# Automated Scoring of Chatbot Responses in Conversational Dialogue

Steven Kester Yuwono, Wu Biao, Luis Fernando D'Haro

**Abstract** Rapid advancement in natural language processing (NLP) and machine learning has led to the recent development of many chatbot systems using various algorithms. However, in a conversational dialogue setting, creating a system to communicate with humans in a meaningful and coherent manner remains a challenging task. Furthermore, it is very difficult even for humans to evaluate the responses of a chatbot system given the context of the conversation. In this paper, we will focus on the problem of automatically evaluating and scoring the quality of chatbot responses in human-chatbot dialogue settings. We propose a novel approach of combining the word representations of human and chatbot responses, and using machine learning algorithms, such as support vector machines (SVM), random forests (RF), and neural networks (NN) to learn the quality of the chatbot responses. Our experimental results show that our proposed approach is able to perform well.

## 1 Introduction

Recent advancement in NLP has greatly improved the quality of chatbot systems for conversational dialogues. As a result, many chatbot systems were developed recently [3, 31, 11]. Despite the recent advances in dialogue systems, creating a system to communicate with humans in a natural, coherent, and meaningful manner

Steven Kester Yuwono and Wu Biao
Department of Computer Science, National University of Singapore
13 Computing Drive, Singapore 117417
e-mail: sky@u.nus.edu, happygrass.wu@gmail.com

Luis Fernando D'Haro
Institute for Infocomm Research, A*STAR
1 Fusionopolis Way #21-01 Connexis South Tower - Singapore 138632
e-mail: luisdhe@i2r.a-star.edu.sg

| Instance # | Speaker | Annotated Score(s) | OPT | PES | AVE | Response |
|---|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... | ... |
| 9 | Human | - | - | - | - | Yes, but it's more of a drama than a comedy |
| | Chatbot | 1; 1; 0.5; 0 | 1 | 0 | 0.625 | I really like the first Avenger movie, have you seen it before? |
| 10 | Human | - | - | - | - | Yes, I like it. did you like the batman versus superman movie? |
| | Chatbot | 1; 0.5; 0; 0 | 1 | 0 | 0.375 | I love his black eyes. |
| ... | ... | ... | ... | ... | ... | ... |

**Table 1** Chat snippet from TickTock dataset. `OPT`, `PES`, and `AVE` denotes optimistic, pessimistic, and averaging ground truths respectively.

remains a very challenging task. The difficulty in creating a high quality dialogue system is attributed in part to the difficulties in evaluating the quality of chatbot's responses. This kind of evaluation is a challenging and subjective task, even for humans. As an evidence, the inter-annotator agreement in our dataset is relatively low between crowdsourced annotators or just fair among experts [2]. Where for the same dataset used in our experiments, [2] found that the Fishers Interclass Correlation coefficient and Cohens Kappa coefficient are 0.5454 and 0.3736 respectively that is consider fair, but that is reduced when including non-experts' annotation (0.476 for ICC and 0.317 for Kappa). In this paper, we focus on the task of automatically scoring chatbot's responses.

There are many recent studies [19, 21] which attempts to assess the quality of chatbot responses by comparing them to reference human responses. In other words, sequence to sequence comparison evaluated using BLEU and METEOR score. Our work is novel in the aspect that we are formulating our problem as a classification problem instead of a sequence to sequence problem. We believe that in a conversational dialogue, there is no single valid response but many various ways to respond that are valid in the context of the conversation. Therefore, we are not comparing the similarity of chatbot responses with their respective valid human responses but classifying chatbot responses as valid, acceptable, or invalid.

To tackle this problem, we propose an approach of using various word representations and machine learning algorithms to evaluate or score chatbot responses in a dialogue. In our SVM and random forest model, the words are represented by bag-of-words model. On the other hand, we use distributed word representations (word embeddings) with two neural network architectures, namely convolutional neural network (CNN) [18] and recurrent neural network (RNN) [12], inspired by their recent successes in multiple tasks.

