Authorship Prediction and Personalized Text Generation using Chat Data

Shanti Sanchez Barbero (shan1006@mit.edu)
Jennifer Chen (jennchen@mit.edu)
Luca-Andrei Manea (lucman@mit.edu)
Tessie Xie (cc906@mit.edu)

15.773 Hand-on Deep Learning

Prof. Vivek Farias

Spring 2024

1 Introduction

For this project, we aim to develop a model capable of predicting and capturing the unique messaging styles of four group members based on our WhatsApp chat messages with friends. The primary objective is to accurately identify the author of each message, using neural networks and leveraging patterns and vocabulary of each person's communication style. Then, our next goal is to use a large language model (LLM) specifically to generate text messages tailoring it to our messages. The LLM model should be capable of emulating the distinctive tone and style of a selected individual from the chat messages.

2 Data

For our predictive and text-generation model, we will utilize our WhatsApp chat data. It includes a 'Name' column, which identifies the sender of each message, and another column detailing the message's content. By analyzing these components, we aim to understand patterns in communication and generate text that mirrors these prior messages, providing a comprehensive view of the different communication styles within our dataset.

2.1 Exploratory Data Analysis

For our project, we focused exclusively on predicting and replicating the messaging styles of our four team members. Consequently, we filtered out messages from friends, retaining only those sent by the four of us. During our data analysis phase, we performed an Exploratory Data Analysis (EDA) to examine the distribution of messages each team member contributed. The result (Figure 1) above reveals a significantly unbalanced dataset, with Luca contributing a disproportionately high number of messages. The imbalance is attributed to Luca's frequent use of English for communication, in contrast to Shanti, Jennifer, and Tessie, who less commonly use English for their online interactions.

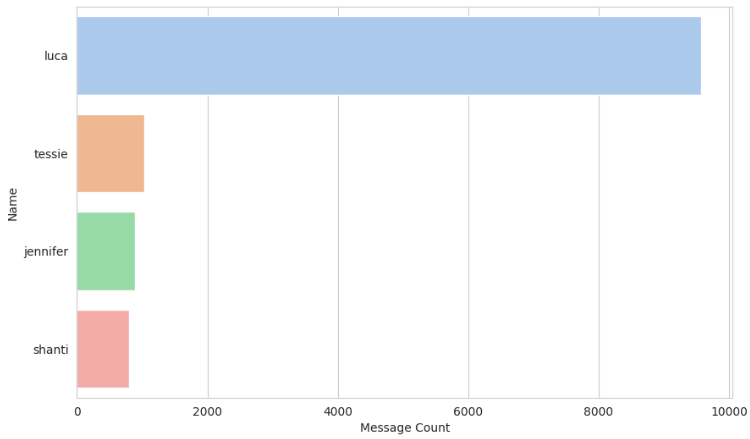

Figure 1: Message Count by Name

Additionally, we utilized word clouds (Figure 2) to gain preliminary insights into the context and messaging styles of the four team members. Common themes across our messages include meeting ("meeting"), agreement ("yeah", "yes", "sure"), and fun (some variation of "haha"). Consequently, for our second objective, we will use the LLM to generate messages based on these themes and others to see how our communications styles might differ.

There are already some insights that can be drawn from the word clouds. All of us seem to express our opinions with "think" and Jennifer and Tessie seem to do so the most, in proportion to all of their messages. The two of them also seem to talk about work a lot (”meeting”, ”email”, ”work”, ”interview”). This might suggest that their personality is more inclinded to discuss these topics. However, it is more likely that this stems from the fact that the two of them exlusively communicate with fellow MBANs on WhatsApp. Most of Shanti’s and Luca’s messages seem to be about expressing some sort of laughter using ”haha”. This can be for two reasons: Either the content of their messages is funnier or their messaging style usually includes some form of laughter to create a nice texting atmosphere.

