Orthographic Structure Matters: Tokenization Failures in Arabic-Script and Related Languages

Manodnya K H
CLASIC,
University of Colorado Boulder
Boulder, CO, USA
manodynak@gmail.com

Luc De Nardi
CLASIC,
University of Colorado Boulder
Boulder, CO, USA
luc.denardi@colorado.edu

Abstract

Multilingual evaluation often relies on language coverage or translated benchmarks, implicitly assuming that subword tokenization behaves comparably across scripts. In mixed-script settings, this assumption breaks down. We examine this effect using polarity detection as a case study, comparing Orthographic Syllable Pair Encoding (OSPE) and Byte Pair Encoding (BPE) under identical architectures, data, and training conditions on SemEval Task 9, which spans Devanagari, Perso-Arabic, and Latin scripts. OSPE is applied to Hindi, Nepali, Urdu, and Arabic, while BPE is retained for English. We find that BPE systematically underestimates performance in abugida and abjad scripts, producing fragmented representations, unstable optimization, and drops of up to 27 macro-F1 points for Nepali, while English remains largely unaffected. Script-aware segmentation preserves orthographic structure, stabilizes training, and improves cross-language comparability without additional data or model scaling, highlighting tokenization as a latent but consequential evaluation decision in multilingual benchmarks. While the analysis spans multiple scripts, we place particular emphasis on Arabic and Perso-Arabic languages, where frequency-driven tokenization most severely disrupts orthographic and morphological structure.

1 Introduction

Polarity detection across mixed-script, low-resource languages is a deceptively difficult problem. The SemEval Task 9 dataset spans five writing systems—Devanagari (Hindi, Nepali), Perso-Arabic (Urdu, Arabic), and Latin (English)—and contains exactly the kind of messy, code-switched, short-form text where standard NLP assumptions tend to break down. In these environments, the downstream classifier is rarely the limiting factor. The real bottleneck is the input representation: how text is segmented, embedded, and contextualized before the model even begins to reason about sentiment.

Prior work on Orthographic Syllable Pair Encoding (OSPE) demonstrated advantages over BPE for Indic language modeling by aligning subword units with orthographic structure (Manodnya and Giri, 2023), but focused primarily on token-level behavior. It did not examine downstream task performance, multilingual interactions, or how segmentation choices behave when multiple writing systems are evaluated together. At the same time, most contemporary multilingual pipelines continue to treat BPE (Sennrich et al., 2016) as a default choice across languages—from English to Arabic to Nepali—despite accumulating evidence that it fragments morphology, destabilizes training, and produces noisy token sequences in abugida and abjad scripts (Kudo and Richardson, 2018).

In this work, we take a more engineering-grounded approach and evaluate tokenization inside a controlled end-to-end setting. We extend OSPE to Urdu and Arabic, retain standard BPE for English, and integrate these per-language subword inventories into a custom multilingual transformer trained from scratch. The architecture includes token, positional, syllable, and language embeddings; a stack of attention and long-convolution blocks tailored to short social-media inputs; and multi-task auxiliary heads (language modeling, next-token prediction, and syllable prediction) to regularize training in low-resource conditions.

This setup allows us to compare OSPE and BPE as part of a full evaluation pipeline rather than in isolation. Under identical model, data, and training conditions, the OSPE-based system achieves higher macro-F1 on SemEval Task 9, exhibits smoother optimization dynamics, and avoids the NaN-loss failures observed with multilingual BPE. The gains are most pronounced for low-resource and morphologically complex languages—most notably Nepali

(+26.7 F1)—while English performance remains largely unchanged. These patterns indicate that segmentation choices materially affect measured performance when scripts with different structural properties are evaluated together.

Taken together, our results suggest that multilingual polarity detection failures are often rooted in representation rather than classification. When segmentation disrupts script-level structure, downstream evaluation conflates modeling limitations with artifacts of the input pipeline. In such settings, tokenizer design becomes a first-order factor in how multilingual behavior is observed and compared, rather than a neutral preprocessing detail. In practice, this makes tokenizer design a first-order factor in the reliability and comparability of multilingual evaluation, especially when multiple scripts are assessed within a single benchmark. This work frames tokenization as an implicit evaluation decision that directly affects cross-language comparability in multilingual benchmarks.

