# How Generic Can Dialogue Breakdown Detection Be? The KTH entry to DBDC3

*José Lopes*

KTH Royal Institute of Technology, Sweden

jdlopes@kth.se

## Abstract

This paper describes the KTH entry to the dialogue breakdown challenge. Our motivation to participate was to study whether breakdown detection could be generalized between task and non-task oriented dialogues. Our first approach used a reduced feature set that we developed for task-oriented breakdown detection and SVMs to model the task. The second approach used embeddings to represent the utterances and Recurrent Neural Networks (RNN) to model the task. The performance of the RNN-based system was in general better, although the first approach achieved higher recall and f-measure for breakdowns. Regarding the comparison between systems that the model the task as an RNN, the choice of embedding used to represent the turn revealed to be decisive in the performance of the system. The system submitted that used pre-trained word embeddings, was the system which achieved the best accuracy among those that participated in the challenge.

**Index Terms**: breakdown detection, dialogue systems, human-computer interaction

## 1. Introduction

When having a dialogue, we often experience situations where the conversation cannot easily proceed. This was how Martinovsky and Traum defined *breakdown* in the context of human machine interaction [1]. Detecting when these breakdowns were caused by the system behavior is known problem in the spoken dialogue systems research, and has been studied by several researchers in the context of task oriented systems [2, 3, 4, 5, 6, 7]. The problem proposed proposed in the 3rd Dialogue Breakdown Detection Challenge [8] is to detect these breakdown points in chat-based dialogue systems which are not task oriented. Since the domain of the dialogues is not limited, it is reasonably to expect that the difficulty of the task should increase, compared to breakdown detection in task-oriented dialogue systems.

The organizers of the challenge have gathered data from four different chatbots: TickTock [9], IRIS [10], CIC the chatbot used in the Deep Hack Turing school-hackathon and YI a chatbot developed at the Moscow Institute of Physics and Technology. 100 dialogues from each chatbot were available for development purposes. Each of the dialogues used contained 10 system responses. Each of these was annotated with one of the following three labels: Not a breakdown (NB) if it was possible to continue the conversation, Possible Breakdown (PB) if was difficult to continue the conversation smoothly and Breakdown (B) if it is difficult to continue the conversation. The annotation task was done on crowdsourcing platforms. Every system response was annotated by 30 different crowd-workers. The challenge task is to predict the most likely label and the label distribution in a set of unlabeled 50 dialogues from each chat-based system provided by the organizers. The ground truth for the most likely label was done by getting the majority vote from the annotators. Table 1 shows an example dialogue from the TickTock data.

| | Utterances | Annotation |
|---|---|---|
| 1 | S: I felt I was 17 again. | NB |
| 2 | U: Why's that? | |
| 3 | S: Do you want to go to my friend's party tonight? | PB |
| 4 | U: Where are you from? | |
| 5 | S: That's a lot to assume. Any thoughts? | B |
| 6 | U: Not right now. | |

Table 1: *Example of annotation in a dialogue extracted from the TickTock data.*

The challenge organizers provided different metrics which should be used for the evaluation purposes. The classification related metrics used were accuracy, precision, recall and f-measure. These were computed both for the classification of turns labeled as B and treating PB and B as a single label (PB+B). The distribution related metrics defined were the Jensen-Shannon Divergence (JSD) and the Mean Square Error (MSE). These compare the distribution of the labels predicted, with the annotation distribution. Both these metrics were computed in three different ways: the distribution of all labels (NB,PB,B), treating PB and B as a single label (NB, PB+B) or treating NB and PB as a single label (NB+PB,B). A detailed description of the challenge task can be found in [8].

In essence this could be seen as the same problem of detecting breakdown communications in task-oriented dialogues. Could the methods used for task-oriented system breakdown detection work in non-task oriented dialogues as well? Is the feature set very dependent of the application? How generic are the features that we used for task-oriented breakdown detections? These are some of the questions that we address in this paper. To do so, we are going to use a subset of a feature set previously used for breakdown detection in task-oriented dialogues. Furthermore, we will compare the trained model with other models trained with different feature sets and using an RNN. In this case we have used Long-Short Term Memory (LSTM).

