Online Learning for Conversational Agents

Vânia Mendonça✉, Francisco S. Melo, Luísa Coheur, and Alberto Sardinha

Instituto Superior Técnico and INESC-ID,
Av. Prof. Doutor Aníbal Cavaco Silva, Porto Salvo, Portugal
{vania.mendonca, luisa.coheur, jose.alberto.sardinha}@tecnico.ulisboa.pt,
fmelo@inesc-id.pt

Abstract. Agents relying on large collections of interactions face the challenge of choosing an appropriate answer from such collections. Several works address this challenge by using offline learning approaches, which do not take advantage of how user-agent conversations unfold.
In this work, we propose an alternative approach: incorporating user feedback at each interaction with the agent, in order to enhance its ability to choose an answer. We focus on the case of adjusting the weights of the features used by the agent to choose an answer, using an online learning algorithm (the Exponentially Weighted Average Forecaster) for that purpose. We validate our hypothesis with an experiment featuring a specific agent and simulating user feedback using a reference corpus. The results of our experiment suggest that the adjustment of the agent's feature weights can improve its answers, provided that an appropriate reward function is designed, as this aspect is critical in the agent's performance.

Keywords: Online learning; Exponentially Weighted Average Forecaster; Conversational agents

1 Introduction

Several agents rely on large collections of data from where to get their answers (e.g., TV drama scripts [11], Twitter interactions [5], movie scripts [4] and subtitles [1, 15]). Using such large collections makes these systems more likely to be able provide an answer to a variety of potential requests.

Given a considerably large collection of interactions (each interaction being composed of a trigger and an answer), the challenge lies in *how* to select an appropriate answer from this collection. Several works address this challenge by modelling human-agent conversations using offline learning approaches [18, 21, 22, 13]. Approaches of this sort have the downside of not taking advantage of how user-agent conversations unfold. An alternative approach would be to incorporate user feedback in an attempt to enhance the system's ability to choose an answer at each interaction, as discussed by [7]. This approach has been followed by several works in the task of choosing a dialogue strategy, achieving promising results [12, 19, 9, 17, 20, 23]. These works formulate their problems as a Markov

Decision Problem and apply reinforcement learning techniques to learn policies from user feedback (either simulated or from real users).

In this work, we focus on the following scenario: given a user request, an agent retrieves a set of candidate interactions from its source of data, and then applies a set of weighted features to those candidates in order to choose the most appropriate answer to the request. Our hypothesis is that, if these weights were iteratively adjusted considering user feedback at each interaction, the agent would be more capable of retrieving an appropriate response. In this scenario, and unlike the works mentioned above, we do not explicitly model states, as there is only one state. Moreover, there is feedback for each of the agent's possible actions. Therefore, we propose the use of an online learning algorithm suitable for this kind of scenario: the Exponentially Weighted Average Forecaster [14].

To validate this hypothesis, we devised an experiment using an existing dialogue engine, Say Something Smart (SSS) [1, 15]. Given a user request, SSS retrieves a set of candidate interactions from a corpus of interactions based on movie subtitles, and chooses the answer from the most voted interaction according to a set of weighted criteria. We compare the performance of weights iteratively adjusted based on feedback against the performance achieved by different sets of fixed weights, using a reference corpus of actual movie dialogues to simulate user feedback.

2 Related Work

In order to be more robust to a variety of user requests, several works on conversational agents and dialogue systems rely on large collections of data from where to get their answers. The system by Lasguido et al [11] uses a corpus of TV drama series and retrieves the candidate answers from the dialogues that are most similar to the input in terms of syntactic and semantic features. In Bessho et al [5], a corpus of Twitter interactions is used as the main source of candidate answers (when it cannot find a candidate within that corpus, the system delegates the request to a crowd to get a response in real time). The IRIS chatbot [4] provides answers based on a corpus of interactions extracted from movie scripts (Movie-DiC [3]). The Filipe chatbot, based on the dialogue system Say Something Smart (SSS) [2, 15], follows a similar approach, but using a corpus of movie subtitles instead (Subtle [1, 15]). Finally, the TickTock chatbot [24] incorporates crowdsourcing of the appropriateness of responses present in the corpus.

