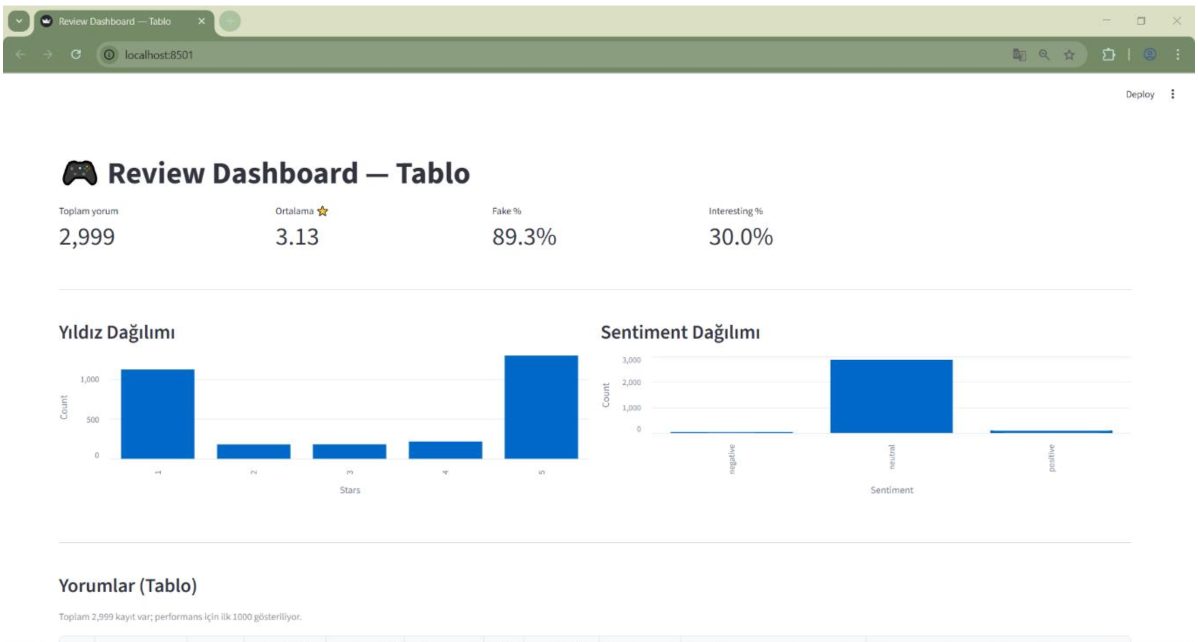
Case Study 2: Google Play Review Analysis System

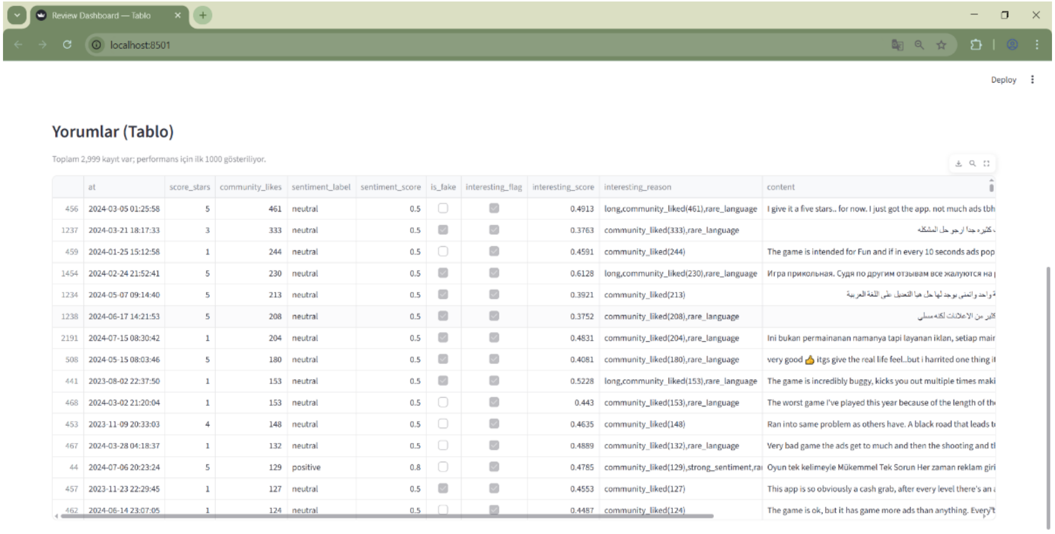
This project first collects all reviews for the game from Google Play, then automatically analyzes each review for sentiment (positive / neutral / negative), detects whether it's likely fake (bot, spam, duplicate, or fabricated), and scores how "interesting" the comment is. Technically, reviews are pulled into a CSV using google-play-scraper; sentiment uses a multilingual model (XLM-RoBERTa); fake detection combines simple rules (links/phones), timing patterns (bursts), and similarity clusters (text embeddings + DBSCAN), with exceptions so natural short praise and genuine "ads/time" complaints aren't flagged incorrectly; interesting reviews are scored using length, emojis/punctuation, likes, rare wording, and zero-shot labels. Run analysis with python analyze.py and open a simple dashboard with streamlit run dashboard.py. Thresholds (burst size, duplicate cluster size, etc.) are easy to tweak in the code.

