# When does text prediction benefit from additional context? An exploration of contextual signals for chat and email messages

**Stojan Trajanovski** Microsoft
**Chad Atalla** Microsoft
**Kunho Kim** Microsoft
**Vipul Agarwal** Microsoft
**Milad Shokouhi** Microsoft
**Chris Quirk** Microsoft

{sttrajan,chatalla,kuki,vipulag,milads,chrisq}@microsoft.com

## Abstract

Email and chat communication tools are increasingly important for completing daily tasks. Accurate real-time phrase completion can save time and bolster productivity. Modern text prediction algorithms are based on large language models which typically rely on the prior words in a message to predict a completion. We examine how additional contextual signals (from previous messages, time, and subject) affect the performance of a commercial text prediction model. We compare contextual text prediction in chat and email messages from two of the largest commercial platforms Microsoft Teams and Outlook, finding that contextual signals contribute to performance differently between these scenarios. On emails, time context is most beneficial with small relative gains of 2% over baseline. Whereas, in chat scenarios, using a tailored set of previous messages as context yields relative improvements over the baseline between 9.3% and 18.6% across various critical service-oriented text prediction metrics.

## 1 Introduction

Email and chat communication tools are increasingly important for completing daily professional and personal tasks. Given the recent pandemic and shift to remote work, this usage has surged. The number of daily active users in Microsoft Teams, the largest business communication and chat platform, has increased from 20 million (2019, pre-pandemic) to more than 115 million in October (2020). On the other hand, email continues to be the crucial driver for formal communication showing ever increasing usage. Providing real-time suggestions for word or phrase auto-completions is known as text prediction. The efficiency of these communications is enhanced by suggesting highly accurate text predictions with low latency. Text prediction services have been deployed across popular communication tools and platforms such as (Microsoft Text Predictions, 2020) or GMail Smart Compose (Chen et al., 2019).

Modern text prediction algorithms are based on large language models and generally rely on the prefix of a message (characters typed until cursor position) to create predictions. We study to what extent *additional contextual signals* improve text predictions in chat and email messages in two of the largest commercial communication platforms: Microsoft Teams and Outlook. Our contributions are summarized as:

- We demonstrate that prior-message context provides the greatest lift in the Teams (chat) scenario. A 5 minute window of prior messages from both senders works the best, with relative gains from 9.3% up to 18.6% across key metrics (total match and estimated characters accepted). This 5 minute window of prior messages from both senders outperforms the corresponding 2 and 10 minute scenarios.

- We find that context about message composition time provides the largest gains for the Outlook (email) scenario, while adding the subject as context only marginally helps. These relative gains are moderate (2-3% across various metrics).

- We conclude that the different characteristics of chat and email messages impede domain transfer. The best contextual text prediction models are custom trained for each scenario, using the most impactful subset of contextual signals.

The remainder of the paper is organized as follows. We give an overview of state-of-the-art related research in Section 2. More details on the signals used for contextualization are provided in Section 3. Section 4 provides information on the language model, performance metrics, and statistical details



*Proceedings of NAACL HLT 2021: IndustryTrack Papers*, pages 1–9
June 6–11, 2021. 

about the data. Experiment results and comparisons are presented in Section 5. We conclude in Section 6. Ethical considerations on the data and processes are discussed in Section 7.

## 2 Related work

Text prediction services have been applied for various applications, including text editor (Darragh et al., 1990), query autocompletion on search engine (Bast and Weber, 2006; Bar-Yossef and Kraus, 2011), mobile virtual keyboard (Hard et al., 2018). Recently prediction service is applied on communication tools for composing email and chat messages to improve user writing productivity (Kannan et al., 2016; Deb et al., 2019; Chen et al., 2019; Microsoft Text Predictions, 2020).

To predict correct text continuation, such applications leverage efficient lookups with pregenerated candidates, using most popular candidates (MPC) (Bar-Yossef and Kraus, 2011), or using large-scale language models (Bengio et al., 2003). State-of-the-art language models (Jozefowicz et al., 2016; Mnih and Hinton, 2009; Melis et al., 2018) rely on the most recent deep learning architectures, including large LSTMs (Hochreiter and Schmidhuber, 1997) or transformers (Vaswani et al., 2017), while prior approaches involve n-gram modeling (Kneser and Ney, 1995; James, 2000; Bickel et al., 2005).

