ZPP Murmuras - HLD

Gustaw Blachowski Szymon Kozłowski
Natalia Junkiert Kamil Dybek

Wprowadzenie

Celem projektu jest stworzenie uniwersalnego rozwiązania do przetwarzania danych uzyskanych z ekranu smartfonu (wspierany system: Android 9 wzwyż), takich jak lokalizacja elementu na ekranie, jego typ (tekst, zdjęcie itp), jego zawartość. Dane te reprezentują zawartość widzianą przez użytkownika - posty na mediach społecznościowych, reklamy na witrynach, itp. System ten ma umożliwić analizę informacji dotyczących treści obserwowanych przez użytkowników, wspierając tym samym badania o charakterze komercyjnym i społecznym.

Przykładowymi zastosowaniami danych przetworzonych przez system są (1) analiza danych o oglądanych przez użytkownika reklamach oraz (2) badanie poglądów politycznych i społecznych.

(1) System będzie umożliwiał wygodne badanie zasięgów kuponów promocyjnych. Dzięki temu będziemy mogli sprawdzić, jaka jest częstotliwość wyświetlania danego kuponu oraz zbadać działania konkurencji. Może to być pomocne w planowaniu przyszłych akcji marketingowych.

(2) (Kierunek rozwojowy): System będzie umożliwiał lokalną analizę danych prywatnych takich jak konwersacje użytkowników. Dostarczy to wiarygodnych danych trudno dostępnych innymi metodami, na przykład poglądów politycznych i społecznych.

Istniejące rozwiązania

Nie wiadomo nam nic o rozwiązaniach bezpośrednio tego problemu. Najbliższe im mogą być istniejące modele multimodalne. Z jednej strony mamy

właśnie ChatGPT, Gemini i wiele innych ogólnodostępnych modeli, lecz przykład ChatGPT pokazuje że nie jest to rozwiązanie bardzo dokładne. Wadą tego typu modeli jest jednak bardzo duża ilośc parametrów. Z drugiej strony mamy też modele Computer Vision, które skupiają się na ekstrakcji tekstu i bounding boxów z obrazów ekranu użytkownika. OmniParser od Microsoftu wydaje się sprawować w tej kwestii bardzo dobrze, ale wynik jego działania wymagałby podobnych rozwiązań do tych, które badamy w ramach preprocessingu. Do tego wymaga on technologii CUDA do uruchomienia więc nie będzie możliwe uruchomienie go na urządzeniu mobilnym.

Rozwiązanie Murmurasa

Istniejące prototypowe rozwiązanie korzysta z danych w formacie CSV opisujących zawartość ekranu. Następnie dane te są poddawane bardzo podstawowej obróbce (usuwanie pustych kolumn itp), a następnie są przetwarzane przez ChatGPT4o-mini (za pomocą prompt-engineeringu) do formatu JSON. Rozwiązanie to ma dwa zasadnicze problemy: nie działa ono lokalnie na urządzeniu mobilnym (model jest niedostępny do uruchomienia lokalnie), a dane są często opisywane niepoprawnie (np. objętość jest traktowana jak cena produktu).

Możliwości rozwoju

Istnieje teoretycznie możliwość na rozwiązanie tych problemów. Istnieją moduły w języku Python takie jak *outlines*[3] bądź *LangChain*[1], pozwalające na wymuszenie w modelach ustrukturyzowanego outputu, z czego pierwsza z nich działa z modelami z *HuggingFace*.

Specyfikacja skończonego projektu

Jako podstawową część projektu planujemy zaimplementowanie wspomnianego rozwiązania jedynie do zastosowania z kuponami.
1. Wymagane jest narzędzie, które przetworzy dane wyekstrahowane z urządzenia do postaci nadającej się do użycia przez model.
2. Wymagane jest użycie narzędzia z dziedziny Machine Learningu do ekstrakcji interesujących z naszej perspektywy danych.

3. Opcjonalne jest narzędzie do postprocessingu danych wyjściowych narzędzia z punktu 2 do wspólnego formatu.
4. Opcjonalne jest wdrożenie powyższych trzech narzędzi na urządzenie mobilne.

Wyzwania

Wymagania sprzętowe

Moc obliczeniowa i pamięć operacyjna

Zarówno współczesne LLMy jak i algorytmy preprocessingu danych wymagają często dużej ilości zasobów [4]; jednoczesnie chcemy aby wszystko działało lokalnie na urządzeniu mobilnym. Wyzwaniem więc będzie dobór narzędzi które nie będą zbyt zasobożerne.

