SELF-HEALING THROUGH ERROR DETECTION, ATTRIBUTION, AND RETRAINING

Ansel MacLaughlin, Anna Rumshisky, Rinat Khaziev, Anil Ramakrishna, Yuval Merhav, Rahul Gupta

Amazon Alexa

ABSTRACT

Negative feedback received from users of voice agents can provide valuable training signal to their underlying ML systems. However, such systems tend to have complex inference pipelines consisting of multiple model-based and deterministic components. Therefore, when negative feedback is received, it can be difficult to attribute the system error to a specific sub-component. In this work, we address this challenge by building a system for error attribution and correction. We prototype attributing errors to the ML models used for domain classification (DC) in the NLU component of an assistant's pipeline, using a combination of a model and rule based system. We propose a simple method to add these detected errors directly to offline DC model training, and study our system's effectiveness on a challenging test set of low-frequency utterances. Our experiments on nine domains suggest that augmenting DC training data with our method significantly improves performance on a majority of them.

***Index Terms*—** Error correction, domain classification, natural language understanding

1. INTRODUCTION

To improve the performance of a machine learning (ML) system, modelers must identify instances where their system is making a mistake, then change the system's behavior to correct it. In the relatively simple setups studied in most academic research, these systems typically consist of a single ML model. Thus, through manual error analysis [1], practitioners can then often innovate on the model architecture, training or pre-training procedure, or data-augmentation process to boost system performance (see the surveys of [2, 3, 4]).

This mostly straightforward process of analyzing errors and improving the ML model, however, does not translate cleanly to real-world commercial ML systems, such as digital assistants like Alexa and Siri. These assistants are not single ML models, but consist of multiple components, including systems for automatic speech recognition (ASR), natural language understanding (NLU), entity resolution (ER) [5, 6], and others. These components are then run in a pipeline to generate the final result returned to each customer. Due to this setup's sequential nature, errors made in an earlier part of the pipeline affect results returned by later components, making it difficult to determine the root cause of a mistake [7].

Further, each of these components can be complex, consisting of multiple ML models and deterministic artifacts, the outputs of which are also pipelined and combined in highly-engineered ways [8]. This complexity, combined with large traffic volumes and decoupled model development [9] (ASR models tend to be trained independently of NLU, etc.), make it hard to determine which models are responsible for the full-system result, and even harder to assign blame to and correct the behavior of a specific model after a mistake.

In this work, we focus on isolating and correcting mistakes made by a specific part of the NLU system, the domain classification (DC) models. DC models classify utterances by domain, which are defined for a specific application such as shopping or playing videos. To correct and improve these models, we develop a multi-stage, offline pipeline to (1) detect defective predictions made by commercial digital assistants, (2) determine whether the DC models are to blame for the defect, and (3) use the blameable examples to improve DC performance. Our pipeline combines SOTA methods for dialog system evaluation adapted for our specific use-case with a set of manually-crafted data filters. We evaluate our pipeline on DC classifiers for nine NLU domains, measuring performance on a challenging internal test set. We find that, not only is our method able to identify, with high precision, a set of DC errors that previously caused dissatisfaction among the system's users, augmenting DC model training with these errors successfully corrects model performance on a majority of our tested domains, significantly improving over a strong in-production DC baseline.

2. RELATED WORK

The prior work most similar to ours is [10], who propose a method to detect and correct DC errors in commercial NLU systems. Their method, however, is more manually intensive, requiring multiple new sets of human-annotated data to train their DC error detection model and label predicted DC errors with their true domains. In contrast, our method reuses *existing* labeled data to train our defect detection and attribution models and does not require annotation of predicted DC errors to incorporate them into offline model training. Further, unlike [10], our method focuses on fixing only errors where the DC models were invoked for the prediction. Since deterministic systems (e.g. rules, FSTs) may make predictions for many utterances, many of the errors detected by the system of [10] cannot be fixed through changes to ML models. To address this, we develop an extensive set of data filters to ensure that our method only detects errors made by the DC models.

Also related is the work of [11], who propose a system for NLU error detection, attribution, and correction. Our work and theirs differ in three key aspects. First, rather than attributing errors to a specific NLU component (DC models), they perform an easier task of detecting errors made by *any* part of the NLU system. Second, [11] only use NLU confidence scores to judge NLU correctness, while we use model-based methods which operate on dialog interaction features along with metadata features such as confidence scores. Since their method can only identify erroneous utterances with low NLU scores, it misses errors with high-confidence scores – an issue because neural models are often poorly calibrated [12]. Finally, most of their error-correction methods involve substantial manual effort, e.g. fixing or writing rules or annotating new data. We, on the other hand, directly add erroneous utterances to offline training sets with no further annotation.

