Disentangling Online Chats with DAG-Structured LSTMs

Duccio Pappadopulo[*1], **Lisa Bauer**[*2], **Marco Farina**[1], **Ozan İrsoy**[1], **Mohit Bansal**[2]
[1]Bloomberg [2]UNC Chapel Hill
{dpappadopulo, mfarina19, oirsoy}@bloomberg.net
{lbauer6, mbansal}@cs.unc.edu

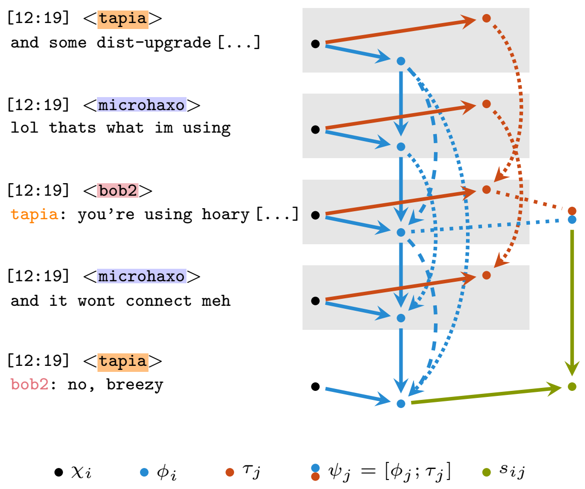
Abstract

Many modern messaging systems allow fast and synchronous textual communication among many users. The resulting sequence of messages hides a more complicated structure in which independent sub-conversations are interwoven with one another. This poses a challenge for any task aiming to understand the content of the chat logs or gather information from them. The ability to disentangle these conversations is then tantamount to the success of many downstream tasks such as summarization and question answering. Structured information accompanying the text such as user turn, user mentions, timestamps, is used as a cue by the participants themselves who need to follow the conversation and has been shown to be important for disentanglement. DAG-LSTMs, a generalization of Tree-LSTMs that can handle directed acyclic dependencies, are a natural way to incorporate such information and its non-sequential nature. In this paper, we apply DAG-LSTMs to the conversation disentanglement task. We perform our experiments on the Ubuntu IRC dataset. We show that the novel model we propose achieves state of the art status on the task of recovering *reply-to* relations and it is competitive on other disentanglement metrics.

1 Introduction

Online chat and text messaging systems like Facebook Messenger, Slack, WeChat, WhatsApp, are common tools used by people to communicate in groups and in real time. In these venues multiple independent conversations often occur simultaneously with their individual utterances interspersed.

It is reasonable to assume the existence of an underlying *thread* structure partitioning the full conversation into disjoint sets of utterances, which ideally represent independent sub-conversations.



```
[12:19] <tapia>
and some dist-upgrade [...]

[12:19] <microhaxo>
lol thats what im using

[12:19] <bob2>
tapia: you're using hoary [...]

[12:19] <microhaxo>
and it wont connect meh

[12:19] <tapia>
bob2: no, breezy
```

Figure 1: Excerpt from the IRC dataset (*left*) and our reply-to classifier architecture (*right*). Blue dots represent a unidirectional DAG-LSTM unit processing the states coming from the children of the current node. Red dots represent the GRU units performing thread encoding. At this point in time, we are computing the score (log-odds) of fifth utterance replying to the third.

The task of identifying these sub-units, *disentanglement*, is a prerequisite for further downstream tasks among which question answering, summarization, and topic modeling (Traum et al., 2004; Shen et al., 2006; Adams and Martell, 2008; Elsner and Charniak, 2010). Additional structure can generally be found in these logs, as a particular utterance could be a response or a continuation of a previous one. Such *reply-to* relationships implicitly define threads as the connected components of the resulting graph topology, and can then be used for disentanglement (Mehri and Carenini, 2017; Dulceanu, 2016; Wang et al., 2008; Gaoyang Guo et al., 2018).

Modeling work on conversation disentanglement spans more than a decade. Elsner and Charniak (2008, 2010) use feature based linear models to find

* Equal contribution

pairs of utterances belonging to the same thread and heuristic global algorithms to assign posts to threads. Mehri and Carenini (2017) and Jiang et al. (2018), while also adopting similar heuristics, use features extracted through neural models, LSTMSs (Hochreiter and Schmidhuber, 1997) and siamese CNNs (Bromley et al., 1993) respectively. Wang et al. (2011) follow a different approach by modeling the interactions between the predicted reply-to relations as a conditional random field.