2 Related Work

Subword tokenization methods such as Byte Pair Encoding (BPE) (Sennrich et al., 2016), WordPiece (Devlin et al., 2019), and SentencePiece (Kudo and Richardson, 2018) are widely used in multilingual NLP due to their simplicity and compatibility with neural architectures. These approaches are largely frequency-driven and operate over UTF-8 characters or bytes, an assumption that aligns well with alphabetic scripts but is less appropriate for writing systems where orthographic structure encodes linguistic information at the syllable level (Arivazhagan et al., 2019; Vania and Lopez, 2017).

Orthographic Syllable Pair Encoding (OSPE) was introduced to address this mismatch for Indic languages written in abugida scripts by treating orthographic syllables as the basic subword unit rather than raw characters or bytes. OSPE demonstrated improved compression ratios, reduced generation of invalid tokens, and more interpretable subword units in language modeling tasks. However, prior work evaluated OSPE primarily in monolingual or single-script settings and focused on intrinsic language modeling behavior, without examining downstream task performance, multilingual interactions, or cross-script evaluation scenarios.

Many multilingual systems rely on shared or jointly trained subword vocabularies across languages to encourage parameter sharing and transfer (Johnson et al., 2017). While effective for closely related or alphabetic languages, this design implicitly assumes that subword units behave comparably across scripts. Recent work on orthographic syllables for related Indic languages (Kunchukuttan and Bhattacharyya, 2016) and script conversion systems (Kunchukuttan et al., 2018) highlights that script structure plays a critical role in representation quality, but these insights have not been systematically incorporated into multilingual evaluation pipelines.

Sentiment and polarity detection have been extensively studied, ranging from early feature-based approaches (Pang et al., 2002; Turney, 2002) to neural and transformer-based models (Socher et al., 2013; Tang et al., 2016; Zhang et al., 2024; Wang et al., 2023). While these methods achieve strong performance in high-resource, single-script settings, relatively little work has examined how tokenization and representation choices affect evaluation outcomes in multilingual, mixed-script benchmarks. In short, segmentation is typically treated as a fixed preprocessing step rather than a variable that can materially influence measured performance.

Our work builds on OSPE by moving beyond intrinsic language modeling and examining tokenization inside a controlled, end-to-end multilingual evaluation setting. By extending orthographic syllable segmentation to Perso-Arabic scripts and evaluating it alongside standard BPE in a mixed-script polarity detection task, we isolate how segmentation choices affect optimization stability and downstream metrics. Unlike prior work that introduces new benchmarks or architectures, we focus on a latent but widely shared assumption in multilingual evaluation pipelines and quantify its impact using an existing benchmark.

3 Dataset

We evaluate our models on the SemEval-2026 Task 9 dataset, a multilingual polarity detection benchmark spanning five typologically diverse languages: Hindi (hi), Nepali (ne), Urdu (ur), Arabic (ar), and English (en). Each instance consists of a short social-media post labeled with one of three polarity classes (positive, negative, neutral). Scripts range from Devanagari to Perso- Arabic to Latin, making the task a natural stress-test for tokenization schemes. We use the official train/dev splits, and apply minimal preprocessing:

Table 1: Distribution of dataset size across languages

Lang	Script	Train Size	Dev Size
hi	Devanagari	2744	686
ne	Devanagari	2005	501
ur	Perso-Arabic	2849	712
ar	Perso-Arabic	3380	845
en	Latin	2676	667
Total	—	13654	3411

- Unicode normalization
- Removal of control characters
- •Preservation of emojis and punctuation (which carry polarity cues)

Table 1 summarizes the data distribution.

4 Methodology

Our approach combines (i) script-aware tokenization, and (ii) a custom multilingual transformer designed to evaluate segmentation effects under tightly controlled conditions. The goal is not to build a state-of-the-art sentiment model, but to isolate how tokenization interacts with multilingual sequence modeling (Rust et al., 2021).