Other related work includes [13], who consider a simpler task where errors can only be caused by NLU ML models (not by other components, rules, etc.) and only conduct experiments on datasets with simulated feedback. Further, [14] and [15] also study NLU defect identification and correction. Their methods, however, require that defective utterances have subsequent, non-defective rephrases to provide fixes, while our method has no such requirement. Finally, our method builds on prior work, such as [16, 17] that have explored applications of self-learning to NLP and computer vision. These methods train an initial classifier on a small set of labeled data then, in a second step, train a new classifier on a mix of the original labeled data and data self-labeled by the initial classifier. Unlike these prior works, however, we use human feedback to inform our self-labeling process.

3. ERROR DETECTION, ATTRIBUTION, AND CORRECTION

Our pipeline for self-healing DC contains three components: (1) a method for system-level defect detection, (2) a method for fine-grained error attribution, and (3) a method for improving the responsible DC model with this data.

3.1. System-level Defect Detection

We first identify the utterances on which the assistant made a mistake. For this coarse-grained prediction, we use the SOTA interaction quality model RoBERTaIQ [18]. We follow the setup described in [18], training RoBERTaIQ on an existing internal dataset containing hundreds of thousands of historical multi-turn dialogues (user utterances and assistant responses). These dialogues are annotated with binary interaction quality labels indicating whether, from the end user's perspective, the assistant system made a mistake or gave a satisfactory result. Since these labels are only representative of the user's perception of their interactions with the assistant, they do not indicate whether a specific component or ML model, such as DC, made a mistake. See [18] for further modeling setup details.

3.2. Fine-grained Error Attribution

Our error attribution method consists of (1) manually-designed filters to remove utterances where the DC model was *not* invoked, and (2) a model-based method to predict whether the DC model was to blame for a system defect.

Deterministic Filtering: Large-scale NLU systems are often modularized into many domains [8, 19]. Each domain's module contains multiple domain-specific components – deterministic artifacts (rules, FSTs) and ML models for DC, IC, and NER. For each input, the NLU system's final output will be selected [8] from each module's deterministic artifact matches and ML model interpretations. Since we are interested in fixing DC model mistakes, we focus only on utterances where the NLU system used the DC predictions. If, for instance, a rule was used instead, any resulting error cannot be fixed by modifying the DC model. Thus, through extensive data analysis, we devise a thorough set of system-specific filters[1] to remove traffic not handled by the DC model.

Model-based Error Attribution: Given a set of filtered erroneous utterances, we need an error-attribution method to determine if the DC model was to blame for an defective system output. For this purpose, we adapt the Failure Point Isolation (FPI) model introduced in [7]. Similar to RoBERTaIQ, FPI operates on features extracted from multi-turn dialog data, including user requests, assistant responses, and the time separation between the requests. Additionally, it also uses intermediate metadata from the dialog system, including predictions and scores from each component, e.g. NLU's top-k interpretations and scores. These features are crucial for error attribution, since they allow FPI to identify patterns in each component's behaviour associated with defective outputs. See [7] for further details on FPI's model architecture.

The original FPI model is fine-tuned on human-annotated data with labels indicating whether a turn was correct or whether a primary system component (e.g. ASR, NLU, ER) caused a failure. We obtained FPI's training data from the original authors to see if it is useful for our DC error-attribution task. We found that the original label set is not fine-grained enough – utterances annotated with NLU errors could have mistakes caused by any NLU ML model (DC, IC, NER) or deterministic system (e.g. an incorrect rule). Since, in this work, we are focused on improving DC, we need fine-grained error attribution to differentiate between errors made by DC versus other parts of the NLU pipeline.

[1]Example filters: exact-match rules not invoked, FSTs not invoked, rules-based contextual systems not invoked

To address this, we create a new fine-grained DC-error-attribution dataset and train a new version of the FPI model (same model architecture, features, experimental design) on it. We call this new model DC-FPI. In order to generate our new dataset, we need a method to determine whether the DC model made a mistake on a given utterance. Unlike [10], who collect new, human-annotated data to train a DC error attribution model, we instead devise a novel, yet simple, method to generate DC error attribution labels by reusing existing production DC model training data. To generate labels, we take a set of labeled utterances used for DC model training and compare their ground-truth labels to the domain originally predicted by the historic production NLU system. We then label an utterance (observed by the historic NLU system) as having a DC error if there is a mismatch between the two.