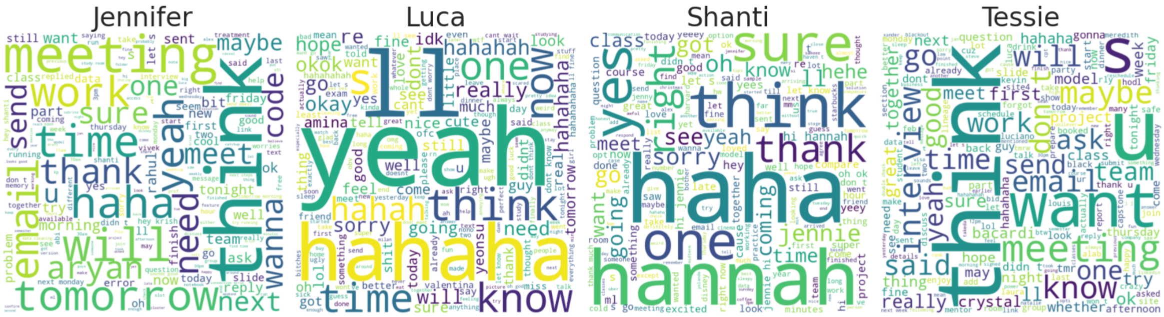

Figure 2: Message Word Cloud

2.2 Data Preprocessing

2.2.1 Unifying Labeling

The chat data, sourced from various phones and WhatsApp accounts, presented inconsistencies in naming conventions. To address this, we standardized the labels for clarity and consistency, adopting uniform identifiers: ‘luca’, ‘jennifer’, ‘tessie’, and ‘shanti’ for the names associated with each set of messages.

2.2.2 Removing non-text content

Upon reviewing the dataset, we noticed an issue with how stickers, pictures, and videos were exported, leading to an improper representation within the data files. To ensure they did not incorrectly influence our model’s understanding of our messaging style, we opted to remove these stickers. Similarly, emojis, being stored in Unicode format, posed a challenge for the model’s interpretation capabilities. To maintain data integrity and model accuracy, we also decided to exclude emojis from our dataset. Additionally, we removed links as we did not want elements of the links to be included as part of the vocabulary when tokenizing the messages which would add bias to the dataset. We, therefore, decided to remove all of this non-text content. This step was crucial for preparing a clean, representative sample of our textual communications for analysis.

2.2.3 Removing duplicate messages

Furthermore, upon examining the dataset, we noticed a high frequency of brief messages, such as ‘okok’, ‘thank you’, and ‘thanks’, recurring throughout the chat histories of individuals. To enhance the dataset’s quality and relevance for our modeling efforts, we chose to eliminate duplicate instances of these repetitive messages for each label, retaining only one instance of each short message. This approach reduces the volume of data the model processes, yet ensures that the remaining messages are more substantive and meaningful for learning and analysis. This step was particularly important for our second objective as the retrieval process used to extract several messages that were the exact same when we included duplicate messages which reduced the capabilities of the LLM to generate messages.

2.2.4 Down-sampling

Drawing from the insights gained through EDA, we decided to balance the dataset by randomly down-sampling Luca's messages. This step was taken to mitigate the model's bias towards predicting messages as Luca's due to his over-representation in the dataset. After implementing this down-sampling, the distribution of messages became more balanced: 1054 messages from Luca, 1038 from Tessie, 886 from Jennifer, and 795 from Shanti. The adjustment ensures a more balanced input for our model, fostering fairer and more accurate predictions across all team members.

2.2.5 Train Test Split

To ensure a consistent distribution of class labels between the training and testing datasets, we utilized stratified sampling. Specifically, we allocated 80% of the data for training purposes, while reserving the remaining 20% for testing.

3 Methodology

3.1 Authorship Prediction

In this section, we describe the models we use to predict who, either Jennifer, Luca, Shanti, or Tessie, wrote each chat message.

