DeepL Hackathon

Priority NOTIFY

Merthan Erdem, Erik Putzier

Table of CONTENTS

Mission STATEMENT

Our goal is to develop an innovative app that significantly enhances business productivity through intelligent message prioritization using specially trained LLMs. The app will send notifications based on the importance of each message, ensuring that crucial information never gets overlooked. Furthermore, we aim to break language barriers by offering seamless communication in multiple languages. With our app, businesses can streamline their workflow and foster efficient collaboration across diverse teams.

THE PROBLEM

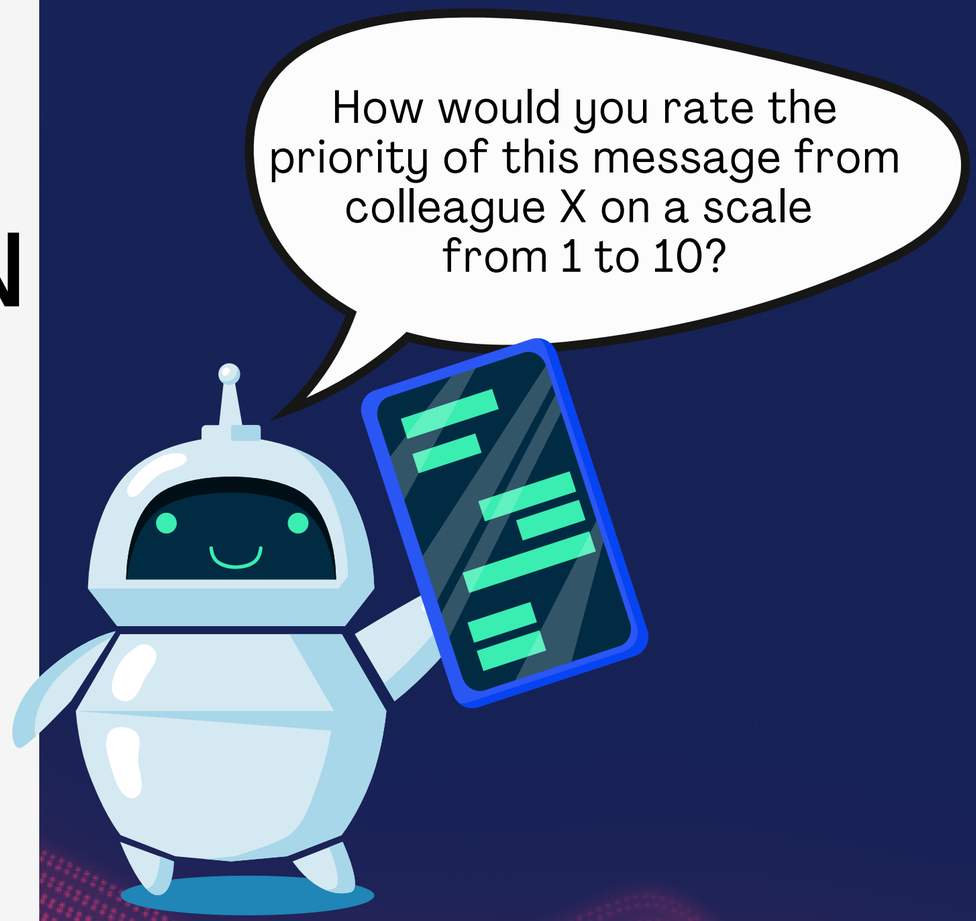
How can we determine which messages are actually important?

Communication can be tricky, as one person may tend to exaggerate, while another may downplay crucial facts. Most importantly, only the recipient of a message can adequately decide its importance based on their individual needs and context.

THE SOLUTION

In an initial setup phase the user ranks messages based on their priority

Upon installing the app, the user must initially rank the importance of various messages from each user on a scale of 1 to 10. This serves as a reference point for the LLM to categorize future messages effectively.



Other
KEY FEATURES

01

Language independence

By harnessing DeepL's state-of-the-art translation, users gain the flexibility to choose the language in which they wish to receive their messages.

02

Messenger capabilities

Our app provides businesses with all the essential messaging functionality required for efficient communication.

03

Smart notifications

Empowering users to receive notifications based on importance, our app enables them to focus on what truly matters.

04

Continuous improvement

Even after the setup phase, the user can continue rating messages based on their importance, further enhancing the accuracy of predictions.

TECHNICAL IMPLEMENTATION

The following slides aim to explain our app's functionality and its underlying technical realization. We will explore the program flow using an exemplary message as an illustration.

First step
RECIPIENT SETUP

Before using the app, the user gets the chance to customize their settings, ensuring a perfectly tailored user experience.

1: select language

The recipient gets to choose the language for receiving messages, and this decision will become crucial later on. In our example, the recipient prefers to receive messages in German.

2: rank messages

In our example, the user has already ranked a significant number of messages. We store these rankings as key-value pairs (text translated to English; importance rated on a scale of 1-10) using Firebase Realtime Database.

