A Natural Language Interface for an Energy System Model

Jonas Hülsmann[*], Lennart J. Sieben[†], Mohsen Mesgar[‡], Florian Steinke[*]
Technische Universität Darmstadt
Darmstadt, Germany
[*]Energy Information Networks and Systems
Email: {jonas.huelsmann, florian.steinke}@eins.tu-darmstadt.de
[†]Email: lennart.sieben@stud.tu-darmstadt.de
[‡]Ubiquitous Knowledge Processing
Email: mesgar@ukp.informatik.tu-darmstadt.de

***Abstract*—Energy system models are widely used for operation and expansion planning, scaling from single houses up to supranational energy grids. They can provide essential input for decision makers. These are, however, often non-technical persons and thus unfamiliar with mathematical modeling, which makes them reliant on others to explain the model results to them. In order to make energy system models more directly accessible, we introduce a chatbot that enables also non-expert users to interact with an energy system model through natural language. Built with state-of-the-art natural language processing tools, it allows to manipulate the model inputs and can interactively answer questions about the results, both in free-form text and via structured plots. We present example interactions for a model of the German energy transition.**

***Index Terms*—energy models, natural language processing, machine learning, accessibility, chatbot**

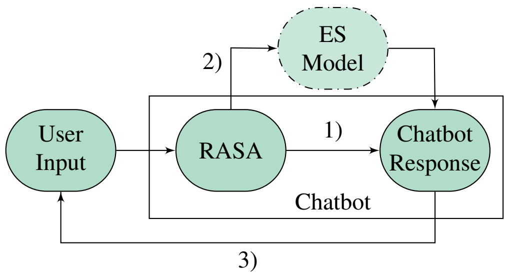


Fig. 1. Possible conversation paths of the proposed energy system chatbot. A user's input can lead to 1) immediate response, 2) interaction with the energy system model is needed and the answer is dependent on the model output, or 3) the chatbot responds with a request for clarification, in case the user's input is not distinct enough.

I. Introduction

Energy system models are applied from intra-day action planning for single power units, e.g. home photovoltaic systems with storage, up to long-term development planning for international energy clusters. Various model frameworks are available, for instance, TIMES [1] or OSeMOSYS [2]. Besides the technical results like operational schedules or quantitative expansion plans, these models can also answer questions of public interest, such as *"What is the cost of energy transition?"* or *"What is the ecological and economical benefit of a wind turbine?"*.

In technical terms, many energy system models are large linear programs that take a host of technical data as inputs and return their optimal decisions in the form of large data tables. Depending on the model's scope, the number of variables of an energy system model can be in the millions. This makes extracting the information of interest or identifying the right parameter to change for a sensitivity analysis a challenging task. For non-experts such a task might even be impossible.

However, those non-experts are often the ones that are in the end responsible and accountable for the decisions that are taken based on the model outcomes. Such a decision could be whether or not to build a certain power plant, or whether or not to push for a regulation that allows wind turbines in the close vicinity of settlements. Model end-users may thus be politicians, company managers, or even regular citizens. Without explanations from third party professionals, such decision makers could neglect or misinterpret the model results and in turn take bad decisions.

In this work, we try to make energy system models more accessible to non-experts, in line with [3]. We describe a chatbot interface enabling the users to directly interact with the energy system model via natural language. Without having to know the technical details of modelling, non-expert users as well as domain experts can quickly access information of a model's output or change model settings and initiate new model evaluations. The model can interactively request additional information from the user, and it presents model results in the form of numerical answers or structured plots. Moreover, the chatbot can answer many general questions about the involved energy entities.

Previous work [4] has shown that a chatbot interface to a smart home can help save up to 10 % of energy consumption by proposing suitable actions. However, no chatbot interacting with energy system models has been described so far.

