Converse-Et-Impera: Exploiting Deep Learning and Hierarchical Reinforcement Learning for Conversational Recommender Systems

Claudio Greco, Alessandro Suglia, Pierpaolo Basile(✉), and Giovanni Semeraro

Department of Computer Science, University of Bari Aldo Moro, Bari, Italy
claudiogaetanogreco@gmail.com, alessandro.suglia@gmail.com,
{pierpaolo.basile,giovanni.semeraro}@uniba.it

Abstract. In this paper, we propose a framework based on *Hierarchical Reinforcement Learning* for dialogue management in a *Conversational Recommender System* scenario. The framework splits the dialogue into more manageable tasks whose achievement corresponds to goals of the dialogue with the user. The framework consists of a meta-controller, which receives the user utterance and understands which goal should pursue, and a controller, which exploits a goal-specific representation to generate an answer composed by a sequence of tokens. The modules are trained using a two-stage strategy based on a preliminary *Supervised Learning* stage and a successive *Reinforcement Learning* stage.

1 Introduction

During their lives, humans need to solve tasks of varying complexity which require to satisfy different types of information needs, ranging from simple questions about common facts (whose answers can be found in encyclopedias) to more sophisticated ones in which they need to know what movie to watch during a romantic evening or what is the recipe for a good lasagne. These tasks can be solved by an intelligent agent able to answer questions formulated in a proper way, eventually considering user context and preferences.

Conversational Recommender Systems (CRS) assist online users in their information-seeking and decision making tasks by supporting an interactive process [17] which could be goal oriented. The goal to reach consists in starting general and, through a series of interaction cycles, narrowing down the user interests until the desired item is obtained [24].

Generally speaking, a dialogue can be defined as a sequence of turns, each one consisting of an action performed by a speaker or by a hearer. A CRS can be considered a goal-driven dialogue system whose main goal, due to its complexity, can be solved effectively by dividing it in simpler goals. Indeed, a dialogue with this kind of system can include different phases, such as chatting, answering questions about specific facts and providing suggestions enriched by explanations, with the aim of satisfying the user information needs during the whole dialogue. The *Hierarchical Reinforcement Learning (HRL)* literature has

F. Esposito et al. (Eds.): AI*IA 2017, LNAI 10640, pp. 372–386, 2017.
https://doi.org/10.1007/978-3-319-70169-1_28

been consistently shown that, given the right decomposition, problems can be learned and solved more efficiently, that is to say in less time and with less resources [19]. The most popular HRL framework, called the *Options* framework [29], explicitly uses subgoals to build temporal abstractions, which allow faster learning and planning. An option can be conceptualized as a sort of macro-action which includes a list of starting conditions, a policy and a termination condition.

In this work we present a CRS called *Converse-Et-Impera (CEI)* which leverages ideas presented in [12] to design a framework able to manage the different phases of a dialogue. The main contributions of our paper are the following:

1. a framework based on HRL in which each CRS goal is modeled as a *goal-specific representation module* which learns a useful representation for the given goal;
2. an *answer generation module* leveraging the learned *goal-specific representations* and the user preferences to generate appropriate answers.

2 Methodology

A dialogue can be considered as a temporal process because the assessment of how "good" an action is depends on the options and opportunities available while the dialogue progresses further. For this reason, action choice requires foresight and long-term planning to complete a satisfying dialogue for the user. We employ the mathematical framework of *Markov Decision Processes (MDP)* [2] represented by states $s \in S$, actions $a \in A$ and a transition function $T : (s, a) \rightarrow s$. An agent operating in this framework receives a state s from the environment and can take an action a, which results in a new state s'. We define the reward function as $F : S \rightarrow \mathbb{R}$. The objective of the agent is to maximize F over long periods of time.

Our work takes inspiration from [12] to design a framework able to manage the different phases of a goal-driven dialogue by learning a stochastic policy π_g which defines a probability distribution over the agent goals $g \in G$ given a state $s \in S$. The agent goals can be considered as high-level actions, thus they can be consistently modeled by the *Options Framework* [29]. In fact, the completion of a goal g can be achieved by a temporally extended course of actions, starting from a given timestep t and ending after some number of steps k. A module of the framework called *meta-controller* is responsible of selecting a goal g_t in a given state s_t. Moreover, an additional module called *controller* selects an action a_t given the state s_t and the current goal g_t following a goal-specific policy π_{ga}, which defines a probability distribution over the actions that the agent is able to execute to satisfy the goal g_t. Given the two modules, an *external critic* evaluates the reward signal f_t which the environment generates for the *meta-controller*, while an *internal critic* is responsible for evaluating whether a goal is reached and providing an appropriate reward $r_t(g)$ to the controller. So, the objective function for the meta-controller is to maximize the cumulative external reward $F_t = \sum_{t'=t}^{\infty} \gamma^{t'-t} f_{t'}$, where γ is the reward discount factor. Similarly,

the objective of the controller is to optimize the cumulative intrinsic reward $R_t(g) = \sum_{t'=t}^{\infty} \gamma_t^{t'-t} r_{t'}(g)$.