- **What it does**
  - Collects all Google Play reviews for the game.
  - Tags each review with sentiment, a fake/not-fake decision, and an "interesting" score.
  - Shows results in a simple Streamlit table and saves them to CSV/JSONL.
- **How to run (quick)**
  - Analyze: python analyze.py → creates patrol_officer_reviews_analyzed.csv
  - Dashboard: **streamlit run dashboard.py** → open the simple UI

- **SCREENSHOTS:**



-



Yorumlar (Tablo)

Toplam 2,999 kayıt var; performans için ilk 1000 gösteriliyor.

	at	score_stars	community_likes	sentiment_label	sentiment_score	is_fake	interesting_flag	interesting_score	interesting_reason	content
456	2024-03-05 01:25:58	5	461	neutral	0.5	☐	☑	0.4913	long,community_liked(461),rare_language	I give it a five stars.. for now. I just got the app. not much ads tbh
1237	2024-03-21 18:17:33	3	333	neutral	0.5	☑	☑	0.3763	community_liked(333),rare_language	ـ كثيره جدا ارجو حل المشكله
459	2024-01-25 15:12:58	1	244	neutral	0.5	☐	☑	0.4591	community_liked(244)	The game is intended for Fun and if in every 10 seconds ads pop
1454	2024-02-24 21:52:41	5	230	neutral	0.5	☑	☑	0.6128	long,community_liked(230),rare_language	Игра прикольная. Судя по другим отзывам все жалуются на
1234	2024-05-07 09:14:40	5	213	neutral	0.5	☑	☑	0.3921	community_liked(213)	ة واحد واتمنى يوجد لها حل هيا التعديل على اللغة العربية
1238	2024-06-17 14:21:53	5	208	neutral	0.5	☑	☑	0.3752	community_liked(208),rare_language	كثير من الاعلانات لكنه مسلي
2191	2024-07-15 08:30:42	1	204	neutral	0.5	☑	☑	0.4831	community_liked(204),rare_language	Ini bukan permainanan namanya tapi layanan iklan, setiap mair
508	2024-05-15 08:03:46	5	180	neutral	0.5	☑	☑	0.4081	community_liked(180),rare_language	very good 👍 itgs give the real life feel..but i harrited one thing it
441	2023-08-02 22:37:50	1	153	neutral	0.5	☑	☑	0.5228	long,community_liked(153),rare_language	The game is incredibly buggy, kicks you out multiple times maki
468	2024-03-02 21:20:04	1	153	neutral	0.5	☐	☑	0.443	community_liked(153),rare_language	The worst game I've played this year because of the length of th
453	2023-11-09 20:33:03	4	148	neutral	0.5	☐	☑	0.4635	community_liked(148)	Ran into same problem as others have. A black road that leads t
467	2024-03-28 04:18:37	1	132	neutral	0.5	☐	☑	0.4889	community_liked(132),rare_language	Very bad game the ads get to much and then the shooting and t
44	2024-07-06 20:23:24	5	129	positive	0.8	☐	☑	0.4785	community_liked(129),strong_sentiment,ra	Oyun tek kelimeyle Mükemmel Tek Sorun Her zaman reklam giri
457	2023-11-23 22:29:45	1	127	neutral	0.5	☑	☑	0.4553	community_liked(127)	This app is so obviously a cash grab, after every level there's an
462	2024-06-14 23:07:05	1	124	neutral	0.5	☐	☑	0.4487	community_liked(124)	The game is ok, but it has game more ads than anything. Every't

-

1) Which NLP models / libraries were used and why?

