ZPP Murmuras - HLD

Gustaw Blachowski Szymon Kozłowski
Natalia Junkiert Kamil Dybek

Introduction

The goal of the project is to create a universal solution for processing phone screen content, such as social media posts and advertisements on websites. In addition, the processing must occur locally on the user's device to protect sensitive data. The data that we will be working with is mostly:

- the location of an element on the screen,
- its type (text, image, etc.),
- its content.

The system aims to enable the analysis of information related to content observed by users, thereby supporting research of both commercial and social nature. Examples of applications include:

1. **Analysis of advertisements viewed by users**

   The system will facilitate the analysis of the reach of promotional coupons. This will allow us to examine the frequency of a particular coupon's display and study competitor actions. These insights can be useful for planning future marketing campaigns.

2. **Research on political and social opinions**

   The system will enable local analysis of private data such as user conversations. This feature will provide reliable data that is difficult to obtain by other methods, for example, insights into political opinions and insights into social opinions.

Existing Solutions

We are not aware of any publicly available solutions that directly address this problem. The closest comparable approaches might be existing multimodal models. On one hand, there are widely available models such as ChatGPT, Gemini, and others. However, as the example of ChatGPT demonstrates, these models are not highly precise solutions for this task. A notable drawback of such models is their very large number of parameters; for example, GPT-3 has 175 billion parameters [1]. On the other hand, there are Computer Vision models that focus on extracting text and bounding boxes from user screen images. Microsoft's OmniParser [4] appears to perform very well in this regard, but its output would still require preprocessing solutions similar to those we are investigating. Furthermore, based on our experiments running OmniParser locally, it appears that it requires CUDA technology for execution, which makes it unsuitable for deployment on mobile devices.

Murmuras' Solution

The existing prototype solution developed by Murmuras uses data in CSV format that describes the screen's content. This data is then subjected to very basic processing (e.g., removal of empty columns), and subsequently processed by ChatGPT-4 mini into JSON format. This solution has two main issues: it does not run locally on mobile devices, as the model is not available for local execution, and the data is often described incorrectly (e.g., volume is treated as the product's price).

Development Opportunities

There is a theoretical possibility to solve these problems. There are Python modules such as *outlines*[5] or *LangChain*[2], which allow for enforcing structured output in models. The first of these works with models from *HuggingFace*.

Specification of the Completed Project

As the core part of the project, we plan to implement the mentioned solution solely for use with coupons.

1. A tool to process the data extracted from the device into a format suitable for use by the model.
2. A Machine Learning tool for extracting the data that is of interest to us.
3. An optional tool for postprocessing the output data from the tool in point 2 into a common format.
4. An application that runs the above three tools on a mobile device. (Optional)

Challenges

Hardware Requirements

Computational Power and RAM

Both modern LLMs and data preprocessing algorithms often require significant resources [6]. At the same time, we want everything to run locally on the mobile device. Therefore, the challenge will be selecting tools that are not too resource-intensive.

Disk Storage

The operation of the application will require a large amount of disk space. We assume that this will not be an issue for the user due to the company's business model. Users are rewarded for installing the solution on their phones.

Benchmarking

To evaluate the quality of our solution, we currently plan to use the benchmark provided by Murmuras. It is based on calculating the similarity function between the output of the used model and the output of the reference model (currently GPT4o-mini). This benchmark may prove to be insufficiently accurate and reliable. There may also be a need to propose an alternative, such as testing the system on artificially generated and labeled data.

Proposed Solution

In the implementation of our solution, we distinguish the following main modules:

Data Acquisition

We will use raw data provided to us by the company Murmuras. The problem is that they are not labeled, so they will not be suitable for use by BERT-based models. Therefore, we will label the data using ChatGPT. Among the mentioned data, we can distinguish .CSV files representing screen views and user session recordings.

Data Preprocessing

We will divide the screen representation in .CSV format into segments corresponding to logical parts (e.g., a single coupon). We will either use a clustering algorithm we have developed or, through fine-tuning, train the LLM model for this task.

Token Labeling

Using a fine-tuned model from the BERT family [3], we will assign classes to individual tokens corresponding to the attributes of interest in the coupons (e.g., price before, price after, etc.).

Model Selection

After preliminary research, we decided to focus on transformer models with parameter counts ranging from 10 million to around 500 million. Most of the options we selected are derivatives of the BERT model [3]. The three main subtypes are:

1. Bert: ~100 million parameters,
2. DistilBert: ~65 million parameters,
3. AlBert: ~11 million parameters.

Their advantage lies in their very small size; experiments [6] have been conducted where the Llama-2 model with 7 billion parameters was run on mobile devices. Models from the BERT family are an order of magnitude smaller.

Postprocessing

Postprocessing will involve aggregating the results of the model.

Deployment on Mobile Devices (Optional)

We will create a mobile application or add functionality to an existing one, in which we will implement the above points. We will create a service running in the background that processes incoming data in real time. If real-time processing turns out to be too resource-intensive, we will implement data storage (assuming daily use of the coupon application for 15 minutes, we estimate the data size to be 15KB) from the screen and process it at night when the user is not using the device. We plan to use the TensorFlow Lite framework (TensorFlow for mobile devices), possibly PyTorch or ONNX, due to their easy integration with applications developed in Android Studio. An interesting option also seems to be the Llama.cpp tool, which has scientific documentation for its use [6]. However, this requires further research.

We aim for the app to be compatible with Android 9+ but do not require it to work on all devices. Data storage and further transmission will be left to the company's app at this time.

Milestones

Research

Planned Completion: 30.11
By the end of November, we aim to have selected the architecture, a specific model, and proposed algorithms for preprocessing and postprocessing.

Proof of Concept

Planned Completion: 31.12
We plan to create a prototype application demonstrating the full functionality.

Idea Gathering for Improvements

Planned Completion: 31.01
January will be a month during which we do not plan intensive work on the project due to exams. We will use this time to potentially finish previous milestones and reflect on the project's direction.

Solution Finalization and Testing

Planned Completion: 30.04
At this stage, we will focus on improving the solution, fixing bugs, and testing.

Bachelor's Thesis

Planned Completion: 30.06
We will focus on writing and refining the bachelor's thesis.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[2] Harrison Chase. LangChain, October 2022.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[4] Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. Omniparser for pure vision based gui agent, 2024.

[5] Brandon T Willard and Rémi Louf. Efficient guided generation for llms. *arXiv preprint arXiv:2307.09702*, 2023.

[6] Jie Xiao, Qianyi Huang, Xu Chen, and Chen Tian. Large language model performance benchmarking on mobile platforms: A thorough evaluation, 2024.