Benchmarking Large Language Models for Detecting Satirical Code-Mixed Sentences

Anonymous ACL submission

Abstract

Code-switching is a natural linguistic phenomenon, yet its impact on high-level reasoning tasks remains largely unexplored in under-resourced languages like Romanian. Natural language processing tasks, such as satire detection, rely heavily on nuanced cultural and syntactic cues that large language models struggle to capture. In this paper, we investigate four methods (i.e., random replacement, phrase translation, dependency parsing, and large language models) for generating synthetic code-mixed Romanian sentences with English, German, French, and Italian. We start with an existing dataset for satire detection, compiled from Romanian news articles and annotated at the sentence level by humans, and augment it with over 600k samples. We provide an in-depth analysis of the dataset generation process. Then, we evaluate several multilingual large language models and show that the best model, GPT-5-mini with medium reasoning effort, achieves 70% accuracy and F1-score in the zero-shot setting, while the best open-source model, RoGemma 2 9B, achieves approximately 55%, showing the limitations of open-source models on this task.

1 Introduction

Code-switching or code-mixing is a common phenomenon in multilingual communities in which speakers alternate between two or more languages within the same sentence (Guzmán et al., 2017). While early natural language processing (NLP) and linguistic studies primarily mapped constraints on widely spoken pairs, the structural behavior of code-switching shifts significantly when applied to morphologically rich languages (Myers-Scotton, 1993).

Mixing Romanian with other European languages is a naturally occurring phenomenon (Wilhelm, 2017; Buzilă, 2025). In the diaspora, many Romanians live, especially in countries such as Germany, France, Italy, Spain, and the UK, making it one of the largest diasporas (OECD, 2019). Therefore, learning these languages is very common. Moreover, in Romania, German communities lived in enclaves (particularly in Transylvania), although the number of speakers has declined considerably over the last century. The French influence, however, was significant at the beginning of the 19th century, as many words entered the language from French. Although this is not considered code-mixing (since some words lacked Romanian equivalents), it suggests that historical multilingualism was common, creating an environment in which foreign terms could be readily adopted (Boia, 2001). Studies on Romanian-English bilingual speech (Bancu, 2013; Greavu, 2022) indicate that the structural integrity of Romanian grammar is strictly maintained as the matrix language, thereby forcing embedded foreign elements to adapt to its complex inflectional and syntactic rules.

Code-switching was previously explored in the context of satire, irony, and humor detection, but primarily in other combinations, such as Hindi-English or Spanish-English. For example, Bansal et al. (2020) investigated Hindi-English code-mixed tweets for detecting humor, sarcasm, and hate speech, arguing that code-mixing can be used to emphasize these phenomena. They also show that it can improve the model's performance on downstream tasks. Belani and Flanigan (2023) proposed a dataset generated from transcribing day-by-day conversations for Spanish-English with various labels, including usage of code-mixing for jokes. Pant and Dadu (2020) proposed mapping the code-switched text to a higher-resource language through transliteration and showing improved performance for sarcasm and hate speech detection. However, this remains an underexplored area due to data scarcity. Consequently, many studies (including ours) rely on synthetic generation. This practice has positively spurred the community to develop

diverse generation techniques, even though most evaluations continue to focus on downstream tasks using existing, naturally code-mixed text (Aguilar et al., 2020; Sheth et al., 2025).

These socio-linguistic factors and data limitations motivate us to study four European languages, namely, English, German, French, and Italian, as target languages for code-switching with Romanian.

In this work, we create the first Romanian code-mixed satirical news dataset, namely CS-SeLeRoSa, using four different code-switching methods and four language pairs (English, German, French, and Italian). We provide an extensive analysis of the generated dataset using various metrics and benchmark several LLMs on the code-mixed dataset.

We summarize the main contributions of this work as follows:

- We generate the first code-mixed Romanian satirical news dataset using four different methods and four language pairs.
- We provide an extensive analysis of the generated dataset using various metrics.
- We release a small gold dataset annotated and verified by human experts. In addition, we thoroughly analyze the effectiveness of the generation methods across all four code-switching settings.
- We benchmark several LLMs on both the large-scale and gold synthetic code-mixed dataset.

2 Related Work

Code-Switching Tasks. Code-switching was evaluated in various natural language processing tasks. For instance, Krishnan et al. (2021) addresses slot filling and intent prediction. It takes the set of text in the original language and, for each item, generates random translations for chunks, then performs label-level alignments. A similar approach was applied by Tarunesh et al. (2021) for the Hindi language. In addition, they proposed a machine translation system which uses an encoder-decoder architecture trained using multiple losses: denoising autoencoding, backtranslation, and cross-entropy. Sterner and Teufel (2025b) investigated the alignment in choosing natural code-switched sentences over manipulated variants and showed that larger models tend to align better with the preferences of fluent bilingual speakers, who understand the social and syntactic nuances of code-switching. In a follow-up work, Sterner and Teufel (2025a) analyzed the syntactic difference between naturally occurring code-switched sentences and manipulated ones. They replaced the words with their parts-of-speech tags and, from the resulting structure, distinguished between sentences in minimal code-switching pairs. They proposed a graph neural network-based approach that achieves scores comparable to those of large language models across various language pairs.

Previously, code-switching was evaluated alongside the detection of satire, irony, and sarcasm (Bansal et al., 2020), specifically for Hindi-English (also known as Hinglish). Shah and Maurya (2021) detected sarcasm from code-mixed social media posts, incongruity through sub-word embeddings in a bidirectional LSTM-based architecture (Hochreiter and Schmidhuber, 1997). Ghanem et al. (2020) proposed the first multilingual and multicultural detection system for irony, in French, English, and Arabic. They evaluated feature-based models, such as random forests, and neural network models, such as a convolutional neural network, noting challenges in the dataset, such as informal language use and Arabic-specific challenges. Recently, more work has begun to address code-mixing in Arabic (Hamed et al., 2025).

Code-Switched Datasets. Pratapa et al. (2018) proposed to investigate the role of the equivalence constraint theory from linguistics to generate code-mixed text. Their approach suggested that using curriculum learning on generated samples can reduce the perplexity of language models. Salaam et al. (2022) created a real-world dataset and generated 30k synthetic sentences for code-mixing, for English to French, Spanish, and German. Multiple languages have been evaluated for code-mixing, and one of the most common combinations is Hindi-English (Doğruöz et al., 2021). Large language models (LLMs) have been utilized in various code-mixed tasks (Zhang et al., 2025; Lee et al., 2024). Recently, Xie et al. (2026) proposed SwitchLingua, the first large-scale multilingual and multi-ethnic code-switching dataset, consisting of 12 languages and over 80 hours of audio. However, it does not include Romanian. They also proposed LinguaMaster, a data synthesis method based on an agentic framework.

3 Method

3.1 Dataset

To generate the synthetic code-mixed dataset, we employed the SeLeRoSa dataset (Smădu et al., 2025), which contains approx. 13.8k manually annotated sentences from Romanian satirical and non-satirical news outlets. The dataset is anonymized, and the official train/validation/test split is 9,800/2,000/2,073. We follow the same split in our experiments. The resulting large-scale dataset is called CS-SeLeRoSa. In addition, we also provide a smaller human-annotated subset of this dataset, called CS-SeLeRoSa-Gold, meant for testing only.

3.2 Translation

To generate the synthetic data, we investigate several Large Language Models (LLMs) for machine translation. Translation is a key component in generating the synthetic code-mixed dataset. The generation of code-switched sentences involves considering the sentence in the source language (i.e., the matrix language is Romanian), and applying a code-switching method that requires accessing a translation system to translate words or phrases to the target language. Then, the translated words or phrases are reintegrated into the original sentence (according to the code-switching method) to form a code-mixed sentence. Note that not all methods (e.g., end-to-end LLM generation) require a separate translation step. For our code-mixing generation setup, we translate the text with Gemma-3 27B using the machine translation prompt in Appendix F. An evaluation of multiple candidate models for translation is provided in §3.3.

3.3 Translation Evaluation

We include the translation evaluation because it is important in the generation pipeline, and we could not find machine translation performance assessments for these models. Therefore, we need to select a suitable machine translation model for the chosen languages. Specifically, we evaluate Gemma 3 (the 27B model, which is the larges one, and 12b in which the intent was if a smaller model performs reasonably well but faster we would select this model) (Kamath et al., 2025) which was trained on over 140 languages and OpenAI's GPT-5-mini (OpenAI, 2025) which is a state-of-the-art proprietary model. We use the FLORES+ dataset[1] (NLLB Team et al., 2024) to assess the models' performance; it contains high-quality translations for over 200 languages. We select Romanian as the source language and select two Germanic languages (i.e., English and German) and two Latin languages (i.e., French and Italian) as target languages. The temperature was set to 0.2 for Gemma 3 models to increase determinism, and left at the default for GPT-5-mini. In addition, we set the reasoning effort to low for OpenAI's model as a reasonable trade-off between performance and cost.

In Table 1, we report the BLUE score (Papineni et al., 2002), character n-gram F-score (chrF) (Popović, 2015), Translation Edit Rate (TER) (Snover et al., 2006), and COMMET (Rei et al., 2020) for each model and language pair. Gemma-3 27B outperforms the other models on all language pairs and metrics. The best scores are obtained for English and French, which are high-resource languages, and more translation pairs can be available. Considering these results, we select Gemma-3 27B as the translation model for generating the code-mixed dataset.

Model	BLEU ↑	CHRF ↑	TER ↓	COMET ↑
		RO → EN		
gemma-3-12b	41.04	67.30	46.23	0.89
gemma-3-27b	**42.80**	**69.37**	**43.31**	**0.90**
gpt-5-mini	40.07	67.60	46.13	0.89
		RO → FR		
gemma-3-12b	28.67	54.76	62.80	0.82
gemma-3-27b	**38.78**	**64.72**	**50.53**	**0.87**
gpt-5-mini	33.84	61.31	55.53	0.85
		RO → DE		
gemma-3-12b	23.61	52.45	70.19	0.82
gemma-3-27b	**30.42**	**60.51**	**59.59**	**0.87**
gpt-5-mini	26.41	57.72	64.28	0.85
		RO → IT		
gemma-3-12b	21.82	52.73	68.53	0.86
gemma-3-27b	**27.11**	**57.36**	**62.22**	**0.89**
gpt-5-mini	23.82	54.94	66.06	0.87

Table 1: Translation performance across models and language pairs (RO-X) on the FLORES+ dataset.

3.4 Synthetic Code-Mix Generation

Random Replacement. Introduced by Krishnan et al. (2021), the random replacement approach performs chunk-level code-switching. For each input sentence, we split it into tokens at word boundaries and then randomly select a subset of words

[1] https://huggingface.co/datasets/openlanguagedata/flores_plus

to replace with their translations. The number of words selected is given by a factor r which in our experiments was set to 50%. As opposed to the original implementation, we use an LLM-based machine translation as described in §3.2, and ignore the anonymization tags during sampling. We acknowledge that the random-replacement strategy naturally yields degraded outputs. However, we applied the method precisely as proposed by Krishnan et al. (2021) without additional filtering to maintain consistency with prior work.