To generate a dataset to train and evaluate DC-FPI, we use a large set of recently-annotated DC data (derived from four recent months in our case). For each utterance, we extract the metadata and textual features for input to DC-FPI and generate a binary label by comparing its annotated domain with the historic NLU DC prediction. We next apply our set of manually-designed filters to keep only utterances where the DC model was used for prediction. We split this data into train, val and test sets in non-overlapping temporal windows. Finally, we apply RoBERTaIQ (§3.1) to the utterances in our val and test sets, keeping only utterances with predicted defects. Validating and testing on this set of defective utterances allows us to measure DC-FPI's ability to attribute system defects to the DC models.

We train a binary DC-FPI classifier, with the label **1** indicating a DC error. On its test set, DC-FPI achieves an F1 of 0.64 (Precision 0.69, Recall 0.59). For our task, however, high precision (correctly identifying DC errors) is more important than high recall. We find that by using a higher classification threshold (e.g. DC-FPI confidence > 0.9) we can achieve strong precision (≈ 0.85) on a large number of domains while maintaining acceptable recall (≈ 0.16) to detect a variety of DC errors (note that a high precision is attractive when user traffic is high in volume). We apply the trained DC-FPI model to a new set of filtered, defective, un-annotated traffic (after the training window) to then identify utterances with DC errors.

3.3. Error Correction for Domain Classification

For our application, we follow the modularized NLU setup of [8], where each of the n domains has its own binary DC classifier. Each classifier is trained on the same dataset, but with different labels depending on which label is in-domain. For example, "*read the great gatsby*" would be labeled **1** to train the Books domain DC classifier and **0** for all others.

Given a defective utterance with a DC error, we only know which domain that utterance *should not* be assigned to. The correct domain is unknown. For offline DC training, we can use this information by adding these erroneous utterances to the hypothesized domain's training data with a negative label. A binary DC model trained on this augmented training set should then predict lower in-domain class probabilities for these and similar utterances at inference. We compare the offline test performance of DC models trained with and without these added negatives to evaluate the utility of our methods.

4. EXPERIMENTAL SETUP

To evaluate the effectiveness of our error-correction methods, we compare DC models trained on standard training sets to those augmented with our predicted DC errors. As our baseline, we use an internal production dataset containing several million utterances annotated with ground-truth domains spanning tens of domains. While training data is the same across domains [8], a separate binary DC classifier is trained for each with domain-specific in/out-of-domain (OOD) labels. For our experiments, we train binary DC classifiers for nine domains: Books, Calendar, CinemaShowTimes, Communication, DailyBriefing, GeneralMedia, Global, Shopping, and Video. We select these domains as they cover a large variety of user requests and have varying traffic volumes.

We augment this baseline training set with domain-specific negatives generated as follows: 1. We apply RoBERTaIQ to a sample of recent unlabeled runtime traffic from our nine selected domains to identify utterances with system defects. 2. Next, our manually-designed filters remove defective traffic where the DC model was not invoked. 3. Finally, we run inference with our DC-FPI model on these filtered erroneous utterances, using a high classification threshold (0.9) to ensure high-precision.

For each our nine domains we now have a set of utterances where (1) the historical production system predicted the corresponding domain, e.g. Books, and returned a result to the user, (2) the production DC model was used for that prediction (3) the user perceived a defect with that result, and (4) incorrect DC is predicted to be the cause of that defect. For each domain, we can then augment our internal DC training set with this domain-specific set of DC errors, label them with the OOD class, and fine-tune a new DC model.

For each domain's DC model, we use the same, in-production model architecture – a 4-layer BERT [20] distilled [21] from a larger BERT teacher pretrained on an internal corpus. Due to production constraints, the input to these models is *only the text of a single utterance*. Unlike RobertaIQ and DC-FPI, they do not leverage extra metadata or prior dialog context. We tune the learning rate and batch size for each DC model.

For evaluation, we use an internal test set of low-frequency utterances, manually annotated with their correct DC. This dataset contains hundreds of thousands of labeled utterances sampled from the tail of the live traffic. Like our training data, it contains utterances from tens of domains, and we generate

Table 1. DC results for Prod (production baseline), IQ (RoBERTaIQ), IQF (RoBERTaIQ+filters), and IQFDC (our method). P/R is short for Precision-Recall AUC. Performance is relative to the prod baseline. For each domain, **bold** entries outperform all other models. * indicates significant improvement over prod baseline: randomization test, $p < 0.05$.

