

Becker, Marcus; Prokop Dayrell de Lima, Erika

Article

Natural Language Processing in IT Ticketing Systems. A conceptual framework for Question-and-Answering machines based on GPT-Algorithms

Research Journal for Applied Management (RJAM)

Provided in Cooperation with:

International School of Management (ISM), Dortmund

Suggested Citation: Becker, Marcus; Prokop Dayrell de Lima, Erika (2023) : Natural Language Processing in IT Ticketing Systems. A conceptual framework for Question-and-Answering machines based on GPT-Algorithms, Research Journal for Applied Management (RJAM), ISSN 2701-6633, BoD – Books on Demand GmbH, Norderstedt, Vol. 4, Iss. 1, pp. 133-158

This Version is available at:

<https://hdl.handle.net/10419/324733>

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.

Becker, Marcus; Prokop Dayrell de Lima, Erika

Natural Language Processing in IT Ticketing Systems

A conceptual framework for Question-and-Answering machines based on GPT-Algorithms

Abstract

This research is a feasibility study to design a Natural Language Processing (NLP) system within a Question-and-Answering (Q&A) environment for internal IT help desk ticketing operations. The proceedings will be used to develop a conversational agent for an IT consultancy company. Tests with few-shot learning algorithms were performed by calibrating two GPT-2 language models. Another benchmark model was tested on the most recent GPT-3 standard. The input data stems from software license release requests of an internal IT help desk. The final model will be a hybrid approach of first guidance by an automated agent and a human expert intervention for more complicated IT problem. The agent will improve itself by constantly evaluating user feedback.

Keywords:

Natural Language Processing, Question Answering, GPT-2, GPT-3, Few-shot learning, Natural Language Generation, Conversational Agent Implementation

JEL Classification Numbers: C67, C88, M15, 031, 032

Acknowledgments

We would like to thank all the experts who contributed to our research with their valuable feedback. Special thanks to Jörg Wiethölter, Jürgen Angele, and Sascha Schramm.

1 Introduction

The human-machine interaction is gaining more and more awareness in daily business operations – especially in times of ongoing digitalization processes. This interaction is not without flaws. In case of disruptions, the IT help desk usually is the first point of contact for internal employees who are experiencing IT problems. Interruptions of work processes are costly for the company. Apart from this, they pose a potential source of accumulating frustration on user side. Therefore, it is pivotal to react promptly to user requests. According to a research conducted by “Forrester Research” in 2020 organizations that have adopted help desk services, increased their overall employee satisfaction and managed to achieve on average a self-service resolution rate of 20%.¹ IBM, the global market leader in IT infrastructure solutions, sees the following major contributions of IT help desks: Service desks standardize IT processes, and automate routine tasks, thus increasing operational efficiency. They align users with IT processes by offering self-service tools and automated workflows that further increase business productivity and free IT staff to focus on more strategic activities. Service desks act as a “hub” for all hardware and software assets in the company. Moreover, they create digitalized contracts, and enable easy information retrieval, IT configuration, and IT maintenance. Finally, IT service desks keep business activities up in times of technical riots, thus avoiding costly downtimes in operations.²

Due to increasing demand for technical support, the IT help desk has to organize its operations in form of a so-called “ticketing system”. A majority of tickets, like software licensing issues, are constantly recurring problems, with pre-existing solutions. Hence, it would constitute a tremendous relief for IT specialists, if such easy-to-solve tickets could be resolved automatically, e.g. in form of a conversational agent, thereby releasing the costlier human IT resources.

The worldwide adoption of the IT Infrastructure Library framework (ITIL) has already shown some effectiveness in improving service desk operations. However, successive improvements, for many IT service providers, still remain a major challenge (Jäntti et al. 2012). The goal of IT service desk agents is to record, classify, diagnose, and resolve user requests (Jäntti et al. 2012). Requests consist of the following: software and hardware failures, requests for resetting passwords, requests for new software packages and licenses, or change requests for existing services. Finally, the help desk constantly receives feedback for its operations. Feedback can be positive as well as negative. An important step in improving operations would be to ensure effective communication between the service desk and the user. This includes a continuous overview of the progress of ticket resolution, which has to be preserved through all possible channels (Jäntti et al. 2012). Moreover, timely ticket resolution increases user satisfaction.

¹ See Charles Betz, Andrew Hewitt, “Service Desk 2020: It’s All About Employee Experience,” Forrester Research, October 6, 2020.

² See <https://www.ibm.com/topics/service-desk>, last accessed on 12.07.2023, 16: 46.

In this study, we design a feasibility study for a Natural Language Processing (NLP) system which operates within a Question-and-Answer (Q&A) environment for first-level IT support. By using the most recently developed Generative Pre-trained Transformer technology (GPT) as a few-shot learning method. Few-shot learning scenarios originate from past IT ticket resolutions. For calibration purposes, we use the pre-implemented open-source-based, *GPT-2 packages gpt-2-simple and german-gpt2*. In a later run of our analyses, we were also granted access to the most recently developed GPT-3 standard called *Playground*.

Task-oriented dialog systems have encountered increased awareness since they can assist users in various, predominantly monotonous, tasks. In a “typical” task-oriented dialog system, the Natural Language Generation (NLG) module converts a system action (e.g., often specified in a semantic form selected by a dialog policy) into a final response (Peng et al. 2020). The response should be adequate, fluent and represent a semantic dialog action to engage users’ attention. Existing methods for NLG can be subdivided into two major categories. (1) Template-based methods require domain expert knowledge to create templates for each domain, with the system filling in so-called “slot values” (Cheyer and Guzzoni, 2014; Langkilde and Knight, 1998). (2) Statistical language models, as presented by Gao et al. (2019), can be built on neural networks. The models are trained on a labeled text corpus. One canonical model is semantically conditioned, such as SC-LSTM (Wen et al. 2015), which encodes dialog acts with one-hot representations and uses it as an extra feature to inform the sentence generation process. The general performance is good but the training amount needs a large amount of domain-specific data. Moreover, it suffers from scalability as dialog combinations grow exponentially.

During our study, we collaborated with a large-scale IT consultancy positioned in Dortmund (North Rhine-Westphalia, Germany). The company already has a pre-existing conversational agent solution that could be enriched by our GPT module. For this, we need a structured example file of previously resolved tickets. Our first tests were appealing with respect to human (qualitative) judgment. The answers fitted well to the respective questions. Even first guidance advices were provided. To give an example: After requesting a license for a specific software product, the agent informed the employee about the costs and the need for approval by a Project Leader or a Manager (see figure 7 in the final evaluation report section). The next step would be to program a conversational agent which directly sends an email to the manager asking for approval. We do not claim that our tool has the ability, nor the intent, to replace skilled human IT resources. But as our anecdotal evidence shows, IT support vastly suffers from work overload. This overload trend will much likely aggravate due to an increasing need for internal digitalization projects and decreasing numbers of skilled workers in the nearby future. Therefore, our work represents a feasibility study that aims to explain how NLP systems can be used to relieve human IT support resources for more complex IT projects. Projects which cannot be resolved by an automated agent.

As prior research conducted by Tarlaci (2019) indicates, unsupervised language models' like GPT-2 have a proud capability to optimize automated Q&A tasks with minimal or no task-specific adaptation. The first public release of GPT-2 (with 124 M parameters) was fine-tuned for a special Q&A task, using the Stanford Question Answering Data-set (SQuAD) 2.0, Huggingface's PyTorch implementation and a modified BERT as Q&A model. The fine-tuned model did not meet the researcher's expectations, regardless of further adjustments on tokenization, hyperparameter fine-tuning and increasing number of epochs. The author indicates that the size of the language model, as well as the complex nature of Q&A tasks in general, are two limiting factors for future deployments of such model. The author further concluded that a more sophisticated fine-tuning experiment, with more performant extraction systems, could improve future results.

Fiore et al. (2020) emphasize the unexploited potential for conversational agents inside a company as Virtual Enterprise Assistants (VEA). Focusing on VEA for in-company IT help desks, a user study was performed with a functional conversational agent prototype to evaluate the suitability for typical IT support cases, e.g. in password resets, answers to Frequently-Asked-Questions (FAQs), and release email attachments from quarantine. In accordance to that, the researchers studied the potential user acceptance. The study was conducted with 12 employees from two companies, a bank and a hospital, that were specifically chosen due to their IT affinity. The results indicate that such an in-company conversational agent is well-perceived by employees. However, the less structured task for answering FAQs aroused negative remarks regarding misclassifications of inputs. Some users had to rephrase their original request since the conversational agent did not understand their first request.

