Human Language Modeling

Nikita Soni, Matthew Matero,
Niranjan Balasubramanian, and **H. Andrew Schwartz**
Department of Computer Science, Stony Brook University
{nisoni, mmatero, niranjan, has}@cs.stonybrook.edu

Abstract

Natural language is generated by people, yet traditional language modeling views words or documents as if generated independently. Here, we propose *human language modeling* (HuLM), a hierarchical extension to the language modeling problem 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 introduce, HaRT, a large-scale transformer model for the HULM task, pre-trained on approximately 100,000 social media users, and demonstrate it's effectiveness in terms of both language modeling (perplexity) for social media and fine-tuning for 4 downstream tasks spanning document- and user-levels: stance detection, sentiment classification, age estimation, and personality assessment.[1] Results on all tasks 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; Fleeson, 2001). Indeed, different ways of incorporating human information into NLP models have recently been shown to improve accuracy on many NLP tasks (Hovy, 2015; Lynn et al., 2017; Huang and Paul, 2019; Hovy and Yang, 2021). At the same time, while language modeling has proven itself fundamental to NLP, it is typically absent the notion of a human producing the natural language.

From a statistical modeling perspective, this absence of human state can be seen as an instance of the *ecological fallacy* – the treatment of multiple observations (i.e. text sequences) from the same source (i.e. human) as independent (Piantadosi et al., 1988; Steel and Holt, 1996).

To address this, we introduce the task of *human language modeling* (HULM), which induces dependence among text sequences via the notion of a human state in which the text was generated. In particular, we formulate HULM as the task of estimating the probability of a sequence of tokens, $w_{t,1:i}$, while conditioning on a higher order state ($\mathbf{U}_{1:t-1}$) derived from the tokens of other documents written by the same individual. Its key objective is:

$$Pr(w_{t,i}|w_{t,1:i-1}, \mathbf{U}_{1:t-1})$$

where t indexes a particular sequence of temporally ordered utterances (e.g. a document or social media post), and $\mathbf{U}_{1:t-1}$ represents the human state just before the current sequence, t. In one extreme, $\mathbf{U}_{1:t-1}$ could model all previous tokens in all previous documents by the person. In the opposite extreme, $\mathbf{U}_{1:t-1}$ can be the same for all users and for values of t reducing to standard language modeling: $Pr(w_i|w_{1:i-1})$.[2] Thus, HULM-based models without history can be used where traditional LMs are applied (and may even perform better).

HULM brings together ideas from human factor inclusion/adaptation (Hovy, 2015; Lynn et al., 2017; Hovy and Yang, 2021) and personalized modeling (King and Cook, 2020; Jaech and Ostendorf, 2018) into the framework of large pretrained language models. Compared to traditional language modeling, HULM offers several technical advantages. First, the human state serves as a higher order structure that induces dependence between the text sequences of the same person/ thus posing a language modeling problem that is a more faithful treatment of human-generated natural language. Second, conditioning on prior texts of an individual can be seen as an implicit integration of text-derived human factors without having to explicitly model the identity of the individual.

[1] Code and pre-trained models available at: https://github.com/humanlab/HaRT.

[2] See section 3 for a full HULM definition.

Findings of the Association for Computational Linguistics: ACL 2022, pages 622 - 636
May 22-27, 2022 ©2022 Association for Computational Linguistics

This enables fine-tuning of such a model to many downstream tasks. Third, using the temporally ordered prior texts for human contexts can be seen as a way to track the dynamic nature of human states (e.g. emotions, daily activities) and be combined to yield more stable personality traits (e.g, extraversion, openness).

To build a language model that effectively addresses the HuLM task, we develop HaRT, a human-aware recurrent transformer. HaRT is built using a new user-state based attention layer, that connects standard word sequence transformer layers in order to incorporate the human context. The recurrent user state allows HaRT to effectively model long contexts necessary to handle all the previous messages written by an individual. We train HaRT on the HuLM task defined over a large collection of social media texts spanning 100K users and apply it (fine-tuning) on 2 downstream message-level tasks: stance detection (Mohammad et al., 2016), and sentiment analysis (Nakov et al., 2013) as well as 2 human-level tasks: age estimation and personality assessment (Schwartz et al., 2013).

Contributions. Our contributions are three-fold: (1) We introduce the task of human language modeling (HuLM), providing a mathematical definition and relation to traditional language modeling; (2) We propose HaRT, a novel transformer-based model for performing HuLM and capable of being fine-tuned to specific tasks including user-level tasks for which traditoinal language models cannot be applied without architectural alterations; (3) We evaluate HaRT, demonstrating state-of-the art performance on five tasks: social media language modeling (perplexity), two document-level tasks (sentiment analysis and stance detection), and two user-level tasks (personality–openness assessment, and age estimation).

2 Related Work

Recent advances in language model pre-training have led to learned representation of text. Pre-training methods have been designed with different training objectives, including masked language modeling (Devlin et al., 2019) and permutation-based auto-regressive language modeling (Yang et al., 2019). These have contributed in building deep *autoencoding* architectures, allowing the same pre-trained model to successfully tackle a broad set of NLP tasks. While pre-training over large collections of text helps models acquire many forms of linguistic and world knowledge(Petroni et al., 2019; Jiang et al., 2020; Rogers et al., 2020), they are still devoid of the information about the text creator.

Recently, it has been suggested that the NLP community address the social and human factors to get closer to the goal of human-like language understanding (Hovy and Yang, 2021). This call builds on a series of studies suggesting that integrating the human context into natural language processing approaches leads to greater accuracy across many applications in providing personalized information access (Dou et al., 2007; Teevan et al., 2005) and recommendations (Guy et al., 2009; Li et al., 2010; De Francisci Morales et al., 2012). The idea of contextualizing language with extra linguistic information has been the basis for multiple models: Hovy (2015) learn age- and gender-specific word embeddings, leading to significant improvements for three text classification tasks. Lynn et al. (2017) proposed a domain adaptaion-inspired method for composing user-level, extra-linguistic information with message level features, leading to improvements for multiple text classification tasks. Welch et al. (2020a) propose a new form of personalized word embeddings that use demographic-specific word representations.