RNN has proven to be successful in sequence prediction tasks such as speech recognition [13], caption generation [29], machine translation [1], and automated essay scoring [27]. On the other hand, CNN has also been very successful in image classification [6], and sequential labelling task [17].

We organize the rest of the paper as follows, we will explain the details of the chat-oriented dialogue evaluation task in the next section. In Section 3, we will elaborate more on our SVM and random forest approach, followed by the neural network approach in the next section. Our implementation and experimental details will be described in Section 5. Finally we conclude our work in the last section.

## 2 Chat-oriented Dialogue Evaluation

WOCHAT[1], a workshop on chatbots and conversational agents releases a number of datasets containing human-chatbot conversations with 624 sessions and 20,440 turns in total. However, not all of them have annotations indicating the validity of the chatbot responses. Only 303 Sessions and 5,879 turns are annotated. A session consists of multiple turns. A **turn** is defined as a continuous response by either human or chatbot which can contain one or more sentences. The number of annotated turns (5,879) stated above consists of both human and chatbot turns. However, we are only interested in the annotated chatbot turns (approximately half of the number of turns above). There are 3 classes to represent the quality of a chatbot turn: **(1) Valid; (2) Acceptable; (3) Invalid**. Annotations are done by both experts and Amazon Mechanical Turk (AMT). As mentioned in the introduction, the weakness in WOCHAT datasets is that the inter-annotator agreement is very low. The Cohen's Kappa coefficient value is 0.3736 between two expert annotators [2]. Some examples of these datasets and the setup of training instances is shown in Table 1.

The task can be divided into: (1) Ground truth construction; (2) Feature extraction; (3) Learning. Since the inter-annotator agreement is very low, it is challenging to establish the ground truth. For simplicity, we assume that *valid, acceptable,* and *invalid* labels are represented by real values: `1.0`, `0.5`, and `0.0`, respectively. We will explore the following ground truth construction methods to aggregate multiple annotations for each turn: **(1) Optimistic; (2) Pessimistic; (3) Averaging**. These methods are described next.

**Optimistic** ground truth construction is done by obtaining the maximum score if there are multiple annotators. For example, if there are three annotations with the following scores: `1`, `0`, `0`; then the ground truth will be `1`. If the maximum score is `0.5`, we will replace it with `1`. Similarly, for **Pessimistic** ground truth, the minimum score of all annotations for each turn is used as the ground truth. If the minimum score is `0.5`, we will replace it with `0`. Therefore, we can formulate this problem as a binary classification task.

**Averaging** ground truth construction works by obtaining the average of the annotation scores given by multiple annotators. For example, if there are three annotations with the following scores: `1`, `0`, `0`; then the ground truth will be `0.333`. Instead of a binary classification method, we will use regression to tackle the task in averaging setup.

[1] `http://workshop.colips.org/wochat/`

The idea behind optimistic and pessimistic ground truth construction is to decompose the regression (averaging ground truth) problem into two simpler binary classification problems. After the two binary classifiers are trained, the outputs can be combined (by voting/averaging) to solve the regression problem.

## 3 SVM and Random Forest Approach

**Word Representation: Bag-of-words** We use a bag-of-words approach to represent words for SVM and random forest classifiers. Despite not encoding the sequential information of words, bag-of-words model is simple and effective. The vocabulary is first created using the training data. Each word in the dictionary is represented by its frequency in the current turn. Out of vocabulary words are replaced by `<unknown>` token. For each pair of human-chatbot turns, the bag-of-words representations are constructed separately and then concatenated together as the input of the model.

**Support vector machine (SVM)** is a supervised learning model for classification, that performs well in text classification problems [16]. To classify data points in high dimensional space, SVM constructs hyperplanes to separate training data samples by maximizing the margin between data points of different classes. By considering the bag-of-words representation as a data point in a high-dimensional space, a kernel function represents the distance between samples. There are linear and non-linear kernels (such as sigmoid and radial basis function), and the optimization can be done efficiently using iterative methods on matrix operations. SVM can be extended to solve regression problems [10], using the same concept as the original SVM but modifying the loss function with additional constraints.