One challenge in building automatic systems that perform disentanglement is the scarcity of large annotated datasets to be used to train expressive models. A remarkable effort in this direction is the work of Kummerfeld et al. (2019a) and the release of a dataset containing more that $77k$ utterances from the IRC `#Ubuntu` channel with annotated reply-to structure. In the same paper, it is shown how a set of simple handcrafted features, pooling of utterances GloVe embeddings (Pennington et al., 2014), and a feed-forward classifier can achieve good performances on the disentanglement task. Most of the follow-up work on the dataset relies on BERT (Devlin et al., 2019) embeddings to generate utterance representations (Zhu et al., 2020; Gu et al., 2020; Li et al., 2020). Zhu et al. (2020) use an additional transformer module to contextualize these representations, while Gu et al. (2020); Li et al. (2020) use an LSTM. Two exceptions are Liu et al. (2020), which models thread membership in an online fashion and discards reply-to relationships, and the recent Yu and Joty (2020a) which uses pointer networks (Vinyals et al., 2015).

In this short paper, we use DAG-structured LSTMs (İrsoy et al., 2019) to study disentanglement. As a generalization of Tree-LSTMs (Tai et al., 2015a), DAG-LSTMs allow to faithfully represent the structure of a conversation, which is more properly described as a directed acyclic graph (DAG) than a sequence. Furthermore, DAG-LSTMs allow for the systematic inclusion of structured information like user turn and mentions in the learned representation of the conversation context. We enrich the representation learned by the DAG-LSTM by concatenating to it a representation of the thread to which the utterance belongs. This *thread encoding* is obtained by means of a GRU unit (Cho et al., 2014) and captures thread specific features like style, topic, or persona. Finally we manually construct new features to improve username matching, which is crucial for detecting user mentions, one of the most important features for disentanglement.

Our results are summarized in Table 1. The DAG-LSTM significantly outperforms the BiLSTM baseline. Ablation studies show the importance of the new features we introduce. When augmented by thread encoding and a careful handling of posts predicted to be thread starters, the DAG-LSTM architecture achieves state of the art performances on *reply-to* relation extraction on the IRC Ubuntu dataset and it is competitive on the other metrics which are relevant to disentanglement.

2 Methodology

2.1 Problem Statement

A multi-party chat C is a sequence of posts $(c_i)_i$, $i = 1, \dots, |C|$. For each *query* post c_i we look for the set of *link* posts $\mathcal{R}(c_i)$ such that c_i replies to, or *links to*, c_j for $c_j \in \mathcal{R}(c_i)$. When a post c is a conversation starter we define, consistently with Kummerfeld et al. (2019a), $\mathcal{R}(c) = \{c\}$, that is c replies to itself, it is a *self-link*. This reply-to binary relation defines a DAG over C. By taking the union of the reply-to relation with its converse and by calculating its transitive closure, we obtain an equivalence relation on C whose equivalence classes are threads, thus solving the disentanglement problem.

We frame the problem as a sequence classification task. For each query post c_i we consider its L preceding posts $\mathcal{O}_{c_i} \equiv \{c_{i-L-1}, \dots, c_i\}$ and predict one of them as its link. In the IRC Ubuntu dataset, predicting a single link per query post is a good approximation, holding true for more than 95% of the annotated utterances. We use $L = 50$ in the following. As described in Sections 2.2 and 2.3, for each query utterance c_i, we construct a contextualized representation, $\phi_i \equiv \phi(c_i, C)$. We do the same for each of the links $c_j \in \mathcal{O}_{c_i}$, using a representation ψ that can in principle differ from ϕ. We then calculate $p(c_i \text{ replies-to } c_j) \equiv p(c_j|c_i)$ as

$$p(c_j|c_i) \equiv \frac{\exp(s_{ij})}{\sum_{c_k \in \mathcal{O}_{c_i}} \exp(s_{ik})}, \qquad (1)$$

where $s_{ij} \equiv s(\phi_i, \psi_j, f_{ij})$ is a real-valued scoring function described in Section 2.4 and f_{ij} are additional features. The parameters of the resulting model are learned by maximizing the likelihood associated to Eq. 1. At inference time we predict $\hat{j} = \text{argmax}_{c_j \in \mathcal{O}_{c_i}} p(c_j|c_i)$.