In addition to addressing to social and human factors, recent work has also focused on *personalized* language models (King and Cook, 2020; Jaech and Ostendorf, 2018) learning author representations (Delasalles et al., 2019) and personalized word embeddings (Lin et al., 2017) pointing out the importance of personalized semantics in understanding language. Welch et al. (2020b) explore personalized versus generic word representations showing the benefits of both combined. While these models are trained for singular user, Mireshghallah et al. (2021) trains a single shared model for all users for personalized sentiment analysis. However, the approach is not scalable as it is still user specific and expects a unique user identifier.

While not the primary goal, human language modeling may yield effective approaches to extend the context during language modeling. Thus, an aspect of this work can be seen as part of the recent pursuit of sequence models that capture dependencies beyond a fixed context length (Dai et al., 2018; Beltagy et al., 2020). For example, Keskar et al. (2019) and Dathathri et al. (2019) propose controllable language generation using one or more attribute classifiers or control codes. Guu et al. (2020) propose augmented language model pretraining with a latent knowledge retriever which allows the model to retrieve and attend over documents from a large corpus. These models extend context limits, but they do not model the higher order structure capturing a notion of the common source of documents i.e., the author. On the other hand, Yoshida et al. (2020) fits a hierarchical model extension to language modeling by adding recurrence to a pretrained language model. This idea forms a basis for our proposed HuLM architecture, HaRT, but Yoshida et al. do not exploit the inherent higher order structure (i.e. the model was not used for HuLM).

3 Human Language Modeling (HuLM)

Our goal is to re-formulate the language modeling task into one that directly enables a higher-order dependence structure that represents a human generating the language.

Language modeling formulations pose probabilistic questions over text represented as sequences of tokens. The main goal is to model the probability of observing a given token sequence in the language as a whole. In particular language models (LMs) estimate the joint probability of the tokens in the string, defined in terms of the probabilities of each token in the sequence conditioned on the previous tokens.[3] Given a string $\mathbf{W} \in \mathbb{L}$, a sequence of n tokens $\langle w_1, w_2, \cdots, w_n \rangle$, the probability of observing the string $\mathbf{W}$ in the language $\mathbf{L}$ is computed as:

$$Pr(\mathbf{W}) = \prod_{i=1}^{n} Pr(w_i | w_{1:i-1}) \tag{1}$$

We pose the *human language modeling* problem (HuLM), where the goal is to model the probabilities of observing a sequence from the language as generated by a specific person. An initial idea might be to pose this task as conditioning the probability of a string, w_i on a static representation of the person (or user, $\mathbf{U}_{static}$):

$$Pr(\mathbf{W}|\mathbf{U}_{static}) = \prod_{i=1}^{n} Pr(w_i | w_{1:i-1}, \mathbf{U}_{static}) \tag{2}$$

This addresses the first of the two goals we presented in the introduction, namely avoiding the *ecological fallacy* of assuming sequences from the same person are independent. However, it does not respect the idea that people vary in mood and can change. More precisely, human behaviors (language use) are influenced by dynamic human states of being (Fleeson, 2001; Mehl and Pennebaker, 2003). Thus, we pose HuLM with a more general formulation that enables the idea of a dynamic representation of humans, the user state $\mathbf{U}_t$[4]:

$$Pr(\mathbf{W}_t|\mathbf{U}_{t-1}) = \prod_{i=1}^{n} Pr(w_{t,i} | w_{t,1:i-1}, \mathbf{U}_{1:t-1}) \tag{3}$$

where t indexes a particular sequence of temporally ordered utterances (e.g. a document, or set of social media message). While $w_{t,i}$ is drawn from a multinomial distribution, $\mathbf{U}_{1:t-1}$ can be from any discrete or continuous multivariate distribution.

In one extreme, $\mathbf{U}_{1:t-1}$ could model all previous tokens in all previous documents by one person. In the opposite extreme, $\mathbf{U}_{1:t-1}$ can be the same for all values of t, giving a static representation for a user (equivalent to Equation 2) or even static across users which reduces to a standard language modeling version (equivalent to Equation 1). Still, modeling a user via their previous documents provides a seamless way to integrate the user information into language models – the only change is that the models will now have to incorporate more text when they are making predictions. Note that this problem formulation does not directly require explicit modeling of the identity of a user. This makes it easier to handle new users in downstream tasks and test instances, or creating models that can be further fine-tuned to both document- and user-level tasks.

HuLM in Practice. Like traditional langauge models, there are two steps to applying HuLM based models to most tasks and applications: pretraining and fine-tuning. During pre-training, the

[3] Traditional LMs provide estimates of the conditional probabilities often relying on further simplifying assumptions (e.g. Markovian assumptions to handle long sequences.).

[4] We define $\mathbf{U}_t$ as the state *after* the sequence, $\mathbf{W}_t$. Thus, only $\mathbf{U}_{t-1}$ is accessible as given when estimating Pr($\mathbf{W}_t$) conditioned on the user state.

model is trained on unlabeled data over Human Language Modeling (HuLM) pre-training task above. For finetuning, a HuLM based model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters.

4 Human-aware Recurrent Transformer

This section introduces, HaRT, a human-aware recurrent transformer that trains on the human language modeling (HULM) formulation.

HaRT is designed to produce human-aware contextual representations of text at multiple levels. HaRT's design is motivated by two goals: (i) We want to support hierarchical modeling, i.e., to hierarchically represent the set of all-messages written by a user and at the same time have human-aware contextual word representations. This implicitly entails modeling large context size. For example, GPT-2 (Radford et al., 2019) uses a context size of 1024 tokens, whereas our estimate of the average context size for a Twitter user is more than 12000 tokens. (ii) To support user-level tasks (e.g. personality assessment (Lynn et al., 2020)), we need representations of the entire set of messages written by a user capturing the inherent human states that broadly encompasses the user representation.

HaRT addresses the hierarchical language modeling issue by processing all messages written by a user in a temporally ordered sequence of blocks. It uses a recurrence structure to summarize information in each block into a user state vector, which is then used to inform the attention between tokens in the subsequent block. For human-level tasks the aggregate of user states can be used as the representation of the entire context for the user.

The idea of adding recurrence to pre-trained transformers builds on Yoshida et al. (2020)'s method for handling long contexts. However, the main difference is that 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.

4.1 HaRT Architecture

Figure 1 shows the overall architecture for HaRT. It consists of a one modified transformer layer with a user-state based self-attention mechanism over more token-level standard self-attention based transformer layers from a pre-trained transformer (GPT-2).

