Online Conversation Disentanglement with Pointer Networks

Tao Yu[¶] and **Shafiq Joty**[¶†]
[¶]Nanyang Technological University, Singapore
[†]Salesforce Research
{tao003, srjoty}@ntu.edu.sg

Abstract

Huge amounts of textual conversations occur online every day, where multiple conversations take place concurrently. Interleaved conversations lead to difficulties in not only following the ongoing discussions but also extracting relevant information from simultaneous messages. Conversation disentanglement aims to separate intermingled messages into detached conversations. However, existing disentanglement methods rely mostly on handcrafted features that are dataset specific, which hinders generalization and adaptability. In this work, we propose an end-to-end online framework for conversation disentanglement that avoids time-consuming domain-specific feature engineering. We design a novel way to embed the whole utterance that comprises timestamp, speaker, and message text, and propose a custom attention mechanism that models disentanglement as a pointing problem while effectively capturing inter-utterance interactions in an end-to-end fashion. We also introduce a joint-learning objective to better capture contextual information. Our experiments on the Ubuntu IRC dataset show that our method achieves state-of-the-art performance in both link and conversation prediction tasks.

1 Introduction

With the fast growth of Internet and mobile devices, people now commonly communicate in the virtual world to discuss events, issues, tasks, and personal experiences. Among the various methods of communication, text-based conversational media, such as Internet Relay Chat (IRC), Facebook Messenger, Whatsapp, and Slack, has been and remains one of the most popular choices. Multiple ongoing conversations seem to occur naturally in such social and organizational interactions, especially when the conversation involves more than two participants (Elsner and Charniak, 2010). For example,

Time	Sp	Message Text
02:26	system	===zelot joined the channel
02:26	zelot	hi, where can i get some help in regards to issues with mount?
02:26	TuxThePenguin	After taking it out
02:26	hannasanarion	TuxThePenguin, try booting with monitors connected to motherboard
02:26	pnunn	TuxThePonguin, sounds like there is on board graphics as well, so try that without the card
02:26	pnunn	Yeh, just one monitor though
02:27	hannasanarion	pnunn, right
02:27	TuxThePenguin	Makes sense to me :)
02:27	pnunn	process of elimination.
02:27	TuxThePenguin	Along with Occam's Razor
02:27	Bashing-om	zelot: If you are on a supported release of 'buntu, this is a good place to ask.
02:27	TuxThePenguin	Any solution is most likely the simplest one
02:28	wllrt	I'm a emacs newb and looking to prevent rsi.

Figure 1: An excerpt of a conversation from the Ubuntu IRC corpus (best viewed in color). Same color reflects same conversation. Mentions of names are highlighted.

consider the excerpt of a multi-party conversation in Figure 1 taken from the Ubuntu IRC corpus (Lowe et al., 2015). Even in this small excerpt, there are 4 concurrent conversations (distinguished by different colors) among 4 participants.

Identifying or disentangling individual conversations is often considered as a prerequisite for downstream dialog tasks such as utterance ranking and generation (Lowe et al., 2017; Kim et al., 2019). It can also help building other applications such as search, summarization, and question answering over conversations, and support users by providing online help (Joty et al., 2019).

However, often there are no explicit structures or metadata to separate out the individual conversations. Naive heuristics to disentanglement often lead to sub optimal results as Kummerfeld et al. (2019) found that only 10.8% of the conversations

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 6321–6330,
November 16–20, 2020.

in the widely used Ubuntu IRC dialog corpus were extracted correctly by the heuristics employed by Lowe et al. (2015, 2017).

Previous studies have therefore investigated traditional machine learning methods with statistical and linguistic features for conversation disentanglement, *e.g.,* (Shen et al., 2006), (Wang and Oard, 2009; Wang et al., 2011b,a), (Elsner and Charniak, 2010, 2011), to name a few. The task is generally solved by first finding links between utterances, and then grouping them into a set of distinct conversations. Recent work by Jiang et al. (2018) and Kummerfeld et al. (2019) adopt deep learning approaches to learn abstract linguistic features and compute message pair similarity. However, these methods heavily rely on hand-engineered features that are often too specific to the particular datasets (or domains) on which the model is trained and evaluated. For example, many of the features used in (Kummerfeld et al., 2019) are only applicable to the Ubuntu IRC dataset. This hinders the model's generalization and adaptability to other domains.

In this work, we propose a more general framework for conversation disentanglement while avoiding time-consuming and domain-specific feature engineering. In particular, we cast link prediction as a pointing problem, where the model learns to point to the parent of a given utterance (Figure 2). Each pointing operation is modeled as a multinomial distribution over the set of previous utterances. A neural encoder is used to encode each utterance text along with its speaker and timestamp. The pointing function implements a custom attention mechanism that models different interactions between two utterances. This results in an end-to-end neural framework that can be optimized with a simple cross entropy loss. During training, we jointly model the reply-to relationship and pairwise relationship (whether two utterances in the same conversation) under the same framework, so that more contextual and structural information can be learned by our model to further improve the disentangling performance.