- **Hugging Face Transformers**
  - **cardiffnlp/twitter-xlm-roberta-base-sentiment** for multilingual sentiment (works well across many languages including Turkish).
  - **joeddav/xlm-roberta-large-xnli** for zero-shot classification (to label tone like *humorous*, *constructive*, etc.).
- **Sentence-Transformers** (paraphrase-multilingual-MiniLM-L12-v2) to produce multilingual sentence embeddings for similarity / clustering.
- **Classical tools:** TF-IDF for text rarity, DBSCAN (scikit-learn) for cluster detection, regex heuristics for simple signals.
  Why: this mix gives good multilingual accuracy (pretrained models) while keeping cost and complexity reasonable; embeddings + DBSCAN are fast and interpretable for duplicate detection.

2) What strategy was used for fake review detection?

A **hybrid** approach combining simple rules, temporal signals and embeddings:

- **Rules:** regex for URLs/phones/emails → immediate suspicious (bot/promo).
- **Spam (burst):** same text repeated many times in a short window (e.g., ≥5 in 10 min) → spam.
- **Exact duplicate:** identical texts, stronger if same user repeats.
- **Near-duplicate:** sentence embeddings + DBSCAN clustering; use cluster-size thresholds (short texts need much larger clusters than long ones).
- **Fabricated:** text/metadata mismatch or incoherence + another supporting signal → flagged.

- **Safeguards:** whitelist very common short praises and don't mark organic complaint patterns (e.g., "1 min ads / 15s gameplay") as fake based on similarity alone.
  Final output: conservative boolean flag (fake_flag_v2) plus an audit reason.

3) How were sentiment scores computed?

- **Primary:** use the pretrained XLM-RoBERTa sentiment model via transformers.pipeline. For each review we store:
  - sentiment_label = positive / neutral / negative
  - sentiment_score = model confidence (0–1)
- **Fallback:** if the model cannot be loaded, a simple keyword-based heuristic is used (keeps the pipeline usable in restricted environments).

4) How were "interesting" reviews selected? (automatic + examples)

- **Signal fusion + zero-shot:** combine several signals into a single score:
  - text length, punctuation (exclamation/question), emoji count, number of likes, sentiment intensity, TF-IDF rarity, plus zero-shot label scores (humorous, constructive, exaggerated, suggestive, novel).
- **Thresholding:** mark reviews above a percentile threshold (e.g., top 30%) as interesting_flag.
- **Rationale:** produce a short reason string (e.g., long, humorous, community_liked(12)).
- **Examples:**
  - *Humorous & complaint:* "1 min ads, 15s gameplay 😂" → interesting (emoji + complaint + expressive).
  - *Constructive:* "Level 3 crashes — here's how to reproduce…" → interesting (long + constructive).
  - *Exaggerated:* "Addictive, I can't stop 😅" → interesting (strong sentiment + emoji).

5) How is review scraping implemented and is the data continuously updatable?

- **Implementation:** google-play-scraper.reviews_all(pkg, lang, country) loops over many languages and countries to maximize coverage.
- **Checkpointing:** save every reviewId to a checkpoint file so scraping can resume without duplicates.

- **Storage:** append to CSV and write Parquet partitions for efficient incremental storage.
- **Continuous updates:** yes — run the scraper on a schedule (cron or cloud scheduler) to append new reviews incrementally.

6) If you needed to make this scalable and real-time, how would you architect it?

- **Ingest:** scheduled or event-driven scrapers push raw reviews into a message queue (Kafka).
- **Processing pipeline (workers):** containerized workers consume the queue:
    - batch sentiment inference (GPU optional),
    - batched embedding generation (GPU),
    - online or windowed clustering (using ANN + streaming clustering).
- **Embedding & similarity:** dedicated embedding service + ANN index (FAISS/HNSW) for fast near-dup lookups.
- **Storage/Analytics:** materialize outputs into OLAP store (ClickHouse/BigQuery) and Parquet on object storage.
- **Serving:** API (FastAPI) + cache (Redis) + simple dashboard (Streamlit / React).
- **Monitoring:** metrics for fake rate, burst frequency, model latency, and data drift; CI to update thresholds and models.
- **Advantages:** scalable, low-latency inference, and ability to reprocess historical batches.