In this work, we focus on the application of text prediction on production-level online communication tools, to help users compose emails (Chen et al., 2019; Microsoft Text Predictions, 2020), and in addition chat messages. In particular, we focus on examining useful contextual signals to give more accurate predicted text, using time, subject, and prior messages. Various contextualization techniques (e.g., hierarchical RNNs) have been applied to add useful additional signals such as preceding web interaction, linking pages, similar search queries or visitor interests of a page (White et al., 2009); previous sequence of utterances (Park et al., 2018; Zhang et al., 2018; Yoo et al., 2020) or related text snippets (Ke et al., 2018).

## 3 Contextualization concepts

We examine several signals accompanying the main message text: **message compose time**, **subject**, and **previous messages**. We combine these signals with the message body into a single "contextualized" string, using *special tokens* to separate signals, as shown in Figure 1a. This approach is inspired by (Chen et al., 2019), as they showed that concatenating contextual signals into a single input string gave a comparable result to more complex methods encoding these signals separately[1]. The remainder of this section explains details about each contextual signal we use.

**Time** Composition time is a contextual signal which can provide added value for text prediction, enabling suggestions with relevant date-time words, like "weekend", "tonight". We encode local date and time, as shown in Figure 1a, and use <BOT> and <EOT> to separate from other signals.

**Subject** Message subjects often contain the purpose or summarized information of a message. In the email scenario, we use *subject* as context. In the chat scenario, subject is not available, so we use the *chat window name* as a proxy for subject (can be auto-generated or manually set by users). In both cases, the subject context is wrapped with <BOU> and <EOU> special tokens.

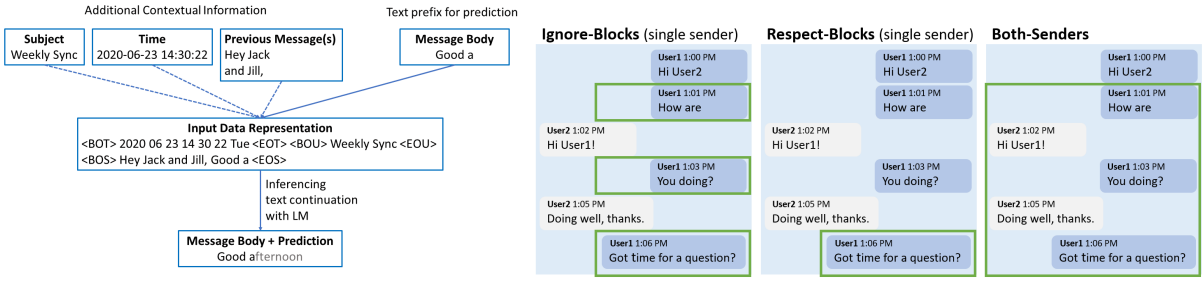
**Previous email messages** Previous messages can provide valuable background information which influences the text of the current message being composed. In the email case, we create pairs of messages and replies. These pairs are concatenated with a <COT> special token to create a single contextual string. In cases where the email was the first in a thread, the prior email context is left blank.

**Previous chat messages** Prior message contextualization for chat scenario is much more complex. Chat conversations typically consist of many small messages sent in quick succession. Given the email and chat message length statistics in Section 4, we expect chat messages to be about 10× smaller than emails. So, we limit chat histories to 20 messages, which is roughly equivalent to an email-reply pair in length. Among these prior messages, any number and any order could be from the current sender, or the other participant.

We segment chat histories by *message blocks* and *time windows*. A series of uninterrupted messages sent by one sender is considered as a single *message block*. Messages sent within the past $N$ minutes are within a *time window*, which enforces recency as a proxy for relevance.

We define three prior message context aggregation modes in the chat scenario (visualized in

[1]They also use subject and previous email as contexts.




(a) Context extraction and encoding.

(b) Aggregating a 5 min prior chat window in various context modes.

Figure 1: Examples of (a) context encoding pipeline and (b) chat prior message aggregation modes.

Figure 1b), mimicking prior email context:

(i) *Ignore-Blocks*: chat messages from the *current sender*, in the past $N$ minutes, ignoring any message block boundaries.

(ii) *Respect-Blocks*: chat messages from the *current sender*, in the past $N$ minutes, confined to the *most recent message block*.

(iii) *Both-Senders*: chat messages from *both senders*, in the past $N$ minutes. When the sender turn changes, strings are separated by a space or a special token <COT>.

For each mode, we consider time windows of $N = \{2, 5, 10\}$ minutes.