Furthermore, the framework supports online decoding, which is naturally provided by the pointer network framework, disentangles a conversation as it unfolds, and can provide real-time help to participants in contributing to the right conversations.

We performed extensive experiments on the recently released Ubuntu IRC dataset (Kummerfeld et al., 2019) and demonstrate that our approach outperforms previous methods for both link prediction and conversation prediction tasks.[1] Ablation studies reveal the importance of different components of the model and special handling of self-links. Our framework is generic and can be applied to chat conversations from other domains.

2 Background

In this section, we give a brief overview of previous work on conversation disentanglement and the generic pointer network model.

2.1 Conversation disentanglement

Most existing approaches treat disentanglement as a two-stage problem. The first stage involves *link prediction* that models "reply-to" relation between two utterances. The second stage is a *clustering* step, which utilizes the results from link prediction to construct the individual conversation threads.

For link prediction, earlier methods used discourse cues and content features within statistical classifiers. Elsner and Charniak (2008, 2010) combine conversation cues like speaker, mention, and time with content features like the number of shared words to train a linear classifier.

Recent methods use neural models to represent utterances with compositional features. Mehri and Carenini (2017) pre-train an LSTM network to predict *reply* probability of an utterance, which is then used in a link prediction classifier along with other handcrafted features. Jiang et al. (2018) model high and low-level linguistic information using a siamese hierarchical convolutional network that models similarity between pairs of utterances in the same conversation. The interactions between two utterances is captured by taking element-wise absolute difference of the encoded sentence features along with other handcrafted features. Kummerfeld et al. (2019) uses feed-forward networks with averaged pre-trained word embedding and many hand-engineered features. Tan et al. (2019) used an utterance-level LSTM network, while Zhu et al. (2019) used a masked transformer to get a context-aware utterance representation considering utterances in the same conversation.

Finding a globally optimal clustering solution for conversation disentanglement has been shown to be NP-hard (McCallum and Wellner, 2005). Previous methods focus mostly on approximating the global

[1] https://github.com/vode/onlinePtrNet_disentanglement

optimal by either using greedy decoding (Wang and Oard, 2009; Elsner and Charniak, 2008, 2010, 2011; Jiang et al., 2018; Aumayr et al., 2011) or training multiple link classifiers to do voting (Kummerfeld et al., 2019). Mehri and Carenini (2017) trained additional classifiers to decide whether an utterance belongs to a conversation or not. Wang et al. (2020) use a multi-task topic tracking framework for conversation disentanglement, topic prediction and next utterance ranking.

Our work is fundamentally different from previous studies in that we treat link prediction as a pointing problem modeled by a multinomial distribution over the previous utterances (as opposed to pairwise binary classification). This formulation allows us to model the global conversation flow. Our method does not rely on any handcrafted features. Each utterance in our method is represented by the utterance text, its speaker and timestamp, which are generic to any conversation. The interactions between the utterances are effectively modeled within the pointer module. Moreover, our framework can work in an end-to-end online setup.

2.2 Pointer Networks

Pointer networks (Vinyals et al., 2015) are a class of encoder-decoder models that can tackle problems where the output vocabulary depends on the input sequence. They use attentions as pointers to the input elements. An encoder network first transforms the input sequence $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ into a sequence of hidden states $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_m)$. At each time step t, the decoder takes the input from the previous step, generates a decoder state $\boldsymbol{d}_t$, and uses it to attend over the input elements. The attention gives a $\mathrm{softmax}$ (multinomial) distribution over the input elements as follows.

$$s_{t,i} = \sigma(\boldsymbol{d}_t, \boldsymbol{h}_i); \quad \boldsymbol{a}_t = \mathrm{softmax}(\boldsymbol{s}_t) \quad (1)$$

where $\sigma(.,.)$ is a scoring function for attention, which can be a neural network or simply a dot product operation. The model uses $\boldsymbol{a}_t$ to infer the output: $\hat{\mathrm{y}}_t = \arg\max(\boldsymbol{a}_t)$.

Similar to the standard pointer network, each pointing mechanism in our approach is modeled as a multinomial distribution over the indices of the input sequence. However, unlike the original pointer network where a decoder state points to an encoder state, in our approach, the current encoder state points to the previous states.

Pointer networks have recently yielded state-of-the-art results in constituency parsing (Nguyen et al., 2020), dependency parsing (Ma et al., 2018), anaphora resolution (Lee et al., 2017), and discourse segmentation and parsing (Lin et al., 2019). To the best of our knowledge, this is the very first work that utilizes a pointer network for conversation disentanglement. It is also a natural fit for online conversation disentanglement.