**Random forest (RF)** is another supervised learning model which performs well for text classification [30]. It is an ensemble method of aggregating the output of multiple decision trees. Bootstrap aggregating (bagging) is used to greatly reduce the variance and hence improve the performance. Given the bag-of-words representation as the input vector, the output of the random forest is simply the majority votes (of multiple decision trees) in the binary classification case, or the average of all the real-valued scores in the regression case. The objective functions used for classification and regression are entropy and mean squared error respectively.

## 4 Neural Network Approach

### *4.1 Model Architecture*

Our neural network architecture is illustrated in Figure 1. We now describe each layer in detail:

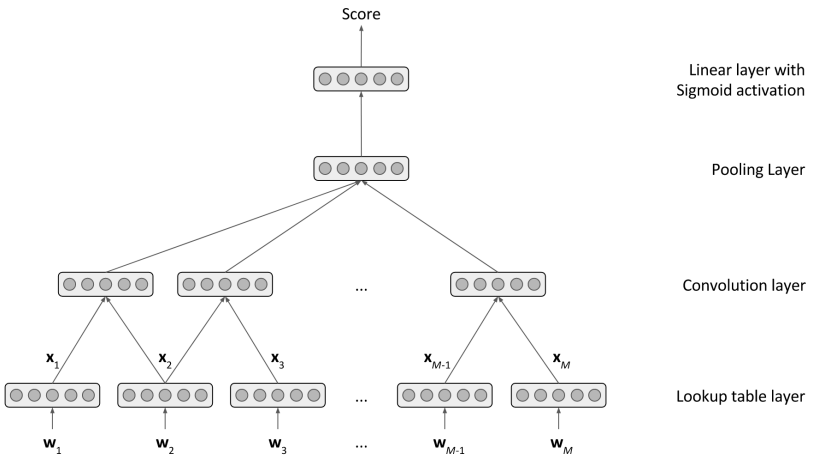


**Fig. 1** The convolutional neural network (CNN) architecture (with window size 2)

**Word Embeddings** Continuous word representations where each word is represented by a word vector (or word embedding) has been proven useful for many NLP tasks [28, 8, 4]. The word embeddings are usually used as additional features to the current machine learning model to improve its performance. Word embeddings can be obtained by training shallow networks such as continuous bag-of-words (CBOW) and skip-gram (SKIP) models [22], training deeper networks [5], or on aggregated global word-word co-occurrence statistics (GloVe) [24].

**Lookup Table Layer:** The first layer of our neural network maps each word into a vector in a $d_{LT}$-dimensional space. Given a sequence of words $\mathbf{W}$ represented by their *one-hot* representations $(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, ..., \mathbf{w}_M)$, the output of the lookup table layer is calculated by Equation 1:

$$\mathrm{LT}(\mathbf{W}) = (\mathbf{E}\mathbf{w}_1, \mathbf{E}\mathbf{w}_2, \mathbf{E}\mathbf{w}_3, ..., \mathbf{E}\mathbf{w}_M) = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_M) \tag{1}$$

where $\mathbf{E}$ is the trainable word embedding matrix and $M$ is the number of words in a turn (by either human or chatbot).

**Convolution Layer:** After the dense representation of the input sequence is computed from the lookup table layer, they are then fed as an input to the convolution layer to extract *local features*. Given a window of word representations of length $l$, (e.g, $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_l$), they are first concatenated to form vector $\bar{\mathbf{x}}$, and then calculate the output vector of length $d_c$ as shown in Equation 2:

$$\mathrm{Conv}(\bar{\mathbf{x}}) = \mathbf{W}\bar{\mathbf{x}} + \mathbf{b} \tag{2}$$

**W** and **b** are the trainable weight and bias parameters respectively, and they are shared across all windows in a sequence.

