Contextual Typeahead Sticker Suggestions on Hike Messenger

Mohamed Hanoosh , **Abhishek Laddha** and **Debdoot Mukherjee**
Hike Messenger, New Delhi, India
{moh.hanoosh, abhishekl, debdoot}@hike.in

Abstract

In this demonstration, we present Hike's sticker recommendation system, which helps users choose the right sticker to substitute the next message that they intend to send in a chat. We describe how the system addresses the issue of numerous orthographic variations for chat messages and operates under 20 milliseconds with low CPU and memory footprint on device.

1 Introduction

In messaging apps such as Facebook Messenger, WhatsApp, Line and Hike, new modalities such as emojis, gifs and stickers are extensively used to visually express thoughts and emotions [Lim, 2015; Donato and Paggio, 2017; Barbieri *et al.*, 2017]. While emojis are mostly used in conjunction with text, stickers can provide a graphic alternative for text messages altogether. Hike[1] stickers are composed of an artwork (e.g., cartoonized characters and objects) and a stylized text for a commonly used chat phrase (See stickers in Fig. 1). Hike has tens of thousands of such stickers in different languages so that its users can engage in rich conversations by chatting with stickers back to back. However, discovering the right sticker when you need it in a chat can be cumbersome because it's not too straightforward to find a sticker that can best substitute your next utterance. To alleviate this problem, we have developed a sticker recommendation system that suggests stickers based on the context of the conversation. Upon receiving a message, the user sees suggestions on stickers that can be used to reply to the message. In case the user starts typing the reply, the suggestions are revised based on the text input; being refreshed on every key press.

The goal of the type-ahead sticker recommendation system is to help users discover the perfect sticker which can be shared in lieu of the text message that they want to send in the context of a chat. The latency of generating such sticker recommendation should be in tens of milliseconds in order to avoid any perceivable delay during typing. This is possible only if the system runs end-to-end on the mobile device without any network calls.

One can potentially think of setting up such sticker recommendation with the help of a supervised model, which learns

[1]https://hike.in/

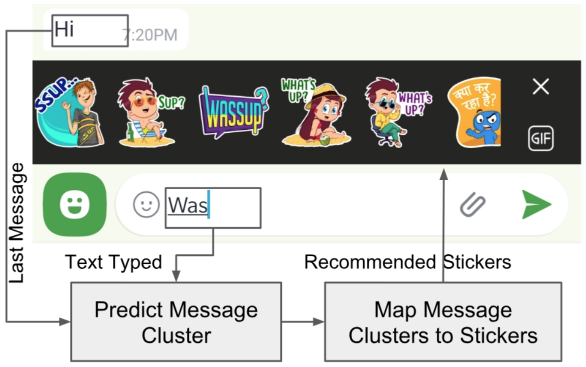

Figure 1: Sticker Recommendation UI and a high level flow diagram

the most relevant stickers for a given chat context defined by the previous message(s) and the text typed in the chat input box. However, due to frequent additions to the sticker inventory and a massive skew in historical usage toward a handful of popular stickers, it becomes difficult to collect reliable, unbiased data to train such an end-to-end model. Moreover, an end-to-end model will need to be retrained frequently to support new stickers and the updated model will have to be synced to all devices. Such regular updates of the model, possibly $\geq$ 10 MB in size, will be prohibitively expensive. Thus, instead of creating an end-to-end model, we decompose the sticker recommendation task into two steps (see Figure 1). First, we predict the message that a user is likely to send based on the last message received and the text typed in the input box. Second, we recommend stickers by mapping the predicted message to stickers that can substitute it. Message prediction can be performed with the help of a language model trained on historical chat corpus. Since the distribution of chat messages does not change as much with time, we do not need to frequently update this model. However, the message-to-sticker mapping needs to be regularly refreshed so that we can account for the relevance feedback observed on stickers as well as add support for new stickers.

Now, messages in our chat corpus have a large number of orthographic variants. Most of our users chat by transliterating from their native languages using an English keyboard. Since transliteration has no definite rules, the same word can be spelt in different ways. e.g., "acchha" (Hindi for "good") has many variants in the English script - "accha", "acha", "achha" etc., all of which are frequent. This problem further compounds for phrases. For instance, we observe 343 orthographic variants of “kya kar raha hai” (Hindi for “What are you doing”) in our dataset. Even in English, people often skip vowels in words as they type in a chat. (*e.g.,* “where are you” → “whr r u”, “ok” → “k”). Further, use of acronyms (*e.g.,* “i don’t know“ → “idk”) and repetition of characters to exaggerate certain words (*e.g.,* “gooooood morning“) is also widespread. As a result, the number of unique messages in the chat corpus are an order of magnitude higher than the number of message intents. Thus, we work with message intents instead of messages to reduce redundancy. We cluster messages in our corpus to identify unique message intents. For assessing message similarity in clustering, we leverage an effective message embedding [Laddha *et al.*, 2019], which creates similar dense representations for semantically similar messages. Once we have a set of high quality message clusters that cover all the frequent message intents observed in our corpus, we predict the message cluster instead of the message in the first step and employ a message cluster to sticker mapping to obtain sticker recommendations in the second step. We observe that 7500 message clusters can cover all intents expressed in the most frequent 34k messages in our corpus. Mapping stickers to message clusters instead of messages helps greatly reduce the size of the map.