3 Our Disentanglement Model

Given a sequence of streaming utterances $\mathcal{U} = \{U_1, U_2, \ldots, U_i, \ldots\}$, our task in link prediction (§3.1) is to find the parent utterances $U_{p_i} \subset U_{\leq i}$ that the current utterance U_i replies to. Here, $U_{\leq i}$ refers to all the previous utterances until i, that is, $U_{\leq i} = (U_0, U_1 \ldots, U_i)$. An utterance can reply to itself (*i.e.*, $i = p_i$), for example, the initial message in a conversation or a system message. Besides, one utterance may have multiple parents, and one parent can be replied to by multiple (children) messages. For example, in Figure 1, both `pnunn` and `hannasanarion` reply to `TuxThePonguin`'s message. The case of one message replying to multiple parents is very rare in our corpus (see Table 1).

After link prediction, we employ a decoding algorithm (§3.3) to construct the individual threads.

3.1 Link Prediction by Pointing

We propose a joint learning framework for conversation disentanglement based on pointing operations where Figure 2 shows the network architecture. It has three main components: (*a*) an utterance encoder and (*b*) a pointer module (*c*) a pairwise classification model. The job of the utterance encoder is to encode each utterance U_i as it comes, while the the pointer module implements a custom pointing mechanism to find the ancestor message $U_p \in U_{\leq i}$ that U_i replies to. The pairwise classification model is to determine whether two utterances U_i and U_j are in the same conversation or not.

3.1.1 Utterance Encoder

As shown in Figure 1, each utterance U_i has three components $<t_i, s_i, m_i>$: the timestamp t_i when the utterance was posted, the speaker s_i who posted it, and the message content m_i. We encode these three components separately.

Encoding timestamp. The timestamp `hour:min` is directly encoded as a two-dimensional vector $[hour, min]$. For example, timestamp `02:26` is encoded as $[02, 26]$.

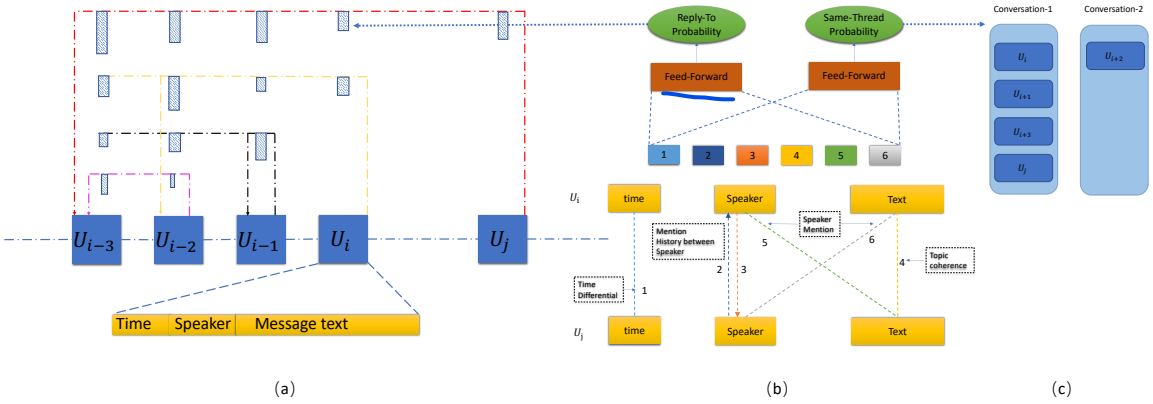

Figure 2: (a) Overview of our Pointer Network joint learning framework for online conversation disentanglement. Each utterance U_i consists of three parts: time (t_i), speaker (s_i) and message text (m_i), which are encoded by the utterance encoder. (b) The Pointer module designs an effective attention mechanism that captures inter-utterance interactions through several features and models link prediction as a multinomial distribution over the previous utterances. (c) The pairwise classification model aims to capture higher-order contextual information. The whole model is trained end-to-end and can decode/disentangle the conversation in an online fashion.

Encoding speaker. In a multi-party conversation, participants mention each other's names to make disentanglement easier, compensating for the lack of visual cues normally present in a face-to-face conversation (O'Neill and Martin, 2003; Elsner and Charniak, 2010). Our goal is to capture the mention relation between utterances in the way we encode the speaker information. For this, each speaker is placed in the same vocabulary as the words, and encoded with a unique identifier (a discrete value).

Encoding message text. We utilize a Bidirectional LSTM or Bi-LSTM (Hochreiter and Schmidhuber, 1997) to encode the raw text message m_i into deep contextual representations of the words. We concatenate the hidden states from both forward and backward LSTM cells. Formally, for a message containing n words $m_i = (w_0, w_1, \ldots, w_n)$, the Bi-LSTM gives $\boldsymbol{H} = (\overrightarrow{\boldsymbol{h}_0} \oplus \overleftarrow{\boldsymbol{h}_0}, \overrightarrow{\boldsymbol{h}_1} \oplus \overleftarrow{\boldsymbol{h}_1}, \ldots, \overrightarrow{\boldsymbol{h}_n} \oplus \overleftarrow{\boldsymbol{h}_n})$, where $\oplus$ denotes vector concatenation.

