OpenNLU: Open-source Web-interface NLU Toolkit for Development of Conversational Agent

Yi Fan Ong, Maulik Madhavi* and Ken Chan†
* Department of Electrical and Computer Engineering, National University of Singapore, Singapore
E-mail: yifan.ong@u.nus.edu, maulik.madhavi@nus.edu.sg
† ST Engineering Land Systems Ltd, Singapore
E-mail: chan.kokhoe.ken@stengg.com

***Abstract*—The Natural Language Understanding (NLU) module in a conversational agent interprets and understands the user's query. As the application scenario evolves, there is a need to periodically update the knowledge database that supports the agent by adapting or re-training the NLU model. Such periodic updates could be time-consuming to the developers and system administrator since it requires manual efforts. In this paper, we present the OpenNLU toolkit, a user-friendly interface for building and updating the knowledge database, and evaluating NLU module. This paper describes in detail the architecture, important features and design of the web-based tool, as well as the backend features which are supported by popular Rasa NLU toolkit, and deep learning libraries such as Tensorflow, and PyTorch. This paper also demonstrates the training and evaluation processes on in-house datasets alongside other benchmarking datasets (ATIS and Snips) to exemplify the usage of OpenNLU toolkit as to validate proof of concepts. The open-source OpenNLU toolkit is available to the research community [1].**

Index Terms: Administrative interface, BERT language model, Natural Language Understanding, Slot tagging, Intent Classification

I. Introduction

Conversational agents (CA) have been deployed for many real-world applications on smartphones and smart devices. The use of natural language in CA offers an easy, flexible and intuitive way in human-machine communication. However, the CA may not be much helpful always because the language understanding and interpretation capability of the assistants is limited as application scenarios evolve.

Natural Language Understanding (NLU) is considered an AI-hard problem [1]. There are inherent complexities in natural language such as anaphora, elision, ambiguity and uncertainty [2]. By providing explicit domain-specific knowledge, conversational assistants can improve understanding of user utterance through contextual inference. General conversational assistants like Alexa, Siri and Google Assistant rarely support domain-specific functionalities because they are not designed to make these inferences [3].

After the initial deployment of CA, the bulk of the workload required lies in the maintenance of the conversational agents to support a new service. Since domain information is evolved, the deployed CA will require to maintain the knowledge source. The maintenance operations include a collection of new additional data, retraining of the NLU model component and evaluates performance with additional data before deployment. The retraining process is necessary to accommodate conversation changes and future growth in the understanding of the dialogue. This iterative training cycle can be tedious and unmanageable if the usability of the data maintenance and training environment is not intuitive. In particular, it is important to have a user-friendly graphical training environment for developers who are not technical experts of NLU. This can help development process with ease, smooth out the workflow of data maintenance operation. A solution would be to simplify the interaction between developer and NLU component of the dialogue system through the use of a user-friendly graphical interface.

Although there are graphical interfaces available for the development in the market, such as Googles Dialogflow, IBMs Watson, Microsofts LUIS, Amazon Lex and Rasa X [4]–[12]. They are built on third-party proprietary Natural Language Processing (NLP) services, they are not easily adopted and scaled for third party applications for open source development.

We consider that it is important that developer of CA should have an access to NLU services as to manage the data and evaluate their performance. That motivates us to build a graphical web interface tool to improve the updating of knowledge database and to support efficient deployment. The tool can accommodate the custom features and provide an interface with backend NLU services.

In this paper, we aim to tackle the aforementioned problem and offer better user convenience to fulfil these requirements by introducing our developer graphical interface, namely, OpenNLU. The tool name OpenNLU stands for Open-sourced NLU, which indicates the service as an open-source graphical NLU interface for developers.

II. Natural Language Understanding

NLU is a research problem that deals with analysis and understanding user input to extract meaningful key information [13]–[15]. In particular, NLU is responsible to extract the useful information such as intent and entities by performing

[1] https://github.com/oyfml/opennlu

intent classification (IC) and entity extraction or slot tagging (ST) tasks, respectively. In the classical approaches, IC is performed using discriminative classifier such as Boosting or SVM [14], [16]–[18] and ST is performed using the sequential classifier such as Hidden Markov Model (HMM) Viterbi decoding or Conditional Random Field (CRF) [15], [19]–[21]. Mathematically, for word sequence $W := \{w_1, w_2, \cdots, w_T\}$, the ST can be referred to as estimation of most probable slot sequence and intent category such that

$$\hat{S} = \underset{i}{argmax} P(S|W) \quad (1)$$

$$\hat{I} = \underset{j}{argmax} P(I|W) \quad (2)$$

where $S := \{s_1, s_2, \cdots, s_T\}$ and I are slot sequence and intent class respectively.