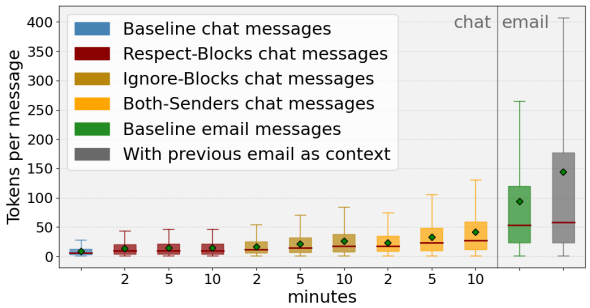



Figure 2: Box-plot statistics: number of tokens in a context-aggregated message from Microsoft Teams and Outlook. Green diamond markers represent the mean, bold red lines are the medians, margins of the boxes are lower and upper quartiles while whiskers end-points are the minimums and maximums.

## 4 Data and Language Model

### 4.1 Data

Our model training depends on real messages from two of the largest commercial communication platforms Microsoft Teams and Outlook; this involves a multi-pronged system for ensuring our customers' privacy. We work within rigorous privacy rules, using tools with privacy features built in, and preprocessing all data through multiple privacy precautions before it is digested by our models. User data from our communication platforms is never visible to humans for analysis, in any raw or preprocessed format. We run this data through our pipelines and are only able to view resulting text prediction metrics. Section 7 contains more details about these privacy precautions.

**Chat messages** We sample Teams data from more than 3.8 billion curated one-on-one chat messages that span 6 months (say May - October 2020), followed by privacy precautions and noise filters. The data is sorted by time and split into train, validation, and test sets in non-overlapping time periods. We use over 90% of the data for training, holding out 75,000 samples for validation and 25,000 samples for testing. Each message is recorded in its respective dataset along with all associated context. In a statistical analysis of the chat message lengths (see Figure 2, blue box) we find that mean tokens number is 9.15 (length in characters is 48), while median tokens number is 6 (with character length 31). Therefore, when iterating character-by-character through the messages, as done in inference for text predictions, the test set has over 1M evaluation points (resampled periodically, see Section 7.1).

**Email messages** In email experiments, we use approximately 150 million Outlook commercial emails from a period of 6 months, which go through the same privacy precautions mentioned above and in Section 7. The emails are then sorted, filtered for noise, and cut into train, validation, and test sets by their date ranges. A statistical analysis of email lengths (see Figure 2, green box) reveals that mean number of tokens is 94 (with length in char-

acters being 561), while the median is 53 tokens (and 316 characters). This is roughly 10× larger than chat messages. When splitting train, test, and validation sets, over 90% of the data is allocated to the training set. The test set is subsampled to 3,000 emails (unlike the 25,000 messages for the chat test set) since this roughly leads to final contextualized datasets of the same size. Each resulting test set contains just over 1 million evaluation points, as in the chat setting.

Additionally, we use the Avocado dataset as a publicly available dataset, which consists of emails from 279 accounts of a defunct IT company referred to as "Avocado" see details in (Oard et al., 2015), for debugging and validation, allowing us to directly view data and outputs. This dataset is split into validation and test sets, each with roughly 3,000 emails for evaluation.

### 4.2 Prior-message aggregation statistics

When applying the chat-specific prior-message grouping modes defined in Section 3, the number of prior messages fetched as context varies. Table 1 presents details on how many messages the different aggregation modes end up grouping. Both single-sender modes introduce smaller volumes of context than the *Both-Senders* mode. For example, the amount of prior messages grouped in the 5 minutes *Ignore-Blocks* mode is similar to the 2 minutes *Both-Senders* mode; where 2.5 chat messages are combined on average, and 56-59% of chat messages have at least one message as context. For emails, only around 50% have prior email context.

The number of tokens per contextualized message (including current and aggregated prior messages) varies between the email scenario and various aggregation modes in the chat scenario. Figure 2 provides statistics on these aggregated message lengths. In the chat case, the *Both-Senders* mode with a 10 minute time window results in the largest aggregate length, with a median around 27 tokens, and mean above 40 tokens. The *Respect-Blocks* mode does not show significant length increases as the time window grows, due to the message block boundary limits. For emails, the median total tokens remains similar regardless of including the previous message. This is because half of emails are not part of an email-reply pair.