The remainder of this paper is structured as follows: Section II briefly reviews the employed energy system model. The framework and data behind the chatbot interface is explained

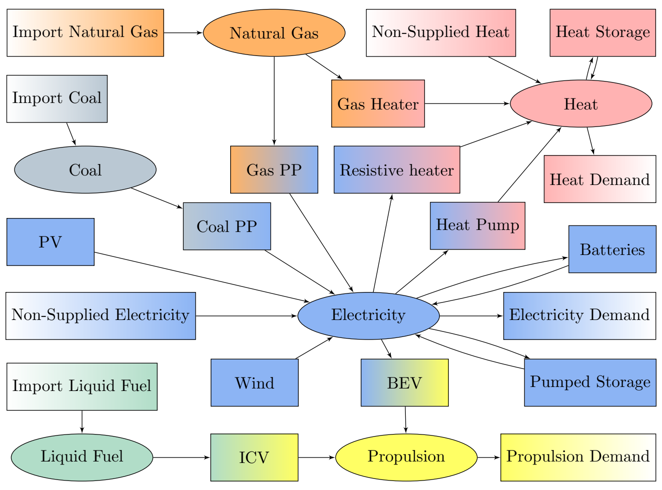


Fig. 2. Schematic overview of the employed energy system model of Germany. Different energy forms are displayed as ellipses, conversion processes as rectangles. In- and output arrows of a conversion process show which energy forms are transformed into the other. Conversion processes without input or output represent fixed energy demands or energy import into the model domain.

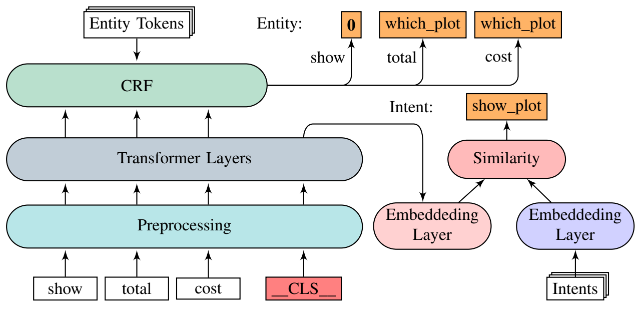


Fig. 3. The RASA DIET classifier for the instruction *"show total cost"*. Each word is represented as a one-hot vector and the sentence as the sum of those in the __CLS__ vector. All input vectors are processed by a neural network and two transformer layers. In the conditional random field (CRF), each processed word vector is assigned to an entity. The CLS vector is further processed by an additional embedding layer and compared to an embedding of all possible intents.

in section III, its integration into a complete application in section IV. A video with example interactions is linked in section V followed by a conclusion in section VI.

II. Employed Energy System Model

The employed model allows exploring transformation paths for the German multi-modal energy system in the time range from 2020 to 2050.

Mathematically, it is a linear program as described in [5], [6]. Each technology is represented as a conversion process, converting different energy forms into each other. For each conversion process, a variety of parameters like costs, capacities, specific emissions or time-related restrictions can be defined. The model minimises the total cost of the energy system by choosing both the cost-optimal operation and expansion plan. Additionally, a CO_2 emission limit can be defined. The model is solved with a 5-year resolution, where each year is represented by four weeks from all seasons in three-hourly timesteps. This sparse simulation was chosen to have a short solving time, enabling the chatbot to give interactive responses. The simulation settings can be changed in the conversation if a more detailed simulation is desired.

Figure 2 shows an overview of the model entities. A total of 6 energy forms and 12 conversion processes are part of the model, plus additional 8 conversion processes for resource import and demand.

