Vertext: An End-to-end AI Powered Conversation Management System for Multi-party Chat Platforms

Omer Anjum*
University of Illinois at Urbana-Champaign
Champaign, Illinois
oanjum@illinois.edu

Tanitpong Lawphongpanich*
University of Illinois at Urbana-Champaign
Champaign, Illinois
tl6@illinois.edu

Tianyi Tang*
University of Illinois at Urbana-Champaign
Champaign, Illinois
ttang13@illinois.edu

Wen-mei Hwu
University of Illinois at Urbana-Champaign
Champaign, Illinois
w-hwu@illinois.edu

Sanjay Patel
University of Illinois at Urbana-Champaign
Champaign, Illinois
sjp@illinois.edu

Chak Ho Chan*
University of Illinois at Urbana-Champaign
Champaign, Illinois
chchan2@illinois.edu

Yucheng Liang*
University of Illinois at Urbana-Champaign
Champaign, Illinois
yliang36@illinois.edu

Shuchen Zhang*
University of Illinois at Urbana-Champaign
Champaign, Illinois
szhan114@illinois.edu

Jinjun Xiong
IBM Thomas J. Watson Research Center
Yorktown Heights, New York
jinjun@us.ibm.com

*These authors contributed equally to this research.

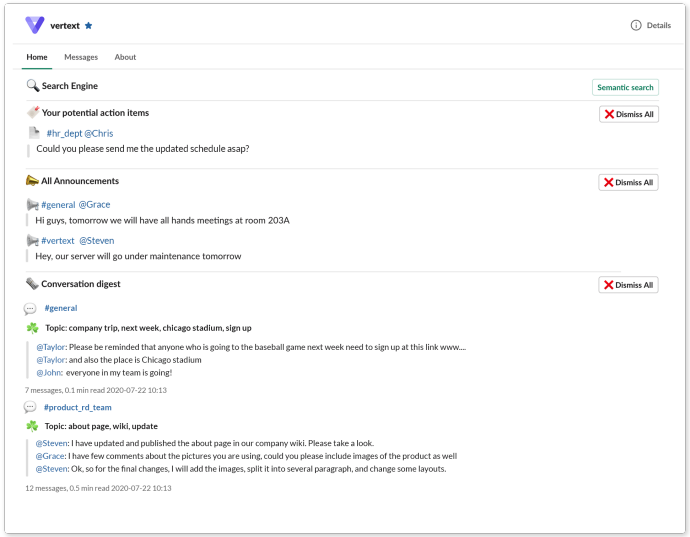

Figure 1: Vertext home tab

ABSTRACT

Online communication platforms like Slack and Microsoft teams have become increasingly crucial for a digitized workplace to improve business efficiency and growth. However, these chat platforms can overwhelm the users with unstructured long streams of back and forth discussions scattered in various places. Thus, discussions become challenging to follow, leading to an increased likelihood of missing valuable information. Moreover, with the unsatisfying keyword-based chat search, users spend a significant amount of time to read, digest, and recall information from the conversations at the cost of productivity. In this paper, we present Vertext, an end-to-end AI system that ingests user conversations and automatically extracts information such as announcements, task assignments, and conversation summary. Moreover, Vertext gives a unique search experience to the users by providing search results along with their context, with an improved performance enabled by semantic search. For the ease of user interaction, all the information is consolidated on a single dashboard provided by Vertext.

CCS CONCEPTS

• **Information systems** → **Internet communications tools**; **Chat**; *Texting*; **Search interfaces**; • **Human-centered computing** → **Human computer interaction (HCI)**; • **Computing methodologies** → **Natural language processing**; **Discourse, dialogue and pragmatics**.