During a dialogue, in a given timestep t, the system receives a user utterance $x = \langle x_1, x_2, \ldots, x_m \rangle$ and the user identifier $u \in U$. Each word x_i is encoded in a vector representation (embedding) $\boldsymbol{x}_i$ by using a lookup operation on the word embedding matrix $\boldsymbol{W}_w \in \mathbb{R}^{|V| \times d_w}$ and the user identifier u is encoded in an embedding $\boldsymbol{u}$ by using a lookup operation on the user embedding matrix $\boldsymbol{W}_u \in \mathbb{R}^{|U| \times d_u}$, where d_w is the word embedding size and d_u is the user embedding size. The sequence of word embeddings $\boldsymbol{x}_i$ is encoded using a bidirectional *Recurrent Neural Network (RNN)* encoder with *Gated Recurrent Units (GRU)* as in [26] which represents each word x_i as the concatenation of a forward encoding $\overrightarrow{\mathbf{h}_i} \in \mathbb{R}^h$ and a backward encoding $\overleftarrow{\mathbf{h}_i} \in \mathbb{R}^h$. From now on, we denote the contextual representation for the word x_i by $\tilde{\mathbf{x}}_i \in \mathbb{R}^{2h}$. In order to allow the system to keep track of the relevant information until the dialogue turn t, we equip it with an *RNN* with *GRU* units which encodes $\tilde{\boldsymbol{x}_m}$ to generate a representation of the dialogue turn called dialogue state $\boldsymbol{d} \in \mathbb{R}^{d_d}$.

We define the *meta-controller* policy π_{MC} as a feedforward neural network which receives in input the dialogue state $\boldsymbol{d}$ to predict the goal $g \in G$. Formally, the meta-controller is defined by the following equation:

$$\pi_{MC} = P(g|x, u) = \text{softmax}(\boldsymbol{d} W_{MC}), \tag{1}$$

where softmax(y) is the *softmax* activation function applied to the vector y to obtain a probability distribution over the set of possible goals G and $W_{MC} \in \mathbb{R}^{d_d \times |G|}$ is a weight matrix.

Due to the various requirements of each goal, differently from [12], we design a *goal-specific representation module* which represents relevant aspects of the current state exploited by the agent to complete the current goal. Given the current goal g, the system encodes it in $\boldsymbol{g}$ by using a lookup operation on the goal embedding matrix $\boldsymbol{W}_g \in \mathbb{R}^{|G| \times d_g}$, where d_g is the goal embedding size. After that, the system asks the *goal-specific representation module* ϕ_g for a score vector $\boldsymbol{z} \in \mathbb{R}^{|V|}$ which represents the agent attention towards specific tokens in the vocabulary V. In the implementation of the framework we support two different goals which belong to the set G namely *chitchat* and *recommendation*. Therefore, we suppose that each conversation can be divided in turns which can be associated to one of the two available goals. For each of them, we provide a *goal-specific representation module* which generates the score vector $\boldsymbol{z}$ according to a defined strategy. The *chitchat* module is intended to support general conversation utterances (e.g., "Hey", "My name is John", etc.) so it simply returns a vector of zeros as a result of the fact that it has not focused its attention on any token. In order to complete the *recommendation* goal, we have designed a module called *IMNAMAP* presented in [9] and inspired to [26], which is able to generate a score vector by applying an attention mechanism over multiple documents retrieved from a knowledge base to uncover a possible inference chain that starts at the query and the documents and leads to the attention scores.

In order to fully support the user during the conversation, the system should be able to exploit information gathered during previous turns that can be useful in subsequent turns. For instance, during a given turn an agent may know the preferred movie director by the user and in a successive turn needs to leverage it in order to suggest relevant items to him. For this reason, the system applies an *intra-dialogue attention mechanism* to refine the score vector $\boldsymbol{z}_t$ according to $\boldsymbol{z}_k$ where $k = 1, \ldots, t$. In particular, each $\boldsymbol{z}_k$ is concatenated with $\boldsymbol{g}_t$ and $\boldsymbol{d}_t$ and given in input to a feedforward neural network in order to estimate a relevance score $r_{t,k}$. Then, a probability distribution over the score vectors is generated using $\tilde{\boldsymbol{r}}_t = \text{softmax}([r_{t,1}, \ldots, r_{t,t}])$. Finally, the refined score vector is evaluated as $\tilde{\boldsymbol{z}}_t = \sum_{k=1}^{t} \tilde{\boldsymbol{r}}_{t,k} \boldsymbol{z}_k$. In this way, we give to the agent the capability to understand the most relevant information extracted during the conversation.