Phrase Translation. It (Salaam et al., 2022) is a three-stage approach: phrase identification, phrase translation, and phrase reintegration. For phrase identification, as opposed to the original implementation, we employ multi-RAKE (Rose et al., 2010), which supports the languages we investigate. In the phrase translation step, we use the LLM-based machine translation system from §3.2 to translate the identified phrases. Finally, in the phrase reintegration step, we replace the original phrases with their translations. To increase diversity in code-mixed sentences, we randomly select which phrases to replace using uniform sampling. The motivation for this approach is that the chosen groups of words should make sense on their own to be translated coherently.

To extract phrases from the text, we use the `vgrabovetsmulti_rake` library from GitHub[2]. According to the documentation and Rose et al. (2010), the algorithm is not limited to keywords (i.e., single words); it can also return multi-word keyphrases. Note that the algorithm does not make a difference between the specific kinds of keyphrases (e.g., noun-phrase, verb-phrase, etc). But in our case, this distinction is not relevant. Instead, it is relevant that the phrase must be self-contained (i.e., as a standalone, it should be complete and have a meaning). RAKE does that by removing stop words and punctuation. The work proposing the phrase-based code-switching approach (Salaam et al., 2022) used the UCPhrase unsupervised model (Gu et al., 2021), which was trained on English data. However, this approach does not support other languages out of the box. The `multi_rake` package already supports Romanian and the other target languages, so we opted for this solution instead.

Dependency Trees. Gregorius and Okadome (2022) proposed the generation of code-switching text using a machine translation system and a syntactic dependency parser. This approach does not require training models on parallel corpora; its components are plug-and-play and already available. The idea relies on finding a code-switching point by leveraging the properties of the dependency graph and then using the machine translation system from §3.2 to translate the phrase spanned by the sub-tree to the other language. This takes the idea from the previous method to the next level by considering sub-trees that are large enough to be self-contained, providing more context for the translator. However, it is therefore limited by the quality of the dependency parser and the machine translation system. For the implementation, we rely on the Spacy[3] dependency parser for Romanian.

Large Language Models. Large Language Models (LLMs) have demonstrated impressive capabilities in various natural language processing tasks, including text generation, translation, and code-mixing (Yang and Chai, 2025). We explore the use of LLMs to generate code-mixed sentences directly in the zero-shot setting. Therefore, we prompt the model (i.e., Gemma-3 27B) to generate a code-mixed version in the given languages, preserving the original meaning (see Appendix F). Note that in this case, the generation is performed end-to-end by the LLM, and therefore the translation does not follow §3.2.

3.5 Dataset Statistics

In Table 2, we summarize the statistics of the generated code-mixed dataset CS-SeLeRoSa. We used 4 language pairs, i.e., Romanian (RO) to English (EN), German (DE), French (FR), and Italian (IT), and 4 code-mixing methods described in §3.4. For each language pair and method, we generated at most $K = 3$ code-mixed examples (with IDs from 1 to 3, 0 indicating the original sentence). In total, we generated approx 220k samples per iteration. Per method, we generate approx 165k samples. In the end, we maintain the original dataset split into train, validation, and test sets, by the original source sentence. Statistics about the human-annotated data are presented in Appendix D.4.

3.6 Code-Mixing Evaluation

Automatic Evaluation. Due to the large size of the synthetic dataset, it is challenging to assess its qual-

[2] https://github.com/vgrabovets/multi_rake

[3] https://spacy.io/

Language				
RO	DE	EN	FR	IT
13,873	165,725	165,725	165,725	165,803
K (code-switch variation)				
k=0	k=1	k=2	k=3	
13,873	221,956	221,716	219,306	
Generation Method				
Original	Random	Phrase	Dependency	LLM
13,873	166,476	165,934	164,092	166,476
Split				
Train	Test	Validation		
478,592	100,975	97,284		

Table 2: Summary of the entire CS-SeLeRoSa code-mixed dataset statistics.

Method	CMI	M-Index	I-Index	LE	SP-Avg
		RO → EN			
Random	39.29	0.44	0.47	0.94	0.46
Phrase	15.93	0.18	0.22	0.55	0.21
Dependency	18.75	0.21	0.16	0.57	0.16
LLM	18.64	0.21	0.19	0.62	0.21
		RO → FR			
Random	40.12	0.44	0.47	0.94	0.46
Phrase	20.52	0.24	0.27	0.65	0.26
LLM	22.40	0.26	0.24	0.71	0.25
Dependency	21.61	0.25	0.20	0.64	0.21
		RO → DE			
Random	36.34	0.41	0.45	0.91	0.44
Method **Phrase**	26.66	0.31	0.31	0.76	0.30
LLM	21.17	0.24	0.24	0.69	0.25
Dependency	25.05	0.29	0.22	0.73	0.23
		RO → IT			
Random	40.93	0.43	0.49	1.00	0.48
Phrase	22.48	0.25	0.30	0.72	0.29
LLM	25.55	0.28	0.27	0.80	0.29
Dependency	23.54	0.26	0.23	0.72	0.23

Table 3: Aggregated code-switching metrics by language and code-switching method.

ity manually. Therefore, we employ several automated metrics that capture diversity, semantic similarity, and readability (Srivastava and Singh, 2021; Guzmán et al., 2017). Code-mixing Index (CMI) (Das and Gambäck, 2014) indicates the degree of code-mixing in a text, considering the number of words from each language. The Multilingual Index (M-Index) (Barnett et al., 2000) measures the language distribution from the text, where 1 indicates an equal distribution of languages. The Integration Index (I-Index) (Guzmán et al., 2017) quantifies the probability of language switching in the text. The Language Entropy (LE) (Guzmán et al., 2017) computes the number of bits required to encode the language distribution from the text. Finally, the Switch Point Average (SP-Avg) (Jayanthi et al., 2021) measures the probability of the language being switched in the text. To compute these metrics, we use the CodeMixedNLP toolkit[4] (Jayanthi et al., 2021). For token-level language identification, we employ a series of heuristics based on language-specific special characters. If the word cannot be identified in this way, we use the lingua toolkit[5], which supports the languages evaluated. Table 3 shows that using the random replacement method achieves much more frequent switching between languages, while the other methods are more balanced.

In addition, we measure the dissimilarity between generations at the token level (i.e., derived from the Jaccard index) and the semantic level (i.e., derived from BERTScore (Zhang et al., 2020)).

Given two sentences S_1 and S_2 provided as a list of tokens, the Jaccard similarity is computed as:

$$J(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} \quad (1)$$

We define the Jaccard dissimilarity as $1 - J(S_1, S_2)$. On the other hand, BERTScore measures the semantic similarity between two texts based on the pairwise cosine similarity of their embeddings. We measure the dissimilarity as $1 - \texttt{BERTScore}(S_1, S_2)$. We employ the multilingual XLM-RoBERTa-large pretrained model (Conneau et al., 2020) to embed the sentences. In Table 4, we report the average scores between all pairs of generated sentences for each method and language pair. As observed before, the random replacement method generates more dissimilar sentences due to the lack of context when translating individual words. Additionally, the LLM-based code-mixing produces slightly more dissimilar sentences than the other two methods, which could be due to the model's conditioning to generate diverse outputs (see Appendix F).

Manual Evaluation. We select a subset of sentences from the generated dataset for human evaluation. Specifically, we randomly sample 50 non-code-mixed sentences and extract the corresponding generated code-mixed sentences for each of the four methods and four languages, resulting in a total of 715 samples. These samples were then filtered to remove instances of unnatural code-

[4] https://github.com/murali1996/CodemixedNLP
[5] https://github.com/pemistahl/lingua-py

Method	BERTScore			Jaccard		
	Mean	Std	Median	Mean	Std	Median
			RO → EN			
Random	8.03	1.91	7.90	73.28	7.85	72.77
Phrase	3.33	1.51	3.37	29.80	13.23	30.68
Dependency	3.31	0.95	3.22	37.37	6.53	37.89
LLM	3.61	2.05	3.08	39.06	19.66	34.41
			RO → FR			
Random	8.11	1.92	7.94	67.73	9.01	67.68
Phrase	3.57	1.65	3.59	28.35	13.02	29.28
Dependency	3.66	1.13	3.67	34.57	8.05	35.70
LLM	3.87	1.97	3.37	36.83	16.79	33.88
			RO → DE			
Random	10.24	1.87	10.18	73.61	7.62	73.12
Phrase	4.22	1.86	4.27	30.30	13.65	30.83
Dependency	4.23	1.07	4.26	36.67	7.25	37.46
LLM	3.49	1.92	3.03	28.35	16.80	24.34
			RO → IT			
Random	7.86	1.80	7.75	68.77	8.91	68.76
Phrase	3.58	1.61	3.65	28.37	13.07	28.70
Dependency	3.57	1.01	3.54	35.11	7.41	35.86
LLM	4.04	1.88	3.61	38.03	17.72	34.81

Table 4: Semantic distance and Jaccard dissimilarity for code-switched sentences by language and method.

switching or significant grammatical errors, resulting in a final set of 391 high-quality code-mixed sentences. We call this subset CS-SeLeRoSa-Gold, and, due to its data selection strategy, is meant only for model evaluation. More details on the annotation protocol and human evaluation statistics are provided in Appendix D.

4 Experimental Setup

4.1 Models

On the CS-SeLeRoSa test set, we evaluate several LLMs in the zero-shot setting, with and without chain-of-thought prompting (Wei et al., 2022), as well as fine-tuning. We include open-source and proprietary models in the evaluation. Specifically, for open-source models, we evaluate Gemma 3 27B (Kamath et al., 2025) and Llama 3.1 8B (Grattafiori et al., 2024), and Romanian LLMs (Masala et al., 2024) such as RoMistral 7B, RoLlama 3.1 8B, and RoGemma 2 9B. For the closed-source models, we employ OpenAI's models, such as GPT-4.1 mini and the GPT-5-mini reasoning model. For GPT-5-mini, we do not include chain-of-thought prompting, as the model already generates a reasoning trace before responding.

4.2 Hyperparameters

Inference. The evaluated models (excluding the sample generation stage) use the following hyperparameters during inference. The temperatures were set to 1.0 for the Gemma-3 model, 0.8 for the other open-source models, and to the default for the proprietary models. The maximum number of tokens was set to 2048 for all models. The reasoning effort was set too low for the proprietary models in the translation setting and medium in the satirical code-switching classification setting. The number of runs per prompt was set to 3 for the open-source models and to 1 for the proprietary models (due to budget constraints). We used the unquantized versions of the models.