## 2. Background

In a recent European project, a research team at KTH investigated different methods to detect breakdowns in task-oriented dialogues (or miscommunications as we defined them in the original publication [11]). These methods used a feature set that was designed to fit the datasets used during the project [12]. Features tried to capture repetitions between turns, user feedback, corrections or answers to explicit confirmation requests, for instance. The main goal was to find breakdowns in the dialogues, but the approach could be comparable to similar works that aimed to improve confidence measures [2], predict the right

moment to transfer the call to an operator [7] or predict user corrections [13]. These approaches are substantially different than those developed for the previous challenges [14]. First because task-oriented is substantially different from non-task oriented dialogue. And second, all the task-oriented dialogue systems here cited were spoken dialogue systems, which increase the range of features available.

Several of the systems submitted to the first breakdown challenge use neural networks to model the task, utterances are represented in terms of word frequencies, bag-of-words or sentence embeddings. This reflects the recent advances in natural language processing, but it would be interesting to compare both approaches, discuss the advantages of each method and their use cases. This paper tries to contribute to this discussion.

## 3. Data Preparation

We developed our system using the development data provided by the challenge organizers. This data was used not only to train the breakdown detection module, but first it was used to train a document embedding. The document embedding was trained using all the utterances (8483 in total) in the development using the implementation of [15] provided in gensim [16] library. Each utterance was represented by a vector with dimensionality 100. This kind of representation would take into account the word order in the utterance.

For each of the chatbots available, we sampled the development data in order to have 5 % the dialogues for validation and the remaining 95% for training. This ensures that there is a similar distribution of different chatbots used in train, validation and unlabeled evaluation set provided later by the organizers.

## 4. Breakdown detectors

In this section we will describe the two approaches to the problem that we performed. In Section 4.1 the one based on our previous work on breakdown detection and in Section 4.2 the one using LSTMs. Both will be described in detail followed by the results achieved on the validation data and the reason behind the configurations chosen for challenge submission.

### 4.1. SpeDial feature set with SVM

Our first attempt was to use the system that was developed in the SpeDial project [12] for call center dialogue systems. To extract features we consider snippets of four turns, two system turns and two user turns. The snippet starts on a system turn and ends on user turn. For instance, to be able to predict the correct label for turn 5 in the dialogue in Table 1, we will include features from turns 3 to 6.

The complete set of features is in the SpeDial project can be found in [11]. However, most of the features used rely on dialogue acts, which were not available in DBDC data. Therefore, we limit ourselves to a subset of features that we could extract. For instance, the set of features for turn 5, includes the cosine distance between the turn 3 and the previous system (1) and user turn (2), the cosine distance between user turn 4 and previous system (1) and user turn (2), the cosine distance between turn 5 and turns 3 and 4, and the cosine distance between turn 6 and turns 2 and 4. In addition, for all the pairs used to compute the cosine distance, we also compute the number of overlapped words between those utterances. Finally, we add the number of words in utterance 5 and the turn number relative to the total number of turns in the dialogue.

The sample SpeDial feature set had in total 21 features (**SPD-BoW**). In the original approach, the cosine distance between two utterances was computed representing the utterances as a Bag-of-Words. Besides this we extracted features in three other different ways. **SPD-WrdEmb** and **SPD-DocEmb** use the same set of features, but instead of representing the utterance as a bags-of-words, we represent them as embeddings. For **SPD-WrdEmb** we use the procedure described in [17], where the utterance embedding is given by the average sum of the word embeddings in the utterance. We computed the word embeddings using Google News pre-trained embeddings [1]. For **SPD-DocEmb**, we used the utterance embedding trained with the development data. Finally, the last feature set contains all the features from **SPD-DocEmb**, plus the Bayesian Surprise [18] for all utterances in the four turn snippet. Bayesian Surprise is a method that has been successfully applied to summarization tasks. The intuition behind it is that given a background knowledge collected from a dataset, the most surprising sentences relative to that background knowledge should be included in the summary. From observation of the data provided, it seemed that breakdowns occur when the utterances have very little connection with past utterances in the dialogue, and therefore we expected that they would have a high surprise value. This last feature set has 25 features, whereas the others have 21.