KEYWORDS

artificial intelligence; collaborative chat platforms for business; transformer; deep average network; semantic search; Slack; Microsoft Teams; conversation disentanglement; search engine for chat; dialog act classification; natural language processing

ACM Reference Format:
Omer Anjum, Chak Ho Chan, Tanitpong Lawphongpanich, Yucheng Liang, Tianyi Tang, Shuchen Zhang, Wenmei Hwu, Jinjun Xiong, and Sanjay Patel. 2020. Vertext: An End-to-end AI Powered Conversation Management System for Multi-party Chat Platforms. In *Companion Publication of the 2020 Conference on Computer Supported Cooperative Work and Social Computing (CSCW '20 Companion), October 17–21, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3406865.3418570

CSCW '20 Companion, October 17–21, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8059-1/20/10..
https://doi.org/10.1145/3406865.3418570

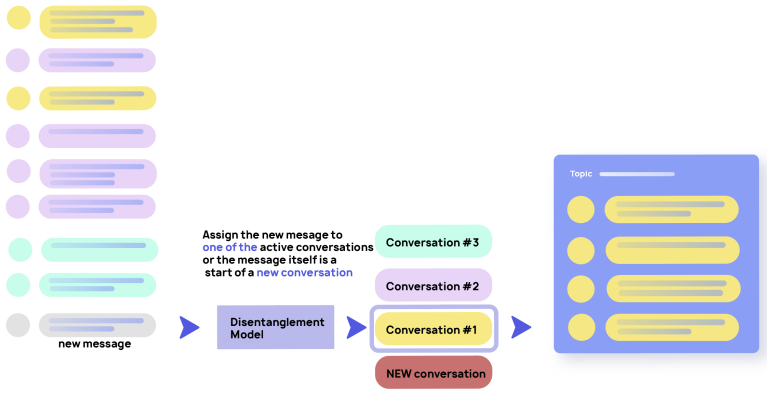

Figure 2: Conversation disentanglement

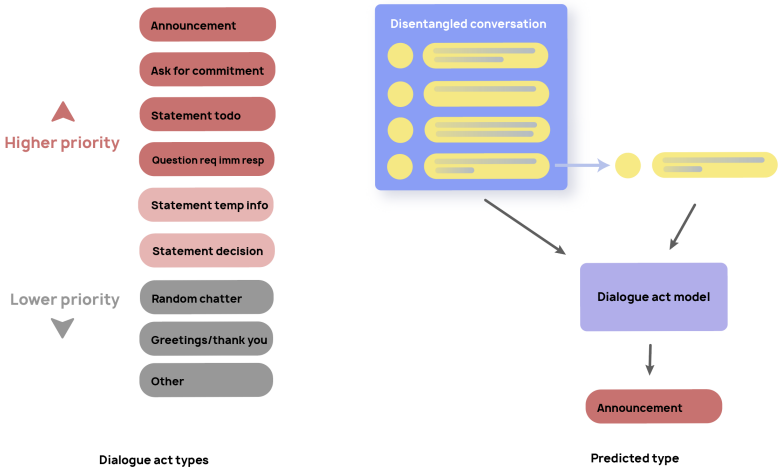

Figure 3: Dialog act classification

INTRODUCTION

Messaging applications like Slack, Microsoft Teams, Hive, and Google Hangouts provide a quick and easy way of exchanging information. It has led to the wide adoption of messaging apps by businesses worldwide to improve productivity. We see it as a significant transformation where messaging apps are replacing emails as a primary channel for communication.

While chat platforms offer potential benefits, they come with their challenges. Unlike emails where message is sent as a complete blob of text with all its context, conversations in chat platforms are scattered over with multiple short messages. And in the case of concurrent ongoing conversations, messages belonging to different conversations are also interleaved, making discussions challenging to follow[5, 10, 14]. Moreover, as the size of a team or the collaboration grows, the frequency of the incoming messages increases, reaching a point where it becomes difficult to keep track of all the conversations[7]. It leads to a potential risk of losing valuable information. Another challenge is searching through conversations, which is different compared to searching through documents. Conversations use an informal language structure, making it difficult to extract information using existing information retrieval methods. Moreover, due to conversations scattered over multiple messages, keywords from the search query may also scatter over multiple messages. Consider two messages as an example: *"message 1: We are having a meeting tomorrow, message 2: Here's the zoom link"*. For an example query "meeting zoom link", the keywords "meeting" and "zoom link" are scattered across two different messages. Furthermore, there may be vocabulary mismatch between query and the target message. Thus, finding all the coherent set of messages in the unstructured chat history, along with their context to match the query, becomes a daunting challenge.

