A Additional Notes on Data Preparation

we obtain a rating matrix of 265,905 users and 11,382 movies. We filter the data according to a few criteria:

- users who watched less than 50 movies are filtered out.
- moves which are watched less than 50 users are filtered out.
- movies which are filmed before 1950 are filtered out.
- movies whose average rates are less than 2 and users who average rates are less than 2 are filtered out.

We also remove some movie sets which are too difficult or too easy to predict based on their distance scores. For example, we filter out movie sets where the cosine similarity of the correct movie and the averaged incorrect movies is less than 0.75. After filtering, the remaining data comprises 5,330 movies, rated by 65,181 users.

We tested different types of embedding features such as movie IDs (i.e., MovieLens's ratings), movie text (i.e., Wiki-text), and knowledge base features (e.g., director's name). The movie ID features turn out to be the best performing for recommendation performance. After training, the model finds reasonable close neighbors; for example, for "Ice Age", the model identifies "Shrek 2", "Shrek", "Monsters Inc.", and "Finding Nemo" as close.

B Data Collection: Full Description

In our annotation interface, we provide action buttons for workers to click on in order to interact with the system. When a button is clicked, the corresponding system message is shown. For example, if an expert clicks on a movie button to recommend that movie, the system displays a recommendation message to the seeker, using a simple template. Similarly, if a seeker clicks to accept or reject the recommendation, a templated message with the decision is automatically delivered to the expert.

If an expert recommends the correct movie, a seeker accepts the correctly recommended movie, or a seeker rejects an incorrectly recommended movie, they receive a reward (points, which can translate into bonus money if enough points are earned); otherwise, the system encourages them to focus more on the task and get more points. The amount of reward points awarded is calculated based on the similarities between the average of the seeker's movie set and each candidate movie in the expert's set, using a softmax. The similarity scores are calculated using the euclidean distance between movie embedding vectors (see Section C).

Overall, a total of 1,034 unique workers created 9,125 dialogues, over a duration of 2.5 weeks.

C Supervised training: Details

This section gives more details about the supervised training phase.

Encoding textual inputs: Textual inputs are encoded differently for the dialogue context and for the movie descriptions. The dialogue history context h_t for predicting utterance x_{t+1} comprises the history of all previous utterances $x_1, \cdots, x_t$. Each utterance is encoded with an LSTM (Hochreiter and Schmidhuber, 1997). The dialogue context is then obtained by averaging over all utterances:

$$h_t = \mathtt{AVG}(\mathtt{LSTM}(\mathtt{x}_1), \ldots, \mathtt{LSTM}(\mathtt{x}_t)) \tag{9}$$

For the movies, we found that using bags of words instead worked better. We encode each sentence of a movie description as a bag of words, and then average all the resulting representations to obtain m_j, the representation of the j-th movie:

$$E(\mathtt{m}_j) = \mathtt{AVG}(\mathtt{BOW}(\mathtt{m}_j)) \qquad \mathtt{for}\, j \in 1..K \tag{10}$$

Aligning dialogue context and movie descriptions: we use dot-product attention(Chen et al., 2017) between the dialogue context and each of the movie descriptions:

$$\mathtt{c}_j = h_t \cdot m_j \qquad \mathtt{for}\, j \in 1..K \tag{11}$$

Generating utterances: **Generate** The expert can produce two types of utterances, according to whether it is recommending a movie or asking for more input from the seeker. For `Recommend`, the response is produced by a template: "How about this movie, `[MOVIE]`?" where `[MOVIE]` is the movie that the expert is recommending. For `Speak`, the next utterance is generated by taking the dialogue context history h_t and the average of all movie representations $\mathtt{M} = \mathtt{AVG}(m_1, .., m_K)$, and inputting them into a seq2seq generative model with attention (Bahdanau et al., 2015). The model is then trained to minimize the negative log likelihood of the true next utterance x_{t+1} according to