A consecutive work of Fiore et al. (2020) underlines the capabilities of conversational agents to relieve employees of repetitive work in in-house support. Moreover, the authors indicate success conditions for embedding conversational agents in the company's IT landscape. First, it helps to mention that the agent is not human at the beginning of each conversation. Second, a direct offer to escalate the users' concerns to a real human person should be provided. Finally, the authors point out that special Natural Language Understanding (NLU) techniques are necessary for a successful development and maintenance of conversational agents, a shortcoming that most likely has been overcome by the most recently developed ChatGPT 3.5 technology.

A pioneer study of exploiting state-of-the-art pre-trained language models for NLG in task-oriented dialog systems was performed by Peng et al. (2020). The model is semantically-conditioned on GPT-2, named SC-GPT, and built upon Huggingface PyTorch Transformer technology with 345 M parameters. It follows a three-stage training procedure: first, like GPT-2, massive plain language pre-training, second an additional controlled pre-training stage on vast amounts of dialog-act labeled data, in order to allow the generation of texts, third a fine-tuning stage, on a specific task with a limited amount of domain labels. According to Peng et al. (2020) the results indicate that with minimal training examples,

SC-GPT is able to generate adequate natural language responses. The authors trained their model on a new benchmark dataset, called “FEWSHOT-WOZ” based on the “MultiWOZ” dataset developed by Budzianowski et al. (2018) and the “Cambridge NLG datasets” of Wen et al. (2016). The resulting dataset consists of seven domains (Restaurant, Laptop, Hotel, TV, Attraction, Train, Taxi). Although, the domain of “Laptop” might have some references relating to IT help desk topics, we doubt, that the model would be feasible to handle real-life IT help desk problems on a daily basis, especially for highly specialized IT topic for a technical consultancy firm. Moreover, as most scientific language models, the model presented in Peng et al. (2020) is calibrated on stylized data that has been pre-processed with additional speech tags that alleviates the model training but needs additional man power to either manually incorporate speech tags, or in case of automated speech tagging, binds some resources of validating these tags. Our feasibility study aims at developing model that incorporates some “Natural Language Understanding” (NLU) components that do not need additional tagging efforts.

To the best of our knowledge, the realizations of conversational agents in real-life companies using the most recent GPT technology are sparse. On top, IT ticketing services pose an extra challenge due to the complexity of the underlying IT problems, especially, for the higher demands of an IT consultancy.

Our research paper is organized as follows. Section 2 presents a general overview of our model architecture. In Section 3, we develop our main research question, i.e. our objective to implement a GPT-based NLP system for Q&A tasks in IT support processes. Further, we design a feasibility study on how to incorporate this NLP system in a large-scale IT consultancy company. We provide a technical guideline and highlight methodological questions, which should be resolved before starting with the IT implementation. Section 5 outlines the model limitations and opens up for the debate in future research. Section 6 summarizes the paper.

2 Natural Language Processing in Question-Answering-Systems

2.1 General model architecture

A very general conversational agent architecture can be partitioned into the following components: NLU, Dialogue Management and an additional NLG component (Adamopoulou/Moussiades 2020). Taking into consideration the pipeline architecture represented in figure 1, where each Q&A turn is processed by employing a five-step procedure. The transcriptions of user inputs are first processed by the NLU module, where the user intention and other key information are extracted. Afterwards, the information is formatted to the dialog management component, i.e. question, statement, or a request for action, is the input to the NLG, through which a system response in natural language is generated (Peng et al. 2020). Finally, the answer is provided to the user by text, voice or image, according to the interaction methods used in the conversational agent. The agent asks follow-up questions after the

intent is recognized (Kucherbaev et al. 2018). It can be subdivided into a dialogue state tracking component, responsible for monitoring the mapped intent (and entities) until that point of the conversation, and a dialog policy component, which produces a dialog act based on the fact (or entities) retrieved from external resources, such as database or a knowledge base (Peng et al. 2020).

A database serves as part of the backend, where all relevant information is retrieved. Once the appropriate information is extracted. It is forwarded to the Dialog Management Module and then to the Response Generation Module (Adamopoulou/Moussiades 2020). The response to the user is generated based on the dialog act emitted by the Dialog Policy Module. Several approaches can be utilized for generating the user response, including Rule-based, Retrieval-based, and Generative-based methods. In the Rule-based model, the response is selected from a predefined set of rules without generating new text responses. The Retrieval-based model offers more flexibility by selecting the most suitable response through the examination and analysis of available resources (Adamopoulou/Moussiades 2020). The Generative model employs Natural Language Generation (NLG) to generate responses in a human-like natural language manner, considering the previous inputs (Peng et al. 2020; Adamopoulou/Moussiades 2020). Hybrid conversational agents combine a rule-based or retrieval-based approach with a generative approach to handle user inputs that do not match any predefined rules mentioned earlier. Furthermore, the operation of a conversational agent can involve human intervention as well (Mathur/Lopez 2019; Adamopoulou/Moussiades 2020).

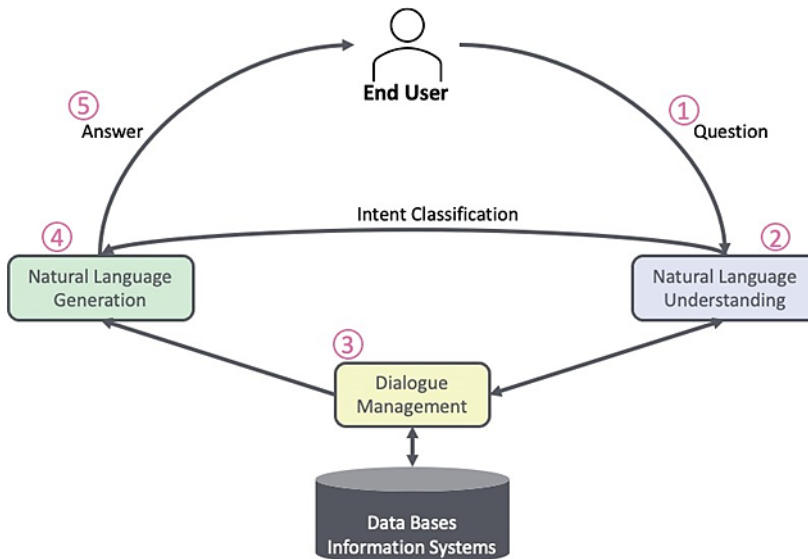


Figure 1: General Architecture of Q&A system based on text generation
Source: Hirschman/Gaizauskas 2001; Peng et al. 2020; Adamopoulou/Moussiades 2020

2.2 GPT Technology

2.2.1 General Framework of GPT

The Generative Pre-trained Transformers (GPT) are a series of language models developed from the Transformer model. Transformer models are a series of connected encoder and decoder nodes (Vaswani et al. 2017). Thus, they belong to a most recently developed sub-category of Deep learning models that are pre-dominantly used in the field of NLP. Transformers thereby replace the previously used Recurrent Neural Network (RNN) models, such as Long-Short-Term-Memory (LSTM), and Convolutional Neural Networks (CNN) models.

GPT is an NLP model based on unsupervised machine learning methods. Without any particular task-specific training and on any data subject, the GPT-2 model can read and “understand” a text, summarize or transcribe it. GPT-3 is able to answer questions about the information contained in it. Moreover, it is capable of completing, or even generating paragraphs with grammatical and syntactic consistency. GPT- assigns a probability to a sequence of words for a given size. It obtains a probability of occurrence for each sequence determined by previously seen training data. Training data is collected and retrieved from an extensive number of web sources. The notable innovation of GPT-2 consists in its learning method: “*Zero-shot-learning*” covers a technology where text, that has never been learned before, is inserted, thus starting from zero, and which evaluates answer’s consistency. Instead of rearranging the sequences for fine-tuning, as previously conducted by GPT-1, the input to GPT-2 provides a format that expects the model to understand the nature of the task. This method uses the embedding of classes and words, associated to a vector format, thereby enabling the algorithm to recognize unrepresented text. This process is realized by associating text with other previously learned topics or words. Most state-of-the-art NLP models were trained for specific tasks, using supervised learning methods. However, these traditional Machine Learning models need a large amount of data to learn and fail to generalize tasks other than the ones they were trained on.

