FPI: Failure Point Isolation in Large-scale Conversational Assistants

Rinat Khaziev
Amazon Alexa AI
rinatk@amazon.com

Usman Shahid
University of Illinois Chicago
hshahi6@uic.edu

Tobias Röding
Amazon Alexa AI
rodingtr@amazon.com

Rakesh Chada
Amazon Alexa AI
rakchada@amazon.com

Emir Kapanci
Amazon Alexa AI
emirk@amazon.com

Pradeep Natarajan
Amazon Alexa AI
natarap@amazon.com

Abstract

Large-scale conversational assistants such as Cortana, Alexa, Google Assistant and Siri process requests through a series of modules for wake word detection, speech recognition, language understanding and response generation. An error in one of these modules can cascade through the system. Given the large traffic volumes in these assistants, it is infeasible to manually analyze the data, identify requests with processing errors and isolate the source of error. We present a machine learning system to address this challenge. First, we embed the incoming request and context, such as system response and subsequent turns, using pre-trained transformer models. Then, we combine these embeddings with encodings of additional metadata features (such as confidence scores from different modules in the online system) using a "mixing-encoder" to output the failure point predictions. Our system obtains 92.2% of human performance on this task while scaling to analyze the entire traffic in 8 different languages of a large-scale conversational assistant. We present detailed ablation studies analyzing the impact of different modeling choices.

1 Introduction

Conversational assistants have become increasingly prevalent in every-day life. With them, users can control appliances at home, get current weather information, or get help with recipes in the kitchen through simple voice commands. A typical dialog system processes user requests in multiple stages (see Figure 1). First, a voice trigger (or wake word) (Sigtia et al., 2018) model determines whether the user is speaking to the assistant. Following the trigger component, an Automatic Speech Recognition (ASR) (He et al., 2019) module converts user audio stream into a set of discrete text tokens. This text is sent to the Natural Language Understanding (NLU) component, which analyzes what the user request means. The domain classifier (DC) categorizes the user's request into a set of pre-defined topics, the intent classifier (IC) assigns an intent which represents what the user is trying to accomplish, and the entity recognition and resolution component (ERR) recognizes and resolves known entities in the users request. The system generates the best possible response (Result stage) using several subsystems that are specific to each dialog assistant (e.g., dialog management, re-ranking, etc). Finally the response is rendered into a human-like speech using a Text-to-Speech (TTS) system.

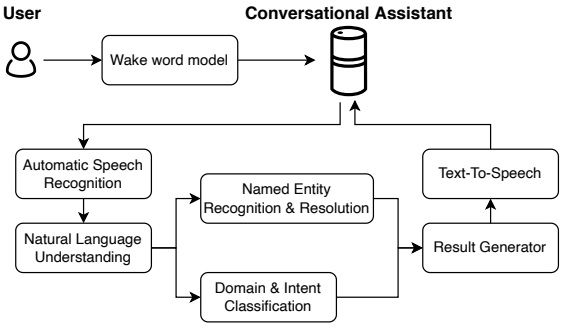

Figure 1: Component-level architecture of a typical conversational assistant.

When such a system makes an error, the complexity of the processing pipeline makes it extremely challenging to isolate the source of a defect. An error in an upstream component (e.g. ASR) can propagate through the system to the final response to the user. In such cases it is likely that multiple components starting from the first source of the error (referred to as "root" or "failure point" hereby) produce erroneous outputs. However, it is critical to identify this error to improve the overall system. Given the large traffic volumes, manual analysis to identify root causes for processing errors is infeasible.

In this work, we develop a machine learning model that predicts which component of a conver-

Proceedings of NAACL-HLT 2022: Industry Track Papers, pages 141 - 148

sational assistant caused the system to fail when processing user requests, a Failure Point Isolation (FPI) model. Our system helps to monitor the performance of the system holistically and improve the components of the dialog assistant that result in defective user interactions. The FPI model takes multiple inputs including the request text, system response, and subsequent turns which together help in capturing implicit feedback from the customer. We leverage recent progress in pre-trained Transformer-based language models to encode this information. We then combine these with encodings of metadata features such as confidence scores from different components in the online system using additional Transformer-based "mixing" layers to output the source of error or mark a request as correctly processed. Our model is trained on a small number of examples annotated with the source of error and then applied on all traffic for failure point isolation.