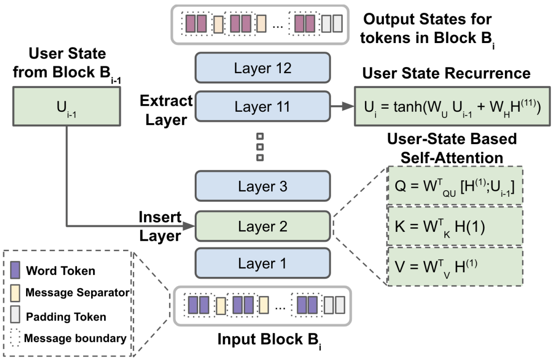

Figure 1: HaRT architecture: HaRT processes a user's messages in blocks. It produces contextualized representations of messages in each block conditioning on a recurrently computed user state. The user state is inserted into an earlier layer (layer 2) to inform the self-attention computation via a modified query transform. The previous user state is then recurrently updated using the output of a later layer (layer 11).

Inputs and Outputs Each input instance to HaRT consists of a temporally ordered sequence of messages (by message created time) from a given user a, $\mathcal{M}_a = \langle M_1, \cdots, M_n \rangle$. We segment these messages into fixed sized blocks, $\mathcal{B}_a = \langle B_1, \cdots, B_k \rangle$. We sequentially fit messages into blocks, separating messages using a newly introduced special token $< |insep| >$. If the number of tokens in a block falls short of the block size, we fill it with padded tokens. k is a hyperparameter during training used to cap the maximum number of blocks controlling the amount/size of user history that is fed to the model. If the messages for a user fill less than k blocks, we pad the rest to maintain the same size for each instance.

For each block B_i, 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. We use 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.

User-State based Self-Attention HaRT constructs a user-state representation vector by combining information from each block in a recurrent manner. After processing the inputs in a given block B_i, HaRT extends the previous user state U_{i-1} with information from current block B_i using the output representations $H^{(E)}$ from one of the later layers (we denote as the extract layer L_E). The recurrence for the new user state U_i is:

$$U_i = tanh(W_U U_{i-1} + W_H H^{(E)}) \qquad (4)$$

The user state for the first block U_0 is initialized with the average of the (pretrained GPT-2) layer 11 outputs for words from the messages of more than 500 users (of the train set) computed using Schwartz et al. (2017).

To produce the user-state conditioned contextual representations at a given layer, HaRT uses a modified self-attention procedure to one of the earlier layers, which we denote as the insert layer (L_{IN}). The idea is to create a new query transform which includes the user-state vector, so that the attention between tokens is informed by the context of the previous messages written by the user. To this end, we take input hidden states to this insert layer H_i^{IN-1}, concatenate it with the user-state vector from the previous block U_{i-1} and then apply a linear transformation (using W_q) to obtain the query vectors (Q_i^{IN}) for the self-attention computation.

$$Q_i^{IN} = W_q^T [H_i^{(IN-1)}; U_{i-1}] \qquad (5)$$

The key, value transforms and the rest of the self-attention computation and further processing in the transformer to produce the output representations from the layer, all remain the same as in the original GPT-2 model.

Implementation Choices There are multiple alternatives for a HaRT implementation including how to construct the user state, where and how to inject user state information. In our preliminary experiments we experimented with different extract layers but found that constructing user state from the penultimate layer (Layer 11) and injecting the user state in a single earlier layer (Layer 2 used by Yoshida et al. (2020)) to modify the query transformation was the most effective empirically.

4.2 Pre-training HaRT

HaRT is pre-trained using the HuLM task in an autoregressive manner.

The HuLM task as defined in Equation 3 asks to predict a token that appears in a token sequence (i.e. a user's social media message) given the previous tokens in the sequence while also conditioning on previous user states. We turn this task into a pre-training objective defined over block segmented token sequences from a user. For each block of a given user, the task is to predict each token in the block while conditioning on (i) the previous tokens within the current block which are directly available as input, and also (ii) the tokens from the previous blocks that are available to HaRT through the recurrent user state. Formally, the pre-training objective is to maximize:

$$\prod_{a \in \text{Users}} \prod_{t=1}^{|\mathcal{B}_a|} \prod_{i=1}^{|B_t^{(a)}|} Pr(w_{t,i} | w_{t,1:i-1}, B_{1:t-1}^{(a)}) \qquad (6)$$

where, $w_{t,i}$ is the i^{th} token in the t^{th} block ($B_t^{(a)}$) for user a.

Pre-training data For the pre-training corpus we combine a subset of the Facebook posts dataset from Park et al. (2015), a subset of the County Tweet Lexical Bank (Giorgi et al., 2018) appended with newer 2019 and 2020 tweets, in total spanning 2009 through 2020. We filter the datasets to only include tweets marked as English from users who have at least 50 total posts and at least 1000 words in total, ensuring moderate language history for each user. The resulting dataset consists of just over 100,000 unique users, which we split into a train dataset consisting of messages from 96,000 users, a development dataset that consists of messages from 2000 users that were not part of the training set (unseen) and new messages from 2500 users seen in the training set, and a test set of messages from a separate set of 2000 unseen users that are neither in training or the development set.

We refer to this as the HuLM-Corpus (HLC).

4.3 Fine-tuning HaRT

In the tradition of transformers for traditional language modeling, HaRT shares the same architecture for both pre-training and fine-tuning except for the output layers. It has a unified architecture across different downstream tasks. For finetuning, HaRT is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters. Apart from using the labeled data from the downstream tasks, we also use the historical messages (when available) from the respective users to replicate the format of pre-training inputs and to benefit from the knowledge of the user.

5 Evaluation: Human Language Modeling

We seek to compare HaRT with a standard language model that is exposed to the same data but without modeling the notion of a user. Thus, we compare HaRT's human language modeling performance to the model it was based, GPT-2. For calibration we report performance on GPT-2's original pre-trained version (GPT-2$_{\text{frozen}}$), and a version of the LM that was fine-tuned on the HuLM-Corpus (GPT-2$_{\text{HLC}}$).

We train and evaluate the models using the train and test splits of the HuLM-Corpus described in Section 4.2. For hyperparameter search, we use the full development set of both seen and unseen users. Each training instance for HaRT is capped to 8-blocks of 1024-tokens each. Following previous work fine-tuning transformer language models for social media (V Ganesan et al., 2021), GPT-2 was trained over individual messages. We train both for five epochs and set the learning rate, batch size, and stopping patience based on the development set (see Appendix A.3). For HaRT, we initialize all GPT-2 self-attention layers with the corresponding weights in the pre-trained GPT-2. The user-state based self-attention layer weights (query, key, and value) are normal initialized with 0 mean and 0.02 standard deviation.

