HaRT2: User-State Is All You Need?

Abhinav Sharma, Sugam Dipak Devare and **Vivek Raju Golani**
Department of Computer Science, Stony Brook University
{abhinsharma, sdevare, vgolani}@cs.stonybrook.edu

Abstract

Natural language is generated by people, yet traditional language modeling views words or documents as if generated independently. Recently, the Human Aware Recurrent Transformer model, a large-scale transformer model for the human language modeling task was introduced whereby a human level exists to connect sequences of documents (e.g. social media messages) and capture the notion that human language is moderated by changing human states. We evaluate the impact of the user-states in the accuracy of HaRT model on document level tasks. We conduct experiments on 2 downstream tasks - stance prediction of users on the topic of feminism and hate speech detection on social media data of users. Results on stance prediction indicate that predicting stance on the standalone user-state decreases accuracy for document-level tasks. Results on hate-speech detection on the baseline HaRT model meet or surpass the current state-of-the-art.

1 Introduction

Language use, like any human behavior, is moderated by underlying human states of being (Mehl and Pennebaker, 2003 [1]; Fleeson, 2001 [2]). Indeed, different ways of incorporating human information into NLP models have recently been shown to improve accuracy on many NLP tasks (Hovy, 2015 [3]; Lynn et al., 2017 [4]; Huang and Paul, 2019 [5]; Soni et al., 2022 [6]). The main objective of our project is to use the HaRT model (Soni et al., 2022 [6]) to evaluate the impact of adding a user-state based self-attention mechanism to the transformer architecture.

The HaRT model previously used the representation of the last non-pad token of a message as its representation for message-level tasks, and use the average of the user-states from all the blocks of a user as that user's representation for user-level tasks. However this approach doesn't explain what information is contained in the user-state embeddings and why user-state can't be used directly for performing document level tasks.

We conduct experiments on 2 document level tasks. For the stance prediction task, we try concatenating the user-state to last non-padded token of the document and we try using the standalone user-state to understand the importance of added user-state in language models. We also use the baseline HaRT model for the sentimental analysis task of hate speech detection to evaluate the need of a user-state altogether.

We modify the baseline HaRT model to answer the following questions:

1. Can we predict the stance of an user using only its user-state representation?

2. Can we group users based on the trained user state vectors?

3. Can we extend the HaRT model to detect hate speech?

The main outcomes of our analysis are as follows:

1. We directly use the user-state to evaluate performance of HaRT on document level task of stance prediction and see a drop in accuracy in stance prediction.

2. We also do a principal component analysis of 274 user-state representations to find that users states do not form clusters as per their stance on a topic.

3. We create a balanced dataset for tweets on abusive speech and run the baseline HaRT model for the sentimental analysis task of hate speech detection that meets the current state-of-art results.

2 Your Task

The input to the model consists of a time-stamp based ordered sequence of tweets from a user. These tweets are segmented into block size of 512, separating tweets using the ”insep” token. If the number of tweets tokens in a blocks is less than the block size i.e., 512, padding is used to keep the block size constant. For each block, the model outputs,

1. Contextualized representations of the tweet tokens of the block conditioned on the previous user state.

2. An updated representation of the user state, which now includes the information from the current block.

2.1 Baseline Model

The HaRT architecture consists of one modified transformer layer over additional token-level, conventional self-attention based transformer layers from a pre-trained GPT-2, with a user-state based self-attention mechanism. HaRT models the input data (language) in the context of its source (user) along with inter-document context, thus enabling a higher order structure representing human context.

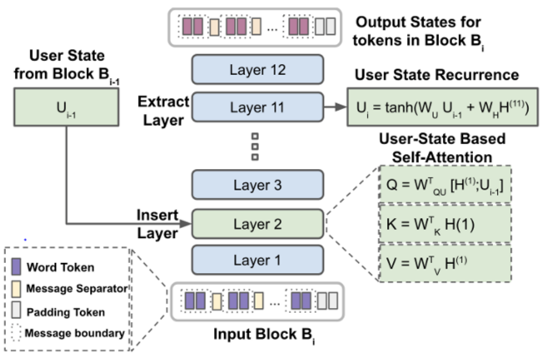


Figure 1: Blocks of messages are processed by HaRT. It generates contextualized representations of the messages in each block based on the user state that is continually computed. Through the use of a modified query transform, the user state is introduced into layer 2, an earlier layer, to inform the self-attention calculation. The output of a subsequent layer is then periodically used to update the previous user state (layer 11).