2.2 Contextual Post Representation

The construction of the ϕ and ψ representations closely follows İrsoy et al. (2019). Every post c_i is represented as a sequence of tokens $(t^i_n)_n$. An embedding layer maps the tokens to a sequence of d_I-dimensional real vectors $(\omega^i_n)_n$. We use the tokenizer and the word embeddings from Kummerfeld et al. (2019a), $d_I = 50$. We generate a representation χ_i of c_i by means of a single BiLSTM layer unrolled over the sequence of the token embeddings $(\upsilon^i_n)_n \equiv \text{BiLSTM}[(\omega^i_n)_n]$ followed by elementwise max-affine pooling $\chi_i = \max_n \text{Affine}[(\upsilon^i_n)_n]$.

To obtain the contextualized representations ϕ, we use a DAG-LSTM layer. This is an N-ary Tree-LSTM (Tai et al., 2015a) in which the sum over children in the recursive definition of the memory cell is replaced with an elementwise max operation (see Appendix). This allows the existence of multiple paths between two nodes (as it is the case if a node has multiple children) without the associated state explosion (İrsoy et al., 2019). This is crucial to handle long sequences, as in our case.

At each time step the DAG-LSTM unit receives the utterance representation χ_i of the current post c_i as the input and all the hidden and cell states coming from a labeled set of children, $\mathcal{C}(c_i)$, see Figure 1. In our case $\mathcal{C}(c_i)$ contains three elements: the previous post in the conversation (c_{i-1}), the previous post by the same user of c_i, the previous post by the user mentioned in c_i if any. More dependencies can be easily added making this architecture well suited to handle structured information. The DAG-LSTM is unrolled over the sequence $(\{\chi_i, \mathcal{C}(c_i)\})_i$, providing a sequence of contextualized post representations $(\phi_i)_i$. We also consider a bidirectional DAG-LSTM defined by a second unit processing the reversed sequence $\tilde{c}_i \equiv c_{|C|-i+1}$. Forward and backward DAG-LSTM representations are then concatenated to obtain ϕ.

2.3 Thread Encoding

The link post representation ψ can coincide with the query one, $\psi_j \equiv \phi_j$. One potential issue with this approach is that ψ does not depend on past thread assignments. Furthermore, thread-specific features such as topic and persona, cannot be easily captured by the hierarchical but sequential model described in the previous section. Thus we augment the link representations by means of *thread encoding* (Liu et al., 2020). Given a query, c_i, and a link c_j posts pair, we consider the thread $\mathcal{T}(c_j) = (c_{t_i})$, $t_i < t_{i+1}$, $t_{|\mathcal{T}(c_j)|} = j$, to which c_j has been assigned. We construct a representation τ_j of such thread by means of a GRU cell, $\tau_j = \text{GRU}[(\chi(c))_{c\in\mathcal{T}(c_j)}]$. ψ_j is then obtained by concatenating ϕ_j and τ_j. At training time we use the gold threads to generate the τ representations, while at evaluation time we use the predicted ones.

2.4 Scoring Function

Once query and link representations are constructed we use the scoring function in Eq. 1 to score each link against the query utterance, with s a three-layer feed-forward neural network. The input of the network is the concatenation $[\phi_i; \psi_j; f_{ij}]$, where f_{ij} are the 77 features introduced by Kummerfeld et al. (2019a). We augment them by 42 additional features based on Levenshtein distance and longest common prefix between query's username and words in the link utterance (and viceversa). These are introduced to improve mention detection by being more lenient on spelling mistakes (see 2.5 for precise definitions).

2.5 User Features

While IRC chats allow systematically tagging other participants (a single mention per post), users can address each other explicitly by typing usernames. This allows for abbreviations and typos to be introduced, which are not efficiently captured by the set of features used by Kummerfeld et al. (2019b). To ameliorate this problem we construct additional features. Given a pair of utterances c_1 and c_2 we define the following:

- Smallest Levenshtein distance (D_L) between $c_1(c_2)$'s username and each of the word in $c_2(c_1)$; 5 bins, $D_L = i$ for $i = 0, \dots, 4$ or $D_L > 4$.