4.1 Tokenization Strategy

We use two families of tokenizers:

4.1.1 OSPE for Devanagari and Perso-Arabic scripts

OSPE (Orthographic Syllable Pair Encoding) segments text according to script- specific orthographic syllables before applying frequency-based merges. We adapt OSPE to four languages:

- Hindi (hi), Nepali (ne): Devanagari consonant–vowel structure
- Urdu (ur): Urdu-specific extensions of Arabic script
- Arabic (ar): abjad with diacritic-bearing clusters

This requires only modifying the segmentation function; the merge algorithm remains identical to the original OSPE formulation.

4.2 BPE for English

English is tokenized using standard SentencePiece BPE, chosen because OSPE provides no structural advantage in alphabetic scripts and would behave identically to character-level BPE. This also provides a realistic multilingual comparison point: OSPE where it matters, BPE where it doesn't.

4.2.1 Per-Language Vocabularies and Offsets

Rather than training a single mixed vocabulary—which tends to overweight English and distort low-resource languages—we train separate tokenizers per language and assign each a fixed, non-overlapping vocabulary offset. Example:

- hi: 0–1999
- ne: 2000–3999
- ur: 4000–5999
- ar: 6000–7999
- en: 8000–9999

Every token is indexed as:
$[global\_id = lang\_offset[L] + local\_id]$
This preserves script identity and prevents cross-language token collisions, which stabilizes multilingual training without requiring shared subwords.

4.2.2 Multilingual Mini-Batching

We train using a stratified multilingual sampler. Each batch samples equally across languages, regardless of dataset size, preventing larger datasets (Arabic, Urdu) from dominating the learning dynamics. This is critical for isolating tokenizer effects rather than data imbalance effects.

4.3 Custom Transformer Architecture

To avoid coupling our results to any particular pretrained model, we implement a compact trans- former from scratch, designed specifically for short-form social-media inputs:

4.3.1 Embedding Layers

- Token embeddings (global vocab)
- Positional embeddings (up to 128 tokens)
- Syllable-type embeddings (6 OSPE-derived types)
- Language embeddings (one per language) The inclusion of syllable-type and language embeddings operationalizes OSPE's structural information inside the model.

4.4 Encoder Blocks

Each block contains:

1. Multi-head self-attention
2. Feed-forward network
3. Depthwise long convolution across the sequence
4. Residual + layer norm

The long-convolution branch improves handling of elongated vowel sequences, stacked diacritics, and repeated characters—phenomena more common in Indic and Arabic social media text than in English.

4.4.1 Multi-Task Heads

To stabilize multilingual learning, the model jointly predicts:

1. Polarity label (primary task)
2. Token reconstruction (LM)
3. Next-token prediction
4. Syllable-type prediction

These auxiliary tasks improve vocabulary utilization and reduce overfitting, especially for low-resource scripts.

4.5 Training Setup

Models are trained with a batch size of 32 and a maximum sequence length of 128 using the AdamW optimizer with a learning rate of

$$2 \times 10^{-4} \tag{1}$$

for 3 epochs. Training jointly optimizes the polarity classification objective along with auxiliary language modeling, next-token prediction, and syllable-type prediction losses, weighted at 1.0, 0.5, 0.5, and 0.3 respectively. All architectural choices, optimization settings, and loss weights are held constant across OSPE and BPE conditions to ensure that observed differences can be attributed solely to tokenization effects.

5 Experiments

Our experiments are designed to isolate the effect of tokenization on multilingual polarity detection. We hold the model architecture, training pipeline, and hyperparameters constant across all conditions, varying only the tokenizer subsystem.

5.1 Experimental Conditions

We evaluate three configurations:

5.1.1 OSPE-only model

- OSPE tokenizers for hi/ne/ur/ar
- BPE for en
- Custom transformer architecture described in §4
- Per-language vocabularies with fixed offsets

5.1.2 BPE-only baseline

- SentencePiece BPE trained separately per language
- Identical vocabulary sizes (3k each)
- Same architecture, embedding sizes, batches, and losses

5.1.3 Joint multilingual BPE

- A single BPE tokenizer trained on con- catenated multilingual text
- Same vocabulary budget (15k total)
- Included to verify whether shared sub- words help or hurt in mixed-script settings

The OSPE-only condition is the core system. The BPE-only and joint-BPE baselines serve as the primary ablations.

