Addressee and Response Selection for Multi-Party Conversation

Hiroki Ouchi
Nara Institute of Science and Technology
ouchi.hiroki.nt6@is.naist.jp

Yuta Tsuboi
IBM Research - Tokyo
yutat@jp.ibm.com

Abstract

To create conversational systems working in actual situations, it is crucial to assume that they interact with multiple agents. In this work, we tackle addressee and response selection for multi-party conversation, in which systems are expected to select *whom* they address as well as *what* they say. The key challenge of this task is to jointly model *who* is talking about *what* in a previous context. For the joint modeling, we propose two modeling frameworks: 1) static modeling and 2) dynamic modeling. To show benchmark results of our frameworks, we created a multi-party conversation corpus. Our experiments on the dataset show that the recurrent neural network based models of our frameworks robustly predict addressees and responses in conversations with a large number of agents.

1 Introduction

Short text conversation (STC) has been gaining popularity: given an input message, predict an appropriate response in a single-round, two-party conversation (Wang et al., 2013; Shang et al., 2015). Modeling STC is simpler than modeling a complete conversation, but instantly helps applications such as chat-bots and automatic short-message replies (Ji et al., 2014).

Beyond two-party conversations, there is also a need for modeling *multi-party conversation*, a form of conversation with several interlocutors conversing with each other (Traum, 2003; Dignum and Vreeswijk, 2003; Uthus and Aha, 2013). For example, in the Ubuntu Internet Relay Chat (IRC), several users cooperate to find a solution for a technical issue contributed by another user. Each agent might have one part of the solution, and these pieces have to be combined through conversation in order to come up with the whole solution.

User	Addressee	Utterance
User 1	-	I have a problem when I install ...
SYSTEM	-	did you set initial params ?
User 2	-	Show the error message, and ...
User 1	*SYSTEM*	how ?
User 1	User 2	ok just a moment !
SYSTEM	[To Whom?]	[What?]
	1. User 1 2. User 2	1. see this URL : http://xxxx 2. It 's already in os

Figure 1: Addressee and response selection for multi-party conversation. A *SYSTEM* is required to select an appropriate addressee from the interlocutors in the conversational context and an appropriate response from the fixed set of candidates.

A unique issue of such multi-party conversations is *addressing*, a behavior whereby interlocutors indicate to whom they are speaking (Jovanović and Akker, 2004; Akker and Traum, 2009). In face-to-face communication, the basic clue for specifying addressees is turning one's face toward the addressee. In contrast, in voice-only or text-based communication, the explicit declaration of addressee's names is more common.

In this work, we tackle *addressee and response selection* for multi-party conversation: given a context, predict an addressee and response. As Figure 1 shows, a system is required to select an addressee from the agents appearing in the previous context and a response from a fixed set of candidate responses (Section 3).

The key challenge for predicting appropriate addressees and responses is to jointly capture *who* is talking about *what* at each time step in a context. For jointly modeling the speaker-utterance information, we present two modeling frameworks: 1) *static modeling* and 2) *dynamic modeling* (Section 5). While speakers are represented as fixed vectors in the static modeling, they are represented as hidden state vectors that dynamically change with time steps in the dynamic modeling. In practice, our models trained for the task can be applied to retrieval-based conversation systems, which retrieves candidate responses from a large-scale repository with the matching model and returns the highest scoring one with the ranking model (Wang et al., 2013; Ji et al., 2014; Wang et al., 2015). Our trained models work as the ranking model and allow the conversation system to produce addressees as well as responses.

To evaluate the trained models, we provide a corpus and dataset. By exploiting Ubuntu IRC Logs[1], we build a large-scale multi-party conversation corpus, and create a dataset from it (Section 6). Our experiments on the dataset show the models instantiated by the static and dynamic modeling outperform a strong baseline. In particular, the model based on the dynamic modeling robustly predicts appropriate addressees and responses even if the number of interlocutors in a conversation increases.[2]

We make three contributions in this work:

1. We formalize the task of addressee and response selection for multi-party conversation.

2. We present modeling frameworks and the performance benchmarks for the task.

3. We build a large-scale multi-party conversation corpus and dataset for the task.

2 Related Work

This work follows in the footsteps of Ritter et al. (2011), who tackled the response generation problem: given a context, generate an appropriate response. While previous response generation approaches utilize statistical models on top of heuristic rules or templates (Levin et al., 2000; Young et al., 2010; Walker et al., 2003), they apply *statistical machine translation based techniques* without such heuristics, which leads to recent work utilizing the SMT-based techniques with neural networks (Shang et al., 2015; Vinyals and Le, 2015; Sordoni et al., 2015; Serban et al., 2016).

As another popular approach, *retrieval-based techniques* are used to retrieve candidate responses from a repository and return the highest scoring one with the ranking model (Ji et al., 2014; Wang et al., 2015; Hu et al., 2014; Wang et al., 2013; Lu and Li, 2013). Stemming from this approach, the next utterance classification (NUC) task has been proposed, in which a system is required to select an appropriate response from a fixed set of candidates (Lowe et al., 2015; Kadlec et al., 2015). The NUC is regarded as focusing on the ranking problem of retrieval-based system, since it omits the candidate retrieving step. The merit of NUC is that it allows us to easily evaluate the model performance on the basis of accuracy.