We present extensive experimental results to characterize the performance of our model and the impact of different modeling choices. Using only encoding of the request text, we achieve an F_1 score of 24.2% for FPI on our test sets. This improves to 40.3% by leveraging the full dialog context and system response. We see a further improvement to 51.4% by including additional metadata features. We also present ablation studies to characterize the impact of different text encoders and architecture choices for the mixing layer that further improve F_1 score to 53.3%. We show that this corresponds to 92.2% of human performance on the FPI task using a "golden" test set created by combining annotations from multiple highly-trained annotators.

2 Related Work

Several works have attempted evaluating dialog systems using deterministic or machine learning-based methods. The majority can be classified into the following groups: word-overlap metrics, user sentiment based approach, or component-specific error attribution.

Word-overlap metrics models like BLUE (Papineni et al., 2002) and ROUGE (Lin, 2004) are not well-suited for evaluating real-world conversational assistants. Liu et al. (2016) has demonstrated that the word-overlap metrics do not correlate wit=[p human judgement. As conversational assistants can also perform real-world functions (e.g., turning on lights), evaluation of such systems based on the textual response alone does not fully capture the set of actions taken by the system. Finally, the deterministic metrics are not fine-grained enough to identify which component of the system was responsible for the defective interaction. Hence, word-overlap metrics have a limited ability to provide prescriptive feedback to developers.

Schmitt et al. (2012) proposed evaluating dialog systems based on the user perception and Interaction Quality (IQ). Here each dialog is assigned a numerical score as evaluated by an annotator. Schmitt and Ultes (2015); Bodigutla et al. (2020); Gupta et al. (2021) developed models that used features derived from the logs of the dialog system to build predictive IQ models. Gupta et al. (2021) demonstrated that transformer-based architectures without log-derived features can outperform previously-developed models. Lowe et al. (2017) proposed a similar to IQ metric, ADEM, and trained a predictive mode. Sinha et al. (2020) developed a transformer-based model, MAUDE. This model is trained using contrastive learning and produces scores that correlates with human judgment. The sentiment or quality-based metrics allow for monitoring real-life dialog systems, however they do not provide actionable insight into the performance of the system components.

Chada et al. (2021) and Sethi et al. (2021) have built systems that attribute errors in the NLU component of a conversational assistant. Chada et al. (2021) focused on building transformer-based models that detect NLU intent and domain classifications errors. Sethi et al. (2021) detect NLU domain and intent errors in the dialog system using confidence scores produced by the NLU models. When attributing NLU errors, they focus on root-causing issues in the training data (e.g., low-resource intent, mislabeling, etc). Though this feedback is actionable, neither of these works attempt to root-cause errors in other components of dialog systems.

These aforementioned approaches have limitations when it comes to failure point isolation in large-scale conversational assistants. Instead of focusing on a small portion of a dialog system, we create an automated error attribution system that can provide insights into the root causes of defective interactions at scale for all of the components of a conversational assistant. Unlike approaches that estimate user satisfaction and dialog quality from the user's perspective, our focus is on understanding whether the system delivered the response

that it was designed to deliver and if not, why. In case when the system was designed to perform the action but failed to do so, our model provides clear feedback that can help to improve system performance in the future.

3 Methodology

In this section, we first describe our training and test datasets, and discuss the challenges in constructing them (§ 3.1). Next, we describe the creation of our "golden" test dataset (§ 3.2). Then, we describe the features that we use in our model (§ 3.3). Finally, we present details of the network architecture and model training used to output FPI based on these input features (§ 3.4).

