Online Proactive Escalation in Multi-Modal Automated Assistants

Cynthia Freeman, Ian Beaver
NextIT Corporation,
12809 E Mirabeau Pkwy, Spokane Valley WA 99216 USA
http://www.nextit.com

Abstract

Existing research on escalation recommendation often relies on acoustic features, obtainable in Spoken Dialog Systems (SDS). It is less understood how multi-modal dialog systems can recommend escalations online without access to such features. Several machine learning techniques are evaluated using only text features common to all dialog systems. We then present and implement a general criteria for online escalation recommendation based on the conversation structure, presence of *correctional language*, user request repetition, user *intent*, and polarity. Our method is designed to work with any automated assistant using text or speech input, even where inputs can alternate between text and speech. We achieve higher precision using less training data than several standard machine learning techniques on a dataset consisting of 7,754 conversations with live multi-modal automated assistants.

Introduction

To maintain quality of service, users are sometimes directed to human representatives when a conversation between the user and an automated assistant cannot be resolved. This transfer is known as an *escalation*. Escalations may be user or system-initiated depending on which party first determines that progress in completing a task is not being achieved. We focus on the topic of system-initiated escalation and how to determine which conversations should escalate.

Our company creates Intelligent Virtual Assistants (IVAs) on behalf of other organizations for the purposes of product support and customer service. These IVAs range from information retrieval (IR) agents for tasks such as finding insurance or financial forms to large scale mixed-initiative dialog systems containing hundreds of tasks and tens of thousands of user intents. User interaction may occur through voice input, text input, or fully multi-modal interfaces that allow for any combination of voice, text, clicking-on-controls, and web content.

Many IVAs provide the means to escalate a conversation to a human operator if necessary. Our task is to build a system to recommend when an escalation should occur, regardless of the input channel, to prevent user abandonment. Dialog systems contain a component for Natural Language Understanding (NLU) which maps a textual representation to a representation of the meaning, or *intent*, expressed by the user (Lison 2013). As the textual input to the NLU is the lowest common denominator of the various interfaces supported by multi-modal IVAs, we use this textual form for escalation recommendation. In this way, we ensure that regardless of the channel employed by the user, the recommendation system can still function.

In reviewing current literature on proactive escalation methods, we found no method suitable for our application. All existing methods rely to some extent on acoustic features generated by the Automatic Speech Recognition (ASR) component or other features dependent on language model implementation. Such features may not be present in a multi-modal IVA. Therefore, we set out to create a method with no reliance on features specific to channel; only the input and output of the NLU are consulted. We first evaluated several standard Machine Learning (ML) techniques but were not satisfied with their performance. We developed an algorithm that not only outperforms the ML approaches using textual features but also does not require a large labeled training set for each language domain *and* is considerably more time efficient.

We begin with a review of the literature involving the recommendation of escalations. We discuss our data and the evaluation of several standard ML techniques. However, we obtain even higher precision on an escalation criteria we develop, discussed in Algorithm 1. We also detail how this algorithm is implemented. Finally, we discuss the results, addressing limitations and outlining future work.

Related Works

Similar research on recommending escalations typically involves Spoken Dialogue Systems (SDS) such as AT&T's *How May I Help You* (HMIHY) application (Gorin, Riccardi, and Wright 1997). These systems have access to a wide range of acoustic features that are used in machine learning to identify problematic conversations and transfer a customer to a human customer care representative before the conversation fails. In (Langkilde et al. 1999), a machine learning program, RIPPER, achieved an accuracy of 72% in identifying problematic dialogues after the very first exchange and 86.7% accuracy given the whole conversation. However, over 50 features were available from spoken di-

alogues, of which almost a quarter were acoustic and ASR features. Acoustic features were also used in (Schmitt et al. 2010) where 55% accuracy was achieved on over 40,000 phone calls and (Walker et al. 2000) where 92.9% accuracy was achieved on 4,692 dialogues collected with the HMIHY system. A slightly higher accuracy of 93% was obtained using JRip in (Meena, Skantze, and Gustafson 2015), but this is for *offline* detection of miscommunication which consumes the completed conversation whereas we are attempting *online* recommendation of escalation.