Much work has been published in the natural language processing (NLP) domain to address some of the individual challenges. [4, 8, 10, 11] propose deep learning models to disentangle conversations as the fundamental step of organizing conversation information.

[13, 14] addresses the importance of structured representations such as discourse acts which are the signals most favored by the users, to help prioritize their attention.

Another important NLP component is a search engine. Current chat platform search relies heavily on the traditional keyword matching approach, which is not satisfactory for a good user experience [6]. While lots of research is ongoing to improve different NLP components, we think there is a dire need to leverage the existing NLP components and build an NLP pipeline and ecosystem to solve these challenges. We build Vertext, an end-to-end NLP system assembling different NLP components such as conversation disentanglement, dialogue act, and search engine together (Figure 1). It is a unique combination where different components leverage each other's output and enable four downstream tasks that Vertext is currently supporting: 1) search, 2) action items detection, 3) announcements detection, and 4) conversation summary.

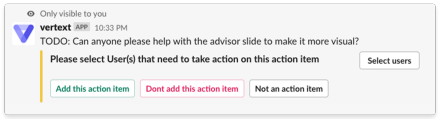

Figure 4: Confirmation message for an action item

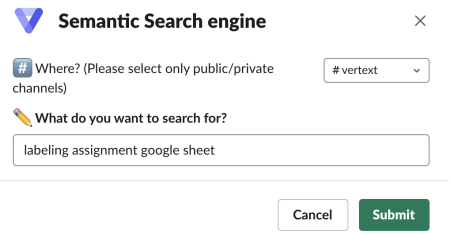

Figure 5: Search engine interface

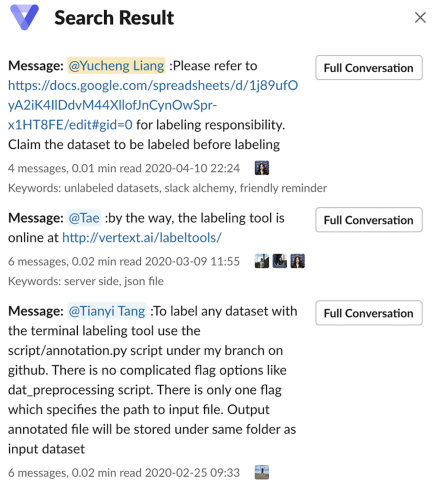

Figure 6: Search engine result

PIPELINE

Conversation disentanglement

Conversation disentanglement refers to the task of assigning interleaved messages in a multi-party chat to the correct conversation thread (Figure 2). It serves as the base of all downstream tasks in the Vertext ecosystem, and thus requires a significant amount of efforts to guarantee the reliability of the model. The most challenging part, just like many AI applications, is the collection of annotated data from the domain of interest such as Slack. To overcome the scarcity of data, we manually labeled messages from project-oriented channels with less than 25 users to train our disentanglement model, using our own in-house Vertext labeling web tools.

The disentanglement model is based on [11], an LSTM-based neural network that aims to assign a message to its corresponding thread by matching the conversation flow and semantic similarity. Inspired by [10], we handcraft 19 context features as additional inputs to the model to further improve its performance in our application. In [11], the authors use the Universal Sentence Encoder (USE) [4] to generate sentence embeddings, while we use Fasttext [8] as it exhibits the best tradeoff between model performance and embedding generation runtime. We apply a k-fold cross validation to evaluate the model. The average testing accuracy is 72% for k=10.