**Recurrent Layer:** Alternatively, the convolution layer above can be replaced by a recurrent layer. The dense representation of the input sequence are then fed as an input to the recurrent layer to get a representation of the whole sentence(s) instead of just extracting $l$-gram *local features*. Following [27], we use every output of the intermediate states of the RNN and perform pooling in the next layer. There are three well-known RNN units: basic recurrent units [12], gated recurrent units (GRU) [7], and long short-term memory units (LSTM) [15]. Based on our experimental results, LSTM outperforms the other two units and hence we only report results using it.

**Pooling Layer:** The output of the convolution or recurrent layer $(\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_M)$, are fed to the pooling layer. In this paper, we will be using **mean over time** (a special case of mean pooling where the window size is $M$) or **attention mechanism** (weighted average). Many studies have shown that attention mechanism is useful in many NLP tasks [1, 14, 25].

**Mean over time** is defined in Equation 3:

$$\text{MeanOverTime}(\mathbf{H}) = \frac{1}{M}\sum_{t=1}^{M}\mathbf{h}_t \tag{3}$$

and **Attention** is defined in Equation 4:

$$s_t = \mathbf{v}\cdot\tanh(\mathbf{W}\mathbf{h}_t), \quad s'_t = \text{softmax}(\mathbf{s})_t, \quad \text{Attention}(\mathbf{H}) = \sum_{t=1}^{M} s'_t\mathbf{h}_t \tag{4}$$

where **W** is a trainable matrix of size $d_c \times d_c$ and **v** is a trainable vector of size $d_c$. To learn more complex functions, **W** is introduced to increase the number of parameters and tanh is introduced to add non-linearity. **W** and **v** are *shared* across all time-steps $t$. To make sure that the weight for each time-step sums to 1, softmax function is performed on all of the weights $\mathbf{s} = (s_1, s_2, ..., s_M)$.

**Linear Layer with Sigmoid Activation:** The linear layer maps the pooled vector into a scalar value. This mapping is a simple linear transformation (fully connected layer). Since we need to predict a real number between 0 and 1, a sigmoid function is used to ensure the scalar value falls in the range of $(0,1)$. The mapping of linear layer after applying the sigmoid function is shown in Equation 5:

$$\text{s}(\mathbf{x}) = \sigma(\mathbf{w}\cdot\mathbf{x} + b) \tag{5}$$

where **x** is output of the pooling layer, **w** is the weight vector, and $b$ is the bias value.

### *4.2 Training*

In our experiment, we do not utilize early stopping methods. We train the neural network for a specified number of epochs and evaluate the model on a development set in every epoch. The epoch with the best F1-score or Pearson correlation coefficient on the development set is selected as the final model. To prevent overfitting, we have adopted dropout regularization. We also clip the gradient if gradient norm is larger than a certain threshold. We use RMSProp optimization algorithm [9] to minimize loss function over the training data. To address the problem of class imbalance, we tried a loss function scaling but it did not improve our model. For the Binary Classification we adopted **binary cross-entropy** loss function and for the regression neural network we used the **mean squared error (MSE)** loss function.

## 5 Experiments

### *5.1 Evaluation Metrics*

The standard evaluation metrics for the binary classification task are recall, precision, F1-score, and accuracy. We use F1-score and accuracy as our evaluation metric for binary classification. In the optimistic case, a positive classification refers to class `1` (valid) and negative refers to class `0` (invalid). In the pessimistic case, positive refers to class `0` (invalid) and negative refers to class `1` (valid). For the regression problem we use the Pearson's correlation coefficient.