While the strategy of relying on large amounts of data allows these systems to provide diverse responses to most requests, these might not always be the most adequate. Some works deal with this challenge by modelling human-agent conversations, employing to this end diverse deep learning representations. This is the case of the works presented by Serban et al [18], in which the system learns offline to emulate the training dialogues, Xu et al [21], whose system incorporates loose-structured domain knowledge in order to capture semantic relevance between sentences in a conversation, Yao et al [22], who propose a

neural conversation model that models the intention across turns and produces specific responses from scratch, and Li et al [13], who address the issue of the consistency of the agent's responses within a dialogue.

The aforementioned works depend on previous offline training and, in some cases, they rely on domain-specific knowledge; moreover, the machine learning representations they use might be too costly to be efficiently deployed and scalable to realistic settings. As such, several works focus on the use of learning strategies that incorporate feedback, which allows to improve the systems at each user-agent interaction, as discussed by Cuayáhuitl & Dethlefs [7]. Below we present a brief description of such works.

Levin et al [12] follow an hybrid approach by combining supervised learning (to estimate the model of the user) with reinforcement learning, simulating a user interaction with the system, to learn the weights in the system's objective function (an application similar to our scenario).

Singh et al [19] take a two-step approach: first, they train the system by having it interacting with users, in order to use the resulting dialogues to build a Markov Decision Problem (MDP); then they use reinforcement learning to obtain the optimal dialogue policy for the built MDP. This policy outperformed several standard policies in the experiment reported.

Gašić et al [9] use reinforcement learning to learn a dialogue policy directly from human interaction, using a reward signal provided by users at the end of each dialogue. the authors compared this approach with the use of a random policy and an offline one in an experiment with humans. They collected the first 680 dialogues and observed that the online approach significantly outperformed the random policy and was not significantly inferior to a offline trained policy.

Pietquin et al [17] use reinforcement learning (both online and batch approaches) to learn an optimal policy as the system interacts with (simulated) users, with the online approach obtaining the best results.

Su et al [20] seek to simultaneously optimize both the dialogue policy and the reward model of their system by extracting the features of the previous dialogue turn. The reward model estimates the success of that turn based on the extracted features and, depending on the degree of uncertainty regarding that estimate, the user is queried for feedback. This system achieved approximately 91% of subjective success and an F-Score of 95% on the reward model evaluation.

In contrast to these works, Yu et al [23] address a scenario of non-task oriented conversation. They use reinforcement learning to learn a policy from user feedback, simulated using the chatbot A.L.I.C.E. in one experimental setting, and coming from actual humans in another setting. The authors compared the policy learned using reinforcement learning with both a random and a greedy policy, and observed that it outperforms those policies in all metrics of interest (turn-level appropriateness, conversational depth and information gain).

3 Learning from Feedback in a Conversational Agent Scenario

Let us recall the scenario in hands: an agent receives a user request and looks for an answer in a large collection of interactions (pairs trigger-answer). For each request, the agent obtains a set of candidate interactions from that collection and then applies a set of weighted features to those candidates in order to choose the most appropriate answer to the request.

Our goal is to accommodate feedback in the process of selecting the weights assigned to each of the features used to assess the quality of different candidate answers. Ultimately, we are interested in the feedback provided by users, which imposes two important requirements:

- The algorithm should learn *incrementally* from successive feedback, allowing the performance of the system to immediately incorporate each piece of feedback provided by users;
- The learning algorithm used should be *fast* at incorporating user feedback, since user interactions are potentially expensive.

In light of the requirements above, we adopt an online approach, choosing a standard sequential learning algorithm known as Exponentially Weighted Average Forecaster (EWAF), a generalization of the Weighted Majority algorithm of Littlestone & Warmuth [14]. EWAF precisely addresses the two requirements above: it has well-established performance guarantees, which include a bound on how fast it converges [6]. We describe this algorithm in Section 3.1 how we applied it to our scenario in Section 3.2.