Our proposed addressee and response selection task is an extension of the NUC. We generalize the task by integrating the addressee detection, which has been regarded as a problematic issue in multi-party conversation (Traum, 2003; Jovanović and Akker, 2004; Uthus and Aha, 2013). Basically, the addressee detection has been tackled in the spoken/multimodal dialog system research, and the models largely rely on acoustic signal or gaze information (Jovanović et al., 2006; Akker and Traum, 2009; Ravuri and Stolcke, 2014). This current work is different from such previous work in that our models predict addressees with only textual information.

For predicting addressees or responses, how the context is encoded is crucial. In single-round conversation, a system is expected to encode only one utterance as a context (Ritter et al., 2011; Wang et al., 2013). In contrast, in multi-turn conversation, a system is expected to encode multiple utterances (Shang et al., 2015; Lowe et al., 2015). Very recently, individual personalities have been encoded as distributed embeddings used for response generation in two-party conversation (Li et al., 2016). Our work is different from that work in that our proposed personality-independent representation allows us to handle new agents unseen in the training data.

[1] http://irclogs.ubuntu.com/

[2] Our code, corpus, and dataset are publicly available at https://github.com/hiroki13/response-ranking

	Type	Notation
Input	Responding Agent	a_{res}
	Context	$\mathcal{C}$
	Candidate Responses	$\mathcal{R}$
Output	Addressee	$a \in \mathcal{A}(\mathcal{C})$
	Response	$\boldsymbol{r} \in \mathcal{R}$

Table 1: Notations for the ARS task.

3 Addressee and Response Selection

We propose and formalize the task of *addressee and response selection* (ARS) for multi-party conversation. The ARS task assumes the situation where a responding agent gives a response to an addressee following a context.[3]

Notation

Table 1 shows the notations for the formalization. We denote vectors with bold lower-case (e.g. $\mathbf{x}_t$, $\mathbf{h}$), matrices with bold upper-case (e.g. $\mathbf{W}$, $\mathbf{H}_a$), scalars with italic lower-case or upper-case (e.g. a_m, Q), and sets with bold italic lower-case or cursive upper-case (e.g. $\boldsymbol{x}$, $\mathcal{C}$) letters.

Formalization

Given an input conversational situation $\boldsymbol{x}$, an addressee a and a response $\boldsymbol{r}$ are predicted:

$$\text{GIVEN} : \boldsymbol{x} = (a_{res},\ \mathcal{C},\ \mathcal{R})$$

$$\text{PREDICT} : a, \boldsymbol{r}$$

where a_{res} is a responding agent, $\mathcal{C}$ is a context and $\mathcal{R}$ is a set of candidate responses. The context $\mathcal{C}$ is a sequence of previous utterances up to the current time step T:

$$\mathcal{C} = (\boldsymbol{u}_{a_1,1}, \cdots, \boldsymbol{u}_{a_T,T})$$

where $\boldsymbol{u}_{a_t,t}$ is an utterance given by an agent a_t at a time step t. Each utterance $\boldsymbol{u}_{a_t,t}$ is a sequence of N_t tokens:

$$\boldsymbol{u}_{a_t,t} = (w_{a_t,t,1}, \cdots, w_{a_t,t,N_t})$$

where $w_{a_t,t,n}$ is a token index in the vocabulary $\mathcal{V}$.

[3] In actual situations, responses can be addressed to multiple agents. In this work, we assume the situation where one specific agent can be the addressee of a response.

To predict an addressee a as a target output, we select an agent from a set of the agents appearing in a context $\mathcal{A}(\mathcal{C})$. Note that a ground-truth addressee is always included in $\mathcal{A}(\mathcal{C})$. To predict an appropriate response $\boldsymbol{r}$, we select a response from a set of candidate responses $\mathcal{R}$, which consists of Q candidates:

$$\mathcal{R} = \{\boldsymbol{r}_1, \cdots, \boldsymbol{r}_Q\}$$

$$\boldsymbol{r}_q = (w_{q,1}, \cdots, w_{q,N_q})$$

where $\boldsymbol{r}_q$ is a candidate response, which consists of N_q tokens, and $w_{q,n}$ is an token index in the vocabulary $\mathcal{V}$.

4 Dual Encoder Models

Our proposed models are extensions of the *dual encoder* (DE) model in (Lowe et al., 2015). The DE model consists of two recurrent neural networks (RNN) that respectively compute the vector representation of an input context and candidate response.