3.1.2 Pointer Module

Given the encoded representation of the current utterance U_i, our pointer module computes a probability distribution over the previous utterances $U_{<i}$ which represents the probability that U_i replies to an utterance $U_j \in U_{<i}$. The module implements an association function between U_i and U_j by incorporating different kinds of interactions. Figure 2(b) gives a schematic diagram of the module, which has five different components:

(*a*) Time difference: The time difference between U_i and U_j is computed as: $\boldsymbol{t}_{i,j} = [\boldsymbol{t}_i - \boldsymbol{t}_j]$.

(*b*) Mention: To determine whether U_i mentions s_j (speaker of U_j) in its message m_i, we compute:

$$mention_{i,j} = \sum_{w_k \in m_i} \mathbb{1}(s_j = \text{Index}(w_k)) \quad (2)$$

where $\mathbb{1}$ is the Indicator function that returns 1 if the index of w_k in m_i matches the speaker id s_j.

(*c*) Mention history: The pointer module also keeps track of mention histories between two speakers. It not only computes whether s_i mentions s_j in m_i, but it also keeps track of whether s_i, s_j mentioned each other in their previous messages and how often. It maintains an external memory $\boldsymbol{M}$ (a matrix) to record this. At each step, we compute both $mention_{i,j}$ and $mention_{j,i}$, and update the memory $\boldsymbol{M}$ to be used in the next step.

$$\boldsymbol{M}_{i,j} = \boldsymbol{M}_{i,j} + \text{mention}_{i,j} \quad (3)$$
$$\boldsymbol{M}_{j,i} = \boldsymbol{M}_{j,i} + \text{mention}_{j,i} \quad (4)$$

The memory $\boldsymbol{M}$ grows incrementally as the model sees new speakers during training and inference.

(*d*) Topic coherence: To model textual similarity between m_i and m_j, we use a similar method as Chen et al. (2016). Let $\boldsymbol{H}_i = (\boldsymbol{h}_{i,0}, \ldots, \boldsymbol{h}_{i,p})$ and $\boldsymbol{H}_j = (\boldsymbol{h}_{j,0}, \ldots, \boldsymbol{h}_{j,q})$ be the Bi-LSTM representations for m_i and m_j, respectively from the utterance encoder layer. We compute soft alignment between m_i and m_i as follows.

$$\boldsymbol{H}_i^{'} = \text{softmax}(\boldsymbol{H}_i \boldsymbol{H}_j^T)\boldsymbol{H}_j \quad (5)$$

$$\boldsymbol{H}_j^{'} = \text{softmax}(\boldsymbol{H}_j \boldsymbol{H}_i^T)\boldsymbol{H}_i \quad (6)$$

In Eq. 5, we use the vectors in $\boldsymbol{H}_i$ as the query vectors to compute attentions over the key/value vectors in $\boldsymbol{H}_j$, and compute a set of attended vectors $\boldsymbol{H}_i^{'} = (\boldsymbol{h}_{i,0}^{'}, \ldots, \boldsymbol{h}_{i,p}^{'})$, one for each $\boldsymbol{h}_i \in \boldsymbol{H}_i$. Eq. 6 does the same thing but uses $\boldsymbol{H}_j$ as the query vectors and $\boldsymbol{H}_i$ as the key/value vectors to compute the attended vectors, $\boldsymbol{H}_j^{'} = (\boldsymbol{h}_{j,0}^{'}, \ldots, \boldsymbol{h}_{j,q}^{'})$.

Then we enhance the interactions by applying difference and element-wise product between the original representation $\boldsymbol{H}$ and the attended representations $\boldsymbol{H}^{'}$ as follows.

$$\boldsymbol{h}_i^f = [\boldsymbol{h}_i; \boldsymbol{h}_i^{'}; \boldsymbol{h}_i - \boldsymbol{h}_i^{'}; \boldsymbol{h}_i \cdot \boldsymbol{h}_i^{'}] \quad (7)$$

$$\boldsymbol{h}_j^f = [\boldsymbol{h}_j; \boldsymbol{h}_j^{'}; \boldsymbol{h}_j - \boldsymbol{h}_j^{'}; \boldsymbol{h}_j \cdot \boldsymbol{h}_j^{'}] \quad (8)$$

The final representation $\boldsymbol{h}_{i,j}$ is computed as

$$\boldsymbol{h}_{i,j} = \boldsymbol{h}_i^f \oplus \boldsymbol{h}_j^f \quad (9)$$

where $\oplus$ denotes concatenation.

(*e*) Pointing: After computing the above four types of interactions between each pair of utterances, we concatenate them and feed them into a feed-forward network to compute the pointing distribution over all the previous utterances.

$$\boldsymbol{f}_{i,j} = \boldsymbol{t}_{i,j} \oplus \textit{mention}_{i,j} \oplus \textit{mention}_{j,i} \oplus \boldsymbol{M}_{i,j} \oplus \boldsymbol{M}_{j,i} \oplus \boldsymbol{h}_{i,j} \quad (10)$$

$$\text{score}(U_i, U_j) = \tanh(\boldsymbol{w}^T \boldsymbol{f}_{i,j})$$
$$p(U_i, U_j) = \frac{\exp(\text{score}(U_i, U_j))}{\sum_{s=0}^{i} \exp(\text{score}(U_i, U_s))} \quad (11)$$
$$\text{for } j \in (0, ..., i)$$

where $\boldsymbol{w}$ is a shared linear layer parameter. We use cross entropy (CE) loss for the pointer module.

$$\mathcal{L}_{\text{link}}(\theta) = -\sum_{j=0}^{i} y_{i,j} \log p(U_i, U_j) \quad (12)$$

where $y_{i,j} = 1$ if U_i replies to U_j, otherwise 0, and θ are the model parameters.