3.1 Training Dataset

To train the FPI model, we created a dataset containing real-world data by extracting a mix of random and targeted samples from a commercial large-scale conversational assistant. Our dataset contains approximately 11.5MM de-identified user requests in 8 different languages. The training dataset was split into train, validation, and dev using a 75/5/20 scheme such that user sessions do not overlap in any split. All requests were manually annotated using internal tools as correct or incorrect. Incorrect requests were further labeled with one of five error types, corresponding to one of the stages of a conversational assistants processing pipeline (see Figure 1). These include:

1. **False Wake (FW)** errors that capture incorrect trigger system predictions
2. **ASR** errors that capture the incorrect transcription of the user speech
3. **NLU** errors that contain domain classification (DC) and intent classification errors (IC)
4. **ERR** errors that capture entity recognition and resolution errors
5. **Result** errors made by the response generation component when the system took an incorrect action even though all previous steps succeeded

When there are several potential errors in a dialog, we only mark one of them as the root cause of the system failure: the first failing stage in the processing pipeline, ordered from Wake Word to Result stages. Figure 2 shows some example turns of what different errors can look like in the processing pipeline. In the first turn, the ASR error would be marked as a fatal error and the root cause of the defective system response. In the second turn, the ASR error would be marked as non-fatal as subsequent components are able to recover from the error and produce a correct system response. In the last turn the system performed as designed, however it could not fulfill user request.

3.2 "Golden" Test Dataset

Given the vast data volumes, failure point isolation in a complex dialog system is a challenging task even for humans. For example, ERR error analysis requires inspecting entity data (such as music catalogs). Further, the definition of the failure point can be ambiguous without subsequently rerunning and correcting each component of a dialog system. As a result, the error attribution labels can have poor quality and consistency across different annotators.

To create a suitable test set for evaluating the accuracy of our model, we leveraged a more sophisticated "golden" annotation workflow. This more labor- and time-intensive workflow does not rely on a single annotator but on a combination of multiple annotators, and an ensemble of machine learning models to make the labeling decision. First, each request gets labeled by the annotators and the ensemble model in parallel. Whenever there is a disagreement on the labels, the request is evaluated by a highly trained annotator who makes the final decision. We annotated approximately 58k sessions through this workflow to create a "golden" test set with higher annotation quality than our training set. The "golden" dataset is not used for training out model, however it is used to report F_1 score of the models we train in this work and compare model performance to humans.

3.3 Feature Engineering

We train our FPI model on a multi-turn dataset, which includes user request, system response (TTS), and the interaction metadata. The interaction metadata is parsed from the logs of the online production system and includes the outputs of the machine learning models executed at each stage of the data processing pipeline and identifiers of the systems that made the final prediction, in the case of multiple models competing for response generation.

We limit user dialog to previous, current, and next user interactions. Using the context of the user interaction, we are aiming to capture implicit customer feedback that a production system might

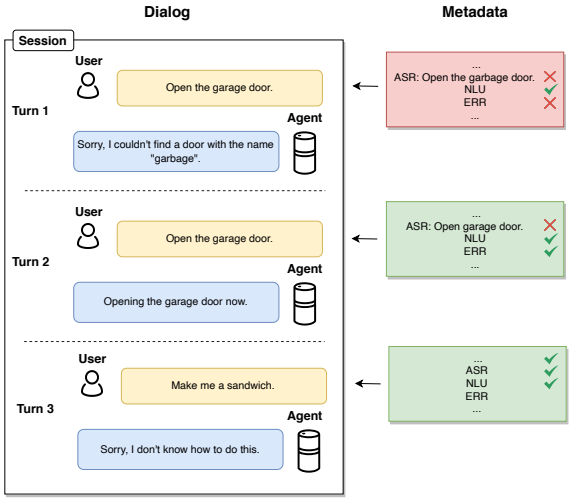

Figure 2: Construction of FPI model features using previous, current and next turn to get features from the whole dialog.