3.1 The Exponentially Weighted Average Forecaster

EWAF addresses sequential prediction problems with expert advice. An *expert* is defined as a function $E : H_t \to \Delta(A)$ that, at each step t, maps the history H_t of past events (we denote by $\mathcal{H}_t$ the set of all histories of events up to time t) to a distribution $E(H_t)$ over the set of alternatives $\mathcal{A}$ (we denote by $\Delta(\mathcal{A})$ the set of all distributions over $\mathcal{A}$). A sequential prediction problem with expert advice is then an iterated game between a predictor P and "nature". At each step t, the predictor has access to a set of experts $\mathcal{E} = \left\{E^1, \ldots, E^K\right\}$ and observes the distribution over $\mathcal{A}$ proposed by each expert $E^k \in \mathcal{E}$. It then proposes a distribution $P(H_t)$ over the set $\mathcal{A}$. At the same time, "nature" selects an element $a_t \in A$. Each expert $E^k \in \mathcal{E}$ incurs a loss

$$\ell_t^k = \left|E^k(H_t) - a_t\right| \tag{1}$$

and the predictor incurs a loss

$$\ell_t^P = \left|P(H_t) - a_t\right|. \tag{2}$$

The EWAF algorithm associates a weight ω^k to each expert $E^k \in \mathcal{E}$. Then, at each step t, computes

$$P_{EWAF}(a \mid H_t) = \frac{\sum_{k=1}^{K} \omega^k E^k(a \mid H_t)}{\sum_{k=1}^{K} \omega^k}. \tag{3}$$

The weights $\omega^k, k = 1, \ldots, K$, are updated according to the loss incurred by each expert as

$$\omega_{t+1}^k = \omega_t^k e^{-\eta \ell_t^k}, \tag{4}$$

where η is a parameter of the algorithm. By setting $\eta = \sqrt{8 \log |\mathcal{E}| / T}$ it can be shown that

$$\sum_{t=1}^{T} \ell_t^P - \min_{k=1,\ldots,K} \sum_{t=1}^{T} \ell_t^k \leq \sqrt{\frac{T}{2} \log |\mathcal{E}|}, \tag{5}$$

ensuring that the predictor P quickly reaches a performance similar to that of the best expert [6].

3.2 Learning the Agent's Feature Weights

The learning process can be formalized as follows. At each step t, the user formulates a request $u(t)$ to the agent. The user request is used to retrieve a set of candidate interactions $C(t) = \{c_1(t), \ldots, c_N(t)\}$. In order to choose an answer, the agent will use K features. Each feature $f_k, k = 1, \ldots, K$ gives a score to each candidate interaction $c_n(t) \in C(t)$. The "user" is provided with the highest scored candidate according to each feature, $c(t)^k = (T^k, A^k)$, and evaluates the quality of the respective answer with a reward $r_t(c(t)^k) \in [0, 1]$. Finally, the feature weights $w_1, \ldots, w_K$ are updated as a function of the reward $r_t(c(t)^k)$.

In order to apply the EWAF algorithm to the learning of the feature weights, $w_1, \ldots, w_K$, we associate with each feature f_k an expert $E_k, k = 1, \ldots, K$ as:

$$E^k(c \mid H_t) = \begin{cases} 1 & \text{if } c = \text{argmax}_{c \in C(t)} f_k(C(t), c, u(t)). \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

In other words, each expert E_k selects the interaction $c(t)^k \in C(t)$ that maximizes the feature f_k. By setting

$$w_k = \frac{\omega^k}{\sum_{k=1}^{K} \omega^k} \tag{7}$$

the agent's selection criterion becomes that of P_{EWAF}. Finally, by setting

$$\ell_t^k = -r_t(c(t)^k) \tag{8}$$

we can use the EWAF weight update to adjust the weights w_k associated with each feature $f_1, \ldots, f_K$. In particular, we update the weights considering the sum of the rewards r_t^k received so far, $R^k(1, ..., t)$, as shown in Equation 9:

$$\omega^k(t+1) = e^{\eta R^k(1,...,t)} \tag{9}$$

We compute η according to Equation 10 (in which K is the number of experts, U is the expected number of iterations – in this case, the number of input pairs u –, and β is a configurable parameter).

$$\eta = \sqrt{\frac{\beta \log K}{U}} \tag{10}$$

4 Evaluation

To evaluate our contribution, we start by defining the following research question: *Can iteratively adjusted weights outperform fixed weights?*. As explained in Section 1, our problem differs from those addressed in existing works in that we do not explicitly model states and we have feedback for each of the agent's possible action. Therefore, we cannot directly compare our contribution with those from such works.