3.1.3 Pairwise Classification Model

In the above pointer module, we only consider first-order interaction between two utterances in an online fashion (*i.e.,* looking at only previous utterances). However, higher-order information derived from the entire conversation may provide more contextual information for the model to learn useful disentanglement features. We propose a joint learning framework to consider both first- and higher-order information simultaneously in a unified framework; see Figure 2(b)-(c).

For the higher-order information, we train a binary pairwise classifier that decides whether two utterances should be in the same conversation. For any two arbitrary utterances we use the same feature function from Eq. 10 (*i.e.,* the parameters are shared with the pointer module) and feed them into a binary logistic classifier. The probability of two utterances belongs to the same conversation is:

$$p_{\text{pair}}(U_i, U_j) = \text{sigmoid}(\boldsymbol{w}^T \boldsymbol{f}_{i,j}) \quad (13)$$

where $\boldsymbol{w}$ is the classifier parameter. We use a binary cross entropy loss for this model.

$$\mathcal{L}_{\text{pair}}(\theta) = -y_{i,j} \log \hat{y}_{i,j} - (1 - y_{i,j}) \log(1 - \hat{y}_{i,j}) \quad (14)$$

where $\hat{y}_{i,j} = p_{\text{pair}}(U_i, U_j)$, $y_{i,j} = 1$ if U_i and U_j are in the same conversation, otherwise 0, and θ are the model parameters. Since the pairwise classifier is trained on all possible pairs of utterances in a conversation, it models higher-order information about the conversation clusters.

3.2 Training

The final training loss of our model is:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{link}}(\theta) + \lambda \mathcal{L}_{\text{pair}}(\theta) \quad (15)$$

where λ is the hyper-parameter for tuning the importance of the pairwise classification loss. We use Glove 128-dimensional word embedding (Pennington et al., 2014), pre-trained by Lowe et al. (2015) on the #Ubuntu corpus. The hidden layers in the Bi-LSTM are of 256 dimensions. We optimize our model using Adam (Kingma and Ba, 2014) optimizer with a learning rate of 1×10^{-5}. For regularization, we set the dropout at 0.2 and L_2 penalize weight with 1×10^{-7}.

3.3 Decoding

Our framework naturally allows us to disentangle the threads in an online fashion. As a new utterance

U_i arrives, the utterance encoder encodes it into a vector. The pointer module then computes a multinomial distribution over the previous utterances $U_{\leq i}$ (Eq. 11) by modelling pair-wise interactions, and then finds the parent message U_{p_i} as follows.

$$p_i = \underset{j=\{0,\ldots,i\}}{\arg\max} \; p(U_i, U_j) \tag{16}$$

To the best of our knowledge, this is the first work using an end-to-end framework for online conversation disentanglement. In our analysis of several conversations, we found that the self-links (an utterance pointing to itself) play a crucial role in clustering performance. Mistakes in correctly identifying a self-link will result in two misclusterings. To address this, we did some simple adjustment to our decoding method. We raise the threshold for self-link prediction to make a more conservative prediction of self-links. In particular, during decoding we first find the parent with the highest probability, but if it turns out to be a self-link, we see if the probability passes the preset threshold, otherwise the utterance with the second highest probability will be predicted as the parent. The tuning of the threshold parameter for self-link is done on the development set.

4 Experiment

In this section, we present our experiments — the dataset used, the evaluation metrics, experimental setup, and the results with analysis.

4.1 Dataset

The dataset used for training and evaluation is from (Kim et al., 2019), which is the largest dataset available for conversation disentanglement (Kummerfeld et al., 2019). It consists of multi-party conversations extracted from the #Ubuntu IRC channel. A typical conversation starts with a question that was asked by one participant, and then other participants respond with either an answer or follow-up questions. This leads to a back-and-forth conversation between multiple participants. An example of the Ubuntu-IRC data is shown in Figure 1. We follow the same train, dev, test split as the Dialog System Technology Challenges 8 (DSTC8) (Kim et al., 2019).[2] Table 1 reports the dataset statistics.

[2]The ground truth for test set can be found at http://jkk.name/irc-disentanglement/

	Train	Dev	Test
Total # links	52641	2145	4265
Total # conversations	6201	526	370
Avg link distance	8	8	7
Median link distances	3	3	3
Avg parents per utterance	1.03	1.05	1.04
Avg # of utterances per conv.	9	5	12
Median # of utterances per conv.	5	4	6

Table 1: Statistics of train, dev and test datasets.