The architecture of GPT-2 is a 12-layer Decoder-only Transformer, trained within the *attention framework*³, rather than foregoing recurrence and convolutional-based architectures. GPT-2 is able to generate coherent (i.e. understandable by humans) paragraphs of text. The technology accomplishes state-of-the-art performance on language modelling, thereby outperforming benchmarks as RNN/LTSM/CNN-based models. GPT-2 performs basic reading comprehension, machine translation, Q&A and summarizing tasks. The model use cases range from writing grammar assistance and auto-completion (for ordinary and coding purposes), to more advanced applications such as conversational agents, creative writing engines, unsupervised translation models, and enhanced speech recognition

³ Attention mechanisms, as proposed by (Vaswani, 2017), enable the prediction model to discriminatively concentrate on sections of input text which are most important. Thus, attention mechanisms allow the modelling of dependencies regardless to their distance in the input or output sequence.

systems. The model was first released in stages using 124 million, 355 million, 774 million and finally 1.5 billion parameters, due to the authors concerns for malicious purposes, which could be generation of misleading news articles, impersonation of others online, automated production of abusive or faked content to post on social media and spam and phishing content (Radford et al. 2019).

2.2.2 GPT modules for task-oriented conversational agents

The majority of current dialog systems provide semantically correct responses; however, they strive to correspond to the variation and fluency in human language. NLG is the last and major phase to generate user-facing system speeches. The output of NLG is closely tied with the perceived quality and usability of a dialogue system (Zhu et al. 2019).

Current generative networks are trained on voluminous amounts of data providing them with the capacity to generalize to unseen sample (or test) data. However, once these models are deployed, there is no control over the generation. Although, transformer-based models have achieved good results on a range of NLP tasks in recent years, the lack of control remains a challenge.

To tackle the above problem, controllable text generation procedures have been established. The attributes to control can be of the following types (Prabhumoye et al. 2020):

- Stylistic Attributes such as politeness, sentiment, formality
- Demographic attributes such as gender or age
- Contextual attributes such as information, keywords, or entities
- Ordering of information such as events, e.g. plot summaries etc.

The majority of controllable text generation tasks can be framed as conditional language generation tasks and can be performed at various stages. Prabhumoye et al. (2020) divide the generation process into five modules, which can be used. As illustrated in figure 2, the generation pipeline starts with an input or a source sequence and an output or a target sequence. The first input module, i.e. the external input module, is responsible for the initialization h_0 . The second input module, i.e. is the sequential input module, denoted by x_t , where t is a time index of the generation process. The third module is the generator operations module. It is responsible for carrying out consistent operations or calculations on all the inputs at each time step. The fourth module is the output module. The output o_t is further projected on to the vocabulary space to predict the token \widehat{X}_t at each time step. Finally, the fifth module is the training objective module. This module deals with the loss functions used for training.

There are various techniques used to modulate each of the five modules to perform controllable text generation sequences. In dialogue systems it would be meaningful to control the response generation

according to the semantic action. In other words, the response is conditioned on the dialogue act. For this purpose, the generator operation (module 3 in figure 2) should be considered. This module takes in the external input h_0 , the sequential input x_t at time step t and performs the same set of computations G to return an output o_t . Changes can be made to the set of operations G in order to include a so-called control vector in computing o_t .

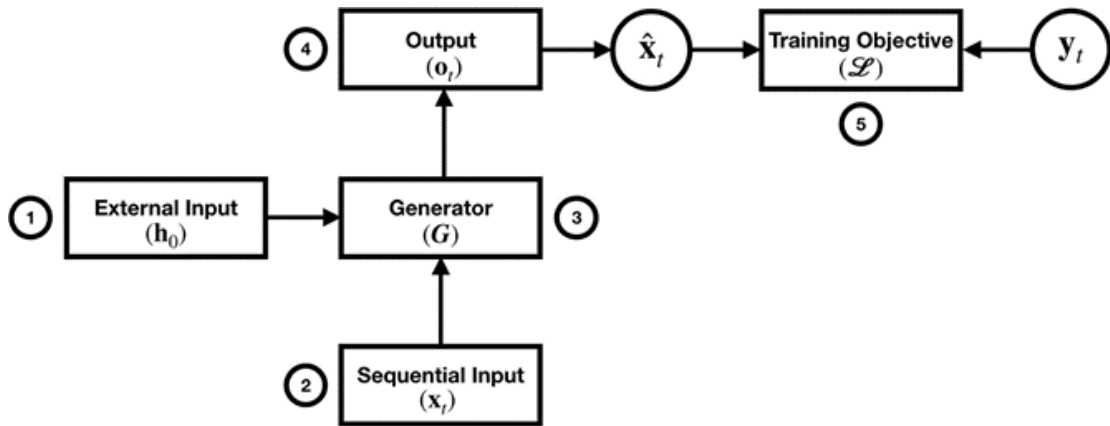


Figure 2: Modules for controlling a natural language generation task
Source: Prabhumoye et al. 2020

RNNs, LSTMs, and Gated Recurrent Units can be used to model controllable text generation tasks. However, these techniques still have trouble in handling long text sequences. As a result, attention mechanism algorithms are employed on the source sequence, as discussed in Prabhumoye et al. (2020). By incorporating self-attention, Transformers possess the capability to process input sequences of varying lengths and effectively capture the interdependencies among different slots. Peng et al. (2020) propose a multiple task and domain learning on large amounts of annotated (dialog act, response) pairs to the transformers pre-training stage. This optimization allows the developer to semantically condition the generation of responses of GPT models, and to adapt it to the task specific domain.

In order to improve the language model's behaviour, an iterative process named "Process for Adapting Language Models to Society (PALMS) with Values-Targeted Data-sets is used to fine-tune the model on a curated dataset. Empirical evidence shows that adjusted language models perform well even on small and hand-curated datasets. Additional tests conducted with GPT-3 even show considerably more robust results. The approach is promising and might be considered to further works with future GPT versions (Solaiman/Dennison 2021).

2.2.3 Our proposed hybrid solution for an IT-Support Q&A-System

In accordance with the internal guidelines for system development, the solution is to be developed as a micro-service architecture⁴, thus enabling re-usability and integrability with other applications. The proposed solution enables the end user to have a two-way text conversation with a conversational agent, without having to generate a ticket each time the users ask for assistance. Combining different technologies such as machine learning, NLP and integrated dialogue tools, our hybrid solution is able to maintain a conversational flow between applications and users.

The rules to trigger either the Rule-Based NLG module, or the GPT module, should be defined in advance. There are different possibilities for defining policies. One solution is to trigger GPT via a fallback intent. If no intent is found the conversational agent passes the user request to the GPT model. This would require GPT to be more generic, i.e. fine-tuned with several kinds of annotated response-request pairs to condition the answers, thereby following a dialog flow format. Another solution is to specifically define intents, which will trigger the GPT model to generate an answer. This alternative has nameable benefits, for instance a specific GPT model can be used and fine-tuned with response-request pairs extracted from the IT support's database. A predefined response text is not needed. Instead the output of GPT is passed (via conversational agent) as an answer.

After receiving user input through an application, it can be transmitted to the backend using, for example, a REST API. This standard procedure is frequently employed in enterprise settings, facilitating seamless integration of the system into existing frameworks and thus enabling expansion through in-store methods. The conversational agent plugin establishes communication with the application to retrieve the relevant input and obtain the corresponding output. Subsequent actions are carried out within the conversational agent plugin, and the resulting output is then sent back to the application. The conversational agent retains the context of the conversation and the user, enabling it to execute the necessary actions based on this information.