To address our research question, we set up an experiment in a concrete scenario, using a reference corpus to simulate user input and feedback. We describe our approach to account for user feedback in Section 4.1; we present our choice for the scenario and reward function in Section 4.2; we describe the procedure followed in our experiment in Section 4.3, and then we present and discuss the results obtained in Section 4.4.

4.1 Simulating User Feedback

In a first approach to validate the proposed learning approach, we use a reference corpus to simulate the user feedback. As such, at each step t, a "user interaction" $u(t) = (T_{u(t)}, A_{u(t)})$ is selected from the reference corpus. The trigger $T_{u(t)}$ is presented to the agent as being a user request. The agent proceeds as usual, retrieving a set $C(t)$ of candidate interactions from its collection of interactions, and each expert E^k scores the different candidate interactions in $C(t)$ as described in the previous section. Then, for each expert, a user reward r_t^k is computed, using as reference the answer $A_{u(t)}$ from the user interaction $u(t)$, so that it measures how well the candidate answer A^k that received the highest score by E^k matches the reference answer, $A_{u(t)}$. The weights ω^k of each expert E^k are then updated considering the sum of the rewards received so far, $R^k(1, ..., t)$. In other words, we use the reference corpus to automatically compute the user feedback.

Our choice for the reference corpus was the Cornell Movie-Dialogs (CMD) corpus[1], which contains over $80,000$ conversations from different sources [8], all

[1] This corpus is available in `http://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html` (last accessed in 07/21/2016).

```
SubId - 100679
DialogId - 10
Diff - 20927
T - Did you see that?
A - What is it?
```

Fig. 1. Example of an interaction in the Subtle format.

of them corresponding to actual movie conversations. Each entry in the conversations file of the CMD corpus lists all the lines that compose that conversation. For our purposes, we select only the first two lines of each interaction, acting as a pair $(trigger, answer)$.

4.2 Experimental Scenario: Say Something Smart

As our evaluation scenario, we decided to use Say Something Smart (SSS), the dialogue engine behind the open domain agent Filipe [2, 15]. Given a user request, SSS looks up for a set of answer candidates in Subtle, a corpus of interactions built from movie subtitles [1, 15], and returns the best answer according to a combination of K configurable features [15]. The interactions in Subtle were extracted from OpenSubtitles[2] and are organized as pairs of of consecutive subtitles – the first element of the pair is the trigger (`T`), and the second is the answer (`A`), as illustrated in Figure 1. The interactions are indexed using the Lucene engine [16].

For each request u, Lucene retrieves a set of up to N candidate interactions $C = \{c_1, \ldots, c_N\}$, where each interaction c_n is a trigger-answer pair, (T_n, A_n), and N is a configurable value. SSS then has each feature f_k comparing every interaction $c_n \in C$ to the user input u and scoring them. SSS outputs the answer associated with the most voted interaction c^* as the reply to the user input u.

For the purpose of our experiment, we adapted SSS in order to include a learning module that selects a pair trigger-answer $u(t) = (T_{u(t)}, A_{u(t)})$ from the reference corpus and sends the trigger $T_{u(t)}$ to Lucene, which retrieves N candidates from the collection of interactions. Each candidate is scored by each feature f_k, and then each feature is evaluated by a reward function (simulating user feedback). We define our reward function as the similarity between the answer selected by the criterion k and the reference answer $A_{u(t)}$:

$$r_t^k = \text{Jac}(A_n, A_{u(t)}). \tag{11}$$

where Jac is the Jaccard similarity measure[3]. The value obtained is rounded by α decimal places. Finally, the weights w_k associated with each feature f_k are updated as a function of the accumulated reward $R^k(1, ..., t)$, as explained in the previous section. An overview of this process is shown in Figure 2.

[2] `http://www.opensubtitles.org/` (last accessed on 07/23/2016).

[3] The Jaccard similarity coefficient measures the similarity between two sets A and B as $Jac(A, B) = \frac{|A \cap B|}{|A \cup B|}$ [10].