### *5.2 Setup*

There are 11 datasets (from many different chatbots) released by WOCHAT. In our experiment, we chose the three largest datasets, namely TickTock [31], IRIS [3], and Joker [11]. We use F1-score and accuracy as our evaluation metric for binary classification, and Pearson correlation coefficient for regression. Since the dataset is small, we use 10-fold cross validation. In each fold, the data is split into 80% training, 10% development, and 10% test. For experiments using SVM and RF, the development set is ignored. One instance of training is defined as a pair of 2 turns, one by human, and another one by chatbot (the reply). The ground truth is the annotated score with respect to the chatbot turn, as we are not interested in the quality of human response. The aforementioned training instances are illustrated in Table 1. `OPT`, `PES`, and `AVE` in Table 1 denotes optimistic, pessimistic, and averaging ground truths respectively. Only the annotated turns are used in our experiments because we can only evaluate our model on instances with ground truth. The number of chatbot annotated turns and the number of *valid* and *invalid* annotations (for op-

timistic and pessimistic ground truth) are summarized in Table 2. Figure 2 shows the histogram of the averaging ground truth we used for the regression problem. We tokenized the sentences using NLTK [20], and lowercase the text.

| Dataset | Total | Ground Truth | Valid | Invalid |
|---|---|---|---|---|
| TickTock | 2731 | Optimistic | 1786 | 945 |
| | | Pessimistic | 940 | 1791 |
| IRIS | 790 | Optimistic | 661 | 129 |
| | | Pessimistic | 244 | 546 |
| Joker | 535 | Optimistic | 362 | 173 |
| | | Pessimistic | 142 | 393 |

**Table 2** The number of annotated chatbot turns for each dataset

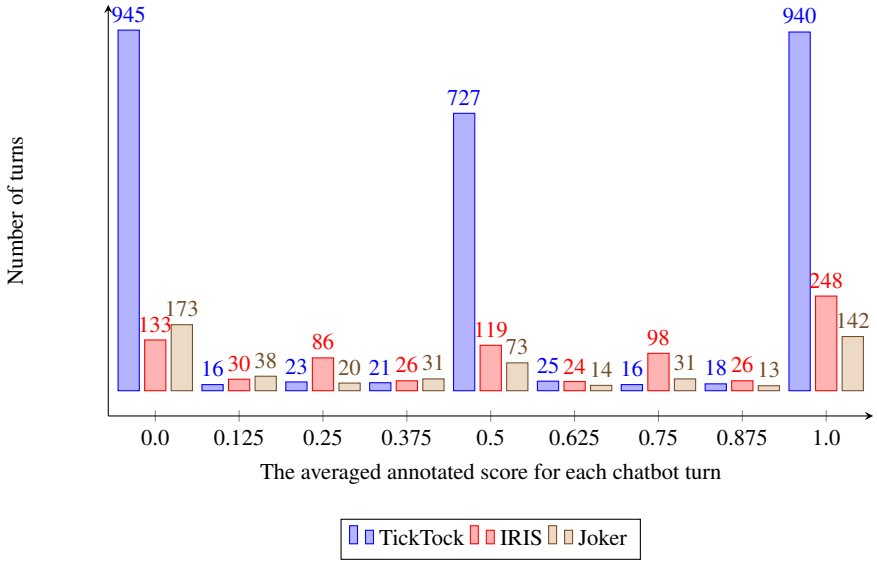


**Fig. 2** The number of annotated chatbot turns for regression problem

### 5.2.1 SVM and Random Forest

We combine each training instance consisting of two turns (one by human and one by chatbot) by concatenating their bag-of-words representations into one larger vector, and use it as the input to train the SVM and RF.

For SVM, we set the number of maximum iteration and tolerance value (for stopping criterion) to 10,000 and 0.001 respectively. Experimental results showed that more iterations did not improve the performance of the model. Different kernels such as linear, polynomial, sigmoid and radial basis function were explored. In our experiments, we set the kernel to be sigmoid as it outperforms the other kernels in this task. To address the problem of class imbalance, we tried balancing the class weights but it did not improve our model and therefore it is not reported.

For random forest, the minimum sample split is set to 2 and we do not limit the number of features in each node. We explore different number of trees and maximum tree depth. Larger number of trees can reduce variance, and limiting the depth of the trees can prevent overfitting. We limit the depth of the trees in random forest to 20. The number of trees are set to 50. The objective functions used for the classification and regression are entropy and mean squared error (MSE) respectively. Our SVM and RF models are implemented in Python using scikit-learn package [23].