In (Horvitz and Paek 2003), call duration was used as an indicator for when to escalate, and probabilistic models were constructed to generate policies identifying the best point in time to transfer callers to human operators. Although a useful feature for auditory assistants, duration of a conversation is not applicable to textual assistants; some customers may read or type more slowly than others, and pauses between turns are common. To demonstrate this, we collected a set of 8k conversations with a text-based IVA that logged explicit user escalation requests in order to measure correlation with duration. A correlation coefficient of .269 (calculated using the Scipy stats module) was obtained. Alternatively, one might consider the number of turns in the conversation before escalation. For this, we obtained a correlation coefficient of .26; therefore, turn or duration based strategies alone do not appear fruitful in predicting escalation.

Our strategy for recommending escalation is inspired by (Krahmer et al. 1999) where a SDS provides train timetable information. Although their goal was to detect a single misunderstood turn and not necessarily recommend escalations, we find that the negative cues used by these authors are helpful in the recommendation of escalation. The authors assume the Principle of Minimal Collaborative Effort; both the user and system want the dialogue to be finished as efficiently as possible and with success. Cues are examined and certain combinations of cues have the best predictive potential for discovering the presence or absence of problematic conversations. Cues include turn length, marked or unmarked word order (topicalization or extraposition), confirmation, the presence or absence of an answer, corrections or repetitions, and new information. The highest precision is achieved with a combination of correction and repetition cues on a small set of 120 dialogues; users tend to repeat their requests and correct the system in its interpretation of these requests when there are communication problems. Our criteria for escalation recommendation is heavily influenced by this work.

Methods

We collected 7,754 conversations (20,808 user turns) across two commercial multi-modal IVAs deployed on corporate travel websites and mobile applications. The IVAs help with booking and changing flights and answer various travel-related queries. The conversations were manually tagged for whether or not they should have escalated by 2 reviewers, generating a Cohen's κ of .6. A third reviewer was used to break ties in cases of disagreement, and the majority determined the final tag. In addition to determining if a conversation should escalate, reviewers also notated the turn on which an escalation should occur. The escalation point, if existent, was averaged between the reviewers and rounded down to the nearest integer. Rounding down favors the user as he or she will be escalated faster. Rounding up benefits the company by saving money from delaying the use of human customer assistants. Of the 7,754 conversations, 1,268 were marked for escalation by a majority. A random 80-20 split was used to create training and testing sets. The training set consisted of 6,203 conversations of which 1,027 should escalate. The testing set consisted of 1,551 conversations of which 241 should escalate.

Preliminary Experiments

We initially experimented with several standard machine learning algorithms. As we were not only interested in a model's ability to determine *if* an escalation should occur but also *when*, the models were trained and tested on a cumulative turn basis. For example, if a conversation in the training set is four turns long and is tagged for escalation on the final turn, the model will be trained on $(turn1, 0), (turn1 + turn2, 0), ..., (turn1 + ... + turn4, 1)$.

As high accuracy was achieved using JRip in (Meena, Skantze, and Gustafson 2015), we tried WEKA (Hall et al. 2009) JRip. Preprocessing of conversations was done using WEKA String to Word Vector. Fourteen rules were generated from the training set. A precision of .387 and recall of .051 was obtained on the test set for escalations. We also experimented with WEKA's Random Forest (RF) (with 100 trees), resulting in higher precision (.735) but equally terrible recall (.036). WEKA's SVM had better recall (.562) and precision (.317) compared to JRip. Default parameters were used. We then trained a Convolutional Neural Network (CNN) inspired by (Kim 2014) using Keras. We used the same parameters and CNN build in (Rakhlin 2016). GenSim's Word2Vec was used to preprocess data. Over 100 epochs on CNN-rand, we obtained .776 precision and .718 recall on escalations. We also tried several Recurrent Neural Network (RNN) methods, but it appeared there was not enough data to train a useful model.