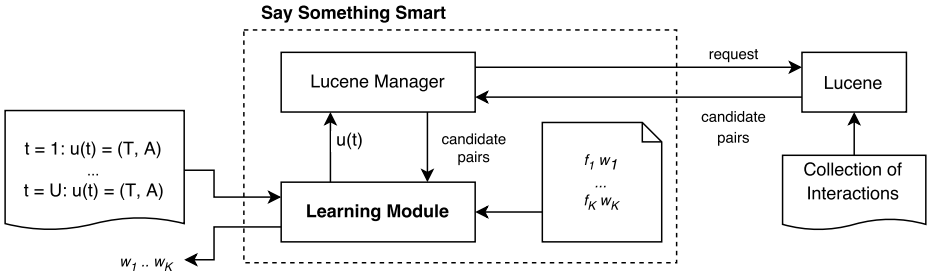

Fig. 2. SSS incorporating a learning module.

Concerning the values used in the experiment, we considered the same ones from the work of Magarreiro et al [15]: the maximum number N of candidate interactions retrieved by Lucene was set to 20, and the criteria considered were the following:

f_1: Frequency of the answer A_n in the subtitle corpus;
f_2: Similarity between the answer A_n and the user input u using the Jaccard similarity measure;
f_3: Similarity between the trigger T_n and the user input u, also using the Jaccard similarity measure;

However, we did not consider the fourth criterion reported in Magarreiro et al [15] (f_4 – Time difference between the trigger T_n and the answer A_n), as CMD corpus does not contain such information. In Magarreiro et al [15], the weight for this criterion was set to zero in the configuration that achieved the best results, therefore its removal should not impact the evaluation procedure.

4.3 Experimental Procedure

In our experiment, we aim at comparing the performance of the weights learned using the online approach described in Section 3 against sets of fixed weights: the ones reported as best by Magarreiro et al [15], and six sets of random weights.

Our simulation was designed in a cross-validation fashion with 10 folds. For each fold, the algorithm learns sets of weights w_k using different configurations of the algorithm's meta-parameters β (parameter of the weight update) and α (decimal places of the rounding made to the reward $r_t(c(t)^k)$). When learning, we used 60000 interactions from the CMD corpus (i.e., 80% of the corpus) as the collection of interactions used by SSS (having had converted them to the Subtle format), and 18000 (9×2000) other interactions as the reference corpus.

In order to assess the performance of the weights as they were iteratively adjusted, at each 300 iterations, we "froze" the weights obtained and ran SSS with them. We computed the accuracy of the system, i.e., the percentage of iterations in which SSS was able to choose the candidate answer that matched the

input reference answer. To make this possible, the remaining 2000 interactions in the corpus were used both as the reference corpus and as the collection of interactions.

4.4 Results and Discussion

We compared the performance, in terms of accuracy (%), of the following sets of weights:

- The weights iteratively adjusted using different combinations of meta-parameters: $\alpha \in \{0, 4\}$ and $\beta \in \{4, 16\}$;
- The weights reported as best by Magarreiro et al [15]: all the three features with the same weight: 33.3(3);
- Six sets of randomly generated weights:
  - $w_1 = 64$, $w_2 = 22$, $w_3 = 14$
  - $w_1 = 48$, $w_2 = 45$, $w_3 = 7$
  - $w_1 = 43$, $w_2 = 0$, $w_3 = 57$
  - $w_1 = 12$, $w_2 = 81$, $w_3 = 7$
  - $w_1 = 40$, $w_2 = 41$, $w_3 = 19$
  - $w_1 = 19$, $w_2 = 56$, $w_3 = 25$

The results of this comparison are shown in Figure 3, where each marker point over the solid lines represents the accuracy obtained by the set of weights computed at a given iteration, while the dashed line represents the accuracy of the fixed weights by Magarreiro et al [15] and the dotted line represents the accuracy of the sets of random weights. The accuracy curves represent the average accuracy across the 10 cross-validation folds. In the case of the sets of random weights, we also averaged across different sets of weights.

The weights obtained when α was set to 0 outperformed both those of Magarreiro et al [15] and random weights, converging within about 8000 iterations (and 4000 iterations in the case of $\beta = 16$), and achieving an accuracy of nearly 80%. In contrast, the weights obtained when α was set to 4 underperformed when compared to those of Magarreiro et al [15] and to random weights. These results suggest that iteratively adjusting weights might or might not improve performance, showing high sensitivity to a particular meta-parameter: α. Let us recall that α regulates the rounding of the reward value. In practical terms, $\alpha = 0$ (rounding to the unit) benefits features that receive the maximum reward (1.0) often, which is the case of f_3, and penalizes those that receive a reward between 0.0 and 0.5 more often than a reward between 0.5 and 1.0, which is the case of f_1 and f_2. On the other hand, as expected, the parameter β does not impact how good the results are, but rather how fast these are obtained, since this parameter regulates how fast the weights change. From these results, we observe that the choice of reward function (and, particularly, the numerical precision of the values that can be returned by such function) is critical to its performance.