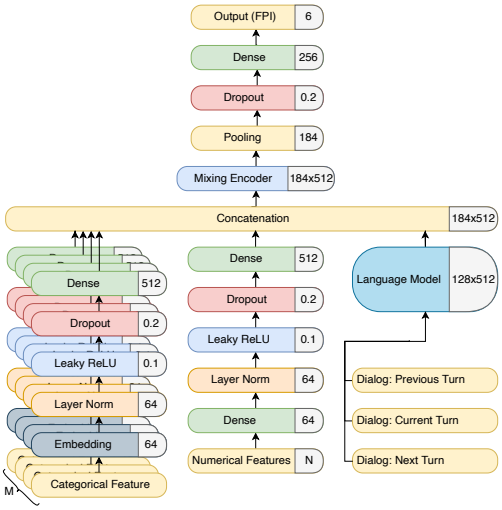

Figure 3: The architecture of the Failure Point Isolation (FPI) model with multi-modal feature embedding networks. M and N are counts of categorical and numerical features respectively.

be lacking. Our feature set includes multiple *categorical features* (e.g., NLU intent predictions), *numerical features* (e.g., confidence scores logged by run-time models, or time difference between user turns), and *text data*, in the form of user request text and system response (TTS) collected from a user session.

Due to a large number of features available in the logs and the complexity of our end-to-end system, we group the categorical and numerical features into 5 major groups for ablation studies: Wake Word (WW) features, ASR features, NLU features, Result features. WW, ASR, and NLU features include the confidence scores produced by the component models. Result features include the details of which sub-system produced response and whether requested action could be fulfilled by the system. We tokenize the text data using the *sentencepiece* tokenizer before inputting them to Transformer-based encoders that we describe in the next section.

3.4 Model Architecture and Training

Executing our model on already processed user sessions gives us two advantages. First, we gather a holistic view into the execution of all asynchronous components by constructing model features from the system logs. Second, there are no latency limitations, which means we can leverage large transformer-based models (Vaswani et al., 2017).

The four main components of our model are: categorical feature embedding networks, a numerical embedding network, a transformer-based language model, and mixing layers built on top of the embedding networks to produce the final predictions. We process all of the numerical features jointly using a single embedding network (see Figure 3). Each of the categorical features are embedded separately. The numerical and categorical features are concatenated with the language model embeddings and are passed to "mixing" layers. Textual features, request text and system response, are processed jointly by multi-lingual transformer-based models (Wolf et al., 2020; Paszke et al., 2019). In order to constrain model latency we use mT5 (Xue et al., 2021) and XLM-R (Conneau et al., 2020) models in their smallest configuration with 170M and 270M parameters. The mixing layer consists of the encoder layers and a final feedfoward block that produces model predictions.

The FPI model is trained using a multi-stage procedure. First, we fine-tune the language models alone on FPI labels without metadata features (Stage 1). This step is necessary for domain-specific adaptation of the models pre-trained on generic datasets. Second, we fine-tune the metadata encoder jointly with textual features on the FPI labels using our training dataset, but do not update the language model during this stage (Stage 2 warm-up). Finally, we fine-tune the whole model on the FPI dataset (Stage 2 fine-tuning). The details of our training setup and computational budget are reported in Appendix A.

4 Results

In this section, we report the results of the experiments with the FPI model. First, we investigate the importance of system response and extended context size (§ 4.1). Second, we illustrate importance of the features derived from logs (§ 4.2). We demonstrate the effect of language model size in (§ 4.3). We perform experiments with fine-tuning language models on the task-specific dataset (§ 4.4) and compare performance of the best performing model to a standard annotator (§ 4.5). The models reported in subsections § 4.1-4.3 were trained using only stage 2 of the training procedure (§ 3.4) with mean-pooling layers unless specified otherwise. F_1 scores are reported from a single training run on the "golden" dataset described in § 3.2.

