Context-Aware Conversation Thread Detection in Multi-Party Chat

Ming Tan [*†] **Dakuo Wang**[*‡] **Yupeng Gao**[*‡] **Haoyu Wang**[†]
Saloni Potdar[†] **Xiaoxiao Guo**[‡] **Shiyu Chang** [‡] **Mo Yu**[‡]
[†]IBM Watson [‡]IBM Research

Abstract

In multi-party chat, it is common for multiple conversations to occur concurrently, leading to intermingled conversation threads in chat logs. In this work, we propose a novel Context-Aware Thread Detection (CATD) model that automatically disentangles these conversation threads. We evaluate our model on three real-world datasets and demonstrate an overall improvement in thread detection accuracy over state-of-the-art benchmarks.

1 Introduction

In multi-party chat conversations, such as in Slack[1], multiple topics are often discussed at the same time (Wang et al., 2019; Zhang et al., 2017). For example, in Table 1, Alice and Bob talk about work, while Bob and Chuck chat about lunch, and this results in intermingled messages. Automatic conversational thread detection could be used to disentangle and group the messages into their respective topic threads. The resulting thread information could in turn be used to improve response relevancy for conversational agents (Shamekhi et al., 2018) or improve chat summarization quality (Zhang and Cranshaw, 2018).

Unlike most of today's Email and Forum systems that use threaded structure by default. However, the Instant Messaging systems (e.g., Slack) often require users to manually organize messages in threads. A recent study (Wang et al., 2019) found that users most likely do not manually create threads. On average, only 15.3 threads were created per Slack channel with 355 messages, when they discuss group projects.

Prior work on conversation thread disentanglement is often based on pairwise message comparison. Some solutions use unsupervised clustering methods with hand-engineered features (Wang and Oard, 2009; Shen et al., 2006), while others use supervised approaches with statistical (Du et al., 2017) or linguistic features (Wang et al., 2008; Wang and Rosé, 2010; Elsner and Charniak, 2008, 2010, 2011; Mayfield et al., 2012). Recent work by (Jiang et al., 2018; Mayfield et al., 2012) adopt deep learning approaches to compute message pair similarity, using a combination of message content and simple contextual features (e.g., authorship and timestamps).

Thread	Message
T1	Alice: Bob, please send me the proposal.
T2	Bob: Where are we going for lunch?
T1	Bob: Sent.
T1	Alice: Thanks.
T2 ?	Chuck: How about the Thai place?

Table 1: Messages and threads in a chat channel. Does Chuck's message belong to T1, T2, or a new thread?

However, linguistic theories (Biber and Conrad, 2019) differentiate the following three concepts: register, genre and style, to describe the text varieties. Register refers to the linguistic features such as the choice of words in content. Genre and Style refer to the conversational structure such as the sentence sequence and distribution. All aforementioned thread disentanglement methods fail to take into account the *contextual information of the thread*, or the *conversational flow and genre*.

A *thread's contextual information* is a useful feature for thread-detection because considering the relationship between a single new input message and an existing message alone may not be enough to accurately determine thread membership. Hence, using the full thread context history during comparison can improve pre-

[*] Equal contributions from the corresponding authors: mingtan@us.ibm.com, dakuowang@ibm.com, yupeng.gao@ibm.com.

[1] A instant messaging platform: https://slack.com

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 6456–6461,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

diction. Additionally, the *conversational flow and genre* may also be useful because (Butler et al., 2002) suggests this represents a conversation's signature. For example, we observe that users act distinctively in public Q&A (StackOverflow) and enterprise Q&A (IBM Social Q&A) online community (Wang et al., 2016), even when they are answering a similar question. Based on these hypotheses, we propose two context-aware thread detection (**CATD**) models. The first model (**CATD-MATCH**) captures contexts of existing threads and computes the distance between the context and the input message; the second model (**CATD-FLOW**) captures the conversational flow, and computes the language genre consistency while attaching the input message to a thread. We also combine them with a dynamic gate for further performance improvement, followed by an efficient beam search mechanism in the inference step. The evaluation proves our approach improves over the existing methods.

The contribution of this work is two-fold: 1) We propose context-aware deep learning models for thread detection and it advances the state-of-the-art; 2) Based on the dataset in (Jiang et al., 2018), we develop and release a more realistic multi-party multi-thread conversation dataset for future research.