Fine-tuning. We also include fine-tuning results of the open-source models. We use an efficient 4-bit quantization with QLoRA (Dettmers et al., 2023) to fine-tune the open-source models. The models were fine-tuned for 3 epochs, with a linear learning-rate scheduler starting at 1e-4 with a 10% warm-up, a batch size of 32, and a context length of 1024 tokens. We used LoRA rank of 16, $\alpha = 32$, and dropout of 0.05 for all attention layers (`up_proj`, `down_proj`, `gate_proj`, `k_proj`, `q_proj`, `v_proj`, and `o_proj`). For optimization, we used Adam (Kingma and Ba, 2015). The remaining parameters are used by the HuggingFace Trainer by default. We did not fine-tune the proprietary models due to budget constraints.

4.3 Evaluation Metrics

For the satirical code-switching classification, we report accuracy (Acc.) and F1-score (F1). For open-source models, we report the average and standard deviation over 3 inference runs with different random seeds. For proprietary models, we report results on a single run.

5 Results

Overall Results. We present the results in Table 5 on the entire test set, and Table 6 on the gold standard dataset. Overall, the proprietary models outperform the open-source models by 15% in both accuracy and F1-scores. The reasoning model GPT-5-mini achieves 70% on both metrics. On the other hand, the best performing open-source model overall is RoGemma 2 9B, which was specifically pretrained on the Romanian language. In addition, we observe that Romanian-trained models outperform their general counterparts.

Model	Standard		Chain-of-Thought		Fine-Tuning	
	Acc (%)	F1 (%)	Acc (%)	F1 (%)	Acc (%)	F1 (%)
Gemma 3 27B	45.09 ±0.13 †	33.21 ±0.27 †	47.47 ±0.82 †	38.12 ±1.70 †	**77.10** ±0.36	**76.74** ±0.61
Llama 3.1 8B	42.02 ±15.89	45.42 ±11.13	47.90 ±2.11 †	46.18 ±2.03 †	19.74 ±8.66	30.27 ±11.44
RoGemma 2 9B	55.73 ±0.44	54.22 ±0.62	59.98 ±0.66	59.81 ±0.85	76.16 ±0.37	76.25 ±0.26
RoLlama 3.1 8B	53.87 ±0.30 †	51.08 ±0.79 †	48.52 ±0.65 †	43.74 ±0.98 †	76.24 ±0.07	76.22 ±0.26
RoMistral 7B	35.76 ±12.40	38.13 ±10.10	46.31 ±0.60 †	47.17 ±1.07 †	17.34 ±10.66	26.69 ±13.22
GPT-4.1 mini	57.09 †	54.95 †	**62.11** †	**61.64** †	-	-
GPT-5 mini	**70.29** †	**70.45** †	-	-	-	-

Table 5: Results on the code-switched dataset. We present the results with and without chain-of-thought (CoT) prompting, as well as the fine-tuning setting. Bold indicates the best performance. † indicates statistically significant vs the best result RoGemma 2 9B ($p < 0.05$, Welch's t-test for N>1, McNemar's test for N=1).

Model	Standard		Chain-of-Thought		Fine-Tuning	
	Acc (%)	F1 (%)	Acc (%)	F1 (%)	Acc (%)	F1 (%)
Gemma 3 27B	41.94 ± 0.55	31.14 ± 0.78 †	43.31 ± 1.93	35.01 ± 3.02 †	**75.45** ± 0.42	**74.71** ± 0.42
Llama 3.1 8B	40.24 ± 14.04	43.77 ± 9.51	42.97 ± 1.91	41.02 ± 2.42 †	15.26 ± 8.59	23.41 ± 11.98
RoGemma 2 9B	56.86 ± 1.21	56.47 ± 1.25	**62.06** ± 0.84	**62.90** ± 0.93	68.97 ± 1.54	69.08 ± 1.21
RoLlama 3.1 8B	54.39 ± 0.60	53.71 ± 1.36	47.66 ± 1.89	44.87 ± 2.29 †	69.82 ± 1.66	69.83 ± 1.13
RoMistral 7B	34.95 ± 11.43	37.75 ± 9.04	44.25 ± 1.51	45.58 ± 1.33 †	14.24 ± 8.92	22.40 ± 11.61
GPT-4.1 mini	50.38	47.96 †	59.85	60.20	-	-
GPT-5 mini	63.17	63.66 †	-	-	-	-

Table 6: Overall results on the gold standard dataset, with and without chain-of-thought (CoT) prompting, as well as the fine-tuning setting. Bold indicates the best performance. † indicates statistically significant vs the best result RoGemma 2 9B ($p < 0.05$, Welch's t-test for N>1, McNemar's test for N=1).

Model	w/o CoT		w/ CoT	
	Acc (%)	F1 (%)	Acc (%)	F1 (%)
Gemma 3 27B	52.97±0.35	47.56±0.60	55.48±0.18	51.23±0.40
Llama 3.1 8B	43.91±12.19	47.46±7.97	52.16±1.89	52.21±1.66
RoGemma 2 9B	60.38±0.23	60.02±0.30	62.49±1.19	62.67±1.25
RoLlama 3.1 8B	60.97±0.25	61.30±0.75	51.92±1.22	48.60±1.94
RoMistral 7B	39.73±11.56	42.35±8.13	48.61±0.13	48.36±0.76
GPT-4.1 mini	63.39	62.98	**67.97**	**68.08**
GPT-5 mini	**71.88**	**72.09**	-	-

Table 7: Results on the baseline (non-code-switched) dataset. We present the results with and without chain-of-thought (CoT) prompting.

When evaluating only on the gold set, the discrepancy between open-source and proprietary models remains diminished. GPT-5-mini remains the best-performing model, achieving around 63-64%. However, the best performing open-source model is RoGemma 2 9B with approx. 62-63% across both metrics. We note similar performance differences between base LLMs and their Romanian pre-trained counterparts.

Compared with the baseline results (see Table 7 for the results on the non-code-switched subset of the dataset), we observe a degradation in performance of at most 10%, which emphasizes the current limitations of the current models that are not as resilient in the code-mixed settings as in the monolingual setup.

Detailed Analysis. To better quantify the performance differences across target languages and code-switched generation method, in Table 8, we present the detailed results for all models and language pairs, including the standard deviations for open-source models. We also include the fine-tuned results for open-source models, which show significant improvements over the zero-shot and chain-of-thought results.

We note that the random-replacement strategy naturally yields degraded outputs. However, we applied the method exactly as proposed by Krishnan et al. (2021), without additional filtering, to maintain consistency with prior work. In Table 8,