The *controller* exploits the refined score vector $\tilde{\boldsymbol{z}}_t$ to generate a sequence of tokens $a = \langle a_1, \ldots, a_n \rangle$ leveraging the *Sequence-to-sequence* framework [28] (also called *Encoder-Decoder* [6]). The *Sequence-to-sequence* framework consists of two different modules: an *RNN encoder* which represents the input sequence and an *RNN decoder* able to decode the output sequence using a context vector $\boldsymbol{c}$. In this work, the context vector $\boldsymbol{c}$ is represented by the final state of the encoded user utterance $\tilde{\boldsymbol{x}_m}$. The RNN decoder generates latent representations $\langle \tilde{\boldsymbol{a}}_1, \ldots, \tilde{\boldsymbol{a}}_n \rangle$ for the system response using *Persona-based GRU* units [13] to exploit the user preferences in the text generation task. A 2-layer feedforward neural network receives in input $\tilde{\boldsymbol{a}}_i$ and $\tilde{\boldsymbol{z}}_t$ to generate a probability distribution over the tokens of the vocabulary V, for each token a_i. Formally, the feedforward neural network is defined by the following equation:

$$\tilde{\boldsymbol{a}}_i = softmax([\boldsymbol{a}_i, (\tilde{\boldsymbol{z}}_t \boldsymbol{W}_{ih})] \boldsymbol{W}_{ho}), \tag{2}$$

where $\boldsymbol{W}_{ih} \in \mathbb{R}^{|V| \times d_{ih}}$ and $\boldsymbol{W}_{ho} \in \mathbb{R}^{(h+d_{ih}) \times |V|}$ are weight matrices, d_{ih} is the input-to-hidden layer size and $[\cdot, \cdot]$ is the concatenation operator.

Inspired by [25,33], we develop a two-stage training strategy for our conversational agent composed by a preliminary *Supervised Learning (SL)* stage and a successive *Reinforcement Learning (RL)* stage. The motivation behind the adoption of the above-mentioned training strategy is to take into account historical data collected from previous interactions between a user and a system, which can be used to learn an initial effective policy for the meta-controller and the controller which can be further improved by the successive RL stage.

In the SL stage, the agent learns to replicate the conversations which belong to a given dataset by minimizing a loss function which takes into account both the meta-controller and the controller errors with regard to the training data. Given a dataset D_{MC} which consists of N dialogs, each of them composed by T turns $(x_{i,j}, u_i, g_{i,j})$, we define the supervised loss function for the meta-controller policy π_{MC} as follows:

$$L_{MC}(D_{MC}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{T} CE(g_{i,j}, \pi_{MC}(x_{i,j}, u_i)), \tag{3}$$

where CE is the cross-entropy loss function which is used in order to evaluate the error in the meta-controller prediction $\pi_{MC}(x_{i,j}, u_i)$ with regard to the target goal $g_{i,j}$. Given a dataset D_C which consists of N dialogs, each of them composed by T turns $(x_{i,j}, u_i, g_{i,j}, (a_{i,j,1}, \ldots, a_{i,j,n}))$, we define the supervised loss function for the controller policy π_C as follows:

$$L_C(D_C) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{T} \sum_{k=1}^{n} \omega(a_{i,j,k}) CE(a_{i,j,k}, \pi_C(x_{i,j}, u_i, g_{i,j})) \tag{4}$$

where CE is the cross-entropy loss function and $\omega(y_{i,k,j})$ is a function which defines a weight associated to a token $a_{i,j,k}$ equal to 2 if the position j refers to the last utterance in position $T-1$ and $a_{i,j,k} \in E$, where E is the set of entities defined in the dataset (e.g. movies, actors, ...), 1 otherwise. The function ω weights more errors done on generated suggestions. The meta-controller and the controller are jointly trained by minimizing a loss function L_a which linearly combines the two loss functions and applies L_2-regularization on it as follows:

$$L_a(D_{MC}, D_C) = L_{MC}(D_{MC}) + L_C(D_C) + \alpha L_2(\boldsymbol{W}_w, \boldsymbol{W}_u, \boldsymbol{W}_g). \tag{5}$$

A joint training has the benefit to refine the shared representations employed by the agent to represent the user utterances, the user embedding and the goal embedding in order to obtain good performance in both tasks.

Given a set of experiences D_{MC} which consists of N dialogs, each of them composed by T turns $(x_{i,j}, u_i, g_{i,j}, R_{i,j})$, we define a loss function based on *REINFORCE* [31] for the meta-controller policy π_{MC} as follows:

$$\begin{aligned} \lambda(i,j) &= \varepsilon_{MC} E(\pi_{MC}(g_{i,j}|x_{i,j}, u_i)) \\ L_{MC}(D_{MC}) &= -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{T} \log(\pi_{MC}(g_{i,j}|x_{i,j}, u_i)) \\ &\quad \cdot (R_{i,j} - b(x_{i,j}, u_i)) + \lambda(i,j) \\ &\quad + \alpha L_2(\boldsymbol{W}_w, \boldsymbol{W}_u, \boldsymbol{W}_g), \end{aligned} \tag{6}$$