Pamięć dyskowa

Funkcjonowanie aplikacji będzie wymagać użycia dużej przestrzeni dyskowej. Zakładamy że nie będzie to problemem dla uzytkownika ze względu na model biznesowy firmy (uzytkownicy są wynagradzani za zainstalowanie rozwiązania na telefonie)

Benchmarking

W celu oceny jakości naszego rozwiązania obecnie planujemy posługiwać się benchmarkiem zapewnionym nam przez Murmuras. Bazuje on na obliczaniu funkcji podobieństwa między wynikiem użytego modelu a wynikiem modelu wzorcowego (obecnie jest to GPT4o-mini). Benchmark ten może okazać się niewystarczająco dokładny i miarodajny. Może także pojawić się konieczność zaproponowania alternatywy, przykładowo testowania systemu na sztucznie wygenerowanych i poetykietowanych danych.

Propozycja rozwiązania

W implementacji naszego rozwiązania wyróżniamy następujące 5 głównych modułów:

Pozyskiwanie danych

Będziemy korzystać z surowych danych dostarczonych nam przez firmę Murmuras. Problemem jest to, że nie są one poetykietowane, więc nie będą one nadawać się do użycia przez modele klasy BERT. Dlatego też, korzystając z Chata-GPT, poetykietujemy je. Wśród wspomnianych danych wyróżnić można dane w postaci .CSV reprezentujące widoki ekranu oraz nagrania sesji uzytkownika.

Preprocessing danych

Podzielimy reprezentację ekranu w formie .CSV na segmenty odpowiadające logicznym częściom (np. pojedynczy kupon). Będziemy korzystać z opracowanego przez nas algorytmu klastrowania lub metodą subtelnego strojenia douczymy model LLM do tego zadania.

Etykietowanie tokenów

Korzystając z subtelnie dostrojonego modelu z rodziny BERT [2] przypiszemy kolejnym tokenom klasy odpowiadające interesującym nas atrybutom kuponów (np. cena przed, cena po, itd.)

Wybór modelu Po wstępnym researchu postanowiliśmy skupić się na modelach typu transformer o liczbie parametrów z zakresu 10 do około 500 milionów. Większość wybranych przez nas opcji to pochodne modelu BERT[2].
3 główne podtypy to: 1. Bert 100mln parametrów
2. DistilBert 65mln parametrów
3. AlBert 11 mln parametrów
Ich zaletą jest bardzo mały rozmiar; przeprowadzono eksperymenty[4] w których na urządzeniach mobilnych uruchamiano model Llama-2 z 7 miliardami parametrów. Modele z rodziny BERT są rzędy wielkości mniejsze.

Postprocessing

Postprocessing będzie polegać na agregacji wyników modelu.

Deployment na urządzeniu mobilnym (Opcjonalne)

Stworzymy aplikację mobilną bądź dodamy funkcjonalność do istniejącej w ramach której zaimplementujemy powyższe punkty. Utworzymy usługę działającą w tle i przetwarzający napływające dane w czasie rzeczywistym. Jeśli okaże się, że przetwarzanie w czasie rzeczywistym jest zbyt kosztowne zaimplementujemy przechowywanie danych (przy założeniu dziennego użycia aplikacji z kuponami przez 15min, ilość danych szacujemy na 15KB) z ekranu i ich analizę w nocy, gdy użytkownik nie korzysta z urządzenia. Planujemy wykorzystać framework TensorFlow Lite (TensorFlow dla urządzeń mobilnych), ewentualnie PyTorch bądź ONNX ze względu na łatwą integrację z aplikacjami rozwijanymi w Android Studio. Ciekawą opcją wydaje się także narzędzie Llama.cpp, którego użycie jest udokumentowane naukowo[4]. Na ten moment wymaga to jednak więcej badań. Chcemy, by aplikacja była kompatybilna z Androidem 9+. Nie jest wymagane, by aplikacja działała na wszystkich urządzeniach.
Przechowywanie i dalsze przekazywanie danych pozostawiamy na ten moment aplikacji firmy.

Kamienie Milowe

Research

Planowane ukończenie 30.11
Do końca listopada zamierzamy mieć wybraną architekturę, konkretny model i propozycje algorytmów odpowiedzialnych za preprocessing i postprocessing.

Proof of Concept

Planowane ukończenie 31.12
Planujemy stworzyć prototypową aplikację demonstrującą całościową funkcjonalność.

Sesja/zbieranie pomysłów na ulepszenia

Planowane ukończenie 31.01
Styczeń będzie miesiącem, w trakcie którego nie planujemy bardzo intensywnej pracy nad projektem ze względu na sesję. Przeznaczymy ten czas na

ewentualne dokończenie poprzednich kamieni milowych oraz na przemyślenie kierunku projektu.

Rozwój docelowego rozwiązania

Planowane ukończenie 30.04
Na tym etapie zajmiemy się ulepszeniem rozwiązania, usunięciem błędów i testowaniem.

Praca Licencjacka

Planowane ukończenie 30.06
Skupimy się na napisaniu i dopracowaniu pracy licencjackiej.

Literatura

[1] Harrison Chase. LangChain, October 2022.

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

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

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