	Books		**Calendar**		**CinemaShowTimes**		**Communication**	
	F1	P/R	F1	P/R	F1	P/R	F1	P/R
Prod	–	–	–	–	–	–	–	–
IQ	-1.8	-5.0	-7.0	-5.8	**+14.7***	-1.2	-11.1	-11.9
IQF	-2.2	-4.5	-4.4	-5.1	+12.2	-7.8	-11.8	-7.3
IQFDC	**+1.7***	**+0.3***	**+1.5***	**+0.4**	+10.8*	**+1.9**	-1.3	-1.5

	DailyBriefing		**GeneralMedia**		**Global**		**Shopping**		**Video**	
	F1	P/R	F1	P/R	F1	P/R	F1	P/R	F1	P/R
Prod	–	–	–	–	–	–	–	–	–	–
IQ	+0.1	-1.9	-21.9	-35.4	-1.5	-3.2	-0.8	-0.7	-1.0	-2.3
IQF	-1.6	-2.1	-23.5	-35.0	-4.0	-5.3	-0.8	-0.5	-0.6	-1.0
IQFDC	**+1.3***	**+0.3**	-6.3	-8.4	-0.4	-0.5	**+0.3***	**+0.1***	0.4	**0.0**

nine different binary labelings for our nine DC classifiers. We evaluate on such utterances as we find that error-causing user queries tend to be low-frequency since the head of the agent's traffic is often handled with deterministic systems.

We evaluate our method against three baselines: 1. **Production**: the in-production baseline is trained on the internal DC training set with no augmentation. 2. **RoBERTaIQ**: we augment the DC training set with negatives sampled from the set of defective utterances detected by RoBERTaIQ. No data filters or DC-FPI. 3. **RoBERTaIQ+filters**: same as RoBERTaIQ, except we filter the defective utterances with our manually-crafted data filters. No DC-FPI.
We evaluate with F1 and precision-recall AUC and report relative performance change versus the production baseline. For RoBERTaIQ and RoBERTaIQ+filters, we augment with the same number of utterances (per domain) as our full method.

5. RESULTS

Table 1 shows the test results for our IQFDC method and the baselines. We find that adding our method's negatives improves over the production baseline on six of nine domains. On those domains, it increases relative F1 by 3.1% and PR-AUC by 0.6%, on average. Our method degrades performance on three domains (Communication, GeneralMedia, Global); analyzed in the next section.

We also observe that neither baseline augmentation method (RoBERTaIQ, RoBERTaIQ+filters) is consistently better than the production model, often lagging behind by more than 2% PR-AUC. These weak performances highlight the importance of our full method's fine-grained error attribution with DC-FPI. Unless, by chance, the DC model is to blame for most erroneous utterances in a given domain, many utterances augmented with these baselines will have errors caused by other models or components. This noise in the augmentation data helps cause the performance decreases found for both baselines across almost all domains.

5.1. Error Analysis

We analyze in further depth the three domains (Communication, GeneralMedia, Global) where our method underperforms the production baseline. For each domain, we focus on two sets of data: (1) predicted errors added to the DC training set; (2) test utterances which our method's DC misclassified. We learn the following from these analysis:

Correctly identified negatives may not always improve production model's performance: For analysis one, we analyze utterances with predicted DC errors from each of the three domains, along with their multi-turn dialog contexts. We find that, while most indeed have DC errors, some are ambiguous and difficult to classify when viewed in isolation. This ambiguity is no problem for the models we use to detect and attribute errors as they use multi-turn dialog features. Since our production DC models, however, do not make use of such features, adding ambiguous negatives, therefore, might hurt performance if the negative domains are reasonable without added context. Future work can evaluate contextual DC models [22] that might better learn from the complex erroneous utterances our method detects.

Domains that may not benefit from adding negative data can be identified based on the performance of the DC-FPI model: Breaking down the performance of the DC-FPI model, we observe that under-performant domains observe a weaker error attribution accuracy. For instance DC-FPI's performance on GeneralMedia, Global and Communications are 0.42, 0.31 and 0.35, respectively. Therefore, while we observe that not all domains may benefit from our method, such domains can be filtered out based on the performance of the DC-FPI model.