A generic RNN, with input $\mathbf{x}_t \in \mathbb{R}^{d_w}$ and *recurrent* state $\mathbf{h}_t \in \mathbb{R}^{d_h}$, is defined as:

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t) = \pi(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t) \quad (1)$$

where π is a non-linear function, $\mathbf{W}_x \in \mathbb{R}^{d_h \times d_w}$ is a parameter matrix for $\mathbf{x}_t$, $\mathbf{W}_h \in \mathbb{R}^{d_h \times d_h}$ is a parameter matrix for $\mathbf{h}_{t-1}$, and the recurrence is seeded with the $\mathbf{0}$ vector, i.e. $\mathbf{h}_0 = \mathbf{0}$. The recurrent state $\mathbf{h}_t$ acts as a compact *summary* of the inputs seen up to time step t.

In the DE model, each word vector of the context $\mathcal{C}$ and the response $\boldsymbol{r}_q$ is consumed by each RNN, and is then summarized into the context vector $\mathbf{h}_c \in \mathbb{R}^{d_h}$ and the response vector $\mathbf{h}_q \in \mathbb{R}^{d_h}$. Using these vectors, the model calculates the probability that the given candidate response is the ground-truth response given the context as follows:

$$Pr(y(\boldsymbol{r}_q) = 1 | \mathcal{C}, \boldsymbol{r}_q) = \sigma(\mathbf{h}_c^{\mathrm{T}}\ \mathbf{W}\ \mathbf{h}_q) \quad (2)$$

where y is a binary function mapping from $\boldsymbol{r}_q$ to $\{0, 1\}$, in which 1 represents the ground-truth sample and 0 represents the false one, σ is the logistic sigmoid function, and $\mathbf{W} \in \mathbb{R}^{d_h \times d_h}$ is a parameter matrix. As extensions of this model, we propose our multi-party encoder models.

5 Multi-Party Encoder Models

For capturing multi-party conversational streams, we jointly encode *who* is speaking *what* at each time step. Each agent and its utterance are integrated into the hidden states of an RNN.

We present two multi-party modeling frameworks: (i) *static modeling* and (ii) *dynamic modeling*, both of which jointly utilize agent and utterance representation for encoding multiple-party conversation. What distinguishes the models is that while the agent representation in the static modeling framework is fixed, the one in the dynamic modeling framework changes along with each time step t in a conversation.

Modeling Frameworks

As an instance of the static modeling, we propose a static model to capture the *speaking-orders* of agents in conversation. As an instance of the dynamic modeling, we propose a dynamic model using an RNN to track *agent states*. Note that the agent representations are independent of each personality (unique user). The personality-independent representation allows us to handle new agents unseen in the training data.

Formally, similar to Eq. 2, both of the models calculate the probability that the addressee a_p or response $\boldsymbol{r}_q$ is the ground-truth given the input $\mathbf{x}$:

$$Pr(y(a_p) = 1|\mathbf{x}) = \sigma\left([\mathbf{a}_{res}\,;\,\mathbf{h}_c]^{\mathrm{T}}\,\mathbf{W}_a\;\mathbf{a}_p\right) \quad (3)$$

$$Pr(y(\boldsymbol{r}_q) = 1|\mathbf{x}) = \sigma\left([\mathbf{a}_{res}\,;\,\mathbf{h}_c]^{\mathrm{T}}\,\mathbf{W}_r\,\mathbf{h}_q\right) \quad (4)$$

where y is a binary function mapping from a_p or $\boldsymbol{r}_q$ to $\{0, 1\}$, in which 1 represents the ground-truth sample and 0 represents the false one. The function σ is the logistic sigmoid function. $\mathbf{a}_{res} \in \mathbb{R}^{d_a}$ is a responding agent vector, $\mathbf{a}_p \in \mathbb{R}^{d_a}$ is a candidate addressee vector, $\mathbf{h}_c \in \mathbb{R}^{d_h}$ is a context vector, $\mathbf{h}_q \in \mathbb{R}^{d_h}$ is a candidate response vector. These vectors are respectively defined in each model. $\mathbf{W}_a \in \mathbb{R}^{(d_a+d_h)\times d_h}$ is a parameter matrix for the addressee selection probability, and $\mathbf{W}_r \in \mathbb{R}^{(d_a+d_h)\times d_h}$ is a parameter matrix for the response selection probability. These model parameters are learned during training.

On the basis of Eqs. 3 and 4, a resulting addressee and response are selected as follows:

$$\hat{a} = \underset{a_p \in \mathcal{A}(\mathcal{C}_T)}{\operatorname{argmax}}\, Pr(y(a_p) = 1|\mathbf{x}) \quad (5)$$

$$\hat{\boldsymbol{r}} = \underset{\boldsymbol{r}_q \in \mathcal{R}}{\operatorname{argmax}}\, Pr(y(\boldsymbol{r}_q) = 1|\mathbf{x}) \quad (6)$$

where $\hat{a}$ is the highest probability addressee of a set of agents in the context $\mathcal{A}(\mathcal{C})$, and $\hat{\boldsymbol{r}}$ is the highest probability response of a set of candidate responses $\mathcal{R}$.