We use the Bag of Words, GloVe, BERT, and text-embedding-ada-002 embedding methods to get the transformed training data. Detailed explanations of each approach will be provided in the subsequent subsections. After transformation, the data is fed into a neural network comprising three hidden layers, each with 32 neurons and employing ReLU activation. A dropout layer with a rate of 0.2 follows each hidden layer to prevent overfitting. The output layer of the network uses softmax activation to facilitate multi-class classification. During the training process, we use 20 epochs, a batch size of 32, and a 0.2 validation split. After having experimented in the initial phases of our project, we found that this network architecture works the best.

3.1.1 Bag of Words

We begin with a base model characterized by a maximum token length of 1000, utilizing unigrams and employing a multi-hot output configuration. Then, we try to investigate the effects of three hyperparameters: maximum token length (`max_token`), n-gram range (`ngram`), and output mode (`output_mode`), on the model's performance. This exploration aims to understand how variations in these hyperparameters can influence the results of our text analysis model.

3.1.2 GloVe

In this model, the "trainable" hyperparameter is set to true, enabling the model to be fine-tuned on our specific dataset, which consists of messages. This configuration allows the model to adjust and optimize its parameters specifically for the characteristics and nuances of our data.

3.1.3 BERT

We adapt BERT to our specific needs by fine-tuning it on our dataset, which comprises messages. Given that our dataset is relatively small and BERT has been pre-trained on a diverse array of topics, we anticipate that the model's performance will improve through this fine-tuning process.

3.1.4 ADA-002

Text-embedding-ada-002 (hereinafter referred to as ada-002) is a new embedding model from OpenAI that replaces five separate models for text search, text similarity, and code search. It outperforms the previous most capable model, Davinci, at most tasks, while lowering the price of computation[1]. In our project, we train the ada-002 embeddings using our chat dataset. The model generates embeddings with a dimensionality of 1536.

3.2 Personalized Text Generation using LLMs

For this task, our goal is to use Retrival-Augmented Generation (RAG) to get relevant messages and use clever prompt engineering to employ an LLM to mimic the unique messaging styles of us based on our chat messages. Ideally, this model should then be capable to generate messages as if they were written by any of the four members. Initially inspired by a YouTube tutorial[2] on integrating a RAG approach with a pre-trained LLM, we encountered challenges due to the slow performance and high computational costs associated with storing embeddings in a Chroma database. To address these issues, we opted for an alternative approach still using RAG but different in how the data was stored. The embeddings and API calls between the two models were done with the same specifications (embedding: text-embedding-ada-002 & GPT model: gpt-3.5-turbo). This means that the results were the same, therefore, we decided to use the approach we saw in class for convenience.

To evaluate the model's effectiveness, we prompt it to generate messages about the following topics: exam, project, meeting, homework, late, okay, funny, food, eat, and happy birthday. The first five topics are about working and studying which we say as important topics during EDA. Agreeing and expressing laughter is also something that seemed to be apparent in a lot of messages so we also want include these topics. Finally, we were curious to see how the LLM would write messages of us talking about food, eating and wishing someone happy birthday. These topics are not only common topics in the chats but also common topics in our lives.

After creating embeddings for the messages, we use RAG to find the top 10 most similar and important messages for each topic and each person. This approach ensures the model has ample historical messages to learn from and allows us to directly compare its outputs with the original messages in our dataset. This method provides a clear benchmark for assessing the model's ability to accurately mimic the messaging styles of the group members under typical conversation topics. We then include the context (the relevant messages) in the prompt and then feed that prompt to the LLM to generate a message. Note that we use each prompt individually for each of the four of us. We create three different prompts. This means that in the end we generate $3 * 10 = 30$ message for each of us, so a total of 120 messages.

For the three prompts (which can be found in the appendix), we use certain variables:

- top_n: number of most similar/relevant messages to make up the context, in our case we use 10
- instruction: the topic the message should be about, these topics are the aforementioned 10 topics
- avg_context_length we saw that the LLM would write very long messages, so we want to tell it to write a message that is around the average length of the context messages
- context: the 10 messages retrieved concatenated by "\n—\n", an example for the topic "project" can be found in the appendix (Table 3)

[1] Blazing Fast Search with text-embedding-ada-002 in Node.js

[2] https://youtu.be/tcqEUSNCn8I?si=5C-4F-VwNKzbUv4-

In all three prompts, the LLM is asked to act as a person and to write a brief message about a topic given past messages around a certain length. We specifically added "act as a person" and "it is very important..." as we saw it improve results.