Point of Escalation

If a model chooses to escalate a conversation earlier than the tagged turn, the model is *aligned* with the customer as the customer will be escalated faster. If the model escalates a conversation later than the tagged turn, the model is aligned with the company. Alignments were calculated from conversations where the model and reviewer majority agreed that the conversation should escalate (even though the point of escalation may differ). So either reviewer 1 and reviewer 2 both believe the conversation should escalate, or they disagree, but reviewer 3 believes the conversation should escalate. In both cases, there are two reviewers that tag a 1 for the conversation. Suppose conversation Z has a majority vote to escalate, *and* the model chooses to escalate Z also. Suppose reviewer A and reviewer B tag Z for escalation, forming the majority. Let

$$X = \text{model's Predicted Turn} - \text{A's tagged turn}$$

$$Y = \text{model's Predicted Turn} - \text{B's tagged turn}$$

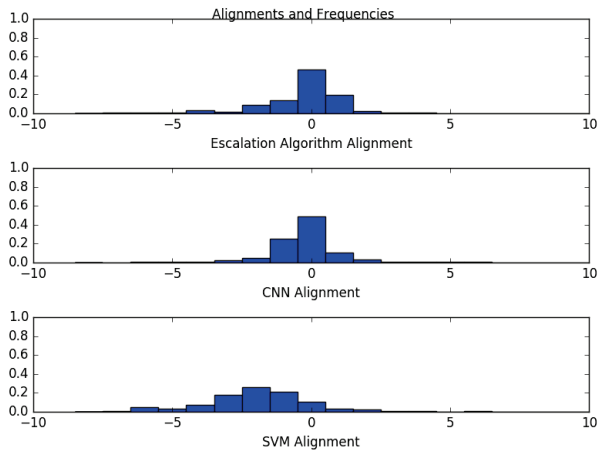

Figure 1: Alignments of CNN, SVM, and Algorithm 1

The alignment for a model on Z is the average of X and Y.

Thus, if an alignment is negative, the model prefers escalating early and favors the customer. If the alignment is positive, the company is favored. The number of conversations where the model and reviewer majority agreed that the conversation should escalate will be notated by $agree_{model}$. All frequencies were normalized by their respective $agree_{model}$. Figure 1 includes alignment charts for the CNN ($agree_{CNN} = 173$) and SVM ($agree_{SVM} = 130$) discussed above. JRip and RF were not included due to exceptionally poor recall.

CNN is more fair to both the customer and company whereas the SVM tends to favor the customer and escalate earlier. Indeed the skew value (calculated using SciPy) is .444 for the CNN and 6.96 for the SVM. A skew value greater than 0 indicates left skew whereas a skew value less than 0 represents right skew. A normally distributed dataset should have close to 0 skew. The alignment of our escalation algorithm ($agree_{EA} = 156$) is also shown in Figure 1 and will be discussed in the following sections.

Escalation Algorithm Development

While the CNN model produced nearly acceptable results, the number of false positives was still too high for production use. In addition, the burden of collecting and tagging sufficient data from each IVA in order to train a CNN for its language domain was too great. For example, a model trained from insurance claims language cannot be reused in product support, so a new model must be constructed. We require an approach that can be reapplied to any IVA with a minimal amount of language domain specific tuning.

Inspired by (Krahmer et al. 1999), which achieved good results on single turn misunderstanding detection, albeit on a small test set, we construct a similar detection strategy for escalation. Using only the set of 6,203 training conversations, we performed a manual analysis to determine if there were common structures in the conversations or user turns that would indicate persistent communication issues. If problems are detected, the system should perform an escalation *before* the user explicitly asks for one. As false positives would be very confusing to the users and expensive for the company, we must be conservative in our approach by favoring precision over recall. The result of this manual analysis is presented below in Algorithm 1 and is hereafter referred to as the Escalation Algorithm (EA).

	User Turn	**Intent Hit**
1	im trying to buy upgrade, cant find it	Paid Upgrades
2	its not allowing me to buy an upgrade. Im already checked in	Paid Upgrades
3	this virtual assistant is NO assistance	IDK
4	I want to buy an upgrade for my flight today. Your links are not allowing me to do it	Paid Upgrades

Table 1: A customer *clarifying* his or her request with repetition, ultimately ending with frustration.