Perplexity Table 1 reports the perplexity of all three models on the messages from the unseen users of the development split and the entire test split of HuLM-Corpus. The frozen pre-trained GPT-2 (GPT-2$_{\text{frozen}}$) fares poorly to the domain mismatch while the fine-tuned version (GPT-2$_{\text{HLC}}$) fares much better. However, the human language model HaRT achieves the best performance by a large margin, with a significant reduction in perplexity by more than 46% on the test set relative to GPT-2$_{\text{HLC}}$ ($p < .001$).[5]

Effect of History Size. We further analyze the effect of history size by varying the amount of language, in terms of blocks, used per user. Figure 2 shows that adding more history in general helps, with a big reduction in perplexity going from 2 to 4 blocks and a further reduction going from 4 to 8 blocks. Adding more context can induce a need to effectively balance likelihood of finding more important signals against the increasing chances of it drowning in less important information.

Model	Dev (*ppl*)	Test (*ppl*)
GPT-2$_{\text{frozen}}$	112.82	116.35
GPT-2$_{\text{HLC}}$	47.61	48.51
HaRT	**27.49***	**26.11***

Table 1: Comparing HaRT as a language model to GPT-2$_{\text{frozen}}$, the frozen pre-trained GPT-2 and GPT-2$_{\text{HLC}}$, the GPT-2 model fine-tuned on the HuLM-Corpus. HaRT shows large gains with a substantial reduction in perplexity compared to both versions of GPT-2. Bold font indicates best in column and * indicates statistical significance $p < .05$ via permutation test w.r.t GPT-2$_{\text{HLC}}$

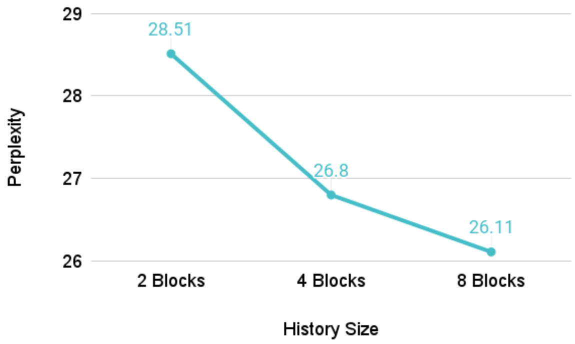

Figure 2: . Perplexity scores, on test sets as a function of history size (number of blocks) used when training HaRT. Each block consists of 1024 tokens. Adding more history improves language modeling performance with big reduction going from 2 to 4 blocks and a smaller reduction from 4 to 8 blocks.

6 Evaluation: Fine-tuning for Downstream Tasks

Here, we evaluate the utility of fine-tuning HaRT for document- and user-level tasks. Just as standard transformer language models are fine-tuned for tasks, we take our pre-trained HaRT model and fine-tune it for stance detection, sentiment classification, age estimation, and personality (openness) assessment tasks. For both sets of tasks we compare fine-tuning the GPT-2$_{\text{HLC}}$ as a non-user-based LM baseline and also report previously published results from other task specific models, most of which employ historical context for re-

[5] In addition to this improvement for unseen users, we also see similar relative benefits when tested on instances from seen users which we report in Appendix A.2.

Model	*Age* (r)	*OPE* (r_{dis})	*Stance* ($F1$)	*Sentiment* ($F1$)
GPT-2HLC	0.839	0.521	68.60	76.75
HaRT	**0.868***	**0.619***	**71.10***	**78.25***

Table 2: We fine-tune HaRT and GPT-2HLC (GPT-2 fine-tuned for LM on the same data) for 4 downstream tasks: Age, Openness (OPE), Stance, and Sentiment, and find HaRT to perform better on all 4 tasks. For age and openness, we fine-tune HaRT only for the recurrence module, and fine-tune only the last 2 layers of GPT-2HLC. For stance and sentiment, we fine-tune full models. Results are reported in pearson r for Age, disattenuated pearson r for OPE and weighted F1 for Stance/Sentiment. Bold indicates best in column and * indicates statistical significance $p < .05$ via permtuation test.

Model	*Stance* (*F1*)	*Sentiment* (*F1*)
MFC	54.2	28.0
Lynn et al. (2019)	65.9	69.5
MeLT	66.6	63.0
BERTweet	68.8	77.9
HaRT	**71.1***	**78.3***

Table 3: We compare HaRT's performance on document level downstream tasks: Stance and Sentiment, against state of the art results. We also fine-tuned pretrained GPT-2, BERTweet (Nguyen et al., 2020), and MeLT (Matero et al., 2021) on both tasks for baselines. HaRT performs the best in both tasks with a substantial gain. Results are reported in weighted F1. Bold indicates best in column and * indicates statistical significance $p < .05$ w.r.t BERTweet via permutation test.

spective tasks. All hyperparameter settings and training details for the GPT-2HLC and HaRT models for each task are listed in Appendix A.3.

6.1 Document-Level Tasks

We consider two document-level tasks that require models to read an input document (message) written by a user and output a label (stance of the user towards a topic or the sentiment expressed in the text). To fine-tune HaRT on these tasks, with each document we collect and attach previous messages written by the same users, represented using the procedure we outlined in Section 4.3. Thus, HaRT processes this input to produce message- and human-contextualized token-level representations. We represent the document by its last non-padded token representation and feed it to classification layer with a prior layer norm for predicting the output label. GPT-2HLC, without hierarchical structure, only uses the input document to make predictions. We fine-tune all parameters of HaRT and GPT-2HLC, as well as the classification layer weights using the standard cross-entropy loss (calculated only over the last non-padded token of the target (labeled) messages).

Stance Detection. For stance detection we use the SemEval2016 dataset (Mohammad et al., 2016), 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) 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.

Sentiment Analysis. We use message-level sentiment annotations indicating positive, negative, and neutral categories from the SemEval-2013 dataset (Nakov et al., 2013). As with stance, we use a part of the extended dataset from Lynn et al. (2019) to get associated message history, and preserve the train/dev/test split of the same.

6.2 User-Level Tasks

We evaluate HaRT for age estimation and personality (openness) assessment, social scientific tasks which require producing outcomes at the user-level. We use a subset of the data from consenting users of Facebook who shared their Facebook posts along with demographic and personality scores (Kosinski et al., 2013; Park et al., 2015).