6. CONCLUSION

In this work, we propose a system to detect, attribute, and correct errors in commercial NLU systems. Using the ML models for DC as a testbed, we focus on finding and correcting system defects caused by incorrect DC. These efforts are complicated by the pipelined setup of commercial dialog systems – many ML and deterministic systems contribute to the full system response and could cause an error. To address this challenge, we develop and adapt SOTA ML models and manually-crafted data filters to determine whether DC models caused a system error. Finally, we propose a method to add these errors to offline DC model training to correct model behaviour and increase performance. We conduct experiments on nine NLU domains, and find that our method often significantly boosts performance – on five of the nine it improves F1 by 3.1% relative, on average, over an in-production baseline.

Next steps in the project can extent the error attribution to multiple SLU components and enhancing their performance through a single error attribution system. We also target to enrich the feature set for our error attribution system to improve accuracy for domains where we don't observe benefit.

7. REFERENCES

[1] A. Ng, "Advice for applying machine learning," Stanford CS 229: Machine learning, 2011.

[2] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *ArXiv*, vol. abs/2003.08271, 2020.

[3] C. Shorten, T. M. Khoshgoftaar, and B. Furht, "Text data augmentation for deep learning," *Journal of Big Data*, vol. 8, 2021.

[4] D. Otter, J. Richard Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 604–624, 2021.

[5] T. Wang, J. Chen, M. Malmir, S. Dong, X. He, H. Wang, C. Su, Y. Liu, and Y. Liu, "Optimizing NLU reranking using entity resolution signals in multi-domain dialog systems," in *Proceedings of NAACL: Industry Papers*, 2021.

[6] W. Ruan, Y. Nechaev, L. Chen, C. Su, and I. Kiss, "Towards an asr error robust spoken language understanding system," in *INTERSPEECH*, 2020.

[7] R. Khaziev, U. Shahid, T. Röding, R. Chada, E. Kapanci, and P. Natarajan, "Fpi: Failure point isolation in large-scale real-world conversational ai agents," in *Proceedings of NAACL: Industry Papers*, 2022.

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

[9] Y. Nechaev, W. Ruan, and I. Kiss, "Towards nlu model robustness to asr errors at scale," in *KDD 2021 Workshop: Data-Efficient Machine Learning*, 2021.

[10] R. Chada, P. Natarajan, D. Fofadiya, and P. Ramachandra, "Error detection in large-scale natural language understanding systems using transformer models," in *Findings of ACL-IJCNLP*, 2021.

[11] P. Sethi, D. Savenkov, Arabshahi F, J. Goetz, M. Tolliver, N. Scheffer, I. Kaynar Kabul, Y. Liu, and A. Aly, "Autonlu: Detecting, root-causing, and fixing nlu model errors," *ArXiv*, vol. abs/2110.06384, 2021.

[12] A. Kumar, S. Sarawagi, and U. Jain, "Trainable calibration measures for neural networks from kernel mean embeddings," in *ICML*, 2018.

[13] T. Falke and P. Lehnen, "Feedback attribution for counterfactual bandit learning in multi-domain spoken language understanding," in *Proceedings of EMNLP*, 2021.

[14] S. Park, H. Li, A. Patel, S. Mudgal, S. Lee, Y. Kim, S. Matsoukas, and R. Sarikaya, "A scalable framework for learning from implicit user feedback to improve natural language understanding in large-scale conversational AI systems," in *Proceedings of EMNLP*, 2021.

[15] T. Falke, M. Boese, D. Sorokin, C. Tirkaz, and P. Lehnen, "Leveraging user paraphrasing behavior in dialog systems to automatically collect annotations for long-tail utterances," in *Proceedings of COLING: Industry Track*, 2020.

[16] Yue Yu, Lingkai Kong, Jieyu Zhang, Rongzhi Zhang, and Chao Zhang, "Actune: Uncertainty-based active self-training for active fine-tuning of pretrained language models," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2022, pp. 1422–1436.

[17] David McClosky, Eugene Charniak, and Mark Johnson, "Effective self-training for parsing," in *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, 2006, pp. 152–159.

[18] S. Gupta, X. Fan, D. Liu, B. Yao, Y. Ling, K. Zhou, T. Pham, and C. Guo, "Robertaiq: An efficient framework for automatic interaction quality estimation of dialogue systems," *ArXiv*, 2021.

[19] G. Tur and R. De Mori, *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, Wiley, 2011.

[20] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL*, 2019.

[21] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *ArXiv*, vol. abs/1503.02531, 2015.

[22] Tzu-Hsiang Lin, Yipeng Shi, Chentao Ye, Yang Fan, Weitong Ruan, Emre Barut, Wael Hamza, and Chengwei Su, "Contextual domain classification with temporal representations," in *Proceedings of NAACL: Industry Papers*, 2021.