2 Methodology

We model thread-detection as a topic detection and tracking task by deciding whether an incoming message starts a new thread or belongs to an existing thread (Allan, 2002). The goal is to assign each message m_i in the sequence, $M = \{m_i\}_{i=1}^N$, a thread label t_i, such that the complete thread label sequence $T = \{t_i\}_{i=1}^N$ contains multiple threads $(\mathcal{T}^1, \mathcal{T}^2, \cdots)$, where each thread $\mathcal{T}^l$ contains all messages with the same label. We denote $\mathcal{T}^l_{i-1} = \{m_{\delta(l,k)}\}$ as the l-th thread context already detected with $M_{i-1} = \{m_j\}_{j=1}^{i-1}$, and $\delta(l,k)$ is a function returning the index of the *last* k-th message of $\mathcal{T}^l_{i-1}$ in M_{i-1}. $m_{\delta(l,k)}$ is always before m_i, but may not be the last message m_{i-1}.

The training and inference steps are as follows: we train an LSTM-based thread classification model to obtain the membership of the message m_i to an existing- or a new thread, given the existing message sequence's thread tags $T_{i-1} = \{t_j\}_{j=1}^{i-1}$, which form L threads $\{\mathcal{T}^l_{i-1}\}_{l=1}^L$ (Subsection 2.1). During inference, we use this model to sequentially perform message thread labelling (Subsection 2.2).

2.1 Context-Aware Thread Detection

We first adopt the Universal Sentence Encoder[2] with deep averaging network (USE) (Cer et al., 2018) to get a static feature representation for each message in the form of sentence embeddings. We encode each message m_j as $\mathbf{enc}(m_j)$, by concatenating the USE output with two 20-dimensional embeddings: (1) **User-identity** difference between m_j and m_i. (2) **Time difference** by mapping the time difference between m_j and m_i into 11 ranges (from 1 minutes to 72 hours, details in Appendix A). These two features are also used in (Jiang et al., 2018), and another baseline model **GTM** uses only these features (Elsner and Charniak, 2008).

Given a message sequence M_{i-1}, which has been detected with L threads $\{\mathcal{T}^l_{i-1}\}_{l=1}^L$, we predict the thread tag $t_i \in \{\mathcal{T}^l_{i-1}\}_{l=1}^L \cup \{\mathcal{T}^{L+1}_{i-1}\}$ for m_i, where $\mathcal{T}^{L+1}_{i-1}$ indicates that m_i starts a new thread. As shown in Fig.1, we adopt a message-level single directional LSTM to encode each thread $\mathcal{T}^l_{i-1}$, whose inputs are $\mathbf{enc}(\cdot)$ of maximum K last messages (set to 20) in the thread, denoted as $\{m_{\delta(l,k)}\}_{k=1}^K$. The messages outside that window are viewed as irrelevant to the prediction of m_i. In Fig.1, we propose two CATD models, CATD-FLOW and CATD-MATCH, each one capturing the semantic relationship between the new message and the existing thread contexts.

CATD-FLOW (left in Fig.1): This model considers each thread as a conversation flow for a particular topic with its own genre, and the current message should belong to the thread with which it is more likely to form a fluent conversation. Therefore, we concatenate the $\mathbf{enc}(m_i)$ to each LSTM sequence of $\mathcal{T}^l_{i-1}$, and get the last output $\mathbf{e}^l_{\textbf{flow}}$ to dot-product a trainable vector $\mathbf{w}$ and compute the probability of m_i being labeled with $\mathcal{T}^l_{i-1}$,

$$P(t_i = \mathcal{T}^l_{i-1}|M_i, T_{i-1}) = \frac{\exp(\gamma \tanh(\mathbf{w} \cdot \mathbf{e}^l_{\textbf{flow}}))}{\sum_{\mathcal{T}^{l'}_{i-1}} \exp(\gamma \tanh(\mathbf{w} \cdot \mathbf{e}^{l'}_{\textbf{flow}}))} \qquad (1)$$

where γ is a scaling hyper-parameter. We differentiate $\mathcal{T}^{L+1}_{i-1}$ from other existing threads by introducing a parameter $\mathbf{u}$ as the only input for LSTM.