For these user-level tasks we can leverage the recurrent user states in HaRT to produce a representation of the user. We represent the input as described in Section 4.3, and use the average of the user-states vectors from the non-padded blocks of each user and layer norm it to make predictions using a linear classifying layer to predict 1 label (regression task). We use only 4 blocks of history when training to fine-tune.

For GPT-2HLC, since it can't directly handle all of the users text in one go, we replicate the user label for each message of the respective users and train the model to predict the label for each message using the last non-padded token of the message. To make the final prediction, we average the predictions across all messages from respective users and calculate the performance metric using this average as in (V Ganesan et al., 2021).

For these user level tasks that require aggregate information, for both models, fine-tuning the entire set of parameters was worse than fine-tuning fewer layers. For GPT-2HLC fine-tuning only the last two layers gave the best performance. For HaRT fine-tuning only the recurrence module gave the best performance on development sets. We report results with these best dev settings. We use the mean squared error (MSE) as the training loss.

Age Estimation Similar to the pre-training data, we filtered the above dataset for English language instances and included only the users with a minimum of 50 posts and a minimum of 1000 words. Age was self-reported and limited to those 65 years or younger. This resulted in a dataset of 56,930 users in train, 1836 users in dev, and 4438 users in test which was a subset of the test set (5000 users) from Park et al. (2015). We evaluate on both the test sets and report Pearson correlation (r) metric on the latter for comparison purposes. We include results with the filtered data in Appendix (Table 8).

Personality Assessment. We evaluate on the assessment of openness based on language (one's tendency to be open to new ideas) (Schwartz et al., 2013). To allow for direct comparisons, we use the same test set (n=1,943) as Lynn et al. (2020) and use a subset of their training set (66,764 users) of which 10% were sampled as dev set, and report disattenuated pearson correlation (r_{dis}) to account for questionnaire reliability Lynn et al. (2018). As with age estimation, we report results with the filtered dataset in Appendix (Table 8).

6.3 Results

Table 2 summarizes the performance of HaRT against the baseline of fine-tuning a non-human-aware language model, GPT-2HLC. We see that HaRT yields substantial gains over GPT-2HLC across both user-level and document-level tasks, demonstrating clear benefits in all settings.

Document-Level Tasks Table 3 compares HaRT with task-specific baselines for stance and sentiment detection including (i) Lynn et al. (2020) which used historical contexts to incorporate both explicit and text-derived latent human factors, (ii) MeLT (Matero et al., 2021) which used a superset of the same historical contexts used here but for message-level language modeling, and (iii) BERTweet (Nguyen et al., 2020) which uses a large collection of tweets to pretrain an autoencoder that is then fine-tuned for target tasks. Sentiment results are weighted F1 scores over the three sentiment categories. Stance results are an average of weighted F1 scored over five different topics from respective topic-specific fine-tuned models. HaRT outperforms all models demonstrating the substantial benefits of human language modeling for these document-level downstream tasks.

Model	*Age* (r)	*OPE* (r_{dis})
V Ganesan et al. (2021)	0.795	0.511
Sap et al. (2014)	0.831	-
Lynn et al. (2020)	-	**0.626**
HaRT	**0.868***	**0.619**

Table 4: Comparison of HaRT's performance on user level downstream tasks: Age and Openness (OPE), against state of the art results. V Ganesan et al. (2021) use lesser number of users (10000) in training. Results are reported in pearson r for Age and disattenuated pearson r for OPE. Bold indicates best in column and * indicates statistical significance between HaRT and (Sap et al., 2014) ($p < .05$) using a bootstrap sampling test. We also find no statistical difference between HaRT and (Lynn et al., 2020) ($p = .35$).

User-Level Tasks Table 4 compares HaRT with task-specific baselines for Age and Openness tasks that use the superset of the same data used by HaRT. For Age, HaRT outperforms all baselines including a strong non-neural lexica based predictor (Sap et al., 2014), and a RoBERTa-based system that uses carefully chosen frozen embeddings (V Ganesan et al., 2021). For Openness, HaRT is better than the frozen RoBERTa (Liu et al., 2019) embeddings and is comparable to Lynn et al. (2020)'s hierarchical attention model. These results also suggest the potential of HaRT's user states as a representation for user-level tasks.

6.4 No Historical Context.

HaRT can also be used anywhere a typical transformer language model is used by simply not feeding any historical context. Here, we seek to use our pre-trained HaRT as a language model that is fine-tuned to the messages (for the respective tasks) without any historical context. Table 5 com-

Model	*Sentiment* *($F1$)*	*Stance* *($F1$)*
GPT-2HLC frozen	62.7	57.7
HaRT nohist, frozen	62.7	58.6
GPT-2HLC	76.8	68.6
HaRT nohist	**77.7***	**70.8***

Table 5: Results with experiments on Stance and Sentiment downstream tasks using only the labeled instances and no history. We compare HaRT with GPT-2HLC by training only the classification head (*frozen*) and additionally, by fine-tuning the models. Bold indicates best in column and * indicates statistical significance $p < .05$ via permutation test w.r.t GPT-2HLC. Results are reported in weighted F1.

Model	*Sentiment* *($F1$)*	*Stance* *($F1$)*
HaRT NOT PT	63.47	66.26
HaRT W/O RECUR	77.04	68.73
HaRT	**78.25***	**71.10***

Table 6: Results with the ablation experiments on Stance and Sentiment downstream tasks. We experiment without the recurrence module (W/o recur), and HaRT without HuLM PT, and compare with HaRT. Bold indicates best in column and * indicates statistical significance $p < .05$ via permutation test w.r.t HaRT w/o recur. Results are reported in weighted F1.

pares the performances of HaRT and GPT-2HLC for the two document-level downstream tasks Stance, and Sentiment. For a fair comparison, we use the same data inputs for both the pre-trained models which consists of only the labeled messages and no historical context. We evaluate in 2 ways: 1) freezing the model and training only the classification layer using the outputs from the penultimate transformer layer, and 2) fine-tuning all model parameters along with a classification head with a layer norm prior to it. HaRT is at par or better with GPT-2HLC for both frozen and fine-tuned versions, showing that it can provide gains even when historical context is unavailable. Hyperparameters settings are described in Appendix A.3.

6.5 Ablation Studies

In this section, we perform ablation experiments on HaRT to better understand their relative importance and report the results in Table 6.