The study presented in [17] explains the capability of Deep Belief Network to classify the intent from count vector representation. Later to incorporate the temporal information, Recurrent Neural Network (RNN) based models were explored to take in word sequence as input and outputs the probable slot sequence [15]. Further, the sequence to sequence frameworks of supervised learning was also used as multi-task learning to identify slots, intent as well as domain [22], [23]. To further exploit the alignment information from sequence to sequence model, an attention-based RNN model was proposed in the study [24]. The study presented in [25] proposes the bidirectional interrelation model between slots and intent classification. Recently, some studies were done on Bidirectional Encoder Representations from Transformers (BERT) to leverage contextual language information captured from a large amount of unlabeled data to jointly perform ST and IC [26].

A. Benchmark of NLU Frameworks

There are several platform services available publicly for creating NLU data and training the NLU of chatbot or dialogue system. These services have different complexity levels and integration capabilities. The several service platforms are Rasa [27], IBM Watson, Microsoft LUIS and Google Dialogflow. The research study was conducted to investigate their NLU benchmark capabilities for all of these services [28]. Rasa is the only open-sourced service platform in the list that is open for research and development. Rasa was observed to achieve similar high performing results as LUIS as per the study in [28]. This demonstrates that Rasa remains competitive with other third-party commercial service platforms and because of its code-transparency, demonstrates the suitability of customising the platform service for practical usage.

B. Scope of the work

As such, OpenNLU will be incorporating Rasa as the primary backend NLU framework for model development. Besides Rasa, backend tools are also supported using popular deep learning frameworks such as Tensorflow and PyTorch. In this work, we consider NLU to classify the intent and predict the slots. There are scopes of NLU such as domain classification, named-entity recognition, etc. We first describe the OpenNLU framework and its designed components separately with their features. Next, we demonstrate the feasibility of OpenNLU on standard databases with their use in framework and results.

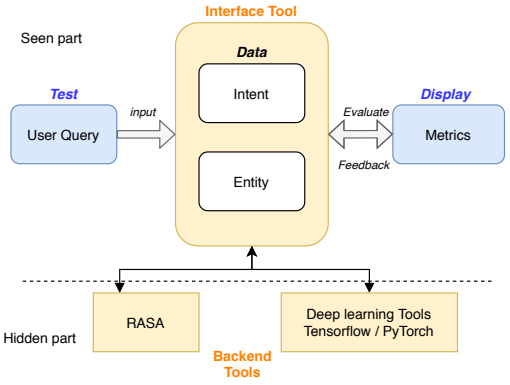

Fig. 1. Block diagram and components of OpenNLU toolkit.

III. OpenNLU Toolkit

A. Design Principles

The major consideration behind the development of OpenNLU is to ensure a high ease of usability. The design and implementation of the web interface aim to abide by the following principles or characteristics:

- Learnability: intuitive, easy and quick to learn
- Efficiency: efficient to use; simplicity in interaction
- Memorability: easy to remember; inclusion of prompts and help screens
- Errors: low error rate; triggers to suggest user action
- Satisfaction: closure feedback to the user after completion of a task

Figure 1 shows the block diagram and components of OpenNLU toolkit. The input text query from the user is given to the interface (frontend) part.

B. Web Interface Tools

The graphical interface is implemented as a web-based application. The main objective is to allow the interface to be accessed without considering cross-platform compatibility, thereby operating with only a browser and network connection. However, it is recommended to run the interface tool in the local machine as the average computing device might not be equipped with sufficient computing power for simultaneous training operations.

The interface tool incorporates a couple of frameworks to support its web application functionality. The main framework is Flask, which is a lightweight web micro-framework that is designed for flexible and quick web development. For the visually appealing template and designs, the CSS framework Bootstrap is used, alongside with other supporting technologies such as Jinja for HTML template engine and JavaScript

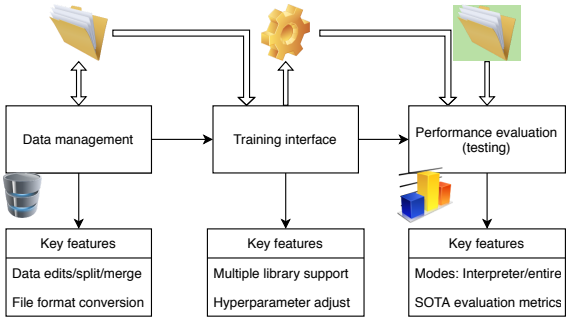

Fig. 2. Overall workflow of OpenNLU and key features.

