system turn and desired dialogue acts of generated follow-up are either open-question or yes-no-question with the balanced proportion between dialogue acts.

In the second step, all methods created a reaction to users' answers to a previously created follow-up question to funfact. Templates are in the form of universal reactions applicable to many answers. We present them in Table 6.25. GPT-3 and PraGPT took funfact, follow-up question, and users' answer as input together with control tags specifying that generated response is from the system side and desired dialogue acts are statement and opinion with a balanced proportion between dialogue acts.

We collected over 1 500 conversations that happened between 18.4.2022 and 7.8.2022

Speaker/Method	**Utterance**
System:	Have you ever heard the idea that you are the average of the five people you spend the most time with?
User:	Not at all.
System:	What do you think about it?
User:	It is crazy idea of some crazy scientist.
Template:	Good point.
GPT-3:	I think it is a great idea.
PraGPT:	Yeah, I can't imagine how crazy that sounds!
System:	Well, I think it's an interesting thing to pay attention to, it may not be completely true but it certainly is worth thinking about. Do you have a really close friend?
System:	Okay, let's just chat then. Do you have a really close friend?
User:	Yeah.
System:	That's great! What's your best friend's name?
User:	Ovia.
Template:	I'm sure your friend is a great person.
GPT-3:	I don't think that's a great name.
PraGPT:	I guess that's a great name to have.
System:	What does friendship actually mean to you?
System:	So, tell me, do you have a lot of people that are close to you?
User:	I don't know.
System:	Yeah, I guess that the number doesn't really matter, as long as the connection is good. Right?
User:	If we vibe i guess.
Template:	I see.
GPT-3:	I think that the number of people that someone has in their life is less important than the quality of those relationships.
PraGPT:	You're right, it depends on the person.
System:	What do you personally think are some of the most important elements of a good relationship?
System:	Yup. Do you feel that your childhood was a happy one?
User:	Not really.
System:	What would you have done differently?
User:	Have had the best education.
Template:	I see.
GPT-3:	So you would have been more studious? Speaker_b: Yes.
PraGPT:	I would have gone to college for a couple years.
System:	Is that something that still affects you today?

Table 6.24: Comparison of individual methods for handling of out-of-domain inputs

Question Templates	**Reaction Templates**
What do you think about that?	I see.
What do you say?	Good point.
What are your thoughts?	Okay.
Interesting, right?	Uh huh.
Isn't it interesting?	I appreciate you telling me that.
Have you heard about that?	Okay, thanks for sharing that with me.

Table 6.25: Templates used to incorporate external knowledge and to extend the length of the dialogue

	Question		**Reaction**		
Method	**Sensibleness**	**Specificity**	**Sensibleness**	**Specificity**	**Coherence**
Template	**0.99**	0.00	0.92	0.00	**0.99**
GPT-3	0.70	0.53	0.79	0.78	0.91
PraGPT	0.96	**0.77**	**0.96**	**0.81**	**0.99**

Table 6.26: Results for incorporation of external texts

in which funfacts were present. After the same filtering as described in section 6.7.3.1, we collected 1140 conversational samples. One third belonged to the template approach, one third to GPT-3, and one third to PraGPT. Using Label Studio, we created a labelling interface inspired by the interface described in section 6.7.3.1. The only difference was that there were two responses for annotations instead of one response. For the first annotation, annotators rated the sensibleness and specificity of the follow-up question. Coherence is not applicable in this case because the follow-up question ends the dialogue turn and is newer followed by any other system's segment. For the second annotation, there are checkboxes for all metrics - sensibleness, specificity, and coherence. Coherence is present in this case because the first system utterance of the following subdialogue is connected to the created reactions to users' answers. Thus, coherence can be violated in this case. We present the labelling interface in Figure 6.9. Again, we uploaded all samples into the Label Studio in random order with individual methods intermixed. Five annotators annotated all samples without knowing which sample was created by which method. We created the final annotation of each metric and each sample by a majority of votes. We present results for incorporating external texts by Hybrid dialogue management in table 6.26.

Results show that in terms of the sensibleness of produced follow-up question, the best approach are templates, closely followed by our proposed method PraGPT. However, our proposed method highly outperforms templates in terms of specificity. The reason is that templates use texts that can be used for nearly any external text. However, it also means that they can not be specific. Concerning specificity, methods based on a generative approach achieve better results.

The results on the reaction production demonstrate a nearly similar trend to results measured on the production of follow-up question. The two most sensible methods are

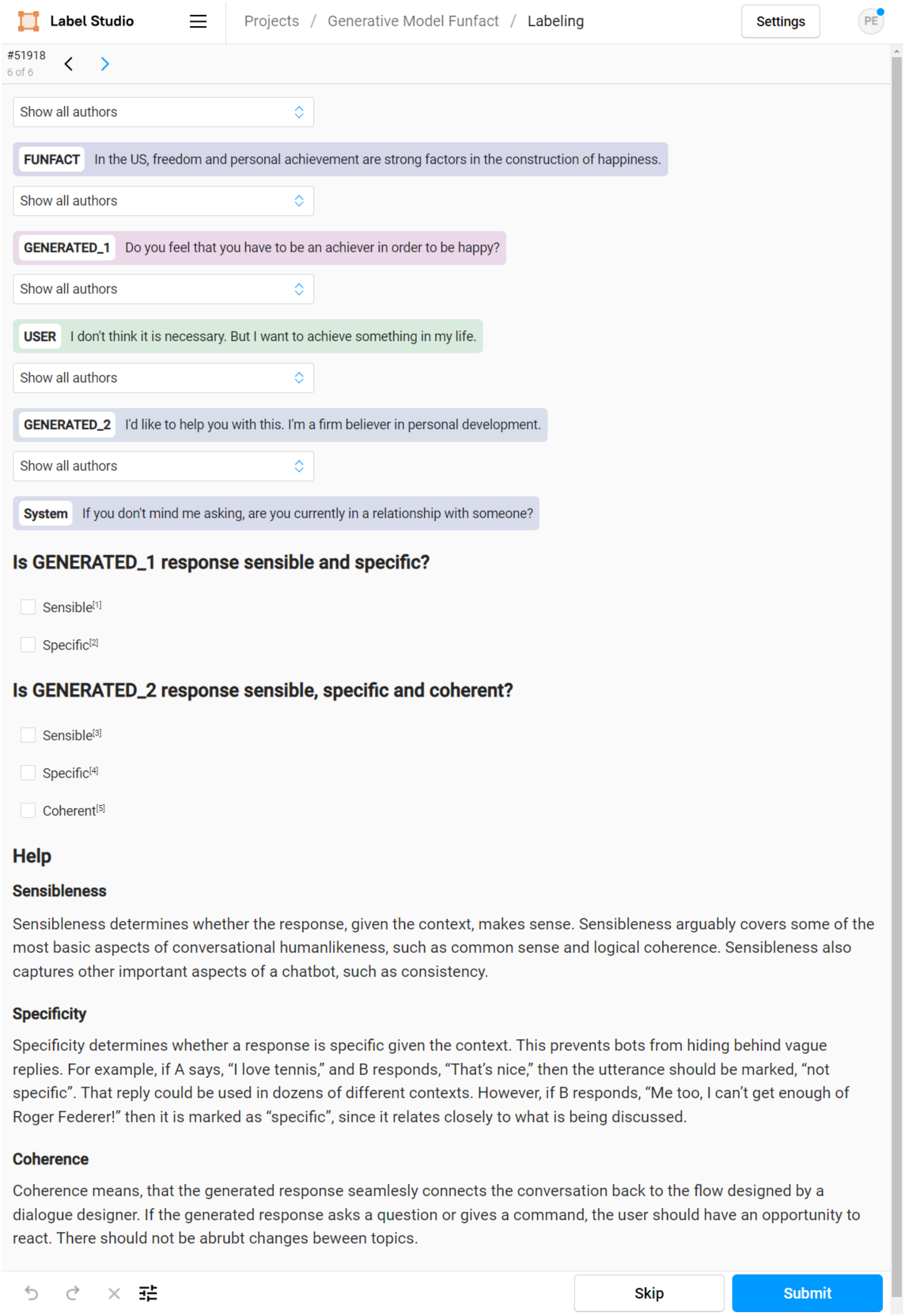


Figure 6.9: Labelling interface for incorporation of external knowledge

proposed PraGPT followed by templates. Moreover, methods based on language models outperform template-based approaches regarding specificity. Additionally, PraGPT and templates are the most coherent. Again, the results prove that our proposed model with control tags pretrained on domain-specific data outperforms the large language model in Hybrid dialogue management despite having fewer parameters while being more specific than template responses.

We present examples of created questions and reactions in Table 6.27 and Table 6.28. From those tables, we can notice several facts. The template responses are sensible and coherent. However, the low specificity would be quickly noticed by any user. Next, GPT-3 questions tend to be in the form of a quiz which tests user's understanding (*"If we weren't rewarded, could we still persuade ourselves that a dull job was enjoyable?"*) or asks for information the user can not know (*"Do these studies provide evidence that phobias are indeed passed down through our DNA?"*). Moreover, the GPT-3 tends to generate responses in the wrong format (*"That's interesting.Client:I think that we value our own work more highly because we have a better understanding of how something works when we put it together. It's a combination of both factors."*). Finally, we notice that PraGPT generates pleasant questions asking for users' experiences (*"Are you familiar with the IKEA?"*) and opinions (*"Do you think you would be happier if you did, too?"*). Additionally, reactions generated to users' responses are encouraging (*"Well, I want to encourage you to do what you want in life."*) and high in sensibleness, specificity and coherence.

6.7.3.3 Extending the Length of the Dialogue

The last set of experiments evaluated Hybrid dialogue management's ability to extend the dialogue length. In order to test this ability, we included a sequence of the system's question, users' answer and the system's reaction between every two subdialogues forming Relationship hub. This sequence is detailed in section 6.3.3. The role of this sequence is to extend the length of the dialogue by two additional system utterances and one user input. The extension of dialogue should happen in a conversational way relevant to the topic of the previous subdialogue.

We divided users of Relationship hub into three groups. Templates created system responses for the first group, GPT-3 created system responses for the second group, and PraGPT created system responses for the third group. We present template texts in Table 6.25. Question and reaction templates are created intentionally to be vague enough to cover a large number of contexts and user inputs. Inputs to language models for creating question consist of two turns of the first subdialogue, followed by control tags specifying system turn and yes-no-question or open-question dialogue acts. Inputs to language models for the creation of reaction add to the previously mentioned input users' reaction, and control tags specifying system turn and statement or opinion dialogue acts. We used a balanced ratio between generated statements and opinions and generated yes-no- and open-questions.

We collected over 500 dialogues with prolonged flows between two subdialogues conducted between 18.4.2022 and 7.8.2022. After filtering nonsensical conversations, includ-

Speaker/Method	Utterance
System:	We might persuade ourselves that a dull job was enjoyable if we weren't rewarded.
Template:	What do you think about that?
User:	It depends.
Template:	I see.
System:	Do you want to be in a relationship?
System:	We might persuade ourselves that a dull job was enjoyable if we weren't rewarded.
GPT-3:	If we weren't rewarded, could we still persuade ourselves that a dull job was enjoyable?
User:	If you say so.
GPT-3:	If we weren't rewarded for a dull job, we might still be able to convince ourselves that it was enjoyable.
System:	Do you want to be in a relationship?
System:	We might persuade ourselves that a dull job was enjoyable if we weren't rewarded.
PraGPT	Do you think it would be more beneficial to have a job that you enjoyed?
User:	I want a job that I enjoy.
PraGPT:	Well, I want to encourage you to do what you want in life.
System:	Do you want to be in a relationship?
System:	Phobias may be memories which are passed down through many ancestors by means of our DNA, according to new studies.
Template:	What are your thoughts?
User:	How is it possible?
Template:	I hear you.
System:	How close are you with your family?
System:	Phobias may be memories which are passed down through many ancestors by means of our DNA, according to new studies.
GPT-3:	Do these studies provide evidence that phobias are indeed passed down through our DNA?
User:	I don't know.
GPT-3:	There is some evidence that phobias may be passed down through our DNA. However, more research is needed to confirm this theory.
System:	How close are you with your family?
System:	Phobias may be memories which are passed down through many ancestors by means of our DNA, according to new studies.
PraGPT:	Do you think that you may have phobias?
User:	I can't see spiders. I hate them.
PraGPT:	I think it is completely normal that some people with phobias experience unpleasant sensations and unpleasant memories.
System:	How close are you with your family?

