ZPP Murmuras - HLD

Gustaw Blachowski Szymon Kozłowski
Natalia Junkiert Kamil Dybek

Wprowadzenie

Celem projektu jest stworzenie uniwersalnego rozwiązania do analizy danych tekstowych, który w przyszłości będzie można rozszerzyć na inne rodzaje danych zbieranych przez urządzenia mobilne, takie jak dane lokalizacyjne (GPS) czy dane multimedialne (zdjęcia, filmy, dźwięki). System ten ma umożliwić pozyskiwanie i analizę informacji dotyczących treści odbieranych i generowanych przez użytkowników, wspierając tym samym analizy o charakterze komercyjnym i społecznym.

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

(1) System będzie umożliwiał zbieranie danych o wyświetlanych przez użytkownika reklamach oraz analizowanie tych danych z perspektywy strategii reklamowych i rynkowych. Dane takie jak liczba kliknięć i czas poświęcony na interakcję lub wyświetlanie reklam umożliwi ocenianie efektywności kampanii reklamowych oraz dostosowanie strategii marketingowych do preferencji użytkowników. Dzięki temu możliwe będzie lepsze targetowanie reklam, optymalizacja budżetów marketingowych oraz zwiększenie zaangażowania odbiorców poprzez prezentowanie bardziej trafnych i interesujących treści reklamowych.

(2) System będzie umożliwiał lokalną analizę danych prywatnych takich jak prywatne wiadomości użytkownika, dając unikalną możliwość zbierania wiarygodnych danych o poglądach politycznych i społecznych. Ze względu na

wrażliwość takich informacji, dane takie będą musiały być z dużą dokładnością anonimizowane, na przykład implementując rozwiązania z zakresu *differential privacy*.

Istniejące rozwiązania

jakie już toole istnieją i jaką mają efektywność? murmuras ma jakiegoś wstępnego solva, napisać o jego limitacjach i co chcemy poprawić względem ich solva

Zastosowania biznesowe

We wprowadzeniu podane zostały przykłady zostosowań biznesowych tego systemu. W następującej sekcji zostanie to doprecyzowane.

Specyfikacja skończonego projektu

wymagania klienta oraz jak planujemy to zaadresować

Wyzwania

0.1 Ograniczenia prawne

0.2 Zapewnianie prywatności użytkowników

Zapewnienie prywatności danych prywatnych użytkowników poprzez *differential privacy* oraz skuteczne rozróżnienie danych prywatnych od półprywatnych lub publicznych

0.3 Wymagania sprzętowe

Wymagania sprzętowe LLMów mogą okazać się zbyt duże na analizę zebranych danych w czasie rzeczywistym, zatem koniecznym może się okazać przetwarzanie danych w nocy.

0.4 Zaimplementowanie modelu spełniające wymagania

Zapewnienie prywatności danych prywatnych użytkowników poprzez *differential privacy* oraz skuteczne rozróżnienie danych prywatnych od półprywatnych lub publicznych. Zaimplementowanie *federated learning*, aby dotrenowywać model. Zrozumienie kontekstu treści, wykrywając niuanse językowe oraz intencje autora. Zapewnienie wysokiej jakości analizy danych oraz dokładne wyniki, szczególnie przy analizie niejednoznacznych bądź niepełnych danych.

Propozycja rozwiązania

Ogólny zarys

Aplikacja mobilna zbierająca dane w urządzeniu i douczająca lokalny model-klasyfikacyjny/rozpoznający interesujące nas sekcje.

Zbieranie widoków z ekranu smartfonu

- Android Accessibility API - istniejące narzędzie wewnątrz firmy, wymagana integracja
- Maskowanie danych prywatnych - NER (Named Entity Recognition), modele językowe

Preprocessing widoków

Standaryzacja danych umożliwiająca uczenie maszynowe na tych danych.