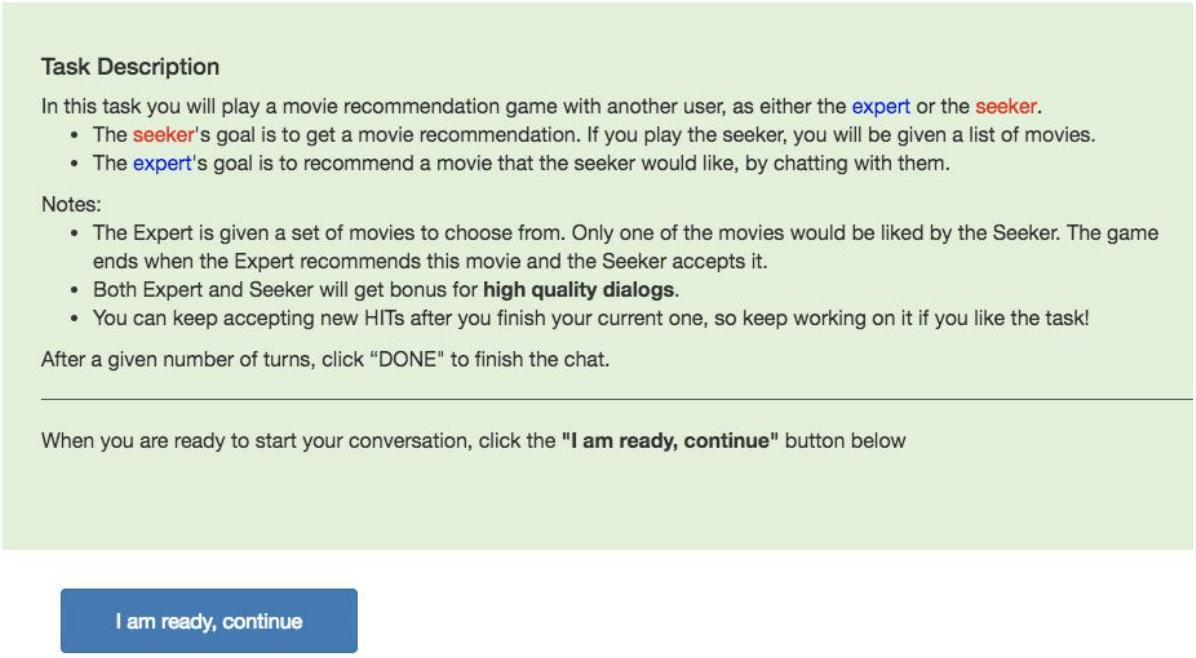


Figure 7: Interface of our data collection (1): task description page.

the model distribution p_{gen}:

$$\mathcal{L}_{gen} = -\log p_{gen}(x_{t+1}|h_t, \mathtt{M}), \textit{where} \quad (12)$$

$$\mathtt{M} = \mathtt{AVG}(m_1, .., m_K) \quad (13)$$

We *include* `Recommend` utterances in the $\mathcal{L}_{gen}$ calculation; as a result, the generation loss is also a partial indicator of other aspects such as Decide and Predict, in addition to the corresponding specific losses (see below).

Predicting the correct movie to recommend: Predict Let y denote the correct movie. The prediction module is trained by minimizing the negative log likelihood of y according to the distribution of a softmax predictor over the $\mathtt{c}_j$ inputs described above:

$$\mathcal{L}_{predict} = -\log p(y|c_1, \cdots, c_K), \quad \text{where} \quad (14)$$

$$\mathtt{c}_j = h_t \cdot m_j \qquad \text{for } j \in 1..K \quad (15)$$

When making a recommendation, the expert recommends the top candidate: $\arg\max_c\{\mathtt{r}_1..\mathtt{r}_K\}$. We also experimented with using a soft representation for the target movie distribution, for example through a softmax over similarities. For instance, in Figure 2, the hard ground-truth movie distribution is $\{1, 0, 0, 0, 0\}$, and the soft version is $\{0.37, 0.15, 0.16, 0.16, 0.15\}$. But the hard version always outperformed the soft version in our experiments.

Deciding when to recommend: Decide The expert needs to decide whether to to recommend a movie or speak to elicit more information. We model this using a two-layer perceptron that takes the movie prediction distribution scores and the dialogue context as input, and predicts whether to make a recommendation or not. Training is conducted by minimizing the negative log likelihood of the ground truth decision:

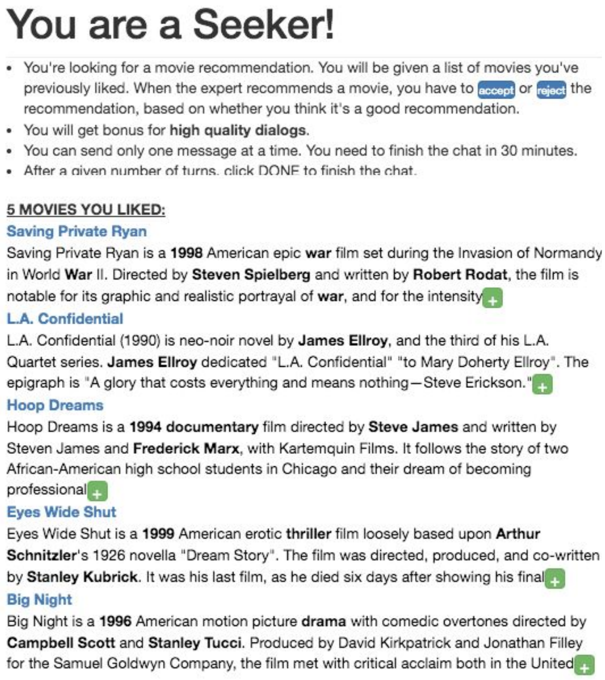
$$\mathcal{L}_{decide} = p_{MLP}(d_{t+1}|h_t, c_1, \cdots, c_K) \quad (16)$$

We also experimented with other functions of the movie prediction distribution (e.g., skewness and kurtosis (Mardia, 1970)), but the multi-layer perceptron (`MLP`) always performed better.