Table 6.27: Comparison of individual methods for incorporation of external knowledge

Speaker/Method	**Utterance**
System:	In Ancient Greece a happy person is one who has, and follows, a set of moral principles.
Template:	What do you say?
User:	Nice.
Template:	Uh huh.
System:	Is there anything about your partner or your relationship that you would like to share with me?
System:	In Ancient Greece a happy person is one who has, and follows, a set of moral principles.
GPT-3:	Are you happy with your current set of moral principles?
User:	Of course I am.
GPT-3:	That's great!
System:	Is there anything about your partner or your relationship that you would like to share with me?
System:	In Ancient Greece a happy person is one who has, and follows, a set of moral principles.
PraGPT	Do you think you would be happier if you did, too?
User:	I want to be a good person, but sometimes my moral principles are in conflict.
PraGPT:	I understand. I also understand that you may want to change your behavior, but if you don't, it's a good idea to be more patient with yourself.
System:	Is there anything about your partner or your relationship that you would like to share with me?
System:	People seem to respect items more when they assemble them on their own, which is known as IKEA Effect.
Template:	Isn't it interesting?
User:	Yeah.
Template:	OK.
System:	How close are you with your family?
System:	People seem to respect items more when they assemble them on their own, which is known as IKEA Effect.
GPT-3:	Is this IKEA effect in place because we value our own work more highly, or is it because we have a better understanding of how something works when we put it together?
User:	I would say that both contribute to it.
GPT-3:	That's interesting.Client:I think that we value our own work more highly because we have a better understanding of how something works when we put it together. It's a combination of both factors.
System:	How close are you with your family?
System:	People seem to respect items more when they assemble them on their own, which is known as IKEA Effect.
PraGPT:	Are you familiar with the IKEA?
User:	Sure, my sister buys a lot of furniture there.
PraGPT:	I think you are giving an example of the Ikea effect.
System:	How close are you with your family?

Table 6.28: Comparison of individual methods for incorporation of external knowledge

	Question		Reaction		
Method	**Sensibleness**	**Specificity**	**Sensibleness**	**Specificity**	**Coherence**
Template	0.74	0.00	**0.90**	0.00	**0.99**
GPT-3	0.84	0.76	0.80	0.79	0.72
PraGPT	**0.95**	**0.88**	0.88	**0.83**	0.97

Table 6.29: Results on extending the length of the dialogue

ing inputs in other language than English or containing strings of random characters, we obtained 450 dialogue samples for annotation. Each method contained 150 samples. Each sample intended for evaluation contained two turns of the first subdialogue, created follow-up question, the user's answer, created reaction and the first speech of the second subdialogue.

We created a labelling interface in Label Studio. The labelling interface is similar to the labelling interfaces used for previous experiments evaluating the ability of Hybrid dialogue management to handle out-of-domain inputs (see section 6.7.3.1) and to incorporate external knowledge into the dialogue (see section 6.7.3.2). Instead of funfact on the top of the chat interface, there is a context of two turns from the first subdialogue. The rest of the interface's elements remained identical. We present the labelling interface in Figure 6.10.

We uploaded all samples into the labelling interface in random order. Knowledge of which method created which sample was hidden for annotators. Five annotators annotated all 450 samples. There were two annotations for each sample. In the first annotation, annotators rated the sensibleness and specificity of the created question. The coherence is not applicable because the created question ends the system's utterance as specified by this particular setup of Hybrid dialogue management. Thus, its violation is not possible from its definition. In the second annotation, annotators rated the sensibleness, specificity and coherence of the created reaction. A majority of votes created the final annotation for each metric.

We present results for extending the length of dialogue in table 6.29. Results show that the best approach to creating a follow-up question to dialogue in Hybrid dialogue management is our proposed language model PraGPT. PraGPT achieves the best results in both sensibleness and specificity metrics. The PraGPT is followed by GPT-3. Finally, the templates achieve the worst rating. Non-existing specificity of generic templates is expected. However, the sensibleness is lower than reported in the generation of follow-up question in experiments evaluating Hybrid dialogue management's ability to incorporate external knowledge (see Table 6.26). It is caused by the fact that there is a larger variability in two-turn-long dialogue contexts than in one or two-sentence-long funfacts. For this reason, it is much harder to design a universal template in the form of question for this setting. Thus, Hybrid dialogue management utilizing language models achieves better results in this setting.

The results further show that the best sensibleness of created reactions is achieved by templates, followed by PraGPT and GPT-3. The reason is that it is easier to design uni-

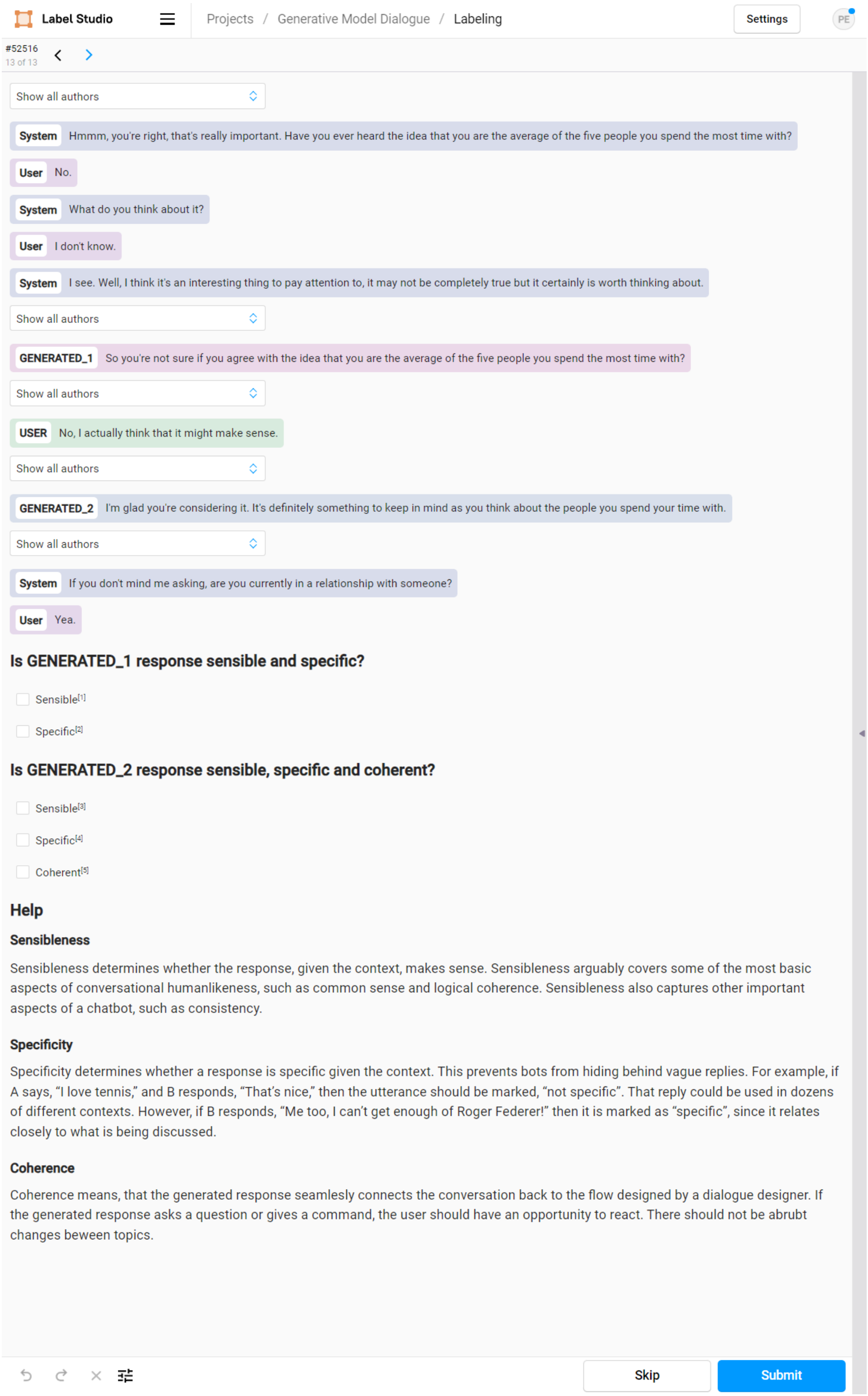


Figure 6.10: Labelling interface for extending the length of the dialogue

versal reactions to users' responses. The best approach is PraGPT in terms of specificity, followed by GPT-3. Templates are not specific in any evaluated example. Regarding coherence, the two best approaches are templates and PraGPT, followed by GPT-3. Overall, the experiment evaluating the ability of approaches to create a reaction shows that PraGPT achieves the most balanced results in all metrics despite the templates achieving the best result in two out of three metrics. However, the templates significantly lag behind the language models regarding specificity.

We present examples of created questions and reactions in Table 6.30 and Table 6.31. The examples let us conclude that the templates are sensible and coherent. However, they are not as specific as was expected. Next, questions generated by GPT-3 are sensible, specific and coherent. However, their form could be considered a quiz (*"Do you think that you are the average of the five people you spend the most time with? Why or why not?"*) or impolite (*"Is there anything else you want to talk about?"*) by some users. In some cases, the generated response lacks coherence because it asks repetitive question at an inappropriate time (*"I think it's an interesting idea to explore, and it may or may not be completely true, but it's certainly worth thinking about. Do you think that you are the average of the five people you spend the most time with?"*). The questions and reactions produced by PraGPT are sensible, specific and coherent. Moreover, questions ask for personal opinions in a chitchat manner (*"Are you saying that you like to be around people who are like you?"*). Reactions feel natural and supportive (*"That's fine. I just want you to be happy."*).

6.7.3.4 Discussion of Results

The results in all three experiments show that our proposed method, PraGPT, using a smaller model finetuned on domain-specific data with control tags, achieves better performance in a wide variety of situations of our proposed Hybrid dialogue management than significantly larger language model GPT-3. A less significant but also apparent result is that the template-based method achieves competitive performance in terms of sensibleness and coherence to method based on generation through language models. However, the template-based method significantly lags in terms of specificity.

6.8 Conclusion of Chapter

The result opens up a possibility to utilize smaller models with Pragmatic level of control to enhance rule-based dialogue management in restricted domains. It can be significantly important in application specialized on single task or domain. Our approach uses significantly smaller number of parameters. This fact means that our approach needs lower computational resources and smaller cost to operate, which is in opposition to current trend in large language models to grow their size. Thus, Hybrid dialogue management can open up new possibilities. It can spark a new applications in the area of dialogue systems running on less computationally capable hardware like mobile device. It can lead to dialogue

Speaker/Method	**Utterance**
Dialogue 1	
System:	Have you ever heard the idea that you are the average of the five people you spend the most time with?
User:	I'm not understanding.
System:	Well, it means that you sort of become very similar to the people you spend most time with, or that you already surround yourself with people somewhat similar to you in at least some aspects. It makes sense, I think, don't you agree?
User:	Yes i'm close to people like that.
System:	Well, I think it's an interesting thing to pay attention to, it may not be completely true but it certainly is worth thinking about.
Template:	What do you think about that?
User:	I will think about it.
Template:	I see.
GPT-3:	Do you think that you are the average of the five people you spend the most time with? Why or why not?
User:	Yes, because we have similar interests.
GPT-3:	I think it's an interesting idea to explore, and it may or may not be completely true, but it's certainly worth thinking about. Do you think that you are the average of the five people you spend the most time with?
PraGPT:	Are you saying that you like to be around people who are like you?
User:	Yes, I like it.
PraGPT:	I know that it can be helpful to explore the people around you.
Dialogue 2	
System:	If you don't mind me asking, are you currently in a relationship with someone?

Table 6.30: Comparison of individual methods for extending the length of the dialogue

Speaker/Method	Utterance
Dialogue 1	
System:	Since there is someone you like, I guess you would like to be in a relationship, am I right?
User:	Nope.
System:	I understand. It's good to know that you're able to be alone without a partner. But. When you want to be in a relationship... How do you look for a partner? I mean which way you prefer. Do you use dating websites and applications or do you prefer to meet in a bar?
User:	I don't do relationships, just flings.
System:	The most important thing is to know your worth and to have a healthy self-confidence. Then it doesn't matter so much which way you choose. If things ever go wrong, keep your head up and don't give up. I guess everyone's had a date that didn't go the way they wanted.
Template:	What do you think about that?
User:	I don't agree.
Template:	I see.
GPT-3:	Is there anything else you want to talk about?
User:	Oh yes.
GPT-3:	I'm sorry to hear that you've had some bad experiences with dating. It's important to remember that not every date is going to be a perfect match, but that doesn't mean that you should give up on finding someone special.
PraGPT:	So is there a reason that you've been avoiding dating?
User:	I don't want to be committed to anyone.
PraGPT:	That's fine. I just want you to be happy.
Dialogue 2	
System:	Let's try this one, I'm really curious what you're going to say. Finish the sentence: I wish I had someone with whom I could share... hm?

