Appendices: A Large-Scale Corpus for Conversational Disentanglement

Jonathan K. Kummerfeld[1*] **Sai R. Gouravajhala**[1] **Joseph J. Peper**[1]
Vignesh Athreya[1] **Chulaka Gunasekara**[2] **Jatin Ganhotra**[2]
Siva Sankalp Patel[2] **Lazaros Polymenakos**[2] **Walter S. Lasecki**[1]

Computer Science & Engineering[1] T.J. Watson Research Center[2]
University of Michigan IBM Research AI

A Data and Code

As well as this document, we also include the key data and code used in this work. Due to file size limitations we do not include all of the resources we have constructed. For additional content, see `https://jkk.name/irc-disentanglement`. Of the data, we provide:

- ASCII versions of the messages.
- Graph annotations.
- Annotation guidelines.

We do not include other forms of the messages (raw text or tokenised text) and other forms of the annotations (conversation clusters).

Of the source code, we provide:

- System code for our neural model.
- Tools for useful conversions (e.g., graphs to clusters).
- Evaluation metrics

We do not include a range of other tools for pre- and post-processing of the data, collection of statistics, and other forms of format conversions (e.g. Elsner's format to/from ours).

Format: The data is stored with stand-off annotations. There are a set of ascii text files containing the original logs. For each file there is a matching annotation file with lines of the format "number number -" where the two numbers refer to lines in the text file, indicating that the two lines are linked. If the two numbers are equal, the link is a self-link, indicating the start of a conversation.

* jkummerf@umich.edu

B Data Selection

The paper provides the most important aspects of the data selection process. For completeness, we provide additional details here.

Pilot: Partially chosen randomly, and partially at times of light and heavy use of the channel.

Dev & Test: Ten random points in the logs each, with 250 and 500 messages at each point respectively. Samples were chosen at random, but we excluded cases with a large number of system messages (a third or more of the messages) or many in a row (thirty or more). These came up particularly in the early days of the channel, when a lot of additional system logging occurred and not many users were active.

Train: Data was sampled in three ways: (1) the same way as Dev and Test, (2) the same way again, but with only 100 messages annotated (used to check annotator agreement), and (3) time spans of one hour, chosen to sample a diverse range of conditions in terms of the number of messages, the number of participants, and what percentage of messages are directed. When counting users we take into consideration system messages that announce changes to names, so that each user is only counted once. To count directed messages we considered the first word in the message, removing a trailing colon or comma if present. This approach is approximate, but covers the vast majority of uses we observed and so was deemed sufficient for data selection. We sorted the data independently for each factor and divided it into four equally sized buckets, then selected four random samples from each bucket, giving 48 samples (4 samples from 4 buckets for 3 factors).

Channel Two: Drawn from another channel, this is the message log annotated by Elsner and Charniak (2008), plus the 100 message gap they did not annotate between their dev and test sets.

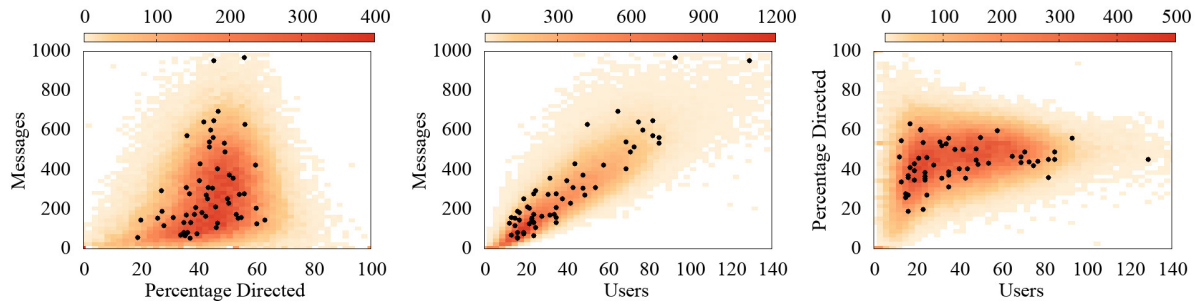

Figure 1: Data statistics (heatmap) and the samples for the third type of training annotations (circles). Values are calculated for each hour in the data, then hours are bucketed and counted for the purposes of visualisation (buckets are width 3, 20, and 2, for users, messages, and percentage directed respectively). Note, the colour scale varies between plots, as the highest observed count depends on the factors defining the buckets.