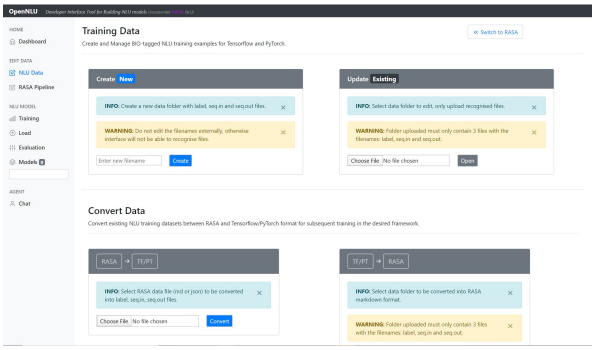

Fig. 3. Screenshot of data management features of OpenNLU

for client-side scripting. The front-end technology is important for the sleek and alluring design and also client-side scripting and HTML template engine allows the interface to perform essential form checks and error triggers without having to interrupt and call backend processes, which might be busy with threading a requested NLU computation.

C. Backend Tools

As shown in Figure 1, the backend of OpenNLU is supported by Rasa open-source library [27] and two popular deep learning libraries, namely Tensorflow and PyTorch [29], [30]. Their popularity in building advanced neural network models provides an excellent fundamental base to incorporate future compatible NLU models. Their library and support are extensive and have been a standard in many research studies. There is also large communal support, which can allow for stimulation in implementation for future OpenNLU features.

D. Features of OpenNLU

The overall workflow and key features of OpenNLU are shown in Figure 2. The features of the interface tool are to prioritise for ease of interaction between the developer and the NLU model. The design layout of the interactive web controls remains intuitive to the user to improve convenience in the development process of NLU models. The features are separated into their respective categories which can be seen in the menu navigation bar on the left side of the interface as shos in Figure 3. Next, we will describe each feature in detail.

1) Data Management: For unrestricted control over data manipulation, OpenNLU has implemented a variety of data processing features for the respective NLU frameworks. First, we have data format conversions. As the data format for Rasa (i.e,*markdown* and *json*) and Tensorflow/PyTorch (TF/PT) (i.e., BIO) are different, OpenNLU offers data format conversion by entering only data format. Another feature of the data management module is to merge smaller data or split the large data into subsets, namely, train, test, and dev or merge two datasets. This makes openNLU different than third party proprietary services such as Dialogflow, Amazon Lex, etc. Please refer to Figure 3.

2) Training interface: As discussed earlier, OpenNLU support Rasa and deep learning frameworks under the hood for training IC and ST. The training process of Rasa is dictated by the configuration pipeline, which is customisable to specify intrinsic components, such as featurizers, intent classifiers or entity extractors [31]. The backend algorithm for the Tensorflow and PyTorch is the implementation of the BERT model for Joint Intent Classification and Slot Filling [26]. The developers can able to modify the hyperparameters of the Joint BERT model and tune the model w.r.t. updated data which is not possible for third party proprietary services such as Dialogflow, Amazon Lex, etc. There is also the scope of replacement where the Joint BERT model can be replaced with newly researched or explored NLU model as the tool is backed up with TF/PT.

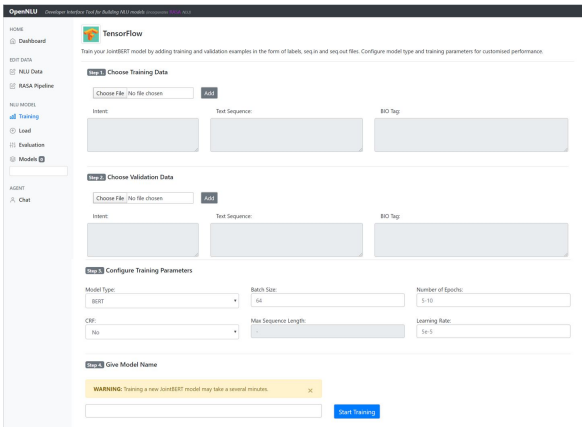

Fig. 4. Screenshot of model training feature of OpenNLU.