CATD-MATCH (right in Fig.1): An alternative view of determining the thread tag of m_i is to find

[2] https://tfhub.dev/google/universal-sentence-encoder/2

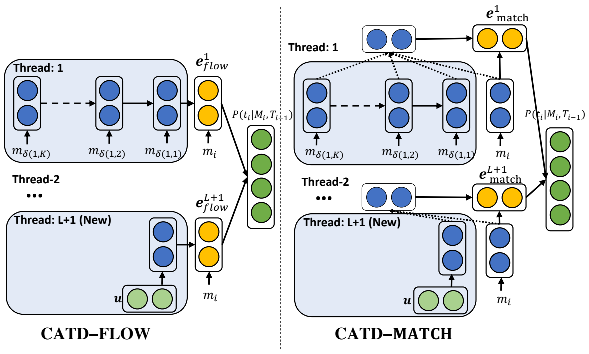

Figure 1: CATD Models: After being encoded by USE with user and time embeddings, ie. $\mathbf{enc}(\cdot)$, the last K messages for each thread $\mathcal{T}^l_{i-1}$ in history are encoded by an LSTM. **CATD-FLOW** concatenates the current message m_i as the final step of each LSTM, and gets the last output of LSTMs $\mathbf{e}^l_{\mathbf{flow}}$ to perform thread classification. **CATD-MATCH** runs LSTM on $\mathcal{T}^l_{i-1}$ and m_i separately, performs attention to obtain the context embeddings and then gets the matching vector $\mathbf{e}^l_{\mathbf{match}}$ for thread classification. For a newly-generated thread, we use a parameter $\mathbf{u}$ as the only input for its LSTM.

a thread $\mathcal{T}^l_{i-1}$ semantically closest to m_i. Thus we independently encode each thread and m_i with the parameter-shared LSTM (Only one LSTM step for m_i). In order to dynamically point to more related messages in the thread history, we use an one-way attention, which has been successfully adopted in many NLP tasks (Tan et al., 2016; Hermann et al., 2015). Specifically, given the sequence outputs of each thread's LSTM, $\{\mathbf{h}^l_{\delta(l,k)}\}^K_{k=1}$, we perform a weighted mean pooling to get a context embedding $\mathbf{e}^l_{cxt}$, attended by the one-step LSTM-encoded m_i, denoted as $\hat{\mathbf{h}}_i$.

$$\alpha^l_k = softmax(\mathbf{h}^l_{\delta(l,k)} \cdot \hat{\mathbf{h}}_i)_{k=[1,K]} \quad (2)$$

$$\mathbf{e}^l_{cxt} = \sum_k \alpha^l_k \mathbf{h}^l_{\delta(l,k)} \quad (3)$$

Next, we compute a matching vector $\mathbf{e}^l_{\mathbf{match}}$ below, which again computes dot-product $\mathbf{w}$ for classification. The function $\mathcal{N}(\mathbf{x})$ normalizes $\mathbf{x}$ by l_2 norm. $\otimes$ is element-wise multiplication.

$$\mathbf{e}^l_{\mathbf{match}} = \mathcal{N}(\mathbf{e}^l_{cxt}) \otimes \mathcal{N}(\hat{\mathbf{h}}_i) \quad (4)$$

CATD-COMBINE: Finally, we propose the dynamically-gated combination of FLOW and MATCH model, such that the $\mathbf{e}^l_{\mathbf{flow}}$ in Eq. 1 is replaced by a combination of the two models,

$$\mathbf{e}^l_{\mathbf{combine}} = (1-g^l)\mathbf{e}^l_{\mathbf{match}} + g^l\mathbf{e}^l_{\mathbf{flow}} \quad (5)$$

$$g^l = \text{sigmoid}\left(\left|\mathcal{N}(\mathbf{e}^l_{cxt}) - \mathcal{N}(\hat{\mathbf{h}}_i)\right| \cdot \mathbf{w}'\right) \quad (6)$$

where $\mathbf{w}'$ is a parameter vector. g^l is determined by the distance between $\mathcal{N}(\mathbf{e}^l_{cxt})$ and $\mathcal{N}(\hat{\mathbf{h}}_i)$. We use this dynamic gate g to linearly combine the two models. g is computed based on the difference of the MATCH vector of context and the input message. Intuitively, if they are close, both FLOW and MATCH will be considered equally for prediction. Otherwise, the model dynamically computes the weights of MATCH and FLOW.