These different feature sets were computed for the dialogues in the development set. An SVM model was trained using the scikit-learn [19] implementation. We evaluated the resulting models on the validation data. The results are shown in Table 2. Since the output from the model is a label, the corresponding class gets 1.0 probability, while the others get 0.0 probability.

| Features | Accuracy | F1 (B) | F1 (PB+B) |
|---|---|---|---|
| SPD-BoW | 0.41 | 0.49 | 0.81 |
| **SPD-WrdEmb (Run 1)** | 0.41 | 0.48 | 0 .82 |
| SPD-DocEmb | 0.41 | 0.48 | 0.83 |
| SPD-DocEmb+Bayesian Surprise | 0.40 | 0.47 | 0.78 |

Table 2: *Results for the systems trained with the SpeDial feature set and extensions.*

The results show very similar performances across the different feature sets. The use of embeddings rather than bag-of-words, has smaller performance improvements. Bayesian surprise does not seem to help in this case. In the future, the computation of Bayesian Surprise could be enriched with some dialogue related features [20]. Another possible explanation for the absence of improvement using Bayesian Surprise could be that the prior Dirichlet distribution was derived from the whole dataset. Improvements could possibly be achieved if the prior distribution is computed for each dialogue instead.

In the end we have decided to submit the **SPD-WrdEmb** as one of our systems. We expect that the representation of the utterance as embeddings to be more generic than BoW.

### 4.2. LSTM-based systems

Despite results achieved with the SpeDial feature set were better than one could expect given that the features were designed for a different kind of task, we decided to approach the problem of detection breakdowns using LSTMs. As it can be understood by the way we have chosen our features in the previous approach,

[1] Google News 100B 3M words `https://github.com/3Top/word2vec-api`

there is a strong temporal dependency in the way this problem is framed.

The LSTM were implemented with Tensorflow [21] using the Keras toolkit [22]. Keras expects a tensor as input data and the target data. Here a tensor is a three-dimensional matrix. The first dimension is the number of training units, in our case the number of dialogues. The second dimension is the number of time-steps in each unit, in our case the number of turns. In this particular task all the dialogues had a fixed length of 20 turns (for the dialogues with 21 turns we have either dropped the first system turn which was not annotated or the last user turn which did not have any following system turn). The third dimension is the number of features that were used to represent each turn. The target value is, in this case, the label with the majority of the votes. A few decisions had to be made when preparing the data. Since we are detecting breakdowns on the system side, should the user turns be included or not? Or should we use only system turns in the validation set to optimize the model hyper parameters? If we end up deciding to include the user turns how should we deal with them? We performed several experiments where only used system turns as time steps both in the train and validation and other experiments where we only use system turns as time steps in the validation set. We will not report the results from these experiments here, but the general trend is that including user turns as time steps helps to improve breakdown detection. To be able to include the user turns, we have attributed the same target value to all of them. This target value was different from the three labels defined for system turns, for instance, considering the example dialogue in Table 1, turns 2, 4 and 6 with all be labeled as U (user turn). When predicting the label for a given system turn we took the label with higher probability from the labels defined for system turns. We computed the probability distributions for the system turn labels by normalizing the probabilities found by the sum of these probabilities.

As for the models trained with SVM we have explored different utterance representations: utterance embeddings using both averaged word embeddings (**WrdEmb**) and using the document embedding trained with all the development data (**DocEmb**), and representing utterances as Bag-of-Words (**BoW**) trained from the development data. We also explored combinations between these representations by concatenating the vectors resulting from the representation of each utterance with the different methods.

We have experimented different LSTM configurations in the input layer, but we kept a $0.1$ dropout and a $0.1$ recurrent dropout. The output layer was implementing using a Dense layer with softmax activation with 4 outputs corresponding to the 4 labels used (NB, PB, B and U). In the cases where there were intermediate layers they were also implemented using Dense layers, but with rectifier linear unit activation. We have chosen Adadelta as our optimizer [23]. All the models trained for 100 epochs and we selected the model that achieved the highest accuracy on the validation data to be used later in the evaluation set. Table 3 and show the results achieved in the validation set for different combinations of LSTM configurations and features, for classification-related metrics in the upper part and for distribution metrics in the lower part.