where $R_{i,j}$ is the discounted cumulative reward for the current state, ε_{MC} is a weight associated to the entropy regularizer $E(\pi_{MC})$ and $b(x_{i,j}, u_i)$ is the *baseline* implemented as a feedforward neural network which estimates the expected future return from the current state received by the agent. Given a set of experiences D_C which consists of N dialogs, each of them composed by T turns $(x_{i,j}, u_i, g_{i,j}, (R_{i,j,1}, \ldots, R_{i,j,n}), (a_{i,j,1}, \ldots, a_{i,j,n}))$, we adopt the loss function proposed in [34] for the controller policy as follows:

$$\begin{aligned} \rho(i,j,k) &= \log(\pi_C(y_{i,j,k}|(y_{i,j,k-1}, \ldots, y_{i,j,1}), x_{i,j}, u_i, g_{i,j})) \\ \sigma(i,j,k) &= (R_{i,j,k} - b(y_{i,j,k}, x_{i,j}, u_i, g_{i,j})) \\ \lambda(i,j,k) &= \varepsilon_C E(\pi_C(y_{i,j,k}|(y_{i,j,k-1}, \ldots, y_{i,j,1}), x_{i,j}, u_i, g_{i,j})) \\ L_C(D_C) &= -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{T} \sum_{k=1}^{n} \rho(i,j,k)\sigma(i,j,k) + \lambda(i,j,k) \\ &\quad + \alpha L_2(\boldsymbol{W}_w, \boldsymbol{W}_u, \boldsymbol{W}_g) \end{aligned} \tag{7}$$

where $R_{i,j,k}$ is the cumulative reward for the current state, ε_C is a weight associated to the entropy regularizer $E(\pi_C)$ and $b(y_{i,j,k}, x_{i,j}, u_i, g_{i,j})$ is the *baseline* implemented as a feedforward neural network which estimates the expected future return from the current state received by the agent. The entropy regularization is considered in the RL community as another relevant optimization trick for Policy-based methods which is applied in order to prevent premature convergence to sub-optimal policy and improve exploration [32].

3 Experimental Evaluation

The experimental evaluation aim is to evaluate the performance of the developed agent in the generation of accurate suggestions and appropriate response to the user utterances. In order to achieve our objective, we defined a general procedure that can be used on classical recommender system datasets to generate a goal-oriented dialogue (Table 1) exploiting user preferences. The details of the defined procedure are reported in [27].

Table 1. Dialogue generated from the *MovieLens 1M* dataset. To help reading, *Wikidata* ids have been replaced with the corresponding entities.

Agent	Utterance	Goal
user	Hey	chitchat
bot	What is your name?	chitchat
user	I am Sofia	chitchat
bot	How old are you?	chitchat
user	I am 25	chitchat
bot	What is your occupation?	chitchat
user	sales/marketing	chitchat
bot	Which are your favourite movies?	recommendation
user	I love Speed	recommendation
bot	I suggest you Erin_Brockovich Witness Pocahontas Four_Weddings_and_a_Funeral Anna_and_the_King. Do you like them?	recommendation
user	I hate them	chitchat
bot	What are your favourite directors?	recommendation
user	I like John_Waters	recommendation
bot	What are your favourite actors?	recommendation
user	I like Josef_Sommer Wade_Williams Marg_Helgenberger Jeroen_Krabb	recommendation
bot	What genres do you like?	recommendation
user	My favourite genres are teen_film romance_film biographical_film	recommendation
bot	I suggest you Erin_Brockovich Witness Ever_After Simply_Irresistible Hairspray. Do you like them?	recommendation
user	I like them	chitchat
bot	I am glad that you like them	chitchat

We applied the dialogue generation procedure on two well-known recommender system datasets such as *Movielens 1M (ML1M)* [10] and *Movie Tweetings (MT)* [8]. For *ML1M* we leveraged the available demographic information

associated to the user in order to extend the generated conversation with useful contextual questions about the user name, age and occupation. For each dataset, we retrieved information associated to the properties *director (wdt:P57)*, *cast member (wdt:P161)* and *genre (wdt:P136)* associated to the items coming from the *Wikidata* knowledge base. The generated datasets *ML1M* and *MT* are composed of 157,135 and 48,933 dialogs whose mean lengths are 14.78 and 6.64, respectively. In addition, ML1M contains 19,039 tokens, while MT contains 53,988 tokens.

In order to assess the effectiveness of the two-stage training strategy, we designed an evaluation procedure in which we first evaluated the *CEI* model after the SL training and then we evaluated the trained model after the successive RL training to demonstrate the improvement of the model performance.

3.1 Supervised Learning Setting

In the SL stage the *CEI* model was trained on a dataset obtained by removing the utterances corresponding to the "refine" step from the generated dialogues because it should replicate the dialogues ignoring the additional "refine" steps. Intuitively, we want that the model avoids to learn from incorrect dialogue turns.