For each block Bi, HaRT outputs (i) contextualized representations of the tokens within the block conditioned on the previous user state U_{i-1} , and (ii) an updated representation of the user state, U_i, which now also includes the information from the current block B_i. For finetuning, HaRT is first initialized with the pretrained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate finetuned models, even though they are initialized with the same pre-trained parameters. The historical messages (when available) from the respective users is used to replicate the format of pre-training inputs and to benefit from the knowledge of the user.

2.2 Issues

Currently, the HaRT model does not directly take into consideration the user state when predicting the stance or sentiment of the user and is only used to compute the self-attention block in document-level tasks.

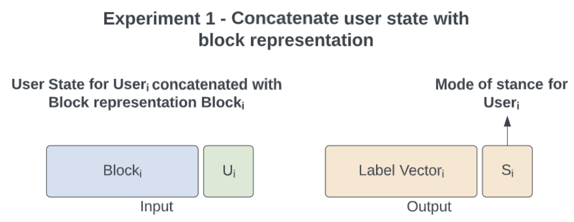
The HaRT model also did not have any analysis of the information contained in the user-state and whether users can be grouped together based on their downstream task labels.

3 Approach

Following are the ideas we have tested as part of this project.

3.1 Using user-state with transformer output:

To understand if user-state vector can be used along with the representation of the last non-padded token of the transformer output to improve performance on document-level tasks, we concatenated the user-state vector with the transformer block output for training and testing. We have assumed the label for this user-state vector to be the ”mode” of the labels ie. the stance the user takes most often.

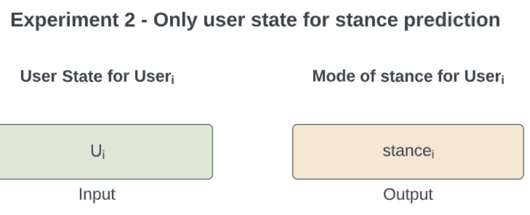


Experiment 1 - Concatenate user state with block representation
User State for User$_i$ concatenated with Block representation Block$_i$
Mode of stance for User$_i$
Block$_i$
U_i
Input
Label Vector$_i$
S_i
Output

Figure 2: Input-output representation for user-state with transformer output for stance prediction.

3.2 Using only the user-state:

The extension to the above idea was to use only the user-state vector for document level tasks. This helped us comprehend the capability of the user-state when it comes to tasks which mainly rely on the input text. The label assumption is same as the above approach.



Experiment 2 - Only user state for stance prediction
User State for User$_i$
Mode of stance for User$_i$
U$_i$
stance$_i$
Input
Output

Figure 3: Input-output representation for using only the user-state for stance prediction.

3.3 Stance information in user-state:

We also wanted to check if the user-state vector contains information about the stance of the user and try to visualize the clusters of users in embedded space based on their stance and compare it with the ground truth.

3.4 Using HaRT for hate speech detection:

Finally, we wanted to see if the fine-tuned HaRT model can outperform the GPT2 model in detecting hatespeech from Twitter data.

3.5 Implementation Details

For the implementation of the project, we used the repository for the HaRT model. Since language models were designed to the used with different downstream tasks by using a pre-trained model we did not make any architectural changes to the HaRT model and only modified and fine-tuned the task-level layers of the model. Before starting with the scenario testing, we did a comprehensive break-down of the HaRT code repository to understand the structure of the code, data-processing steps performed for document and user-level tasks and the architecture of the HaRT model. We used Google Colaboratory to fine-tune the model for different the tasks described in the introduction. The free tier provides 16 GB GPU which is sufficient for training with reduced block size of 512.