Model	Random			Dependency			Phrase			LLM		
	ZS	CoT	FT	ZS	CoT	FT	ZS	CoT	FT	ZS	CoT	FT
						ro → en						
Gemma 3 27B	$25.0_{\pm 0.1}$	$25.9_{\pm 0.9}$	$75.2_{\pm 0.9}$	$37.3_{\pm 0.6}$	$41.9_{\pm 2.4}$	$\mathbf{77.6}_{\pm 0.6}$	$34.7_{\pm 0.2}$	$37.8_{\pm 1.5}$	$74.3_{\pm 1.0}$	$28.7_{\pm 0.4}$	$36.3_{\pm 3.7}$	$\mathbf{77.3}_{\pm 0.7}$
Llama 3.1 8B	$44.5_{\pm 12.4}$	$41.6_{\pm 3.7}$	$58.4_{\pm 22.1}$	$48.4_{\pm 9.7}$	$51.1_{\pm 1.8}$	$26.6_{\pm 36.4}$	$46.0_{\pm 11.6}$	$48.1_{\pm 1.9}$	$27.9_{\pm 33.7}$	$45.8_{\pm 10.9}$	$48.0_{\pm 2.5}$	$24.5_{\pm 13.0}$
RoGemma 2 9B	$39.5_{\pm 0.7}$	$39.3_{\pm 0.5}$	$\mathbf{76.3}_{\pm 0.3}$	$56.0_{\pm 0.5}$	$59.9_{\pm 0.9}$	$77.4_{\pm 0.3}$	$52.6_{\pm 0.5}$	$54.9_{\pm 0.9}$	$75.2_{\pm 0.1}$	$53.0_{\pm 0.4}$	$57.9_{\pm 1.4}$	$76.0_{\pm 0.8}$
RoLlama 3.1 8B	$40.9_{\pm 1.1}$	$38.6_{\pm 0.4}$	$75.8_{\pm 0.5}$	$57.9_{\pm 0.6}$	$46.3_{\pm 1.9}$	$77.1_{\pm 0.1}$	$54.6_{\pm 0.6}$	$43.2_{\pm 1.1}$	$\mathbf{75.7}_{\pm 0.8}$	$54.9_{\pm 0.5}$	$43.8_{\pm 1.4}$	$75.8_{\pm 0.5}$
RoMistral 7B	$37.3_{\pm 10.3}$	$43.3_{\pm 2.2}$	$4.3_{\pm 6.1}$	$42.5_{\pm 8.9}$	$49.1_{\pm 0.7}$	$28.1_{\pm 23.8}$	$39.9_{\pm 10.1}$	$46.0_{\pm 1.4}$	$12.4_{\pm 10.8}$	$42.0_{\pm 9.5}$	$47.3_{\pm 1.4}$	$11.4_{\pm 8.1}$
GPT-4.1 mini	42.1	48.9	-	61.8	64.3	-	55.7	60.2	-	57.3	62.9	-
GPT-5 mini	66.6	-	-	71.4	-	-	69.9	-	-	71.5	-	-
						ro → de						
Gemma 3 27B	$25.0_{\pm 0.1}$	$28.3_{\pm 0.8}$	$76.3_{\pm 0.7}$	$37.2_{\pm 0.4}$	$41.3_{\pm 0.8}$	$\mathbf{77.4}_{\pm 0.9}$	$35.9_{\pm 0.3}$	$39.3_{\pm 1.1}$	$75.1_{\pm 1.0}$	$28.8_{\pm 0.3}$	$34.8_{\pm 0.7}$	$\mathbf{77.5}_{\pm 0.2}$
Llama 3.1 8B	$41.2_{\pm 13.3}$	$40.1_{\pm 2.5}$	$5.9_{\pm 5.4}$	$45.8_{\pm 10.9}$	$47.6_{\pm 2.4}$	$32.2_{\pm 31.8}$	$43.4_{\pm 12.9}$	$46.1_{\pm 2.0}$	$12.0_{\pm 2.3}$	$42.6_{\pm 11.3}$	$45.3_{\pm 1.8}$	$7.0_{\pm 1.6}$
RoGemma 2 9B	$47.2_{\pm 0.7}$	$56.1_{\pm 1.3}$	$\mathbf{76.8}_{\pm 0.1}$	$58.1_{\pm 0.5}$	$62.9_{\pm 0.8}$	$75.1_{\pm 1.0}$	$53.4_{\pm 0.7}$	$59.8_{\pm 0.7}$	$\mathbf{75.7}_{\pm 0.2}$	$53.4_{\pm 0.5}$	$61.1_{\pm 1.0}$	$76.5_{\pm 0.2}$
RoLlama 3.1 8B	$37.9_{\pm 1.1}$	$38.8_{\pm 0.7}$	$\mathbf{76.8}_{\pm 0.8}$	$53.4_{\pm 0.8}$	$45.2_{\pm 1.3}$	$76.6_{\pm 0.3}$	$49.9_{\pm 0.7}$	$43.5_{\pm 1.7}$	$74.6_{\pm 0.4}$	$48.7_{\pm 1.1}$	$42.4_{\pm 0.8}$	$76.4_{\pm 0.3}$
RoMistral 7B	$36.2_{\pm 10.4}$	$45.0_{\pm 0.9}$	$54.5_{\pm 24.7}$	$39.4_{\pm 9.5}$	$49.0_{\pm 0.8}$	$14.7_{\pm 11.9}$	$38.2_{\pm 10.3}$	$47.1_{\pm 0.9}$	$16.3_{\pm 0.8}$	$38.3_{\pm 10.0}$	$47.9_{\pm 0.9}$	$12.3_{\pm 2.7}$
GPT-4.1 mini	45.7	54.5	-	58.8	65.6	-	52.9	62.0	-	51.3	62.6	-
GPT-5 mini	66.7	-	-	71.0	-	-	70.9	-	-	70.4	-	-
						ro → fr						
Gemma 3 27B	$26.1_{\pm 0.1}$	$30.4_{\pm 1.6}$	$76.2_{\pm 0.4}$	$38.9_{\pm 0.3}$	$43.7_{\pm 1.6}$	$\mathbf{77.0}_{\pm 0.9}$	$36.9_{\pm 0.3}$	$41.3_{\pm 2.2}$	$\mathbf{77.6}_{\pm 0.1}$	$30.0_{\pm 0.2}$	$37.5_{\pm 3.0}$	$76.2_{\pm 0.2}$
Llama 3.1 8B	$40.6_{\pm 15.4}$	$41.8_{\pm 2.2}$	$25.5_{\pm 35.9}$	$45.7_{\pm 12.6}$	$49.1_{\pm 1.3}$	$8.8_{\pm 4.2}$	$44.4_{\pm 12.8}$	$47.8_{\pm 1.4}$	$9.8_{\pm 2.2}$	$43.0_{\pm 12.9}$	$46.4_{\pm 1.9}$	$6.4_{\pm 3.5}$
RoGemma 2 9B	$50.6_{\pm 0.8}$	$58.6_{\pm 1.4}$	$\mathbf{77.1}_{\pm 0.2}$	$57.7_{\pm 0.6}$	$63.5_{\pm 0.7}$	$76.6_{\pm 0.5}$	$56.4_{\pm 1.0}$	$61.3_{\pm 0.6}$	$75.5_{\pm 0.6}$	$54.6_{\pm 0.4}$	$62.1_{\pm 1.3}$	$\mathbf{77.2}_{\pm 0.2}$
RoLlama 3.1 8B	$44.4_{\pm 1.1}$	$41.7_{\pm 0.9}$	$76.9_{\pm 0.2}$	$56.5_{\pm 0.8}$	$46.9_{\pm 1.0}$	$76.5_{\pm 0.7}$	$54.2_{\pm 0.9}$	$45.6_{\pm 0.9}$	$77.1_{\pm 0.2}$	$51.9_{\pm 0.8}$	$45.0_{\pm 0.7}$	$74.3_{\pm 0.5}$
RoMistral 7B	$35.1_{\pm 10.5}$	$44.8_{\pm 1.4}$	$24.9_{\pm 35.2}$	$39.0_{\pm 10.2}$	$47.6_{\pm 1.3}$	$54.3_{\pm 25.7}$	$37.5_{\pm 10.6}$	$47.0_{\pm 1.8}$	$13.5_{\pm 3.3}$	$37.0_{\pm 10.7}$	$46.2_{\pm 1.0}$	$27.4_{\pm 15.8}$
GPT-4.1 mini	47.5	55.2	-	59.6	64.3	-	55.4	62.5	-	55.1	62.0	-
GPT-5 mini	69.3	-	-	71.2	-	-	71.1	-	-	70.8	-	-
						ro → it						
Gemma 3 27B	$27.6_{\pm 0.3}$	$35.0_{\pm 2.3}$	$\mathbf{76.7}_{\pm 0.9}$	$40.8_{\pm 0.3}$	$45.7_{\pm 1.3}$	$\mathbf{77.2}_{\pm 0.4}$	$39.2_{\pm 0.3}$	$42.7_{\pm 1.6}$	$\mathbf{77.0}_{\pm 0.7}$	$30.7_{\pm 0.4}$	$39.4_{\pm 2.4}$	$\mathbf{78.3}_{\pm 0.3}$
Llama 3.1 8B	$47.2_{\pm 8.0}$	$41.8_{\pm 2.0}$	$24.9_{\pm 35.2}$	$48.6_{\pm 9.1}$	$48.5_{\pm 1.9}$	$25.9_{\pm 36.6}$	$47.7_{\pm 10.0}$	$48.0_{\pm 2.2}$	$12.4_{\pm 3.3}$	$46.9_{\pm 9.8}$	$44.2_{\pm 1.9}$	$52.7_{\pm 34.2}$
RoGemma 2 9B	$55.1_{\pm 1.1}$	$61.7_{\pm 0.5}$	$76.2_{\pm 0.2}$	$58.7_{\pm 0.6}$	$64.0_{\pm 0.8}$	$75.2_{\pm 0.6}$	$56.9_{\pm 0.7}$	$62.5_{\pm 0.8}$	$76.1_{\pm 0.4}$	$57.3_{\pm 0.7}$	$63.3_{\pm 0.4}$	$76.8_{\pm 0.1}$
RoLlama 3.1 8B	$43.5_{\pm 0.9}$	$41.5_{\pm 1.2}$	$76.3_{\pm 0.4}$	$54.9_{\pm 0.9}$	$45.4_{\pm 1.4}$	$76.4_{\pm 0.3}$	$54.2_{\pm 0.8}$	$44.9_{\pm 0.8}$	$76.6_{\pm 0.4}$	$49.4_{\pm 1.2}$	$44.5_{\pm 0.0}$	$76.2_{\pm 0.2}$
RoMistral 7B	$34.1_{\pm 10.7}$	$47.8_{\pm 1.9}$	$33.7_{\pm 29.5}$	$38.2_{\pm 10.4}$	$49.4_{\pm 1.5}$	$17.2_{\pm 24.3}$	$37.8_{\pm 10.4}$	$47.9_{\pm 1.0}$	$24.4_{\pm 12.6}$	$34.4_{\pm 10.0}$	$47.8_{\pm 0.9}$	$13.8_{\pm 9.7}$
GPT-4.1 mini	52.8	61.7	-	60.8	66.0	-	58.2	64.2	-	56.9	64.1	-
GPT-5 mini	70.1	-	-	71.8	-	-	71.3	-	-	72.1	-	-

Table 8: F1-scores breakdown for each model and code-switching generation method, in the zero-shot (ZS), chain-of-thought (CoT), and fine-tuning (FT) settings, for CS-SeLeRoSa test set. Bold indicates the best performance for each method and language pair.

we show the results broken down by model and method; models generally underperform. However, the performance differences are not very significant in some situations. For instance, with GPT-5 mini, the performance variance across methods is at most 5%. Therefore, some models are more robust to such noisy data. In other cases, this data serves as an adversarial attack, reducing performance by more than 10% compared to other methods. Fine-tuning on such data yields results competitive with other code-switching generation methods, indicating that models can effectively adapt even in noisy environments. While not aimed at maximizing naturalness or simulating real-world usage, it serves as a valid, fast-to-implement data augmentation strategy and provides a useful stress test for assessing model robustness.

6 Conclusions

In conclusion, we investigated the generation of a synthetic code-mixed Romanian satirical news dataset using four different methods and four language pairs. We provided an extensive analysis of the generated dataset using various metrics from the code-switching literature and benchmarked several LLMs on the novel dataset. Our findings indicate that the choice of code-mixing method significantly impacts the quality and characteristics of the generated text. In addition, when we evaluated LLMs on this task, we found that open-source models lag behind proprietary models by a considerable margin, particularly among reasoning models, indicating that there is still room for improvement.

7 Limitations

Our work has several limitations.

Dataset Quality. We only investigate synthetic code-mixing generation without assessing for human evaluation on the entire dataset. Some methods, such as random replacement, produce low-quality outputs. Even though those samples do not reflect real-world scenarios, we believe they can still be used to analyze robustness to noisy inputs. However, we mitigate the dataset evaluation by using various metrics, as indicated in §3.6, and by conducting a human evaluation on a small subset of the data in Appendix D.

Romanian as Matrix Language. We only consider the Romanian language as the base language; however, even if we added more source languages, the number of experiments and costs would increase exponentially. Instead, we focused on four target languages with more resources than Romanian, drawn from two language families.

Small LLM Evaluation. Our evaluation is limited to several small LLMs due to the increased number of experiments and costs. We tried to mitigate this by selecting various models pre-trained on general and Romanian-specific corpora.

8 Ethics Statement

Our work used publicly available datasets and models. The SeLeRoSa dataset (Smădu et al., 2025) is released under the CC BY-NC-SA 4.0 license, with the claim that the original copyright remains with the original authors of the news articles, under the existing copyright laws. The models are either publicly available on HuggingFace or proprietary, but provided using public APIs. Our usage of these resources complies with their intended use. Upon acceptance, we will release our generated code-mixed dataset and code for academic purposes only, under the same license.

References

Gustavo Aguilar, Sudipta Kar, and Thamar Solorio. 2020. LinCE: A centralized benchmark for linguistic code-switching evaluation. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 1803–1813, Marseille, France. European Language Resources Association.

Ariana Bancu. 2013. A comparative analysis of romanian-english and romanian-spanish code-switching patterns.

Srijan Bansal, Vishal Garimella, Ayush Suhane, Jasabanta Patro, and Animesh Mukherjee. 2020. Code-switching patterns can be an effective route to improve performance of downstream NLP applications: A case study of humour, sarcasm and hate speech detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1018–1023, Online. Association for Computational Linguistics.

Ruthanna Barnett, Eva Codó, Eva Eppler, Montse Forcadell, Penelope Gardner-Chloros, Roeland van Hout, Melissa Moyer, Maria Carme Torras, Maria Teresa Turell, Mark Sebba, Marianne Starren, and Sietse Wensing. 2000. The lides coding manual: A document for preparing and analyzing language interaction data version 1.1—july, 1999. *International Journal of Bilingualism*, 4(2):131–132.