4.2 Metrics

We consider two kinds of metrics to evaluate our disentanglement model: link level and conversation level. For link-level, we use precision, recall and F-1 scores. For cluster level evaluation, we use the same clustering metrics from DSTC8 and (Kummerfeld et al., 2019). This includes:

(*a*) Variation of Information (VI). This is a measure of information gain or loss when going from one clustering to another (Meilă, 2007). It is the sum of conditional entropies $H(Y|X) + H(X|Y)$, where X and Y are clusterings of the same set of items. We used the bound for n items that $VI(X;Y) \leq \log(n)$, and present $1 - VI$, so that the larger the value the better.

(*b*) Ajusted Random Index (ARI). A measure (also referred to as **1V1**) (Hubert and Arabie, 1985) between two clusterings by considering all links of samples and counting links that are assigned in the same or different clusters in the predicted and true clusterings. ARI is defined as:

$$\frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}}$$

where n_{ij} are number of overlapping links between predicted cluster i and ground truth cluster j, whereas a_i and b_j indicate row and column level summation over n_{ij}.

(*c*) Exact Match F-1. Calculated using the number of perfectly matching conversations, excluding conversations with only one message.

4.3 Models Compared

Feed Forward Model. We use the feed-forward model from (Kummerfeld et al., 2019) as the baseline model, which outperforms previously proposed disentanglement models. For the DSTC8

Model	Cluster Prediction					Link Prediction			Self-Link Prediction		
	VI	1V1	P	R	F1	P	R	F	P	R	F1
FF (T)	66.7	10.0	0.4	0.5	0.7	19.7	19.0	19.4	23.0	98.0	60.5
Ptr-Net (T)	69.3	22.5	2.6	4.2	3.2	25.0	24.0	24.5	60.2	64.3	62.3
FF (−T)	90.2	62.1	26.8	32.3	29.3	71.3	68.7	70.0	80.1	91.0	85.5
Ptr-Net (-T)	89.5	61.0	27.3	29.5	28.4	71.0	67.2	69.2	78.2	90.0	84.1
SHCNN	87.1	62.3	20.9	31.0	25.1	71.7	69.1	70.4	79.5	80.0	80.3
FF	92.2	69.6	38.7	41.6	40.1	74.5	71.8	73.1	86.2	92.5	89.4
+ Self-Link	92.0	70.4	41.9	40.1	41.0	74.2	71.5	72.8	92.1	90.0	91.0
Ptr-Net	92.3	70.2	33.0	38.9	36.0	74.7	72.7	73.7	81.8	92.1	87.0
+ Joint train	93.1	71.3	37.2	42.5	39.7	74.0	71.3	72.7	79.5	**93.6**	86.6
+ Self-link	93.0	74.3	42.2	40.9	41.5	**74.8**	**72.7**	**73.7**	92.2	89.4	91.3
+ Joint-train & Self-link	**94.2**	**80.1**	**44.9**	**44.2**	**44.5**	74.5	71.7	73.1	**92.8**	90.2	**91.5**

Table 2: Experimental results on the Ubuntu test set. "T" suffix means the model uses only utterance text. "−T" indicates the model excludes utterance text. "Joint Train" indicates the model is trained with the joint learning objective (Eq. 15), "Self Link" indicate the model is decoded with self-link threshold re-adjustment.

challenge, the author (one of the task organizers) provided a trained model[3], which has two feed-forward layers. The input is 77 hand-engineered features combined with 128 dimension word average embeddings from pre-trained Glove. We will denote this model as FF model below.

Pointer Network. This is our model. For computational simplicity, we did not compute the attention over all the previous utterances, rather we set a fixed window size of 50. This means for the current utterance, we will calculate the attention with itself and 50 previous utterances during training and decoding. In our training data, about 97% of the utterances' parents are located in this window. In the #Ubuntu Data, according to the statistic of Table 1, one utterance only have 1.03 parent utterances on average. So, given an utterance we only predict its most likely parent.

4.4 Results

We present our main results in Table 2. For analysis purposes, in the table we also show the results for two variants of the models: (*i*) when the models consider only the utterance texts, as denoted by (T) suffix; (*ii*) when the models exclude the utterance text, as denoted by (−T) suffix. In addition, we present how the models perform specifically on self-links predictions, as correctly identifying self-links turns out to be quite crucial for identifying the conversations, as we will explain later. We also report the performance of the Siamese hierarchical convolutional neural network (SHCNN) from (Jiang et al., 2018) on #Ubuntu dataset. However, SHCNN mainly focuses on modeling message content representations and only incorporates four context features: speaker identicality, absolute time difference, and the number of duplicate words. So the performance is not as good as the feed-forward model with many hand-engineered features.

Link Prediction. We can see that our Pointer Network has better **link prediction** accuracy compared to the baseline, when it uses the message texts. The reason that the baseline performs slightly better in the absence of message texts is because it uses several meta features from the whole thread that capture more structural information. On the other hand, our model has access to only time and speaker information in the absence of message texts. Thus, we can say that our model can capture textual similarity or topical coherence better compared to the baseline.