Through the conversation with the conversational agent, the user can receive faster responses compared to the traditional Information Technologies Service Management (ITSM) application structure. If the answers are not satisfactory, the user can ask the conversational agent to generate a ticket. This ticket will be assigned to the IT support and it will be identified with the user's name. The entire conversation history will be stored to support the request follow-up, allowing the subsequent IT specialist to understand the context of the underlying query. Thus, the ticketing system will be maintained. However, the interaction method should be changed from e-mail to conversational agent, which would perform the service at first level, trying to offer answers to FAQs. The user can access the conversa-

⁴ Micro-services approach to modularize software in systems which are divided into smaller modules. A micro-service should only be able to perform a single task and communicate with other programs. This enables micro-services to interact with each other (Wolff 2018).

tional agent application and directly ask a question, the conversational agent parses the natural language input, as provided by the end user, and triggers one of the NLG modules, according to the identified intent. This answers back with the solutions in the dialogue. Together with the given answer, a user feedback is asked, e.g. a question like “Was this answer helpful?” or “Was your question answered?”. If the feedback is negative (dashed line number 5 in figure 3), the conversational agent should escalate the request to second level support by triggering a creation of a ticket to the (human) IT-Support agent. Additionally, in case that the intent cannot be identified (a so-called *fallback*), the creation of a ticket to the IT-Support can also be triggered. This ticket can contain the conversation history, thereby ensuring that the technician is informed. In this phase the data for Q&R- pairs can be collected and used to enrich the IT service database. Note, that both negative, as well as positive, feedback can be considered, i.e. the first one indicating bad quality, or incorrect answers, and the second one indicating good quality, or correct answers. In this feasibility study our first focus lies on negative feedback loops to exclude early frustration potential on employees’ side. In future adoptions, positive feedback loops can be incorporated to increase the probability that the algorithm will give similar advices for similar requests. This represents a type of reinforcement learning strategy to improve our language model in next generations. The overall target will be to maximize user satisfaction. Finally, the IT Support agent proceeds with the process by categorizing the ticket, working on it and replying to the end user. The communication channel, once a ticket is created, can follow the standard e-mailing process. If the conversational agent is unable to answer the user question, thus creating a ticket to the second-level IT Support, the user can receive both, a notification with the ticket number, as well as a response from the IT Support agent. Consequently, the only change on the current ticketing process is the conversational agent, which is creating tickets in name of the end user. The whole process is visualized in figure 3 once again.

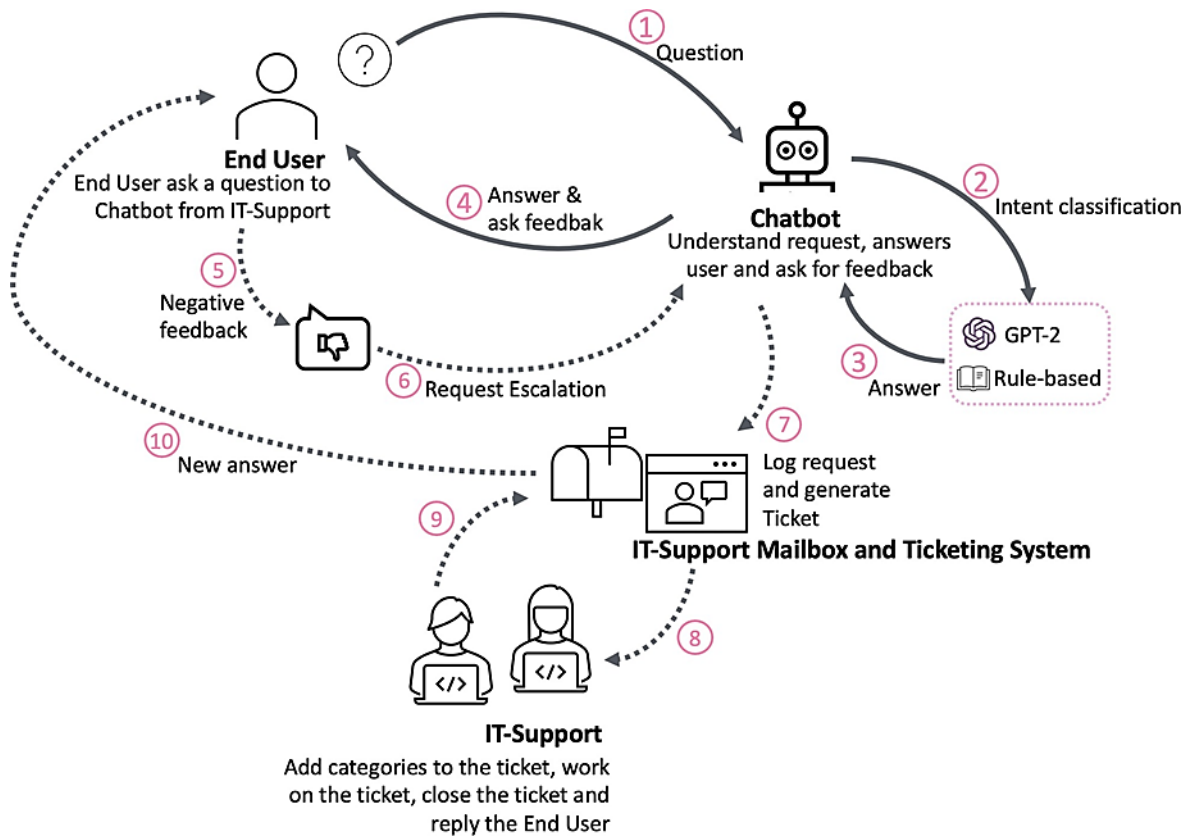


Figure 3: IT-Support process with a hybrid conversational agent for Q&A pairs
Source: authors' own illustration

Subsequently to the process implementation (including data collection for training purposes), the solution can evolve to a hybrid semantically conditioned conversational agent. The data from the initial phase can be used in an additional pre-training stage to semantically condition GPT generation; this would increase the quality of answers, as it would be more domain specific. Feedback loops (both positive and negative) should be continued.

3 A mini-workplan for implementing large-scale NLP models in IT Ticketing Systems

3.1 Towards developing a GPT-based System for Q&A tasks

This paper focuses on the IT support unit. Our approach aims to reduce the workload of a human IT specialist in order to work on larger scale IT problems. Moreover, faster replies increase the user's satisfaction, and vice versa, reduce opportunity costs resulting from ongoing conversation. We propose the use of a transformer-based NLP model, in this particular case, GPT. GPT is embedded in an

already structured IT ticketing process. We experiment with versions of GPT and further explore GPT's potential to work in a conversational agent solution for Q&A tasks. Our objective divides into the following stages:

- 1) Defining data sources and extraction methods: In order to enable our model to answer questions sent from the end users to the IT-Support, a sample of the data is needed. The first step is to evaluate how data is stored and how a sample can be extracted. Afterwards, pre-processing steps should be applied such as tokenization, removing numbers and extra white spaces.
- 2) Selecting a suitable GPT model and using few-shot learning methods on extracted data from the IT support: In this paper, we will make use of the Huggingface's repository to obtain a pre-trained GPT model. After selecting a model, the environment to perform the few-shot learning has to be configured to operate on the extracted sample from IT-Support.⁵
- 3) Compose a preliminary feasibility analysis of a Q&A System: We motivate the viability of using a hybrid solution. It aims to lead the tickets, which are currently assigned to human attendants, to an automated service. The idea is to illustrate that this filtering at the entrance and in the ticket ward will be able to promote the optimization of the resources used within the IT support unit. It further improves the quality of the service, thereby meeting the growing need of an instant communication tool to provide better user experience and to increase response capacity.

We propose to introduce GPT in the existing conversational agent model process. It is one of the most recent technologies for developing NLP models based upon transformers. GPT intends to take another step towards relating users and service providers. This approach allows the IT support to re-use resources, thus increasing ticket solution capacity.

3.2 The original IT Ticketing Workflow

The IT support unit consists of a team of specialists, who are the first point of contact for all internal and external users needing information technological guidance. If the user has not the ability to interact with the system, e.g. in form of a Q&A machine, the sole option is to generate a "ticket". That ticket is assigned in a queue until it is finally assigned to one of the attendants. This stage can take hours to days. A delay has severe consequences. Since it often prevents the user from continuing his or her work. However, in a variety of circumstances, the underlying problem belongs to a series of repeating questions that, if stored automatically, could have been resolved easily, and in a timely manner, by sending an automated reply emerging from a corpus of pre-trained solutions. The simplified process

⁵ The model as well as its checkpoints have to be imported. The checkpoint should be saved within "run1" folder since we are not training the model. Note, we only perform a few-shot learning algorithm. Further, the parameters of the model have to be defined, as well as its context with examples of Q&A pairs.

can be described as follows: The end user sends an email with a question (all end user requests, such as ordering Hardware/Software, Directory Services, Account configuration, granting Licenses/Subscriptions, Network Services, and more, will be referred to as Question) to the IT-Support mailbox. The Ticketing System, which runs in the background of this mailbox, identifies this email and generates a ticket associated to this specific user and question. This ticket is picked up by a member of the IT-Support team, who manually selects categories related to this question and starts working on the ticket. The end user has also access to the Ticketing System for tracking the status of the ticket (the status can be either open, in progress, or closed). When the ticket is resolved, IT support answers the end user and closes the ticket. This answer goes as an email response to the user. For sake of simplicity, only the 'happy flow' will be taken into account. Hence, scenarios where the IT-Support has to ask the end user for more information, tickets will be "escalated" to another IT Support agent. Further, the ticket has to be approved by a manager. Otherwise, the ticket cannot be marked as resolved. Figure 4 below illustrates the previously described process.