From the results shown in Tables 3, we have chosen the one that performed better on the performance metrics ($BoW + DocEmb\_BS2\_128 \times 4$, Run3) and the one with best distribution metrics ($WrdEmb\_BS32\_64 \times 4$, Run2).

### 4.3. Experiments with the evaluation data

The model used in our Run 1 system was trained using all the data provided in the development dataset, including the validation data. The models used in Runs 2 and 3 were those whose results were shown in Table3. The results on the evaluation data are shown in Table 4.

Compared to the other systems, we have achieved good results, especially concerning the metric performances. Our Run 2 system has even achieved the best performance with respect to accuracy. It is curious to observe that our Run 1, with the SpeDial feature set, achieves very reasonable performances in terms of f-measure. We were expecting this system to perform poorly on distributions metrics due the naive way we were assigning the probability distribution. The performance achieved by our Run 3 system was rather unexpected since it was the system with the best results with respect to the performance metrics. In the following section these results will be analyzed in detail.

## 5. Discussion

Overall the three systems submitted achieved, in our point of view, a satisfying performance. It is true that we are far from achieving the performance when we investigated the same problem in task-oriented dialogue, however detecting breakdowns in non-task oriented dialogue is a much more complex task. The system submitted in Run 1 utilizes a subset of features designed for breakdown detection in task oriented dialogue. This system has very high performance specially in Recall ($0.83$ for B and $0.84$ for PB+B in the evaluation set) and F-measure for PB and B labels. This result makes sense, since the cost of breakdown in task oriented dialogues is very high and therefore it is highly desirable to implement a conservative model that could detect all the breakdowns, even if it sometimes predicts breakdowns when they do not exist. The features used rely mostly on similarities between turns, but the reason why they work here might be different from the reason they are effective in task-oriented dialogues. While in task oriented dialogues having high similarity between turns could mean that the system is stuck, trying to fill one slot value, here high similarity between turns could mean that the dialogue is somehow coherent, since there are things that the system reuses from the previous user turns. So whereas in task oriented dialogues high similarity would be very likely to be a breakdown, in this non task-oriented high similarity should more likely be not a breakdown. These results show that the hypothesis of having generic features for breakdown detections is partially true.

As mentioned before, the performance of our Run 3 system was rather disappointing given that it was the system with the best performance on the validation set. On the other hand, the Run 2 system was submitted since the probability distributions generated from this model were the most similar to the original annotators distribution. But in fact, this is the system that achieved the best accuracy from those that took part in the challenge. These systems differed in a number of settings. Both had two layers, although in the first layer the Run 3 system had 128 nodes instead of 64. We have tried to run the feature set of Run 3 with 64 nodes, but the performance was worse, which can be explained by the number of features used which is higher in Run 3. Regarding the deepness of the model, we do not observe major benefits from adding extra layers. The model which used BoW+WrdEmb as features and an input layer of 128 nodes, followed by an intermediate 64 nodes layer did not beat the mod-

| Feature Set | # Features | Layers | Batch Size | Accuracy | F1 (B) | F1 (PB+B) |
|---|---|---|---|---|---|---|
| *WrdEmb (Run 2)* | *300* | *64x4* | *32* | 0.54 | 0.52 | 0.75 |
| DocEmb | 100 | 64x4 | 2 | 0.58 | 0.55 | 0.66 |
| BoW | 2993 | 64x4 | 2 | 0.57 | 0.56 | **0.79** |
| WrdEmb + DocEmb | 400 | 64x4 | 2 | 0.59 | 0.52 | 0.74 |
| BoW + WrdEmb | 3293 | 128x64x4 | 2 | 0.58 | 0.56 | **0.79** |
| *BoW + DocEmb (Run 3)* | *3093* | *128x4* | *2* | **0.6** | **0.58** | **0.79** |
| All | 3393 | 128x4 | 2 | 0.57 | 0.54 | 0.79 |