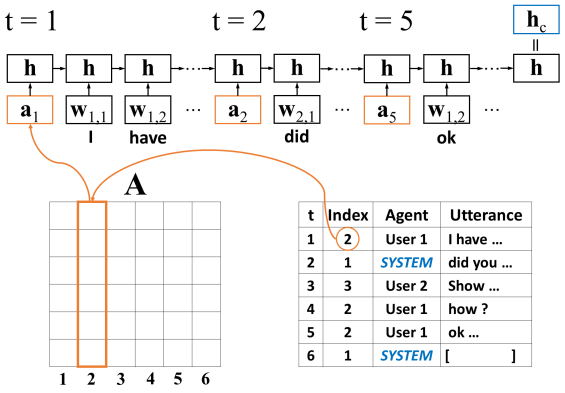

Figure 2: Illustrative example of our static model.

5.1 A Static Model

In the static model, agent matrix $\mathbf{A}$ is defined for the agent vectors in Eqs. 3 and 4. This agent matrix can be defined arbitrarily. We define the agent matrix $\mathbf{A}$ on the basis of *agents' speaking orders*. Intuitively, the agents that spoke in recent time steps are more likely to be an addressee. Our static model captures such property.

The static model is shown in Figure 2. First, agents in the context $\mathcal{A}(\mathcal{C})$ and a responding agent a_{res} are sorted in descending order based on each latest speaking time. Then the order is assigned as an agent index $a_m \in (1, \cdots, |\mathcal{A}(\mathcal{C})|)$ to each agent. In the table shown in Figure 2, the responding agent (represented as `SYSTEM`) has the agent index 1 because he spoke at the most recent time step $t = 6$. Similarly, `User 1` has the index 2 because he spoke at the second most recent time step $t = 5$, and `User 2` has the index 3 because he spoke at the third $t = 3$.

Each speaking-order index a_m is associated with the a_m-th column of the agent matrix $\mathbf{A}$:

$$\mathbf{a}_m = \mathbf{A}[*, a_m]$$

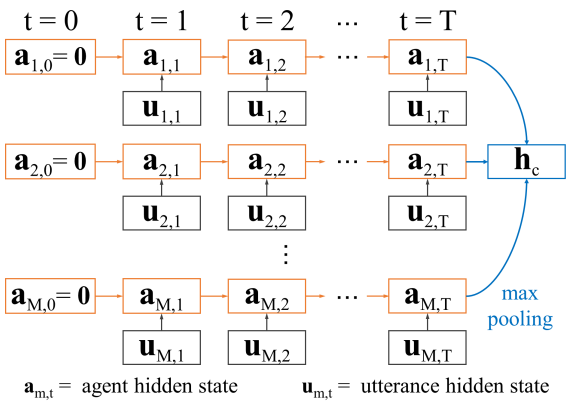

Figure 3: Illustrative example of our dynamic model.

Similarly, a responding agent vector a_{res} and a candidate addressee vector a_p in Eqs. 3 and 4 are respectively extracted from $\mathbf{A}$, i.e. $\mathbf{a}_{res} = \mathbf{A}[*, a_{res}]$ and $\mathbf{a}_p = \mathbf{A}[*, a_p]$.

Consuming the agent vectors, an RNN updates its hidden state. For example, at the time step $t = 1$ in Figure 2, the agent vector $\mathbf{a}_1$ of `User 1` is extracted from $\mathbf{A}$ on the basis of agent index 2 and then consumed by the RNN. Then, the RNN consumes each word vector $\mathbf{w}$ of `User 1`'s utterance. By consuming the agent vector before word vectors, the model can capture which agent speaks the utterance. The last state of the RNN is regarded as $\mathbf{h}_c$. As the transition function f of RNN (Eq. 1), we use the Gated Recurrent Unit (GRU) (Cho et al., 2014; Chung et al., 2014).

For the candidate response vector $\mathbf{h}_q$, each word vector $(\mathbf{w}_{q,1}, \cdots, \mathbf{w}_{q,N_q})$ in the response $\boldsymbol{r}_q$ is summarized with the RNN. Using these vectors $\mathbf{a}_{res}$, $\mathbf{a}_p$, $\mathbf{h}_c$, and $\mathbf{h}_q$, we predict a next addressee and response with the Eqs. 3 and 4.

5.2 A Dynamic Model

In the static model, agent representation $\mathbf{A}$ is a fixed matrix that does not change in a conversational stream. In contrast, in the dynamic model, agent representation $\mathbf{A}_t$ tracks each agent's hidden state which dynamically changes with time steps t.

Figure 3 shows the overview of the dynamic model. Initially, we set a zero matrix as initial agent state $\mathbf{A}_0$, and each column vector of the agent matrix corresponds to an agent hidden state vector. Then, each agent state is updated by consuming the utterance vector at each time step. Note that the states of the agents that are not speaking at the time are updated by zero vectors.

Formally, each column of $\mathbf{A}_t$ corresponds to an agent state vector:

$$\mathbf{a}_{m,t} = \mathbf{A}_t[*, a_m]$$

where an agent state vector $\mathbf{a}_{m,t}$ of an agent a_m at a time step t is the a_m-th column of the agent matrix $\mathbf{A}_t$.