### 5.2.2 Neural Networks

There are two turns (one by human and one by chatbot) in each training instance. Hence we designed our architecture, such that there is one NN for human turn and and another NN for the chatbot turn. The output vectors from the two NNs are concatenated, before going into the pooling layer.

We use RMSProp optimizer with decay rate ($\rho$) 0.9 and learning rate 0.001. Mini-batch size is 16, and we train for 30 epochs. The vocabulary is created using the most 4,000 frequent words. However, the dataset is very small such that the number of words for training is always less than 4,000 and hence all words in the training set are included in the vocabulary list. Out of vocabulary words are replaced by `<unknown>` token. The network is regularized by using dropout [26] with probability 0.8. Word embedding dimension ($d_{LT}$) is set to 100. The output dimension of the hidden layer for the RNN and the number of filters for the CNN ($d_c$) are set to 300. The convolution window size ($l$) is set to 3. We experimented by initializing the lookup table layer with various pre-trained word embeddings, but our best results were obtained using the publicly available word embedding for GloVe[2] (trained on Wikipedia and Gigaword 5, 6 billion tokens). Our neural network is implemented in Python using Keras[3] and the source code is publicly available at `https://github.com/yulonglong/ChatbotScorer`.

| Model | F1-score | | | Accuracy | | |
|---|---|---|---|---|---|---|
| | TickTock | IRIS | Joker | TickTock | IRIS | Joker |
| Baseline (Majority) | 0.790 | 0.911 | 0.804 | 0.654 | 0.837 | 0.676 |
| SVM | 0.790 | 0.911 | 0.804 | 0.654 | 0.837 | 0.676 |
| Random Forest | 0.790 | 0.910 | 0.796 | **0.660** | 0.835 | 0.665 |
| CNN MoT | 0.784 | 0.904 | 0.788 | 0.658 | 0.828 | **0.682** |
| LSTM Attention | 0.785 | 0.901 | 0.774 | 0.657 | 0.823 | 0.656 |

**Table 3** Averaged F1-score and accuracy with optimistic ground truth

| Model | F1-score | | | Accuracy | | |
|---|---|---|---|---|---|---|
| | TickTock | IRIS | Joker | TickTock | IRIS | Joker |
| Baseline (Majority) | 0.792 | 0.817 | 0.846 | 0.656 | 0.691 | 0.735 |
| SVM | 0.792 | 0.817 | 0.846 | 0.656 | 0.691 | 0.735 |
| Random Forest | **0.793** | **0.884** | 0.865 | **0.672** | **0.820** | 0.778 |
| CNN MoT | 0.783 | 0.870 | **0.866** | 0.668 | 0.802 | **0.795** |
| LSTM Attention | 0.793 | 0.874 | 0.864 | 0.670 | 0.807 | 0.785 |

**Table 4** Averaged F1-score and accuracy with pessimistic ground truth

| Model | Pearson Correlation Coefficient | | |
|---|---|---|---|
| | TickTock | IRIS | Joker |
| Baseline | $0.0024 \pm 0.061$ | $0.0014 \pm 0.104$ | $0.0033 \pm 0.138$ |
| SVM | 0.225 | 0.457 | 0.333 |
| Random Forest | **0.309** | 0.464 | **0.465** |
| CNN MoT | 0.277 | 0.481 | 0.455 |
| LSTM Attention | 0.261 | **0.505** | 0.381 |
| Voting CNN MoT | 0.269 | 0.486 | 0.449 |

**Table 5** Averaged Pearson correlation coefficient with averaging ground truth

### *5.3 Results and Discussion*

For binary classification, we adopted a simple baseline where it always predicts the majority class in the training set (Table 3 and 4).