4 Evaluation

Our aim is to understand the role of user-state embeddings in assessing document level tasks. One can evaluate the HaRT model for our task by restructuring the output representation for document level tasks using user-state and visualizing the user-state representations against their target labels.

4.1 Dataset Details

Stance Detection. For stance detection we use the SemEval2016 dataset (Mohammad et al., 2016 [7]), which contains tweets annotated as being in favor of, against, or neutral toward one of five targets: atheism, climate change as a real concern, feminism, Hillary Clinton, and legalization of abortion. This data only includes labeled tweets from users and not any history, so we use the extended dataset from Lynn et al. (2019) [8] and preserve the train/dev/test split of the same. To maintain (message created time) temporal accuracy in our autoregressive model, we only used the part of the extended dataset (history) that consists of messages posted earlier than the labeled messages.

Feminism	Train	Test	Dev
Tweets	2341	481	583
Users	274	64	150

Table 1: Train/Test/Dev splits of the Dataset for Feminism task.

Hate Speech Detection. For hate speech detection, we created a dataset with almost equal number of abusive and normal statements. The original dataset, with 100k tweets annotated using the CrowdFlower platform was collected for (Founta et al. 2018). The data contains the time of tweet, user id, tweet id and tweet message along with the label abusive or normal. We arranged tweets of every user in ascending order of time to maintain temporal structure.

Hate Speech	Train	Test	Dev
Tweets	5344	1145	1145
Users	3545	1017	1009

Table 2: Train/Test/Dev splits of the Dataset for Hate Speech Detection task.

4.2 Evaluation Measures

Here, we evaluate the utility of fine-tuning HaRT for document- level tasks. Just as standard transformer language models are fine-tuned for tasks, we take the pre-trained HaRT model and fine-tune it for stance detection on the topic feminism and sentiment analysis - hate speech detection tasks. For hate speech detection task we compare fine-tuning the GPT-2HLC as a nonuser-based LM baseline and report these results in Table 4. All hyperparameter settings and training details for the GPT-2HLC and HaRT models are same as HaRT baseline this task.
We further fine-tune the HaRT model for stance detection on the topic of feminism. We modify the model output representations to include user-state in the output in some form to gauge the direct dependency of output on the user-state and find that including the user-state in the output representation has an adverse effect on model performance. The accuracy of model with no document level representation is the least and the model with last token representation of the document as output has highest accuracy.

4.3 Baselines

HaRT shares the same architecture for both pre-training and fine-tuning, with the exception of the output layers, like transformers for conventional language modeling. Its architecture is consistent with several downstream responsibilities. The downstream jobs labeled data is used to fine-tune all of the parameters after HaRT has been initialized with the pretrained parameters. Despite being started with the identical pretrained parameters, each downstream task has its own finetuned models. In addition to using the labeled data from the downstream activities, we also leverage previous messages from the relevant users (where available) to mimic the pre-training inputs format and to capitalize on the users experience.

On the relevant training datasets, we fine-tune HaRT for document-level tasks with an input instance cap of 8 blocks of 512 tokens each and no input instance cap during evaluation. Using train and dev sets, as well as history when available, we train for 10 epochs with a 1 user train batch size, 20 users.

4.4 Results

Table-3 shows that Baseline HaRT model performs better than model with modified output representations to include user-state and Table-4 signifies importance of user-state in HaRT model.

Feminism	Baseline	Exp-1	Exp-2
Accuracy	0.62	0.56	0.49
F1	0.63	0.58	0.49
Precision	0.69	0.65	0.54
Recall	0.62	0.56	0.50

Table 3: We compare HaRT baselines's performance on document level downstream task - Stance, against the the results from the two experiment models. (i) Exp-1 : Concatenating user-state with transformer output. (ii) Exp-2 : Using only the user-state as output.

Hate Speech	GPT2	HaRT
Accuracy	0.91	0.95
F1	0.92	0.94
Precision	0.93	0.95
Recall	0.90	0.94

Table 4: We compare HaRT's performance on document level downstream task - Hate Speech, against the the GPT2 model.