5.2 Evaluation Metric

We use macro-averaged F1, the official metric for SemEval Task 9, because:

- The classes are moderately imbalanced
- Macro-F1 penalizes models that underperform on low-resource languages
- We are specifically interested in how tokenization affects per-language behavior, not just overall accuracy

  We report both overall macro-F1 and per- language macro-F1.

5.2.1 Training Stability Diagnostics

Beyond accuracy, we track training dynamics:

- Loss curve smoothness
- Gradient explosion/NaN events • Token distribution entropy
- Effective sequence lengths
- Vocabulary coverage rates
- Merge collapse events (BPE) vs merge stability (OSPE)

These diagnostics help us understand why OSPE performs better, not just whether it does.

5.3 Implementation Details

All experiments use:

- PyTorch with MPS acceleration (Apple Silicon)
- Same random seeds for all tokenizer training and model training
- Stratified multilingual batching to ensure equal cross-language exposure
- No data augmentation, no transliteration, and no normalization beyond Unicode cleanup

OSPE, BPE, and model training code are implemented from scratch to ensure complete control over the pipeline and avoid hidden frameworks that may bias behavior.

5.4 Runtime and Computational Footprint

Training a full OSPE+BPE evaluation (OSPE model + two BPE baselines) requires:

- 1.5 – 2.2 hours on M1/M2-series hardware
- Peakmemory 6–8GB
- No GPU-specific kernels; all layers are standard PyTorch operations

The runtime budget is deliberately small: this work evaluates segmentation behavior without relying on large-scale pretraining or massive models.

6 Results

We report overall macro-F1 along with per- language performance for all three systems: OSPE-only, BPE-only, and joint multilingual BPE. All models share identical architecture and training settings.

6.1 Overall Performance

The OSPE model achieves the strongest macro-F1 across the board. Relative to per-language BPE baselines, OSPE offers consistent gains, especially in low-resource and morphologically complex scripts.

Table 2: model performance

Model	Macro-F1
OSPE (ours)	0.74
BPE-per-language	0.68
BPE-joint	0.63

These improvements hold under repeated runs with different random seeds.

6.2 Per-Language Performance

The benefit of OSPE correlates strongly with script complexity. English (Latin script) shows minimal difference, while Nepali, Urdu, and Arabic gain the most.

Table 3: Model comparison

Language	Script	BPE	Joint-BPE	OSPE	Δ (OSPE – BPE)
hi	Devanagari	0.70	0.66	0.75	+0.05
ne	Devanagari	0.56	0.50	0.83	+0.27
ur	Perso-Arabic	0.61	0.55	0.71	+0.10
ar	Perso-Arabic	0.64	0.59	0.72	+0.08
en	Latin	0.80	0.78	0.81	+0.01

- OSPE yields massive gains for Nepali (+26.7 F1), whose Devanagari morphology collapses under BPE.
- Urdu and Arabic gain +8–10 F1 due to OSPE's ability to treat consonant–diacritic clusters as atomic.
- Hindi benefits modestly (+5 F1), consistent with patterns reported in prior OSPE work.
- English shows no meaningful difference, validating the hybrid "OSPE where it matters, BPE where it doesn't" approach.

6.3 Training Stability

The OSPE model trains smoothly, while both BPE baselines show instability:

- Joint-BPE exhibits frequent loss spikes and occasional NaN collapses toward later epochs.

- Per-language BPE is more stable, but still shows higher variance and slower convergence.
- OSPE's token sequences converge to shorter, more consistent lengths, reducing gradient noise.

6.4 Qualitative Error Analysis

OSPE reduces three major error patterns:

1. Broken morphemes
   - BPE fragments core stems (e.g., Hindi verb endings, Arabic roots), confusing polarity cues.
   - OSPE produces subwords that map more cleanly to grammatical units.
2. Diacritic misalignment (Arabic/Urdu)
   - Joint-BPE treats diacritics as independent tokens, producing meaningless sequences.
   - OSPE attaches them correctly to base consonants.
3. Over-fragmentation in low-resource languages
   - Nepali suffers worst under BPE, producing character-level splintering.
   - OSPE preserves orthographic syllables, giving stable embeddings.