Training Procedure: Following (Jiang et al., 2018), apart from a new thread, we consider the candidate threads (**Active Threads**) in Eq. 1 only from those appearing in one hour time-frame before m_i. During training, we treat the messages of a channel as a single sequence, and optimize Eq. 1 with training examples, containing m_i and its active threads. Though messages are sorted by time, the training examples are shuffled during training.

2.2 Thread Inference

During inference, we want to find the optimal thread labeling by maximizing:

$$\log P(T|M) = \sum_{i=1}^{N} \log P(t_i|M_i, T_{i-1}) \quad (7)$$

where t_i are selected from active threads and the new thread. However, searching the entire space of T is unfeasible. Hence, we resort to **Beam Search**, a generalized version of greedy search. It predicts sequentially from m_1 to m_N, while keeping B states in the beam. For each m_i, each state in the beam is a candidate T_{i-1}. Each new state T_i is ranked after labeling t_i for m_i:

$$\begin{aligned}\max_{\mathcal{T}^l_{i-1}} P(T_i|M_i) &= \max_{\mathcal{T}^l_{i-1}}[P(t_i = \mathcal{T}^l_{i-1}|M_i, T_{i-1}) \\ &\quad \times \; P(T_{i-1}|M_{i-1})] \end{aligned} \quad (8)$$

where $\mathcal{T}^l_{i-1}$ is selected from the active threads in the previous state and a new thread tag. The new states with scores lower than top-B candidates are discarded. Similar to training, the active threads are also pruned by the "one-hour" constraint. However, they are not extracted from the ground-truth, but from previously-detected threads.

3 Experiments

Datasets: We conduct extensive experiments on three publicly available datasets from Reddit datasets. We strictly follow (Jiang et al., 2018) to construct our data. Comments under a post can be treated as messages in a single conversational thread, and we merge all comments in a

	Gadgets			Iphones			Politics		
	NMI	ARI	F1	NMI	ARI	F1	NMI	ARI	F1
GTM	.662	.319	.344	.685	.323	.343	.798	.032	.032
CISIR-SHCNN	.668	.328	.359	.697	.345	.364	.773	.191	.196
CISIR-USE	.661	.334	.364	.699	.333	.354	.760	.182	.187
CATD-FLOW	.688	.374	.400	.740	.410	.424	.826	.420	.423
CATD-MATCH	**.703**	.385	.404	.740	.428	.441	.831	.427	.430
CATD-COMB.	.694	**.395**	**.413**	**.750**	**.434**	**.445**	**.834**	**.461**	**.464**

Table 2: CATD models are compared with baselines wrt. metrics of NMI, ARI and F1 for the three datasets

sub-reddit to construct a synthetic dataset of interleaved conversations. We take three sub-reddits to build three datasets, Gadgets, IPhones and Politics. [3] The data statistics and examples are shown in Appendix B.

Reddit Dataset Improvement: We use the same pre-processing method in (Jiang et al., 2018): we discard the messages which have less than 10 words or more than 100 words. Conversations less than 10 messages are also discarded. We guarantee that no more than 10 conversations happen at the same time. In their work, different message pairs of the same thread might be included in both train and test sets. Instead, we split the datasets on the thread level because in realistic settings, test threads should be completely unseen in train set. [4]

Experimental Setup: We use Adam (Kingma and Ba, 2015) to optimize the training objective (Eq. 1). During training, we fix γ in Eq. 1 as 10. In inference, this value may influence the search quality. We set it as 20.0 by the validation accuracy on Politics. We set LSTM output dimensions to 400, the batch size to 10 and the beam size to 5 by default. We train 50 epochs and select the model with the best validation-set performance.

Baseline: (1) CISIR-SHCNN (Jiang et al., 2018): A recently proposed model based on CNN and ranking message pairs. (2) CISIR-USE: We replace CNN encoder in CISIR with a USE to test the effect of different sentence encoders. (3) GTM (Elsner and Charniak, 2008): A graph-theoretical model with chat and content specific features.