At the heart of the EA is the presence or absence of a *clarify section*. A clarify section consists of an explanation of the problem, an optional reaction to the automated assistant's response, and a restatement of the first explanation (lines 1-10 in Algorithm 1). Notice that a clarification is not considered an exact repeat of the previous input (lines 3,8). Clarify sections occur frequently since many customers initially do not know they are speaking to an automated assistant. Upon this realization, customers tend to repeat and clarify their requests (Table 1). If the assistant is not providing an appropriate response to the user's question, a customer typically responds with *correctional language* (e.g. "No, that is not what I meant."). If a clarify section is found, we determine if it is followed by the presence of correctional language, an explicit escalation request, remarks on the uselessness of the assistant, or anger. Otherwise, we check if there are 3 or more user turns following the clarify section that contain similar requests. If so, we choose to escalate the conversation as, by this point, the automated assistant has returned an unsatisfactory response at least 4 times.

For the same reason, if no clarify section is present, recommend escalation if there are 4 or more similar requests, or if correctional language, explicit escalation requests, remarks on the uselessness of the assistant, or anger are present in the conversation after the first turn. We make these checks *after* the first turn to give the automated assistant a chance to rectify the situation if the customer begins the conversation frustrated or immediately requests to speak to a different party. Finally, we check if the agent has returned the same response multiple times (lines 1,2,4 in Table 1), or has responded with an I Don't Know (IDK) multiple times. An IDK response occurs when the IVA cannot determine with high confidence what the user means and will reply with something like "I'm sorry I didn't understand you, try rewording your question" (line 3 in Table 1).

Algorithm 1: Escalation Recommendation Algorithm

```
1  clarify = false;
2  sim = calcSimilarity(userTurns[0],userTurns[1]);
3  if sim ≠ 1 and sim ≥ simThresh:
4      clarify = true;
5      index = 2;
6  else:
7      sim = calcSimilarity(userTurns[0],userTurns[2]);
8      if sim ≠ 1 and sim ≥ simThresh:
9          clarify = true;
10         index = 3;
11 if clarify == true:
12     afterClarify = userTurns[index:];
13     if detectCorrectionLang(afterClarify):
14         return true;
15     if detectEscalationLang(afterClarify):
16         return true;
17     if minSent(afterClarify) ≤ sentThresh:
18         return true;
19     sameReq = countSame(afterClarify,simThresh);
20     if sameReq ≥ 3:
21         return true;
22 else:
23     sameReq = countSame(userTurns,simThresh);
24     if sameReq ≥ 4:
25         return true;
26     if detectCorrectionLang(userTurns[1:]):
27         return true;
28     if detectEscalationLang(userTurns[1:]):
29         return true;
30     if minSent(userTurns[1:]) ≤ sentThresh:
31         return true;
32     if numRepeats(agentTurns) ≥ rptThresh:
33         return true;
34     if numIDKs(agentTurns) ≥ idkThresh:
35         return true;
36 return false;
```

Implementation Details

To determine the presence of correctional language (lines 13 and 26 in Algorithm 1), a set of 34 regular expressions was created from manual analysis of our training data[1]. For example, to detect the correctional language in line 2 of Table 2, a pattern such as "`^you did( not|nt) answer (my|the) question.+`" could be used. User turn was case-normalized, and punctuation was stripped prior to correctional language determination.

For the determination of explicit escalation requests (lines 15,28), two methods can be used. If the IVA understands escalation language, we can simply test if the IVA detected this in any of the user turns so far. If not, a stand-alone classifier for escalation language can be used. For our experiments, we implemented the latter exactly as described in (Beaver and Freeman 2016) so that we did not rely on any specific IVA implementation.

[1]https://s3-us-west-2.amazonaws.com/anon-share/FLAIRS2017_correctional_res.txt

	User Turn	Intent Hit
1	Hi -agent-, the seat map only shows rows D to F. Where did A to C go?	Seating Chart
2	you did not answer my question. Rows A, B and C are not visible on the website.	-agent-'s Misunderstanding
3	Seat A, B and C are not visible on the online seating chart.	Seating Chart
4	what is the telephone number to contact a human?	Contact Phone Numbers

Table 2: A customer clarifies his or her request with repetition, corrects the automated assistant, and ends the conversation with an explicit request for escalation.