3) Performance Evaluation: The performance evaluation features are available once the training performed or loading the model. Upon training or loading a model, the evaluation page will be made available for user access. The current toggled model will be an active model. Users can do so by selecting the model name in the model directory at the side navigation bar. The active model will appear highlighted in blue and its interpreter will be selected for evaluation. For performance evaluation, there consists of two functions, one is

TABLE I
DIFFERENT RASA PIPELINES AND ITS COMPONENTS

Components	spaCy pipeline	supervised pipeline	ConveRT pipeline
Tokenizer	Spacy	Whitespace	ConveRT
Featurizer	Spacy	CountVectors	ConveRT
IC	Sklearn	Embedding	Embedding
ST	CRF	CRF	-

the evaluation of a single message using the interpreter, while the other is evaluating an entire test dataset. Other third-party proprietary services such as Dialogflow or Amazon Lex do not have to evaluate the performance on entire data over their developer interface, whereas OpenNLU offers this feature. Please refer to Figure 5 for evaluation feature.

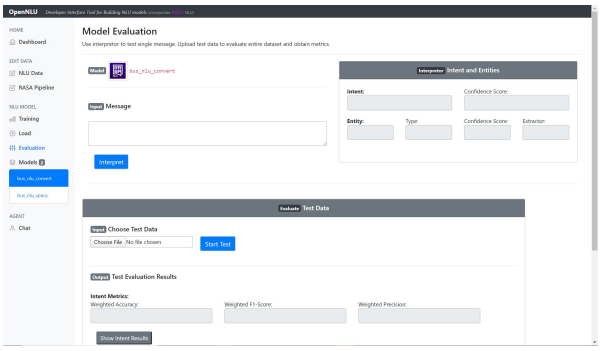

Fig. 5. Screenshot of evaluation feature of OpenNLU.

IV. RESULTS AND DISCUSSIONS

This section presents the NLU performance, in particular IC and ST, on standard databases such as ATIS and Snips. We also present the result on our inhouse collected data for chatbot of the autonomous transportation system. We have considered the three kinds of pipelines or NLU systems using Rasa, namely, spaCy, supervised and ConveRT [31], [32]. Basically, they are using pre-trained word embedding, supervised word embedding, and ConveRT transformer models, respectively. Table I shows different components used in these pipelines. More details are available here [31]. For deep learning framework, we have used Joint IC and ST using the BERT model, with the exact same hyper-parameters [26]. The neural network architecture for the Joint BERT model is shown in Figure 6.

A. Database

In this work, we use our data for chatbot used for autonomous bus transportation [33]. This inhouse dataset contains information about a passenger inquiry that we will refer to as 'Autonomous Bus' (AB) database in this study. The slot values are representing bus stops, stations, or location names. We also used widely used benchmarks, namely ATIS and Snips dataset for the comparison. The domain of ATIS data is related to air flight reservations [34]. Snips data is related to queries uttered to a mobile device suggesting a request or action [35].

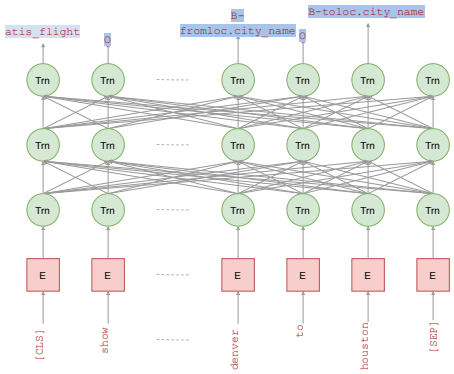

Fig. 6. Joint BERT architecture for IC and ST. After [26]

TABLE II
STATISTICS OF DIFFERENT DATASETS FOR THE EXPERIMENTS

Database	Total examples train/validation/test	Intents	Slots
AB	1057/166/296	10	9
ATIS	4478/500/893	20	100
Snips	13084/700/700	10	72

The statistic of different datasets used in this work such as number of examples, intents and slots are shown in Table II.