The effectiveness of the model was evaluated against some baselines such as `Random`, which is a random hierarchical agent which generates random goals and utterances composed by random tokens, and `Seq2seq`, which is an agent able to generate a response exploiting the popular *Sequence-to-sequence* framework trained through SL on the generated dialogues. The comparison with `Random` was performed in order to prove that the designed agent is able to learn to replicate the training dialogues, while the comparison with `Seq2seq` was designed to demonstrate the effectiveness of the proposed architecture with respect to a classical *Sequence-to-sequence* model which has been adopted in different conversational agents [30]. The current implementation of the `Seq2seq` model had all the weights initialized from a normal distribution $\mathcal{N}(0, 0.05)$, employed a bidirectional RNN with *GRU* units to encode the user utterance and applies a dropout of 0.5 on the RNN encoder input and output connections [35]. The word embedding size d_w was fixed to 50 and the GRU output size was fixed to 256. The model was trained using the *Adam* optimizer [11] using a learning rate of 0.001 and applying gradient clipping considering as the maximum L_2 norm value 5, as suggested in [22]. An L_2 regularization factor was applied on the word embedding matrix $\boldsymbol{W}_w$ weighted by a constant value of 0.0001. The training procedure exploited mini-batches of size 32 for both the datasets and we decided to apply an early stopping procedure considering the loss function value on the validation set. In particular, we stopped the training of the model when the validation loss function was higher than the lowest value obtained for 5 consecutive times. Otherwise, we interrupted the training at the epoch 30. This work used the official *TensorFlow* implementation of the `Seq2seq` model[1].

[1] https://www.tensorflow.org/api_docs/python/tf/contrib/seq2seq.

In order to demonstrate the appropriateness of the meta-controller and controller implementation, we evaluated both the ability of the meta-controller to select specific goals and the ability of the controller to generate personalized responses. Particularly, to assess the effectiveness of the meta-controller implementation we evaluated the *Precision* of the trained model on the test set leveraging the goal associated to each turn of the conversation. For the controller we used two different metrics to assess the effectiveness of the proposed suggestions for a specific user and the goodness of the generated system responses from a "linguistic" perspective. Specifically, we used the F_1-measure, which gives an idea of the goodness of the list of suggestions according to the ground truth suggestions present in the test set related conversation. Moreover, a *per-user* F_1-measure was evaluated considering all the unique proposed suggestions by the trained model in the test set dialogues with regard to all the positively rated items of a given user present in test conversations. The system responses have been evaluated using the *BLEU* [21] measure by comparing the generated sequences with the corresponding ones present in the test set. In particular, we exploited the *sentence-level BLEU* with smoothing function presented in [14] and whose effectiveness has been confirmed in [5].

As regards the *CEI* model parameters, we fixed the embedding size d_w, d_u e d_g to 50. In addition, due to the complexity of the model, we applied dropout of 0.5 on all the GRU cell input and output connections, a dropout of 0.2 on the *IMNAMAP* search gates and a dropout of 0.5 on all the hidden layers in fully connected neural networks employed in the architecture. The batch size was fixed to 32 and we applied the same early stopping procedure employed in the *seq2seq* model. The training procedure exploited the *Adam* optimizer with a learning rate of 0.0001 and we decided, due to the different datasets sizes, to apply a lower L_2 regularization weight α equal to 0.0001 on the *ML1M* dataset whereas we fixed α to 0.0001 on *MT*. To stabilize the training procedure and avoid that the model converges to poor local minima, we applied gradient clipping considering as the maximum L_2 norm value 5. We exploited the same procedure used in [9] to manage all the facts related to the given user query. In particular, the search engine returns at most the top 20 relevant facts for the user query. All the tokens which compose the knowledge base facts are stored in the vocabulary V as well as all the other tokens which belong to the dataset. The conversations which belong to the dataset were tokenized using the *NLTK* default tokenizer[2]. The model was implemented using the *TensorFlow* framework [1].

Firstly, we evaluated different configurations of the *CEI* model in order to find the best configuration on a specific dataset. In the evaluation, we have tried different values for a limited set of parameters while all the others are fixed to the above-mentioned values. In particular, the *GRU* output size can assume a value in the set $\{128, 256\}$, the *inference GRU* output size s in the *IMNAMAP* model can assume a value in the set $\{128, 256\}$, the output representation of the dialogue state RNN q can assume a value in the set $\{256, 512\}$ and the hidden layer size r in the controller RNN decoder can assume a value in the set $\{1024, 2048\}$.

[2] NLTK word tokenizer documentation: https://goo.gl/L4Y1Rc.

Thanks to the preliminary comparisons, we have selected the best configuration of the CEI model on the two datasets, which is the one that has the higher average score between all the evaluated measures (in the tables of results the best configurations are highlighted in bold). After that, we compared the best CEI model with the proposed baselines. Table 2 shows the experimental comparison between the best performing configuration and the other baselines according to the evaluation measures. An *NA* value in the table is associated to a configuration for which, due to its implementation constraints, it is not possible to evaluate the measure (i.e., the *seq2seq* model it is not equipped with a meta-controller so it is not possible to evaluate its effectiveness).