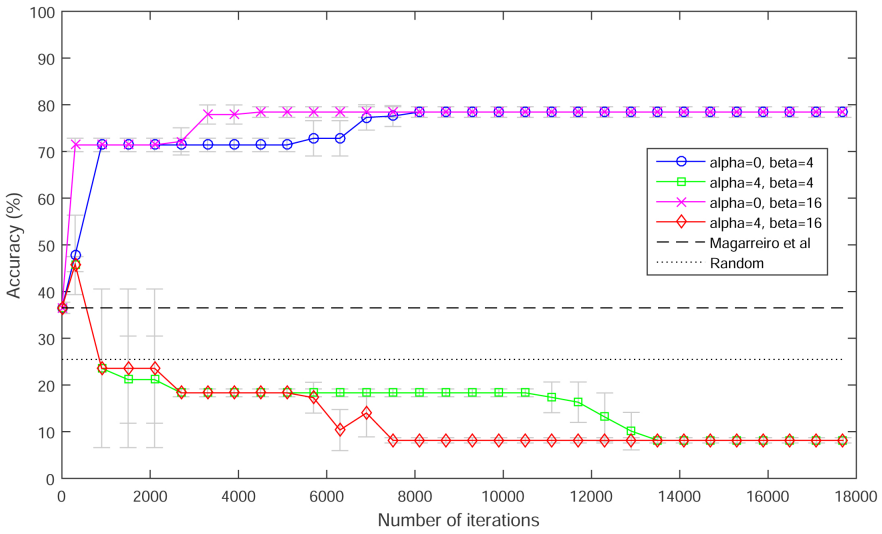

Fig. 3. Comparison in terms of accuracy between the weights iteratively adjusted using different combinations of meta-parameters α and β, the weights reported as best by Magarreiro et al [15] and random sets of weights.

5 Conclusions and Future Work

In this work, we proposed the use of an online approach to improve an agent's performance at each interaction with a user. In particular, we applied an online algorithm, the Exponentially Weighted Average Forecaster, to the problem of adjusting the weights of the features used by a given agent to choose an answer to a given request from a large collection of interactions. We devised an experiment to validate this hypothesis, in which we adapted an existing dialogue engine, Say Something Smart, in order to incorporate the Exponentially Weighted Average Forecaster, and used a reference corpus to simulate user feedback. The results achieved indicate that adjusting the feature weights based on feedback might improve the performance of the agent, but it strongly depends on the design of the feedback (reward) function. Further exploration of the current work includes experimenting different online approaches and reward functions in the process of learning weights.

Acknowledgements This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013, and under project CMUP-ERI/HCI/0051/2013. Vânia Mendonça is funded by an FCT grant with reference SFRH/BD/121443/2016.