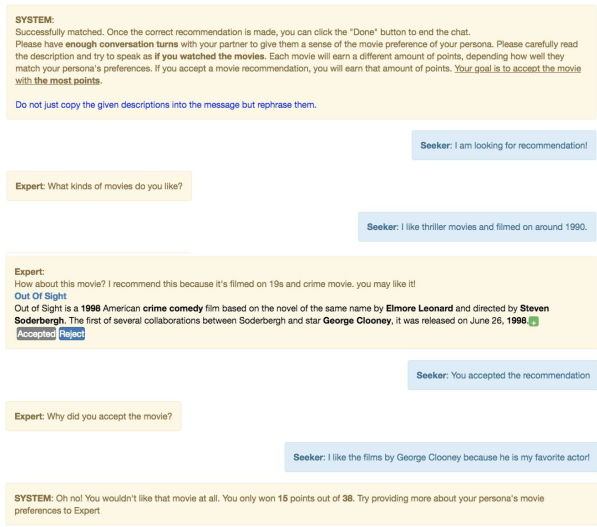
Supervised loss of the overall system: The overall objective function of the full supervised system is as follows:

$$\mathcal{L}_{sup} = \alpha\mathcal{L}_{gen} + \beta\mathcal{L}_{predict} + (1-\alpha-\beta)\mathcal{L}_{decide} \quad (17)$$

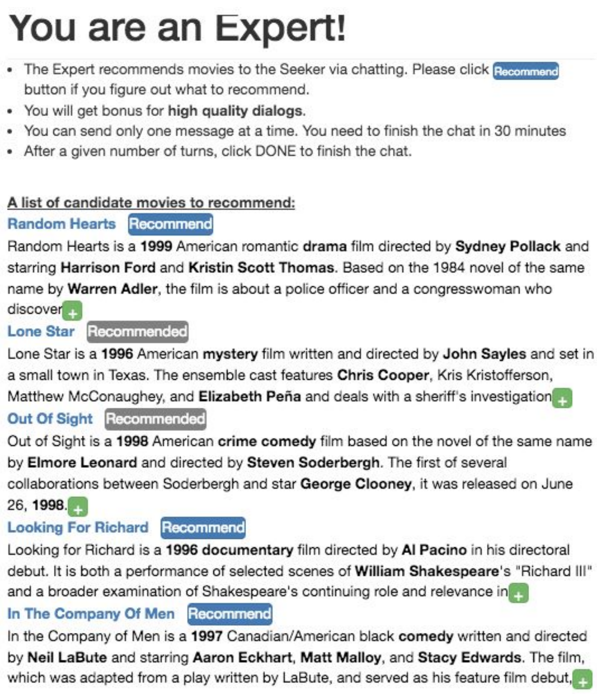
where α and β are weight terms that control the balance between the different objectives and are empirically optimized on the validation set. For the predict and decide losses, we use annealing at the beginning of training, with all the weight being given to the generate loss, and the weights of the other two being gradually increased.



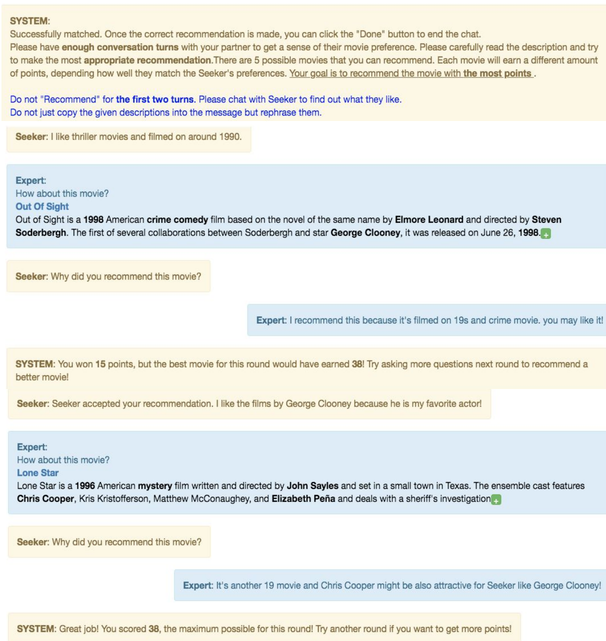
(a) Seeker's left panel



(b) Seeker's right panel



(c) Expert's left panel



(d) Expert's right panel

Figure 8: Interface of our data collection (2): seeker's and expert's pages.