

TEKNİK DEĞERLENDİRME / TÜRKÇE ÖZET

Türkçe yorumlarda model karşılaştırması

Jev API, hazır Laya ve tam fine-tuning uygulanmış Laya'yı Türkçe ikili duygu analizinde aynı sabit girdilerle değerlendirdik.

15.000

test yorumu

40.000

Laya-FT eğitim yorumu

5.000

geliştirme yorumu

SONUÇLAR

Macro-F1 (%) · yüksek değer daha iyi

Model	Ana test 10.000 yorum	Ayrı koleksiyon 5.000 yorum
Jev API	%93,59	%88,86
Laya Hazır model	%77,74	%76,29
Laya-FT Tam fine-tuning	%93,26	%83,72

Macro-F1, iki sınıfın F1 puanlarını eşit ağırlıkla özetler.

Fine-tuning kazandırdı; diğer koleksiyonda fark sürdü.

Laya'nın macro-F1 skoru ana testte **15,52**, ayrı koleksiyonda **7,43 yüzde puan** arttı. Jev iki bölümde de daha yüksek gözlenen skoru aldı; ana testteki küçük fark kesin üstünlük kanıtı değil.

Toplam skor tek başına yeterli değil. Ayrı koleksiyonda olumsuz yorumları yakalama oranı, Laya'da %92,60 iken Laya-FT'de %75,36 oldu.

KAPSAM

Tek eğitim koşusu; kaynak etiketlerine göre ölçüm. Etiketler bağımsız insan doğrulamasından geçmedi. Ayrı koleksiyonun platform bağımsızlığı doğrulanmadı. Bu ön değerlendirme, asıl projenin üretim performansını doğrulamaz.

API and Local Models for Turkish Sentiment Analysis:

A Fixed-Test Comparison Before and After Full Fine-Tuning

Bora Erol Özkan

One Eye Systems

25 September 2026

Abstract

This research note compares Jev API, pretrained Laya Multilingual and a fully fine-tuned Laya model on binary Turkish review sentiment classification. All three systems receive the same fixed inputs for 10,000 main-test reviews and 5,000 reviews from a separately held-out collection. Laya is fine-tuned on 40,000 distinct reviews; checkpoint selection uses a separate 5,000-review development set. Fine-tuning increases macro-F1 from 77.74% to 93.26% on the main test and from 76.29% to 83.72% on the held-out collection. Jev achieves 93.59% and 88.86%, respectively. All 45,000 test predictions complete successfully. Local inference is measured on an Apple M1 Pro; API latency includes network overhead. The findings show a substantial task-specific gain from fine-tuning, alongside a remaining performance gap on a different collection. They describe one training run against source labels, not a human-validated or demonstrably platform-independent benchmark.

Keywords: Turkish sentiment analysis; full fine-tuning; multilingual models; fixed-test evaluation; local inference.

1 Introduction

For a narrowly defined classification task, a hosted API and a locally deployed model offer different practical arrangements. The relevant empirical question is how much a task-specific training set improves the local model, and whether that improvement persists beyond the collection used for training. We examine this question using Jev version `jev-1.13.0`[1], Laya Multilingual[2] and its fine-tuned counterpart (Laya-FT). This is a comparison of three specified systems under one frozen protocol, not a claim of state-of-the-art performance across Turkish sentiment datasets.

2 Data and experimental protocol

2.1 Collections and separation

The main collection is *MüşteriYorumları*[3], which contains e-commerce reviews with star ratings. Ratings of one or two stars are mapped to negative sentiment, four or five stars to positive sentiment, and three-star reviews are excluded. The collection’s original splits are regrouped for this study; the scores are therefore not directly comparable with its published split-based evaluations. *Turkish Product Reviews*[4] supplies the second test collection, using its existing binary labels.

Training, development, calibration and the main test draw only from the main collection. The second collection is excluded from all three fitting or selection partitions. The training set contains 14,253 negative and 25,747 positive reviews (35.63% and 64.37%); it is neither oversampled nor augmented with synthetic examples. The calibration partition is reserved and unused in this comparison. Test labels are used only for scoring, never as model inputs.

Cleaning removes empty or malformed records, exact duplicates, conflicting label groups and detected groups shared across collections. Near-duplicate candidates are retrieved using character five-gram Min-Hash/LSH and verified at Jaccard similarity ≥ 0.85 ; texts shorter than 20 characters receive exact matching

Table 1: Frozen study partitions. Only the training set is class-imbalanced.

Partition	Collection	Negative	Positive	Total
Training (Y)	Main	14,253	25,747	40,000
Development	Main	2,500	2,500	5,000
Calibration (unused)	Main	2,500	2,500	5,000
Main test (X)	Main	5,000	5,000	10,000
Held-out test (X)	Second	2,500	2,500	5,000

only. Candidate retrieval is approximate, so undetected near duplicates remain possible. Email addresses, phone numbers and URLs are masked. A shared tokenizer-based length policy produces the same input text for all systems. The Laya profile allows 1,024 tokens overall and a 256-token question/option budget. Partition and task hashes are fixed before evaluation. The second collection’s original platform family is undocumented; it is therefore called a *held-out collection*, not an independent-source test.