### 4.3 Language model

Once the message data is preprocessed and jointly encoded with contextual signals, it is passed as an input to the Language Model. The production system uses a two-layer (550, 550) LSTM (with 6000 sampled softmax size loss) which is optimized to maximize the Estimated Characters Accepted metric (described in Section 5.1). All contextualization experiments use the production model architecture as the baseline. Both baseline and contextual models are trained on 16 GPUs.

We have conducted experiments with more complex language models (e.g., transformers, deeper LSTMs), but we use the production model in this paper as (i) its simpler architecture enables large-scale low-latency text prediction serving and (ii) the goal of this work is to explore how different contextual signals add to the baseline performance.

| Configuration | % msgs with context | mean msgs as context |
|---|---|---|
| 2 min Respect-Blocks | 30.76% | 1.44 |
| 5 min Respect-Blocks | 34.19% | 1.54 |
| 10 min Respect-Blocks | 35.54% | 1.59 |
| 2 min Ignore-Blocks | 43.31% | 1.76 |
| 5 min Ignore-Blocks | 55.94% | 2.51 |
| 10 min Ignore-Blocks | 63.24% | 3.23 |
| 2 min Both-Senders | 58.90% | 2.51 |
| 5 min Both-Senders | 70.20% | 3.99 |
| 10 min Both-Senders | 76.10% | 5.40 |

Table 1: Microsoft Teams chat message statistics - amount of aggregated context per message.

## 5 Experiments and results

We conduct experiments for both email and chat messages with individual contextual signals (time, subject, prior messages) and combinations of those.

### 5.1 Performance Metrics

In all experiments, we level the **S**uggestion **R**ate (SR) (number of suggestions per message), then evaluate model variant performance against the following text prediction metrics:

- **MR**: **M**atch **R**ate is the ratio of the number of matched suggestions and the total number of generated suggestions.
- **ChM / sugg**: **Ch**aracters **M**atched per **sugg**estion is the average number of matched characters per given suggestion
- **Est. ChS / sugg**: **Est**imated **Ch**aracters **S**aved per **sugg**estion is the estimated number of characters that the user is saved from typing, per suggestion. (Based on observed acceptance probabilities from real users.)

| Configuration / context mode | MR | ChM / sugg | Est. ChS / sugg | TM | ChM | Est. ChA |
|---|---|---|---|---|---|---|
| Chat name | +5.38%↑ | +6.05%↑ | +7.83%↑ | +5.22%↑ | +5.99%↑ | +7.86%↑ |
| Time | -3.49%↓ | -4.28%↓ | -6.33%↓ | -3.48%↓ | -4.25%↓ | -6.36%↓ |
| Time+Chat name | -13.98%↓ | -14.72%↓ | -16.57%↓ | -13.96%↓ | -14.67%↓ | -16.57%↓ |
| 2 min Respect-Blocks | +5.91%↑ | +7.65%↑ | +12.65%↑ | +5.95%↑ | +7.72%↑ | +12.62%↑ |
| 2 min Ignore-Blocks | +5.91%↑ | +7.68%↑ | +12.95%↑ | +5.78%↑ | +7.62%↑ | +12.84%↑ |
| 2 min Both-Senders | +5.91%↑ | +8.77%↑ | +16.87%↑ | +6.01%↑ | +8.77%↑ | +17.01%↑ |
| 5 min Respect-Blocks | +5.65%↑ | +7.79%↑ | +13.86%↑ | +5.56%↑ | +7.74%↑ | +13.72%↑ |
| 5 min Ignore-Blocks | +6.72%↑ | +9.01%↑ | +15.66%↑ | +6.59%↑ | +8.96%↑ | +15.48%↑ |
| **5 min Both-Senders** | **+9.41%↑** | **+11.76%↑** | **+18.67%↑** | **+9.30%↑** | **+11.72%↑** | **+18.67%↑** |
| 10 min Respect-Blocks | +5.65%↑ | +7.79%↑ | +13.55%↑ | +5.63%↑ | +7.67%↑ | +13.53%↑ |
| 10 min Ignore-Blocks | +6.99%↑ | +9.32%↑ | +15.66%↑ | +6.92%↑ | +9.24%↑ | +15.56%↑ |
| 10 min Both-Senders | +8.06%↑ | +10.57%↑ | +17.77%↑ | +7.86%↑ | +10.51%↑ | +17.55%↑ |
| Time+ 5 min Respect-Blocks | +3.76%↑ | +5.51%↑ | +10.24%↑ | +3.84%↑ | +5.51%↑ | +10.22%↑ |
| Chat name+5 min Respect-Blocks | +5.11%↑ | +6.36%↑ | +9.34%↑ | +5.13%↑ | +6.35%↑ | +9.40%↑ |
| Time+Chat name+5 min Respect-Blocks | +5.38%↑ | +7.79%↑ | +14.16%↑ | +5.43%↑ | +7.82%↑ | +14.26%↑ |
| Time+Chat name+5 min Ignore-Blocks | +5.11%↑ | +6.97%↑ | +12.05%↑ | +5.15%↑ | +6.99%↑ | +12.00%↑ |
| Time+Chat name+5 min Both-Senders | +8.87%↑ | +11.53%↑ | +18.37%↑ | +8.91%↑ | +11.52%↑ | +18.36%↑ |