Dialog act classification

Dialog act classification is the task of classifying an utterance by its communicative function in a conversation. In our work, an utterance can be obtained by sentence tokenization, and each turn can contain one or several utterances. The labels are tailored for the downstream tasks including search, action items detection and announcement detection. There are sixteen labels in total, including announcement, ask for commitment, statement todo, statement others, etc (Figure 3, 4). Training data is obtained by manual labeling on the same set of conversation data as the disentanglement model.

In this version, we simplify the problem of auto-detection of announcement, action item, or question that requires an immediate response in each message. Our proposed dialogue act model consists of two stages. Each stage has a model we call 'RDAN', which is a Recurrent Neural Network (RNN) concatenated to a Deep Average Network (DAN) model[9]. Input to the RDAN at the first stage is the word embeddings, POS tags and the same message features used in the disentanglement model. At the first stage, RDAN predicts labels, including announcements, questions, and statements. First stage prediction is an additional input feature for the second stage, with other input features the same as those given at the first stage. The output of the second stage is the final classification, which further determines the subtype of each category of dialogue act. The model is able to correctly classify these dialog acts with 70% accuracy, 50% average precision and 50% average recall.

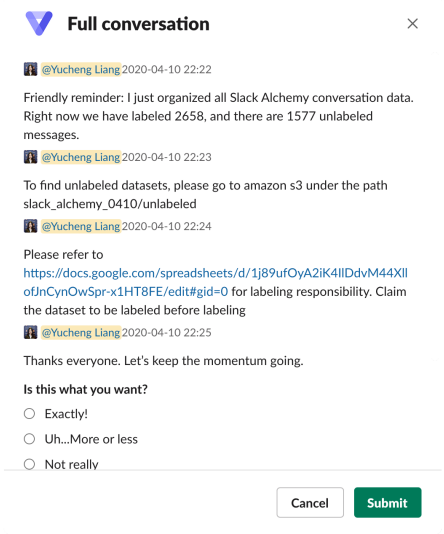

Figure 7: Search engine result with expanded conversation

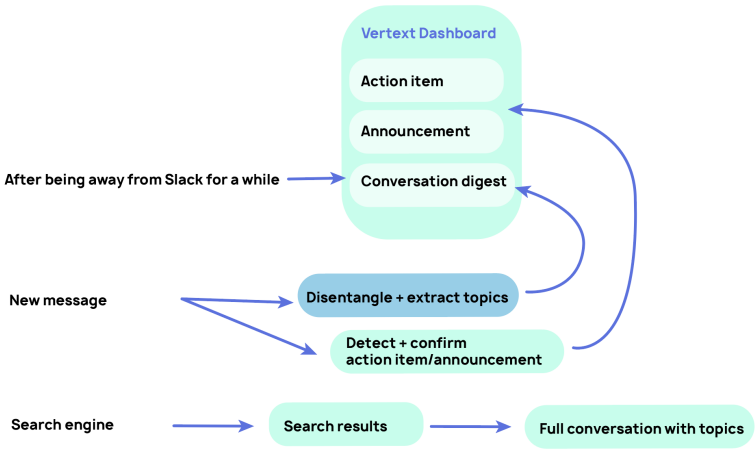

Figure 8: Vertext UI flow

Search engine

A good search engine allows users to hold less information in minds and helps them recall information faster. The current search engine on chat history as provided by Slack takes the traditional keyword matching approach [2] based on Lucene [1] ranking messages based on the occurrence and frequency of query terms appeared in the messages. Such an approach can fail when users do not remember the exact phrase but search for synonyms instead. Moreover, this approach does not always capture the intent of the user behind the query. Vertext search engine addresses this problem by adding a semantic understanding of the user query and the chat history (Figure 5,6). We use fast sentence embedding [3] to represent the semantic meaning of the query and the messages in the chat history. Each message will receive a score, which is the cosine similarity between the embedding vector of the message and that of the query. The weighted sum of the embedding score and BM25 score [12] becomes the new score for ranking the messages. To measure the model performance, we hand-labeled 348 queries. Our proposed search model outperforms slack's search by 5% in precision and 14% in the recall. In addition to providing more accurate search result, we enrich the search experience by allowing users to view the matched message in its corresponding conversation, as detected by the conversation disentanglement model (Figure 7). This additional feature can remind the users of the context of the matched message. In particular, the context is essential when a user is looking for a file or link or the answer to a question, which is in general a part of the context but not the query itself.