Przetwarzanie z wykorzystaniem MLa

- Klasyfikacja sekcji - webscraping
- Wykorzystanie LLM (ewentualnie alternatywy) do pozyskania z tekstu oznaczonych i ustrukturyzowanych informacji
- Aktualizacja globalnego modelu - *federated learning*

Rozwinięcie planu implementacji

Aplikacja mobilna

- Zbieranie danych: TBD

Narzędzia

- Android Studio: `https://developer.android.com/studio`
- Android accessibility services: `https://developer.android.com/guide/topics/ui/accessibility/service`
- Wewnętrzny tool firmy: Android Accessibility API

Modele ML

Use Cases

- Klasyfikacja
- Scraping

Architektury

- LLM (transformers): elastyczność, potencjalna multimodalność kosztem dużych wymagań sprzętowych
- RNN: lightweightowy, prosty w samodzielnej implementacji, mniejsza elastyczność
- Federated learning: rozproszenie procesu trenowania między urządzenia i synchronizowanie modeli co jakiś czas

Frameworki

- TensorFlow Federated: prosty flow od stworzenia modelu poprzez jego konwersję na TensorFlow Lite aż do deploymentu, ale nie wiadomo na ten moment, czy wspiera heterogeniczny FL.

- HeteroFL (licencja MIT): Framework pozwalający na Federated Learning przy heterogenicznych modelach lokalnych, co pozwoli na deployment rozwiązania na większej liczbie urządzeń.

Wyzwania

- Wymagania sprzętowe LLMów mogą okazać się zbyt duże na analizę zebranych danych w czasie rzeczywistym - możliwe rozwiązanie: przetwarzanie danych w nocy.
- Pozyskanie danych i zasobów do trenowania modelu. Opracowanie algorytmu przetwarzającego dane zbierane na urządzeniach w dowolnej sytuacji.
- Zapewnienie prywatności danych prywatnych użytkowników poprzez *differential privacy* oraz skuteczne rozróżnienie danych prywatnych od półprywatnych lub publicznych.
- Zaimplementowanie *federated learning*, aby dotrenowywać model.
- Zrozumienie kontekstu treści, wykrywając niuanse językowe oraz intencje autora.
- Zapewnienie wysokiej jakości analizy danych oraz dokładne wyniki, szczególnie przy analizie niejednoznacznych bądź niepełnych danych.

Kolejne kroki pracy

- Implementacja algorytmu standaryzacji danych: Dane które otrzymamy mają strukturę drzewiastą; trzeba te dane posegregować (np. ze względu na głębokość drzewa, ojców, etc.) i ułożyć w formę, którą łatwo będzie zaserwować modelowi do nauki.
- Prototyp możliwy do utworzenia poza urządzeniami docelowymi; pozyskanie danych treningowych - otrzymane od firmy, w miarę możliwości samodzielnie wygenerowane.
- Wybór modelu: Interesują nas modele, które będziemy w stanie uruchomić i trenować na telefonie.

- Istniejący wewnątrz firmy prototyp korzysta z prompt engineeringu w Chat-GPT, ale jest to model zbyt duży i wymagający; jedną z opcji jest model bez promptów, gdyż wpływają one negatywnie na wymagania modelu, bądź wyjście poza LLM.
- Chcemy, aby nasz model był kompatybilny z frameworkami Federated Learning, takimi jak TensorFlow oraz HeteroFL.
- Przetestujemy i porównamy różne modele poza urządzeniem docelowym.
- Mamy dwie opcje odnośnie ternowania mocelu: trenować cały model lokalnie na telefonach, dzięki czemu dane prywatne będą miały większy wpływ na wynik, albo wytrenować podstawowy model na serwerze i dotrenowywać go na urządzeniach końcowych, co jest tańsze obliczeniowo.
- Integracja modelu z aplikacją.

Deployment gotowego modelu na urządzenia końcowe

Model będzie klasyfikował dane do późniejszego użycia. Model powinien być w stanie nadal się uczyć na nowych danych w oparciu o uczenie federacyjne.

Źródła

- LLM finetuning course: `https://maven.com/parlance-labs/fine-tuning`
- Huggingface NLP course: `https://huggingface.co/learn/nlp-course/en/chapter1/1`