For regression, we perform the following steps to obtain the baseline scores. First, we count the number of *valid*, *acceptable*, and *invalid* chatbot annotations in the training data. Then we use random generator that outputs either `1`, `0.5`, or `0` with probability according to their respective distribution/count. This generator is used to predict the instances in the test set and compute the Pearson correlation coefficient against the ground truth. The generator was run 1,000 times to obtain the mean and standard deviation results of the Pearson correlation (see Table 4). Running the 1000 experiments and following the probability distribution for each category is one possible way to represent the lower bound for the correlation rather than the actual performance of a specific baseline system.

[2] `https://nlp.stanford.edu/projects/glove/`

[3] `https://github.com/fchollet/keras/`

All of the results are computed by averaging the scores from 10-fold cross validation (including the baselines). For the NN experiments, we run each experiment 5 times with different seeds and average the results. The results of our best performing models are summarized in Table 3, 4 and 5. In the cases where the systems outperform the baselines, the best score for each dataset is highlighted in bold.

Observing the results in Table 3, most of the models are unable to outperform the baseline in the optimistic case. The best model, random forest, only slightly outperforms the baseline in TickTock dataset. However, the models perform much better in the pessimistic case (Table 4) where almost every model outperforms the baseline on all datasets. Moreover, the results for regression problem are also promising where almost all of the models outperform the baseline with very large margin. In addition, the combination of optimistic and pessimistic CNN performs very well. The voting is done by obtaining real-valued output of the two CNNs from both optimistic and pessimistic case, and average the two values for each test instance. Pearson correlation coefficient is then calculated using the new combined score.

Our analysis shows that the good performance of the models in the pessimistic case is because of the standard opening and closing remarks before starting a conversation. Most of the time, the words and phrases used for opening and closing conversations are highly similar, therefore the models are able to always correctly predict them as *valid*. Furthermore, the chatbot responses are most likely to be *valid* when the response is short, and the models have learnt this feature as well. Some examples are illustrated in Table 6.

| **Opening and Closing Remark** | |
|---|---|
| `Human` | : Hi |
| `Chatbot` | : Hello... USER_NAME |
| `Human` | : Hi SYSTEM_NAME |
| `Chatbot` | : Hi |
| `Human` | : Bye bye |
| `Chatbot` | : Ok, USER_NAME... bye bye. |
| `Human` | : Bye bye SYSTEM_NAME |
| `Chatbot` | : Ok, USER_NAME... good bye... |
| **Short Response** | |
| `Human` | : I will have some pizza for LUNCH |
| `Chatbot` | : Totally! |
| `Human` | : You want me to give you my number? |
| `Chatbot` | : I do. |
| `Human` | : Ok, I see, I also have my procedures |
| `Chatbot` | : Ok. |

**Table 6** Examples of turns where the models perform well in the pessimistic case

Our analysis also shows that the proportion of out-of-vocabulary words in the test set is around 10% for IRIS and Joker datasets, and 3% for TickTock dataset. The models might be able to perform better using a larger vocabulary from external corpus instead of building our own vocabulary from the training set. Overall, chat-oriented dialogue evaluation is a challenging and subjective task which requires

more insights and larger datasets to progress further. This claim is supported by the examples given in Table 1, where 3 out of 4 annotators gave different scores to the chatbot response. It shows that the validity of chatbot responses is very subjective and difficult for human to evaluate.

## 6 Conclusion

In this paper, we have addressed the task of automated scoring of chatbot response or also known as chat-oriented dialogue evaluation. We have explored three different methods to construct the ground truth from multiple annotations for each training instance, namely optimistic, pessimistic, and averaging. We formulate the problem as a simple binary classification problem and a more complex regression problem. We have explored a few machine learning models such as SVM, random forest, CNN, and RNN, and our models are shown to be able to outperform the baselines, especially in the regression problem. The combination of outputs from the two CNN models from the optimistic and pessimistic case has produced good result. Therefore, it validates the idea of decomposing the regression problem as two simpler binary classification problems (optimistic and pessimistic) and combining them at a latter stage, to solve the original regression problem.