Ritu Belani and Jeffrey Flanigan. 2023. Automatic identification of code-switching functions in speech transcripts. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7438–7448, Toronto, Canada. Association for Computational Linguistics.

Lucian Boia. 2001. *Romania: Borderland of Europe*. Reaktion Books, London.

Paul Buzilă. 2025. Recent developments of romanian in contact with other languages in europe. *Journal of Language Contact*, 18(2):1 – 9.

Jenny Chim, Julia Ive, and Maria Liakata. 2025. Evaluating synthetic data generation from user generated text. *Computational Linguistics*, 51(1):191–233.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Amitava Das and Björn Gambäck. 2014. Identifying languages at the word level in code-mixed Indian social media text. In *Proceedings of the 11th International Conference on Natural Language Processing*, pages 378–387, Goa, India. NLP Association of India.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: efficient finetuning of quantized llms. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

A. Seza Doğruöz, Sunayana Sitaram, Barbara E. Bullock, and Almeida Jacqueline Toribio. 2021. A survey of code-switching: Linguistic and social perspectives for language technologies. 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 1654–1666, Online. Association for Computational Linguistics.

Bilal Ghanem, Jihen Karoui, Farah Benamara, Paolo Rosso, and Véronique Moriceau. 2020. Irony detection in a multilingual context. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II*, page 141–149, Berlin, Heidelberg. Springer-Verlag.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Arina Greavu. 2022. Morphological and syntactical aspects of romanian/ english codeswitching. *Swedish Journal of Romanian Studies*, 5(2):191–220.

Bryan Gregorius and Takeshi Okadome. 2022. Generating code-switched text from monolingual text with dependency tree. In *Proceedings of the 20th Annual Workshop of the Australasian Language Technology Association*, pages 90–97, Adelaide, Australia. Australasian Language Technology Association.

Xiaotao Gu, Zihan Wang, Zhenyu Bi, Yu Meng, Liyuan Liu, Jiawei Han, and Jingbo Shang. 2021. Ucphrase: Unsupervised context-aware quality phrase tagging. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 478–486, New York, NY, USA. Association for Computing Machinery.

Gualberto Guzmán, Joseph Ricard, Jacqueline Serigos, Barbara E. Bullock, and Almeida Jacqueline Toribio. 2017. Metrics for modeling code-switching across corpora. In *Interspeech 2017*, pages 67–71.

Injy Hamed, Caroline Sabty, Slim Abdennadher, Ngoc Thang Vu, Thamar Solorio, and Nizar Habash. 2025. A survey of code-switched Arabic NLP: Progress, challenges, and future directions. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 4561–4585, Abu Dhabi, UAE. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Sai Muralidhar Jayanthi, Kavya Nerella, Khyathi Raghavi Chandu, and Alan W Black. 2021. CodemixedNLP: An extensible and open NLP toolkit for code-mixing. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 113–118, Online. Association for Computational Linguistics.

Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean- Bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, and 191 others. 2025. Gemma 3 technical report. *CoRR*, abs/2503.19786.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Jitin Krishnan, Antonios Anastasopoulos, Hemant Purohit, and Huzefa Rangwala. 2021. Multilingual code-switching for zero-shot cross-lingual intent prediction and slot filling. In *Proceedings of the 1st Workshop on Multilingual Representation Learning*, pages 211–223, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jaeseong Lee, YeonJoon Jung, and Seung-won Hwang. 2024. COMMIT: Code-mixing English-centric large language model for multilingual instruction tuning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3130–3137, Mexico City, Mexico. Association for Computational Linguistics.

Mihai Masala, Denis Ilie-Ablachim, Alexandru Dima, Dragos Georgian Corlatescu, Miruna-Andreea Zavelca, Ovio Olaru, Simina-Maria Terian, Andrei Terian, Marius Leordeanu, Horia Velicu, Marius Popescu, Mihai Dascalu, and Traian Rebedea. 2024. "vorbești românește?" a recipe to train powerful Romanian LLMs with English instructions. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11632–11647, Miami, Florida, USA. Association for Computational Linguistics.

Carol Myers-Scotton. 1993. *Duelling Languages: Grammatical Structure in Codeswitching*. Oxford University Press.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, and 20 others. 2024. Scaling neural machine translation to 200 languages. *Nature*, 630(8018):841–846.

OECD. 2019. *Talent Abroad: A Review of Romanian Emigrants*. Talent Abroad. OECD Publishing, Paris.

Kayode Olaleye, Arturo Oncevay, Mathieu Sibue, Nombuyiselo Zondi, Michelle Terblanche, Sibongile Mapikitla, Richard Lastrucci, Charese Smiley, and Vukosi Marivate. 2025. AfroCS-xs: Creating a compact, high-quality, human-validated code-switched dataset for African languages. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33391–33410, Vienna, Austria. Association for Computational Linguistics.

OpenAI. 2025. Gpt-5 system card. https://cdn.openai.com/gpt-5-system-card.pdf. Accessed: 2025-10-06.

Kartikey Pant and Tanvi Dadu. 2020. Towards code-switched classification exploiting constituent language resources. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop*, pages 37–43, Suzhou, China. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Adithya Pratapa, Gayatri Bhat, Monojit Choudhury, Sunayana Sitaram, Sandipan Dandapat, and Kalika Bali. 2018. Language modeling for code-mixing: The role of linguistic theory based synthetic data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1543–1553, Melbourne, Australia. Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, pages 1–20.

Cesa Salaam, Franck Dernoncourt, Trung Bui, Danda Rawat, and Seunghyun Yoon. 2022. Offensive content detection via synthetic code-switched text. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6617–6624, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Aditya Shah and Chandresh Maurya. 2021. How effective is incongruity? implications for code-mixed sarcasm detection. In *Proceedings of the 18th International Conference on Natural Language Processing (ICON)*, pages 271–276, National Institute of Technology Silchar, Silchar, India. NLP Association of India (NLPAI).

Rajvee Sheth, Himanshu Beniwal, and Mayank Singh. 2025. COMI-LINGUA: Expert annotated large-scale dataset for multitask NLP in Hindi-English code-mixing. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 7973–7992, Suzhou, China. Association for Computational Linguistics.

Răzvan-Alexandru Smădu, Andreea Iuga, Dumitru-Clementin Cercel, and Florin Pop. 2025. Selerosa: Sentence-level romanian satire detection dataset. *arXiv preprint arXiv:2509.00893*.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.

Vivek Srivastava and Mayank Singh. 2021. Challenges and limitations with the metrics measuring the complexity of code-mixed text. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 6–14, Online. Association for Computational Linguistics.

Igor Sterner and Simone Teufel. 2025a. Code-switching and syntax: A large-scale experiment. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11526–11533, Vienna, Austria. Association for Computational Linguistics.

Igor Sterner and Simone Teufel. 2025b. Minimal pair-based evaluation of code-switching. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18575–18598, Vienna, Austria. Association for Computational Linguistics.

Ishan Tarunesh, Syamantak Kumar, and Preethi Jyothi. 2021. From machine translation to code-switching: Generating high-quality code-switched text. 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 3154–3169, Online. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Csilla-Anna Wilhelm. 2017. Between simplification and complexification. german, hungarian, romanian noun and adjective morphologies in contact. *Journal of Language Contact*, 10(1):56 – 75.

Peng Xie, Xingyuan Liu, Yequan Bie, Tsz Wai Chan, Yangqiu Song, Yang Wang, Hao Chen, and Kani

Chen. 2026. Switchlingua: The first large-scale multilingual and multi-ethnic code-switching dataset. *Advances in Neural Information Processing Systems*, 38.

Yilun Yang and Yekun Chai. 2025. Codemixbench: Evaluating code-mixing capabilities of llms across 18 languages. *Preprint*, arXiv:2507.18791.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Wenbo Zhang, Aditya Majumdar, and Amulya Yadav. 2025. Chai for llms: Improving code-mixed translation in large language models through reinforcement learning with ai feedback. *Preprint*, arXiv:2411.09073.

A Motivation

The motivation and ultimate goal of our work, including the dataset construction and extensive evaluation, is multifold:

First, it evaluates the current state of models' understanding of satire at a granular level. Satire often occurs sporadically rather than persistently throughout an entire text. Sentence-level labels capture this nuance, which is crucial for analyzing long-form texts or instances of sporadic satire.

Second, code-switching is a phenomenon accentuated by globalization. As presented in §1, Romania has one of the largest diasporas in the European Union, particularly in the countries we investigate for code-switching. Given the growing influence of LLMs through chat-based applications in daily life, which is likely to increase further, we believe evaluating the models' performance under such conditions is sound. Mixing languages (as suggested by Grattafiori et al. (2024)) could affect performance, and our results indicate this (see Table 8 vs Table 7; in all cases, baseline performance is superior to code-mixed performance).

Third, for satire itself, as described above, there are very few public sources (they usually occur in informal settings (Pant and Dadu, 2020)), making it even more challenging to find code-switching in this setting. Therefore, naturally, we developed a code-switched dataset to address this shortcoming until more reliable resources could be generated. Note that this is less of an issue with other languages, such as Hindi.

B Dataset Structure

For each sample in our dataset, we maintain the generation index; therefore, subsets can be easily evaluated, avoiding the diversity issue of our dataset. More specifically, in addition to the columns from the SeLeRoSa dataset[6], our code-mixed dataset adds the following fields for each sample:

- `code_switch_language`: the target language for code switching (the source is by default Romanian);
- `k`: the generation index, k=0,1,2,3; where k=0 means the original sentence;
- `cs_method`: the code switching generation method (none, random, dependency, llm, phrase), where none means the original sentence.

[6] https://huggingface.co/datasets/unstpb-nlp/SeLeRoSa

C Additional Experimental Setup and Results

C.1 Models

In Table 9, we present the model checkpoints used in our experiments, along with their sizes. All models used Romanian in their pre-training data. In what follows, we summarize the training data and its importance to each model's performance.

Llama 3.1 (Grattafiori et al., 2024) was trained on a mix of publicly available data totaling more than 15T tokens. The model was optimized for several languages, including English, German, French, and Italian; however, the training data included more languages, such as Romanian.

Gemma 3 models (Kamath et al., 2025) were trained on various data, including web documents (from over 140 languages), code, and images, totaling 14T tokens for the 27B parameters model. However, we could not find exact details about the data.

The Romanian language models (Masala et al., 2024) were instruction-fine-tuned on 2.7M instructions and conversations translated from various existing datasets. Because all base models used the same instruction-fine-tuning data, the observed performance disparities stem from differences in their pretraining strategies, data, and architectures.

Regarding the closed-source models (GPT 4.1 and 5 mini), as with previous models, the data is a mix of publicly available and proprietary sources. The exact size and nature of the datasets are unknown to the public.