Pre-training We assess the impact of pre-training by evaluating the downstream performance of a version of the HaRT model that has not been pre-trained on the HuLM task. Instead of using the weights from HuLM pre-training, we use HaRT with initialized weights as described in Section 5. Table 6 shows HuLM pre-training benefits – pre-training adds substantial gain of 14.78 points and 4.84 points in weighted F1 for sentiment analysis and stance detection respectively.

Recurrence We assess the importance of recurrent user state by first pre-training HaRT without its recurrent module and then fine-tuning it for the downstream tasks. We still use the same batching as described in Section 4.2 but the information from a block no longer propagates to the next block in the forward pass, and backpropagation is still done on all blocks of a user together. Without the recurrence module we see a drop of 1.21 points and 2.37 points in the weighted F1 measure for sentiment and stance respectively. Interestingly, HaRT outperforms HaRT without recurrence, consistent with the idea that models benefit from user history on tasks that involve a user.

7 Conclusions

Language is deeply human. Yet, language models in wide-spread use today lack a notion of the human that generates the language. Motivated by other advances in human-centered language processing and psychological theory that suggest language is moderated by human states, we introduced *human language modeling*. HULM extends LMs with the notion of a user and their states via their previous messages. In this first step toward large human language models, we developed a human-aware transformer (HaRT) that uses a recurrence mechanism to model user states and show that pre-training this transformer on the human language modeling task yields significant gains in both generation and fine-tuning for multiple downstream document- and user-level tasks.

Overall, state-of-the-art results with HaRT, a model neither trained on substantially larger data nor adding many parameters, suggests progress for transformers not based on massive increases in data or parameters but on a task grounded in language's "natural" generators, people.

8 Ethical Considerations

While the multi-level human-document-word structure within HULM can enable bias correcting and fairness techniques (discussed next), the ability to better model language in its human context also presents opportunities for unintended harms or nefarious exploitation. For example, models that improve psychological assessment are not only useful for research and clinical applications, but could be used to target content for individuals without their awareness or consent. In the context of use for psychological research, such models may risk release of private research participant information if trained on private data without checks for exposure of identifying information. To negate this potential, we only release a version of HaRT that is without training on the consented-use private Facebook data until differential privacy standards can be verified. Unlike other human-centered approaches, HaRT is not directly fed user attributes as part of the pre-training thus the model parameters do not directly encode user attributes.

HuLM aims to join a growing body of work to make AI more human-centered, and thus more applicable for interdisciplinary study of the human condition as well as leading to new clinical tools for psychological health. At this point, our models are not intended to be used in practice for mental health care nor labeling of individuals publicly with personality or age scores. While modeling the human state presents opportunities for reducing AI bias, prior to clinical or applied use, such models should be evaluated for failure modes such as error across target populations for error or outcome disparities (Shah et al., 2020). All user-level tasks presented here were reviewed and approved or exempted by an academic institutional review board (IRB).

9 Acknowledgments

This work was supported by DARPA via Young Faculty Award grant #W911NF-20-1-0306 to Stony Brook University; the conclusions and opinions expressed are attributable only to the authors and should not be construed as those of DARPA or the U.S. Department of Defense. This work was also supported in part by NIH R01 AA028032-01 and by the National Science Foundation under the grant IIS-1815358.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Zihang Dai, Zhilin Yang, Yiming Yang, William W. Cohen, J. Carbonell, Quoc V. Le, and R. Salakhutdinov. 2018. Transformer-xl: Language modeling with longer-term dependency.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*.

Gianmarco De Francisci Morales, Aristides Gionis, and Claudio Lucchese. 2012. From chatter to headlines: harnessing the real-time web for personalized news recommendation. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 153–162.

Edouard Delasalles, Sylvain Lamprier, and Ludovic Denoyer. 2019. Learning dynamic author representations with temporal language models. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 120–129. IEEE.

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.

Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. 2007. A large-scale evaluation and analysis of personalized search strategies. In *Proceedings of the 16th international conference on World Wide Web*, pages 581–590.

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.

Salvatore Giorgi, Daniel Preoţiuc-Pietro, Anneke Buffone, Daniel Rieman, Lyle Ungar, and H. Andrew Schwartz. 2018. The remarkable benefit of user-level aggregation for lexical-based population-level predictions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1167–1172, Brussels, Belgium. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

Ido Guy, Naama Zwerdling, David Carmel, Inbal Ronen, Erel Uziel, Sivan Yogev, and Shila Ofek-Koifman. 2009. Personalized recommendation of social software items based on social relations. In *Proceedings of the third ACM conference on Recommender systems*, pages 53–60.

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.

Dirk Hovy and Diyi Yang. 2021. The importance of modeling social factors of language: Theory and practice. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 588–602, Online. Association for Computational Linguistics.

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)*.

Aaron Jaech and Mari Ostendorf. 2018. Personalized language model for query auto-completion. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 700–705, Melbourne, Australia. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, J. Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Milton King and Paul Cook. 2020. Evaluating approaches to personalizing language models. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2461–2469, Marseille, France. European Language Resources Association.

Michal Kosinski, David Stillwell, and Thore Graepel. 2013. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, 110(15):5802–5805.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670.

Zih-Wei Lin, Tzu-Wei Sung, Hung-Yi Lee, and Lin-Shan Lee. 2017. Personalized word representations carrying personalized semantics learned from social network posts. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 533–540. IEEE.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Veronica Lynn, Niranjan Balasubramanian, and H. Andrew Schwartz. 2020. Hierarchical modeling for user personality prediction: The role of message-level attention. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5306–5316, Online. Association for Computational Linguistics.

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.

Veronica Lynn, Alissa Goodman, Kate Niederhoffer, Kate Loveys, Philip Resnik, and H. Andrew Schwartz. 2018. CLPsych 2018 shared task: Predicting current and future psychological health from childhood essays. In *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic*, pages 37–46, New Orleans, LA. Association for Computational Linguistics.

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.

Matthew Matero, Nikita Soni, Niranjan Balasubramanian, and H. Andrew Schwartz. 2021. MeLT: Message-level transformer with masked document representations as pre-training for stance detection. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2959–2966, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Fatemehsadat Mireshghallah, Vaishnavi Shrivastava, Milad Shokouhi, Taylor Berg-Kirkpatrick, Robert Sim, and Dimitrios Dimitriadis. 2021. Useridentifier: Implicit user representations for simple and effective personalized sentiment analysis. *arXiv preprint arXiv:2110.00135*.