III. Chatbot Methodology

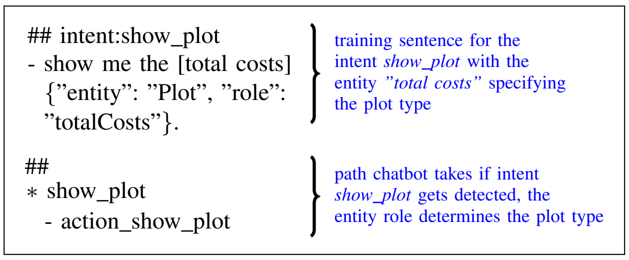
For the development of chatbots, multiple machine learning frameworks exist. Example frameworks are LUIS, Watson, Dialogflow, wit.ai – to only name a few [7]. For this chatbot, we have adapted an instance of the RASA framework [8], an open-source and state of the art framework with respect to intent and entity recognition [9]. An intent is a classification category for the user input that describes the type of reply the chatbot will give. An entity exists within an intent and can further specify the input's meaning. For example, if the user asks to see a plot of the total cost, *show_plot* is the general intent and *"total cost"* is the entity – see Section III-B.

In Figure 1, the basic structure of the user interactions with the chatbot are displayed. Some intent classifications by RASA do not require interaction with the energy system model, e.g. small talk or questions that are not directly related to the model. For these inputs, the chatbot can give an immediate response. If the user input is classified to require model interaction, the connected energy system model is used. The chatbot is also capable of sending follow-up questions or remarks, e.g. if the user's text input is faulty, or a clarification of certain specifics about the input is required.

A. *Training data and different question types*

For RASA's intent classification mechanic, we specified training data to be able to engage the user in multi-turn conversations. At maximum three past messages are needed for the longest multi-turn conversation that our chatbot can handle. However, five past messages are selected in order to have some leeway in case non-related user inputs occur during a conversation. If the limits are too large, the chatbot could take reference to an input that the user has not in mind anymore. Moreover, it is important that not too many messages are taken into account so that the real-time calculations are not delayed significantly and a better user experience is guaranteed. The policy that incorporates these past messages is called *TED Policy* in RASA [10].

To enable the chatbot to answer different types of questions (e.g. FAQ, small talk or model questions) RASA offers a technique called *retrieval action*. Classifiers using this technique are able to detect – besides their regular set of classes – an input belonging to a special type of category, which is then processed further with a different classifier. The main DIET classifier detects the intents requiring energy system model interactions. Additionally, it can detect small talk and the implemented FAQs. Inputs classified to be within those



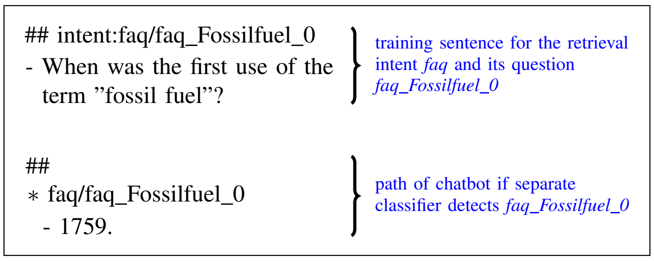
```
## intent:show_plot
- show me the [total costs]
  {"entity": "Plot", "role":
  "totalCosts"}.

##
* show_plot
  - action_show_plot
```

training sentence for the intent *show_plot* with the entity *"total costs"* specifying the plot type

path chatbot takes if intent *show_plot* gets detected, the entity role determines the plot type

(a) Single training sentence for a model request. The prefix *action_* in the reply of the chatbot jumps to a custom implemented function, which in this case, shows a plot of the current models total cost.



```
## intent:faq/faq_Fossilfuel_0
- When was the first use of the
  term "fossil fuel"?

##
* faq/faq_Fossilfuel_0
  - 1759.
```

training sentence for the retrieval intent *faq* and its question *faq_Fossilfuel_0*

path of chatbot if separate classifier detects *faq_Fossilfuel_0*

(b) Single training sentence for the FAQ with retrieval actions. The DIET classifier categorises the data into the general intent category *faq* and then with another dedicated classifier the data gets classified into the specific question *faq_Fossilfuel_0*.