The training data used to generate these checkpoints influences the models' performance, as suggested in §4. In zero-shot settings, RoGemma 2 and RoLlama 3.1 consistently outperform their base counterparts on the gold-standard dataset. While closed-source models achieve the highest scores, Romanian-pretrained models remain highly competitive. This likely reflects the relative volume and composition of their pre-training data. Additionally, the Llama 3 model card (Grattafiori et al., 2024) notes that mixing languages could yield unexpected results, a finding we validated by observing poor performance.

Model Name	Checkpoint	Size
Gemma3-12B	google/gemma-3-12b-it	12B
Gemma3-27B	google/gemma-3-27b-it	27B
Llama-3.1-8B	meta-llama/Llama-3.1-8B-Instruct	8B
RoGemma2-9B	OpenLLM-Ro/RoGemma2-9b-Instruct-DPO	9B
RoLlama3.1-8B	OpenLLM-Ro/RoLlama3.1-8b-Instruct-DPO	8B
RoMistral-7B	OpenLLM-Ro/RoMistral-7b-Instruct-DPO	7B
GPT-4.1-mini	gpt-4.1-mini-2025-04-14	–
GPT-5-mini	gpt-5-mini-2025-08-07	–

Table 9: Model checkpoints used for code-switching and translation.

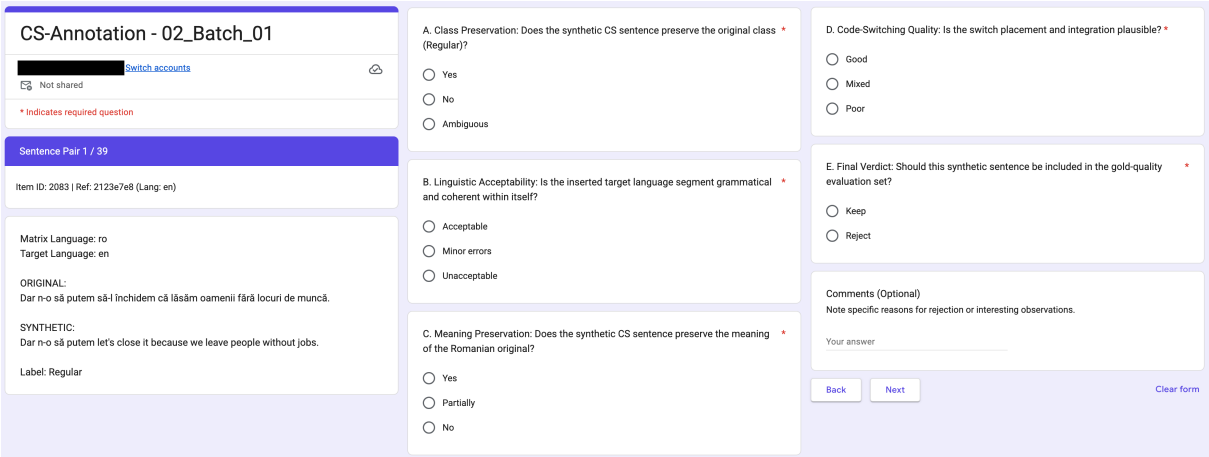

Figure 1: Screenshot of an example from the Google Form used during the annotation process. It indicates the batch identifier, progress through the batch, sample identifier, matrix, and target languages, the original and synthetic sentences, and the original label. After this, there are the five mandatory questions and an optional comments field. The original screenshot was anonymized and split into three columns for presentation purposes.

C.2 Computational Resources and Costs

For inference, we used an NVIDIA A100 80GB GPU, employing vllm[7] to improve inference speed, with an OpenAI-compatible endpoint. For the proprietary models, we use the OpenAI API[8]. The total cost of the experiments on the proprietary models is approximately 100 USD.

D Human Evaluation

The annotation process involved four annotators proficient in Romanian and knowledgeable of the target languages (i.e., English, German, French, and Italian). The annotators were provided with detailed guidelines to ensure consistency in their evaluations.

D.1 Annotation Protocol

The annotation campaign was done using Google Forms[9]. Each annotator was presented with the original Romanian sentences, the target language segments used for code-switching, and the original satirical label. Inspired by Chim et al. (2025), the annotators were provided guidelines with instructions and examples to evaluate each synthetic sentence based on the following criteria:

- **Class Preservation**: Does the synthetic CS sentence preserve the original class (satire or regular)? Yes/No/Ambiguous
- **Linguistic Acceptability**: Is the inserted target language segment grammatical and coherent within itself? Acceptable/Minor errors/Unacceptable
- **Meaning Preservation**: Does the synthetic

[7] https://github.com/vllm-project/vllm
[8] https://platform.openai.com/docs/models
[9] forms.google.com

CS sentence preserve the meaning of the original Romanian sentence? Yes/Partially/No

- **Code-Switching Quality**: Is the switch placement and integration plausible? Good/Mixed/Poor
- **Final Verdict**: Should the synthetic sentence be kept or rejected? Keep/Reject

The annotators were instructed to reject any synthetic sentence that introduces hate speech, targeted insults, or violent content not present in the original. The full annotation guidelines are provided in the Supplemental Materials.

The annotation process was performed in two phases. First, **Calibration Phase**, where annotators labeled a small set of sentences to align their understanding of the guidelines. Second, **Batches Annotation Phase**, where the remaining workload was distributed in manageable batches. To make the process easier to leverage, we provided the data in batches of approx. 40 examples, containing code-mixed sentences whose target languages were known by the annotator. An example is presented in Figure 1. At the end, we checked the agreement among annotators, and in case of disagreements, they discussed to reach a consensus. A final human verification was then performed on all samples to ensure response quality and identify any potential issues.

In addition, to the best of our knowledge, we note that this is the first work to include such a rich set of categories labeled by human annotators for code-switched sentences. Most works limit themselves to fewer criteria (e.g., naturalness and acceptability (Sterner and Teufel, 2025b)) or employ humans in the loop to validate and fix the samples (Olaleye et al., 2025).

Since the samples are synthetically generated, we provide this minimal set of categories to label to reduce annotation bias. The questions are designed to identify potential misinterpretations of the guidelines and to fix them in a timely manner. That is, if we found contradictions during annotation (e.g., the annotator considered the linguistic acceptability unacceptable; however, the final verdict was to keep the sample), we addressed them in the adjudication and final verification stages. The number of options per question was kept low (i.e., 2-3 categorical options).

We also note that throughout the annotation process, the human experts identified additional aspects that could be interesting to investigate. For example, the phonetic/pronunciation aspect affects how easily integrated a code-switched phrase is in the matrix sentence. Additionally, some words may be pronounced similarly despite having different meanings or parts of speech (e.g., "joc" in Romanian can mean the noun "game" or the first-person singular verb "to play"). However, we leave this investigation as future work.

D.2 Data Selection

To obtain a representative subset, we select sentences from the generated dataset for human evaluation. Specifically, we randomly sampled 50 non-code-mixed Romanian sentences from 49 news sources within the test set. We extracted the corresponding generated code-mixed sentences for each of the four methods and four languages. We deduplicated the generated samples and removed those that were not translated before sampling. In total, we annotated 715 samples. The base sentences contained 37 satirical (i.e., 74%) and 13 regular (i.e., 26%) samples, distributed among social (i.e., 78%) and technology and science (i.e., 22%) domains. For each language, between 150 and 200 samples were manually labeled (see also Table 10).

D.3 Annotation Statistics

Label Distribution. Following human-led adjudication on the initially selected data, we derived a final high-quality benchmark of 391 samples (54.7% of the total annotated set), as presented in §D.1. The per-language acceptance distribution is presented in Table 10. Investigating the acceptance rate at a more granular level in Figure 2, the LLM-generation and dependency-based approaches achieved the highest rates, over 60% and, in some cases, 80%. Also, as expected, the random replacement method generates low-quality samples (i.e., acceptance rate is under 20%), mainly due to violations of linguistic rules or mistranslations caused by a lack of context. The most common reason for rejection was the poor quality of code-switching, with 63.9% of the samples being annotated as poor code-switching quality (see Figure 3). This is followed with a significant margin by linguistic errors (34%) and meaning loss (26.9%). Only 8% of the rejection cases involved a loss of class label, usually due to ambiguity or clarity reasons.

Investigating the label distribution per criterion (Figure 4), we note a skew towards the *yes* label

Language	Adjudicated	Keep	AR (%)
English	199	117	58.8%
French	198	118	59.6%
German	159	70	44.0%
Italian	159	86	54.1%
Total	**715**	**391**	**54.7%**

Table 10: Language distribution of the original annotated set and human verification acceptance rates (AR) after the annotation process.

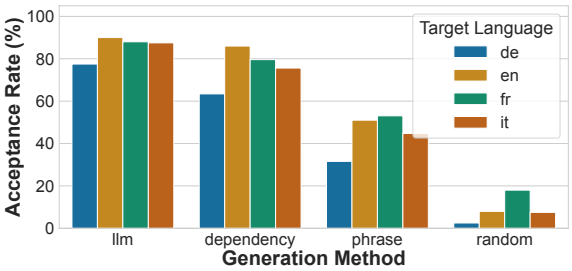

Figure 2: The acceptance rate by method and target language.

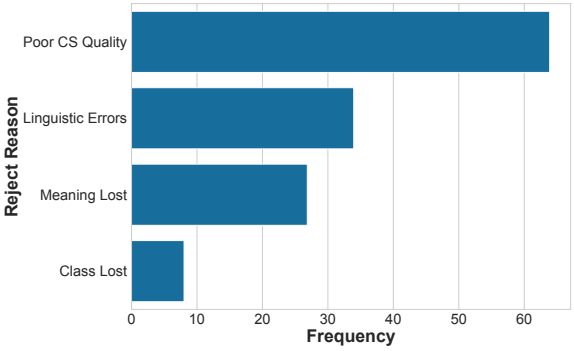

Figure 3: The main reasons the samples were rejected, as a percentage of the rejected samples. Note that multiple reasons could lead to a rejected outcome.

for the *class preservation* criterion. Comparing the keep/reject verdicts, we see that the positive class mostly yields a favorable outcome. However, for class preservation, the keep rate is lower for the *yes* label than for similar labels in other categories, indicating that this metric is a weaker signal for the final verdict.

Criterion Importance. To investigate which criteria most strongly influence annotators' final verdicts, we employ logistic regression modeling. We train a separate model for each language as well as a global one. In each model, the final verdict (keep/reject) is the binary dependent variable, while the four other criteria are independent variables. This allows us to compute regression coefficients that quantify the relative importance of each criterion in determining the final verdict. Formally, we model:

$$\begin{aligned} \textit{FinalVerdict} = \beta_0 + \beta_1 \cdot CP + \beta_2 \cdot LA \\ + \beta_3 \cdot MP + \beta_4 \cdot CQ + \epsilon \end{aligned} \quad (2)$$

where CP is class preservation, LA is linguistic acceptability, MP is meaning preservation, CQ is code-switching quality, and ϵ is the error term representing unexplained variance in the model. The magnitude and sign of each coefficient (β_i) indicate the strength and direction of influence for each criterion. Larger absolute values suggest that the criterion has a stronger effect on the final verdict decision.