## 7 Acknowledgement

We want to thank to Sunil Sivadas and Rafael Banchs from the Institute for Infocomm Research, A*STAR for their meaningful guidance and comments.

## References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Proceedings of the 3rd International Conference on Learning Representations (2015)
2. Banchs, R.E.: Expert-generated vs. crowd-sourced annotations for evaluating chatting sessions at the turn level. In: WOCHAT: Workshop on Chatbots and Conversational Agent Technologies (2016)
3. Banchs, R.E., Li, H.: IRIS: A chat-oriented dialogue system based on the vector space model. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics System Demonstrations (2012)
4. Bansal, M., Gimpel, K., Livescu, K.: Tailoring continuous word representations for dependency parsing. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (2014)
5. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. Journal of Machine Learning Research **3**, 1137–1155 (2003)

6. Cao, C., Liu, X., Yang, Y., Yu, Y., Wang, J., Wang, Z., Huang, Y., Wang, L., Huang, C., Xu, W., , Ramanan, D., Huang, T.S.: Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In: Proceedings of the IEEE International Conference on Computer Vision (2015)
7. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (2014)
8. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. Journal of Machine Learning Research **12**, 2493–2537 (2011)
9. Dauphin, Y.N., de Vries, H., Bengio, Y.: Equilibrated adaptive learning rates for non-convex optimization. In: Advances in Neural Information Processing Systems 28 (2015)
10. Drucker, H., Burges, C.J.C., Kaufman, L., Smola, A.J., Vapnik, V.: Support vector regression machines. In: Advances in Neural Information Processing Systems 9 (1997)
11. Duplessis, G.D., Letard, V., Ligozat, A.L., Rosset, S.: Purely corpus-based automatic conversation authoring. In: Proceedings of the 10th International Conference on Language Resources and Evaluation (2016)
12. Elman, J.L.: Finding structure in time. Cognitive Science **14**(2), 179–211 (1990)
13. Graves, A., Jaitly, N.: Towards end-to-end speech recognition with recurrent neural networks. In: Proceedings of the 31st International Conference on Machine Learning (2014)
14. Hermann, K.M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., Blunsom, P.: Teaching machines to read and comprehend. In: Advances in Neural Information Processing Systems 28 (2015)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
16. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: Proceedings of the 10th European Conference on Machine Learning (1998)
17. Kim, S., Banchs, R.E., Li, H.: Exploring convolutional and recurrent neural networks in sequential labelling for dialogue topic tracking. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (2016)
18. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Computation **1**(4), 541–551 (1989)
19. Liu, C.W., Lowe, R., Serban, I., Noseworthy, M., Charlin, L., Pineau, J.: 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 (2016)
20. Loper, E., Bird, S.: NLTK: The natural language toolkit. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics (2002)
21. Lowe, R., Noseworthy, M., Serban, I.V., Angelard-Gontier, N., Bengio, Y., Pineau, J.: 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) (2017)
22. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems 26 (2013)
23. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)

24. Pennington, J., Socher, R., Manning, C.: GloVe: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (2014)
25. Rush, A.M., Chopra, S., Weston, J.: A neural attention model for abstractive sentence summarization. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (2015)
26. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**, 1929–1958 (2014)
27. Taghipour, K., Ng, H.T.: A neural approach to automated essay scoring. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (2016)
28. Turian, J., Ratinov, L., Bengio, Y.: Word representations: A simple and general method for semi-supervised learning. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (2010)
29. Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: A neural image caption generator. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015)
30. Xu, B., Guo, X., Ye, Y., Cheng, J.: An improved random forest classifier for text categorization. Journal of Computers **7**(12), 2913–2920 (2012)
31. Yu, Z., Papangelis, A., Rudnicky, A.: TickTock: A non-goal-oriented multimodal dialog system with engagement awareness. In: Proceedings of the Association for the Advancement of Artificial Intelligence Spring Symposium (2015)