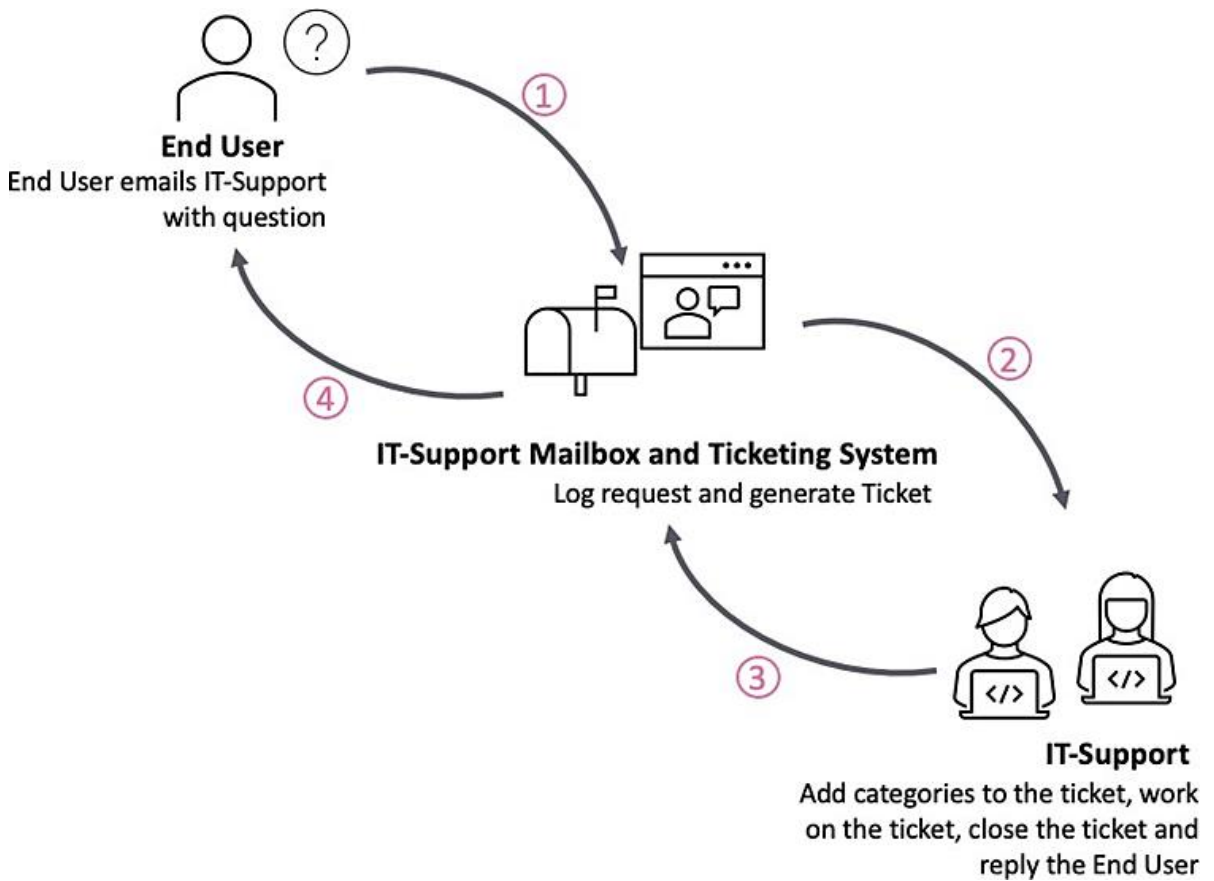


Figure 4: Q&A process within an IT ticketing system
Source: authors' own illustration

3.3 Data Gathering and Pre-processing

The data is stored in a semi-structured format in a MS-SQL-Database. The process is depicted from a data perspective in figure 5.

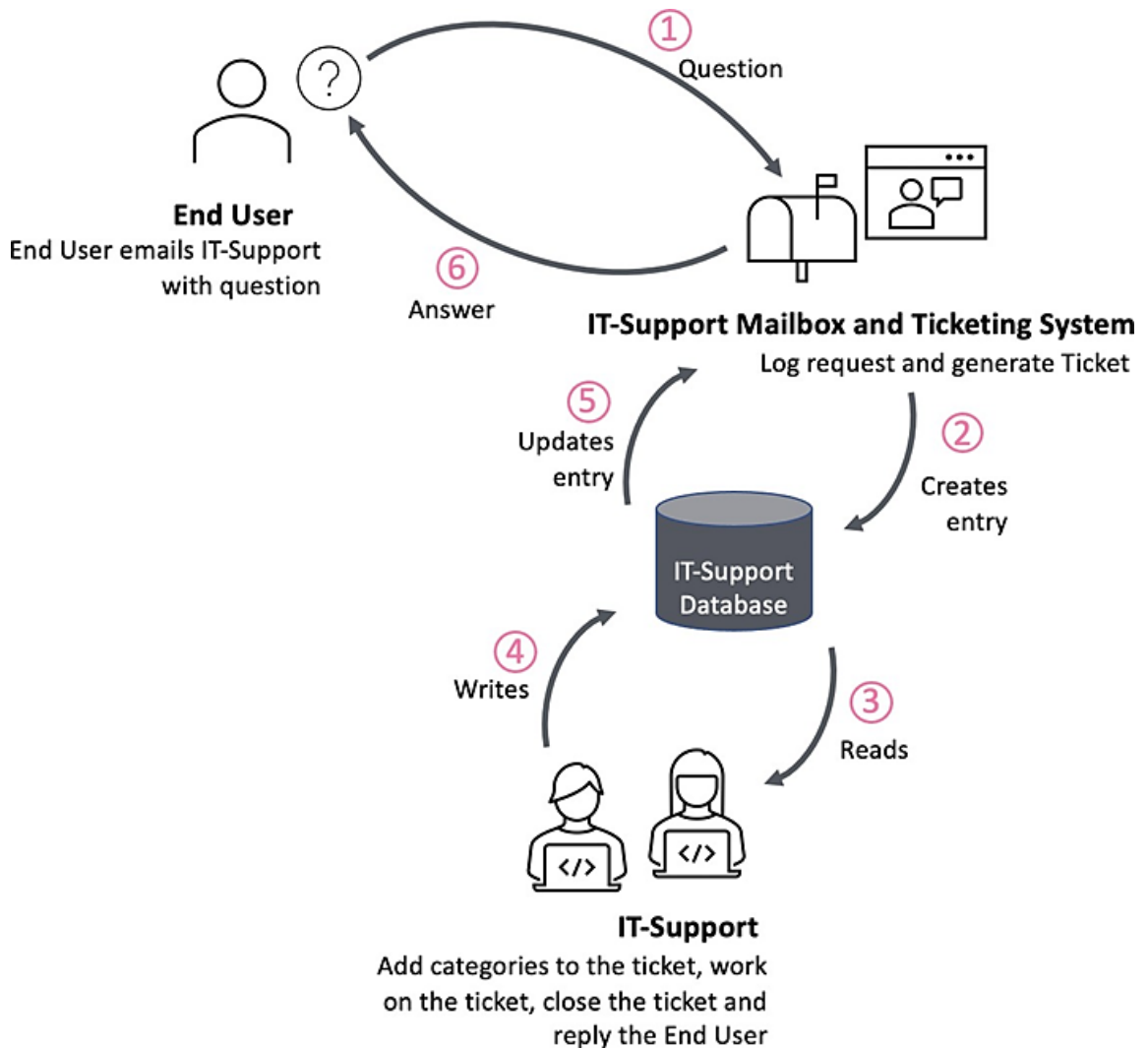


Figure 5: Exemplified data flow within a Q&A ticketing system
Source: authors' own illustration

The definition of data sources and extraction methods was conducted with the collaborating IT consultancy firm. Assessing data in a large-scale company demands considerable effort, due to restrictions, volume, and variety. As previously indicated, the data from the IT support is stored in a semi-structured format. All data is in German language. However, since most requests belong to an IT affiliated domain, most terms are in English, e.g. software names or licenses. Direct access to the database

was not granted due to the highly sensitive characteristics of the user data. In order to enable further analysis, i.e. on a “manageable” amount of data, a random sample with 733 MB of data and 76,508 observations, was generated.