Lang.	CP	LA	MP	CQ	Acc.
English	-0.001	1.373	2.070	1.844	94.0
German	0.770	1.883	1.004	1.333	90.6
French	0.209	1.232	2.127	1.833	93.4
Italian	0.156	1.759	1.468	1.884	91.8
All	0.556	1.858	2.089	1.858	92.9

Table 11: Logistic regression coefficients indicating criterion influence on final verdict, along with the model accuracy (Acc.) as a percentage. Higher values indicate a stronger influence on accepting the sample.

We train the logistic regression for at most 1,000 epochs or until convergence. Table 11 presents the logistic regression coefficients learned for each annotator across languages. The regression models achieve over 90% accuracy in predicting the final verdict. Upon investigating the regression coefficients, we confirm that class preservation yields the lowest signal. On the other hand, meaning preservation represents the most important feature, while language acceptability and code-switching quality are, to a lesser degree, similarly important.

Inter-Annotator Agreement. Table 12 shows the overall inter-annotator agreement across criteria, showing moderate to strong agreement. The agreement between annotators for the final verdict is 0.518 on Krippendorff's alpha (α), indicating substantial agreement. We note an exception to *class preservation* criteria, where $\alpha = 0.178$, due to the skew towards the *yes* label. In Table 13, we show the breakdown by language and criteria. As noted before, class preservation presents the lower agreements. Usually, the agreement ranges from moderate to strong.

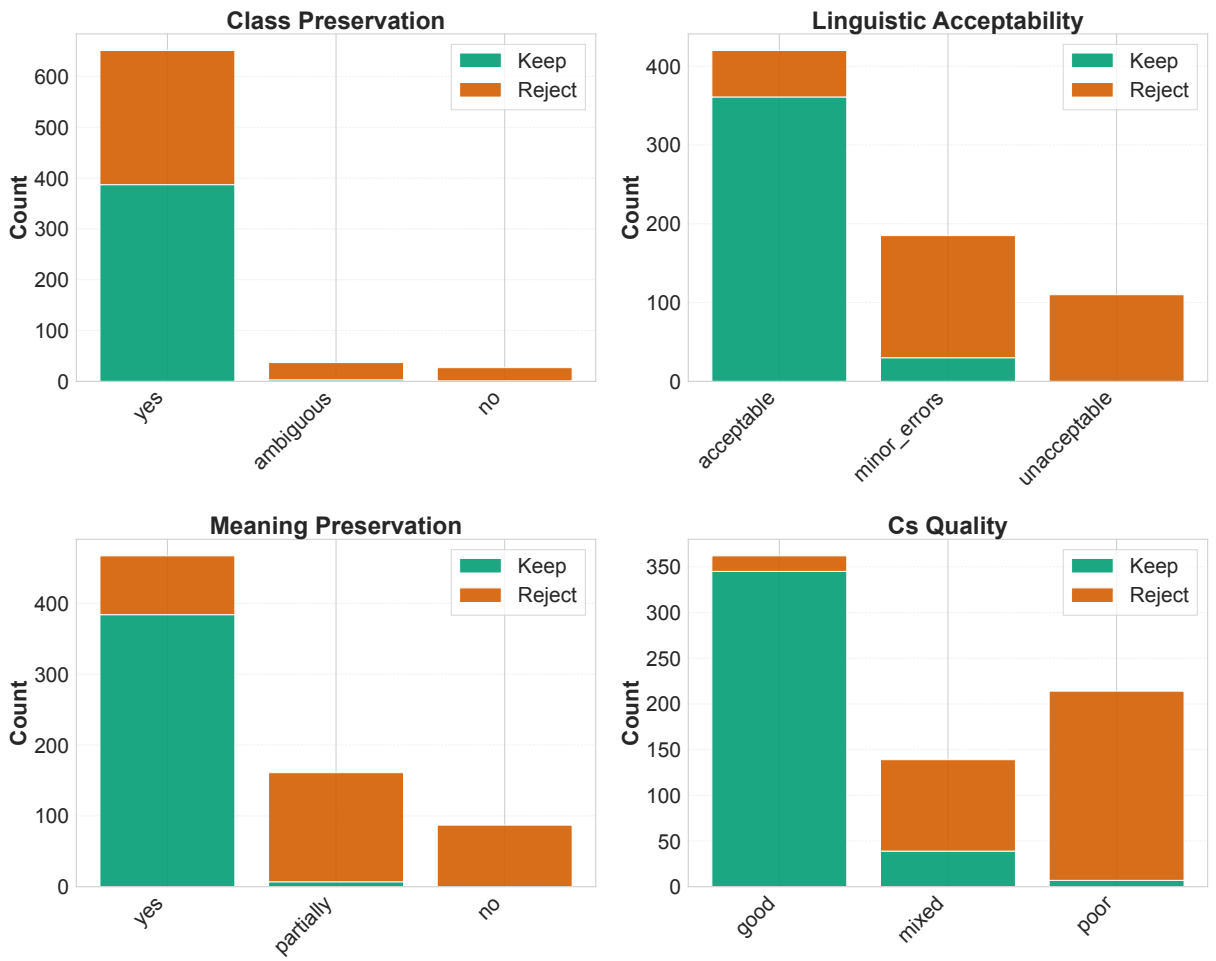

Figure 4: Per-criteria distribution of the human labels after the adjudication process, on class preservation, linguistic acceptability, meaning preservation, and code-switching quality.

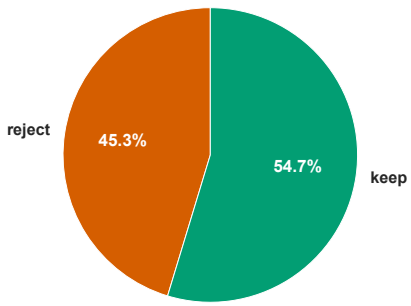

Figure 5: Final verdict distribution after the adjudication process.

Criterion	α	Agreement (%)
Class Preservation	0.178	85.0%
Linguistic Acceptability	0.603	72.3%
Meaning Preservation	0.609	76.5%
CS Quality	0.655	66.6%
Final Verdict	0.518	75.9%

Table 12: Inter-annotator agreement in terms of Krippendorff's alpha (α) and agreement rate across all criteria for all N=715 samples.

Disagreement Assessment. We investigate the main reasons for disagreement during the annotation process in Figure 6. The main categories where we noticed disagreements were code-switching quality assessment, especially at the phrase level. Additionally, random generation posed greater challenges when considering all characteristics except the verdict, for which annotators generally agreed that those same characteristics should be rejected. The least challenges were raised by class preservation and meaning preservation across methods, except for random. The original class label of the sentence (i.e., whether satire or non-satire) does not provide any meaningful influence on the results.

D.4 Gold Standard Dataset Statistics

The final dataset resulted from the adjudication process; 391 samples remained in the gold set with

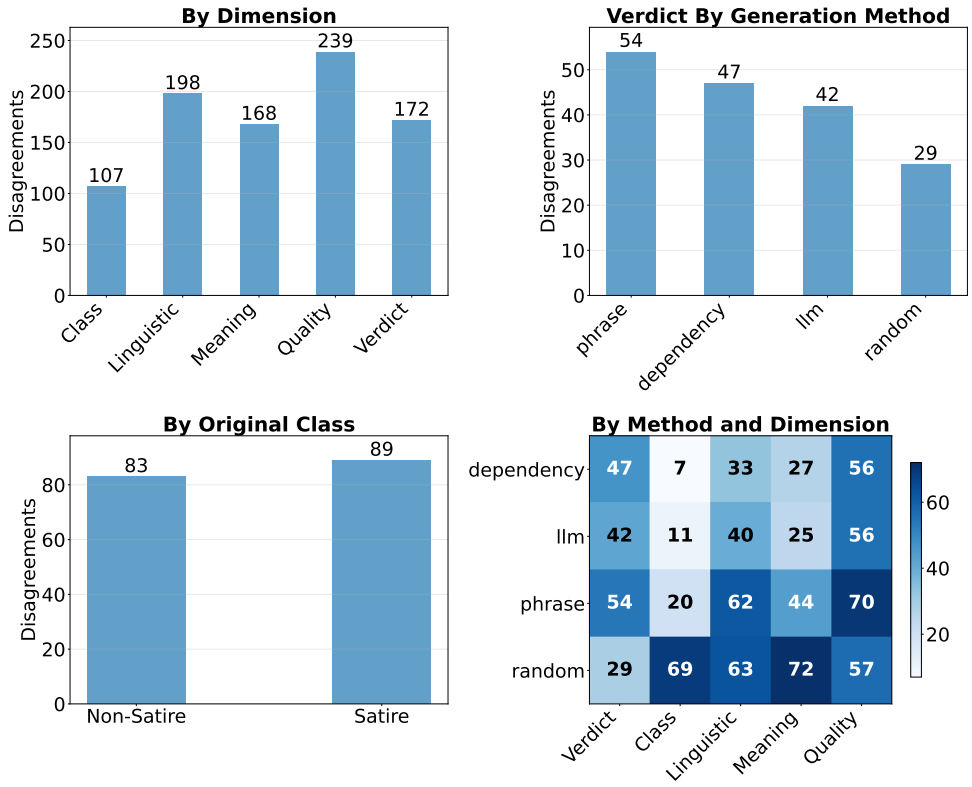

Figure 6: Main disagreements between annotators considering category, generation method, and original class.

Language	CP	LA	MP	CSQ	Verdict
English	0.205	0.476	0.629	0.679	0.438
German	0.315	0.723	0.503	0.731	0.634
French	-0.024	0.364	0.644	0.514	0.416
Italian	0.220	0.672	0.675	0.670	0.617

Table 13: Inter-annotator agreement (Krippendorff's α) across target languages by verdict, linguistic acceptability (LA), code-switching quality (CSQ), class preservation (CP), and meaning preservation (MP).

Method	DE	EN	FR	IT	Total
LLM	31	45	44	35	155
Dependency	26	43	39	31	139
Phrase	12	25	26	17	80
Random	1	4	9	3	17
Total	70	117	118	86	391

Table 14: Distribution of the gold standard dataset across code-switching generation methods and target languages.

"keep" as the final verdict (i.e., 54.7%).

In Table 14, we present the distribution of samples in the gold standard dataset across code-switching methods and target languages. Most samples are from LLM and dependency-based generation, while the fewest are from random-based generation. Considering the language distribution, there are approximately 120 samples mixed in English and French, and 70-86 mixed in German and Italian. In Table 15, we provide summary statistics of the gold standard dataset used for evaluating code-switching generation methods. The final distribution of the data is 32% satirical samples and 68% regular samples. The average sentence length is 21 words, with overall coverage of method-language pairs at 89.6% (e.g., German-random pair is missing).