Our algorithm makes use of thresholds for similarity and sentiment polarity, as well as number of repeated answers and IDKs. These are tuned using grid search and discussed in the following section. Polarity was measured using TextBlob's sentiment classifier, and a threshold is set for what constitutes an escalation (`sentThresh` in Algorithm 1). Sentence similarity is used to determine the number of same requests in a conversation or if a clarify section is present (lines 2,7,19,23). The similarity threshold is tunable to the sensitivity of the method in use (called `simThresh` in Algorithm 1).

Two methods for measuring similarity were compared: cosine similarity and Elasticsearch. A simple implementation of cosine similarity, which ranges from 0 (least similar) to 1 (identical), was used to measure surface similarity without considering semantics. We also experimented with Elasticsearch where every user turn was stored in an index along with the conversation ID and the order it appeared in. To measure the similarity between two turns, A and B, we queried the index using the text of A, the conversation ID of B, and the order ID of B. This results in a single match, the turn B, along with a relevance score. We treated relevance as a measure of similarity. The relevance score was calculated by the *practical scoring function* within Lucene, the underlying engine used by Elasticsearch. A relevance score can be 0 (no similarity) or any positive number. The greater the value, the more similar. Score thresholds are very specific to the query structure, and data and must be optimized appropriately. We did not observe a statistically significant difference between Elasticsearch and cosine similarity performance. As cosine similarity is both faster and less complex, we choose it as the similarity function in our algorithm.

Determination of Optimal Conversation Features

In order to determine the optimal values for the four thresholds in Algorithm 1, we perform a grid search on the training dataset. We set the values of the thresholds to be all combinations of the following: `sentThresh` $\in \{-0.3, -0.4, \ldots, -1\}$, `simThresh` $\in \{0.4, 0.5, \ldots, 1\}$, `rptThresh` $\in \{1, 2, \ldots, 6\}$, and `idkThresh` $\in \{1, 2, \ldots, 6\}$. Running the EA against the $6,203$ conversa-

Model	$\mathbf{P}_E$	$\mathbf{R}_E$	$\mathbf{F1}_E$	$\mathbf{P}_{NE}$	$\mathbf{R}_{NE}$	$\mathbf{F1}_{NE}$
EA	**.876**	.647	.744	.938	.983	**.960**
CNN	.776	**.718**	**.746**	**.942**	.957	.950
JRip	.387	.051	.090	.834	.983	.902
SVM	.317	.562	.405	.892	.750	.815
RF	.735	.036	.069	.833	**.997**	.908

Table 3: Precision (P), recall (R), and the F1 score (F1) for all models. A subscript of E indicates a metric for the escalation class whereas NE represents no escalation.

% Training Data	EA Time (s)	CNN Time (s)	Increase
1	11	70	6.36x
5	94	339	3.61x
10	139	619	4.45x
25	361	1,466	4.06x
50	672	2,881	4.23x
100	1,375	6,469	4.71x

Table 4: Time needed to either tune the thresholds for EA or train the CNN given a percentage of the training set.

tions in the training set, we measure the precision, recall, and F-1 score of each combination.

To determine the optimal values over the $2,016$ combinations we first rounded the F-1 scores of the results to the nearest hundredth. As many combinations of precision and recall can create similar F-1 scores, we simply selected the values with the highest precision (.883) in the top F-1 score (.75). As previously stated, precision is by far more important than recall as false positives are confusing to users and expensive for the companies. False negatives, on the other hand, are less essential as they present no difference to the user experience with the addition of our system. The optimal values selected were $\texttt{sentThresh} = -0.7$, $\texttt{simThresh} = 0.4$, $\texttt{rptThresh} = 4$, and $\texttt{idkThresh} = 4$. These are the threshold values used to measure EA performance on the test set in Table 3.