Cluster (or Conversation) Prediction. Now if we compare the performance at the **cluster-level**, we see that our model performs much better when it uses only textual information compared to the baseline. In the absence of textual information, it performs on par with the baseline. However, when we compare the full model, we notice that its results are lower in some cluster-level measures (see 'Ptr-Net' results in the last block), which we did not expect given that it has higher accuracy on link prediction. Therefore, we performed a case

[3] https://github.com/dstc8-track2/NOESIS-II/tree/master/subtask4

Link Type	Percentage
System Messages	41%
Start of Conversation	36%
Isolated Messages	33%

Table 3: Self-link statistics on Ubuntu Dataset

Model	Cluster					Link		
	VI	1V1	P	R	F1	P	R	F1
FF	92.2	79.3	38.7	41.6	40.1	74.5	71.8	73.1
FF +G	93.6	76.4	53.4	53.5	53.5	77.0	74.2	75.6
Ptr-Net	92.3	70.2	33.0	38.9	36.0	75.4	72.7	74.1
Ptr-Net +G	94.5	79.3	55.1	54.7	54.9	78.4	75.6	77.0

Table 4: "+G" indicates replacing self-link prediction with ground truth labels.

study, and the study reveals that one particular kind of links, which we call "self-link", are very crucial to the cluster-level results.

Self-links. Kummerfeld et al. (2019) mention that most of the self-links are system messages like `"===zelot just join the channel"`. However, according to our statistics (shown in Table 3), only 41% self-links are system messages, and we have identified two other types of utterances that reply to themselves:

- *Start of a conversation:* These messages do not reply to any previous message but will be replied afterward.
- *Isolated Messages:* These are non-system messages, but reply to no previous message and never been replied afterwards.

Handling Self-links. To see how much our models get affected by inaccurate self-link predictions, we performed an experiment using ground truth self-links, where we replace those predictions with ground truth self-link. From Table 4, we see that although it shows only 3% improvement on overall link-prediction for our model, but for cluster prediction, it increases by 13% F1 for FF model and 19% for our model. The reason is if a self-link is predicted wrong and the utterance links to another conversation, it may destroy two true positive clusters. So the performance on self-link prediction could be a bottleneck for the clustering task.

Table 5 gives the detailed self-link prediction results for the FF model. This shows that the model

Link Type	P	R	F1
System Messages	99.0	100	99.5
Start of Topic	75.0	80.0	77.5
Isolated Messages	66.7	80.0	73.3

Table 5: Self-link prediction results for the baseline model (Kummerfeld et al., 2019).

can predict the system messages with almost 100% accuracy, but for other kinds of self-links, the performance is not that high. Experimental results in Table 2 show that our proposed Pointer Network has worse results on self-links compared to the FF model (compare "FF" and " Ptr-Net" in the forth and fifth block), which explains why our model does not perform well on clustering metrics.

When we do the simple adjustment to our decoding method by raising the threshold for self-link prediction, we see significant improvements in clustering. Note that this adjustment also improves the performance of the baseline (FF+Self-Link).

Joint Learning. The results in Table 2 show that joint learning further improves the clustering result. Combined with joint training, our online decoding algorithm achieves state-of-the-art results.

Ablation Studies. To show the importance of encoding textual and non-texture features like speaker and time, we trained the models with only textual features and without textual features; see the (T) (−T) variants in Table 2.

Intuitively, it is hard for the models to come up with a good prediction with only textual features, since the utterances in the same conversation usually talk about similar topics. This makes it difficult for the models to identify the right parent. Therefore, the results in Table 2 show a huge drop in performance when no speaker and time information are used and only textual features are used (see the first block of results with (T) suffix). This indicates that time and speaker mention information play a crucial role in disentanglement.

Similarly, the performance also goes down for the models when they do not consider textual information; see (−T) variants in Table 2. Although compared to the text only, the drop is less.

5 Conclusion

In this paper, we have proposed a novel online framework for disentangling multi-party conversa-

tions. In contrast to previous work, our method reduces the effort of complicated feature engineering by proposing an utterance encoder and a pointer module that models inter-utterance interactions. Moreover, we propose a joint-training framework that enables the pointer network to learn more contextual information. Link prediction in our framework is modeled as a pointing function with a multinomial distribution over previous utterances. We also show that our framework supports online decoding. Extensive experiments have been conducted on the #Ubuntu dataset, which show that our method achieves state-of-the-art performance on both link and conversation prediction tasks without using any handcrafted features.

There are some possible future directions from our work. We have shown in our experiments that self-link predictions have a significant impact on clustering results. This reminds us that neither our and most of the existing methods took good advantage of graph information in disentangling conversations. This can be done in two ways, encoding, and decoding. From the encoding side, it would be ideal to encode an utterance within its context. One challenge for this problem is that conversations are tangled, so sequential encoding methods like the one of (Sordoni et al., 2015) would not be appropriate. From the decoding side, a promising direction would be to make global inference in a more efficient way.