Prompt 1 is the initial iteration in which we ask the LLM to only use the context messages and to not add extra content to avoid the LLM inventing too many unnecessary details.

Prompt 2 and 3 are more extensive than prompt 1. They are also more similar as we saw good results with prompt 2 and tried to improve them even further with prompt 3. Prompt 2 insists on preserving the original tone, word choice, and sentence structure found in the context messages. We also included "Do not make up stuff and do not paraphrase too much!" as we saw that the outputs from prompt 1 would convey the meanings similar to what we would say but with the wrong words. Prompt 3 is an extention of that in which we additionally tell the LLM not to use synonyms and to focus on words present in the messages. The prompt also includes the request to break down the task and first identify common words in the context messages which we assume would improve the results as we have seen in class.

Given that the messages were written by us, we are able to manually validate the messages and decide if they sound like something we would write. An alternative approach would be to cross-reference the generate message with the context messages to see how similar they are. However, we did not want to automate this process since we wanted and were curious to see and investigate the generated messages to validate them by ourselves.

4 Results and Discussion

4.1 Authorship Prediction Model Performance

Table 1 summarizes the model performance. As a baseline, simply predicting every chat message in the test set to belong to the predominant category, 'Luca', provides an accuracy of 27.95%.

Among all the data transformation models, the ada-002 model stands out with the highest accuracy at 62.5%, considerably outperforming the base BoW model's 53.38% accuracy. Notably, fine-tuning the BoW model parameters - such as increasing `max_token` to 2500 and altering the `output_mode` to `'count'` (without removing duplicate messages) - yielded slight enhancements in performance, indicating that parameter tuning can influence results meaningfully. However, switching from unigrams to bigrams in the BoW model leads to a decrease in accuracy (51.66%), potentially due to the short nature of chat messages where word pairs do not provide additional contextual benefits.

The more advanced models, GloVe and BERT, demonstrated higher accuracy than BoW, highlighting the effectiveness of sophisticated embedding techniques in better understanding the patterns and nuances in the chat data.

Figure 3 and Figure 4 present the accuracy and loss curves for the ada-002 model during training. From Figure 3, we observe an increasing trend in training accuracy with each epoch, suggesting that the model is effectively learning from the data. Validation accuracy also shows an upward trend, but it plateaus after around 15 epochs, indicating that further learning may not yield significant gains on unseen data.

Table 1: Model Performance Summary

Model	Parameters	Test Accuracy
Baseline accuracy		27.95%
BoW: base	`max_token = 1000,` `ngrams = 1,` `output_mode = 'multi_hot'`	53.38%
BoW: change `max_token`	`max_token = 2500`	55.76%
BoW: change `ngrams`	`ngrams = 2`	51.66%
BoW: change `output_mode`	`output_mode = 'count'`	55.36%
GloVe	`output_dim = 300,` `trainable = True`	55.90%
BERT	`trainable = True,` `learning_rate = 5e-5`	57.75%
ADA-002		62.25%