Fig. 4. Example training data for a request about the model in Figure 4(a) and data for a FAQ pair in Figure 4(b).

two categories will be forwarded to seconds-stage classifiers to determine the correct answers. For each of the resulting three classifiers, multiple training examples and the corresponding responses had to be defined. The used training data sets are:

- For the small talk capability, a dataset by Microsoft was integrated. This dataset is called *Personality Chat* and includes 90 different intents for training [11].
- The FAQ training data was created by data scraping from Wikipedia and extracting question and answer pairs. The extraction was done with the tool *Question Generation* [12]. The articles about *fossil fuels* and about *renewable energy in Germany* were used for question generation. 87 question-answer pairs were created this way.
- The training data for the model questions was specified manually, because it is specific to the functionalities of the underlying energy system model. As these types of questions take direct reference to the energy model, the answers are dynamic and dependent on the current energy model. Here, 28 different intents were created.

Figure 4 shows the format of the data for a model question and a FAQ pair. Additionally to the training data, a reply of the chatbot has to be defined. Notably, the prefix *action_* in Figure 4(a) lets RASA jump to custom implemented code and not a static answer, which results in inputs about the model being answered dynamically.

Additionally, we modified the FAQ training data to get more training examples. The modifications were done by randomly removing characters or words in the questions. This can prevent the overfitting of a neuronal classifier for the FAQ.

B. Intent & Entity Classification Methodology

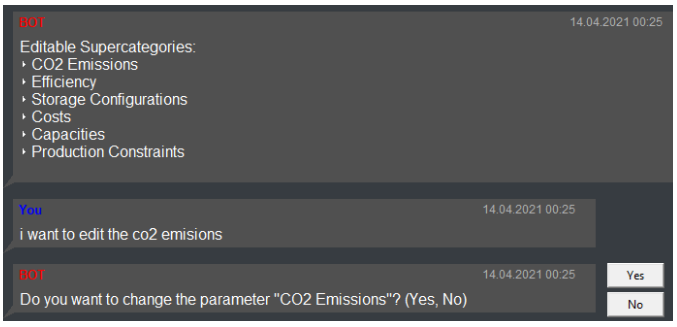
The main classifier used is RASA's Dual Intent and Entity Transformer (DIET) architecture [13]. Schematically this architecture is shown in Figure 3. The example input in Figure 3 is *"show total cost"*. This phrase needs to be classified to its intent *show_plot*. Additional to this intent classification, an intent can have entities included. The entity in this example is what type of plot should be displayed. In this case, the entity that gets classified is *"total"* and *"cost"*.

In the following, the processing steps that the DIET classifier traverses are shortly described. Each part of the user input, which is separated by a whitespace, counts as a token and goes through two steps of preprocessing. First, each token is transformed into a one-hot encoded vector with the dimension of all unique words in the training data. The encoding for the complete input text is done by summing up all the one-hot encoded vectors – this path is marked with *__CLS__* in Figure 3. Second, all encoded vectors are fed step-by-step through a single-layer neural network, bringing the one-hot encoded token vectors to a fixed dimension of 256. Neural networks are used here to compress encoded vectors to latent representations with smaller dimensions.

The preprocessed vectors then go through two transformer layers, which assign more semantic meaning between the words of the input, whereby patterns and word relations to each other can be learned [14]. The vectors, which were single word tokens initially, are further used for entity recognition. The vector for the whole sentence is used for intent classification. For the intent classification, all defined intent labels also get one-hot encoded. In order to calculate a similarity, the transformer output for *__CLS__* and the encoded intent labels are embedded into a single semantic vector space with a dimension of 20. The similarity is subsequently calculated with a loss function giving the resulting intent loss [13]. A loss function is used, which maximises the similarity L_{intent} of the output vectors of the embedding layers in Figure 3. With the outputs of the embedding layers being defined as h_{CLS} and h_{intent} for all specified intents, a similarity S is maximised by choosing the best fitting intent. The matched intent is defined as $S^+ = h^T_{\text{CLS}} h^+_{\text{intent}}$ and unmatched intents as $S^- = h^T_{\text{CLS}} h^-_{\text{intent}}$. The intent loss function is given by

$$L_{\text{intent}} = -\langle S^+ - \log(e^{S^+} + \sum_{\Omega^-} e^{S^-})\rangle, \tag{1}$$

with Ω^- being the set of the negative sample and $\langle . \rangle$ the average over all examples for the current intent during training [13]. Entity classification works similarly. If a token got assigned to an entity in the training data, the entity label for the token gets one-hot encoded and a loss is calculated in the conditional random field (CRF) block with the token outputs from the transformer layers. This entity classification in the


