Conditioning LLMs to Generate Code-Switched Text: A Methodology Grounded in Naturally Occurring Data

Anonymous ACL submission

Abstract

Code-switching (CS) is still a critical challenge in Natural Language Processing (NLP). Current Large Language Models (LLMs) struggle to interpret and generate code-switched text, primarily due to the scarcity of large-scale CS datasets for training. This paper presents a novel methodology to generate CS data using LLMs, and test it on the English-Spanish language pair. We propose back-translating natural CS sentences into monolingual English, and using the resulting parallel corpus to fine-tune LLMs to turn monolingual sentences into CS. Unlike previous approaches to CS generation, our methodology uses natural CS data as a starting point, allowing models to learn its natural distribution beyond grammatical patterns. We thoroughly analyse the models' performance through a study on human preferences, a qualitative error analysis and an evaluation with popular automatic metrics. Results show that our methodology generates fluent code-switched text, expanding research opportunities in CS communication, and that traditional metrics do not correlate with human judgement when assessing the quality of the generated CS data. We release our code and generated dataset under a CC-BY-NC-SA license.[1]

1 Introduction

Code-Switching (CS) consists of mixing two or more languages within a single utterance and is a common phenomenon in multilingual settings (Tucker, 2001). Although it is mainly present in spoken interactions, it can also be found in written interactions on-line (Appel and Muysken, 2005; Sarkisov, 2021), where it appears jointly with other features of informal speech. Example 1 shows an utterance where the speaker switches between English and Spanish.

(1) Why make everybody *sentarse atrás pa' que* everybody has to move *pa' que se salga*.
Why make everybody *sit at the back so that* everybody has to move *so that she may get out.*[2]

(Poplack, 1980)

Despite the prevalence of code-switching, most research in Natural Language Processing (NLP) assumes monolingualism as a standard for human communication. However, this implicit decision means that state-of-the-art models are not able to properly interpret or generate CS data. Even advances in *multilingual* language modelling (Lin et al., 2022; Chowdhery et al., 2023) have not led to significant improvements, and performance on CS data is still poor compared to performance on monolingual data (Aguilar et al., 2020; Winata et al., 2021). This occurs because there is little CS text available in the multilingual pretraining data. Similarly, there are no parallel datasets available to learn to generate CS in a supervised fashion, as one would expect for tasks such as machine translation. Finally, there is no clear methodology for evaluating automatically generated CS text, as it has specific needs different from other text generation tasks.

It is therefore crucial to develop methodologies to enable models to generate natural CS text and simultaneously implement robust evaluation frameworks that can assess how well NLP systems handle CS across multiple tasks. We argue that both of these goals require models that can conditionally generate CS from monolingual text. Consequently, our research focuses on the development of a methodology to fine-tune and evaluate LLMs on the task of CS generation, following two main research questions:

[1] URL to be announced upon acceptance.

[2] In all examples of CS featured in this paper, Spanish parts are shown in italics, in both the original instance and its translation.

RQ1: Is it viable to approach CS generation with natural CS text as a starting point?

RQ2: Do automatic metrics for Natural Language Generation (NLG) correlate well with human judgement for the task of CS generation?

We propose a novel method to generate CS text from monolingual text using LLMs, which requires less data for fine-tuning. We apply this to the English-Spanish pair (RQ1). Unlike prior work, our approach leverages naturally occurring code-switched data instead of artificial examples, allowing models to learn its natural distribution. Additionally, our source texts come from informal contexts, making the output both code-switched and informal — two intertwined phenomena. We evaluate our methodology both qualitatively, with a study on human preferences and a manual error analysis, and quantitatively, using automatic NLG metrics, which allows us to study the correlation with between human and automatic evaluation for this task (RQ2).