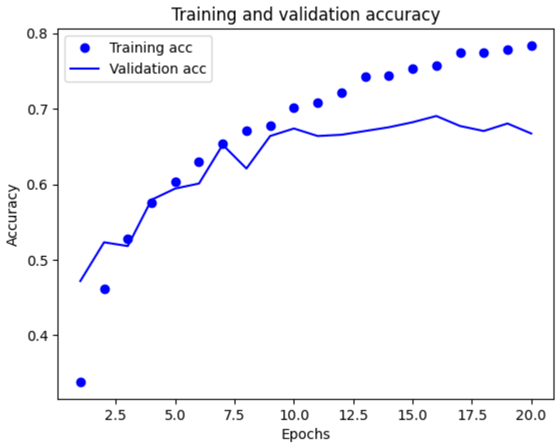

Figure 3: ADA-002 Accuracy Curves

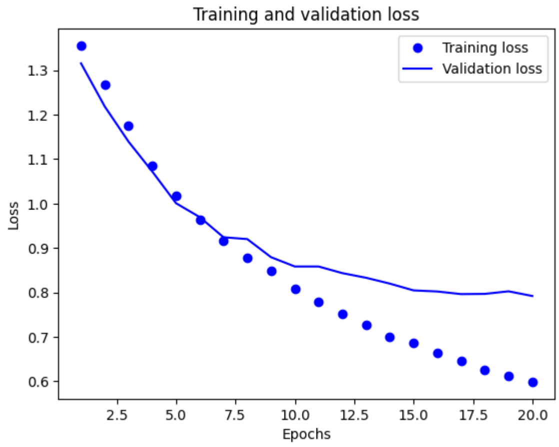

Figure 4: ADA-002 Loss Curves

From Figure 4, we observe that training loss decreases sharply, while validation loss decreases alongside but starts to become flat after around 15 epochs. These curves indicate that the model tends to converge on a solution that balances between fitting the training data and generalizing to new, unseen data.

Figure 5 shows a heatmap of the confusion matrix for ada-002 model's predictions. Each row of the confusion matrix is the actual labels and each column is the predictions. The matrix shows that the model performs best in correctly predicting Luca, with a 74% success rate, followed by Jennifer with a 64% rate, Shanti with a 57% rate, and Tessie with a 52% rate, as indicated by the darker shades along the diagonal.

The off-diagonal values present instances of misclassification. For example, Tessie is often misclassified as Jennifer, occurring 22% of the time. Additionally, there is a notable confusion between Jennifer and Shanti, with 16% of Shanti's messages being incorrectly labeled as Jennifer. Such misclassification could be attributed to the fact that Jennifer, Shanti, and Tessie use WhatsApp for more formal conversations, such as organizing meetings and discussing projects. This overlap in usage context could lead to certain keywords and phrases becoming less distinctive in differentiating between these users.

For the other models, the confusion matrices can be found in the Colab notebook. The main insights are as follows: For BERT, the confusion matrix looks similar and most messages get

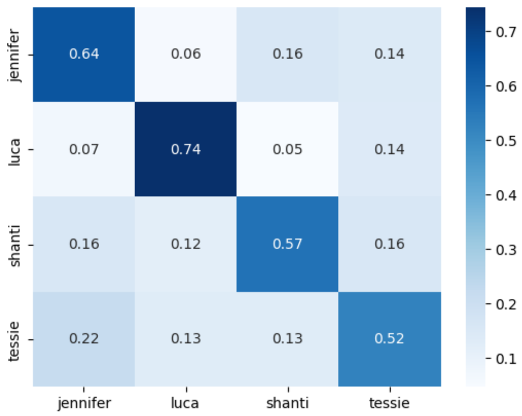

Figure 5: Confusion Matrix for ADA-002 Predictions

misclassified as Jennifer's messages. For GloVE, most messages get misclassifed as Luca's and for BoW most get misclassifed as Jennifer's and Tessie's.

It is important to underline the fact that throughout all models, and not just the best one, some of Shanti's, Jennifer's, and Tessie's messages seem to be hard to distinguish. Furthermore, for the other models, Luca's and Tessie's messages seem to be hard to distinguish which might indicate that they have similar messaging styles and that the ada-002 embeddings are able to captures that better than the other embedding methods.