- Largest length of common prefix (ℓ) between $c_1(c_2)$'s username and each of the word in $c_2(c_1)$; 5 bins, $\ell = i$ for $i = 3, \dots, 6$ or $\ell > 6$.

- Binary variable indicating whether $c_1(c_2)$'s username is a prefix of any of the words in $c_2(c_1)$.

These amount to a total of 42 additional features for each pair of posts.

Model	Graph			Cluster		
	P	R	F	P	R	F
Kummerfeld et al.	73.7	71.0	72.3	34.6	38.0	36.2
Zhu et al.*	73.2	69.2	70.6	35.8	32.7	34.2
Li et al.*				42.3	<u>46.2</u>	44.1
Yu and Joty	74.7	72.7	73.7	33.0	38.9	36.0
+ self.	<u>74.8</u>	<u>72.7</u>	<u>73.7</u>	42.2	40.9	41.5
+ joint train, self.	74.5	71.7	73.1	<u>44.9</u>	44.2	<u>44.5</u>
BiLSTM (↓)	73.9	71.2	72.5	31.3	37.5	34.1
DAG-LSTM	74.9	72.2	73.6	37.3	42.3	39.6
− user features (↓)	74.0	71.3	72.6	33.6	39.7	36.4
− mention link	74.5	71.8	73.1	33.5	38.3	35.7
+ self. (↑)	74.9	72.6	73.8	41.1	41.1	41.1
+ thread enc.	75.0	72.3	73.7	37.3	**42.5**	39.7
+ thread enc., self.	**75.2**	**72.7**	**73.9**	**42.4**	41.7	**42.0**

Table 1: Results of our experiments (*bottom*, best in bold) and literature (*top*, best underlined). The ↑(↓) sign indicates the model being significantly better (worse) ($p < 0.05$) than the DAG-LSTM entry based on a McNemar test (McNemar, 1947) conducted on the *test* set. User features and mention links are included in this baseline model, thread encoding and self-link threshold tuning are not. Starred entries use contextual embeddings.

Model	Self-links		
	P	R	F
BiLSTM	79.6	94.6	86.5
DAG-LSTM	82.8	93.8	88.0
+ self-links threshold	87.7	92.4	90.0
DAG-LSTM + thread enc.	81.4	93.8	87.2
+ self-links threshold	89.8	90.6	90.2

Table 2: Thread starters (*self-links*) performances for our models in Table 1, before and after thresholding.

3 Results

3.1 Evaluation

We conduct our experiments on the Ubuntu IRC dataset for disentanglement (Kummerfeld et al., 2019a; Kim et al., 2019). We focus on two evaluation metrics defined in Kummerfeld et al. (2019a): *graph* F_1, the F-score calculated using the number of correctly predicted reply-to pairs; *cluster* F_1, the F-score calculated using the number of matching threads of length greater than 1.

3.2 Experiments

As a baseline, we use a BiLSTM model in which $\phi_i(=\psi_i)$ is obtained as the hidden states of a bidirectional LSTM unrolled over the sequence $(\chi_i)_i$. The base DAG-LSTM model uses both username and mentions to define the children set $\mathcal{C}$ of an utterance. Bidirectionality is left as a hyperparameter. All our experiments use the same architecture from section 2 to construct the utterance representation χ. We train each model by minimizing the negative log-likelihood for Eq. 1 using Adam optimizer (Kingma and Ba, 2019). We tune the hyperparameters of each architecture through random search.[1] Table 1 shows the test set performances of the models which achieve the best graph F_1 score over the dev set. Optimizing graph over cluster score is motivated by an observation: dev set cluster F_1 score displays a much larger variance than graph F_1 score, which is roughly four-fold after subtracting the score rolling average. By picking the iteration with the best cluster F_1 score we would be more exposed to fluctuation and to worse generalization, which we observe.

[1]We refer to the Appendix for details.

3.3 Self-Links Threshold Tuning

As noted by Yu and Joty (2020b), the ability of the model to detect self-links is crucial for its final performances. In line with their findings, we also report that all our models are skewed towards high recall for self-link detection (Table 2).