	Model Variations	NMI	ARI	F1
A	COMBINE (share)	.832	.420	.422
B	COMBINE (concat)	.828	.446	.448
C	SPLIT	.824	.417	.420
D	FLOW (K=5)	.813	.395	.398
E	FLOW (K=10)	.823	.414	.417
F	FLOW (K=20)	.826	.420	.423
G	MATCH (K=5)	.820	.399	.402
H	MATCH (K=10)	.823	.405	.408
I	MATCH (K=20)	.831	.427	.430
J	MATCH (K=20, bi-LSTM)	.832	.428	.430
K	COMBINE (K=5)	.811	.378	.381
L	COMBINE (K=10)	.822	.403	.405
M	COMBINE (K=20, B=1)	.828	.452	.455
N	COMBINE (K=20, B=5)	**.834**	**.461**	**.464**
O	COMBINE (K=20, B=10)	.833	.431	.433

Table 3: Analysis on Politics dataset

Evaluation Metrics: Normalized mutual information (NMI), Adjusted rand index (ARI) and F1 score, following (Jiang et al., 2018). F1 is computed based on all message pairs in a test set. Also, following their work, we assume the candidate threads of each message for our models and baselines are obtained from the ones which have messages in the previous hour. For examples, the CISIR-SHCNN models will take pairs only within the one-hour frame.

Main Results: Table 2 compares the CATD models and baselines on NMI, ARI and F1. CISIR models are generally better than non-deep-learning GTM. There is a clear gap between CISIR-USE and our proposed models, which proves our models' improvement is not due to the usage of USE but the new model structures. CATD models are significantly superior to all baselines and CATD-COMBINE generally performs best. Specifically, all baselines failed on Politics, probably because there are more threads in Politics than the other two datasets (see Appendix B), making disentanglement more difficult. But CATD models achieve better results because they encode ac-

[3] We did not use the IRC dataset from (Jiang et al., 2018), because of its extremely small number of data, 39 threads and 491 messages.

[4] This dataset is released in https://github.com/SLAD-ml/thread_detection_data

tive threads in parallel, while considering longer history in each thread.

Analysis : In Table 3, we analyze our models on Politics, the largest dataset. First, we examine the effect of K. For all CATD models, with K from 5 to 20 (D-F, G-I, K, L and N), and all metrics improve, showing the importance of the longer history in LSTM. Second, we adopt bidirectional LSTMs (J) for CATD-MATCH, without an obvious improvement, probably because most messages in the datasets can be fully comprehended only with previous history. This assumption is consistent with a mild improvement when we increase beam size from 1 (M) to 5 (N). We see a lower ARI with beam size as 10 (O), because of the incorrect candidates at lower ranking positions. Finally, the models are generally good when beam=1, enabling an "online" detection without knowing the future messages, which can not be directly fulfilled by most pairwise prior work.

	Model Variations	NMI	ARI	F1
A	COMBINE (share)	.699	.403	.412
B	COMBINE (concat)	.675	.366	.396
C	SPLIT	.692	.359	.382
D	FLOW (K=5)	.676	.368	.398
E	FLOW (K=10)	.690	.354	.386
F	FLOW (K=20)	.688	.374	.400
G	MATCH (K=5)	.667	.366	.396
H	MATCH (K=10)	.674	.368	.399
I	MATCH (K=20)	.703	.385	.404
J	MATCH (K=20, bi-LSTM)	**.707**	**.405**	.412
K	COMBINE (K=5)	.686	.312	.330
L	COMBINE (K=10)	.687	.369	.399
M	COMBINE (K=20, B=1)	.694	.395	.411
N	COMBINE (K=20, B=5)	.694	.395	.413
O	COMBINE (K=20, B=10)	.692	.394	**.414**

Table 4: Analysis on Gadgets dataset

Comparison with Model Variations: In Table 3, we also shared the LSTM parameters for MATCH and FLOW models (A), with 4% drop on ARI. This is because we need two independent LSTMs to capture different linguistic features. Next, we combine FLOW and MATCH (B) by concatenating $\mathbf{e}^l_{\mathbf{flow}}$ and $\mathbf{e}^l_{\mathbf{match}}$, resulting in 1.5% drop on ARI, which proves the benefit of the gate in CATD-COMBINE. Also, we break the links between LSTM nodes and perform one-step LSTM on all the history messages (C), leading to over 4% drop on ARI. This reflects the necessity of a RNN encoding inter-messages information.