4.1 Importance of using system response and extended context

context	TTS	F_1 scores						
		FW	ASR	ERR	NLU	Result	Correct	Avg
current	✗	7.8	21.6	1.1	3.1	7.2	87.8	21.4
current	✓	5.8	31.2	10.9	18.9	41.3	90.5	33.1
extended	✓	16.1	40.3	15.5	29.4	48.4	92.1	40.3

Table 1: F_1 scores of models trained with different context (current turn vs extended context) with request text and TTS, as indicated by TTS column. "current" indicates that the model was trained with the current turn, "extended" indicates that the model was trained with previous, current, and next turns.

Table 1 summarizes F_1 results of the experiments that quantify the effect of adding TTS and dialog context on model performance. Thus, adding TTS to the user request improves macro average F_1 score from 21.4% to 33.1%. Further on, we find that extending dialog context to previous and next turn improves F_1 score by another 7.2% absolute to 40.3% . As indicated by consistent gains in Result, ASR, and NLU classes, this set of experiments confirms our hypothesis: extended context captures implicit feedback (e.g., rephrasing) from the customer.

4.2 Importance of features derived from the logs of system components

Based on our experiments (Table 2), adding features derived from the logs of the dialog assistant's online components improves ability to detect errors in those components. For example, NLU, ASR, and WW F_1 scores gain 4.3% , 10.1%, and 17.2% absolute when respective feature sets are added to the FPI model. Additionally, adding NLU features leads to improving ASR and ERR scores, and adding ASR features yields improvements in the WW class.

component features	F_1 scores						
	FW	ASR	ERR	NLU	Result	Correct	Avg
-	16.1	40.3	15.5	29.4	48.4	92.1	40.3
Result	16.7	42.5	22.9	28.9	48.7	91.7	41.9
NLU	14.4	45.2	22.7	33.7	50.1	92.9	43.2
ASR	17.8	49.4	23.8	30.8	46.4	93.1	43.5
WW	33.3	40.6	18.8	30.3	49.9	92.5	44.2
all	**39.7**	**53.8**	**27.5**	**39.9**	**53.4**	**94.0**	**51.4**

Table 2: F_1 scores of the models trained with full dialog (including TTS) on different sets of features (see § 3.3).

The model trained with the full feature set (bottom row of the Table 2) demonstrates the best performance in this experiment set with a macro-average F_1 score of 51.4%. It benefits from the implicit feedback provided by the dialog text and features derived from logs of all of the system components.

4.3 Performance with a larger language models

component features	F_1 scores						
	FW	ASR	ERR	NLU	Result	Correct	Avg
-	25.9	46.3	20.7	34.6	53.6	93.0	45.7
all	29.7	53.7	27.3	39.7	54.3	93.8	49.8

Table 3: Results of the feature ablation studies with XLM-R model with 270M parameters.

Table 3 presents F_1 scores of the FPI network trained with the XLM-R language model (§ 3.4). Based on the results of our experiments, using bigger models improves F_1 scores of the model trained using dialog as the only features (first row in Table 3) from 40.3% macro-average for a model trained using mT5 to 45.7% for a model trained with the XLM-R model. The advantage of using a larger language models disappears when we leverage a full feature set. Thus, the XLM-R-based FPI model demonstrates 49.8% macro-average F_1 score, which is comparable to the mT5-based model trained with the same feature set (§ 4.2).

4.4 Effect of fine-tuning language models on task-specific data

Pooling layer	F_1 scores						
	FW	ASR	ERR	NLU	Result	Correct	Avg
mean	38.8	52.6	27.2	38.5	52.7	93.8	50.1
max	**40.9**	**55.1**	**30.4**	**42.5**	**57.8**	**94.1**	**53.5**

Table 4: F_1 scores of the FPI models trained with language models fine-tuned on the task-specific data.

The results of training model with stage 1 (language model fine-tuning) and stage 2 are reported in the Table 4. In addition to using fine-tuned language models, we have also varied the pooling method in the "mixing layer" of our network (see additional study in Appendix B). We observe that the network trained with mean-pooling layer did not gain improvements from multi-stage process. However, the network trained with a max-pooling layer demonstrates 53.5% macro average F_1 score, outperforming the model trained only with the stage 2 (§ 4.2).