2.2 Models and training

Every system receives the same Turkish sentiment question and definitions of the two classes. No test results are used to select the question, training recipe or checkpoint. Jev predictions are not training targets. The local baseline uses Laya Multilingual at revision `e4e9ddf21a7b`, with Laya 0.3.20 from commit `23a17522aa49`; full identifiers accompany the manuscript. Training follows the pinned Laya decision-model implementation rather than substituting a generic sequence-classification head[5].

Laya-FT is trained on one NVIDIA A100-SXM4-40GB GPU in Google Colab using PyTorch 2.8.0 and bfloat16 mixed precision. All 321,908,995 parameters are trainable; the task loss does not directly supervise the separate act/escalate output. The objective combines the upstream proper-reward, noisy-logit policy-gradient term with unit-weight cross-entropy. Training uses AdamW, encoder/head learning rates of $2.5 \times 10^{-5}/10^{-4}$, weight decay 0.01, cosine decay to 10^{-6} , gradient clipping at 1.0, microbatches of two and 16-step accumulation (effective batch size 32). Examples are shuffled without replacement, with no class weighting or balanced resampling. Four epochs produce 5,000 optimiser updates. The predeclared selection rule is development macro-F1, earliest checkpoint on ties, and patience two. The fourth epoch is selected (development macro-F1 93.66%). One training seed, 20260924, is evaluated.

Checkpoint continuation preserves optimiser and random-number state. Following a Drive write-limit error, the last 325 updates are completed on the same A100 using temporary Colab storage. The exported model passes archive and file-hash verification, CUDA reload checks and a tolerant comparison of decoded probabilities on 20 fixed development examples after Mac import. These are transport and implementation checks, not additional quality results.

2.3 Evaluation and measurement

Accuracy and macro-F1 are computed against the source labels. For class c , $P_c = TP_c/(TP_c + FP_c)$, $R_c = TP_c/(TP_c + FN_c)$ and $F1_c = 2P_cR_c/(P_c + R_c)$; macro-F1 averages the two class F1 scores. Negative recall is thus distinct from the precision of negative predictions. Table 2 also reports 95% percentile intervals from 2,000 bootstrap replicates, stratified by source and class and resampling detected duplicate groups. These intervals describe test-sample uncertainty conditional on the fitted model, not variation across training seeds. No new calibration or decision-threshold optimisation is applied.

Both local models run separately on the same Apple M1 Pro with 16 GB unified memory, macOS 26.6.2 and PyTorch 2.8.0: MPS, FP32, batch size one, ten development warm-up examples, and no AMP, compilation or CPU fallback. GPU synchronisation brackets timing. Jev runs separately and uses one review per HTTP request. Its first 2,756 main-test requests use serial fresh connections; the rest use a connection pool with up to eight concurrent requests. These latency regimes are kept separate.

3 Results

Fine-tuning raises macro-F1 by 15.52 percentage points on the main test and 7.43 points on the held-out collection. Jev’s observed advantage over Laya-FT is 0.32 and 5.14 points, respectively. The small main-test

Table 2: Classification results against source labels (percent).

Model	Accuracy	Macro-F1	95% interval	P_{neg}	R_{neg}
<i>Main test: 10,000 reviews</i>					
Jev	93.59	93.59	[93.06, 94.07]	95.70	91.28
Laya	78.41	77.74	[76.96, 78.54]	71.10	95.72
Laya-FT	93.27	93.26	[92.78, 93.75]	96.02	90.28
<i>Held-out collection: 5,000 reviews</i>					
Jev	88.90	88.86	[88.00, 89.67]	93.83	83.28
Laya	76.88	76.29	[75.11, 77.44]	70.45	92.60
Laya-FT	83.84	83.72	[82.74, 84.77]	90.75	75.36

P_{neg} : negative precision; R_{neg} : negative recall. Intervals quantify test-sample uncertainty, not training-seed variation.

gap should not be read as a statistically established superiority: no paired significance test or repeated-seed experiment is reported here.

The improvement also changes the error balance. On the held-out collection, Laya-FT raises negative precision from 70.45% to 90.75%, while negative recall falls from 92.60% to 75.36%. Its confusion matrix, with true negative/ positive classes as rows and predicted negative/positive classes as columns, is $\begin{bmatrix} 1884 & 616 \\ 192 & 2308 \end{bmatrix}$. Thus higher aggregate performance does not imply fewer missed negative reviews. Both-class precision, recall, F1 and all confusion matrices are included in the accompanying machine-readable results.