C Models

C.1 Baselines

Link to Previous: This method does not separate conversations at all, but provides a reference point for metrics. Every message is linked to the previous non-system message and system messages are left unlinked. For graphs, this is the majority class baseline. For conversations, this corresponds to putting all messages in a single conversation, a common baseline in clustering.

Elsner and Charniak (2008): This system has two phases, one to assign a score to a pair of messages indicating if they are part of the same conversation, and one to extract a set of conversations. The scoring phase uses a linear maximum-entropy model that predicts if two messages are from the same conversation using a range of features based on metadata (e.g. time of message), text content (e.g. use of words from a technical jargon list), and a model of unigram probabilities trained on additional unlabeled data. To avoid label bias and reduce computational cost, only messages within 129 seconds of each other are scored. The extraction phase uses a greedy pass in which each message is linked to the earlier message with the highest score from phase one, or with no message if all scores are less than zero.

We retrained the unigram probability model and the max-ent model using our training set. Since the system can only work with one file at a time, we merged our annotations into a single file, with a two hour gap between the last message from one file and the first from the next.

Lowe et al. (2015, 2017):[1] This work focused on dialogue modeling, but used a rule-based approach to extract conversations from the same Ubuntu channel we consider. Their heuristic starts with a directed message and looks back over the last three minutes for a message from the target recipient. If found, these messages are linked to start a conversation, and then follow up directed messages from one user directed at the other are added. Undirected messages from the two participants are added if the speaker is not part of any other conversation at the same time. Conversations are filtered out if they do not have two participants and four or more messages.

We tried running their heuristic in two ways. First, we provided all of the data at once and gave them additional context - the entire day on which a sample was from. If the standard context extended into the previous day we provided that entire day as well. The motivation for this was because the heuristic was developed to be run on the entire Ubuntu dataset, not small samples. This produced extremely poor results, with no completely correct conversations (as reported in an earlier version of this paper). Second, we ran the system separately for each sample and provided the same context as other systems received. These are the results shown in the paper.

Wang and Oard (2009): A weighted combination of three estimates of the probability that two messages are in the same conversation. The estimates are based on time differences, and two measures of content similarity. For each message they find the highest scoring previous message and create a link if the score is above a tuned threshold.

Mehri and Carenini (2017): A system combination approach with several models: (1) an RNN that encodes context messages and scores potential next messages, (2) a random forest classifier

[1] No code was provided with the paper, but it was released at https://github.com/npow/ubuntu-corpus.

that uses the RNN's score plus hand-crafted features to predict if one message is a response to another, (3) a variant of the second model that is trained to predict if two messages are in the same conversation, (4) another random forest classifier that takes the scores from the previous three models and additional features to predict if a message belongs in a conversation. They train on the original Elsner data, graphs annotated for a subset of the development data, and the conversations from Lowe et al. (2015).

C.2 New Statistical Models

We explored a range of approaches, combining different statistical models and inference methods. We used the development set to identify the most effective methods and to tune all hyperparameters. Tuning was conducted via random search, sampling 100 configurations, then refining the search and sampling another 100.

C.2.1 Inference

First, we treated the problem as a binary classification task, independently scoring every pair of messages. Second, we treated it as a multi-class task, where a message is being linked to itself or one of 100 preceding messages.[2] Third, we applied greedy search so that previous decisions could inform the current choice. In all three cases, we experimented with cross-entropy and hinge loss functions. On the development set we found the first approach had poor performance, while the third approach did not yield consistent improvements over the second.

By using the second approach we are unable to recover graphs. Experiments with methods to select multiple links did not yield improvements.

C.2.2 Model

We used a feedforward network, implemented with DyNet (Neubig et al., 2017). We varied the number of layers, the non-linearity, hidden dimensions, input features, and sentence representation. We also explored alternative structures including adding features and sentence representations from messages before and after the current message.

As input we use two versions of the messages, one that is just split on whitespace, and one that has (1) tokenization, (2) usernames replaced with a special symbol, and (3) words that occur less than 65 times in the complete logs replaced with a word shape token. Using the messages and associated metadata, the model uses a range of manually defined features as input.

[2] In theory, a message could respond to any previous message. We impose the limit to save computation.

Per Message
Type of message (system or utterance)
How many users it is directed at
How long ago this user last wrote
If the same user wrote right before or after
Are they a bot
If this message is targeted
If their previous message was targeted
Year it was sent
Hour of the day
Messages per minute in this log
Pairwise
Number of words in common
Time difference
Number of intervening messages
Is the user the same
Is each message directed at the other user
In between, are there messages from either user
Previously, did one user address the other
Do they have the same target

Table 1: Message features as input to our models. All are represented as values between 0 and 1 (most as a binary choice).