Table 2: Microsoft Teams chat messages experiment results with various contextualization modes. First column is the experiment configuration, other columns are **relative gains**, over the noncontextual baseline, of the performance metrics (Section 5.1) with a leveled suggestion rate of 0.5.

- **TM**: **T**otal **M**atches is the number of suggestions which match the upcoming text.
- **ChM**: **Ch**aracters **M**atched is the number of matched characters from all suggestions.
- **Est. ChA**: **Est**imated **Ch**aracters **A**ccepted is the estimated[2] total number of suggested characters accepted by users.

### 5.2 Experiments with chat messages

The performance results for chat messages from Microsoft Teams compared to the non-contextual baseline model are shown in Table 2. For comparability, we train the model's confidence threshold to level each model's suggestion rate (SR) at 0.5 suggestions / message.

Contextualization with just the chat window name (subject) yields moderate gains, possibly because the typically short chat messages are so sparse on context that a chat topic name, or participant names from a chat header, provides a starting foothold for relevance. In contrast, from the last table rows, we see that the benefits from subject context diminish once prior messages are used as a context, suggesting that the subject proxy is much weaker than prior message context. Table 2 also shows that compose-time can act as a confounding context signal for chat messages, especially in experiments with no prior messages as a context. This is possibly due to the numerically-heavy time encoding confusing the model in contrast to the short text of chat messages. The experiments also show that the benefits of these contextual signals are not additive.

All three prior message aggregation modes (*Ignore-Blocks*, *Respect-Blocks*, and *Both-Senders*) show gains across all performance metrics, with all time window sizes. *Both-Senders* mode achieves the most significant relative gains: above 9.3% for Match Rate and the Total Matches; more than 11.7% for the character match and character match per suggestion; and more than 18.6% for the characters saved per suggestion and character acceptance. This indicates that messages from the other sender provide significant value, when used with a well-tuned time window. It provides relevant conversation context from all senders, eliminating confusing gaps between messages, and enables suggestions in response to questions posed by the other sender. In particular, the *Ignore-Blocks* mode does worse than *Both-Senders*, since *Ignore-Blocks* can violate conversation continuity, including messages $[k, k+2]$ from the current sender, and skipping message $k+1$ from the other sender.

For the single-sender modes, *Respect-Blocks* generally performs slightly worse as it utilizes only part of the messages taken by the *Ignore-Blocks* mode. This indicates that seeing a longer prefix of the current message block (more similar to writing a long email) makes an impact on text prediction in chat messages. Lastly, we observe that a 5 minute time window works better than 2 and 10 minute time windows. Shorter time windows seem to miss important prior context while a larger windows lead

[2]Based on observed acceptance probabilities on large-scale production traffic, users tend to accept longer suggestions.

| Configuration / context mode | MR | ChM / sugg | Est. ChS / sugg | TM | ChM | Est. ChA |
|---|---|---|---|---|---|---|
| Subject | -0.81%↓ | -0.36%↓ | +0.76%↑ | -0.74%↓ | -0.35%↓ | +0.76%↑ |
| **Time** | **+2.02%↑** | **+2.25%↑** | **+2.88%↑** | **+2.01%↑** | **+2.22%↑** | **+2.83%↑** |
| Previous Email | -9.72%↓ | -10.56%↓ | -13.05%↓ | -9.65%↓ | -10.56%↓ | -13.12%↓ |
| Time+Subject | +0.20%↑ | +0.47%↑ | +1.06%↑ | +0.23%↑ | +0.49%↑ | +1.11%↑ |