Each vector of the matrix is updated at each time step, as shown in Figure 3. An agent state vector $\mathbf{a}_{m,t} \in \mathbb{R}^{d_a}$ for each agent a_m at each time step t is recurrently computed:

$$\mathbf{a}_{m,t} = g(\mathbf{a}_{m,t-1}, \mathbf{u}_{m,t}), \;\; \mathbf{a}_{m,0} = \mathbf{0}$$

where $\mathbf{u}_{m,t} \in \mathbb{R}^{d_w}$ is a summary vector of an utterance of an agent a_m and computed with an RNN. As the transition function g, we use the GRU. For example, at a time step $t = 2$ in Figure 3, the agent state vector $\mathbf{a}_{1,2}$ is influenced by its utterance vector $\mathbf{u}_{1,2}$ and updated from the previous state $\mathbf{a}_{1,1}$.

The agent matrix updated up to the time step T is denoted as $\mathbf{A}_T$, which is max-pooled and used as a summarized context vector:

$$\mathbf{h}_c = \max_i \; \mathbf{A}_t[i]$$

The agent matrix $\mathbf{A}_T$ is also used for a responding agent vector $\mathbf{a}_{res}$ and a candidate addressee vector $\mathbf{a}_p$, i.e. $\mathbf{a}_{res} = \mathbf{A}_T[*, a_{res}]$ and $\mathbf{a}_p = \mathbf{A}_T[*, a_p]$. $\boldsymbol{r}_q$ is summarized into a response vector $\mathbf{h}_q$ in the same way as the static model.

5.3 Learning

We train the model parameters by minimizing the joint loss function:

$$\mathcal{L}(\boldsymbol{\theta}) = \alpha \, \mathcal{L}_a(\boldsymbol{\theta}) + (1 - \alpha) \, \mathcal{L}_r(\boldsymbol{\theta}) + \frac{\lambda}{2} ||\boldsymbol{\theta}||^2$$

where $\mathcal{L}_a$ is the loss function for the addressee selection, $\mathcal{L}_r$ is the loss function for the response selection, α is the hyper-parameter for the interpolation, and λ is the hyper-parameter for the L2 weight decay.

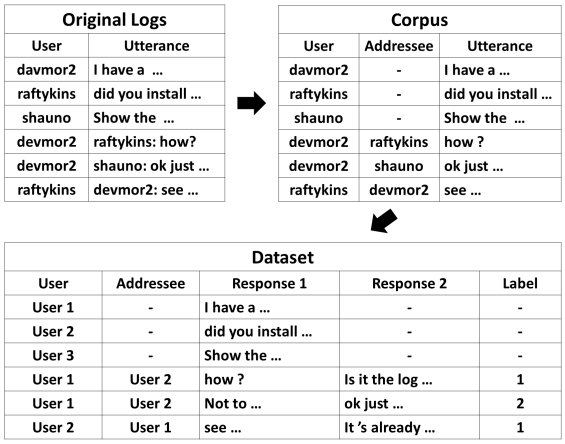

Figure 4: The flow of the corpus and dataset creation. From the original logs, we extract addressee IDs and add them to the corpus. As the dataset, we add candidate responses and the labels.

For addressee and response selection, we use the cross-entropy loss functions:

$$\mathcal{L}_a(\boldsymbol{\theta}) = -\sum_n [\log\ Pr(y(a^+) = 1|\boldsymbol{x}) + \log\left(1 - Pr(y(a^-) = 1|\boldsymbol{x})\right]$$

$$\mathcal{L}_r(\boldsymbol{\theta}) = -\sum_n [\log\ Pr(y(\boldsymbol{r}^+) = 1|\boldsymbol{x}) + \log\left(1 - Pr(y(\boldsymbol{r}^-) = 1|\boldsymbol{x})\right]$$

where $\boldsymbol{x}$ is the input set for the task, i.e. $\boldsymbol{x} = (a_{res}, \mathcal{C}, \mathcal{R})$, a^+ is a ground-truth addressee, a^- is a false addressee, $\boldsymbol{r}^+$ is a ground-truth response, and $\boldsymbol{r}^-$ is a false response. As a false addressee a^-, we pick up and use the addressee with the highest probability from the set of candidate addressees except the ground-truth one ($\mathcal{A}(\mathcal{C}) \setminus a^+$). As a false response, we randomly pick up and use a response from the set of candidate responses except the ground-truth one ($\mathcal{R} \setminus \boldsymbol{r}^+$).

6 Corpus and Dataset

Our goal is to provide a multi-party conversation corpus/dataset that can be used over a wide range of conversation research, such as turn-taking modeling (Raux and Eskenazi, 2009) and disentanglement modeling (Elsner and Charniak, 2010), as well as for the ARS task. Figure 4 shows the flow of the corpus and dataset creation process. We firstly crawl Ubuntu IRC Logs and preprocess the obtained logs.