BOT 14.04.2021 00:25

Editable Supercategories:
- CO2 Emissions
- Efficiency
- Storage Configurations
- Costs
- Capacities
- Production Constraints

You 14.04.2021 00:25

i want to edit the co2 emisions

BOT 14.04.2021 00:25

Do you want to change the parameter "CO2 Emissions"? (Yes, No)

Fig. 5. Further model parameters can be changed with a keyword matching algorithm. It works by comparing each word of the user input to words in a displayed list and an additional synonym dictionary in the background. As a similarity metric the Jaro-Winkler distance is used [16]. The categories with the highest probability are presented to the user and they then can be selected for editing. Even spelling mistakes lead to a correct recognition.

CRF block respects surrounding entity labels of a token, which RASA uses to further capture the relations of neighbouring words [13], [15]. The entity loss from the CRF block is then added to the intent loss and the classifier optimises this summed up loss. In the practical prediction case, DIET chooses the labels that produce the smallest loss, as shown for the example in Figure 3. Out-Of-Vocabulary words are handled by ignoring them. DIET is trained by minimising the sum of the entity and intent losses.

In the case that an input is a FAQ or small talk, the DIET classifier only detects whether the input belongs to either of those categories. The text is then processed further in two respective classifiers, which are quite similar to the intent classification part of the DIET architecture, except no transformers are used. Instead of being compared to intent labels, inputs are compared to all question-answer pairs within their training data category.

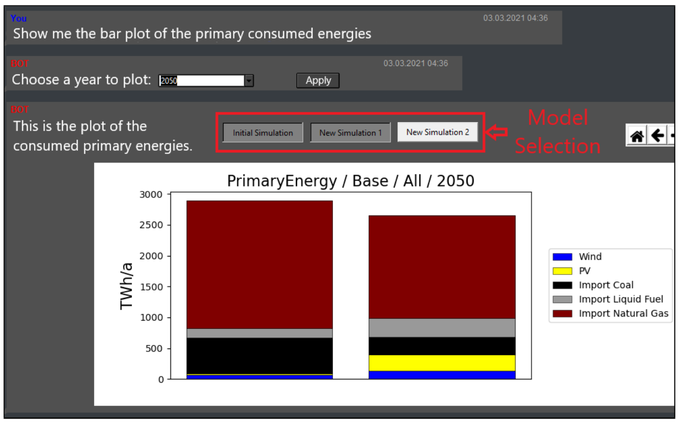
IV. Application Integration

The complete application integrates the raw natural language processing (NLP) interface, see Figure 6, with a second way of specifying scenarios and simulation parameters via a structured interface, see Figure 5. The idea is that a non-expert user will start with the NLP interface only and can, once he has become familiar with the terms, get more control via the structured interface. This helps to give more control over the models.

The structured interface based on partial word matching, not RASA, is described in Section IV-A. The additional environment for comparing several different simulation runs is described in section IV-B, security feature in section IV-C.

A. Request and change model parameters

Requesting and changing parameters of the energy system model is possible in two different ways. The first way is to do it during a conversation with the chatbot. An example input by the user to request a model parameter could be *"How much CO_2 does the system emit?"*. An example input to change a model parameter would be *"Change the CO_2 prices of the model."*. This would lead to a follow-up question by the chatbot for clarification, asking for which year of the simulation which price should be set.