UI flow

The Vertext App lives in Slack platform designed both for mobile and desktop. The user flow is customized to meet the Slack API limitations. We designed the UX which provides users an overview of their unread notifications and messages in one place. Users can visit the Vertext dashboard after being inactive on Slack for a while (Figure 8). Besides, we also design the automated workflow such that some labels can be gathered naturally from user interactions with our system to reinforce our machine learning models.

The home tab serves as the entry point for the users to view all the information presented by Vertext. The dashboard consists of lists of action items, announcements, and conversation summaries. While users are conversing in their channels, the Vertext bot will detect incoming messages and classify whether a particular message contains action items, announcements, or neither. Whenever there is a match, the bot will send an ephemeral prompt to the sender to ask for confirmation. For example, if the user determines it as an action item, the prompt will further ask for the potential target(s), i.e. members in the workspace who need to take actions from this message.

REFERENCES

[1] 2020. https://lucene.apache.org/core/3_0_3/api/core/org/apache/lucene/search/Similarity.html

[2] 2020. Search at Slack. https://slack.engineering/search-at-slack/
[3] Oliver Borchers. 2019. Fast sentence embeddings. https://github.com/oborchers/Fast_Sentence_Embeddings.
[4] 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. *CoRR* abs/1803.11175 (2018). arXiv:1803.11175 http://arxiv.org/abs/1803.11175
[5] Micha Elsner and Eugene Charniak. 2010. Disentangling Chat. *Computational Linguistics* 36, 3 (2010), 389–409. https://doi.org/10.1162/coli_a_00003
[6] Eli Finkelshteyn. 2019. I love you, Slack. But your worst flaw will make me quit you. https://qz.com/1703430/slack-needs-to-fix-its-poor-search-functionality/
[7] 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. Association for Computational Linguistics, 1812–1822. https://doi.org/10.18653/v1/N18-1164
[8] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *CoRR* abs/1607.01759 (2016). arXiv:1607.01759 http://arxiv.org/abs/1607.01759
[9] Chandra Khatri, Rahul Goel, Behnam Hedayatnia, Angeliki Metanillou, Anushree Venkatesh, Raefer Gabriel, and Arindam Mandal. 2018. Contextual topic modeling for dialog systems. In *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 892–899.
[10] Jonathan K. Kummerfeld, Sai R. Gouravajhala, Joseph J. Peper, Vignesh Athreya, Chulaka Gunasekara, Jatin Ganhotra, Siva Sankalp Patel, Lazaros C Polymenakos, and Walter Lasecki. 2019. A Large-Scale Corpus for Conversation Disentanglement. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 3846–3856. https://doi.org/10.18653/v1/P19-1374
[11] 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. Association for Computational Linguistics, 6456–6461. https://doi.org/10.18653/v1/D19-1682
[12] Andrew Trotman, Antti Puurula, and Blake Burgess. 2014. Improvements to BM25 and Language Models Examined. In *Proceedings of the 2014 Australasian Document Computing Symposium (ADCS '14)*. Association for Computing Machinery, New York, NY, USA, 58–65. https://doi.org/10.1145/2682862.2682863
[13] Amy Zhang, Bryan Culbertson, and Praveen Paritosh. 2017. Characterizing online discussion using coarse discourse sequences. (2017).
[14] Amy X. Zhang and Justin Cranshaw. 2018. Making Sense of Group Chat through Collaborative Tagging and Summarization. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW, Article 196 (Nov. 2018), 27 pages. https://doi.org/10.1145/3274465