For training we varied the gradient update method, learning rate, learning rate decay, loss type, batch size, weight decay, dropout (both at input and the hidden layers), gradient clipping, and weights for the loss based on error types.

The best model was relatively simple, with two fully-connected hidden layers, 512 dimensional hidden vectors, softsign non-linearities, and sentence representations that used the average and max of 100-dimensional word embeddings trained using GloVe (Pennington et al., 2014) on messages from the entire Ubuntu IRC channel. We trained the model with a cross-entropy loss, no dropout, weight decay of $1e^{-7}$, and stochastic gradient descent using a learning rate of 0.018804, learning rate decay of 0.103, gradient clipping of 3.74. We used early stopping, running for up to 20 iterations over the training data, stopping after no improvement in 5 iterations and saving the model that scored highest on the development set.

We also considered a linear version of the model using the same input and output, just no hidden layers or nonlinearity. We tuned hyperparameters separately for the linear model, also using random search. The final values were a learning rate of 0.0465, learning rate decay of 0.086, gradient clipping of 2.165 and weight decay of $1e^{-6}$.

For evaluation, we averaged results from ten

models with different random seeds. We also use the ten models to form ensembles in three ways:
Union: Any edge predicted by one of the models is included. Self-links are only returned if all models predict that the message should link to itself.
Vote: Edges that every model predicted are kept, all others are discarded. Any message that does not have an antecedent after that is linked to itself.
Intersect: First, we convert the graph output to conversations. Then, every conversation that all ten models agree on is kept. Other conversations are broken up to have every message as its own conversation.

D Dialogue Modeling

For our experiment on training next utterance selection models for dialogue, we did the following:

- Trained 10 FF models.[3]
- Ran all 10 models on the entire Ubuntu chat logs. The dataset was sliced up for parallel processing. To avoid odd edge affects, the 10 models saw versions sliced 10 different ways.
- Applied the Intersect approach, but requiring only 7 or more models to agree.

We then filtered the conversations so that (1) the first message is not directed, (2) there are exactly two participants not counting the channel bot (a helper and the person seeking help), (3) no more than 80% of the messages are by a single participant, and (4) there are at least three messages. This gave 114,201 conversations.[4]

Using the new conversations, we performed a next utterance selection task. Systems were given partial conversation cut off before a message from a helper and needed to predict the next utterance from a set of 10 options, where nine are randomly chosen messages from helpers in the full dataset. We measure performance with Mean Reciprocal Rank, and by counting the percentage of cases when the system places the correct next utterance among the first k options (Recall@k).

We applied two standard models to this task: a dual-encoder model (DE, Lowe et al., 2017), and the Enhanced Long Short-Term Memory model (ESIM, Chen et al., 2017). We implemented both models in Tensorflow (Abadi et al., 2015). Words were represented as the concatenation of (1) word embeddings initialized with 300-dimensional GloVe vectors, and (2) the output of a bidirectional LSTM over characters with 40 dimensional hidden vectors. All hidden layers were 200 dimensional, except the ESIM selection layer, which had 256 dimensions. Finally, we limited the input sequence length to 180 tokens. We trained with batches of size 128, and the Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of 0.001 and an exponential decay of 0.95 every 5000 steps.

[3] These models have a slightly different configuration than the models used in the main experiments (they are the best configuration before conducting the random hyperparameter search).

[4] This filtering process is slightly different from the one used in DSTC 7 track 1 (Gunasekara et al., 2019).

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1.

Micha Elsner and Eugene Charniak. 2008. You Talking to Me? A Corpus and Algorithm for Conversation Disentanglement. In *Proceedings of ACL-08: HLT*.

Chulaka Gunasekara, Jonathan K. Kummerfeld, Lazaros Polymenakos, , and Walter S. Lasecki. 2019. Dstc7 task 1: Noetic end-to-end response selection. In *7th Edition of the Dialog System Technology Challenges at AAAI 2019*.

D. P. Kingma and J. Ba. 2014. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*.

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. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

Ryan 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).

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)*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The Dynamic Neural Network Toolkit. *arXiv preprint arXiv:1701.03980*.

Jeffrey Pennington, Richard Socher, and Christopher 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, Doha, Qatar. 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*.