To help with this, we introduce two thresholds θ and δ, which we compare with $\hat{p}$, the argmax probability Eq. 1, and Δp, the difference between the top-2 predicted probabilities. Whenever the argmax is a self-link: if $p < \theta$, we predict the next-to-argmax link, otherwise we predict both the top-2 links if also $\Delta\hat{p} < \delta$. On the dev set, we first fine-tune θ to maximize the self-link F_1 score and the fine-tune δ to maximize the cluster F_1 score.

3.4 Results Discussion

Table 1 shows our main results. Our DAG-LSTM model significantly outperforms the BiLSTM baseline. We perform ablation studies on our best DAG-LSTM model showing that while both user features and mention link provide a performance improvement for both cluster and graph score, only user features ablation results in a significant change. Self-links threshold tuning improves performances, particularly on cluster score for both models, highlighting the importance of correctly identifying thread starters.

The DAG-LSTM model with thread encoding achieves state of the art performances in predicting *reply-to* relations. This is particularly interesting especially when we compare with models employing contextual embeddings like Zhu et al. (2020). For the cluster scores, the best model is the pointer network model of Yu and Joty (2020a), which is anyway within less than 0.5% of the best contextual model, and within 2.5% of our model. The difference mainly arises from a difference in recall and corresponds to an absolute difference of less than 10 true positive clusters on the test set. Further comparisons with existing literature are limited by code not being available at the moment.

4 Conclusions

In this paper we apply, for the first time, DAG-LSTMs to the disentanglement task; they provide a flexible architecture that allows to incorporate into the learned neural representations the structured information which comes alongside multi-turn dialogue. We propose thread encoding and a new set of features to aid identification of user mentions.

There are possible directions left to explore. We modeled the reply-to relationships in a conversation by making an assumption of conditional independence of reply-to assignments. This is possibly a poor approximation and it would be interesting to lift it. A challenge with this approach is the computational complexity resulting from the large dimension of the output space of the reply-to classifier. We notice that thread encoding allows a non-greedy decoding strategy through beam search which would be interesting to further explore.

Acknowledgments

We thank the reviewers for their useful feedback. We thank Andy Liu, Camilo Ortiz, Huayan Zhong, Philipp Meerkamp, Rakesh Gosangi, Raymond (Haimin) Zhang for their initial collaboration. We thank Jonathan Kummerfeld for discussions. This work was performed while Lisa interned at Bloomberg, and was later supported by DARPA MCS Grant N66001-19-2-4031, NSF-CAREER Award 1846185, and a NSF PhD Fellowship. The views are those of the authors and not of the funding agencies.

References

P. H. Adams and C. H. Martell. 2008. Topic Detection and Extraction in Chat. In *2008 IEEE International Conference on Semantic Computing*, pages 581–588.

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature Verification Using a "Siamese" Time Delay Neural Network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Andrei Dulceanu. 2016. Recovering Implicit Thread Structure in Chat Conversations. *Romanian Journal of Human-Computer Interaction*, 9(3).

Micha Elsner and Eugene Charniak. 2008. You Talking to Me? a Corpus and Algorithm for Conversation Disentanglement. *ACL-08: HLT*, page 834.

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

Gaoyang Guo, Chaokun Wang, Jun Chen, Pengcheng Ge, and Weijun Chen. 2018. Who is Answering Whom? finding "Reply-To" Relations in Group Chats with Deep Bidirectional LSTM Networks. *Cluster Computing*, pages 1–12.

Jia-Chen Gu, Tianda Li, Quan Liu, Xiaodan Zhu, Zhen-Hua Ling, and Yu-Ping Ruan. 2020. Pre-Trained and Attention-Based Neural Networks for Building Noetic Task-Oriented Dialogue Systems. In *AAAI 2020 Workshop on The Eighth Dialog System Technology Challenge*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780.

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.

Seokhwan Kim, Michel Galley, Chulaka Gunasekara, Sungjin Lee, Adam Atkinson, Baolin Peng, Hannes Schulz, Jianfeng Gao, Jinchao Li, Mahmoud Adada, et al. 2019. The Eighth Dialog System Technology Challenge. *arXiv preprint arXiv:1911.06394*.

Diederik P Kingma and J Adam Ba. 2019. A Method for Stochastic Optimization. arxiv 2014. *arXiv preprint arXiv:1412.6980*, 434.