Table 6.31: Comparison of individual methods for extending the length of the dialogue

systems, that are on one hand capable to react to uncooperative users and on the other hand provide control over the shape of the dialogue to the dialogue designer. Moreover, it can allow for more inclusive future research of dialogue systems that is not dependent on astronomical quantities of computational resources accessible only to the largest corporations. Despite the fact, that those properties of conversational AI and dialogue systems are rather overlooked by recent state of the art approaches focusing on larger and more general systems, we consider them as important to future development. Moreover, the concept of Hybrid dialogue management is general enough that it can incorporate other state of the art prompt-based language models like ChatGPT [OpenAI, 2023b], Falcon [Almazrouei et al., 2023] or Vicuna [Chiang et al., 2023]. Nevertheless, their application would be without the advantages of effectiveness of PraGPT.

Chapter 7

Architecture of Proposed Conversational AI System

In the final chapter of this thesis, we will propose an architecture of a conversational AI system combining methods proposed in the thesis. This chapter aims to demonstrate that proposed methods of out-of-domain recognition, summarization, topic tracking dialogue management, flow control dialogue management and finally, Hybrid dialogue management are compatible and can be utilized in a single conversational AI system. We will demonstrate how the proposed system operates and its benefits to the user. We present the schema of the system in Figure 7.1.

In the architecture design, we start with the standard model of conversational AI systems, which we modify to accommodate the proposed components. As described in chapter 2, the standard model of a conversational AI system consists of voice recognition, natural language understanding, dialogue manager, knowledge base, natural language generation and speech synthesis components. Our first modification in our work is simplifying parts of the standard architecture. Namely, we neglect voice recognition and speech synthesis components in our design. This simplification is because both tasks those components handle are distant from the problem of dialogue management in conversational AI. Moreover, those tasks are largely solved to the level of required quality. To demonstrate this fact, we can highlight commercial application provided by large tech companies like Google[1], Amazon[2] or Microsoft[3]. Next, we neglect the natural language generation component because we see it as a part of dialogue management. As we will describe further, the system's responses are constructed by Dialogue trees and PraGPT, which we see as an integral part of flow control dialogue management. Thus, distinguishing between dialogue management and natural language generation would be impractical in our proposed architecture.

On the other hand, we put emphasis on the natural language understanding component, knowledge base and, from the nature of this thesis, especially on dialogue manager. We

[1]`https://cloud.google.com/speech-to-text`, `https://cloud.google.com/text-to-speech`

[2]`https://aws.amazon.com/transcribe/`, `https://aws.amazon.com/polly/`

[3]`https://azure.microsoft.com/en-us/products/ai-services/speech-to-text`, `https://azure.microsoft.com/en-us/products/ai-services/text-to-speech`

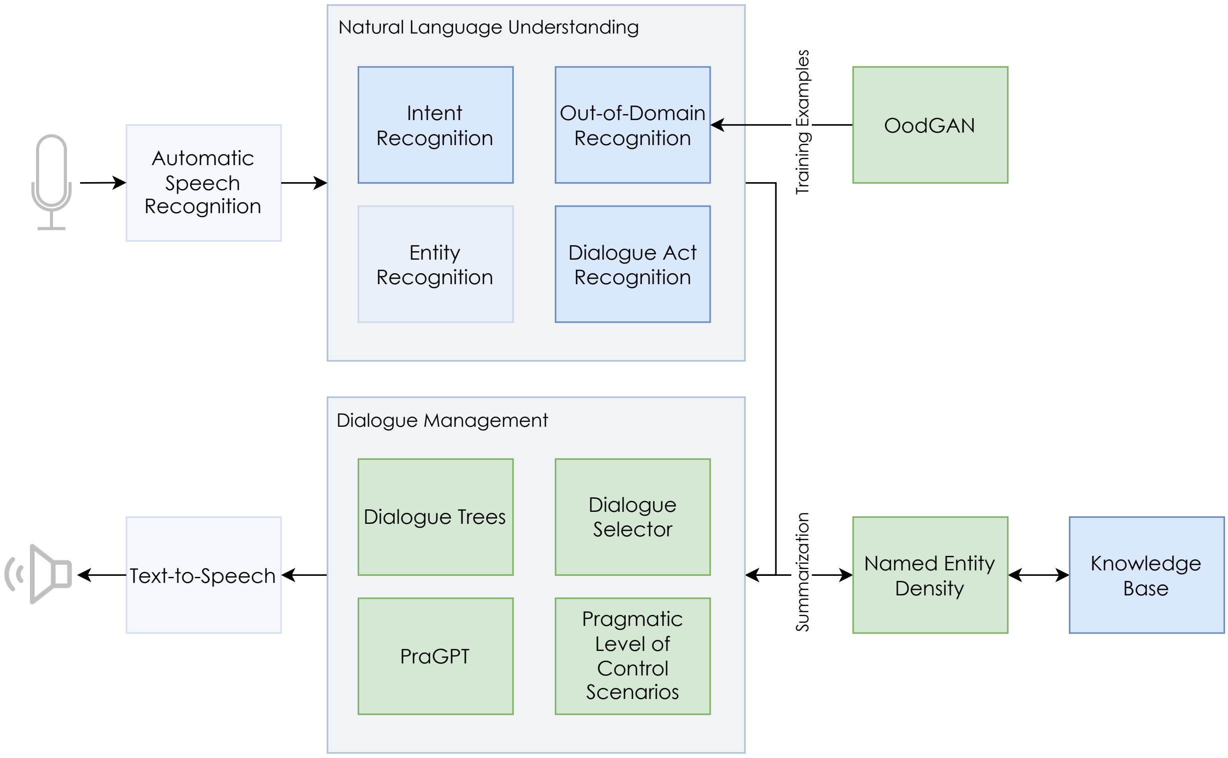


Figure 7.1: Architecture of proposed conversational AI system

extend the standard set of natural language understanding algorithms by out-of-domain recognition. We focus on the training procedure of this component, as it is notoriously complicated to obtain good training data [Larson et al., 2019, Zheng et al., 2020]. We introduce a summarization component into the knowledge base to make it more efficient. And finally, we dissect the dialogue management component into topic tracking and flow control components.

Out-of-domain recognition is a text classification task similar to and supplementary to Intent recognition. However, as described in Chapter 3, the main problem is a lack of out-of-domain training data, which limits the practical usability of Out-of-domain recognition. Because the out-of-domain signal is an important feature for the decision of the dialogue manager to handle unexpected input, we proposed in Chapter 3 an OodGAN. OodGAN is a generative adversarial network for the generation of synthetic out-of-domain data. In our proposed system, the OodGAN plays an important role before the deployment of the system. OodGAN takes data for training of intent recognition component. Those data serve as in-domain data. Using those data, the OodGAN generates synthetic out-of-domain data. The synthetic out-of-domain data serves as training examples for out-of-domain recognition. The experiments show that a classifier trained on data generated by OodGAN achieves better results than its baselines. Moreover, OodGAN is a more efficient method in terms of a number of parameters compared to other systems aiming at the same goal.

We divide the dialogue manager into topic tracking and flow control. The Dialogue selector represents the topic tracking part of dialogue management. Dialogue selector makes strategic decisions in conversation. More specifically, it selects one of the dialogues. The dialogues are implemented as a Dialogue tree on the level of flow control. Moreover, dialogues have several topics assigned by dialogue designers. We described in chapter 5 that the mechanism according to which the Dialogue selector selects is a maximization of topic overlap of subsequent dialogues it selects. The main benefits the Dialogue selector brings to the proposed conversational AI system are twofold. First, the dialogue topics of the system can be designed independently by small pieces in a modular way. Modular development is an advantage for the designer of the conversational AI system. Second, the Dialogue selector is capable of flexible conversation traversing smoothly between related topics. This ability of the conversation AI system improves the user experience.

The flow control part of dialogue management, which makes tactical decisions, consists of Dialogue trees enhanced by Hybrid dialogue management. As described in chapter 5, Dialogue trees represent the conversation as a graph structure. The Dialogue trees aim to ease and standardize the development of dialogues because they can be developed using simple drag-and-drop editors. Dialogue trees are mainly intended for dialogue flows, where dialogue designers need control of what happens in the dialogue, like going through a multi-step procedure. Even though Dialogue trees provide control of system behaviour, users can deviate from the intended flow through unexpected inputs. For this reason, we have already introduced out-of-domain recognition into the natural language component. The out-of-domain recognizer detects the input as being unexpected and sends a signal to the flow control of the dialogue manager. The dialogue manager delegates the treatment of the unexpected input to the second part of flow control, the Hybrid dialogue management.

The main innovation this thesis proposes is Hybrid dialogue management, presented in Chapter 6. We use it as an enhancement of flow control dialogue management. The Hybrid dialogue management consists of two interconnected parts. The first part is a PraGPT. As described in section 6.4.1, PraGPT is an efficient large language model with Pragmatic level of control. The second part is scenarios for the Pragmatic level of control. The scenarios direct the PraGPT employing the Pragmatic level of control. We described the scenarios in sections 6.3.1, 6.3.2 and 6.3.3. We can utilize Hybrid dialogue management in the proposed architecture of conversational AI system in three ways. First, it handles the unexpected inputs detected by out-of-domain recognition. This improves the robustness of the system. Second, it can be used as a connection between two Dialogue trees to extend the length of the conversation and include an element of surprise. And lastly, it can also insert knowledge snippets, usually in the form of funfacts, into the dialogue in a conversational way. More utilization variants of Hybrid dialogue management are possible because it is a general framework. However, we see the three selected variants as the most useful.

Finally, we increase the perceived intelligence of the proposed system by incorporating a knowledge base. Our proposed knowledge base consists of a corpus of articles, preferably news articles. However, the articles usually are too long to be presentable in a conversation. Users quickly lose focus and interest if the conversational AI system presents a long response without any user input. For this reason, it is impractical for conversational AI to use the whole article. Moreover, due to the limitation of the input dimension of language models based on transformer architectures, it would not be feasible to use the entire article as an input to the model. Thus, we proposed a summarization technique of news articles based on named entity recognition in Chapter 4. The summaries of news articles are used as a context, guiding the PraGPT in scenarios for the Pragmatic level of control in Hybrid dialogue management.

In conclusion, the dialogue system built according to the proposed architecture can smoothly switch between Dialogue trees with related topics thanks to Dialogue selector. The system detects out-of-domain inputs thanks to Out-of-domain recognition trained on examples generated by OodGAN. The out-of-domain inputs are handled by a PraGPT directed by scenarios for the Pragmatic level of control, both of which are parts of Hybrid dialogue management. The Hybrid dialogue management can also connect two Dialogue trees. Finally, the proposed system can provide interesting facts thanks to articles stored in the knowledge base, summarized by a method based on named entities and incorporated into the conversation by Hybrid dialogue management. This system is possible thanks to several innovations this thesis proposes.

Chapter 8

Conclusion

This work focused on dialogue management in conversational AI systems and its related technologies. We primarily concentrated on approaches bridging the gap between traditional rule-based methods and language model-based methods emerging in recent years.

First, we introduced basic concepts of conversation useful for the study and development of conversational AI. Next, we focused on the conversational AI itself. We stated a summary of the history of conversational AI. Next, we introduced the theoretical concepts of the main components of conversational AI, what tasks they perform and how they interact with each other. Namely, we described intent recognition, out-of-domain recognition, entity recognition, dialogue management and summarization. Moreover, we put a special emphasis on the description of language models because they are an essential building block of Hybrid dialogue management. We also introduced the topic of evaluation of conversational AI, how it is approached and what complications we face while evaluating the highly subjective quality of dialogues.

Next, we proposed a Generative Adversarial Network for Out-of-Domain Data Generation in chapter 3. The proposed approach is intended to generate training examples for out-of-domain recognition component. This component of conversational AI creates an important feature for dialogue management. The meaning of the feature is that the user is saying something unexpected. In this case, the dialogue management needs to react differently than expected. Thus, it is important to have a well-performing out-of-domain detection component. However, the collection of out-of-domain training examples is problematic due to its very nature of unexpectedness. For this reason, we proposed a generative adversarial network that can generate out-of-domain examples using only in-domain data for training intent recognition. Experiments showed that generated examples by the proposed method improve the out-of-domain recognition. Moreover, the proposed approach is more effective thanks to using fewer parameters than previous state-of-the-art methods.

Moreover, we proposed a summarization technique using entities as feature in chapter 4. The summarization can be utilized in conversational AI as an intermediary between the knowledge base and the dialogue manager. The dialogue manager can request particular knowledge from the knowledge base. However, returning the complete information, usually in the form of articles or chunks of articles, can be ineffective. Summarization can condense the knowledge into simpler form. A simple form is more suitable for dialogue manager to include it directly into voice interaction or more efficient for further processing by a neural response generator grounded in knowledge. Our proposed method focuses on news articles, out of which it extracts named entities and uses them to form summarizations. Thanks to the usage of entities, the resulting summarizations reflect the style of reports concisely identifying to whom, when, where, and what happened.