6.5 Ablation: Removing Auxiliary Tasks

Removing the LM/next-token/syllable auxiliary heads reduces OSPE's gains:

The auxiliary tasks help the model fully exploit OSPE's cleaner subword structure.

7 Discussion

The central result from our experiments is surprisingly simple: in mixed-script multilingual settings, a transformer's measured performance is disproportionately influenced by the tokenizer rather than the architecture. Even with identical model capacity, loss functions, and training regime, switching from BPE to OSPE changes training stability, effective sequence length, representation quality, and ultimately macro-F1—sometimes by more than 25 points. These effects emerge consistently across runs and point to three recurring themes.

7.1 Tokenization as Latent Structure Recovery

Scripts such as Devanagari, Urdu, and Arabic encode linguistic structure at the orthographic level. Consonant clusters, matras, diacritics, root–pattern morphology, and phonotactic groupings all carry semantic and syntactic cues. OSPE's segmentation aligns with these syllable-level units, whereas BPE operates over frequency-driven character sequences that often ignore script structure.

This difference becomes especially visible in polarity detection, where sentiment frequently depends on negation particles, intensifiers, cliticized forms, emphatic diacritics, and reduplicated syllables. When these units are fragmented during tokenization, the downstream model must reconstruct structure that is already present in the script, increasing variance and reducing robustness.

7.2 Cross-Script Multilinguality Magnifies Tokenizer Fragility

Multilingual BPE implicitly assumes that shared merges across scripts are neutral or beneficial. Our results suggest that this assumption does not hold in mixed-script, low-resource settings. Shared multilingual merges tend to overweight English due to raw frequency effects, dilute rare Devanagari and Perso-Arabic subwords, introduce degenerate merges between unrelated scripts, and disrupt morphological boundaries in low-resource languages.

By contrast, OSPE avoids these failure modes by using per-language vocabularies with explicit offsets, preserving script identity without requiring transliteration or shared subword inventories. In this sense, cross-script token sharing is not free: for some languages, it measurably degrades evaluation outcomes.

7.3 Auxiliary Tasks Amplify Structure-Preserving Segmentation

The auxiliary language modeling, next-token, and syllable-type prediction heads consistently strengthen representations learned from OSPE-based segmentation. Two effects are apparent. First, auxiliary objectives regularize learning, particularly in languages with limited labeled polarity data, leading to lower-variance training. Second, the syllable-type head provides lightweight morphological supervision, encouraging the model to internalize the structure exposed by OSPE.

BPE-based models benefit less from these objectives, likely because their token boundaries are less linguistically stable and less consistent across scripts.

7.4 Why English Is Largely Unaffected

English performance varies little across tokenization schemes, reinforcing a script-dependent interpretation of our results. Orthographic syllable modeling appears to matter most in abugida and abjad scripts, while alphabetic scripts do not exhibit the same sensitivity. This supports the hybrid strategy adopted in our experiments—OSPE where script structure is informative, and BPE where it is not—and cautions against uniform tokenization choices in multilingual evaluation pipelines.

7.5 Implications for Multilingual NLP Evaluation

Three practical implications follow from these observations. First, tokenizer choice can dominate measured performance: in our setting, segmentation alone accounts for large F1 differences without changes to architecture, data, or training regime. Second, script-aware segmentation reduces evaluation noise: treating tokenization as interchangeable across scripts can obscure or exaggerate differences between languages, particularly in low-resource settings. Third, inductive bias matters before scale: the structural bias introduced through segmentation and syllable embeddings plays a role often attributed to pretraining or model size.

From an evaluation perspective, these results suggest that multilingual benchmarks should explicitly document and justify tokenization choices, particularly when comparing languages across scripts and resource levels.

7.6 Tokenization as Part of the Model–Evaluation Interface

Tokenization functions as a learned interface between text and computation rather than a neutral preprocessing step. In multilingual systems, it determines effective sequence length, embedding density, contextual coverage, morphological fidelity, and even gradient behavior during training.

From an evaluation standpoint, this means that tokenizers shape what downstream metrics are able to capture. When segmentation fractures script-level structure, evaluation outcomes conflate modeling limitations with artifacts of the input pipeline. Structure-preserving tokenization reduces this distortion by exposing information the script already encodes. These effects are especially pronounced in Arabic-script languages, where consonant–diacritic groupings and root–pattern morphology are systematically fragmented by standard subword tokenizers.