4.2 Personalized Text Generation using LLMs Performance

4.2.1 Prompt 1 Analysis

Since prompt 1 contained the instruction to only use the historical messages, it did a good job overall at including the most frequent words but it added extra content that did not make sense like "Struggling with homework, exams, and thesis work at Harvard." For the topics "exam", "project" and "homework", the outputs performed relatively good. But because of the nature of the topics, the improvement was not very noticeable. For the "meeting", "okay" and "late" topics, it was able to recognize the specific wording style while keeping the original content. Lastly, for the topics "funny", "food", "eat" and "happy birthday" we observed a good performance as well.

4.2.2 Prompt 2 Analysis

This prompt gave more emphasis on not using too many words not available in the context since that is what the other prompt was struggling with. Specifically for the topics "exam", "project" and "homework" where there was not a distinctive messaging style or word used by each user, it performed overall very well. For the "meeting", "okay" and "late" topics, we noticed that it generated much more formal messages than we would adding words like "convene" or "shall we" when the context did not include them. Specifically for the "okay" topic, we all had a specific style and it had a better performance on Luca's messages since it captured his distinctive "okok". Lastly, for the topics "funny", "food", "eat" and "happy birthday" even though the messages were more formal compared to the historical messages, it managed to mimic the assortment and frequency of used words by each user. An example of the created prompts can be found in the appendix (Table 2).

4.2.3 Prompt 3 Analysis

This prompt was more detailed and an emphasis was laid on mirroring the tone and linguistic patterns without paraphrasing too much. However, for this prompt it added words from the context without making sense such as "Looks so tasty, options to Starbucks. They taste so good!". For the topics "exam", "project" and "homework", it was able to paraphrase the context in a more similar style to us and took more into account the historical messages. For the "meeting", "okay" and "late" topics, the improvement was similar to the prompt 2 results. Lastly, for the topics "funny", "food", "eat" and "happy birthday", we observed an overall improved performance for both mimicking the style and considering the full context of the historical messages.

4.2.4 Prompt Discussion

One of the main lessons from this project is the huge importance of prompt engineering since it can significantly enhance the model's ability to understand the task at hand and leads to more accurate, relevant, and contextually appropriate responses. We kept adding restrictions to the prompts trying to keep the messages short, rely on the historical messages and combine words from these messages and it was sometimes inaccurate and unpredictable. For this same reason, it was really hard to determine which prompt was yielding the best results since every prompt had its own advantages and disadvantages.

5 Conclusion and Lessons Learned

In this project, we were able to predict authorship of personalized text achieving an accuracy of 62.25% with the ada-002 embedding model and write personalized messages using gpt-3.5-turbo. The results for both parts of the project are good but there is a lot of room for improvement. Even with the best ada-002 embedding model the test accuracy for goal 1 is only around 60% and the context messages that are found in the RAG part are not the best messages from the dataset. This, however, does make sense as the embedding model is not perfect, and it is nowhere near perfect for text search.

Given that we do not have the capacity to compete with OpenAI in creating advanced embedding methods, there are other potential improvements that could be done to improve the results of this project. The results for task 1 suggest that more data as in more messages would not help in improving the performance, but having longer messages would help because there would be more nuance and less common pattern in the data. Furthermore, as the text messages are shorter in nature it would be useful to find and only use texts that mirror the texing style of a person distinctly. Messages like "sure" or "haha" are not useful just by themselves but if they appear as parts of other messages.

Prompt engineering was one of the most valuable part of this project, as just changing one word in the prompt could lead to completely different results. It would be helpful to try more different prompts to understand what elements work and which ones do not. Then, another challenging part would be to combine all those elements in a way that the LLM outputs the expected combination. We have seen ourselves that even if we combine the elements that work well of different prompts, the results might not be what was expected. This itself is time-consuming and it does not help that there is not much research on this topic to facilitate the prompt design.

Finally, instead of using RAG, it might be helpful to fine-tune the LLM with the chat messages directly as this is quite a specfic task that only depends on the extra data and not on other knowledge the LLM might have.