In the final experimental evaluation we can observe that all the values associated to the measures related to the effectiveness of the suggestions are pretty satisfying compared to the one obtained by *Random* and *Seq2seq*. In particular, on both the *MT* and the *ML1M* dataset, there is a marked difference in terms of *per-user* F_1 and F_1 measure between *CEI* and the baselines which is justified by the ability of the *intra-dialogue attention mechanism* to propagate the attention scores collected during the conversation to the controller which exploits them in the response generation procedure. It is worth noting that on the *MT* dataset the *BLEU* measure evaluated for the *Seq2seq* model is higher than the one for our model. This is a relevant factor which calls for further investigation on the real ability of the RNN decoder with *Persona-based units* to understand when to generate a token exploiting the RNN latent representation and when to use the score vector generated from the *intra-dialogue attention mechanism*.

Table 2. Evaluation between the best performing CEI model and all the baselines on MT and ML1M.

Configuration	MC precision		C BLEU		C F1		C per-user F1	
	MT	ML1M	MT	ML1M	MT	ML1M	MT	ML1M
Random	0.332	0.333	0	0.001	0	0	0	0.002
Seq2seq	NA	NA	**0.791**	0.839	0.022	0.01	0.044	0.029
CEI SL	**1.0**	**1.0**	0.784	**0.851**	**0.110**	**0.07**	**0.170**	**0.108**

3.2 Reinforcement Learning Setting

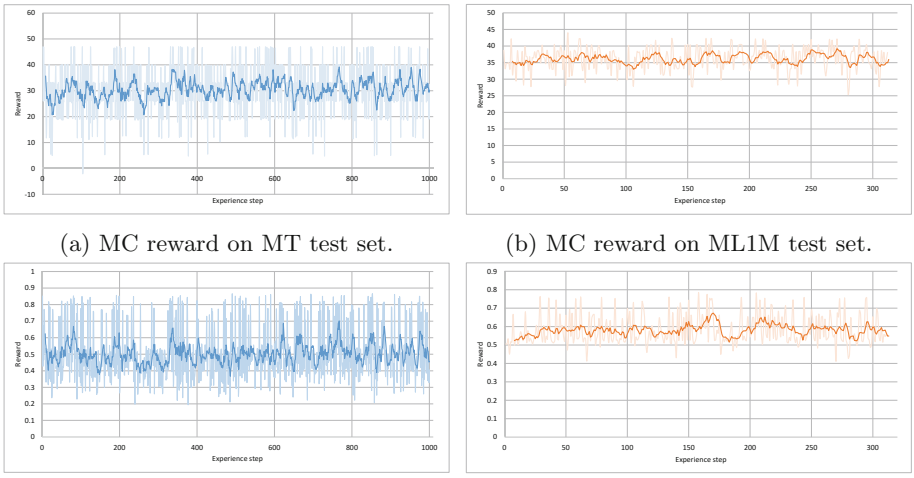
For the RL stage, a particular *OpenAI Gym* [4] environment called *Recommender System Environment (RSEnv)* is designed to let a hierarchical agent interact with a simulated user according to his/her preferences extracted from the dialogue datasets. For each user we build the user profile by composing the available information collected during the dialogue generation procedure concerning his/her name, age, occupation and preferences. For each simulated dialogue, the environment selects a random user that will interact with the bot. The environment checks the presence of particular keywords in the bot utterance and generates the appropriate answer by exploiting the user profile. When the agent generates

a sequence of tokens, it receives a reward which is evaluated by the *BLEU* metric so that, in the long-term, it should be able to understand which are the appropriate tokens that should be generated in specific states. Therefore, the agent keeps answering the user until a list of suggestions is proposed to him/her and, according to the current user preferences, a reward is generated as the mean between the F_1-measure and the *BLEU* which represents the user satisfaction for the proposed suggestions. Depending on the value of the evaluated reward signal, the user answers with a positive utterance (i.e., "I like them") or a negative utterance (i.e., "I hate them"). It is in this moment that the system receives a supplementary reward signal for the meta-controller – extrinsic reward – which is equal to 50 if the user satisfaction score is over 0.5, -5 otherwise. There are some limit cases that the environment should be able to manage in order to support an effective training procedure. In particular, if the system accidentally generates a sequence of tokens which does not contain a recognized keyword, a special tag "unknown" is used to notify to the agent that the environment is not able to process its utterance. In this case, a reward signal of -5 is returned for the meta-controller and a reward equal to 0 is returned to the controller. Another case that the environment should manage is when the agent asks the environment for information which are not present for the current user. In this case, the environment notifies the agent that it does not know the answer and returns a "nothing" message. Furthermore, in order to prevent that the agent is trapped in a loop in which the environment answers always with the same utterance, we interrupt the interaction between the agent and the environment after 10 turns. If the system is not able to complete the dialogue in a number of turns which is less than the predefined threshold value, it is penalized with a reward of -5 for the meta-controller and receives 0 for the controller.

For each dataset, we selected only the best configuration in the SL training phase for the successive RL training stage. In this way, we expect to assess, thanks to a quantitative evaluation, if the system is able to improve its performance by learning from its own experiences with users. We exploited the *REINFORCE* algorithm for the meta-controller and the controller in order to fine-tune their policies directly from the experiences collected by the agent. The model is trained using the SL training procedure is used in the RL training procedure during which it interacts with the environment by exploiting the meta-controller and the controller policy to collect experiences from which it can learn from.