Results

Precision and recall for both classes (E and NE) are displayed for all models in Table 3. Although CNN performance may increase if given more training data than we collected, it is important to realize that our EA has two advantages over the CNN. One, the EA needs considerably less training data for threshold tuning. To discover this, we took random samples from the training data and used them to both train a CNN and tune the EA thresholds. We then measured their performance on the test set (Figure 2). The EA precision remains stable as the training set decreases until more than 98% of training data is removed, whereas the CNN drops steadily until around 9% where performance becomes unpredictable. This also demonstrates that EA threshold tuning is not sensitive to overfitting. By 4% of training data (248 conversations), EA precision is still identical to that of 100% of training data! Two, the EA requires less time for threshold tuning than CNN training time (Table 4). Times were generated from training the CNN and tuning the EA on a server with 48 2.2GHz cores and 64 GB of RAM.

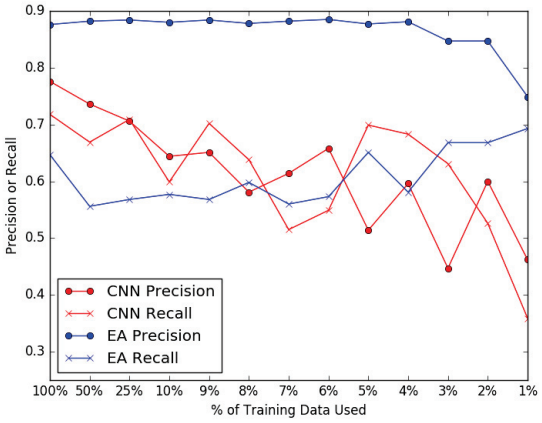

Figure 2: Precision and recall for CNN and EA on the test set given a random sample of the training set.

A 4.5x average increase may not seem significant until one considers that this step will be required for every IVA deployed, and periodic retraining/retuning may be needed.

We include the alignment graph for the EA in Figure 1. The EA appears to be somewhat fair to both the customer and company, although with a skew value of -1.52, there is a slight right skew, showing favor to the company.

Discussion and Future Work

Some limitations of our work and different methods to detect features were considered, and a few are worthy of discussion.

For sentiment polarity, a Naive Bayes classifier was trained using Twitter data from (Sanders 2011). Of the 5,513 manually tagged tweets available from (Sanders 2011), only 3,648 could still be accessed (due to deleted or banned accounts), and 80% were used as training data. Accuracy on the remaining 20% was 72%. TextBlob's sentiment classifier was used in our study instead as it demonstrated better performance in our system. However, TextBlob's sentiment classifier is pre-trained on a movie reviews corpus which may differ significantly from automated assistant chats. Alternative data for training a sentiment classifier could be addressed in future work as well as improved methods for sentiment analysis ((Ding, Liu, and Yu 2008), for example).

In our algorithm, polarity is determined on a by-turn basis. It would be interesting to split a turn into emotionally homogenous parts and assign a sequence of emotions to a turn like in (Roy et al. 2016), as we have observed that many longer turns are multipart with respect to sentiment. Also in (Roy et al. 2016), we could potentially add to our algorithm by determining common conversational structure sequences for sessions that do not escalate and detect deviations from this order.

There are limitations to using regular expressions (REs) to detect correctional language; variation in word ordering or spelling is difficult to account for. More sophisticated methods for this detection such as tagging data for correctional

language and training a classifier for this task could be addressed in future work. However, the check for correctional language can be a considered a minor optimization. Care was taken to only construct REs that did not include any IVA or domain specific language to maximize re-usability. We disabled lines 13 and 26 in Algorithm 1 so that the REs are not even used. We measured no loss of precision and only a 4% drop in recall on the test set from this removal. As this is the only component of the EA that may require manual analysis for a new language domain, it is important to note it has a minor role.

Finally, both IVAs considered in this paper are in the domain of travel. We would like to extend our study to include other language domains and then compare it to our results.