To perform few-shot learning on the IT support dataset, a GPT model has to be configured to be able to answer in a dialog flow format. For this, training demonstrations are required that need additional pre-processing. Note that pre-processing is a very complex and time-consuming task in general. It is not the main topic of our paper. Therefore, only some basic procedures were adopted to meet the minimal viable criteria of feeding the model. The pre-processing was performed by Python by using the following libraries: pandas (for dealing with the tabular data frame), hashlib (for creation of cryptography hashes and message digests), and json (JavaScript Object Notation, a standardized format, such as XML and YAML, used to store and exchange data).

We have filtered only questions with less than 300 characters and related to License, tokenized the data by adding additional “Id”, “Request”, and “Response” columns and discarded all other columns. The pre-processed dataset was saved into a “json” file format with 352 lines, one for each observation that was inside the previously mentioned criteria.

4 Model Deployment and Evaluation

4.1 Objective Setting

The object of this paper, NLP for Q&A systems, will focus on the deployment within the current IT support process. We intend to present a hybrid solution, built upon a generative model which is able to lead the IT tickets, that are currently associated to human attendants, to an automated service.

Our analysis proposes the implementation of a conversational agent, for answering IT questions by using the GPT-2 technology. With this, a semi-structured component is brought to the predominantly human-orientated IT support. We believe that our solution has tremendous demand in software releases and licensing operations.

4.1.1 Task definition

In the following, we want to focus on IT questions related to software licenses. The choice was not only made, as first filtration in form of a feasibility study, but rather as an opportunity to automate an inherently structurable task. The contributing tasks are defined as follows:

1. Few-shot learning with *gpt-2-simple* and *german-gpt2*, using data from the IT support as few-shot demonstration;
2. Preliminary feasibility analysis for an IT support Q&A-System by using generative pre-trained transformers as underlying NLG module.

4.1.2 Technological decisions

For this study, several language models written in Python. The initial decision to go with a transformer-based language model was further refined into either an encoder only transformer, namely BERT (Bidirectional Encoder Representations from Transformers), or a decoder-only transformer model, specifically GPT-2 and GPT-3, by using the libraries *gpt-2-simple*, *german-gpt2* and *gpt3*.

The initial intention was to use an encoder-only transformers model, more specifically BERT (Bidirectional Encoder Representations from Transformers). BERT could be used to extract the answer from a span of text and provide it to the end user. The first problem with BERT is that the input is limited to 512 tokens (only). The second issue is that there is no available knowledge base for BERT to search its answer from. Therefore, using BERT on a large knowledge base would require 1) to find documents that are similar to the question, 2) to apply BERT to the most similar document, which would withdraw additional time resources from the company, 3) to use BERT as a classifier, to build FAQs from the past e-mails sent to the IT service desk. However, considering the given dataset, in combination with the computational restrictions, the GPT-2 (decoder-only transformers model) was assumed to generate more “tangible” results.⁶

The GPT-2 models used in this paper were taken from the Huggingface repository. All models within this repository are licensed under MIT⁷. For the programming framework, Google Colab⁸ was not a feasible solution, first of all due to data protection issues, secondly because the free version of Colab does not have sufficient TPU/GPU nor space (only 6 GB were available on Google drive) to train the model and save checkpoints. The models were implemented locally, on a Mac computer with 16 GB RAM, by using PyCharm⁹. PyCharm was used for programming in Python within Conda.¹⁰ Two virtual environments were created, one for each GPT model, with different Python interpreters in order to comply with the models' requirements.

⁶ We further tested and trained a model on the more recently developed GPT-3 counterpart, which is now commercialized but yielded significantly better results as we will see in section 4.3.

⁷ MIT license permits free of charge to any person with the copy of a software and associated documentation to deal with this software without restrictions to use, copy, modify, merge, publish, distribute, sub-license, and/or sell copies of the Software. For further disclaimer information see <https://mit-license.org>.

⁸ Colab is a web-browser cloud application used for writing and executing code in Python. Colab notebooks are Jupyter notebooks hosted by Colab.

⁹ PyCharm is an integrated development environment for programming in Python language.

¹⁰ Conda is a package and environment manager that can be used with command lines at the Windows prompt, or in a terminal window for Mac OS and Linux.

4.1.3 Pytorch vs. Tensorflow

In this topic we will assess a comparison between the Python Machine Learning frameworks of PyTorch and TensorFlow. On this paper both PyTorch and TensorFlow implementations of transformer models were tested.

Both frameworks run in a Python environment and have tools for quick visual analysis. PyTorch is built on Python classes. It allows the combination with other Python libraries, like SciPy and Numpy. Further, it extends the usual object functionalities (e.g. in printing statements at runtime by showing the data transferred from one object to another). In contrast to this, TensorFlow is a more advanced Machine Learning library, predominantly used for Deep Learning applications. Keras methods facilitate the use of TensorFlow. However, TensorFlow requires some experience with respect to functionality. PyTorch is not as feature-rich as TensorFlow, but all the essential features (necessary for our task) are available.¹¹

4.2 Few-shot learning with GPT-2

GPT-2 model is a language model trained on a wide variety of sources, allowing the model to incorporate idioms not seen in the input text. In other words, even though GPT-2 is not specifically trained on German language it should be able to generate text in German. A limiting factor might be the maximum number of tokens that can be generated by GPT-2, which amounts currently 1024. This corresponds to approximately two paragraphs of German text. But since we consider license requests, that have relatively straight forward requests and answers, we assume that the limiting number of tokens is sufficient for our language model.

Few-Shot (FS) will be used in this work as recommended by Brown et al. (2020). The model is given a few demonstrations of the task at hand, in this case IT tickets, and then conditioned (at some inference time) without updating the internal model weights. An example consists of a question and an answer (as desired completion). After being granted K examples of question and answers, in this case for license requests, in combination with one final example of a question for that the few-shot learner model is expected to provide an answer (completion task). To provide a vivid example of the question and answer pairs, we refer to figure 7.¹² The set of (training) examples with Q&A parts will be called context in the following.

¹¹ For further information see the *PyTorch* documentation.

¹² Although, this figure refers to the evaluation of the *test* set, it gives a good example of the questions in the *training* set since the train-test-split was chosen randomly, i.e. the Q&A parts shown in figure 7, could have also been used for the training set.

The main advantages of few-shot learners are 1) a major reduction in the need for task-specific data and 2) a reduced potential to learn an overly narrow distribution from a large but narrow fine-tuning dataset. The main disadvantage is that few-shot models are generally outperformed by state-of-the-art fine-tuned models.

The scope of this work is delimited to perform a few-shot demonstration with the pre-processed data from an IT-Support, built upon a GPT-2 model. To enable this task additional pre-processing steps are needed. Further, a “json” file containing Q&A pairs is created to shape the model's response into the direction of the intended response. This file served as context for the few-shot learning model. The overall process is again summarised in figure 6.

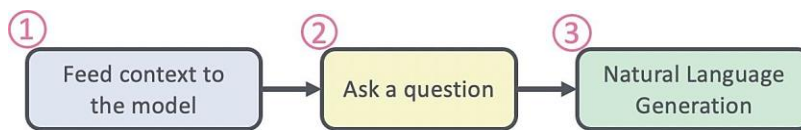


Figure 6: Few-shot learning mechanism with GPT-2
Source: authors' own illustration

4.2.1 Using the gpt-2-simple package

The Python package *gpt-2-simple* was initially developed by Max Woolf. It employs the *TensorFlow* library and combines fine-tuning and generating scripts for Open AI's text generation models.¹³ In order to use the *gpt-2-simple* package, the TensorFlow package (in many use cases) has to be downgraded to version 1.15. Furthermore, this version of TensorFlow is not supported anymore by Python 3.9 but the previous version of 3.7. The model only runs on the local system. The first release of GPT-2 with 117 M parameters, 12 layers of dimension 768 requires a noteworthy amount of disk space, i.e. 500 MB.