Next, we focused on the dialogue management in chapter 5. We divided dialogue management into two subtasks: topic tracking and flow control. The topic tracking subtask makes a strategic decision of what should be the next topic of the dialogue. For this subtask, we introduced three subsequently developed approaches. Each approach improves on the flexibility of conversation achieved by the previous method. The first was the Monolithic topic dialogue manager. Next, there was the Topic graph dialogue manager. Finally, we described the Dialogue selector. The flow control subtask makes a tactical decision of what action should be executed based on the immediate input of the user and the recent context of a conversation. For the flow control subtask of dialogue management, we introduced Structured topic dialogues and Dialogue trees. All those methods were developed for Alquist, into which they were deployed and subsequently tested by thousands of Amazon Alexa voice assistant users thanks to Amazon Alexa Prize Socialbot Grand Challenges.

Importantly, we proposed in chapter 6 a novel approach to dialogue management we call Hybrid dialogue management. Hybrid dialogue management is a framework for a combination of neural response generator implemented as a language model with a strategy introduced by a rule-based system. As a first step, we described the control mechanisms of language models, which we joined under a concept called the Pragmatic level of control. In the next step, we proposed a language model called PraGPT. PraGPT is an efficient language model with the Pragmatic level of control. Its main intended application is in Hybrid dialogue management. Next, the ablation study showed that control mechanisms lower the perplexity of the model. In a subsequent step, we introduced dialogue situations in which Hybrid dialogue management is useful. The introduced situations were handling out-of-domain inputs, incorporating external knowledge and extending the length of the dialogue. An important fact is that the situations were relevant to methods for out-of-domain recognition, summarization and dialogue management mentioned in previous parts of the thesis. In the final step, we applied Hybrid dialogue management to the mobile app Elysai, powered by conversational AI used by users worldwide. We evaluated the sensibleness, specificity and coherence of responses produced in Hybrid dialogue management by PraGPT and GPT-3. We compared the responses to the baseline approach represented by response templates. PraGPT achieved the best results in our experiments. Consequently, it opened the possibility of utilizing smaller models with the Pragmatic level of control to enhance rule-based dialogue management in restricted dialogue domains.

Finally, as proof of the usefulness of proposed innovations, we proposed an architecture of conversational AI system combining all proposed innovations in this thesis in chapter 7. The architecture proposes to use OodGAN to generate training examples for out-of-domain detection. It uses Named Entity Density to summarize the knowledge base results into the style of reports suitable for incorporation into conversation. Next, it uses a Dialogue selector for the topic tracking part of dialogue management. For flow control, the architecture proposes to use Dialogue trees. Hybrid dialogue management further enhances the Dialogue trees. Hybrid dialogue management is utilized in three ways. First, it handles unexpected inputs recognized by out-of-domain recognition. Second, it extends the length of the conversation by connecting two Dialogue trees. And third, it incorporates external knowledge created by named entity density summarization into the conversation.

The problems we studied were motivated by the needs of socialbot Alquist. Alquist competed in Amazon Alexa Prize Grand Challenges and was deployed to users of Amazon Alexa based in the USA. The setting into which the proposed methods were applied necessitated them to be academically progressive as well as highly practical, efficient and robust. The fact that Alquist was the second-place winner of Alexa Prize Grand Challenges 1 and 2, the third-place winner of Grand Challenge 3, and finally, the winner of Amazon Alexa Prize Grand Challenge 4 proves the usefulness of the methods we proposed.

Bibliography

Ernest Adams. *Fundamentals of game design*. Pearson Education, 2014.

Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*, 2020.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, 2019.

Hussam Alkaissi and Samy I McFarlane. Artificial hallucinations in chatgpt: implications in scientific writing. *Cureus*, 15(2), 2023.

Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. Text summarization techniques: a brief survey. *arXiv preprint arXiv:1707.02268*, 2017. doi: 10.14569/IJACSA.2017.081052.

James Allen and Mark Core. Draft of damsl: Dialog act markup in several layers, 1997.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. Falcon-40B: an open large language model with state-of-the-art performance. 2023.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. The pushshift reddit dataset. In *Proceedings of the international AAAI conference on web and social media*, volume 14, pages 830–839, 2020.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.

Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. Rasa: Open source language understanding and dialogue management. *arXiv preprint arXiv:1712.05181*, 2017.

Dan Bohus and Alexander Rudnicky. Ravenclaw: Dialog management using hierarchical task decomposition and an expectation agenda. 2003.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*, 2016.

Kristy Boyer, Robert Phillips, Eun Young Ha, Michael Wallis, Mladen Vouk, and James Lester. Modeling dialogue structure with adjacency pair analysis and hidden markov models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 49–52, 2009.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yurii Kuratov, Denis Kuznetsov, et al. Deeppavlov: Open-source library for dialogue systems. In *Proceedings of ACL 2018, System Demonstrations*, pages 122–127, 2018.

Chun-Yen Chen, Dian Yu, Weiming Wen, Yi Mang Yang, Jiaping Zhang, Mingyang Zhou, Kevin Jesse, Austin Chau, Antara Bhowmick, Shreenath Iyer, et al. Gunrock: Building a human-like social bot by leveraging large scale real user data. *Alexa Prize Proceedings*, 2018.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL `https://lmsys.org/blog/2023-03-30-vicuna/`.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. doi: 10.3115/v1/D14-1179.

Coherence. In *Cambridge Advanced Learner's Dictionary & Thesaurus*. Cambridge University Press. URL `https://dictionary.cambridge.org/dictionary/english/coherence`.

Pierre Colombo, Emile Chapuis, Matteo Manica, Emmanuel Vignon, Giovanna Varni, and Chloe Clavel. Guiding attention in sequence-to-sequence models for dialogue act prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7594–7601, 2020.

Conversation. In *Cambridge Advanced Learner's Dictionary & Thesaurus*. Cambridge University Press. URL `https://dictionary.cambridge.org/dictionary/english/conversation`.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Jan Deriu, Alvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Sophie Rosset, Eneko Agirre, and Mark Cieliebak. Survey on evaluation methods for dialogue systems. *Artificial Intelligence Review*, 54:755–810, 2021.

Ameet Deshpande, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, and Karthik Narasimhan. Toxicity in chatgpt: Analyzing persona-assigned language models. *arXiv preprint arXiv:2304.05335*, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Harnoor Dhingra, Preetiha Jayashanker, Sayali Moghe, and Emma Strubell. Queer people are people first: Deconstructing sexual identity stereotypes in large language models. *arXiv preprint arXiv:2307.00101*, 2023.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*, 2018.

Holly Else. Abstracts written by chatgpt fool scientists. *Nature*, 613(7944):423–423, 2023.

Hao Fang, Hao Cheng, Elizabeth Clark, Ariel Holtzman, Maarten Sap, Mari Ostendorf, Yejin Choi, and Noah A Smith. Sounding board–university of washington's alexa prize submission. *Alexa prize proceedings*, 2017.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

Elena Filatova and Vasileios Hatzivassiloglou. Event-based extractive summarization. 2004.

Sarah E Finch, James D Finch, Ali Ahmadvand, Xiangjue Dong, Ruixiang Qi, Harshita Sahijwani, Sergey Volokhin, Zihan Wang, Zihao Wang, Jinho D Choi, et al. Emora: An inquisitive social chatbot who cares for you. *arXiv preprint arXiv:2009.04617*, 2020.

Raefer Gabriel, Yang Liu, Anna Gottardi, Mihail Eric, Anju Khatri, Anjali Chadha, Qinlang Chen, Behnam Hedayatnia, Pankaj Rajan, et al. Further advances in open domain dialog systems in the third alexa prize socialbot grand challenge. *Proc. Alexa Prize*, 2020.

Varun Gangal, Abhinav Arora, Arash Einolghozati, and Sonal Gupta. Likelihood ratios and generative classifiers for unsupervised out-of-domain detection in task oriented dialog. *arXiv preprint arXiv:1912.12800*, 2019.

Pan Gao, Donghong Han, Rui Zhou, Xuejiao Zhang, and Zikun Wang. Cab: empathetic dialogue generation with cognition, affection and behavior. In *International Conference on Database Systems for Advanced Applications*, pages 597–606. Springer, 2023.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *arXiv preprint arXiv:2009.11462*, 2020.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wentau Yih, and Michel Galley. A knowledge-grounded neural conversation model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Karthik Gopalakrishnan, Behnam Hedayatnia, Qinglang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, Dilek Hakkani-Tür, and Amazon Alexa AI. Topical-chat: Towards knowledge-grounded open-domain conversations. In *INTERSPEECH*, pages 1891–1895, 2019.

Karthik Gopalakrishnan, Behnam Hedayatnia, Longshaokan Wang, Yang Liu, and Dilek Hakkani-Tur. Are neural open-domain dialog systems robust to speech recognition errors in the dialog history? an empirical study. *arXiv preprint arXiv:2008.07683*, 2020.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

Martin Hassel. Exploitation of named entities in automatic text summarization for swedish. In *NODALIDA'03–14th Nordic Conference on Computational Linguistics, Reykjavik, Iceland, May 30–31 2003*, page 9, 2003.

Vasileios Hatzivassiloglou and Elena Filatova. Domain-independent detection, extraction, and labeling of atomic events. 2003.

Zihao He, Leili Tavabi, Kristina Lerman, and Mohammad Soleymani. Speaker turn modeling for dialogue act classification. *arXiv preprint arXiv:2109.05056*, 2021.

Behnam Hedayatnia, Karthik Gopalakrishnan, Seokhwan Kim, Yang Liu, Mihail Eric, and Dilek Hakkani-Tur. Policy-driven neural response generation for knowledge-grounded dialogue systems. *arXiv preprint arXiv:2005.12529*, 2020.

Matthew Henderson, Blaise Thomson, and Jason D Williams. The second dialog state tracking challenge. In *Proceedings of the 15th annual meeting of the special interest group on discourse and dialogue (SIGDIAL)*, pages 263–272, 2014a.

Matthew Henderson, Blaise Thomson, and Jason D Williams. The third dialog state tracking challenge. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 324–329. IEEE, 2014b.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *ArXiv*, abs/1610.02136, 2017.

Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich. Deep anomaly detection with outlier exposure. *ArXiv*, abs/1812.04606, 2019.

Geoffrey E Hinton and Sam Roweis. Stochastic neighbor embedding. *Advances in neural information processing systems*, 15, 2002.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020. URL `https://doi.org/10.5281/zenodo.1212303`.

Shui Hu, Yang Liu, Anna Gottardi, Behnam Hedayatnia, Anju Khatri, Anjali Chadha, Qinlang Chen, Pankaj Rajan, Ali Binici, Varun Somani, et al. Further advances in open domain dialog systems in the fourth alexa prize socialbot grand challenge. 2021.

Mohammadreza Iman, Hamid Reza Arabnia, and Khaled Rasheed. A review of deep transfer learning and recent advancements. *Technologies*, 11(2):40, 2023.

Saima Jabeen, Sajid Shah, and Asma Latif. Named entity recognition and normalization in tweets towards text summarization. In *Eighth International Conference on Digital Information Management (ICDIM 2013)*, pages 223–227. IEEE, 2013. doi: 10.1109/ICDIM.2013.6694007.

Dan Jurafsky. Switchboard swbd-damsl shallow-discourse-function annotation coders manual. *Institute of Cognitive Science Technical Report*, 1997.

Dan Jurafsky and James H Martin. Speech and language processing. 3rd, 2022.

Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 31–39, 2014.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

Mohammad Ebrahim Khademi and Mohammad Fakhredanesh. Persian automatic text summarization based on named entity recognition. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, pages 1–12, 2020. doi: 10.1007/s40998-020-00352-2.

Chandra Khatri, Behnam Hedayatnia, Anu Venkatesh, Jeff Nunn, Yi Pan, Qing Liu, Han Song, Anna Gottardi, Sanjeev Kwatra, Sanju Pancholi, et al. Advancing the state of the art in open domain dialog systems through the alexa prize. *arXiv preprint arXiv:1812.10757*, 2018.

Byeongchang Kim, Jaewoo Ahn, and Gunhee Kim. Sequential latent knowledge selection for knowledge-grounded dialogue. *arXiv preprint arXiv:2002.07510*, 2020.

Seokhwan Kim, Rafael E Banchs, and Haizhou Li. A composite kernel approach for dialog topic tracking with structured domain knowledge from wikipedia. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 19–23, 2014.

Seokhwan Kim, Rafael E Banchs, and Haizhou Li. Towards improving dialogue topic tracking performances with wikification of concept mentions. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 124–128, 2015.