Table 3: Observed single-review latency in milliseconds. Conditions differ; see text.

Model	Test	Execution	n	p50	p95
Jev	Main	Serial	2,756	755.13	837.53
Jev	Main	Pooled, up to 8	7,244	295.26	352.96
Jev	Held-out	Pooled, up to 8	5,000	291.55	348.90
Laya	Main	MPS / FP32	10,000	50.00	70.92
Laya	Held-out	MPS / FP32	5,000	54.74	76.12
Laya-FT	Main	MPS / FP32	10,000	52.36	69.50
Laya-FT	Held-out	MPS / FP32	5,000	56.30	75.03

All systems complete 15,000 of 15,000 test predictions, with zero failed or missing records. Jev also completes 12 development-only technical requests: 15,012 HTTP attempts in total, with no retries or unresolved requests. Its recorded usage is 6,659,848 input and 480,384 output tokens. At the documented rate of USD 0.042 per million input tokens, with output tokens uncharged[1], the calculated API usage cost is USD 0.279713616, including the technical checks. This is a usage calculation, not an independently reconciled invoice. Complete training cost and local energy cost were not measured.

The FT test loops take 857.59 s in total, excluding loading, import checks and report generation. API latency includes network, connection, service and client overhead, but excludes pacing and queue waits; it is not pure model compute time. Moreover, the FT main test starts on battery and switches to AC, whereas the baseline runs on AC throughout. Background load is not controlled. Table 3 therefore describes these executions rather than an equal-power causal comparison of inference speed.

4 Discussion and limitations

The results support task-specific adaptation for this local model: training on the main collection substantially improves both aggregate test scores. However, the gain is smaller on the held-out collection, and the remaining gap to Jev is larger there. This pattern is consistent with sensitivity to collection differences, but it does not isolate a causal domain-shift effect; label conventions, review composition and unobserved provenance may also contribute. No claim of general equivalence to the API model is warranted.

Several limitations constrain interpretation. Source labels have not received independent human adjudication; star ratings can disagree with textual sentiment. Balanced test sets differ from natural review prevalence, while the training set is imbalanced. Only one training seed, two collections and one binary question are studied. Original platform-family independence and pretraining contamination cannot be excluded.

Approximate duplicate detection and the absence of user/product identifiers leave residual leakage risks. Finally, current provider behaviour and hardware conditions need not reproduce the recorded September 2026 timings. Additional baselines, repeated seeds and a blind human-labelled subset would strengthen a subsequent publication.

5 Conclusion

Under a frozen 15,000-review protocol, full fine-tuning moves Laya much closer to Jev on the main collection and improves its held-out-collection score. Jev retains the higher observed macro-F1 on both partitions. The defensible conclusion is a substantial, task-specific local-model improvement with remaining collection sensitivity, not universal parity or a controlled speed advantage.

Data, reproducibility and assistance

The source audit records a CC-BY-SA-4.0 card for MüşteriYorumları. For Turkish Product Reviews, the card’s **unknown** licence field is retained alongside its linked repository’s CC-BY-SA-4.0 text[6]; this discrepancy is not concealed. No raw review text or model weights are redistributed with this note. TRSAv1 and its derivatives are not used. The local study archive retains frozen splits, source revisions, task and model hashes, example-level predictions and execution metadata. The manuscript package supplies aggregate tables and a provenance manifest; it does not constitute a public release of the full benchmark. AI assistance was used for implementation support, analysis support and drafting. Generated labels were not used as ground truth. This research note has not undergone peer review.

References

- [1] TypeSafe AI. *Models: Jev 1.13.0*. Provider documentation. <https://docs.typesafe.ai/models>. Accessed 25 September 2026.
- [2] Convai Innovations. *Laya Multilingual*. Model card. <https://huggingface.co/convaiinnovations/laya-multilingual>. Evaluated revision e4e9ddf21a7b; accessed 25 September 2026.
- [3] D. Altınok / Turkish NLP Suite. *MüşteriYorumları: A Large Scale Customer Sentiment Analysis Dataset for Turkish*. Dataset card. <https://huggingface.co/datasets/turkish-nlp-suite/MusteriYorumlari>. Archived revision 7579c67929d3, inspected 24 September 2026.
- [4] F. Barmanbay. *Turkish Product Reviews*. Dataset card. https://huggingface.co/datasets/fthbrmby/turkish_product_reviews. Evaluated revision 3c8bef32a2b6; accessed 25 September 2026.
- [5] Convai Innovations / Laya contributors. *Laya: source code and fine-tuning implementation*. <https://github.com/NandhaKishorM/laya>. Version 0.3.20, archived commit 23a17522aa49, inspected 24 September 2026.
- [6] F. Barmanbay. *turkish-text-data: LICENCE*. CC-BY-SA-4.0 text in linked source repository, commit b0c2e38111f8. [Pinned repository licence](#). Archived copy inspected 24 September 2026.