Table 3: Microsoft Outlook email messages experiment results with various contextualization modes. First column is experiment configuration, other columns are **relative gains**, over the noncontextual baseline, of the performance metrics (Section 5.1) with a leveled suggestion rate of 3.8.

| Configuration / context mode | MR | ChM / sugg | Est. ChS / sugg | TM | ChM | Est. ChA |
|---|---|---|---|---|---|---|
| Subject | -1.46%↓ | -0.21%↓ | +1.77%↑ | -1.58%↓ | -0.22%↓ | +1.80%↑ |
| **Time** | +0.24%↑ | +1.59%↑ | **+4.87%↑** | +0.20%↑ | +1.55%↑ | **+4.75%↑** |
| Previous Email | -3.89%↓ | -3.50%↓ | -2.43%↓ | -3.85%↓ | -3.42%↓ | -2.43%↓ |
| **Time+Subject** | **+1.70%↑** | **+2.32%↑** | +3.32%↑ | **+1.75%↑** | **+2.34%↑** | +3.41%↑ |

Table 4: Avocado test set (Oard et al., 2015) messages experiment results for various contextualization modes. First column is experiment configuration, other columns are **relative gains**, over the noncontextual baseline, of performance metrics (Section 5.1) with a leveled suggestion rate of 2.5.

to over-saturation of irrelevant information.

### 5.3 Experiments with email messages

The gains from the contextualization in email messages are more moderate compared to those from chat messages. The comparison of the contextualized models with the baseline on commercial Microsoft Outlook emails and Avocado dataset are given in Table 3 and 4 respectively. For emails, the results suggest that time as a context (or time+subject in the Avocado dataset) offers most promising relative gains of 2-3%. This contrasts the observed trend from chat messages. Time is more important for emails since emails are often longer, contain greetings, farewells, and meeting requests with time-related keywords (e.g., "tomorrow", "last night", "after the weekend"). Additionally, numerical tokens from the time context are less likely to outnumber the message content tokens, since emails are about 10×longer than chat messages.

With the chosen architecture, neither subject nor prior message context signals provide value in the email scenario. Subjects may introduce keywords, but the implemented method of encoding context and body into a single string did not demonstrate an ability to pull out those key words for suggestions. Likewise, prior message context did not benefit the email scenario. As Figure 2 shows, emails with prior messages are significantly longer than any of the chat context aggregations. Prior emails may have critical information steering the direction of an email thread, but our production-oriented metric are not significantly affected. The implemented architecture may not be strong enough to isolate and make use of those cues, instead becoming confounded by the vast influx of tokens from another sender. This emphasizes that the email and chat scenarios require different context signals, and may benefit from different underlying architectures.

**Qualitative analysis with the Avocado set** Given our commercial data-visibility constraints due to the privacy considerations, we perform a qualitative analysis on the public Avocado dataset (Oard et al., 2015). Using this public data, we evaluate text predictions from one of the promising email context modes: time context. As shown in Table 5, we use diff tools to identify when the time context model and baseline model create (i) correct suggestions, (ii) wrong suggestions, and (iii) no suggestions. We see that the time context model improves on all three columns. When directly examining cases where the time-context model renders a new correct suggestion, compared to the baseline, we observe a trend of time-related n-grams. Words like "tomorrow", "available", "September" are seen more frequently in correct suggestions (see Figure 3). The same trend is also observed in the Time+Subject model.

| | Time as context / Baseline in Avocado test set | | |
|---|---|---|---|
| cases | correct / wrong | correct / no sugg | no sugg / wrong |
| context win | **256** | **1494** | **2825** |
| context loss | 239 | 1400 | 2553 |

Table 5: Comparing text predictions of time-context model vs baselines. "Context win" row holds counts of cases where contextual model suggestions beat baseline suggestions.

## 6 Conclusions

We study the role of context in text prediction for chat and email platforms. Testing with previous messages, subject, time as additional contextual

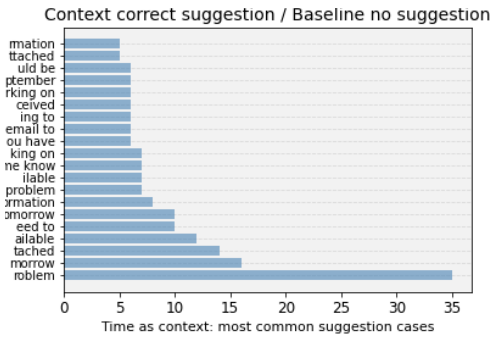



Figure 3: The 20 most common new suggestions triggered by the time-context model, on data points from the Avocado test set (Oard et al., 2015) where the baseline renders zero suggestions.