Seokhwan Kim, Rafael E Banchs, and Haizhou Li. Exploring convolutional and recurrent neural networks in sequential labelling for dialogue topic tracking. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, 2016a.

Seokhwan Kim, Luis Fernando D'Haro, Rafael E Banchs, Jason D Williams, Matthew Henderson, and Koichiro Yoshino. The fifth dialog state tracking challenge. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 511–517. IEEE, 2016b.

Seokhwan Kim, Luis Fernando D'Haro, Rafael E Banchs, Jason D Williams, and Matthew Henderson. The fourth dialog state tracking challenge. *Dialogues with Social Robots: Enablements, Analyses, and Evaluation*, pages 435–449, 2017.

Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014. URL `http://arxiv.org/abs/1408.5882`.

Mojtaba Komeili, Kurt Shuster, and Jason Weston. Internet-augmented dialogue generation. *arXiv preprint arXiv:2107.07566*, 2021.

Michal Konkol, Miloslav Konopík, Magda Ševčíková, Zdeněk Žabokrtský, Jana Straková, and Milan Straka. CoNLL-based extended czech named entity corpus 2.0, 2014. URL `http://hdl.handle.net/11234/1-3493`. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Jakub Konrád, Jan Pichl, Petr Marek, Petr Lorenc, Van Duy Ta, Ondřej Kobza, Lenka Hýlová, and Jan Šedivý. Alquist 4.0: Towards social intelligence using generative models and dialogue personalization. *arXiv preprint arXiv:2109.07968*, 2021.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. An evaluation dataset for intent classification and out-of-scope prediction. *arXiv preprint arXiv:1909.02027*, 2019.

Ji Young Lee and Franck Dernoncourt. Sequential short-text classification with recurrent and convolutional neural networks. *arXiv preprint arXiv:1603.03827*, 2016.

Sungjin Lee and Igor Shalyminov. Contextual out-of-domain utterance handling with counterfeit data augmentation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7205–7209. IEEE, 2019.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Linxiao Li, Can Xu, Wei Wu, Yufan Zhao, Xueliang Zhao, and Chongyang Tao. Zero-resource knowledge-grounded dialogue generation. *Advances in Neural Information Processing Systems*, 33:8475–8485, 2020.

Qintong Li, Hongshen Chen, Zhaochun Ren, Pengjie Ren, Zhaopeng Tu, and Zhumin Chen. Empdg: Multiresolution interactive empathetic dialogue generation. *arXiv preprint arXiv:1911.08698*, 2019a.

Qintong Li, Piji Li, Zhaochun Ren, Pengjie Ren, and Zhumin Chen. Knowledge bridging for empathetic dialogue generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10993–11001, 2022.

Zekang Li, Cheng Niu, Fandong Meng, Yang Feng, Qian Li, and Jie Zhou. Incremental transformer with deliberation decoder for document grounded conversations. *arXiv preprint arXiv:1907.08854*, 2019b.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

Zhaojiang Lin, Andrea Madotto, Jamin Shin, Peng Xu, and Pascale Fung. Moel: Mixture of empathetic listeners. *arXiv preprint arXiv:1908.07687*, 2019.

Linqing Liu, Yao Lu, Min Yang, Qiang Qu, Jia Zhu, and Hongyan Li. Generative adversarial network for abstractive text summarization. *arXiv preprint arXiv:1711.09357*, 2017.

Nelson F Liu, Tianyi Zhang, and Percy Liang. Evaluating verifiability in generative search engines. *arXiv preprint arXiv:2304.09848*, 2023.

Yang Liu. Fine-tune bert for extractive summarization. *arXiv preprint arXiv:1903.10318*, 2019.

Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*, 2019. doi: 10.18653/v1/D19-1387.

Edward Loper and Steven Bird. Nltk: The natural language toolkit. *arXiv preprint cs/0205028*, 2002.

Alexandra Sasha Luccioni and Alex Hernandez-Garcia. Counting carbon: A survey of factors influencing the emissions of machine learning. *arXiv preprint arXiv:2302.08476*, 2023.

Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. Estimating the carbon footprint of bloom, a 176b parameter language model. *arXiv preprint arXiv:2211.02001*, 2022.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015. doi: 10.18653/v1/D15-1166.

Navonil Majumder, Pengfei Hong, Shanshan Peng, Jiankun Lu, Deepanway Ghosal, Alexander Gelbukh, Rada Mihalcea, and Soujanya Poria. Mime: Mimicking emotions for empathetic response generation. *arXiv preprint arXiv:2010.01454*, 2020.

Navonil Majumder, Deepanway Ghosal, Devamanyu Hazarika, Alexander Gelbukh, Rada Mihalcea, and Soujanya Poria. Exemplars-guided empathetic response generation controlled by the elements of human communication. *IEEE Access*, 10:77176–77190, 2022.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

Mitch Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: Annotating predicate argument structure. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

Petr Marek and Štěpán Müller. SumeCzech-NER, 2021. URL `http://hdl.handle.net/11234/1-3505`. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Petr Marek, Štěpán Müller, Jakub Konrád, Petr Lorenc, Jan Pichl, and Jan Šedivý. Text summarization of czech news articles using named entities. *arXiv preprint arXiv:2104.10454*, 2021a.

Petr Marek, Vishal Ishwar Naik, Vincent Auvray, and Anuj Goyal. Oodgan: Generative adversarial network for out-of-domain data generation. *arXiv preprint arXiv:2104.02484*, 2021b.

Andrey Andreyevich Markov. An example of statistical investigation in the text of 'eugene onyegin'illustrating coupling of 'tests' in chains. *Proceedings of the Academy of Sciences of St. Petersburg*, 7:153–162, 1913.

Michael L Mauldin. Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition. In *AAAI*, volume 94, pages 16–21, 1994.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. *arXiv preprint arXiv:2005.00661*, 2020.

Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411, 2004.

Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 234–239. IEEE, 2012.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010.

Fabio Motoki, Valdemar Pinho Neto, and Victor Rodrigues. More human than human: Measuring chatgpt political bias. *Public Choice*, pages 1–21, 2023.

Niels Mündler, Jingxuan He, Slobodan Jenko, and Martin Vechev. Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. *arXiv preprint arXiv:2305.15852*, 2023.

Štěpán Müller. Named entity recognition. Research project, CTU in Prague, 2020a.

Štěpán Müller. Text summarization using named entity recognition. B.S. thesis, CTU in Prague, 2020b.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016. doi: 10.18653/v1/K16-1028.

Ani Nenkova. Automatic text summarization of newswire: Lessons learned from the document understanding conference. 2005.

Peiqing Niu, Zhongfu Chen, Meina Song, et al. A novel bi-directional interrelated model for joint intent detection and slot filling. *arXiv preprint arXiv:1907.00390*, 2019.

Chikashi Nobata, Satoshi Sekine, Hitoshi Isahara, and Ralph Grishman. Summarization system integrated with named entity tagging and ie pattern discovery. In *LREC*, 2002.

Aleksander Obuchowski and Michal Lew. Transformer-capsule model for intent detection (student abstract). In *AAAI*, pages 13885–13886, 2020.

OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023a.

OpenAI. Introducing chatgpt, 2023b. URL `https://openai.com/blog/chatgpt`.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Alok Ranjan Pal and Diganta Saha. An approach to automatic text summarization using wordnet. In *2014 IEEE International Advance Computing Conference (IACC)*, pages 1169–1173. IEEE, 2014. doi: 10.1109/IAdCC.2014.6779492.

Yikang Pan, Liangming Pan, Wenhu Chen, Preslav Nakov, Min-Yen Kan, and William Yang Wang. On the risk of misinformation pollution with large language models. *arXiv preprint arXiv:2305.13661*, 2023.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

Ashwin Paranjape, Abigail See, Kathleen Kenealy, Haojun Li, Amelia Hardy, Peng Qi, Kaushik Ram Sadagopan, Nguyet Minh Phu, Dilara Soylu, and Christopher D Manning. Neural generation meets real people: Towards emotionally engaging mixed-initiative conversations. *arXiv preprint arXiv:2008.12348*, 2020.

Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web: methods and strategies of web personalization*, pages 325–341. Springer, 2007.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023. URL `https://arxiv.org/abs/2306.01116`.

Jan Pichl, Petr Marek, Jakub Konrád, Martin Matulík, Hoang Long Nguyen, and Jan Šedivý. Alquist: The alexa prize socialbot. *arXiv preprint arXiv:1804.06705*, 2018.

Jan Pichl, Petr Marek, Jakub Konrád, Petr Lorenc, Van Duy Ta, and Jan Šedivý. Alquist 3.0: Alexa prize bot using conversational knowledge graph. *arXiv preprint arXiv:2011.03261*, 2020a.

Jan Pichl, Petr Marek, Jakub Konrád, Martin Matulík, and Jan Šedivý. Alquist 2.0: Alexa prize socialbot based on sub-dialogue models. *arXiv preprint arXiv:2011.03259*, 2020b.

Jan Pichl, Petr Marek, Jakub Konrád, Petr Lorenc, Ondřej Kobza, Tomáš Zajíček, and Jan Šedivý. Flowstorm: Open-source platform with hybrid dialogue architecture. *arXiv preprint arXiv:2212.09377*, 2022.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Filip Radlinski, Krisztian Balog, Bill Byrne, and Karthik Krishnamoorthi. Coached conversational preference elicitation: A case study in understanding movie preferences. In *Proceedings of the Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21 (1):5485–5551, 2020.

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, et al. Conversational ai: The science behind the alexa prize. *arXiv preprint arXiv:1801.03604*, 2018.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. Towards empathetic open-domain conversation models: A new benchmark and dataset. *arXiv preprint arXiv:1811.00207*, 2018.

J. Ren, Peter J. Liu, E. Fertig, Jasper Snoek, Ryan Poplin, Mark A. DePristo, Joshua V. Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *NeurIPS*, 2019.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M Smith, et al. Recipes for building an open-domain chatbot. *arXiv preprint arXiv:2004.13637*, 2020.

Alexander Rudnicky and Wei Xu. An agenda-based dialog management architecture for spoken language systems. In *IEEE Automatic Speech Recognition and Understanding Workshop*, volume 13, 1999.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Seonghan Ryu, Sangjun Koo, Hwanjo Yu, and Gary Geunbae Lee. Out-of-domain detection based on generative adversarial network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 714–718, 2018.

Harvey Sacks. Lectures on conversation: Volume i. *Malden, Massachusetts: Blackwell*, 1992.

Harvey Sacks, Emanuel A Schegloff, and Gail Jefferson. A simplest systematics for the organization of turn taking for conversation. In *Studies in the organization of conversational interaction*, pages 7–55. Elsevier, 1978.

Erik F Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*, 2003.

Emanuel A Schegloff and Harvey Sacks. Opening up closings. 1973.

Frederik Schulze and Mariana Neves. Entity-supported summarization of biomedical abstracts. In *Proceedings of the Fifth Workshop on Building and Evaluating Resources for Biomedical Text Mining (BioTxtM2016)*, pages 40–49, 2016.

John R Searle. Minds, brains, and programs. *Behavioral and brain sciences*, 3(3):417–424, 1980.

Magda Ševčíková, Zdeněk Žabokrtský, Jana Straková, and Milan Straka. Czech named entity corpus 2.0, 2014. URL `http://hdl.handle.net/11858/00-097C-0000-0023-1B22-8`. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Omar Shaikh, Hongxin Zhang, William Held, Michael Bernstein, and Diyi Yang. On second thought, let's not think step by step! bias and toxicity in zero-shot reasoning. *arXiv preprint arXiv:2212.08061*, 2022.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Kurt Shuster, Jing Xu, Mojtaba Komeili, Da Ju, Eric Michael Smith, Stephen Roller, Megan Ung, Moya Chen, Kushal Arora, Joshua Lane, et al. Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage. *arXiv preprint arXiv:2208.03188*, 2022.

Jack Sidnell. *Conversation Analysis: An Introduction.* Wiley-Blackwell, Oxford, 2010. URL `https://www.wiley.com/en-us/Conversation+Analysis:+An+Introduction-p-9781405159005`.

Shengli Song, Haitao Huang, and Tongxiao Ruan. Abstractive text summarization using lstm-cnn based deep learning. *Multimedia Tools and Applications*, 78(1):857–875, 2019. doi: 10.1007/s11042-018-5749-3.

Milan Straka, Nikita Mediankin, Tom Kocmi, Zdeněk Žabokrtský, Vojtěch Hudeček, and Jan Hajič. Sumeczech: Large czech news-based summarization dataset. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

Jana Straková, Milan Straka, and Jan Hajič. Neural Architectures for Nested NER through Linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1527. URL `https://www.aclweb.org/anthology/P19-1527`.