	Corpus	Dataset		
		Train	Dev	Test
No. of Docs	7355	6,606	367	382
No. of Utters	2.4 M	2.1 M	13.2 k	15.1 k
No. of Words	27.0 M	23.8 M	1.5 M	1.7 M
No. of Samples	-	665.6 k	45.1 k	51.9 k
Avg. W. / U.	11.1	11.1	11.2	11.3
Avg. A. / D.	26.8	26.3	30.68	32.1

Table 2: Statistics of the corpus and dataset. "Docs" is documents, "Utters" is utterances, "W. / U." is the number of words per utterance, "A. / D." is the number of agents per document.

Then, from the logs, we extract and add addressee information to the corpus. In the final step, we set candidate responses and labels as the dataset. Table 2 shows the statistics of the corpus and dataset.

6.1 Ubuntu IRC Logs

The Ubuntu IRC Logs is a collection of logs from Ubuntu-related chat rooms. In each chat room, a number of users chat on and discuss various topics, mainly related to technical support with Ubuntu issues.

The logs are put together into one file per day for each room. Each file corresponds to a document $\mathcal{D}$. In a document, one line corresponds to one log given by a user. Each log consists of three items $(\texttt{Time}, \texttt{UserID}, \texttt{Utterance})$. Using such information, we create a multi-party conversation corpus.

6.2 The Multi-Party Conversation Corpus

To pick up only the documents written in English, we use a language detection library (Nakatani, 2010). Then, we remove the system logs from each document and leave only user logs. For segmenting the words in each utterance, we use a word tokenizer (`TreebankWordTokenizer`) of the Natural Language Toolkit[4]. Using the preprocessed documents, we create a corpus, whose row consists of the three items $(\texttt{UserID}, \texttt{Addressee}, \texttt{Utterance})$.

First, the IDs of the users in a document are collected into the user ID list by referring to the `UserID` in each log. Then, as the addressee user ID, we extract the first word of each utterance. In the Ubuntu IRC Logs, users follow the *name mention* convention (Uthus and Aha, 2013), in which they express

[4]http://www.nltk.org/

their addressee by mentioning the addressee's user ID at the beginning of the utterance. By exploiting the name mentions, if the first word of each utterance is identical to a user ID in the user ID list, we extract the addressee ID and then create a table consisting of (UsetID, Addressee, Utterance). In the case that addressee IDs are not explicitly mentioned at the beginning of the utterance, we do not extract anything.

6.3 The ARS Dataset

By exploiting the corpus, we create a dataset for the ARS task. If the line of the corpus includes an addressee ID, we regard it as a sample for the task. As the ground truth addressees and responses, we straightforwardly use the obtained addressee IDs and the preprocessed utterances.

As false responses, we sample utterances elsewhere within a document. This document-within sampling method makes the response selection task more difficult than the random sampling method[5]. One reason for this is that common or similar topics in a document are often discussed and the used words tend to be similar, which makes the word-based features for the task less effective. We partitioned the dataset randomly into a training set (90%), a development set (5%) and a test set (5%).

7 Experiments

We provide performance benchmarks of our learning architectures on the addressee and response selection (ARS) task for multi-party conversation.

7.1 Experimental Setup

Datasets

We use the created dataset for the experiments. The number of candidate responses RES−CAND ($|\mathcal{R}|$) is set to 2 or 10.

Evaluation Metrics

We evaluate performance by accuracies on three aspects: addressee-response pair selection (ADR−RES), addressee selection (ADR), and response selection (RES). In the addressee-response pair selection, we regard the answer as correct if both the addressee and the response are correctly selected. In the addressee/response selection, we regard the answer as correct if the addressee/response is correctly selected.

Optimization

The models are trained by backpropagation through time (Werbos, 1990; Graves and Schmidhuber, 2005). For the backpropagation, we use stochastic gradient descent (SGD) with a mini-batch training method. The mini-batch size is set to 128. The hyper-parameter α for the interpolation between the two loss functions (Section 5.3) is set to 0.5. For the L2 weight decay, the hyper-parameter λ is selected from $\{0.001, 0.0005, 0.0001\}$.

Parameters of the models are randomly initialized over a uniform distribution with support $[-0.01, 0.01]$. To update parameters, we use *Adam* (Kingma and Ba, 2014) with the default setting suggested by the authors. As the word embeddings, we used the 300 dimension vectors pre-trained by *GloVe*[6] (Pennington et al., 2014). To avoid overfitting, the word vectors are fixed across all experiments. The hidden dimensions of parameters are set to $d_w = 300$ and $d_h = 50$ in the both models, and d_a is set to 300 in the static model and 50 in the dynamic model.

To identify the best training epoch and model configuration, we use the *early stopping* method (Yao et al., 2007). In this method, if the best accuracy of ADR−RES on the development set has not been updated for consecutive 5 epochs, training is stopped and the best performing model is picked up. The max epochs is set to 30, which is sufficient for convergence.