3: select notification threshold

In our example, the user prefers to receive notifications for messages with an importance level equal to or greater than 8, indicating a preference for receiving only fairly important notifications.

THE SENDERS MESSAGE

In the second step, the example message "please call me immediately" is sent to the recipient. We utilize DeepL's REST-API to detect the language and translate it into English, which, in this case, is unnecessary because our example message is already written in English.

```
import deepl
auth_key = 'auth_key'
text = 'please call me immediately'
target_language = 'EN-US'
translator = deepl.Translator(auth_key)
result = translator.translate_text(text,
    target_lang=target_language)
translated_text = result.text
```

Image: Code of the call we make, using the DeepL python library

Language	Percent	Language	Percent
en	89.70%	uk	0.07%
unknown	8.38%	ko	0.06%
de	0.17%	ca	0.04%
fr	0.16%	sr	0.04%
sv	0.15%	id	0.03%
zh	0.13%	cs	0.03%
es	0.13%	fi	0.03%
ru	0.13%	hu	0.03%
nl	0.12%	no	0.03%
it	0.11%	ro	0.03%
ja	0.10%	bg	0.02%
pl	0.09%	da	0.02%
pt	0.09%	sl	0.01%
vi	0.08%	hr	0.01%

Image: Distribution of languages LLama2 was trained on

WHY TRANSLATE TO ENGLISH?

Translating to English as an intermediary step offers a significant advantage. Most LLMs are trained on predominantly English language data, making them perform substantially better in nearly every task.

For reference, please refer to the language distribution of Meta's latest LLM, "Llama 2".

THE PROMPT

In the third step, we prompt the model to make predictions on a scale of 1 to 10, utilizing the message/rating pairs created by the recipient in step one. For this purpose, we would leverage a free-to-use open-source model such as Google's "BERT."

```
from transformers import BertTokenizer, BertForQuestionAnswering
tokenizer = BertTokenizer.from_pretrained('deepset/bert-base-cased-squad2')
model = BertForQuestionAnswering.from_pretrained('deepset/bert-base-cased-squad2')
text = "please call me immeadiately"
question = """How important do you think this message is on a scale of 1 to 10
where 1 is not important and 10 is very important?
Previous messages where rated like this: (show previous messages and ratings)"""
input = tokenizer(question, text,  return_tensors="pt")
output = model(**input)
start_index = output.start_logits.argmax()
end_index = output.end_logits.argmax() + 1
answer = tokenizer.decode(input.input_ids[0, start_index:end_index])
```

```
...    10
```

Images: Example of how a prompt to the model could look like

THE RESULT

Afterward, we send both the message and its priority rating to the recipient. The message is translated into the recipient's specified language, and the priority rating is used to decide whether the user receives a notification or not. Subsequently, the recipient can score the message to update the database.

```
auth_key = 'auth_key'
text = 'please call me immediately'
target_language = 'DE'
translator = deepl.Translator(auth_key)
result = translator.translate_text(text,
    target_lang=target_language)
translated_text = result.text
```

Check if:
detected level (10)
>
notification threshold
set by user (8)

App IMPLEMENTATION

01

Messaging

Utilizes Firebase & Firebase Cloud Messaging for instant, low-latency message delivery and real-time updates, ensuring efficient and reliable communication.

02

Developed with Kotlin

We based our app on Kotlin, the gold standard for Android app development, leveraging its powerful features and benefits.

03

AndroidX and Jetpack

Incorporating industry-renowned libraries and frameworks, we can guarantee a streamlined future development process, enhancing efficiency and maintaining quality of service.

04

MVVM and Navigation Components

By implementing the MVVM architectural pattern and harnessing the power of Navigation Components, we crafted a state-of-the-art product that stands strong with its robustness and maintainability.

FUTURE CONSIDERATIONS

MODEL FINETUNING

At present, we rely on a generic approach of prompting the model with prior ratings, which imposes certain rigid limitations. To achieve full functionality and flexibility, it is imperative to undertake model fine-tuning. This will enable us to unleash the full potential.

CONTEXT

By implementing a context parameter for both the DeepL API and the model, a more comprehensive understanding of the situation and the task at hand could be attained. Consequently, this has the potential to substantially enhance the handling of edge case scenarios.

FEDERATED LEARNING

By employing federated learning techniques, we can harness the collective interactions of all users to train a single, finely tuned model while simultaneously delivering a personalized version for each user. This approach holds the promise of significantly enhancing prediction quality.

USER PRIVACY

Presently, we have implemented a locally deployed LLM (BERT) that aligns with German data privacy standards. However, as we look to the future, there might be a desire to leverage more advanced models like GPT-4 through cloud services like Azure. In such a scenario, we must be vigilant about adhering to legal regulations.

DeepL Hackathon

CONTACT

m.erdem@campus.tu-berlin.de

erik.putzier@campus.tu-berlin.de

linkedin.com/in/merthan

linkedin.com/in/erikputzier

PriotyNotify