You 03.03.2021 04:36

Show me the bar plot of the primary consumed energies

BOT 03.03.2021 04:36

Choose a year to plot: 2050 Apply

BOT

This is the plot of the consumed primary energies.

Fig. 6. Bar plot of the primary consumed energies in two different energy system models for the simulated year 2050. The model from the right bar can produce up to four times more energy through photovoltaic power (PV) and two times more through wind power, compared to the base model on the left. In addition, the costs for storing energy were lowered and the costs to import coal and gas were increased. This plot was invoked by the user input *"Show me the bar plot of the primary consumed energies"*. The matching of input and reply is done by RASA here.

The second way to change parameters in a more in-depth manner is by using a *keyword matching algorithm* in a separate *model editing mode* of the chatbot. This algorithm works by using categorical options for the type of parameters the user wants to change, e.g.: *CO_2 values*, *costs*, *energy outputs*. The user has the option to select a category by typing in a sentence. This input is compared to each of the categories. The comparison furthermore takes a synonym dictionary into account, so that the word CO_2 would also match with e.g. *carbon-dioxide*. After the categorical selection, the desired parameter to change is determined in the same manner. The similarity is computed by the Jaro-Winkler distance, which is a string metric and gives back a similarity score between 0 and 1, where 1 means that two compared words are identical [16]. For two words in the *keyword matching algorithm* to count as matching, the Jaro-Winkler distance has to be larger than a threshold, 0.8 worked well in our tests.

The chatbot is able to answer more complex inputs that require multiple interactions with the model. Consider the input *"What would change if the [conversion process] produces [number] times more energy per year?"* The *"[conversion process]"* specifies a template for a trained entity inside the energy model, e.g. a *wind power plant*, and *"[number]"* specifies a positive rational number that acts as a factor for the current energy output per year. To answer this question, the energy system model result has to be read for the respective value. A potential request for clarification might follow up if the user input was not specific enough. If all required information is clear, the new value can be set. A request follows if the chatbot should rerun the simulation or wait for further changes.

B. Comparison of modified models

To make the impact of model changes more accessible to users, a visual comparison of results from altered model versions is implemented. The comparison can be done for 17 different plots, including plots for energy production, technology costs or CO_2 emissions. Two different models can be altered using the chatbot, while the base model remains untouched for comparison. In Figure 6 three speech bubbles with a model comparison during a conversation with the chatbot are displayed. They were invoked by the user's input *"Show me the bar plot of the primary consumed energies."* – the plot shows the baseline on the left and an altered model on the right. The model can produce up to four times more energy through photovoltaic power and two times more through wind power. Furthermore, the costs for storing energy were lowered and the costs to import coal as well as gas were increased. Menu buttons above the bar plot inside the chat bubble allow to toggle individual models on and off and allow for further plot interactions like zooming or exporting.

C. Safety layers

All inputs that are not classified as FAQ or small talk are encapsulated in multiple conditional blocks. With these conditional statements, custom errors are caught, and a message gets displayed explaining why the error occurred. An error could be e.g. that a negative energy output was specified, an entity was not correctly recognised, or that a simulation was infeasible. These extra levels of error catching are essential, because the classifiers do not take every boundary condition into account.

V. Experiments

The complete application and exemplary interactions are demonstrated in the video at https://youtu.be/wK38TDXymF8. Exemplary screenshots are given in Figure 5 and Figure 6.

VI. Conclusion and Outlook

With this chatbot, an accessible natural language interface was created, which brings a complex engineering tool to the hands of non-experts, enhancing their decision-making process. The user can easily change model parameters, request results, compare different versions of a model visually, and has an additional layer of safety for incorrect inputs. For user guidance, a catalogue was composed that provides exemplary user inputs to obtain a certain chatbot reply. The chatbot was optimized with regard to the inputs in this catalogue.