| | JSD | | | MSE | | |
|---|---|---|---|---|---|---|
| Feature Set | NB,PB,B | NB,PB+B | NB+B,B | NB,PB,B | NB,PB+B | NB+B,B |
| *WrdEmb* | **0.043** | **0.026** | **0.023** | **0.025** | **0.03** | **0.026** |
| DocEmb | 0.064 | 0.038 | 0.034 | 0.037 | 0.045 | 0.039 |
| BoW | 0.069 | 0.05 | 0.041 | 0.041 | 0.045 | 0.046 |
| WrdEmb + DocEmb | 0.19 | 0.12 | 0.11 | 0.10 | 0.12 | 0.10 |
| BoW + WrdEmb | 0.047 | 0.03 | **0.023** | 0.027 | 0.034 | **0.026** |
| *BoW + DocEmb* | 0.07 | 0.041 | 0.036 | 0.04 | 0.045 | 0.04 |
| All | 0.11 | 0.059 | 0.071 | 0.06 | 0.061 | 0.081 |

Table 3: *Results on the validation data.*

| | | F1 | | JSD | | | MSE | | |
|---|---|---|---|---|---|---|---|---|---|
| Run | Accuracy | B | PB+B | NB,PB,B | NB,PB+B | NB+B,B | NB,PB,B | NB,PB+B | NB+B,B |
| 1 | 0.338 | 0.349 | 0.842 | 0.445 | 0.238 | 0.228 | 0.227 | 0.178 | 0.173 |
| 2 | 0.442 | 0.295 | 0.744 | 0.048 | 0.027 | 0.026 | 0.028 | 0.032 | 0.029 |
| 3 | 0.422 | 0.337 | 0.759 | 0.33 | 0.19 | 0.22 | 0.17 | 0.17 | 0.2 |

Table 4: *Results on the evaluation data.*

els with a single layer before the output layer. This might be due to the amount of data used in training, which seems to be rather small to benefit from deeper architectures. The batch size also differed between these two models. In theory, if the models converged within the 100 epochs, this should not have an impact in the final performance. Later we computed the performance of another system that we did not submit, similar to the model used in Run 2, but trained with a batch size of two, on the evaluation data. This system achieved even slightly better performance metrics, $0.447$ accuracy, $0.364$ F-measure (B) and $0.761$ F-measure (PB+B). The distribution metrics from this model were similar the Run 2 system. Therefore, we hypothesize that the utterance representation made the difference. It seems that using word embeddings trained in a very large dataset and then averaging them to represent an utterance is still a better choice than using an utterance representation that is dependent on the development set. We believe that this has to do with the dataset used to train the document embedding which is rather small, which does not allow the utterance representation to scale to unseen utterances.

In the model used in Run 1 the user turns are included in the model by feature design. In Run 2 and 3 we decided to used them since the observed trend was that the performance improves if we did so. Although this was predictable, it emphasizes that taking into account the user behaviour is important when analyzing data from interactive systems. In this case, we believe that user turns might, in some cases, have provided important cues to whether the next system turn would be a breakdown or not. The way the problem was framed assumes that the system is responsible for the breakdown. While this is true for most cases, there might be some cases where the user is responsible for the breakdown and this is reflected in the user turn.

Finally, we could speculate about the behavior that the systems presented would have in a live system. The goal of the chat-based dialogue systems is to keep the conversation going for as long as possible. Therefore, it is not clear if engaging in an error recovery strategy every time there is a breakdown is a good policy. Maybe it could instead of using a more classical recovery strategy, find ways to continue the dialogue after the breakdown, resuming it from the turn before the breakdown. In such a scenario Run 1 would very likely result in a less fluid dialogue, since its behavior is prevention oriented.

## 6. Conclusions

In this paper we have presented the KTH entry to the Dialogue Breakdown Challenge. The results achieved were, from our point of view, very satisfying. One of the systems presented was even the one with the best accuracy among those submitted to the challenge. Regarding the hypothesis of a generic set of feature for breakdown detection, one could consider that it is true to a certain extent, but features have different interpretations. However, one should be careful when choosing the matter depending on what one wants to find in the data.

It also true that the performance of the system based on the SpeDial system was in general worse than the RNN-based system that we have trained, suggesting that for this task, RNN-based models might be more appropriate and require less feature engineering.