B. Results

Experimental results are performed on the F1 measure, which conveys the balance between precision and recall value. Table III shows IC and ST performance on two benchmarking datasets, namely ATIS and Snips. The Joint BERT model in Tensorflow and PyTorch outperformed the Rasa models by a slight margin. This is because of the embedded contextual information in the BERT language model that has helped slightly over the convention word embedding approach used in Rasa. In larger datasets, i.e., Snips and ATIS, the benefit of using a large language model can be seen as vocabulary outside of training data are handled well. As seen from the ATIS results, Rasa models tend to suffer more due to the lack of domain-specific language. ATIS involves the same domain throughout the entire dataset, where intents have entities that overlap between classes. Their entities are generally country or state names which are also pre-trained in BERT. Therefore, the Joint BERT model has a significant advantage over this aspect. On the other hand, for Snips, there are inherent terminologies that are specific to each intent. These terminologies can be recurrent in the test dataset where the model can correctly differentiate the intent through recognising the terminology as an entity. This is where the Rasa model are seen to perform better as compared to their performance in ATIS. From the results we observed, Rasa models do not fall too far behind the state-of-the-art Joint BERT model and remain competitive for most parts. Therefore, the choice of which model to select for training will be up for the decision of the user since their performance is very similar. In general, the language understanding capabilities of the technologies implemented

TABLE III
EXPERIMENTAL RESULTS WITH OPENNLU ON DIFFERENT DATASET

Frameworks	Intent Classification			Slot tagging		
	AB	ATIS	Snips	AB	ATIS	Snips
ConveRT	0.9898	0.9452	0.9744	-	-	-
spaCy	0.983	0.8816	0.9547	0.7673	0.9018	0.9176
supervised	0.9864	0.9268	0.9685	0.9172	0.9005	0.9184
Tensorflow	0.9763	0.9735	0.9843	0.9734	0.9746	0.9706
Pytorch	0.9657	0.9735	0.9857	0.894	0.9527	0.9584

in OpenNLU have been successful enough to yield positive results, with most models attaining over 90% F1 scores on test set evaluation.

V. CONCLUSIONS

In this paper, we propose and release an open-source toolkit, namely, OpenNLU to address the issue of data maintenance operation by the developer as this graphical web interface enhances the experience and confidence before deployment. The use of the simple and intuitive design in the interface can benefit new users without development expertise to adopt the technology without complications. Also, the customisability, support for complex models and streamline user interactions can allow experienced developers to operate this maintenance cycle quickly. We discussed the key features by OpenNLU in detail that make OpenNLU different from third party services. Lastly, we have also used existing and our inhouse NLU data to validate the proof of concept and show the capability of OpenNLU.

VI. ACKNOWLEDGEMENT

This research is supported by the National Research Foundation, Singapore under its LTA Urban Mobility Grand Challenge Program, Project Code UM01/002, ST Kinetics Autonomous Bus Trial.

REFERENCES

[1] R. V. Yampolskiy, "Turing test as a defining feature of ai-completeness," in *Artificial Intelligence, Evolutionary Computing and Metaheuristics - In the Footsteps of Alan Turing*, ser. Studies in Computational Intelligence. Springer, 2013, vol. 427, pp. 3–17.

[2] A. Ram, R. Prasad, C. Khatri *et al.*, "Conversational AI: the science behind the alexa prize," *CoRR*, vol. abs/1801.03604, 2018.

[3] S. Mennicken, R. Brillman, J. Thom, and H. Cramer, "Challenges and methods in design of domain-specific voice assistants," in *AAAI Spring Symposia*, 2018.

[4] "Dialogflow," https://dialogflow.com/, online; last accessed 23 October 2020.

[5] "Watson Assistant," https://www.ibm.com/cloud/watson-assistant/, online; last accessed 23 October 2020.

[6] "Language Understanding (LUIS)," https://eu.luis.ai/, online; last accessed 23 October 2020.

[7] "Amazon Lex: Conversational AI for Chatbots," https://aws.amazon.com/lex/, online; last accessed 23 October 2020.

[8] "Introduction to Rasa X," https://rasa.com/docs/rasa-x/, online; last accessed 23 October 2020.

[9] N. Sabharwal and A. Agrawal, *Introduction to Google Dialogflow*. Berkeley, CA: Apress, 2020, pp. 13–54.

[10] S. Perez, "Amazon lex, the technology behind alexa opens up to developers," *Tech Crunch*, 2017.

[11] J. D. Williams, E. Kamal, M. Ashour, H. Amr, J. Miller, and G. Zweig, "Fast and easy language understanding for dialog systems with microsoft language understanding intelligent service (LUIS)," in *SIGDIAL Conference*. The Association for Computer Linguistics, 2015, pp. 159–161.