Hsuan Su, Shachi H Kumar, Sahisnu Mazumder, Wenda Chen, Ramesh Manuvinakurike, Eda Okur, Saurav Sahay, Lama Nachman, Shang-Tse Chen, and Hung-yi Lee. Position matters! empirical study of order effect in knowledge-grounded dialogue. *arXiv preprint arXiv:2302.05888*, 2023.

Shuntaro Takahashi and Kumiko Tanaka-Ishii. Evaluating computational language models with scaling properties of natural language. *Computational Linguistics*, 45(3):481–513, 2019.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

Maxim Tkachenko, Mikhail Malyuk, Andrey Holmanyuk, and Nikolai Liubimov. Label Studio: Data labeling software, 2020-2022. URL `https://github.com/heartexlabs/label-studio`. Open source software available from https://github.com/heartexlabs/label-studio.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Alan Turing. Computing machinery and intelligence. *Mind*, 59(236):433, 1950.

Deeksha Varshney, Asif Ekbal, Mrigank Tiwari, and Ganesh Prasad Nagaraja. Emokbgan: Emotion controlled response generation using generative adversarial network for knowledge grounded conversation. *PloS one*, 18(2):e0280458, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Pranav Narayanan Venkit, Sanjana Gautam, Ruchi Panchanadikar, Shomir Wilson, et al. Nationality bias in text generation. *arXiv preprint arXiv:2302.02463*, 2023.

Vladimir Vlasov, Johannes EM Mosig, and Alan Nichol. Dialogue transformers. *arXiv preprint arXiv:1910.00486*, 2019.

Richard Wallace. The elements of aiml style. *Alice AI Foundation*, 139, 2003.

Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

Johannes Welbl, Amelia Glaese, Jonathan Uesato, Sumanth Dathathri, John Mellor, Lisa Anne Hendricks, Kirsty Anderson, Pushmeet Kohli, Ben Coppin, and Po-Sen Huang. Challenges in detoxifying language models. *arXiv preprint arXiv:2109.07445*, 2021.

Jason D Williams, Matthew Henderson, Antoine Raux, Blaise Thomson, Alan Black, and Deepak Ramachandran. The dialog state tracking challenge series. *AI Magazine*, 35(4): 121–124, 2014.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274*, 2017.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Steve Worswick. Mitsuku. *Geraadpleegd van https://www. pandorabots. com/mitsuku*, 2017.

Can Xu, Wei Wu, and Yu Wu. Towards explainable and controllable open domain dialogue generation with dialogue acts. *arXiv preprint arXiv:1807.07255*, 2018.

Jing Xu, Arthur Szlam, and Jason Weston. Beyond goldfish memory: Long-term open-domain conversation. *arXiv preprint arXiv:2107.07567*, 2021a.

Yi Xu, Hai Zhao, and Zhuosheng Zhang. Topic-aware multi-turn dialogue modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14176–14184, 2021b.

Kaichun Yao, Libo Zhang, Dawei Du, Tiejian Luo, Lili Tao, and Yanjun Wu. Dual encoding for abstractive text summarization. *IEEE transactions on cybernetics*, 2018. doi: 10.1109/TCYB.2018.2876317.

Dian Yu and Zhou Yu. Midas: A dialog act annotation scheme for open domain human machine spoken conversations. *arXiv preprint arXiv:1908.10023*, 2019.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip S Yu. Joint slot filling and intent detection via capsule neural networks. *arXiv preprint arXiv:1812.09471*, 2018a.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A Smith. How language model hallucinations can snowball. *arXiv preprint arXiv:2305.13534*, 2023.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*, 2018b.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. Dialogpt: Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536*, 2019.

Yong Zhang, Joo Er Meng, and Mahardhika Pratama. Extractive document summarization based on convolutional neural networks. In *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 918–922. IEEE, 2016. doi: 10.1109/IECON.2016.7793761.

Xueliang Zhao, Wei Wu, Chongyang Tao, Can Xu, Dongyan Zhao, and Rui Yan. Low-resource knowledge-grounded dialogue generation. *arXiv preprint arXiv:2002.10348*, 2020.

Zhiwei Zhao and Youzheng Wu. Attention-based convolutional neural networks for sentence classification. In *Proc. Interspeech 2016*, pages 705–709, 2016. doi: 10.21437/Interspeech.2016-354.

Yinhe Zheng, Guanyi Chen, and Minlie Huang. Out-of-domain detection for natural language understanding in dialog systems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1198–1209, 2020.

Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. E2s2: Encoding-enhanced sequence-to-sequence pretraining for language understanding and generation. *arXiv preprint arXiv:2205.14912*, 2022.

Jiawei Zhou, Yixuan Zhang, Qianni Luo, Andrea G Parker, and Munmun De Choudhury. Synthetic lies: Understanding ai-generated misinformation and evaluating algorithmic and human solutions. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–20, 2023.

Appendix A

Publications of the Author

Publications Related to the Thesis

1. Jan Pichl, **Petr Marek**, Jakub Konrád, Long Hoang Nguyen, Martin Matulík, Jan Šedivý. Alquist: The Alexa Prize Socialbot. *1st Alexa Prize Proceedings*, Las Vegas, NV, USA, 2017.

   Jan Pichl (55%): Conceptualization, System architecture, NLU, Implementation, Experiments, Writing, Project administration.

   Petr Marek (10%): Conceptualization, Dialogue management, Implementation, Experiments, Writing.

   Jakub Konrád (20%): Conceptualization, Knowledge management, Implementation, Experiments, Writing.

   Long Hoang Nguyen (5%): Conceptualization, Implementation, Writing.

   Martin Matulík (5%): Conceptualization, Implementation, Writing.

   Jan Šedivý (5%): Supervision, Conceptualization, Project administration, Review of writing.

   Cited by:

   1. Ram Ashwin, Rohit Prasad, Chandra Khatri, Anushree Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, others. Conversational AI: The science behind the Alexa Prize. 2017.

      Indexed in: Google Scholar.

   2. Khatri Chandra, Anu Venkatesh, Behnam Hedayatnia, Raefer Gabriel, Ashwin Ram, Rohit Prasad. Alexa prize—state of the art in conversational AI. *AI Magazine 39*. 3: 40–55, 2018.

      Indexed in: Scopus, Web of Science, Google Scholar.

   3. Chen Chun-Yen, Dian Yu, Weiming Wen, Yi Mang Yang, Jiaping Zhang, Mingyang Zhou, Kevin Jesse, Austin Chau, Antara Bhowmick, Shreenath Iyer,

others. Gunrock: Building a human-like social bot by leveraging large scale real user data. 2018.

Indexed in: Google Scholar.

4. Larionov George, Zachary Kaden, Hima Varsha Dureddy, Gabriel Bayomi T Kalejaiye, Mihir Kale, Srividya Pranavi Potharaju, Ankit Parag,Shah, Alexander I Rudnicky. Tartan: A retrieval-based socialbot powered by a dynamic finite-state machine architecture. 2018.

   Indexed in: Google Scholar.

5. Ostendorf Mari. Several Steps Forward in Conversational AI. 2018.

   Indexed in: Google Scholar.

6. Malik Szymon, Adrian Lancucki, Jan Chorowski. Efficient Purely Convolutional Text Encoding. 2018.

   Indexed in: Scopus, Google Scholar.

7. Jonell Patrik, Per Fallgren, Fethiye Irmak Doğan, Jose Lopes, Ulme Wennberg, Gabriel Skantze. Crowdsourcing a self-evolving dialog graph. *Proceedings of the 1st International Conference on Conversational User Interfaces*, 2019.

   Indexed in: Scopus, Web of Science, Google Scholar.

8. Li Aaron W, Veronica Jiang, Steven Y Feng, Julia Sprague, Wei Zhou, Jesse Hoey. Follow alice into the rabbit hole: Giving dialogue agents understanding of human level attributes. 2019.

   Indexed in: Google Scholar.

9. Choi Jason Ingyu, Ali Ahmadvand, Eugene Agichtein. Offline and online satisfaction prediction in open-domain conversational systems. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019.

   Indexed in: Scopus, Web of Science, Google Scholar.

10. Huang Minlie, Xiaoyan Zhu, and Jianfeng Gao. Challenges in building intelligent open-domain dialog systems. *ACM Transactions on Information Systems (TOIS)*, 38.3 1-32, 2020.

    Indexed in: Scopus, Web of Science, Google Scholar.

11. Liang Kaihui, Austin Chau, Yu Li, Xueyuan Lu, Dian Yu, Mingyang Zhou, Ishan Jain, Sam Davidson, Josh Arnold, Minh Nguyen, others. Gunrock 2.0: A user adaptive social conversational system. 2020.

    Indexed in: Google Scholar.

12. Zolitschka Jan Felix. A novel multi-agent-based chatbot approach to orchestrate conversational assistants. *Business Information Systems: 23rd International Conference, BIS 2020,* Colorado Springs, CO, USA, June 8–10, 2020, Proceedings 23, 2020.

    Indexed in: Scopus, Google Scholar.

13. Fang Hao. Building A User-Centric and Content-Driven Socialbot. 2020.

    Indexed in: Google Scholar.

14. Song Haoyu, Yan Wang, Weinan Zhang, Zhengyu Zhao, Ting Liu, Xiaojiang Liu. Profile Consistency Identification for Open-domain Dialogue Agents. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

    Indexed in: Scopus, Web of Science, Google Scholar.

15. Li Aaron W, Veronica Jiang, Steven Y Feng, Julia Sprague, Wei Zhou, Jesse Hoey. Aloha: Artificial learning of human attributes for dialogue agents. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

    Indexed in: Scopus, Web of Science, Google Scholar.

16. Nehring Jan, Nils Feldhus, Harleen Kaur, Akhyar Ahmed. Combining open domain question answering with a task-oriented dialog system. *Proceedings of the 1st Workshop on Document-grounded Dialogue and Conversational Question Answering (DialDoc 2021)*, 2021.

    Indexed in: Scopus, Web of Science, Google Scholar.

17. Walker Nicholas Thomas, Stefan Ultes, Pierre Lison. GraphWOZ: Dialogue Management with Conversational Knowledge Graphs. 2022.

    Indexed in: Google Scholar.

2. Jan Pichl, **Petr Marek**, Jakub Konrád, Martin Matulík, Jan Šedivý. Alquist 2.0: Alexa Prize Socialbot Based on Sub-Dialogue Models. *2nd Proceedings of Alexa Prize*, Las Vegas, NV, USA, 2018.

   Jan Pichl (30%): Conceptualization, System architecture, NLU, Implementation, Experiments, Writing, Project administration.

   Petr Marek (30%): Conceptualization, Dialogue management, Implementation, Experiments, Writing.

   Jakub Konrád (30%): Conceptualization, Knowledge management, Implementation, Experiments, Writing.

   Martin Matulík (5%): Conceptualization, Implementation, Writing.

   Jan Šedivý (5%): Supervision, Conceptualization, Project administration, Review of writing.

   Cited by:

   1. Bowden Kevin K, Jiaqi Wu, Wen Cui, Juraj Juraska, Vrindavan Harrison, Brian Schwarzmann, Nicholas Santer, Steve Whittaker, Marilyn Walker. Entertaining and opinionated but too controlling: a large-scale user study of an open

domain Alexa prize system. *Proceedings of the 1st International Conference on Conversational User Interfaces,* 2019.

Indexed in: Scopus, Web of Science, Google Scholar.

2. Gabriel Raefer, Yang Liu, Anna Gottardi, Mihail Eric, Anju Khatri, Anjali Chadha, Qinlang Chen, Behnam Hedayatnia, Pankaj Rajan, Ali Binici, others. Further advances in open domain dialog systems in the third alexa prize socialbot grand challenge. 2019.

   Indexed in: Google Scholar.

3. Szpektor Idan, Deborah Cohen, Gal Elidan, Michael Fink, Avinatan Hassidim, Orgad Keller, Sayali Kulkarni, Eran Ofek, Sagie Pudinsky, Asaf Revach, others. Dynamic composition for conversational domain exploration. *Proceedings of The Web Conference 2020,* 2020.

   Indexed in: Scopus, Web of Science, Google Scholar.

4. Paranjape Ashwin, Abigail See, Kathleen Kenealy, Haojun Li, Amelia Hardy, Peng Qi, Kaushik Ram Sadagopan, Nguyet Minh Phu, Dilara Soylu, Christopher D Manning. Neural generation meets real people: Towards emotionally engaging mixed-initiative conversations. 2020.

   Indexed in: Google Scholar.

5. Fang Hao. Building A User-Centric and Content-Driven Socialbot. 2020.

   Indexed in: Google Scholar.