8 Conclusion

This work shows that multilingual polarity detection outcomes are strongly shaped by tokenization choices. By extending OSPE beyond its original Indic scope and integrating it into a controlled multilingual transformer, we observe more stable training dynamics and substantially higher macro-F1 than BPE baselines, particularly for Nepali, Urdu, and Arabic.

These gains arise without architectural scaling, pretraining, or additional data. The only difference is how text is segmented. From an evaluation standpoint, this highlights tokenization as an architectural decision that directly influences measured performance. Script-aware segmentation methods such as OSPE provide a practical, computationally inexpensive way to reduce fragmentation-induced noise in multili ngual benchmarks.

9 Future Work

Several follow-up directions arise naturally from this work. A first step is pretraining a multilingual OSPE-based encoder on large-scale mixed-script corpora (e.g., 50–100B tokens) to evaluate how orthographic-syllable segmentation scales with model capacity and whether downstream gains compound under pretraining.

Beyond the scripts considered here, extending OSPE to additional writing systems such as Tamil, Sinhala, Burmese, and Amharic is promising, as these languages exhibit strong alignment between orthographic syllables and linguistic units. Mixed writing systems, including Japanese and Korean (kana–kanji–hangul hybrids), also present interesting segmentation challenges.

Another direction is to explore hybrid tokenization strategies that combine OSPE with unigram language-model–based tokenizers, potentially improving generalization to unseen or rare forms. Evaluating OSPE in cross-lingual transfer settings—such as named entity recognition, stance detection, machine translation, and code-switched question answering—would further clarify where structure-preserving segmentation matters most.

Finally, OSPE's reduction in effective sequence length (up to approximately 40% in some scripts) raises questions about hardware-level efficiency, including potential speedups on GPUs, TPUs, or memory-constrained devices. Model-agnostic OSPE adapters for existing multilingual LLMs (e.g., LLaMA, Mistral, Qwen) offer a practical path for comparing aligned versus fragmented segmentation without retraining models from scratch.

10 Limitations

This study isolates tokenization effects using a compact transformer trained from scratch, which allows clean attribution but leaves open how OSPE would interact with large-scale pretrained multilingual models. We train tokenizers separately per language with fixed vocabulary offsets, avoiding harmful cross-script merges but precluding cross-lingual subword sharing that may benefit closely related languages.

The evaluation spans five languages across three writing systems and focuses solely on polarity detection; results may differ for other scripts or tasks such as named entity recognition, question answering, or machine translation. Finally, we compare against BPE-based tokenization only and do not include unigram or SentencePiece-unigram variants, which may behave differently for some scripts.

References

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, and 1 others. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges. *arXiv preprint arXiv:1907.05019*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, and 1 others. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Anoop Kunchukuttan and Pushpak Bhattacharyya. 2016. Faster decoding for subword level phrase-based smt between related languages. *arXiv preprint arXiv:1611.00354*.

Anoop Kunchukuttan, Mitesh M Khapra, Gurneet Singh, and Pushpak Bhattacharyya. 2018. Leveraging orthographic similarity for multilingual neural transliteration. *Transactions of the Association for Computational Linguistics*, 6:303–316.

KH Manodnya and Animesh Giri. 2023. Orthographic syllable pair encoding for language modelling tasks in indic languages. In *2023 IEEE MIT Undergraduate Research Technology Conference (URTC)*, pages 1–6. IEEE.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. *arXiv preprint cs/0205070*.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 1715–1725.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016, the 26th international conference on computational linguistics: technical papers*, pages 3298–3307.

Peter Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 417–424.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? *arXiv preprint arXiv:1704.08352*.

Zhiqiang Wang, Yiran Pang, and Yanbin Lin. 2023. Large language models are zero-shot text classifiers. *arXiv preprint arXiv:2312.01044*.

Wenxuan Zhang, Yue Deng, Bing Liu, Sinno Pan, and Lidong Bing. 2024. Sentiment analysis in the era of large language models: A reality check. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3881–3906.