[12] A. Singh, K. Ramasubramanian, and S. Shivam, *Introduction to Microsoft Bot, RASA, and Google Dialogflow*. Berkeley, CA: Apress, 2019, pp. 281–302.

[13] C. Su, R. Gupta, S. Ananthakrishnan, and S. Matsoukas, "A re-ranker scheme for integrating large scale NLU models," in *IEEE Spoken Language Technology Workshop, SLT*, 2018, pp. 670–676.

[14] G. Tur and R. De Mori, *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons, 2011.

[15] G. Mesnil, Y. N. Dauphin, K. Yao *et al.*, "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 530–539, 2015.

[16] Y. Wang and A. Acero, "Discriminative models for spoken language understanding," in *International Conference on Spoken Language Processing (ICSLP)*, 2006.

[17] R. Sarikaya, G. E. Hinton, and A. Deoras, "Application of deep belief networks for natural language understanding," *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 22, no. 4, pp. 778–784, 2014.

[18] R. E. Schapire and Y. Singer, "Boostexter: A boosting-based system for text categorization," *Mach. Learn.*, vol. 39, no. 2/3, pp. 135–168, 2000.

[19] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *International Conference on Machine Learning ICML*. Morgan Kaufmann, 2001, pp. 282–289.

[20] M. Korpusik and J. R. Glass, "Spoken language understanding for a nutrition dialogue system," *IEEE ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 7, pp. 1450–1461, 2017.

[21] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding," in *INTERSPEECH*, 2007, pp. 1605–1608.

[22] D. Hakkani-Tür, G. Tür, A. Çelikyilmaz, Y. Chen, J. Gao, L. Deng, and Y. Wang, "Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM," in *INTERSPEECH*, 2016, pp. 715–719.

[23] D. Guo, G. Tür, W. Yih, and G. Zweig, "Joint semantic utterance classification and slot filling with recursive neural networks," in *IEEE Spoken Language Technology Workshop, SLT*, 2014, pp. 554–559.

[24] B. Liu and I. Lane, "Attention-based recurrent neural network models for joint intent detection and slot filling," in *INTERSPEECH*, 2016, pp. 685–689.

[25] H. E, P. Niu, Z. Chen, and M. Song, "A novel bi-directional interrelated model for joint intent detection and slot filling," in *ACL, Volume 1: Long Papers*, 2019, pp. 5467–5471.

[26] Q. Chen, Z. Zhuo, and W. Wang, "BERT for joint intent classification and slot filling," *CoRR*, vol. abs/1902.10909, 2019.

[27] T. Bocklisch, J. Faulkner, N. Pawlowski, and A. Nichol, "Rasa: Open source language understanding and dialogue management," *CoRR*, vol. abs/1712.05181, 2017.

[28] X. Liu, A. Eshghi, P. Swietojanski, and V. Rieser, "Benchmarking natural language understanding services for building conversational agents," in *International Workshop on Spoken Dialogue Systems Technology (IWSDS)*. Springer, April 2019, pp. 1–13.

[29] M. Abadi, P. Barham, J. Chen *et al.*, "TensorFlow: A system for large-scale machine learning," in *USENIX Symposium on Operating Systems Design and Implementation, OSDI*, 2016, pp. 265–283.

[30] A. Paszke, S. Gross, F. Massa *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *NeurIPS*, 2019, pp. 8024–8035.

[31] "RASA docs," https://rasa.com/docs/, online; last accessed 23 October 2020.

[32] M. Henderson, I. Casanueva, N. Mrksic, P. Su, T. Wen, and I. Vulic, "ConveRT: Efficient and accurate conversational representations from transformers," *CoRR*, vol. abs/1911.03688, 2019.

[33] M. Madhavi, T. Zhan, H. Li, and M. Yuan, "First leap towards development of dialogue system for autonomous bus," in *International Workshop on Spoken Dialogue Systems Technology (IWSDS)*. Springer, April 2019, pp. 1–6.

[34] C. T. Hemphill, J. J. Godfrey, and G. R. Doddington, "The ATIS spoken language systems pilot corpus," in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley*. Morgan Kaufmann, 1990.

[35] A. Coucke, A. Saade, A. Ball *et al.*, "Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces," *CoRR*, vol. abs/1805.10190, 2018.