Implementation Details

For computational efficiency, we limit the length of a context $\mathcal{C}$ as $\mathcal{C}_{T-N_c+1:T} = (\boldsymbol{u}_{T-N_c+1}, \cdots, \boldsymbol{u}_T)$, where N_c, called *context window*, is the number of utterances prior to a time step t. We set N_c to $\{5, 10, 15\}$. In addition, we truncate the utterances and responses at a maximum of 20 words. For batch processing, we zero-pad them so that the number of words is constant. Out-of-vocabulary words are replaced with <unk>, whose vector is the averaged vector over all word vectors.

[5] Lowe et al. (2015) adopted the random sampling method.

[6] http://nlp.stanford.edu/projects/glove/

	N_c	RES-CAND = 2			RES-CAND = 10		
		ADR-RES	ADR	RES	ADR-RES	ADR	RES
Chance	-	0.62	1.24	50.00	0.12	1.24	10.00
Baseline	5	36.97	55.73	65.68	16.34	55.73	28.19
	10	37.42	55.63	67.79	16.11	55.63	29.48
	15	37.13	55.62	67.89	15.44	55.62	29.19
Static	5	46.99	60.39	75.07	21.98	60.26	33.27
	10	48.67	60.97	77.75	23.31	60.66	35.91
	15	49.27	61.95	78.14	23.49	60.98	36.58
Dynamic	5	49.80	63.19	76.07	23.72	63.28	33.62
	10	53.85	66.94	78.16	25.95	66.70	36.14
	15	**54.88**	**68.54**	**78.64**	**27.19**	**68.41**	**36.93**

Table 3: Benchmark results: accuracies on addressee-response selection (ADR-RES), addressee selection (ADR), and response selection (RES). N_c is the context window. Bolded are the best per column.

Baseline Model

We set a baseline using the term frequency-inverse document frequency (TF-IDF) retrieval model for the response selection (Lowe et al., 2015). We firstly compute two TF-IDF vectors, one for a context window and one for a candidate response. Then, we compute a cosine similarity for these vectors, and select the highest scoring candidate response as a result. For the addressee selection, we adopt a rule-based method: to determine the agent that gives an utterance most recently except a responding agent, which captures the tendency that agents often respond to the other that spoke immediately before.

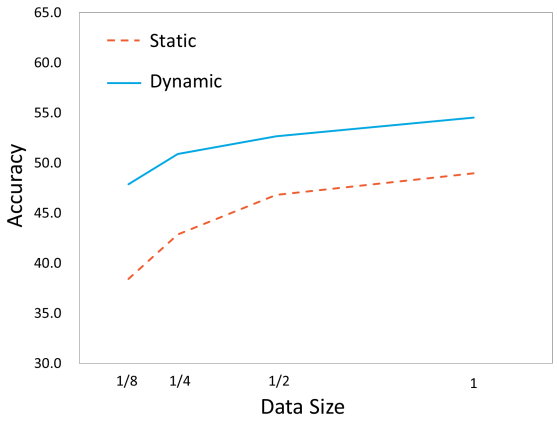

Figure 5: Accuracies in addressee-response selection using different amount of samples for training.

7.2 Results

Overall Performance

Table 3 shows the empirical benchmark results. The dynamic model achieves the best results in all the metrics. The static model outperforms the baseline, but is inferior to the dynamic model.

In addressee selection (ADR), the baseline model achieves around 55% in accuracy. This means that if you select the agents that spoke most recently as an addressee, the half of them are correct. Compared with the baseline, our proposed models achieve better results, which suggests that the models can select the correct addressees that spoke at more previous time steps. In particular, the dynamic model achieves 68% in accuracy, which is 7 point higher than the accuracy of static model.

In response selection (RES), our models outperform the baseline. Compared with the static model, the dynamic model achieves around 0.5 point higher in accuracy.

Effects of the Context Window

In response selection, a performance boost of our proposed models is observed for the context window $\mathcal{N}_c = 10$ over $\mathcal{N}_c = 5$. Comparing the results of the models with the context window $\mathcal{N}_c = 10$ and $\mathcal{N}_c = 15$, the performance is improved but relatively small, which suggests that the performance almost reaches the convergence. In addressee selection, the performance improvements of the static model with the broader context window is limited. In contrast, in the dynamic model, a steady performance boost is observed, yielding an increase of over 5 points between $N_c = 15$ and $N_c = 5$,

No. of Agents	2-5	6-10	11-15	16-20	21-30	31-100	101-305
No. of Samples	3731	5962	5475	4495	5619	7956	18659
	ADR−RES						
Baseline	52.13	43.51	39.98	42.96	39.70	36.55	29.22
Static	64.17	55.92	50.72	53.04	48.69	49.61	42.86
Dynamic	**66.90**	**57.73**	**54.32**	**55.64**	**51.61**	**55.88**	**52.14**
	ADR						
Baseline	84.94	70.82	62.14	65.52	58.89	51.28	41.47
Static	86.33	74.37	66.12	68.54	63.43	59.24	50.99
Dynamic	**87.64**	**76.48**	**69.99**	**72.21**	**66.90**	**66.78**	**62.11**
	RES						
Baseline	60.71	61.24	64.51	65.58	67.93	71.66	71.38
Static	73.60	73.45	74.54	**75.95**	75.17	81.50	81.60
Dynamic	**75.64**	**74.12**	**75.53**	75.17	**76.05**	**81.96**	**81.81**

Table 4: Performance comparison for different numbers of agents appearing in the context. The numbers are accuracies on the test set with the number of candidate responses CAND−RES $= 2$ and the context window $N_c = 15$.