A large fraction of Hike users use low end mobile devices with severe limitations on memory and compute power. To address this, we have a hybrid system for message cluster prediction, which is a combination of a neural network for response prediction that processes chat context on the server, and a trie [Morrison, 1968] based typeahead model that processes the text input on the client. Scores from these two components are combined to produce final scores for message cluster prediction.

Demonstration. The demonstration[2] showcases sticker suggestions as one chats on the Hike app. Further, we walk-through the message prediction flow with a web GUI that allows us to inspect the message clusters produced by the response model and the typeahead model. We can also see the stickers that are mapped to each output cluster. Figure 2 shows a sample message prediction flow. If a user types “go”, the top typeahead prediction is the message cluster for *good*; it’s scored significantly higher than those for *good night* and *good morning*. However, if the last message is “bye tc”, *good night* is predicted as the top message cluster for “go”, since it is a likely response for ‘bye tc”. The combined score for *good night* exceeds that of the other message clusters.

2 System Description

In this section, we briefly describe the key components that constitute our system. In our deployment, the models are trained separately for each geographical region and language.

Message Clustering. We cluster messages in the chat corpus such that each cluster has messages with the same intent. We assess message similarity using embedding trained with the help of the *Input-Response Chat (IRC)* architecture [Laddha *et al.*, 2019]. Our message encoder that uses a combination of Char-CNN [Kim *et al.*, 2016] and GRU is highly

[2]Demo Video: http://hike.in/sr_demo.mp4

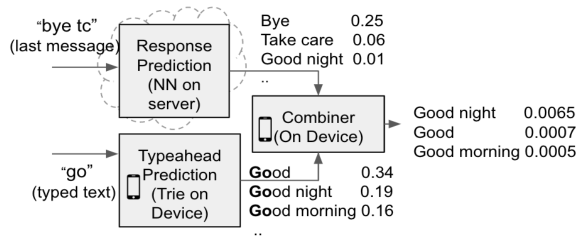

Figure 2: Message Cluster Prediction Flow

effective in deriving similar representations for orthographic variations of a message. Then, we run HDBSCAN [McInnes *et al.*, 2017] to create fine grained message clusters. Top k message clusters by frequency are considered as classes for message cluster prediction.

Response Prediction. Similar to SmartReply [Kannan *et al.*, 2016], which generates short email replies, our response prediction model scores message clusters on their likelihood of being a reply to a given message. We train a neural network on a dataset of messages and their corresponding responses mapped to message clusters [Laddha *et al.*, 2019]. When a message is being routed through the server to its recipient, we predict response message clusters using this model and send the output along with message to the recipient’s device.

Typeahead Prediction. As the user types a message, we update our message cluster predictions using a Patricia Trie [Morrison, 1968] based language model [Laddha *et al.*, 2019] on every key press. The trie stores all frequent messages in our chat corpus as keys along with their cluster identifiers and frequencies as values. It helps efficiently retrieve all the messages that have a prefix matching the text typed by the user. Next, we use the message frequencies to compute the likelihood of each message cluster. The final score for a message cluster is obtained by a linear combination of scores coming from the trie and the response model. The weight of the response model is decayed as the typed input grows in length.

Message Cluster To Sticker Map. Stickers are manually tagged with chat phrases at the time of launch which are used to map stickers to related multiple message clusters. Feedback on suggestions (clicked vs viewed) is used to score the relevance of stickers mapped to a message cluster using a Multi Armed Bandit algorithm [Agrawal and Goyal, 2012] Exploration parameter is tuned to expose the newly launched stickers.

3 Conclusions & Future Work

The sticker recommendation system described in this paper is being used by millions of users on the Hike app on a daily basis. A/B tests showed an 8% improvement in the fraction of messaging users who use stickers. We are actively working on improving different aspects of the system. On high end devices we will deploy a quantized [Jacob *et al.*, 2018] neural network model that predicts the message cluster based on both the message received and the typed input. We are working on expanding the conversational context exploited to last x messages, emotion detected in the chat etc. [Serban *et al.*, 2016] Also, we will personalise the suggestions based on the user’s preferences on art style, gender etc.

References

[Agrawal and Goyal, 2012] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory*, pages 39–1, 2012.

[Barbieri *et al.*, 2017] Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. Are emojis predictable? *arXiv preprint arXiv:1702.07285*, 2017.

[Donato and Paggio, 2017] Giulia Donato and Patrizia Paggio. Investigating redundancy in emoji use: Study on a twitter based corpus. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 118–126, 2017.

[Jacob *et al.*, 2018] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[Kannan *et al.*, 2016] Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, et al. Smart reply: Automated response suggestion for email. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 955–964. ACM, 2016.

[Kim *et al.*, 2016] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *AAAI*, pages 2741–2749, 2016.

[Laddha *et al.*, 2019] Abhishek Laddha, Mohamed Hanoosh, and Debdoot Mukherjee. Understanding chat messages for sticker recommendation in hike messenger. *arXiv preprint arXiv:1902.02704*, 2019.

[Lim, 2015] Sun Sun Lim. On stickers and communicative fluidity in social media. *Social Media+ Society*, 1(1):2056305115578137, 2015.

[McInnes *et al.*, 2017] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11):205, 2017.

[Morrison, 1968] Donald R Morrison. Patricia—practical algorithm to retrieve information coded in alphanumeric. *Journal of the ACM (JACM)*, 15(4):514–534, 1968.

[Serban *et al.*, 2016] Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.