4.5 Label Quality Analysis

	FW	ASR	ERR	NLU	Result	Correct	Avg
F_1^{FPI}	40.9	55.1	30.4	42.5	57.8	94.1	53.5
F_1^{Human}	57.4	61.5	33.8	44.0	56.1	91.6	57.4
F_1^{FPI} / F_1^{Human}, %	71.2	89.6	89.9	96.7	103.1	102.7	92.2

Table 5: F_1 score comparison of the best FPI model (F_1^{FPI}) and standard annotator (F_1^{Human}).

In order to quantify human performance on FPI task, we compared label produced by a single annotator (non-expert), to the final label corrected by a highly trained annotator in our "golden" dataset (see § 3.2). Our analysis shows (see Table 5) that the task of isolating failure points is easier in the following three categories: ASR (F_1 score of 61.52 %), False Wake (F_1 score of 57.4%) and Result (F_1 score of 56%). Detecting NLU and ERR errors is the most difficult task with 44% and 34% F_1 scores in those classes respectively. We use this analysis to understand reasonable limits for our model which is trained on labels from a single annotator as opposed to the "golden" workflow.

The best FPI model, using the max pooling layer and a fine-tuned language model, on average achieves 92.2% of non-expert human F_1 score on the FPI task (see Table 5). The weakest performance is observed in False Wake detection with 71.2% of human F_1. The model achieves approximately 90% of human performance in ASR and ERR classes, 96.7% in NLU, and outperform humans in detecting Result and Correct errors. We believe that the model demonstrates strong performance in Result and Correct classes, as result errors could be captured by the dialog context, when repeating or restating user request often can lead to the same or similar results for the same user.

5 Limitations

During our research we identified several limitations in the FPI system. First, our training dataset only allows a single failure point, however multiple components of a dialog assistant can fail in a real-world system. Hence, it would be useful to extend FPI task for capturing all critical and non-critical errors regardless of whether they resulted in a defective user session. Second, our system provides only a component-level failure point isolation. Future systems could build on our work to identify the sub-components of a dialog assistant responsible for the failure. Next, it would be useful to develop a framework which would allow joint system-level error attribution and assessment of interaction quality (IQ). Such an approach would not only help developers understand system errors but also cases which result in negative customer interaction. We have not experimented with bigger language models for our application, which might demonstrate stronger performance than the models used in this work.

6 Conclusion

We present an effective machine learning system to detect and isolate failure points in a real-world conversational assistant. Such assistants can have a complex hierarchy of modules making error isolation very challenging. By leveraging pre-trained transformer models to process the request text and contextual metadata features, our system obtains 92.2% of human performance. Given the large volumes of traffic in real-world conversational assistants, the manual process of obtaining human annotations for error isolation is prohibitively time consuming and expensive. While achieving human parity, our system automates this process and scales to a large volume of traffic. We conduct detailed ablation studies of our system and illustrate the key components that led to the highlighted gains.

7 Ethical Considerations

The data used in this paper was collected in accordance with applicable policies, terms of use, privacy notices, and customer privacy settings that disclose to customers how their data may be used. The annotators of the data were compensated for their work consistent with applicable laws and regulations.

Acknowledgements

We would like to thank Mustafa Hameed, Adrien Carre, Vivek Gupta, and Sarah Traylor for managing the program; Adam Berger, Krunal Sheth, Anh Nguyen, Daniel Lawrence, and Emmanuel Gonzalez for software development support; Yan Wang and Claude Paugh for data pipeline support; and Xiao Gong and Matthew Tucker for early exploration work.

References

Praveen Kumar Bodigutla, Aditya Tiwari, Spyros Matsoukas, Josep Valls-Vargas, and Lazaros Polymenakos. 2020. Joint turn and dialogue level user satisfaction estimation on multi-domain conversations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3897–3909, Online. Association for Computational Linguistics.