6. Kuratov Yuri, Idris Yusupov, Dilyara Baymurzina, Denis Kuznetsov, Daniil Cherniavskii, Alexander Dmitrievskiy, Elena Ermakova, Fedor Ignatov, Dmitry Karpov, Daniel Kornev, others. Dream technical report for the alexa prize 2019. 2020.

   Indexed in: Google Scholar.

7. Juraska Juraj, Kevin Bowden, Lena Reed, Vrindavan Harrison, Wen Cui, Omkar Patil, Rishi Rajasekaran, Angela Ramirez, Cecilia Li, Eduardo Zamora, others. Athena 2.0: Contextualized Dialogue Management for an Alexa Prize SocialBot. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2021.

   Indexed in: Scopus, Web of Science, Google Scholar.

8. Vasquez-Correa Juan Camilo, Juan Carlos Guerrero-Sierra, Jose Luis Pemberty-Tamayo, Juan Esteban Jaramillo, Andres Felipe Tejada-Castro. One system to rule them all: A universal intent recognition system for customer service chatbots. 2021.

   Indexed in: Google Scholar.

9. Walker Nicholas Thomas, Torbjørn Dahl, Pierre Lison. Dialogue management as graph transformations. *Conversational AI for Natural Human-Centric Interaction: 12th International Workshop on Spoken Dialogue System Technology, IWSDS 2021,* Singapore, 2022.

Indexed in: Scopus, Google Scholar.

10. Vavrecka Michal, Gabriela Sejnova, Petr Schimperk. Personified Robotic Chatbot Based On Compositional Dialogues. *2022 International Conference on Interactive Media, Smart Systems and Emerging Technologies (IMET),* 2022.

    Indexed in: Scopus, Google Scholar.

3. Jan Pichl, **Petr Marek**, Jakub Konrád, Petr Lorenc, Van Duy Ta, Jan Šedivý. Alquist 3.0: Alexa Prize Bot Using Conversational Knowledge Graph. *Alexa Prize Socialbot Grand Challenge 3 Proceedings*, 2020.

   Jan Pichl (19%): Conceptualization, System architecture, NLU, Implementation, Experiments, Writing, Project administration.

   Petr Marek (20%): Conceptualization, Dialogue management, Implementation, Experiments, Writing.

   Jakub Konrád (20%): Conceptualization, Knowledge management, Implementation, Experiments, Writing.

   Petr Lorenc (20%): Conceptualization, NLP, Implementation, Experiments, Writing.

   Van Duy Ta (20%): Conceptualization, Implementation, System architecture, Writing.

   Jan Šedivý (1%): Supervision, Project administration, Review of writing.

   Cited by:

   1. Gabriel Raefer, Yang Liu, Anna Gottardi, Mihail Eric, Anju Khatri, Anjali Chadha, Qinlang Chen, Behnam Hedayatnia, Pankaj Rajan, Ali Binici, others. Further advances in open domain dialog systems in the third alexa prize socialbot grand challenge. 2019.

      Indexed in: Google Scholar.

   2. Axelsson Agnes, Hendrik Buschmeier, Gabriel Skantze. Modeling feedback in interaction with conversational agents—a review. *Frontiers in Computer Science 4,* 2022.

      Indexed in: Scopus, Web of Science, Google Scholar.

   3. Ahlberg Sofie, Agnes Axelsson, Pian Yu, Wenceslao Shaw Cortez, Yuan Gao, Ali Ghadirzadeh, Ginevra Castellano, Danica Kragic, Gabriel Skantze, Dimos V Dimarogonas. Co-adaptive Human–Robot Cooperation: Summary and Challenges. *Unmanned Systems* 10. 02(2022): 187–203, 2022.

      Indexed in: Scopus, Web of Science, Google Scholar.

4. Kann Katharina, Abteen Ebrahimi, Joewie Koh, Shiran Dudy, Alessandro Roncone. Open-domain dialogue generation: What we can do, cannot do, and should do next. *Proceedings of the 4th Workshop on NLP for Conversational AI,* 2022.

   Indexed in: Scopus, Web of Science, Google Scholar.

5. Yamashita Sanae, Ryuichiro Higashinaka. Data Collection for Empirically Determining the Necessary Information for Smooth Handover in Dialogue. *Proceedings of the Thirteenth Language Resources and Evaluation Conference,* 2022.

   Indexed in: Scopus, Web of Science, Google Scholar.

6. Axelsson Agnes, Gabriel Skantze. Do you follow? a fully automated system for adaptive robot presenters. *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction,* 2023.

   Indexed in: Google Scholar.

7. Harrison Vrindavan, Rishi Rajasekaran, Marilyn Walker. A Transformer-based Response Evaluator for Open-Domain Spoken Conversation. 2023.

   Indexed in: Google Scholar.

8. Reddy Revanth Gangi, Sharath Chandra, Mankeerat Singh Sidhu, H Bai, Wenta, Yao, Pranav Pillai, Karan Aggarwal, Liliang Ren, Prathamesh Sonawane, Keyu Han, others. CharmBana: Progressive Responses with Real-Time Internet Search for Knowledge-Powered Conversations. 2023.

   Indexed in: Google Scholar.

4. **Petr Marek**, Vishal Ishwar Naik, Vincent Auvray, Anuj Goyal. OodGAN: Generative Adversarial Network for Out-of-Domain Data Generation. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*: Human Language Technologies: Industry Papers (pp. 238-245), 2021.

   Petr Marek (85%): Conceptualization, Implementation, Experiments, Writing.

   Vishal Ishwar Naik (5%): Supervision, Project administration, Review of writing.

   Vincent Auvray (5%): Supervision, Review of writing.

   Anuj Goyal (5%): Supervision, Review of writing.

   Cited by:

1. Lee Seul, Dong Bok Lee, Sung Ju Hwang. MOG: Molecular Out-of-distribution Generation with Energy-based Models. 2021.

   Indexed in: Google Scholar.

2. Xia Xuan, Xizhou Pan, Nan Li, Xing He, Lin Ma, Xiaoguang Zhang, Ning Ding. GAN-based anomaly detection: A review. *Neurocomputing 493.* (2022): 497–535, 2022

   Indexed in: Scopus, Web of Science, Google Scholar.

3. Lee Gyubok, Hyeonji Hwang, Seongsu Bae, Yeonsu Kwon, Woncheol Shin, Seongjun Yang, Minjoon Seo, Jong-Yeup Kim, Edward Choi. EHRSQL: A Practical Text-to-SQL Benchmark for Electronic Health Records. *Advances in Neural Information Processing Systems 35.* (2022): 15589–15601, 2022.

   Indexed in: Scopus, Google Scholar.

4. Song Yu, Donglin Wang. Learning on graphs with out-of-distribution nodes. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining,* 2022.

   Indexed in: Scopus, Google Scholar.

5. Lang Hao, Yinhe Zheng, Jian Sun, Fei Huang, Luo Si, Yongbin Li. Estimating Soft Labels for Out-of-Domain Intent Detection. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing,* 2022.

   Indexed in: Scopus, Google Scholar.

6. Liu Luping, Yi Ren, Xize Cheng, Zhou Zhao. Diffusion denoising process for perceptron bias in out-of-distribution detection. 2022.

   Indexed in: Google Scholar.

7. Yilmaz Eyup, Cagri Toraman. D2u: Distance-to-uniform learning for out-of-scope detection. *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 2022.

   Indexed in: Scopus, Web of Science, Google Scholar.

8. Lee Seungyeon, Minho Lee. Type-dependent Prompt CycleQAG: Cycle Consistency for Multi-hop Question Generation. *Proceedings of the 29th International Conference on Computational Linguistics*, 2022.

   Indexed in: Scopus, Google Scholar.

9. Zheng Yinhe, Guanyi Chen. Accuracy on In-Domain Samples Matters When Building Out-of-Domain detectors: A Reply to Marek et al.(2021). 2022.

   Indexed in: Google Scholar.

10. Liu Luping, Yi Ren, Xize Cheng, Zhou Zhao. Out-of-distribution Detection with Diffusion-based Neighborhood. 2022.

    Indexed in: Google Scholar.

11. Lang Hao, Yinhe Zheng, Yixuan Li, Jian Sun, Fei Huang, Yongbin Li. A Survey on Out-of-Distribution Detection in NLP. 2023.

    Indexed in: Google Scholar.

12. Lang Hao, Yinhe Zheng, Binyuan Hui, Fei Huang, Yongbin Li. Out-of-Domain Intent Detection Considering Multi-turn Dialogue Contexts. 2023.

    Indexed in: Google Scholar.

13. Akbari Masoud, Ali Mohades, M Hassan Shirali-Shahreza. A Hybrid Architecture for Out of Domain Intent Detection and Intent Discovery. 2023.

    Indexed in: Google Scholar.

14. Hwang Dae Yon, Yaroslav Nechaev, Cyprien Lichy, Renxian Zhang. GAN-LM: Generative Adversarial Network using Language Models for Downstream Applications. *Proceedings of the 16th International Natural Language Generation Conference,* 2023.

    Indexed in: Google Scholar.

15. Koda Satoru, Ikuya Morikawa. OOD-Robust Boosting Tree for Intrusion Detection Systems. *2023 International Joint Conference on Neural Networks (IJCNN),* 2023.

    Indexed in: Scopus, Google Scholar.

5. **Petr Marek**, Štěpán Müller, Jakub Konrád, Petr Lorenc, Jan Pichl, Jan Šedivý. Text Summarization of Czech News Articles Using Named Entities. *The Prague Bulletin of Mathematical Linguistics*, (116), 5-25, 2021.

   Petr Marek (55%): Conceptualization, Implementation, Experiments, Writing.

   Štěpán Müller (25%): Conceptualization, Implementation, Experiments.

   Jakub Konrád (5%): Review of writing.

   Petr Lorenc (5%): Review of writing.

   Jan Pichl (5%): Review of writing.

   Jan Šedivý (5%): Supervision, Review of writing.

   Cited by:

   1. Matsushita Kyoumoto, Takuya Makino, Tomoya Iwakura. Improving Neural Language Processing with Named Entities. *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021),* 2021.

      Indexed in: Scopus, Google Scholar.

   2. Rathi Kartik, Saumy Raj, Sudhir Mohan, Yash Vardan Singh. A Review of state-of-the-art Automatic Text Summarisation. *International Journal of Creative Research Thoughts (2022),* 2022.

      Indexed in: Google Scholar.

3. Rehman Tohida, Debarshi Kumar Sanyal, Prasenjit Majumder, Samiran Chattopadhyay. Named Entity Recognition Based Automatic Generation of Research Highlights. *Third Workshop on Scholarly Document Processing,* 2022.

   Indexed in: Google Scholar.

4. Lima Khadija Akter, Khan Md Hasib, Sami Azam, Asif Karim, Sidratul Montaha, Sheak Rashed Haider Noori, Mirjam Jonkman. A novel Data and Model Centric artificial intelligence based approach in developing high-performance Named Entity Recognition for Bengali Language. *Plos one 18.* 9(2023): e0287818, 2023.

   Indexed in: Google Scholar.

5. Choudhary Ambrish, Mamatha Alugubelly, Rupal Bhargava. A Comparative Study on Transformer-based News Summarization. *2023 15th International Conference on Developments in eSystems Engineering (DeSE),* 2023.

   Indexed in: Scopus, Google Scholar.

6. Vajdecka Peter, Vojtech Svatek, Martin Vita. A Novel Approach to Abstractive Summarization Based on LOF, Sentence-BERT and T5–with Fact Checking Use Case. 2023.

   Indexed in: Google Scholar.

6. Jakub Konrád, Jan Pichl, **Petr Marek**, Petr Lorenc, Van Duy Ta, Ondřej Kobza, Lenka Hýlová, Jan Šedivý. Alquist 4.0: Towards Social Intelligence Using Generative Models and Dialogue Personalization. *Alexa Prize Socialbot Grand Challenge 4 Proceedings*, 2021.

   Jakub Konrád (16%): Conceptualization, Knowledge management, Implementation, Experiments, Writing, Project administration.

   Jan Pichl (16%): Conceptualization, System architecture, NLU, Implementation, Experiments, Writing.

   Petr Marek (16%): Conceptualization, Dialogue management, Neural Response Generator, Implementation, Experiments, Writing.

   Petr Lorenc (16%): Conceptualization, NLP, Experiments, Writing.

   Van Duy Ta (16%): Implementation, Writing.

   Ondřej Kobza (16%): Implementation, Writing.

   Lenka Hýlová (2%): Neural Response Generator, Implementation, Experiments.

   Jan Šedivý (2%): Supervision, Project administration, Review of writing.

Cited by:

1. Kuznetsov Denis, Dmitry Evseev, Lidia Ostyakova, Oleg Serikov, Daniel Kornev, Mikhail Burtsev. Discourse-Driven Integrated Dialogue Development Environment for Open-Domain Dialogue Systems. *Proceedings of the 2nd Workshop on Computational Approaches to Discourse,* 2021.