4.5 Analysis

1. Table 3 summarizes the performance of HaRT baseline against two other models:
   (a) HaRT model concatenating user-state with transformer output.
   (b) HaRT model using only the user-state as output.

   We see that the model using only the user-state vector for stance prediction performs worse when compared to the baseline model and the model using user-state with the transformer output.
   A possible explanation for this can be due to absence of document-level information when only using the user-state representation. This is evident as the baseline HaRT and HaRT using user-state with the transformer output perform better as both have document-level semantic information.Another reason can be the assumption we made about the label of the user (we considered the "mode" of the given labels for a particular user).

2. Figure 4 shows the PCA of user-state vectors. The vector of users with the same stance do not lie close to each other in the embedded space and all the users are sparsely distributed with no visible clusters based on their stance.

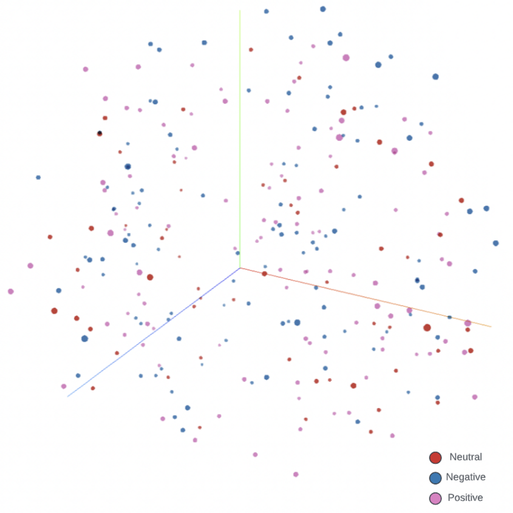


Figure 4: PCA of user-state vectors.

3. Table 4 summarizes the performance of HaRT baseline against GPT2 model on the document-level task of hate speech detection. We see that using the baseline HaRT model with user-state performs slightly better than the GPT2 model reinstating the indirect importance of user-state vector to model language by conditioning on a user-state.

4.6 Code

Github - `https://github.com/sugamxp/HaRT2`

5 Conclusions

1. Combining the user-state embedding to the last token representation of a document did not improve model performance for document-level tasks.

2. Visualizing the user-state embeddings did not lead to any concrete groupings of users against their labels for downstream tasks.

3. User-state addition to Hate Speech Detection task gave better results than baseline GPT2 model.

References

[1] Matthias R Mehl and James W Pennebaker. 2003. The sounds of social life: a psychometric analysis of students' daily social environments and natural conversations. Journal of personality and social psychology, 84(4):857.

[2] William Fleeson. 2001. Toward a structure-and process-integrated view of personality: Traits as density distributions of states. Journal of personality and social psychology, 80(6):1011.

[3] Dirk Hovy. 2015. Demographic factors improve classification performance. 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 752–762, Beijing, China. Association for Computational Linguistics.

[4] Veronica Lynn, Youngseo Son, Vivek Kulkarni, Niranjan Balasubramanian, and H Andrew Schwartz. 2017. Human centered nlp with user-factor adaptation. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1146–1155.

[5] Xiaolei Huang and Michael J Paul. 2019. Neural user factor adaptation for text classification: Learning to generalize across author demographics. In Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (* SEM 2019).

[6] Nikita Soni, Matthew Matero, Niranjan Balasubramanian, and H. Andrew Schwartz. 2022. Human Language Modeling. In Findings of the Association for Computational Linguistics: ACL 2022, pages 622–636, Dublin, Ireland. Association for Computational Linguistics.

[7] Saif M. Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In Proceedings of the International Workshop on Semantic Evaluation, SemEval '16, San Diego, California.

[8] Veronica Lynn, Salvatore Giorgi, Niranjan Balasubramanian, and H. Andrew Schwartz. 2019. Tweet classification without the tweet: An empirical examination of user versus document attributes. In Proceedings of the Third Workshop on Natural Language Processing and Computational Social Science, pages 18–28, Minneapolis, Minnesota. Association for Computational Linguistics.