ZPP Murmuras
High Level Design

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.
This project is being created on the request of **Murmuras**, a company specialized in providing analytical data for commercial as well as scientific purposes.
While the proposed method itself is expected to be applicable in various domains, in our work we will focus on the task of discount coupon gathering.
A **discount coupon** is, in our case, a digital voucher offered to the user in the application published by supermarket chain. With the new applications of this type dynamically appearing, and existing ones evolving, it is hard to maintain a reliable data delivery pipeline that would use conventional scrapping techniques. The usage of Large Language Models (LLMs) in our solution aims to resolve this problem.
Apart from that, use cases of developed method will, among others, include the task of collecting profiles of political views of users without violating privacy of their conversations thanks to the data being processed locally.

Existing Solutions And Their Limitations

We are not aware of any publicly available solutions that directly address our problem. There were attempts to use LLMs for web scrapping [**?**][**?**],

but effective deployment of these solutions on mobile phones seems to be an unexplored ground. A notable drawback of language models used there is their very large number of parameters; for example, GPT-3 has 175 billion parameters [**?**].

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*[**?**] or *LangChain*[**?**], which allow for enforcing structured output in models. The first of these works with models from *HuggingFace*. This, alongside with mobile model deployemnts tools like Llama.cpp[**?**] would allow us to perform local inference.

Specification of the Completed Project

As the core part of the project, we plan to design and implement a functional solution that will be able to extract discount coupons from a data from a diverse set of mobile applications. The deployment of the solution on the Android smartphone in form of a mobile application is optional, but we require our proposal to be theoretically possible to do so. Our solution is expected to be generalizable to wider range of problems, as mentioned above. We will provide description of it that will make this generalization possible.
On top of that, we will design and implement a benchmark to measure performance of the developed proposal and compare it with rejected alternatives.

Challenges

Hardware Requirements

Computational Power and RAM

Both modern LLMs and data preprocessing algorithms often require significant resources [**?**]. At the same time, we want everything to run locally on the mobile device. Therefore, the challenge will be selecting model that is not too resource-intensive and can be deployed on a mobile device. We want this model to have no more than 1 billion trainable parameters to ensure it can be deployed on most of the modern smartphones[**?**].

Disk Storage

The operation of the application will require an amount of disk space in order of magnitude of gigabytes[**?**] as the model weights will be stored locally. 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 data-gathering application on their phones.

Benchmarking

Algorithm

To evaluate the quality of our solution, we propose the following benchmarking metric. Assume a batch of samples S_i in form that represents the content of smartphone screen in .CSV format. We define a *coupon* as a tuple of the following fields:

1. *old_price*: fixed precision number, the standard price of related item
2. *new_price*: fixed precision number, the lower price offered to the customer
3. *name*: string, item name
4. *validity*: string, the date of the coupon expiration
5. *percentage*: string, discount as percent

Any of these fields, excluding *name*, can be $None$.
Next, we define the following weights:

$$w_n = 0.4$$
$$w_p = 0.3$$
$$w_{pr} = 0.2$$
$$w_v = 0.1$$
$$w_{po} = 0.5$$
$$w_{pn} = 0.5$$

Now, we define function assigning similarity to pairs of coupons:

$$S(c_1, c_2) = w_n R(c_1[name], c_2[name]) +$$
$$w_p w_{po} Q(c_1[price\_old], c_2[price\_old]) + w_p w_{pn} Q(c_1[price\_new], c_2[price\_new])$$
$$+ w_{pr} R(c_1[percent], c_2[percent]) + w_v R(c_1[validity], c_2[validity])$$

Where:
$R(s_1, s_2)$ is measure of similarity between two sequences as defined in [**?**] and $Q(x, y)$ is 1 if $x = y$ and 0 else.
With each sample S_i we associate a multiset of coupons expected to be extracted from this sample: C_i.
Then, for a multiset $\tilde{C}_i$ obtained by the pipeline being benchmarked for input sample we follow procedure:

$lonely_i \leftarrow 0$
$similarities_i \leftarrow []$
for each $c \in \tilde{C}_i$ **do**
  $c' \leftarrow \arg\max_{c' \in C_i} S(c', c)$
  if $S(c', c) == 0$ **then**
    $lonely_i \leftarrow lonely_i + 1$
    $continue$
  end if
  $similarities_i.append(S(c, c'))$
  $C_i.pop(c')$
end for
$lonely_i += len(C_i)$
for $i = 0; i < lonely_i; i++$ **do**

$similarities_i.append(0)$
end for
$similarity_i \leftarrow mean(similarities_i)$

We then can average similarities across samples.

Dataset

As benchmarking dataset we selected a representative portion of data provided to us by Murmuras. This portion of data is relatively small as it contains less than 200 coupons which enables fast evaluation and is obtained from a content from different applications to increase reliability of the benchmark.

Proposed Solution

Input Data Format and Acquisition

The solution will operate on input data in format of .CSV files representing XML structure that underlies visible screen content. These .CSV files enable us to get information about, among others, location and content of the text fields and their hierarchical structure.
During mentioned below fine-tuning process, we will utilize data provided to us by Murmuras. As the fine-tuning will be performed in supervised fashion, expected model output will be generated with the help of ChatGPT. At a runtime, pipeline will use data scrapped from smartphone screen by the tool that is provided by Murmuras too.
Currently we are developing two different solutions to the problem.

Solution 1: Labeling Via Lightweight LLM

Our pipeline will consist following modules:

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 small LLM model for this task.

Token Labeling

Using a fine-tuned model from the BERT family [**?**], 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 [**?**]. 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 [**?**] 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 batching the data labeled by the model into coupons.

Solution 2: directly querying LLM

In this more straightforward approach we will fine-tune model llama3.2-1b from Llama family[**?**], chosen due to its size being optimal which is implied by above considerations, its liberal license and support from fine-tuning frameworks[**?**]. The model itself will be run by llama-cpp[**?**][**?**] tool. It will ensure that the model output is in JSON format, which will reduce postprocessing to trivial aggregation.

Model fine-tuning

Fine tuning will be performed in Modal[**?**] cloud environment. To speed up training unsloth[**?**] tool will be used.

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 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.