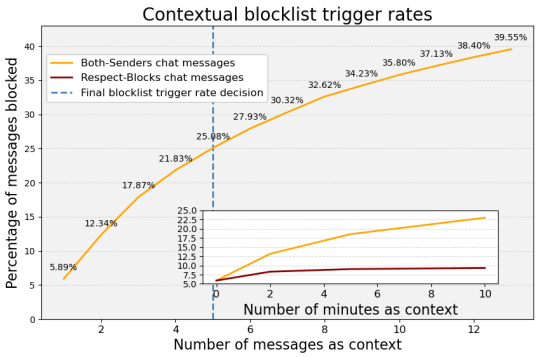



Figure 4: Initial blocklist trigger rates for various contextualization merging modes in Microsoft Teams chat messages.

signals, we find that the different characteristics of emails and chat messages influence the selection of contextual signals to use. Previous message contextualization leads to significant gains for chat messages from Microsoft Teams, when using an appropriate message aggregation strategy. By using a 5 minute time window and messages from both senders, we see a 9.4% relative increase in the match rate, and an 18.6% relative gain on estimated characters accepted. Chat messages are often short and can lack context about a train of thought; previous messages can bring necessary semantics to the model to provide a correct prediction. Benefits are comparatively insignificant for subject and compose time as contextual signals in chat messages.

In the email scenario based on Microsoft Outlook, we find that time as a contextual signal yields the largest boost with a 2.02% relative increase on the match rate, while subject only helps in conjunction with time, and prior messages yields no improvement. More complex models may be needed to reap subject and prior message gains for emails, but the current architecture was chosen for large-scale serving latency.

Future work involves exploring different encodings for contextual signals, such as utilizing hierarchical RNNs (Park et al., 2018; Yoo et al., 2020) to better capture context, or using more advanced architectures such as transformers or GPT-3.

## 7 Ethical Considerations

When working with sensitive data and running a service which generates text predictions for the general public, we are responsible for preserving user privacy and serving fair and inclusive suggestions.

### 7.1 Privacy considerations on user data

Our service framework follows the regulatory requirements of internal company-wise standards and General Data Protection Regulation (GDPR) (2018) to meet the user privacy regulations and customer premises. All customer chat and email data, from Teams and Outlook, used in this work are classified as customer content, which is not visible to humans for any purpose. Only system byproduct data, which is not linkable to specific users or groups, is obtained and viewed for quantitative evaluation. This includes internal service logs or numerical metrics (shown in Section 5.1). We also regularly re-sample training and test data due to our privacy and data retention policies, preserving similar data set sizes. We strictly use only publicly available data, such as the Avocado dataset (Oard et al., 2015), for debugging and visible qualitative evaluation.

### 7.2 Blocklisting

In pursuit of fair, respectful, and responsible suggestions, we employ a blocklist. This blocklist step in our text prediction system consists of a large dictionary containing denigrative, offensive, controversial, sensitive, and stereotype-prone words and phrases. Text from the message body and contextual signals serves as input to the blocklist. Then, if any word or phrase from the blocklist is found in the input, all further suggestions are suppressed for the message.

In the email scenario, the full body and context is used for blocklist checks, resulting in a blocklist trigger rate of 47.42%. This means that 47.42% of our data points contain a blocklisted term in their input text, and we avoid triggering suggestions on those points. Naturally, this rate of blocklist

triggering increases as more context is added to the pool of text being checked.

This phenomenon introduces an added complexity to the chat scenario. A noncontextual baseline chat model would fail to trigger the blocklist on a response to an offensive statement from two messages ago. Figure 4 shows how the blocklist trigger rate varies as larger windows of chat history are used as context. We ensure that all chat models check the past 5 messages against the blocklist, no matter how many prior messages are used for text prediction inference. With 5 prior messages fed to the blocklist in chat conversations, the blocklist trigger rate is 25.08%, instead of 5.89% with no added prior messages.

## Acknowledgements

We would like to thank the members of Microsoft Search, Assistant and Intelligence (MSAI) group for their useful comments and suggestions.

## References

Ziv Bar-Yossef and Naama Kraus. 2011. Context-sensitive query auto-completion. In *Proc. of the 20th Intl. Conf. on World Wide Web (WWW)*, pages 107–116.