   Indexed in: Scopus, Google Scholar.

2. Jin Di, Sijia Liu, Yang Liu, Dilek Hakkani-Tur. Improving Bot Response Contradiction Detection via Utterance Rewriting. *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue,* 2022.

   Indexed in: Google Scholar.

3. Hedayatnia Behnam, Di Jin, Yang Liu, Dilek Hakkani-Tur. A Systematic Evaluation of Response Selection for Open Domain Dialogue. *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue,* 2022.

   Indexed in: Google Scholar.

4. Bowden Kevin K, Marilyn Walker. Let's Get Personal: Personal Questions Improve SocialBot Performance in the Alexa Prize. 2023.

   Indexed in: Google Scholar.

5. Vicente, Frederico, Rafael, Ferreira, David, Semedo, Joao, Magalhaes. The Wizard of Curiosities: Enriching Dialogues with Fun Facts. *Proceedings of the 24th Meeting of the Special Interest Group on Discourse and Dialogue,* 2023.

   Indexed in: Google Scholar.

6. Mai Long, Julie Carson-Berndsen. I already said that! Degenerating redundant questions in open-domain dialogue systems. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics* (Volume 4: Student Research Workshop), 2023.

   Indexed in: Google Scholar.

7. Ondřej Kobza, Jan Čuhel, Tommaso Gargiani, David Herel, **Petr Marek**, Jan Šedivý. Alquist 5.0: Dialogue Trees Meet Generative Models. A Novel Approach for Enhancing SocialBot Conversations. *Alexa Prize Socialbot Grand Challenge 5 Proceedings*, 2023.

   Ondřej Kobza (19%): Conceptualization, System architecture, Neural response generator, Implementation, Experiments, Writing, Project administration.

   Jan Čuhel (19%): Conceptualization, System Architecture, Neural Response Generator, Visuals, Implementation, Experiments, Writing.

   Tommaso Gargiani (19%): Conceptualization, Knowledge Management, Implementation, Experiments, Writing.

David Herel (19%): Conceptualization, Neural Response Generator, Implementation, Experiments, Writing.

Petr Marek (19%): Conceptualization, Hybrid dialogue management, Implementation, Experiments, Writing.

Jan Šedivý (5%): Supervision, Project administration, Review of writing.

Other Publications

1. **Petr Marek**. Hybrid Code Networks using a convolutional neural network as an input layer achieves higher turn accuracy. *Proceedings of the International Student Scientific Conference Poster* – 23/2019, Prague, Czech Republic, 2019.

   Petr Marek (100%): Conceptualization, Implementation, Experiments, Writing.

2. Petr Lorenc, Tommaso Gargiani, Jan Pichl, Jakub Korád, **Petr Marek**, Ondřej Kobza, Jan Šedivý. Metric Learning and Adaptive Boundary for Out-of-Domain Detection. *Natural Language Processing and Information Systems: 27th International Conference on Applications of Natural Language to Information Systems*, NLDB 2022, Valencia, Spain, June 15–17, 2022.

   Petr Lorenc (30%): Conceptualization, Implementation, Experiments, Writing.

   Tommaso Gargiani (30%): Conceptualization, Implementation, Experiments, Writing.

   Jan Pichl (10%): Review of writing.

   Jakub Korád (10%): Review of writing.

   Petr Marek (10%): Review of writing.

   Ondřej Kobza (5%): Review of writing.

   Jan Šedivý (5%): Supervision, Review of writing.

3. Petr Lorenc, **Petr Marek**, Jan Pichl, Jakub Korád, Jan Šedivý. Benchmark of Public Intent Recognition Services. *Language Resources and Evaluation* volume 56, 1023–1041, 2022.

   Petr Lorenc (80%): Conceptualization, Implementation, Experiments, Writing.

   Petr Marek (5%): Review of writing.

   Jan Pichl (5%): Review of writing.

   Jakub Korád (5%): Review of writing.

   Jan Šedivý (5%): Supervision, Review of writing.

Cited by:

1. Vasquez-Correa Juan Camilo, Juan Carlos Guerrero-Sierra, Jose Luis Pemberty-Tamayo, Juan Esteban Jaramillo, Andres Felipe Tejada-Castro. One system to rule them all: A universal intent recognition system for customer service chatbots. 2021.

   Indexed in: Google Scholar.

4. Jan Pichl, **Petr Marek**, Jakub Korád, Petr Lorenc, Ondřej Kobza, Tomáš Zajíček, Jan Šedivý. Flowstorm: Open-Source Platform with Hybrid Dialogue Architecture. *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics*: Human Language Technologies: System Demonstrations, 39-45, 2022.

   Jan Pichl (55%): Conceptualization, Implementation, Experiments, Writing.

   Petr Marek (10%): Conceptualization, Implementation, Review of writing.

   Jakub Korád (10%): Conceptualization, Implementation, Review of writing.

   Petr Lorenc (10%): Conceptualization, Implementation, Review of writing.

   Ondřej Kobza (5%): Review of writing.

   Tomáš Zajíček (5%): Conceptualization, Supervision, Implementation.

   Jan Šedivý (5%): Supervision, Review of writing.

   Cited by:

   1. Pande Charuta, Andreas Martin, Christoph Pimmer. Towards hybrid dialog management strategies for a health coach chatbot. *Proceedings of the AAAI 2023 Spring Symposium on Challenges Requiring the Combination of Machine Learning and Knowledge Engineering (AAAI-MAKE 2023).* Vol. 3433, 2023.

      Indexed in: Scopus, Google Scholar.

Appendix B

Lectures, Presentations and Public Appearances

B.1 Academic

1. Long Hoang Nguyen, Petr Marek, Jan Pichl, Jakub Konrád, Martin Matulík. Alquist Bot Development - Our path to Amazon's Alexa Prize Finals. Machine Learning Meetup in Prague, Prague Czech Republic 2017. https://www.youtube.com/watch?v=bLB3Bsj-WRQ&ab_channel=MachineLearningMeetups

2. Jan Pichl, Petr Marek. Alquist: How to develop an engaging conversation. Institute of Formal and Applied Linguistics UFAL MFF CUNI, Prague Czech Republic 2019. https://lectures.ms.mff.cuni.cz/view.php?rec=448

3. Petr Marek. OodGAN: Generative Adversarial Network for Out-of-Domain Data Generation. NAACL 2021. https://underline.io/lecture/20111-oodgan-generative-adversarial-network-for-out-of-domain-data-generation

4. Petr Marek. Technologies behind conversational social bots. DISTINCT: Dementia: Intersectorial Strategy for Training and Innovation Network for Current Technology Summer School, Prague Czech Republic 2021.

5. Petr Marek. Alexa Prize Socialbot Grand Challenge and Alquist 4.0, with Petr Marek. NLP Highlights, 2021. https://podcasts.apple.com/us/podcast/132-alexa-prize-socialbot-grand-challenge-and-alquist/id1235937471?i=1000536806786

6. Tereza Pazderová, Petr Marek. Konverzační technologie – jejich vývoj a využití v praxi. Masaryk University Faculty of Arts, Brno 2021. https://www.youtube.com/watch?v=yFK4DnOHRps&ab_channel=Kancel%C3%A1%C5%99e-learninguFFMU

B.2 Industry

1. Jan Pichl, Petr Marek, Jakub Konrád, Petr Lorenc, Ondřej Hrách, Martin Matulík, Jan Šedivý. Jak na Alquista? Národní centrum Průmyslu 4.0, Prague Czech Republic 2019.

2. Jan Pichl, Marek Petr. Alquist. Amper, Brno Czech Republic, 2019.

3. Petr Marek. Proč se umělá inteligence neobejde bez dat? KPMG Data Festival, Prague Czech Republic 2020. https://www.youtube.com/watch?v=rkt-OJgWmBO&ab_channel=xPORTBusinessAccelerator

4. Petr Marek. Difference between research in academia, corporate, startup. Machine Learning Meetup in Prague, Prague Czech Republic 2017. https://www.youtube.com/watch?v=TroM1fwh1ac&t=503s&ab_channel=MachineLearningMeetups

B.3 Popular Science

1. Petr Marek. Alquist PechaKucha. PechaKucha Night Plzeň, Pilsen Czech Republic 2018. https://slideslive.com/38911093/petr-marek

2. Petr Marek. Tajemství tvorby mluvící umělé inteligence. Noc Vědců, Prague Czech Republic 2018.

3. Jan Pichl, Petr Marek, Jan Šedivý. Češi uspěli v soutěži o nejlepší chatbot. Studio ČT24, Česká televize, 2018. https://www.ceskatelevize.cz/porady/10101491767-studio-ct24/218411058311228/?fbclid=IwAR3O4d5owGnDrJgnZssl6OoYgLEKaj1T09QX5S8iCmiabzMYxOA_Xu_TBBo

4. Petr Marek. Klíčem k úspěchu je mezioborová spolupráce. TEDxBudweis, České Budějovice Czech Republic 2019. https://www.youtube.com/watch?v=8xHn4wLIOmg&ab_channel=TEDxTalks

5. Jan Pichl, Jakub Konrád, Martin Matulík, Petr Marek, Petr Lorenc, Ondřej Hrách. Koumákova reportáž: Chatbot Alquist z ČVUT. Planeta Yo, Česká televize, 2019. https://www.ceskatelevize.cz/porady/10315711050-planeta-yo/219553117450001/cast/683144/

6. Ondřej Hrách, Jan Pichl, Petr Marek, Jan Šedivý. Chatbot Alquist. Gejzír, Česká televize, 2020. https://www.ceskatelevize.cz/porady/10805121298-gejzir/220562235000004/

7. Jan Pichl, Petr Marek. S Janem Pichlem a Petrem Markem nejen o jejich chatbotu Alquist. Budoucnost R, Český Rozhlas, 2020. https://radiozurnal.rozhlas.cz/kamarad-ktery-dokaze-dobre-poradit-popisuji-vyvojari-svoji-predstavu-budoucnosti-8271828

8. Vítězslav Boch, Petr Marek. S umělou inteligencí se potkáváme každý den. Kdy nahradí dramatiky a novináře? Střepiny, TV Nova, 2021. `https://tn.nova.cz/zpravodajstvi/clanek/429057-clovek-vs-stroj-poznate-kterou-reportaz-napsala-umela-inteligence`

9. Petr Marek. Konverzační AI ted funguje dobře, když člověk spolupracuje. Stewdent Podcast, 2021. `https://www.youtube.com/watch?v=TXGEPc3Vg3k&ab_channel=StewdentPodcast`

10. Petr Marek. Co všichni mají na těch chatbotech? Noc Vědců, Prague Czech Republic 2021. `https://www.youtube.com/watch?v=YQYXXd1fjRA&ab_channel=CIIRC%C4%8CVUT`

11. Petr Marek. Vědci využijí strojové učení na analýzu řeči velryb. Český rozhlas 2021. `https://www.facebook.com/watch/?v=432304241690555`

12. Petr Marek. Kdy už si s umělou inteligencí pořádně popovídáme? Neboli chatboti, voiceboti, socialboti. Týden Inovací 2022.

13. Tommaso Gargiani, Petr Marek, Ondřej Kobza. Alquist ve finále Amazon Alexa Prize. Český rozhlas Plus, 2023.

Appendix C

Awards

1. Jan Pichl, Petr Marek, Jakub Konrád, Long Hoang Nguyen, Martin Matulík, Jan Šedivý. Amazon Alexa Prize Socialbot Grand Challenge 1, 2017. Second Place Winner.
2. Jan Pichl, Petr Marek, Jakub Konrád, Martin Matulík, Petr Lorenc, Jan Šedivý. Amazon Alexa Prize Socialbot Grand Challenge 2, 2018. Second Place Winner.
3. Jan Pichl, Petr Marek, Jakub Konrád, Martin Matulík, Petr Lorenc, Ondřej Hrách, Radka Fléglová, Jan Šedivý. AI Awards, 2018. Projekt Roku 2018.
4. Jan Pichl, Petr Marek, Jakub Konrád, Petr Lorenc, Van Duy Ta, Jan Šedivý. Amazon Alexa Prize Socialbot Grand Challenge 3, 2020. Third Place Winner.
5. Jakub Konrád, Jan Pichl, Petr Marek, Petr Lorenc, Van Duy Ta, Ondřej Kobza, Jan Šedivý. Amazon Alexa Prize Socialbot Grand Challenge 4, 2021. First Place Winner.
6. Ondřej Kobza, Jan Čuhel, Tommaso Gargiani, David Herel, Petr Marek, Jan Šedivý. Amazon Alexa Prize Socialbot Grand Challenge 5, 2023. Third Place Winner.