The adopted training procedure starts by loading the pre-trained model obtained from the SL stage and uses it in a RL scenario so it selects actions according to a state given by the environment and observes a reward according to their goodness. The model parameters are the same as the ones used in the SL training procedure so the parameters are used as they are in the RL setting. The only difference relies in the optimization method employed and in the loss function exploited to optimize the model parameters. We run the training process in the RL scenario for a fixed number of experiences e and we used mini-batches of them to update the model parameters. The advantages of this strategy are twofold: first we are able to obtain more accurate gradient approximation and

second we are able to exploit the power of the GPU leveraged in our experiments to compute simultaneously multiple matrix operations. The batch size is fixed to 8 on the *MT* dataset and to 32 on the *ML1M* dataset. The entropy regularization weights ε_{MC} and ε_C are fixed to 0.01, the number of experiences is fixed to $10,000$ for each dataset and we apply a discount factor on the meta-controller rewards equal to 0.99. In the RL scenario we do not design a joint training procedure because the loss functions of the meta-controller and the controller are updated in different moments, as described in [12]. Moreover, they represents different behaviour strategies that cannot be easily mixed together as they are. We also changed our optimization algorithm from *Adam* to the vanilla *SGD* because we observed during our experiments with *Adam* catastrophic effects on the effectiveness of the policies due to the aggressive behaviour of the algorithm in the early stages of the training. Indeed, according to our experience, it upsets the learned policies making them completely useless for our tasks.



(a) MC reward on MT test set.

(b) MC reward on ML1M test set.

(c) C reward on MT test set.

(d) C reward on ML1M test set.

Fig. 1. Results of the RL experimental evaluation on the MT and ML1M datasets.

As described before, when the RL training stage is completed, we evaluate the performance of the learned model on a test environment which contains users data that the agent have not seen in the SL and RL training stages. The learned model resulting from the training procedure is evaluated according to a similar procedure that is used during the training stage. Particularly, we exploit the user profile information related to the users present in the test set of each dataset and we monitor how the system behaves in terms of two different measures collected during the system interaction with the environment: *MC rewards*, which is the meta-controller mean reward obtained in a mini-batch, and *C rewards*, which is the controller mean reward obtained in the mini-batch. Analyzing the graphs

Fig. 1b, d related to the *ML1M* dataset, we can observe how the meta-controller and the controller are able to interact with the environment. The performance is stable for each batch of examples underlining the capability of the model to generalize quite well over the training experiences. The same trend is observed in the charts Fig. 1a, c for the *MT* dataset which is promising because, despite the significantly different dimensions of the two datasets, the agent performance is still reliable and satisfying.

4 Related Work

In the literature CRSs have been classified under different names according to the strategy adopted in order to extract relevant information about the user and provide recommendation to him/her. *Case-based Critique* systems finds cases similar to the user profile and elicits a critique for refining the user's interests [20,23] throughout an iterative process in which the system generates recommendations in a ranked list and allows the user to critique them. The critique will force the system to re-evaluate its recommendations according to the specified constraints.

A limitation of classical CRSs is that their strategy is typically hard-coded in advance; at each stage, the system executes a fixed, pre-determined action, notwithstanding the fact that other actions could also be available for execution. This design choice negatively effects the flexibility of the system to support conversation scenarios which are not expected by the designer. In fact, despite the effectiveness of these systems in different complex scenarios, as reported in [20], they always follow a predefined sequence of actions without adapting to the user requirements. Moreover, typically they uses a representation for the items that is hand-crafted which is incredibly labour-intensive for complex domains like music, movies and travels. For instance, in the work presented in [3] is exploited a first-order logic representation of the items by an attribute closure operator able to refine the attribute set of the items considered in the current conversation.

To overcome these limitations the work first presented in [15] and then extended in [16–18], proposes a new type of CRS that by interacting with users is able to autonomously improve an initial default strategy in order to eventually learn and employ a better one applying RL techniques. It was first validated in off-line experiments to understand which were the state variables required by the system and then it was evaluated in an online setting in which the system had the task to support travellers. Another relevant work is the one presented in [7]. It exploits a *probabilistic latent factor model* able to refine the user preferences according to a fully online learning recommendation approach.

5 Conclusions and Future Work

CEI represents a framework for a *goal-oriented conversational agent* whose objective is to provide a list of suggestions according to the user preferences. Our

intuition is that a dialogue can be subdivided in fine-grained goals whose achievement allows the agent to successfully complete the conversation with the user. The framework leverages a combination of *Deep Learning* and *Hierarchical Reinforcement Learning* techniques and is trained by using a two-stage procedure.