The main hyperparameters that were optimised are the output dimensions of the embedding layers for the intent classifications plus response selections and the trained epochs. RASA's optimisation tool *nlu-hyperopt* was utilised for this task [17]. Due to the lack of aggregated user data at this point in time, the data is relatively scarce, so the chatbot does not perform well if the input is very different from the catalogue. Furthermore, the chatbot's interactions with the energy system model are not model agnostic, thus a different energy system model would require the interactions to be reprogrammed within the chatbot.

Further work to improve the chatbot could include:

- Expanding the training data set for the FAQ collection and improving it to enable bidirectional questions.
- Collecting additional data for the energy system model interactions through user testing. This could potentially be done trough an online version of the chatbot to reach a bigger audience.

Acknowledgment

This work was funded by German Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung BMBF) in the Project *PlexPlain – Explainable AI for complex linear programs on the example of intelligent energy systems*, funding number 01IS19081.

References

[1] R. Loulou, U. Remme, A. Kanudia, A. Lehtila, and G. Goldstein, "Documentation for the TIMES Model Part II," 2005.

[2] M. Howells, H. Rogner, N. Strachan, C. Heaps, H. Huntington, S. Kypreos, A. Hughes, S. Silveira, J. DeCarolis, M. Bazillian, *et al.*, "OSeMOSYS: the open source energy modeling system: an introduction to its ethos, structure and development," *Energy Policy*, vol. 39, no. 10, pp. 5850–5870, 2011.

[3] J. Hüelsmann and F. Steinke, "Explaining Complex Energy Systems: A Challenge." Poster presented at: "Tackling Climate Change with Machine Learning" *NeurIPS*, December 11, 2020.

[4] U. Gnewuch, S. Morana, C. Heckmann, and A. Maedche, "Designing conversational agents for energy feedback," in *International Conference on Design Science Research in Information Systems and Technology*, pp. 18–33, Springer, 2018.

[5] K. Schaber, F. Steinke, and T. Hamacher, "Transmission grid extensions for the integration of variable renewable energies in europe: Who benefits where?," *Energy Policy*, vol. 43, pp. 123–135, 2012.

[6] K. Schaber, F. Steinke, and T. Hamacher, "Managing temporary oversupply from renewables efficiently: Electricity storage versus energy sector coupling in germany," in *International Energy Workshop, Paris*, 2013.

[7] M. Canonico and L. De Russis, "A comparison and critique of natural language understanding tools," *Cloud Computing*, vol. 2018, p. 120, 2018.

[8] T. Bocklisch, J. Faulkner, N. Pawlowski, and A. Nichol, "Rasa: Open source language understanding and dialogue management," *arXiv: 1712.05181*, 2017.

[9] X. Liu, A. Eshghi, P. Swietojanski, and V. Rieser, "Benchmarking natural language understanding services for building conversational agents," *arXiv: 1903.05566*, 2019.

[10] V. Vlasov, J. E. M. Mosig, and A. Nichol, "Dialogue transformers," *arXiv: 1910.00486*, 2019.

[11] V. S. Kannan, "Personality chat." https://github.com/microsoft/BotBuilder-PersonalityChat, 2018. [Online; accessed 25-March-2020].

[12] S. Patil, "Question generation using hugging face transformers." https://github.com/ patil-suraj/question_generation, 2020. [Online; accessed 25-March-2020].

[13] T. Bunk, D. Varshneya, V. Vlasov, and A. Nichol, "Diet: Lightweight language understanding for dialogue systems," *arXiv: 2004.09936*, 2020.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, p. 5998–6008, 2017.

[15] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, (San Francisco, CA, USA), p. 282–289, Morgan Kaufmann Publishers Inc., 2001.

[16] J. M. Keil, "Efficient bounded jaro-winkler similarity based search," in *BTW 2019*, pp. 205–214, Gesellschaft für Informatik, Bonn, 2019.

[17] T. Wochinger, A. Nichol, and M. Loubser, "Hyperparameter search for rasa nlu." https://github.com/RasaHQ/nlu-hyperopt, 2020.