Further experimentation was performed by using checkpoints resulting into a medium model, with 345 M parameters and 24 layers of dimension 1,024, as well as a large model, with 774 M parameters and 36 layers of dimension 1,280. However, besides the increase in response time, accompanied by a non-negligible space consumption, the medium and large models did not result, in any significant improvement. In order to evaluate different models, we compared the generated answers with the labeled (i.e. true) answers provided by the IT support. For this, we used various text similarity (e.g. Cosine and Manhattan similarity).

¹³ For more details, see the *gpt-2-simple* package documentation.

4.2.2 Using the *german-gpt2* package

The *german-gpt-2* model is “pre-trained” with data provided by Bavarian State Library.¹⁴ The same corpora were used for the German BERT. In contrast to *gpt-2-simple*, this model uses the PyTorch implementation inside the transformer architecture of the Huggingface repository.

The model follows a similar process as *gpt-2-simple*. It uses few-shot learners as discussed in the previous section. An additional pre-processing step was added to allow the model evaluation, namely, a “demonstrations.txt” file, which uses the *json.loads()* method to parse a valid JSON string and converts it into a Python dictionary.

To run the model, a function was created, which takes the user question, the IT response, and the context, as parameters into account. The function calls the *text generation pipeline* (from Huggingface), which takes the task, the model, and the tokenizer (as input parameter and predicts the words that are most likely to follow. For our purpose, the task is set to *text-generation*, and the model as well as the tokenizer were set to *german-gpt2*.

4.3 Model Evaluation

Many NLG tasks seem to be open-ended. Academics, as well as industry practitioners, lead a constant debate for relevant evaluation metrics of generative models. However, the ultimate goal in NLG is to generate text, which is a) valuable and b) understandable for humans. For this precise reason, human judgments are a natural “first-choice” decision criterion for evaluating NLP models in many use cases (Celikyilmaz et al. 2020).

The evaluation process was performed by maintaining the same context. We manually insert a question for the model to answer. This allows us to compare answers of different models for one and the same question. It became evident that further pre-processing steps were necessary. Moreover, results showed that distance metric measures (such as the Euclidean distance) were impracticable for evaluating text similarities. More adequate text similarity measures were used instead such as the Levenshtein-ratio or the Cosine similarity. Finally, we decided to base our model validation on human judgment. For this, we compared the model responses for given questions to previous answers provided the human IT support. According to Hirschman and Gaizauskas (2001) model responses need to fulfill the following 6 criteria: Correctness, conciseness, completeness, coherence, relevance, and finally justification. It showed, that both *gpt-2 simple* and *german-gpt* models did not meet these expectations (see the appendix for an example of model predictions including the Levenshtein-ratio and the Cosine similarity). Further investigation made clear, that the lack of a question mark affected the

¹⁴ The MDZ Digital Library team from the Bavarian State Library (dbmdz for short) offers various open-source models within the Huggingface repository. For further information see the documentation.

model predictions. The repetition of one and the same phrase was another shortcoming that we identified especially for the *german-gpt2* model. The reason lies in the probabilistic model nature. Although *gpt2-simple* is not specifically pre-trained on German language, the model (partly) returns meaningful German phrases as the sample question in the appendix shows. However, the complex nature of Q&A tasks in general poses a major challenge for our model. This challenge is related to the inevitable need to formulate a question, as very open and unstructured text, which is still a difficult task to solve, even for modern NLP algorithms. To solve this problem, the engine needs additional knowledge about language structures, i.e. NLU capabilities as stated previous sections.

During the deployment of the previously discussed GPT-2 models, we were granted access right to the most recent GPT-3 API, called *Playground*. It provides a *general-purpose* “text-in-text-out” interface, which makes it applicable to more general language tasks. To use this API, the user can simply enter a text (prompt) in order to obtain completion. The present chosen task was “Q&A” and the parameters provided to the model were the same as for the *gpt2-simple* and the *german-gpt2* models. All other parameters of GPT-3 were left blank, thus indicating default values. For each model, *gpt2-simple*, *german-gpt2*, and *gpt3*, we followed the process depicted in figure 6 by using the same context for all Q&A training scenarios from license requests. The results, at least for GPT-3, were qualitatively stable, with respect to the above criteria posed by Hirschman and Gaizauskas (2001). An exemplary is shown in figure 7 (in German language). However, further experimentation and research is necessary in order to evaluate the capabilities and limitations of few-shot learner models for Q&A tasks in general. Model Limitations.

Although, we only worked on a small subsample for license requests, we are confident that our study generally shows the feasibility of a more general implementation including other IT relevant topics. We decided on the topic of “software licenses” since the underlying requests follow a clear structure, and moreover, in line with our anecdotal evidence from practice, already covers a large majority of daily IT requests. Being in the early stage of a feasibility analysis, this approach seemed to be the most plausible choice. If an easy structurable, task that poses a large majority of daily request, could not be covered by our model, we can have reasonable doubt to continue our task of implementing a Q&A system for IT help desks. We further believe that our large language model will improve with additional fine-tuning procedures. With enough training data, the model will also learn to build up more complicated, i.e. less structurable, dialog tasks. We seek to perform a structured hyperparameter-optimization (via grid-search), which we have refrained in our feasibility analysis.

GPT was chosen, as pre-trained NLP model. Besides its promising few-shot learning capabilities and actuality, the choice of a specific model poses a major challenge. This is due to inevitable links to BERT, which is an encoder-only transformer-based model for Q&A tasks.

Question:Hallo zusammen, hiermit beantrage ich eine IntelliJ Entwicklerlizenz für die Ultimate Edition.Viele Grüße Dennis

IT-Support Answer: Hallo Dennis, für die Beschaffung fallen Nettokosten in Höhe von insgesamt 499,00€ an. Bitte lass Dir die Kosten explizit von Deinem Vorgesetzten oder Deinem Projektleiter freigeben. In der Freigabe muss für uns ersichtlich sein, dass die Kosten dem Freigebenden bekannt sind. Das Ticket wird für 14 Tage zurückgestellt. Sofern die Freigabe innerhalb dieser 14 Tage bei uns eingeht, werden wir die Beschaffung durchführen. Wenn die Freigabe nach den 14 Tagen bei uns eingeht, wird für den kompletten Vorgang ein neues Ticket erstellt und wieder am Ende der Ticket-Queue einsortiert

gpt2-simple Answer: ich eine IntelliJ Entwicklerlizenz für die Ultimate Edition.BuyableInstoreAndOnline

german-gpt2 Answer: Hallo zusammen, ich möchte gerne nachfragen, ob wir Lizenzen für die Ultimate Edition haben? Ja, wir haben eine Lizenz für die Ultimate Edition. Einige Grüße Christian Flanders

GPT3 Answer: Hallo Dennis, für die Beschaffung fallen Nettokosten in Höhe von insgesamt 499,00€ an. Bitte lass Dir die Kosten explizit von Deinem Vorgesetzten oder Deinem Projektleiter freigeben. In der Freigabe muss für uns ersichtlich sein, dass die Kosten dem Freigebenden bekannt sind. as Ticket wird für 14 Tage zurückgestellt. Sofern die Freigabe innerhalb dieser 14 Tage bei uns eingeht, werden wir die Beschaffung durchführen. Wenn die Freigabe nach den 14 Tagen bei uns eingeht, wird für den kompletten Vorgang ein neues Ticket erstellt und wieder am Ende der Ticket-Queue einsortiert

Figure 7: Qualitative evaluation: Exemplary output of GPT-2-simple, German-gpt2 and GPT-3 for a request of an IntelliJ Developer license (in German language)

Moreover, the configuration of the virtual environment to perform the tests was not so simple. As mentioned before, many examples provided in the code documentation are made for using Google Colab notebooks that were inadequate for our tests (see section 4.1.2). Therefore, our models were stored locally. Lastly, the *gpt2-simple* model was finally selected for the sake of simplicity and model illustration purposes. With more time resources and budget, we would probably favour the most recently deployed GPT-3 standard. Fortunately, this decision can still be postponed until the deployment of the final Q&A ticketing system. In this case, even the adoption of the upcoming GPT-4 standard is even more desirable.