Jonathan K Kummerfeld, Sai R Gouravajhala, Joseph J Peper, Vignesh Athreya, Chulaka Gunasekara, Jatin Ganhotra, Siva Sankalp Patel, Lazaros C Polymenakos, and Walter Lasecki. 2019a. A Large-Scale Corpus for Conversation Disentanglement. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3846–3856.

Jonathan K. Kummerfeld, Sai R. Gouravajhala, Joseph J. Peper, Vignesh Athreya, Chulaka Gunasekara, Jatin Ganhotra, Siva Sankalp Patel, Lazaros C Polymenakos, and Walter Lasecki. 2019b. A large-scale corpus for conversation disentanglement. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3846–3856, Florence, Italy. Association for Computational Linguistics.

Tianda Li, Jia-Chen Gu, Xiaodan Zhu, Quan Liu, Zhen-Hua Ling, Zhiming Su, and Si Wei. 2020. Dialbert: A Hierarchical Pre-Trained Model for Conversation Disentanglement. *CoRR*, abs/2004.03760.

Hui Liu, Zhan Shi, Jia-Chen Gu, Quan Liu, Si Wei, and Xiaodan Zhu. 2020. End-to-End Transition-Based Online Dialogue Disentanglement. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3868–3874. International Joint Conferences on Artificial Intelligence Organization. Main track.

Quinn McNemar. 1947. Note on the Sampling Error of the Difference between Correlated Proportions or Percentages. *Psychometrika*, 12(2):153–157.

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.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

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, page 35–42, New York, NY, USA. Association for Computing Machinery.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015a. Improved Semantic Representations from Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015b. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

David R. Traum, Susan Robinson, and Jens Stephan. 2004. Evaluation of multi-party virtual reality dialogue interaction. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

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. 2011. Learning Online Discussion Structures by Conditional Random Fields. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, page 435–444, New York, NY, USA. Association for Computing Machinery.

Yi-Chia Wang, Mahesh Joshi, William Cohen, and Carolyn Rosé. 2008. Recovering Implicit Thread Structure in Newsgroup Style Conversations. In *AAAI*.

Tao Yu and Shafiq Joty. 2020a. Online Conversation Disentanglement with Pointer Networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6321–6330.

Tao Yu and Shafiq Joty. 2020b. Online conversation disentanglement with pointer networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6321–6330, Online. Association for Computational Linguistics.

Henghui Zhu, Feng Nan, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2020. Who Did They Respond To? Conversation Structure Modeling Using Masked Hierarchical Transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Ozan İrsoy, Rakesh Gosangi, Haimin Zhang, Mu-Hsin Wei, Peter Lund, Duccio Pappadopulo, Brendan Fahy, Neophytos Neophytou, and Camilo Ortiz. 2019. Dialogue Act Classification in Group Chats with DAG-LSTMs. In *SIGIR 2019 Workshop on Conversational Interaction Systems*.

A Appendix

A.1 DAG-LSTM Equations

A DAG-LSTM is a variation on the Tree-LSTM (Tai et al., 2015b) architecture, that is defined over DAGs. Given a DAG, G, we assume that for every vertex v of G, the edges $e(v, v')$ connecting the children $v' \in \mathcal{C}(v)$ to v can be assigned a unique label $\ell_{v,v'}$ from a fixed set of labels.

A pair of states vectors (h_v, c_v) and an input x_v are associated to every vertex v. The DAG-LSTM equations define the states (h_v, c_v), as a function of the input x_v and the states of its children:

$$(h_v, c_v) = \text{DAG-LSTM}(x_v; \{(h_w, c_w) | w \in \mathcal{C}(v)\}). \tag{2}$$

The equations defining such functions are the following:

$$i_v = \sigma\Big(W_{ix} x_v + \sum_{v' \in \mathcal{C}(v)} W_{ih}^{\ell_{v,v'}} h_{v'}\Big) \tag{3}$$

$$f_{vv'} = \sigma\Big(W_{fx} x_v + \sum_{v'' \in \mathcal{C}(v)} W_{fh}^{\ell_{v,v'} \ell_{v,v''}} h_{v''}\Big) \tag{4}$$

$$c_v = i_v \odot u_v + \max_{v' \in \mathcal{C}(v)} f_{vv'} \odot c_{v'} \tag{5}$$

$$h_v = o_v \odot \tanh(c_v) \tag{6}$$

The equations for the o and u gates are the same as those for the i gate by replacing everywhere $i \to o, u$. Bias vectors are left implicit in the definition of i, f, o, and u. $\odot$ represents Hadamard product and max in Eq. 5 represent elementwise max operation.