Rakesh Chada, Pradeep Natarajan, Darshan Fofadiya, and Prathap Ramachandra. 2021. Error detection in large-scale natural language understanding systems using transformer models. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 498–503, Online. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Willian Falcon and The PyTorch Lightning eam. 2020. Pytorch lightning.

Saurabh Gupta, Xing Fan, Derek Liu, Benjamin Yao, Yuan Ling, Kun Zhou, Tuan-Hung Pham, and Chenlei Guo. 2021. Robertaiq: An efficient framework for automatic interactionquality estimation of dialogue systems. In *KDD Workshop: Data-Efficient Machine Learning*, volume 2657. CEUR-WS.

Yanzhang He, Tara N. Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, Qiao Liang, Deepti Bhatia, Yuan Shangguan, Bo Li, Golan Pundak, Khe Chai Sim, Tom Bagby, Shuo-yiin Chang, Kanishka Rao, and Alexander Gruenstein. 2019. Streaming end-to-end speech recognition for mobile devices. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6381–6385.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132, Austin, Texas. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic Turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126, Vancouver, Canada. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle,

A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Alexander Schmitt and Stefan Ultes. 2015. Interaction quality: Assessing the quality of ongoing spoken dialog interaction by experts—and how it relates to user satisfaction. *Speech Communication*, 74:12–36.

Alexander Schmitt, Stefan Ultes, and Wolfgang Minker. 2012. A parameterized and annotated spoken dialog corpus of the CMU let's go bus information system. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3369–3373, Istanbul, Turkey. European Language Resources Association (ELRA).

Pooja Sethi, Denis Savenkov, Forough Arabshahi, Jack Goetz, Micaela Tolliver, Nicolas Scheffer, Ilknur Kabul, Yue Liu, and Ahmed Aly. 2021. Autonlu: Detecting, root-causing, and fixing nlu model errors.

Siddharth Sigtia, Rob Haynes, Hywel B. Richards, Erik Marchi, and John Scott Bridle. 2018. Efficient voice trigger detection for low resource hardware. In *INTERSPEECH*.

Koustuv Sinha, Prasanna Parthasarathi, Jasmine Wang, Ryan Lowe, William L. Hamilton, and Joelle Pineau. 2020. Learning an unreferenced metric for online dialogue evaluation.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–60010. Curran Associates Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mt5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498. Association for Computational Linguistics.

A Training Parameters

We use AdamW (Loshchilov and Hutter, 2019) optimizer with a fixed learning rate of 5×10^{-5} and batch size of 1024 examples. We train the model for 30 epochs or until we reach early stopping criterion, 5 epochs sequential epochs that do not improve validation loss function. A single training run takes up to 90 hours on NVIDIA V100 GPU.

Our training setup is leveraging PyTorch (Paszke et al., 2019), HuggingFace (Wolf et al., 2020), and PyTorch Lightning (Falcon and eam, 2020). Those libraries were used according to their intended use and distributed under BSD or Apache licenses.

B Experiments with the pooling layer

Pooling layer	F_1 scores						
	FW	ASR	ERR	NLU	Result	Correct	Avg
token	26.7	44.2	3.2	19.9	11.9	91.2	32.8
max	37.5	52.4	21.6	38.5	54.5	93.8	49.7
mean	39.7	53.8	27.5	39.9	53.4	94.0	51.4

Table 6: Performance of FPI models trained with different configurations of the pooling layer. "token" value in the *Pooling layer* column represent first-token pooling layer, "max" represent max-pooling layer, and "mean" represents mean pooling configuration.

Our findings indicate that the structure of the pooling layer makes a significant impact on the model performance. The commonly used first-token embedding (Devlin et al., 2019) performed the worst with the macro average F_1 score of 32.8%. The mean and max pooling layers demonstrated better performance with F_1 score of 51.4% and 49.7% respectively. All of the subsequent experiments were conducted with max and mean pooling layers.