Appendix

Colab Link

Google Colab

Prompt 1

Act as a person and use the following {top_n} messages to write a single, short message about "{instruction}". Only use the following messages and do not add extra content. It is very important that the length of the output message is around {avg_context_length} words:

{top_n} Historical messages:
—

{context}

Prompt 2

Act as a person, including word choice and sentence structure, found in the {top_n} historical messages provided. Write a brief message on the given topic, "{instruction}", ensuring accuracy without inventing details. Prioritize mirroring the original tone and linguistic patterns without introducing additional features. Do not make up stuff and do not paraphrase too much! It is very important that the length of the output message is around {avg_context_length} words.

{top_n} Historical messages:
—

{context}

Prompt 3

Act as a person, including word choice and sentence structure, found in the {top_n} historical messages provided. Write a brief message on the given topic, "{instruction}", ensuring accuracy without inventing details. Prioritize mirroring the original tone and linguistic patterns without introducing additional features. Do not make up stuff and do not use synonyms that is very important, mainly use the words from the historical messages!! It is very important that the length of the output message is around {avg_context_length} words. First, identify the common words in the historical messages and use them, use at most one new word (not in the historical messages) in the new message!

{top_n} Historical messages:
—

{context}

Tables

Table 2: Prompt 2 outputs

Scenarios	luca	shanti	tessie	jennifer
Exam	How was the exam? Hope well.	Exam results are out, finally!	Just completed my exam review session.	Will you review the exam questions?
Project	We must meet for a project.	Excited to collaborate on the project, let's make it a success!	Shall we add our project to the sheet?	Shall we divide the tasks for the project?
Meeting	When shall we convene for the project?	Shall we meet at the usual spot later?	Shall we meet for a quick discussion tomorrow afternoon?	Let's meet to discuss the upcoming project details.
Homework	Need to focus on homework and studying for exams.	Shall we tackle homework after class tonight?	Shall we meet tomorrow to divide tasks for homework? Let's discuss.	Shall we divide the homework tasks together?
Late	a bit tardy, but here	Apologies for arriving a bit late.	I might arrive a tad late.	She'll be late, about 2 hours.
Okay	okok sounds good	Alright, got it, thanks.	Alright, we're done now.	Alright, sounds good!
Funny	I'm quite funny sometimes.	Haha, we're a great team!	You are hilarious, indeed.	I love it, haha!
Food	The food here is truly delightful.	Food options look so tasty.	Just ordered delicious dinner, yum!	Sounds good, let's eat here.
Eat	I finally ate, feeling much better.	The food looks so tasty, enjoy it!	We are dining together now.	Shall we eat before the meeting?
Happy Birthday	Happy birthday, wishing you all the best for this year!	happy birthday!	Hope you enjoy tonight and happy birthday!	happy birthday!

Table 3: Top 10 Relevant Historical Messages for Project Scenarios

Scenarios	luca	shanti	tessie	jennifer
Project	basically we have to meet for a projecta	yes ! specially the project solutions	should we put our project on the sheet	we can split the tasks
	and 200 slide powerpoint group project due on tuesday	excited to begin our project together!	we can talk about the opti project	and before next meeting
	final project - b 6 final project - b	excited for collaboration: let's make our project a success	that's the project i talked about in the interview	# next steps
	really really really	of course just finishing something for the projects	that the statement of work	i think we can start writing for the second project
	because right now i need to work on a project...	to talk of things that are not the project yes	hope you get the top projects that you want	what tasks do you want to do
	with the thesis	cause we haven't practice at all the project solutions	do you know about other's project?	do you wanna meet again to work on the deliverables
	and then festival	we hope this email finds you well. we wanted to express our excitement about the opportunity to work with you and on the upcoming project.	and he said we can send the slides to our mentor	great work team!!!
	and this one	i'm super flexible with the project	should both of us do the coding	we can also work on the input and output
	presentation on monday	we hope this email finds you well. we wanted to express our excitement about working with you on the upcoming project.	that's the only project that i have opti in it	can we do 9
	you can work on the train	or eager to get started	cant wait to know our capstone project	and our plots for q2 are a bit different, but i don't think it's a big problem!