As regards with the answer generation module we plan to understand if the current implementation is well suited to take into account the score vector weights generated by the intra-dialogue attention mechanism in order to effectively exploit them in the recommendation phase. In addition, we think that allowing the model to leverage multiple information sources can give it the possibility to grasp different views of the items that can be used to provide more accurate suggestions according to the user preferences.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al.: Tensorflow: large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467 (2016)
2. Bellman, R.: A Markovian decision process. Technical report, DTIC Document (1957)
3. Benito-Picazo, F., Enciso, M., Rossi, C., Guevara, A.: Conversational recommendation to avoid the cold-start problem. In: Proceedings of the 16th International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE 2016 (2016)
4. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI gym (2016)
5. Chen, B., Cherry, C.: A systematic comparison of smoothing techniques for sentence-level BLEU. In: ACL 2014, p. 362 (2014)
6. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
7. Christakopoulou, K., Radlinski, F., Hofmann, K.: Towards conversational recommender systems. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2016, pp. 815–824. ACM, New York (2016)
8. Dooms, S., De Pessemier, T., Martens, L.: Movietweetings: a movie rating dataset collected from Twitter. In: Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys, vol. 2013, p. 43 (2013)
9. Greco, C., Suglia, A., Basile, P., Rossiello, G., Semeraro, G.: Iterative multi-document neural attention for multiple answer prediction. In: Proceedings of the AI*IA Workshop on Deep Understanding and Reasoning: A Challenge for Next-generation Intelligent Agents 2016 co-located with 15th International Conference of the Italian Association for Artificial Intelligence (AIxIA 2016), Genova, Italy, 28 November 2016, pp. 19–29 (2016)
10. Harper, F.M., Konstan, J.A.: The movielens datasets: history and context. ACM Trans. Interact. Intell. Syst. (TiiS) **5**(4), 19 (2016)
11. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

12. Kulkarni, T.D., Narasimhan, K., Saeedi, A., Tenenbaum, J.: Hierarchical deep reinforcement learning: integrating temporal abstraction and intrinsic motivation. In: Advances in Neural Information Processing Systems, pp. 3675–3683 (2016)
13. Li, J., Galley, M., Brockett, C., Spithourakis, G.P., Gao, J., Dolan, B.: A persona-based neural conversation model. arXiv preprint arXiv:1603.06155 (2016)
14. Lin, C.Y., Och, F.J.: Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, p. 605. Association for Computational Linguistics (2004)
15. Mahmood, T., Ricci, F.: Learning and adaptivity in interactive recommender systems. In: Proceedings of the Ninth International Conference on Electronic Commerce, pp. 75–84. ACM (2007)
16. Mahmood, T., Ricci, F.: Adapting the interaction state model in conversational recommender systems. In: Proceedings of the 10th International Conference on Electronic Commerce, p. 33. ACM (2008)
17. Mahmood, T., Ricci, F.: Improving recommender systems with adaptive conversational strategies. In: Proceedings of the 20th ACM Conference on Hypertext and Hypermedia, pp. 73–82. ACM (2009)
18. Mahmood, T., Ricci, F., Venturini, A., Höpken, W.: Adaptive recommender systems for travel planning. Inf. Commun. Technol. Tour. **2008**, 1–11 (2008)
19. Maisto, D., Donnarumma, F., Pezzulo, G.: Divide et impera: subgoaling reduces the complexity of probabilistic inference and problem solving. J. R. Soc. Interface **12**(104), 20141335 (2015)
20. Mc Ginty, L., Smyth, B.: Deep dialogue vs casual conversation in recommender systems (2002)
21. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pp. 311–318. Association for Computational Linguistics (2002)
22. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: ICML, vol. 3, no. 28, pp. 1310–1318 (2013)
23. Ricci, F., Nguyen, Q.N.: Acquiring and revising preferences in a critique-based mobile recommender system. IEEE Intell. Syst. **22**(3), 22–29 (2007)
24. Rubens, N., Elahi, M., Sugiyama, M., Kaplan, D.: Active learning in recommender systems. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 809–846. Springer, Boston, MA (2015). doi:10.1007/978-1-4899-7637-6_24
25. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
26. Sordoni, A., Bachman, P., Trischler, A., Bengio, Y.: Iterative alternating neural attention for machine reading. arXiv preprint arXiv:1606.02245 (2016)
27. Suglia, A., Greco, C., Basile, P., Semeraro, G., Caputo, A.: An automatic procedure for generating datasets for conversational recommender systems. In: Proceedings of Dynamic Search for Complex Tasks - 8th International Conference of the CLEF Association, CLEF 2017, Dublin, Ireland, 11–14 September 2017 (2017)
28. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)

29. Sutton, R.S., Precup, D., Singh, S.: Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. Artif. Intell. **112**(1), 181–211 (1999)
30. Vinyals, O., Le, Q.: A neural conversational model. arXiv preprint arXiv:1506.05869 (2015)
31. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. **8**(3–4), 229–256 (1992)
32. Williams, R.J., Peng, J.: Function optimization using connectionist reinforcement learning algorithms. Conn. Sci. **3**(3), 241–268 (1991)
33. Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al.: Google's neural machine translation system: bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144 (2016)
34. Zaremba, W., Sutskever, I.: Reinforcement learning neural turing machines-revised. arXiv preprint arXiv:1505.00521 (2015)
35. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014)