It seems that we might be very close to reach the limit using a generic set of feature, such as the ones used in Run 1. Although, we could still try to integrate the dialogue-related Bayesian Surprise features described in [20]. Thus, for future breakdown challenges it would be interesting to explore Bidirectional LSTMs to model the task, as other teams did, but using our feature representation. According to our previous finding from task oriented dialogues, knowledge from future turns could improve the system performance, and we believe that here it will also be the case. However, this approach would not be used in real-time systems.

## 7. Acknowledgements

The author would like to thank Kalin Stefanov and Johan Boye for the discussions and technical help.

# 8. References


[1] B. Martinovsky and D. Traum, "The error is the clue: Breakdown in human-machine interaction," UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY CA INST FOR CREATIVE TECHNOLOGIES, Tech. Rep., 2006.

[2] D. Bohus and A. Rudnicky, "Integrating multiple knowledge sources for utterance-level confidence annotation in the cmu communicator spoken dialog system," CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, Tech. Rep., 2002.

[3] R. Higashinaka, Y. Minami, K. Dohsaka, and T. Meguro, "Modeling user satisfaction transitions in dialogues from overall ratings," in *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, 2010, pp. 18–27.

[4] E. Krahmer, M. Swerts, M. Theune, and M. Weegels, "Error detection in spoken human-machine interaction," *International journal of speech technology*, vol. 4, no. 1, pp. 19–30, 2001.

[5] A. Schmitt, B. Schatz, and W. Minker, "Modeling and predicting quality in spoken human-computer interaction," in *Proceedings of the SIGDIAL 2011 Conference*. Association for Computational Linguistics, 2011, pp. 173–184.

[6] G. Skantze, "Exploring human error handling strategies: Implications for spoken dialogue systems," in *ISCA Tutorial and Research Workshop on Error Handling in Spoken Dialogue Systems*, 2003.

[7] M. Walker, I. Langkilde, J. Wright, A. Gorin, and D. Litman, "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*. Association for Computational Linguistics, 2000, pp. 210–217.

[8] R. Higashinaka, K. Funakoshi, M. Inaba, Y. Tsunomori, T. Takahashi, and N. Kaji, "Overview of dialogue breakdown detection challenge 3," in *Proceedings of Dialog System Technology Challenge 6 (DSTC6) Workshop*, 2017.

[9] Z. Yu, Z. Xu, A. W. Black, and A. I. Rudnicky, "Strategy and policy learning for non-task-oriented conversational systems." in *SIGDIAL Conference*, 2016, pp. 404–412.

[10] R. E. Banchs and H. Li, "Iris: a chat-oriented dialogue system based on the vector space model," in *Proceedings of the ACL 2012 System Demonstrations*. Association for Computational Linguistics, 2012, pp. 37–42.

[11] R. Meena, J. Lopes, G. Skantze, and J. Gustafson, "Automatic detection of miscommunication in spoken dialogue systems." in *SIGDIAL Conference*, 2015, pp. 354–363.

[12] J. Lopes, A. Chorianopoulou, E. Palogiannidi, H. Moniz, A. Abad, K. Louka, E. Iosif, and A. Potamianos, "The spedial datasets: datasets for spoken dialogue systems analytics." in *LREC*, 2016.

[13] M. Swerts, D. J. Litman, and J. Hirschberg, "Corrections in spoken dialogue systems." in *INTERSPEECH*, 2000, pp. 615–618.

[14] R. Higashinaka, K. Funakoshi, Y. Kobayashi, and M. Inaba, "The dialogue breakdown detection challenge: Task description, datasets, and evaluation metrics." in *LREC*, 2016.

[15] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1188–1196.

[16] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

[17] J. D. Williams, K. Asadi, and G. Zweig, "Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning," *arXiv preprint arXiv:1702.03274*, 2017.

[18] A. P. Louis, "A bayesian method to incorporate background knowledge during automatic text summarization." Association for Computational Linguistics, 2014.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[20] J. Johnson, V. Masrani, G. Carenini, and R. Ng, "Generating and evaluating summaries for partial email threads: Conversational bayesian surprise and silver standards," in *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, 2017, pp. 263–272.

[21] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[22] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[23] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.