A bidirectional DAG-LSTM, is just a pair of independent DAG-LSTM, one of which is unrolled over the time reversed sequence of utterances. The output of a bidirectional DAG-LSTM is the concatenation of the h states of the forward and backward unit for a given utterance.

A.2 Training and Hyperparameter Tuning

We use adjudicated training, development, and test sets from (Kummerfeld et al., 2019b). Each of these dataset is composed a set of conversation (153 in the training set and 10 in both development and test set) each representing a chunk of contiguous posts from the IRC `#Ubuntu` channel. Each of these conversation contains strictly more than 1000 posts (exactly 1250 and 1500 for dev and test set respectively). Annotations are available for all but the first 1000 posts in every conversation. We apply some preprocessing to these conversations. We chunk the annotated section of every training conversation in contiguous chunks of 50 posts each, starting from the first annotated post. [2] To each of these chunks we attach a past context of 100 posts and a future context of 50, resulting in 200 utterances long chunks. For each of these chunks we keep only those annotated links for which the response utterance lies in the central 50 posts. We do not chunk development and test set, but drop the first 900 post in every conversation.

The various architectures we consider share the same set of parameters to fine-tune. One parameter d_h controls the dimension of the hidden state of the LSTMs and one parameter d_{FF} controls the dimension of the hidden layers of the feed-forward scorer. We use word dropout, apply dropout after the max-affine layer, and apply dropout after activation at every layer of the feed-forward scorer. We clip all gradient entries at 5. We use a single layer of LSTMs and DAG-LSTMs to build the χ and ϕ, ψ representations and we do not dropout any of their units. Similarly we use a single layer GRU for the thread encoder. We list all the hyperparameters in Table 3 together with their range and distribution used for the random search.

Hyperparameter optimization is performed by running 100 training jobs for the base BiLSTM architecture, DAG-LSTM, and DAG-LSTM with thread encoding. Our published results are from the best among these runs. The best sets of parameters we find for each of these architectures are:

- BiLSTM: $d_h = 256$, $d_{FF} = 128$, no word and max-affine dropout, a feed forward-dropout equal to 0.3, and a learning rate of 2.4×10^{-4}.
- DAG-LSTM: $d_h = 64$, $d_{FF} = 256$, no word and max-affine dropout, a feed forward-dropout equal to 0.3, and a learning rate of 7.3×10^{-4}.
- DAG-LSTM with thread encoding: $d_h = d_{FF} = 256$, word and max-affine dropout equal to 0.3, a feed forward-dropout equal to 0.5, and a learning rate of 7.9×10^{-4}.

User feature and mention link ablations are obtained by fixing all parameters of the best DAG-LSTM run (removing the feature we are experimenting with) and running 10 jobs by only changing the random seed.

[2]This may result in the last chunk to have less than 50 posts. This happens for 45 conversations.

Parameter	Domain	Distribution
d_h	$\{64, 128, 256\}$	categorical
d_{FF}	$\{64, 128, 256\}$	categorical
word dropout	$\{0, 0.3, 0.5\}$	categorical
max-affine dropout	$\{0, 0.3, 0.5\}$	categorical
feed-forward dropout	$\{0, 0.3, 0.5\}$	categorical
learning rate	$[10^{-5}, 10^{-3}]$	log-uniform
BiDAG-LSTM	{true, false}	categorical

Table 3: Hyperparameters of the model architectures. During hyperparameter optimization, we perform a random search according to the distributions described above. Categorical distributions have uniform probability mass function.

Each training job is performed on a single GPU and, depending on the architectures, takes from 6 to 12 hours.

A.3 Significance Estimates

We use McNemar test (McNemar, 1947) to evaluate the significance of performance differences between model. Given two models M_A and M_B, we define $n_{A\overline{B}}$ as the number of links correctly predicted by A but not by B. Under the null hypothesis both $n_{A\overline{B}} \sim \text{Bin}(n_{A\overline{B}}, n, 1/2)$, where $n \equiv n_{A\overline{B}} + n_{B\overline{A}}$. We define a model A to be *significantly* better than a model B if the null hypothesis is excluded at 95% confidence level.