Our algorithm features many strengths. As clearly displayed in Figure 2 and Table 4, significantly less training data and time is needed to achieve precision and recall comparable to that of standard machine learning techniques. We observed that reviewers tagged 5.1 conversations per minute on average. The EA may require only a few hours of human time in tagging to tune for each IVA. The CNN method may require over 20 times more tagged data and, therefore, carries a much higher human cost. With significantly lower tuning time for the EA, it is conceivable that humans could tag a number of conversations, re-tune the weights, and deploy them within a single day if needed. Also, our system only relies on conversational structure and text features which will be present in both speech and text based agents, and multimodal agents. Regardless of which channel the human is using, it must be in text form by the time it reaches the NLU where our system would receive its input. Even in simple IR tasks such as a request for a specific document, we observed users restating their request if they were misunderstood and using correctional language.

Conclusion

Our results show that it is possible to achieve high precision (.876) in the recommendation of escalations without access to acoustic features. Whereas the techniques in (Gorin, Riccardi, and Wright 1997; Walker et al. 2000) are applicable to only systems that have access to acoustic features, our methodology can apply to systems with and without such features. For such a subjective task as determining when a conversation should escalate (recall we obtained a Cohen's κ of .6), our escalation algorithm performs better than several standard machine learning techniques *and* requires considerably less data and time for parameter tuning.

Many of the reviewed papers on alternative systems report only accuracy measurements instead of precision and recall. As the accuracy in our system was 93.1%, it is comparable to the accuracies reported in systems relying on acoustic features (55 to 93%). As our method only considers conversational structure and text features, it is less restrictive and can even work in multi-modal environments where turns can alternate between text and speech. Avenues for future research remain such as determining more sophisticated methods for detecting correctional language and training a sentiment classifier more suited to customer requests. This paper has, however, presented an accurate and effective algorithm for escalation recommendation that can be broadly applied.

References

Beaver, I., and Freeman, C. 2016. Detection of user escalation in human-computer interactions. In *INTERSPEECH*, 2075–2079.

Ding, X.; Liu, B.; and Yu, P. S. 2008. A holistic lexicon-based approach to opinion mining. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, 231–240. ACM.

Gorin, A. L.; Riccardi, G.; and Wright, J. H. 1997. How may I help you? *Speech communication* 23(1):113–127.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The weka data mining software: an update. *ACM SIGKDD explorations newsletter* 11(1):10–18.

Horvitz, E., and Paek, T. 2003. Utility-directed coupling of spoken dialog systems and human operators for call routing. Technical report, MSR Technical Report 2003.

Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Krahmer, E.; Swerts, M.; Theune, M.; and Weegels, M. 1999. Problem spotting in human-machine interaction.

Langkilde, I.; Walker, M.; Wright, J.; Gorin, A.; and Litman, D. 1999. Automatic prediction of problematic human-computer dialogues in How May I Help You. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding, ASRUU99*, 1–4. Citeseer.

Lison, P. 2013. *Structured Probabilistic Modelling for Dialogue Management*. Ph.D. Dissertation, University of Oslo.

Meena, R.; Skantze, J. L. G.; and Gustafson, J. 2015. Automatic detection of miscommunication in spoken dialogue systems. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 354.

Rakhlin, A. 2016. Cnn for sentence classification in keras. https://github.com/alexander-rakhlin/CNN-for-Sentence-Classification-in-Keras.

Roy, S.; Mariappan, R.; Dandapat, S.; Srivastava, S.; Galhotra, S.; and Peddamuthu, B. 2016. Qa rt: A system for real-time holistic quality assurance for contact center dialogues. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Sanders, N. 2011. Twitter sentiment corpus. http://www.sananalytics.com/lab/twitter-sentiment/. accessed 2016-05-19.

Schmitt, A.; Scholz, M.; Minker, W.; Liscombe, J.; and Sündermann, D. 2010. Is it possible to predict task completion in automated troubleshooters?. In *INTERSPEECH*, 94–97.

Walker, M.; Langkilde, I.; Wright, J.; Gorin, A.; and Litman, D. 2000. Learning to predict problematic situations in a spoken dialogue system: experiments with how may i help you? In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, 210–217. Association for Computational Linguistics.