Additional problems arose from data quality. We found multiple Q&A pairs that were either not properly documented, or too complex and indirect. A further limitation is the quality of pre-processing steps. Each document needs to have a specific structure. The Q&A pair has to start with an identifying token. In addition to this, redundant information has to be removed. Both, the questions and answers, incorporated greetings, signatures, and other courtesies that are most likely to deteriorate our language model predictions.

Last but not least, the process of user feedback loops is key for our process and should be studied in more detail. The engagement of the corresponding IT support is also a crucial factor. The willingness to cooperate is characterized by a dichotomy between maintaining processes and increasing user experience.

4.4 Future research outlook

Micro-service architectures allow us to easily transfer currently developed NLP technologies to all other company domains. As outlined above, we propose to use conversational agents that fully rely on GPT technology. These agents can be transferred to all other business units that are exposed to constant communication exchange, e.g. “Human Resource Management” or “Administration Offices”. With enough “curated” dialog data even “Law and Compliance” solutions could benefit from NLP systems. Possible limitations might climax in generating strictly guided dialogues. However, a GPT-based solution represents a universal language model, without any need for intent classification. Consequently, beside the continuous improvement of NLP systems as a whole, future research should also concentrate on facilitating the implementation of such NLP systems in daily business operations.

5 Summary

In this study, we have designed an implementation guideline for setting up an NLP module to solve daily Q&A tasks for an internal IT help desk. Our language model, based on the recent GPT few-shot learning technologies such as GPT 2 and GPT 3. We have performed a feasibility study in cooperation with a large-scale IT consultancy company, at the example of software license requests. We have highlighted necessary technological objectives, i.e. the software packages to be used, as well as the data challenges to be overcome. Improvements result in a dynamic fine-tuning process, thereby increasing the training data support. The goal is to incorporate the module in the existing ticketing process, e.g. by using the in-store conversational agent solution. The module solves commonly structurable tasks. More complex IT tickets will be redistributed to human IT specialists as usual. However, the final deployment of our language model would result in a tremendous relief for the future workload of an IT help desk.

Appendix: Example question for evaluating GPT2 predictions for Q&A tasks in IT ticketing services including similarity scores

Question: Guten Morgen, ich brauche die Lizenz für die MS Tools und Microsoft Office. Vielen Dank!
Beste Grüße DB Von meinem Samsung Galaxy Smartphone gesendet die Ultimate Edition.Viele Grüße
Dennis

IT-Support Answer: Hallo Daniel, alle Office-Programme kannst Du unter portal.office.com auf bis zu
5 Geräten installieren. Benötigst Du darüber hinaus noch eine weitere Lizenz? Welche wäre das genau?

gpt2-simple Answer: Ich brauche die Lizenz für die MS Tools und Microsoft Office. Vielen Dank! Beste
Grüße DB Von meinem Samsung Galaxy Smartphone gesendet.

Levenshtein-ratio: 0,955 **Cosine similarity:** 0,368.

german-gpt2 Answer: Hallo, ich brauche die Lizenz für die MS Tools und Microsoft Office.

Vielen Dank!

Levenshtein-ratio: 0,759 **Cosine similarity:** 0,282.

References

- Adamopoulou, E.; Moussiades, L. (2020): Chatbots. History, technology, and applications. In: *Machine Learning with Applications*, 2. (2020).
- Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P. et al. (2020): Language Models are Few-Shot Learners. In: arXiv preprint arXiv:2005.14165 (2020).
- Budzianowski, P.; Wen, T. H.; Tseng, B. H.; Casanueva, I.; Ultes, S.; Ramadan, O. & Gašić, M. (2018): Multiwoz— A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In: arXiv preprint arXiv:1810.00278.
- Cheyner, A.; Didier Guzzoni, D. (2014): Method and apparatus for building an intelligent automated assistant. US Patent 8,677,377.
- Celikyilmaz, A.; Clark, E.; Gao, J. (2020): Evaluation of text generation. A survey. In: arXiv preprint arXiv:2006.14799 (2020).
- Fiore, D.; Thiel, C.; Baldauf, M. (2020): Potenziale von Chatbots für den innerbetrieblichen IT-Support. In: *HMD Praxis der Wirtschaftsinformatik*, 57. (2020), No. 1, pp. 77–88.
- Hirschman, L.; Gaizauskas, R. (2001): Natural language question answering: the view from here. In: *natural language engineering*, 7. (2001), No. 4, pp. 275–300.
- Jäntti, M.; Cater-Steel, A.; Shrestha, A. (2012): Towards an improved it service desk system and processes. A Case Study. In: *International Journal on Advances in Systems and Measurements*, 5. (2012), No. 3 & 4, pp. 203–215.
- Kucherbaev, P.; Bozzon, A.; Houben, G.-J. (2018): Human-aided bots. In: *IEEE Internet Computing*, 22. (2018), No. 6, pp. 36–43.
- Langkilde I.; Knight, K. (1998): Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pp. 704–710.
- Mathur, S.; Lopez, D. (2019): A scaled-down neural conversational model for chatbots. In: *Concurrency and Computation: Practice and Experience*, 31. (2019), No. 10.
- Peng, B.; Zhu, C.; Li, C.; Li, X.; Li, J.; Zeng, M. et al. (2020): Few-shot natural language generation for task-oriented dialog. In: arXiv preprint arXiv:2002.12328 (2020).
- Prabhumoye, S.; Black, A. W.; Salakhutdinov, R. (2020): Exploring Controllable Text Generation Techniques. In: arXiv preprint arXiv:2005.01822.
- Radford, A.; Wu, J.; Amodei, D.; Amodei, D.; Clark, J.; Brundage, M. et al. (2019): Better language models and their implications (https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf). Accessed on 10/17/22.
- Solaiman, I.; Dennison, C. (2021): Process for Adapting Language Models to Society (PALMS) with Values-Targeted Datasets. In: arXiv preprint arXiv:2106.10328 (2021).
- Tarlaci, F. (2019): GPT-2 for Question Answering (<https://fatmatarlaci.wordpress.com/2019/05/08/gpt2qa/>). Accessed on 05/25/21.

- Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N. et al. (2017): Attention is all you need. In: arXiv preprint arXiv:1706.03762 (2017).
- Vijayarani, S.; Ilamathi, M. J.; Nithya, M. et al. (2015): Preprocessing techniques for text mining-an overview. In: International Journal of Computer Science & Communication Networks, 5. (2015), No. 1, pp. 7–16.
- Wen, T. H.; Gasic, M.; Mrksic, N.; Su, P. H.; Vandyke, D. & Young, S. (2015): Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In: arXiv preprint arXiv:1508.01745. Wen, T. H.; Gasic, M.; Mrksic, N.; Rojas-Barahona, L. M.; Su, P. H.; Vandyke, D. & Young, S. (2016): Multi-domain neural network language generation for spoken dialogue systems. In: arXiv preprint arXiv:1603.01232.
- Wolff, E. (2018): Microservices. Grundlagen flexibler Softwarearchitekturen. 2.; aktual. Aufl.; Heidelberg: dpunkt.verlag.
- Zhu, C.; Zeng, M.; Huang, X. (2019): Multi-task Learning for Natural Language Generation in Task-Oriented Dialogue. In: Padó, S. (ed.): The 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing. Proceedings of system demonstrations. Conference on Empirical Methods in Natural Language Processing And the International Conference on Natural Language Processing (EMNLP-IJCNLP 2019), Hong Kong, China, November 3–7. Association for Computational Linguistics (ed.). Stroudsburg, PA : Association for Computational Linguistics (ACL), pp. 1261–1266. (<https://aclanthology.org/D19-1123.pdf>). Accessed on 04/05/23.

Authors

Prof. Dr. Marcus **Becker** is a full-time professor of “Quantitative Methods of Business Information Technology” at the International School of Management (ISM) in Dortmund. He gained international experience in automated Risk Management processes working for HSBC and Deloitte. Besides, he was part of the German Robo Advisory panel at Deloitte’s Risk Advisory service line testing Machine Learning algorithms like LSTM on stock market forecasts.



M. Sc. Erika **Prokop Dayrell de Lima**, is a former graduate student of ISM in “Business Intelligence and Data Science” (BIDS). She gained professional experience in data mining, machine learning, software engineering, and IT project management as a consultant working for Adesso SE. As a Requirements and Issue Manager at Kostal Group she is currently responsible for ensuring the implementation of requirements and resolution of issues throughout the development lifecycle.