In Table 4 and 5, we show the analysis for for Gadgets and Iphones datasets similar to Politics dataset in Table 3. As compared to Politics, we observe that for Gadgets and Iphones, CATD-FLOW models have some fluctuations in performance when we increase K from 5 to 20, which may be due to the limited capability of LSTMs for memorizing long-term history. This issue is more prevalent when the training data size is small.

	Model Variations	NMI	ARI	F1
A	COMBINE (share)	.748	.415	.427
B	COMBINE (concat)	.745	.419	.433
C	SPLIT	.749	.418	.433
D	FLOW (K=5)	.742	.411	.425
E	FLOW (K=10)	.743	.434	.446
F	FLOW (K=20)	.740	.410	.424
G	MATCH (K=5)	.744	.385	.398
H	MATCH (K=10)	.744	.409	.420
I	MATCH (K=20)	.740	.428	.441
J	MATCH (K=20, bi-LSTM)	.739	.430	.445
K	COMBINE (K=5)	.751	.363	.374
L	COMBINE (K=10)	.751	.419	.429
M	COMBINE (K=20, B=1)	.750	.431	.444
N	COMBINE (K=20, B=5)	.750	.434	.445
O	COMBINE (K=20, B=10)	**.750**	**.434**	**445**

Table 5: Analysis on Iphones dataset

4 Conclusion

We propose context-aware thread detection models to perform thread detection for multi-party chat conversations which take into account threads' contextual information. These are integrated into an efficient beam search for inference. Our proposed method advances the state-of-the-art.

References

James Allan. 2002. Introduction to topic detection and tracking. In *Topic detection and tracking*, pages 1–16.

Douglas Biber and Susan Conrad. 2019. *Register, genre, and style*. Cambridge University Press.

Brian Butler, Lee Sproull, Sara Kiesler, and Robert Kraut. 2002. Community effort in online groups: Who does the work and why. *Leadership at a distance: Research in technologically supported work*, 1:171–194.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder.

Wenchao Du, Pascal Poupart, and Wei Xu. 2017. Discovering conversational dependencies between messages in dialogs. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Micha Elsner and Eugene Charniak. 2008. You talking to me? a corpus and algorithm for conversation disentanglement. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 834–842.

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.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701.

Jyun-Yu Jiang, Francine Chen, Yan-Ying Chen, and Wei Wang. 2018. Learning to disentangle interleaved conversational threads with a siamese hierarchical network and similarity ranking. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1812–1822.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations.*

Elijah Mayfield, David Adamson, and Carolyn Penstein Rosé. 2012. Hierarchical conversation structure prediction in multi-party chat. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 60–69. Association for Computational Linguistics.

Ameneh Shamekhi, Q Vera Liao, Dakuo Wang, Rachel KE Bellamy, and Thomas Erickson. 2018. Face value? In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 391. ACM.

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.

Ming Tan, Cicero Dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 464–473.

Dakuo Wang, Youyang Hou, Lin Luo, and Yingxin Pan. 2016. Answerer engagement in an enterprise social question & answering system. *IConference 2016 Proceedings.*

Dakuo Wang, Haoyu Wang, Mo Yu, Zahra Ashktorab, and Ming Tan. 2019. Slack channels ecology in enterprises: How employees collaborate through group chat. *arXiv preprint arXiv:1906.01756.*

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.

Yi-Chia Wang, Mahesh Joshi, William W Cohen, and Carolyn Penstein Rosé. 2008. Recovering implicit thread structure in newsgroup style conversations. In *ICWSM*.

Yi-Chia Wang and Carolyn P Rosé. 2010. Making conversational structure explicit: identification of initiation-response pairs within online discussions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 673–676. Association for Computational Linguistics.

Amy Zhang, Bryan Culbertson, and Praveen Paritosh. 2017. Characterizing online discussion using coarse discourse sequences. *11th AAAI International Conference on Web and Social Media (ICWSM).*

Amy X Zhang and Justin Cranshaw. 2018. Making sense of group chat through collaborative tagging and summarization. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):196.