Holger Bast and Ingmar Weber. 2006. Type less, find more: fast autocompletion search with a succinct index. In *Proc. of the 29th Annual Intl. ACM Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 364–371.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Jour. of Machine Learning Research*, 3:1137–1155.

Steffen Bickel, Peter Haider, and Tobias Scheffer. 2005. Learning to complete sentences. In *European Conf. on Machine Learning (ECML)*, pages 497–504. Springer.

Mia Xu Chen, Benjamin N Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M Dai, Zhifeng Chen, et al. 2019. Gmail Smart Compose: Real-time Assisted Writing. In *Proc. of the 25th ACM SIGKDD Intl. Conf. on Knowledge Discovery & Data Mining*, pages 2287–2295.

John J. Darragh, Ian H. Witten, and Mark L. James. 1990. The reactive keyboard: A predictive typing aid. *Computer*, 23(11):41–49.

Budhaditya Deb, Peter Bailey, and Milad Shokouhi. 2019. Diversifying reply suggestions using a matching-conditional variational autoencoder. In *Proc. of Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 40–47. Association for Computational Linguistics.

European Commission. 2018. EU data protection rules. https://ec.europa.eu/info/law/law-topic/data-protection/eu-data-protection-rules_en. Online; accessed 6 January 2021.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv:1811.03604*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Frankie James. 2000. Modified kneser-ney smoothing of n-gram models. Technical report, RIACS.

Jared Spataro. 2019. 5 attributes of successful teams. https://www.microsoft.com/en-us/microsoft-365/blog/2019/11/19/5-attributes-successful-teams/. Online; accessed 6 January 2021.

Jared Spataro. 2020. Microsoft Teams reaches 115 million DAU—plus, a new daily collaboration minutes metric for Microsoft 365. https://www.microsoft.com/en-us/microsoft-365/blog/2020/10/28/microsoft-teams-reaches-115-million-dau-plus-a-new-daily-collaboration-minutes-metric-for-microsoft-365/. Online; accessed 6 January 2021.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv:1602.02410*.

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufman, Balint Miklos, Greg Corrado, Andrew Tomkins, Laszlo Lukacs, Marina Ganea, Peter Young, and Vivek Ramavajjala. 2016. Smart reply: Automated response suggestion for email. In *Proc. of the ACM SIGKDD Conf. on Knowledge Discovery and Data Mining (KDD)*, page 955–964.

Nan Rosemary Ke, Konrad Żołna, Alessandro Sordoni, Zhouhan Lin, Adam Trischler, Yoshua Bengio, Joelle Pineau, Laurent Charlin, and Christopher Pal. 2018. Focused hierarchical RNNs for conditional sequence processing. In *Proc. of the 35th Intl. Conf. on Machine Learning (ICML)*, volume 80, pages 2554–2563, Stockholm, Sweden.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 181–184.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *6th Intl. Conf. on Learning Representations (ICLR)*, Vancouver, BC, Canada.

Microsoft Text Predictions. 2020. Write faster using text predictions in Word, Outlook. https://insider.office.com/en-us/blog/text-predictions-in-word-outlook. Online; accessed 7 April 2021.

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1081–1088.

Douglas Oard, William Webber, David A. Kirsch, and Sergey Golitsynskiy. 2015. Avocado research email collection LDC2015T03. Philadelphia: Linguistic Data Consortium.

Yookoon Park, Jaemin Cho, and Gunhee Kim. 2018. A hierarchical latent structure for variational conversation modeling. In *Proc. of the Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1792–1801, New Orleans, LA, USA. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008.

Ryen W. White, P. Bailey, and Liwei Chen. 2009. Predicting user interests from contextual information. In *Proc. of the 32nd Intl. ACM Conf. on Research and development in information retrieval (SIGIR)*.

Kang Min Yoo, Hanbit Lee, Franck Dernoncourt, Trung Bui, W. Chang, and Sang-goo Lee. 2020. Variational hierarchical dialog autoencoder for dialogue state tracking data augmentation. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP)*.

Zhuosheng Zhang, Jiangtong Li, Pengfei Zhu, Hai Zhao, and Gongshen Liu. 2018. Modeling multi-turn conversation with deep utterance aggregation. In *Proc. of the 27th Intl. Conf. on Computational Linguistics (COLING)*, pages 3740–3752, Santa Fe, New Mexico, USA. Association for Computational Linguistics.