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.

P Nakov, S Rosenthal, Z Kozareva, V Stoyanov, A Ritter, and T Wilson. 2013. Task 2: Sentiment analysis in twitter. In *Proceedings of the 7th International Workshop on Semantic Evaluation, Atlanta, Georgia*.

Dat Quoc Nguyen, Thanh Vu, and Anh-Tuan Nguyen. 2020. Bertweet: A pre-trained language model for english tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14.

Gregory J. Park, H. A. Schwartz, J. Eichstaedt, Margaret L. Kern, M. Kosinski, D. Stillwell, L. Ungar, and M. Seligman. 2015. Automatic personality assessment through social media language. *Journal of personality and social psychology*, 108 6:934–52.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *ArXiv*, abs/1909.01066.

Steven Piantadosi, David P Byar, and Sylvan B Green. 1988. The ecological fallacy. *American journal of epidemiology*, 127(5):893–904.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Maarten Sap, Gregory Park, Johannes Eichstaedt, Margaret Kern, David Stillwell, Michal Kosinski, Lyle Ungar, and Hansen Andrew Schwartz. 2014. Developing age and gender predictive lexica over social media. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1146–1151, Doha, Qatar. Association for Computational Linguistics.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin EP Seligman, et al. 2013. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PloS one*, 8(9):e73791.

H Andrew Schwartz, Salvatore Giorgi, Maarten Sap, Patrick Crutchley, Lyle Ungar, and Johannes Eichstaedt. 2017. Dlatk: Differential language analysis toolkit. In *Proceedings of the 2017 conference on empirical methods in natural language processing: System demonstrations*, pages 55–60.

Deven Santosh Shah, H Andrew Schwartz, and Dirk Hovy. 2020. Predictive biases in natural language processing models: A conceptual framework and overview. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5248–5264.

David G Steel and D Holt. 1996. Analysing and adjusting aggregation effects: the ecological fallacy revisited. *International Statistical Review/Revue Internationale de Statistique*, pages 39–60.

Jaime Teevan, Susan T Dumais, and Eric Horvitz. 2005. Personalizing search via automated analysis of interests and activities. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 449–456.

Adithya V Ganesan, Matthew Matero, Aravind Reddy Ravula, Huy Vu, and H. Andrew Schwartz. 2021. Empirical evaluation of pre-trained transformers for human-level NLP: The role of sample size and dimensionality. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4515–4532, Online. Association for Computational Linguistics.

Charles Welch, Jonathan K. Kummerfeld, Verónica Pérez-Rosas, and Rada Mihalcea. 2020a. Compositional demographic word embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4076–4089, Online. Association for Computational Linguistics.

Charles Welch, Jonathan K. Kummerfeld, Verónica Pérez-Rosas, and Rada Mihalcea. 2020b. Exploring the value of personalized word embeddings. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6856–6862, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Davis Yoshida, Allyson Ettinger, and Kevin Gimpel. 2020. Adding recurrence to pretrained transformers for improved efficiency and context size. *arXiv preprint arXiv:2008.07027*.

A Appendix

A.1 Pre-training

Twitter Data Collection As mentioned, in section 4.2, we use a combination of data from both Twitter and Facebook data sources. However, since the main Twitter corpus (Giorgi et al., 2018) only spans the years 2009 - 2015, we wanted to supplement our total corpus with newer language data. Generally, we follow the same procedures for data collection as introduced for the 2009 - 2015 years. Thus, we started with a 1% random sample of *publicly available* tweets that can be mapped to US counties. On top of this we also applied the following filters: (1) Removal of non-English tweets, (2) Removal of users who did not tweet at least 3 times a week, (3) Removal of any duplicates among the collected data, and (4) Removal of any tweets containing URLs. We will be including this additional data as part of the CTLB project[6].

Data Size and Splits We sample evenly between Facebook and Twitter at the user-level to collect 50,000 from each and apply the same minimum language use requirement of 1,000 words spanning 50 messages. We show the details of the splits across training/development/testing as well as seen/unseen user categories in figure 3. We keep 4,000 users for development and testing, 2k for each split, that are not at all present in the training portion. For users that we do train on, we select 4,500 to keep 20% of their messages for development and testing sets.

A.2 Perplexity on Seen versus Unseen Users

Benefit of Seen users. By default, our experiments are run under an 'unseen user' condition where by the test corpus contains users that were not in HaRT's training corpus. However, one could argue that this is an unnecessary impairment since further training the human language model doesn't require labels and can often be run on test data. We compare the effect of having seen users during HaRT training by additionally calculating perplexity on test sets with seen users. To make it a fair comparison, since we found our "seen user" corpus was more difficult (perplexity on seen users test set was higher than unseen users test set for GPT-2HLC as well), we use an adjusted perplexity, defined as the ratio of the model's perplexity divided by a non-HuLM upper-bound perplexity on the same test set (GPT-2HLC), normalizing by the difficulty of the test set. As shown in Table 7, we find a small but significant benefit to having seen the users during training.

	Unseen users		Seen users	
Model	*ppl*	*adj-ppl*	*ppl*	*adj-ppl*
GPT-2HLC	48.5	1.00	53.7	1.00
HaRT	27.5	0.57*	27.6	**0.51***

Table 7: Evaluation of benefit of having seen the users during HaRT training. We use adjusted perplexity (*adj-ppl*): the ratio of the perplexity to the upper-bound from not using HaRT during training (i.e.GPT-2HLC) on the same test set – lower implies better performance when normalized by difficulty of the test set. Seen users test set is the set with the messages from the users also available in the train set, while unseen users test set does not have users common with the train set and is the same as the test set in Table 1. Seen users test set is harder for both models. However, normalizing the scores show HaRT to have better performance over seen users test set. Bold font indicates best in column and * indicates statistical significance $p < .05$ via permutation test.