Table 14 presents the code-switching statistics over each target language and generation method. Overall, we notice that for methods such as random, the human annotators preferred those with fewer switching points. Other metrics, such as those for phrase, dependency, and LLM-based generation, remained generally unchanged.

Statistic	Value
Total samples	391
Unique source sentences	50
CS methods	4
Target languages	4
K values	3
Satire samples	125 (32.0%)
Non-satire samples	266 (68.0%)
Avg sentence length	21.0 words
Method-language pair coverage	89.6%

Table 15: Summary statistics of the gold standard dataset used for evaluating code-switching generation methods.

Method	CMI	M-Index	I-Index	LE	SP-Avg
		RO → EN			
Random	24.60	0.28	0.34	0.65	0.34
Phrase	15.51	0.18	0.21	0.56	0.21
Dependency	23.47	0.26	0.19	0.68	0.19
LLM	17.14	0.20	0.20	0.59	0.20
		RO → FR			
Random	36.52	0.41	0.40	0.86	0.40
Phrase	23.18	0.27	0.29	0.75	0.29
Dependency	22.38	0.26	0.22	0.70	0.22
LLM	22.88	0.27	0.27	0.73	0.27
		RO → DE			
Random	-	-	-	-	-
Phrase	27.71	0.32	0.29	0.81	0.29
Dependency	25.77	0.29	0.20	0.75	0.20
LLM	21.00	0.25	0.26	0.70	0.26
		RO → IT			
Random	14.29	0.16	0.19	0.33	0.19
Phrase	23.80	0.27	0.29	0.74	0.29
Dependency	23.13	0.25	0.24	0.74	0.24
LLM	25.94	0.28	0.28	0.80	0.28

Table 16: Aggregated code-switching metrics by language and code-switching method on the gold set (i.e., keep set, N=391).

E t-SNE Visualization

In Figure 7, we present the t-SNE visualization (perplexity of 500) of the original and code-mixed sentences, generated using each method on the four languages: English (EN), German (DE), French (FR), and Italian (IT). We embed the text using the multilingual BERT-cased model[10]. The number of generated samples (i.e., n_cs) may differ between methods since not all produced exactly $k = 3$ variants. We observe that random code-mixing yields separable representations compared to other approaches, indicating a semantic shift when words are randomly translated. This is also emphasized by the semantic similarity results in Table 4, and the annotation process that filtered out a large portion of the random and phrase-generated samples. On the other hand, the other approaches generate code-mixed examples that are closer to the original distribution by considering more context when translating text chunks, thereby better preserving semantics. Across languages, we do not see meaningful differences in the representations, which is expected, given that the goal of the synthetic generation was to preserve the original meaning of the sentence by providing translated variations.

[10] https://huggingface.co/google-bert/bert-base-multilingual-cased

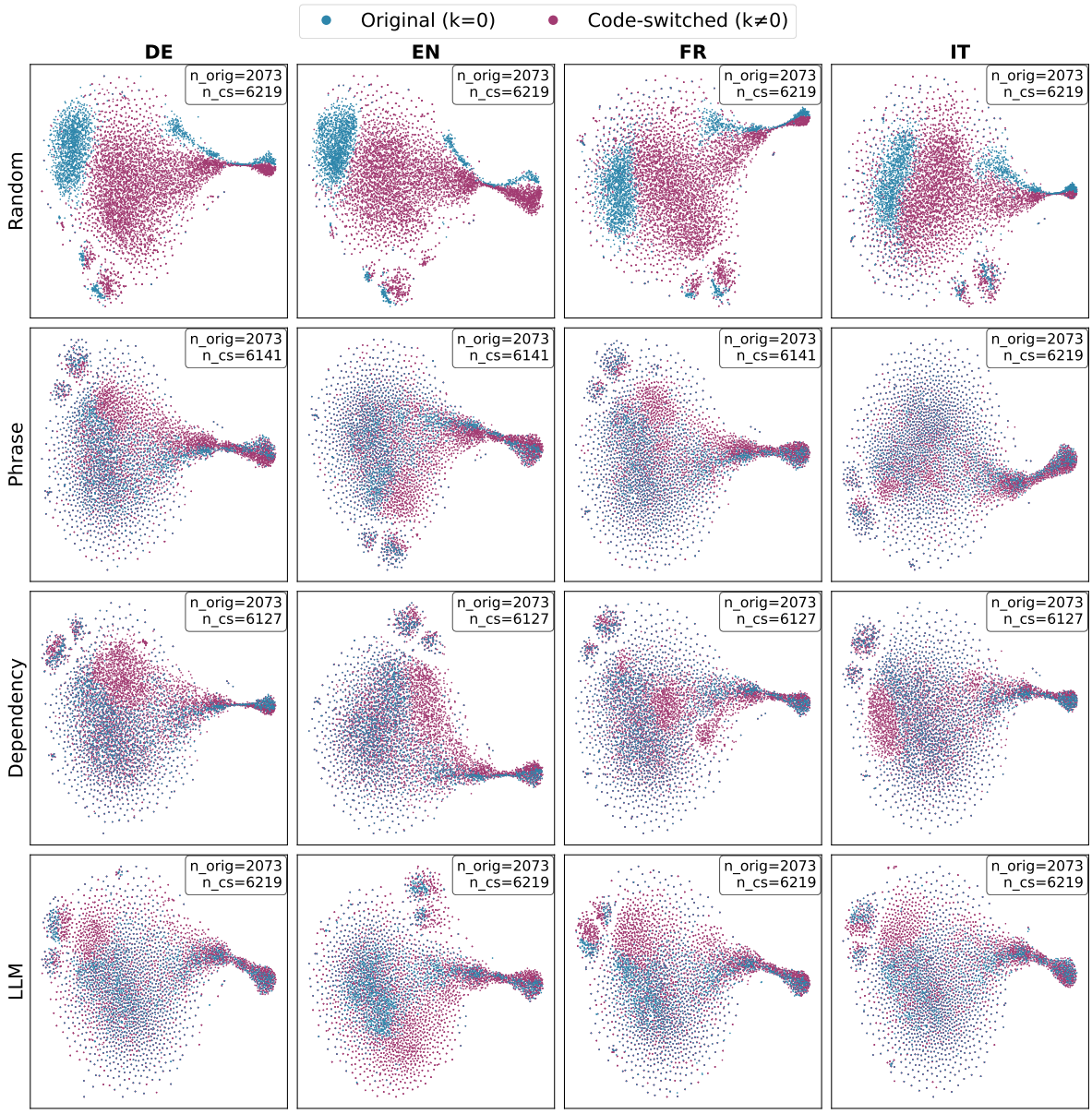

Figure 7: t-SNE visualization of sentence embeddings from different languages (EN - English, DE - German, FR - French, IT - Italian) and code-switching generation methods.

F LLM Prompts

Prompt for Machine Translation

You are an expert translator. Your task is to translate text from {src_lang} to {trg_lang}. Translation Instructions:
- Translate all content inside <input> tags to {trg_lang}.
- Preserve the original formatting, punctuation, and line breaks.
- If any words, phrases, or named entities are already translated or are in the target language, leave them unchanged (even if this results in duplicates).
- Do not translate code, numbers, URLs, or named entities; keep them as they are.
- If the text is code-mixed or partially translated, only translate the parts in {src_lang}.
- Output ONLY the translation, wrapped between <output> and </output> tags, with no extra commentary or explanation.

Prompt for Generating Code-Mixed Sentences

You are a bilingual speaker of {src_lang} and {trg_lang} who frequently mixes words from both languages in conversations.

You will receive a sentence in {src_lang}. Your task is to generate code-mixed text by naturally mixing words or phrases from the {src_lang} language with the {trg_lang} language. You **must** preserve the meaning and maintain grammatical coherence of the original sentence. Try to mix the languages as naturally as possible, similar to how bilingual speakers would. At most several word or phrases should be translated {trg_lang} language. Keep most words from the sentence in {src_lang} language.

You will be provided with a sentence in the source language between <input> and </input> tags. Return only the code-mixed text without explanations, between <output> and </output> tags.

Romanian Prompt for Satire Detection

Ești un expert in detecția satirei și a ironiei în propoziții code-mixed cu limba română și {target_lang}. O să primești o propoziție și trebuie să o adnotezi cu ‘{class0}‘ sau ‘{class1}‘.

Răspunsul final îl vei oferi în următorul format:

“‘
`<response>[RĂSPUNS]</response>`
“‘

unde `[RĂSPUNS]` este ‘{class0}‘ sau ‘{class1}‘.

Nu oferi explicații sau să zici altceva decât "`<response>[RĂSPUNS]</response>`". Te rog să respecți formatul cerut, altfel răspunsul va fi considerat greșit.

English Translated Prompt for Satire Detection

You are an expert in detecting satire and irony in code-mixed sentences combining Romanian and {target_lang}. You will receive a sentence and must annotate it with ‘{class0}‘ or ‘{class1}‘.

Provide the final answer in the following format:

“‘
`<response>[RESPONSE]</response>`
“‘

where `[RESPONSE]` is ‘{class0}‘ or ‘{class1}‘.

Do not provide explanations or say anything other than "`<response>[RESPONSE]</response>`". Please respect the required format, otherwise the answer will be considered incorrect.

Romanian Prompt for Satire Detection using Chain-of-Thought

Ești un expert in detecția satirei și a ironiei în propoziții code-mixed cu limba română și {target_lang}. O să primești o propoziție și trebuie să o adnotezi cu ‘{class0}‘ sau ‘{class1}‘.
 	
Te rog să te gândești pas cu pas înainte de a răspunde.
 	
Răspunsul final îl vei oferi în următorul format:
 	
“‘
`<reasoning>`
`[RAȚIONAMENTUL TĂU]`
`</reasoning>`
`<response>[RĂSPUNS]</response>`
“‘
 	
unde `[RAȚIONAMENTUL TĂU]` este raționamentul tău pas cu pas de maxim 50 de cuvinte și `[RĂSPUNS]` este ‘{class0}‘ sau ‘{class1}‘.
 	
Te rog să respecți formatul cerut, altfel răspunsul va fi considerat greșit.

English Translated Prompt for Satire Detection using Chain-of-Thought

You are an expert in detecting satire and irony in code-mixed sentences combining Romanian and {target_lang}. You will receive a sentence and must annotate it with ‘{class0}‘ or ‘{class1}‘.
 	
Please think step by step before answering.
 	
Provide the final answer in the following format:
 	
“‘
`<reasoning>`
`[YOUR REASONING]`
`</reasoning>`
`<response>[RESPONSE]</response>`
“‘
 	
where `[YOUR REASONING]` is your step-by-step reasoning of maximum 50 words and `[RESPONSE]` is ‘{class0}‘ or ‘{class1}‘.
 	
Please respect the required format, otherwise the answer will be considered incorrect.