References

Erik Aumayr, Jeffrey Chan, and Conor Hayes. 2011. Reconstruction of threaded conversations in online discussion forums. In *Fifth International AAAI Conference on Weblogs and Social Media*.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Micha Elsner and Eugene Charniak. 2008. You talking to me? a corpus and algorithm for conversation disentanglement. In *Proceedings of ACL-08: HLT*, pages 834–842, Columbus, Ohio. Association for Computational Linguistics.

Micha Elsner and Eugene Charniak. 2010. Disentangling chat. *Computational Linguistics*, 36(3):389–409.

Micha Elsner and Eugene Charniak. 2011. Disentangling chat with local coherence models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1179–1189, Portland, Oregon, USA. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification*, 2(1):193–218.

Shenhao Jiang, Animesh Prasad, Min-Yen Kan, and Kazunari Sugiyama. 2018. Identifying emergent research trends by key authors and phrases. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 259–269, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Shafiq Joty, Giuseppe Carenini, Raymond Ng, and Gabriel Murray. 2019. Discourse analysis and its applications. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 12–17, Florence, Italy. Association for Computational Linguistics.

Seokhwan Kim, Michel Galley, Chulaka Gunasekara, Sungjin Lee, Adam Atkinson, Baolin Peng, Hannes Schulz, Jianfeng Gao, Jinchao Li, Mahmoud Adada, Minlie Huang, Luis Lastras, Jonathan K. Kummerfeld, Walter S. Lasecki, Chiori Hori, Anoop Cherian, Tim K. Marks, Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, and Raghav Gupta. 2019. The eighth dialog system technology challenge.

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

Jonathan K. Kummerfeld, Sai R. Gouravajhala, Joseph Peper, Vignesh Athreya, Chulaka Gunasekara, Jatin Ganhotra, Siva Sankalp Patel, Lazaros Polymenakos, and Walter S. Lasecki. 2019. A large-scale corpus for conversation disentanglement. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Changki Lee, Sangkeun Jung, and Cheon-Eum Park. 2017. Anaphora resolution with pointer networks. *Pattern Recognition Letters*, 95:1–7.

Xiang Lin, Shafiq Joty, Prathyusha Jwalapuram, and M Saiful Bari. 2019. A unified linear-time framework for sentence-level discourse parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4200, Florence, Italy. Association for Computational Linguistics.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Ryan Thomas Lowe, Nissan Pow, Iulian Vlad Serban, Laurent Charlin, Chia-Wei Liu, and Joelle Pineau. 2017. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse*, 8(1):31–65.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414, Melbourne, Australia. Association for Computational Linguistics.

Andrew McCallum and Ben Wellner. 2005. Conditional models of identity uncertainty with application to noun coreference. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 905–912. MIT Press.

Shikib Mehri and Giuseppe Carenini. 2017. Chat disentanglement: Identifying semantic reply relationships with random forests and recurrent neural networks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 615–623, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Marina Meilă. 2007. Comparing clusteringsan information based distance. *Journal of multivariate analysis*, 98(5):873–895.

Thanh-Tung Nguyen, Xuan-Phi Nguyen, Shafiq Joty, and Xiaoli Li. 2020. Efficient constituency parsing by pointing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, ACL'20, pages xx–xx, Seattle, USA. ACL.

Jacki O'Neill and David Martin. 2003. Text chat in action. In *Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work*, GROUP '03, pages 40–49, New York, NY, USA. ACM.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP'14*, pages 1532–1543, Doha, Qatar.

Dou Shen, Qiang Yang, Jian-Tao Sun, and Zheng Chen. 2006. Thread detection in dynamic text message streams. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 35–42, New York, NY, USA. ACM.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM.

Ming Tan, Dakuo Wang, Yupeng Gao, Haoyu Wang, Saloni Potdar, Xiaoxiao Guo, Shiyu Chang, and Mo Yu. 2019. Context-aware conversation thread detection in multi-party chat. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6455–6460, Hong Kong, China. Association for Computational Linguistics.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Hongning Wang, Chi Wang, ChengXiang Zhai, and Jiawei Han. 2011a. Learning online discussion structures by conditional random fields. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 435–444. ACM.

Li Wang, Marco Lui, Su Nam Kim, Joakim Nivre, and Timothy Baldwin. 2011b. Predicting thread discourse structure over technical web forums. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 13–25. Association for Computational Linguistics.

Lidan Wang and Douglas W Oard. 2009. Context-based message expansion for disentanglement of interleaved text conversations. In *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics*, pages 200–208. Association for Computational Linguistics.

Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 2020. Response selection for multi-party conversations with dynamic topic tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, EMNLP'20, pages XX–XX, Virtual. ACL.

Henghui Zhu, Feng Nan, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2019. Who did they respond to? conversation structure modeling using masked hierarchical transformer. *arXiv preprint arXiv:1911.10666*.