A.3 Experimental Settings

We use Open AI's pre-trained GPT-2 base model from Radford et al. (2019) made available by the Hugging Face library from Wolf et al. (2019) (transformers version 4.5.1) as our base model. We also make use of Hugging Face's code base to implement HuLM. Our training procedure involves all the default training hyperparameters from Hugging Face's GPT2 config except learning rate and the other specific hyperparams mentioned in the paper. We run a learning rate search sweep on a sampled dataset, for both HaRT and GPT-2HLC, using the Optuna framework from Akiba et al. (2019): 1) in a range of 5e-6 to 5e-4, with 3 trials each of 5 epochs for pre-training, 2) in a range of 5e-6 to 5e-4, with 10 trials each of 15 epochs for fine-tuning stance detection, and 3) in a range of 1e-7 to 1e-5, with 5 trials each of 15 epochs for fine-tuning sentiment analysis. We also setup an early stopping criteria for the downstream task trials, such that we continue the epoch runs till we hit an increase in loss for 3 consecutive runs, and pick the model with the best

[6] https://github.com/wwbp/county_tweet_lexical_bank

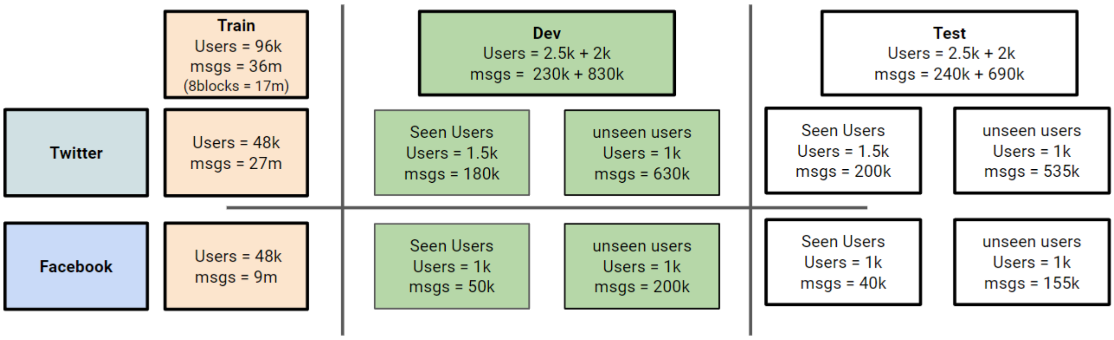

Figure 3: Structure of our pre-training dataset visually showing the data source(FB vs Twt), training/development/testing splits, and seen/unseen users for training and testing. Our dataset totals 100,000 users and approximately 37 million messages. Due to GPU memory restrictions, we limited training to 8 blocks of history which brought our train dataset size to 17 million messages. Dev and Test sets were not limited during evaluation.

F1 score. We couldn't run a similar sweep for user-level tasks due to compute time limits so we try a couple learning rates from document-level tasks but found the same learning rate that we use for pre-training to be better. Many of the experimental/hyperparameters (batch sizes, window sequence sizes and cappings) settings mentioned throughout this work including the number of trials and the number of epochs vary because of computational limitations based on data size and training time.

All pre-training runs are trained on 2 Tesla V100 GPUs of 32GB. Training HaRT takes approx 16 hours for 1 epoch (with train data consisting of 8 blocks (each of 1024 tokens) of 96000 users). Fine-tuning tasks run on a mix of Tesla V100, Quadro RTX 8000, and A100 GPUs based on compute availability. All batch sizes mentioned are per GPU.

Pre-training Settings We use 2.4447e-4 as the learning rate for training HaRT, with 1 user train batch size, 15 users eval batch size and early stopping patience set to 3. For GPT-2$_{\text{HLC}}$, we use the default settings from Wolf et al. (2019) with train and eval batch size set to 60 and early stopping patience set to 3.

Document-level Fine-tuning Settings We fine-tune HaRT for document-level tasks on their respective training data with an input instance capped to 8 blocks of 1024 tokens each, and no capping during evaluation. We train for 15 epochs using train and dev sets - along with history where available - with 1 user train batch size, 20 users eval batch size and early stopping patience set to 6. All models converge within 5 epochs except one stance target - feminism. GPT-2$_{\text{HLC}}$ is fine-tuned with the same data - but not history - using the same settings except a different learning rate (from the hyperparameter sweep mentioned above), train and eval batch size of 60, and max tokens per message set to 200 (consistent with pre-training).

User-level Fine-tuning Settings We fine-tune HaRT for user-level tasks with an input instance capped to 4 blocks of 1024 tokens each, and evaluation data capped to 63 blocks (to allow for dev set evaluation due to compute limitations). For fine-tuning HaRT, we use 4 user train batch size and 20 eval batch size with early stopping patience set to 3. We layer norm the user-states (hidden states of the user state vector) from HaRT, and linearly transform (to embedding dimensions) before averaging the user-states to make the user's age estimation. We train for 30 epochs with warmup steps equivalent to 10 epochs, and a weight decay set to 0.01. We find that for the task of Age estimation the model converges at epoch 21, however for Personality Assessment we find a simple classification linear layer to show better performance (with a convergence seen at epoch 28 when run for 35 epochs). In case of GPT-2$_{\text{HLC}}$ we with the same data (split into into individual messages capped to 200 tokens per message as in pre-training), for 15 epochs (much higher training time as compared to HaRT) with train and eval batch size set to 400, and early stopping patience set to 3.

Model	*Age* (r)	*OPE* (r_{dis})
HaRT (Full test set)	0.868	0.619
HaRT (Filtered test set)	0.872	0.635

Table 8: HaRT's performance on user level downstream tasks: Age and Openness (OPE), on full test sets (5000 users and 1943 users respectively for Age and OPE) from Park et al. (2015) and Lynn et al. (2020), as well as on the resulting test set (4438 users and 1745 users respectively for Age and OPE) after filtering the dataset for English language with users having a minimum of 50 posts and 1000 words (as we do for our pre-training data).

MeLT – Sentiment Fine-tuning Settings To apply MeLT (Matero et al., 2021) to the sentiment task we use use optuna (Akiba et al., 2019) to search both learning rate and weight decay parameters using a search space between 6e-6 and 3e-3 and between 1 and 1e-4 respectively. We keep the same architecture as described in the original MeLT paper, however we make 1 change during fine-tuning and that is the message-vector representation from MeLT is concatenated with the average of the observed tokens for the labeled message to include both local and global context into the fine-tuning layers.

No Historical Context Fine-tuning Settings We run a hyperparameter sweep using Optuna (Akiba et al., 2019) for all models for learning rate (using search space between 5e-6 to 5e-4) and weight decay(using search space between 0.0 and 1.0) with early stopping patience set to 6. We do this for 15 and 10 trials for Stance and Sentiment models respectively, and pick the hyperparameters value for the best model in the same way as described in the Experimental Settings (A.3 section above. We use these values to fine-tune the models for 15 epochs and get the weighted F1 results.