Effects of the Sample Size

Figure 5 shows the accuracy curves of addressee-response selection (ADR−RES) for different training sample sizes. We use $1/2$, $1/4$, and $1/8$ of the whole training samples for training. The results show that as the amount of the data increases, the performance of our models are improved and gradually approaches the convergence. Remarkably, the performance of the dynamic models using the $1/8$ samples is comparable to that of the static model using the whole samples.

Effects of the Number of Participants

To shed light on the relationship between the model performance and the number of agents in multi-party conversation, we investigate the effect of the number of agents participating in each context. Table 4 compares the performance of the models for different numbers of agents in a context.

In addressee selection, the performance of all models gradually gets worse as the number of agents in the context increases. However, compared with the baseline, our proposed models suppress the performance degradation. In particular, the dynamic model predicts correct addressees most robustly.

In response selection, unexpectedly, the performance of all the models gets better as the number of agents increases. Detailed investigation on the interaction between the number of agents and the response selection complexity is an interesting line of future work.

8 Conclusion

We proposed addressee and response selection for multi-party conversation. Firstly, we provided the formal definition of the task, and then created a corpus and dataset. To present benchmark results, we proposed two modeling frameworks, which jointly model speakers and their utterances in a context. Experimental results showed that our models of the frameworks outperform a baseline.

Our future objective to tackle the task of predicting *whether to respond* to a particular utterance. In this work, we assume that the situations where there is a specific addressee that needs an appropriate response and a system is required to respond. In actual multi-party conversation, however, a system sometimes has to wait and listen to the conversation that other participants are engaging in without needless interruption. Hence, the prediction of *whether to respond* in a multi-party conversation would be an important next challenge.

Acknowledgments

We thank Graham Neubig, Yuya Taguchi, Ryosuke Kohita, Ander Martinez, the members of the NAIST Computational Linguistics Laboratory, the members of IBM Research - Tokyo, Long Doung, and the reviewers for their helpful comments.

References

Rieks Akker and David Traum. 2009. A comparison of addressee detection methods for multiparty conversations. In *Workshop on the Semantics and Pragmatics of Dialogue*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*, pages 1724–1734.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv: 1412.3555*.

Frank PM Dignum and Gerard AW Vreeswijk. 2003. Towards a testbed for multi-party dialogues. *Advances in Agent Communication*, pages 212–230.

Micha Elsner and Eugene Charniak. 2010. Disentangling chat. *Computational Linguistics*, pages 389–409.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proceedings of NIPS*, pages 2042–2050.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv: 1408.6988*.

Natasa Jovanović and op den Rieks Akker. 2004. Towards automatic addressee identification in multiparty dialogues. In *Proceedings of SIGDIAL*.

Natasa Jovanović, op den Rieks Akker, and Anton Nijholt. 2006. Addressee identification in face-to-face meetings. In *Proceedings of EACL*.

Rudolf Kadlec, Martin Schmid, and Jan Kleindiest. 2015. Improved deep learning baselines for ubuntu corpus dialogs. *arXiv preprint arXiv: 1510.03753*.

Diederik P. Kingma and Jimmy Lei Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv: 1412.6980*.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, pages 11–23.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *Proceedings of ACL*.

Ryan Lowe, Nissan Pow, Iulian V. Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of SIGDIAL*, pages 285–294.

Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *Proceedings of NIPS*, pages 1367–1375.

Shuyo Nakatani. 2010. Language detection library for java.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Antoine Raux and Maxine Eskenazi. 2009. A finite-state turn-taking model for spoken dialog systems. In *Proceedings of NAACL*, pages 629–637.

Suman V Ravuri and Andreas Stolcke. 2014. Neural network models for lexical addressee detection. In *Proceedings of INTERSPEECH*, pages 298–302.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of EMNL*, pages 583–593.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of AAAI*, pages 3776–3783.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of ACL/IJCNLP*, pages 1577–1586.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of NAACL/HLT*, pages 196–205.

David Traum. 2003. Issues in multiparty dialogues. *Advances in Agent communication*, pages 201–211.

David C Uthus and David W Aha. 2013. Multiparticipant chat analysis: A survey. *Artificial Intelligence*, pages 106–121.

Oriol Vinyals and V. Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv: 1506.05869*.

Marilyn A Walker, Rashmi Prasad, and Amanda Stent. 2003. A trainable generator for recommendations in multimodal dialog. In *Proceedings of INTERSPEECH*. Citeseer.

Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A dataset for research on short-text conversations. In *Proceedings of EMNLP*, pages 935–945.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2015. Syntax-based deep matching of short texts. In *Proceedings of IJCAI*, pages 1354–1361.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. 2007. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language*, pages 150–174.