References

1. Ameixa, D., Coheur, L.: From subtitles to human interactions: introducing the SubTle Corpus From Subtitles to Interactions- Response pairs. Tech. rep. (2013)
2. Ameixa, D., Coheur, L., Fialho, P., Quaresma, P.: Luke, I am your father: Dealing with out-of-domain requests by using movies subtitles. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8637 LNAI, 13–21 (2014)
3. Banchs, R.E.: Movie-dic: A movie dialogue corpus for research and development. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2. pp. 203–207. ACL '12, Association for Computational Linguistics, Stroudsburg, PA, USA (2012), http://dl.acm.org/citation.cfm?id=2390665.2390716
4. Banchs, R.E., Li, H.: Iris: A chat-oriented dialogue system based on the vector space model. In: Proceedings of the ACL 2012 System Demonstrations. pp. 37–42. ACL '12, Association for Computational Linguistics, Stroudsburg, PA, USA (2012), http://dl.acm.org/citation.cfm?id=2390470.2390477
5. Bessho, F., Harada, T., Kuniyoshi, Y.: Dialog system using real-time crowdsourcing and twitter large-scale corpus. In: Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue. pp. 227–231. SIGDIAL '12, Association for Computational Linguistics, Stroudsburg, PA, USA (2012), http://dl.acm.org/citation.cfm?id=2392800.2392841
6. Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning and Games. Cambridge University Press (2006)
7. Cuayáhuitl, H., Dethlefs, N.: Dialogue systems using online learning: Beyond empirical methods. In: NAACL-HLT Workshop on Future Directions and Needs in the Spoken Dialog Community: Tools and Data. pp. 7–8. SDCTD '12, Association for Computational Linguistics, Stroudsburg, PA, USA (2012), http://dl.acm.org/citation.cfm?id=2390444.2390451
8. Danescu-Niculescu-Mizil, C., Lee, L.: Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In: Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011 (2011)
9. Gašić, M., Jurčiček, F., Thomson, B., Yu, K., Young, S.: On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In: 2011 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2011, Proceedings. pp. 312–317 (2011)
10. Jaccard, P.: The distribution of the flora in the alpine zone. New Phytologist 11(2), 37–50 (1912)
11. Lasguido, Sakti, S., Neubig, G., Toda, T., Adriani, M., Nakamura, S.: Developing non-goal dialog system based on examples of drama television. In: The International Workshop on Spoken Dialog Systems (IWSDS). Paris, France (December 2012)
12. Levin, E., Pieraccini, R., Eckert, W.: A stochastic model of human-machine interaction for learning dialog strategies. In: IEEE Trans. on Speech and Audio Processing, Vol. 8 (2000)
13. Li, J., Galley, M., Brockett, C., Gao, J., Dolan, B.: A persona-based neural conversation model. CoRR abs/1603.06155 (2016), http://arxiv.org/abs/1603.06155
14. Littlestone, N., Warmuth, M.K.: The weighted majority algorithm. Inf. Comput. 108(2), 212–261 (1994), http://dx.doi.org/10.1006/inco.1994.1009

15. Magarreiro, D., Coheur, L., Melo, F.S.: Using subtitles to deal with out-of-domain interactions. In: SemDial 2014 - DialWatt (2014)
16. McCandless, M., Hatcher, E., Gospodnetic, O.: Lucene in Action, Second Edition: Covers Apache Lucene 3.0. Manning Publications Co., Greenwich, CT, USA (2010)
17. Pietquin, O., Geist, M., Chandramohan, S.: Sample Efficient On-line Learning of Optimal Dialogue Policies with Kalman Temporal Differences. In: International Joint Conference on Artificial Intelligence (IJCAI 2011). pp. 1878–1883. Barcelona, Spain (July 2011)
18. Serban, I.V., Sordoni, A., Bengio, Y., Courville, A., Pineau, J.: Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models. AAAI 2016 (Special Track on Cognitive Systems) (2015), https://arxiv.org/abs/1507.04808
19. Singh, S., Litman, D., Kearns, M., Walker, M.: Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. Journal of Artificial Intelligence Research 16, 105–133 (2002)
20. Su, P.H., Gasic, M., Mrkšić, N., Rojas Barahona, M.L., Ultes, S., Vandyke, D., Wen, T.H., Young, S.: On-line active reward learning for policy optimisation in spoken dialogue systems. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 2431–2441. Association for Computational Linguistics (2016), http://aclweb.org/anthology/P16-1230
21. Xu, Z., Liu, B., Wang, B., Sun, C., Wang, X.: Incorporating loose-structured knowledge into LSTM with recall gate for conversation modeling. CoRR abs/1605.05110 (2016), http://arxiv.org/abs/1605.05110
22. Yao, K., Peng, B., Zweig, G., Wong, K.: An attentional neural conversation model with improved specificity. CoRR abs/1606.01292 (2016), http://arxiv.org/abs/1606.01292
23. Yu, Z., Xu, Z., Black, A.W., Rudnicky, A.I.: Strategy and Policy Learning for Non-Task-Oriented Conversational Systems. In: Proceedings of the SIGDIAL 2016 Conference. pp. 404–412 (2016)
24. Yu, Z., Xu, Z., Black, A.W., Rudnicky, A.I.: Chatbot Evaluation and Database Expansion via Crowdsourcing. In: In Proceedings of the RE-WOCHAT workshop of LREC, 2016 (2016)