To summarize, the contributions presented in this paper are the following:

- we develop a novel approach for CS generation through fine-tuning LLMs on a psuedo-parallel corpus sourced from natural code-switched data.
- we present a new parallel English-CS corpus, *EN2CS*, with silver train and development splits and a gold post-edited test set, which can be used to train and evaluate model for English-Spanish CS generation.
- we perform both human and automatic evaluation of the results of models fine-tuned on this dataset, and perform a study on the correlation between human and automatic evaluation for CS generation.

2 Related Work

Perspectives in linguistics. CS naturally occurs in communities where two or more languages are in contact, making it a subject of great interest to fields like sociolinguistics and psycholinguistics. From a social perspective, it can be affected by the attitudes of the speakers towards the languages and the CS phenomenon itself. In this respect, it is related and associated with notions of prestige and identity (Heredia et al., 2025). For example, in bilingual communities where a language is minoritized, CS can be regarded as an intrusion of the majority language (Dewaele and Wei, 2014). However, for migrant communities, it may be a way to preserve their mother tongue and as an "emblem of ethnic identity" (Poplack, 1980). Once again, its importance in different social contexts highlights the need to consider CS in NLP research, as it plays a crucial role in linguistic interactions and, consequently, the development of language technologies.

CS in NLP. The processing and understanding of code-switched text can be crucial in the processing of social media data (Bali et al., 2014), and for speech applications, such as speech recognition or speech synthesis (Rallabandi and Black, 2017). In fact, non-monolingual speakers have shown preference for chatbots that use CS (Bawa et al., 2020). Different approaches may include normalization (Parikh and Solorio, 2021), machine translation (Xu and Yvon, 2021) or modeling code-switched text (Gonen and Goldberg, 2019). The survey by Winata et al. (2023) covers trends and advances in NLP for code-switched text, including main fields of interest and future research lines. Doğruöz et al. (2021) explain advances in applications of language technologies for code-switched text from a linguistic and social perspective.

Datasets & benchmarks for CS. The majority of code-switched data is obtained from social media, and other popular data sources include recordings and transcriptions (Winata et al., 2023). There have been several shared tasks that deal with CS, for the tasks of Language Identification (Solorio et al., 2014; Molina et al., 2016) and Sentiment Analysis (Patwa et al., 2020). Two popular benchmarks have been created to answer the demand for evaluation of CS that covers different language pairs and tasks: LINCE (Aguilar et al., 2020), which covers traditional tasks such as Part Of Speech tagging (POS) or Sentiment Analysis (SA); and GLUECoS (Khanuja et al., 2020), which incorporates NLU tasks for the Hindi-English pair. As of today, GLUECoS cannot be used without access to the X API.

CS generation. CS generation has seldom been tackled in previous research. Approaches include using linguistically informed techniques that aim to find out plausible switching points (Pratapa et al., 2018; Gupta et al., 2020; Gregorius and Okadome, 2022), data augmentation (Tarunesh et al., 2021) and, more recently, prompting LLMs for CS generation (Yong et al., 2023). To the best of our knowledge, there is no previous research on CS generation with natural CS as a starting point.

3 Parallel Data Creation

In this work we present a novel approach to generate code-switched text from monolingual sentences. As a first step, we create a synthetic parallel corpus from an initial set of English-Spanish CS sentences with their English monolingual equivalents, generated by the Command R model (Cohere For AI, 2024). We exploit the fact that LLMs struggle to generate CS text given a monolingual sentence, but are able to more reliably convert a CS sentence to its corresponding monolingual version, especially when the target language is English. After having created this pseudo-parallel corpus, we use it to fine-tune LLMs on the task of conditional code-switching generation, presented in Section 4.

3.1 The LINCE benchmark

We use LINCE as a starting point, a popular benchmark that has been widely used to evaluate CS systems (Aguilar et al., 2020), which is available in 6 language pairs. This benchmark contains annotations for 5 different tasks: Language Identification (LID), Part Of Speech tagging (POS), Named Entity Recognition (NER), and Sentiment Analysis (SA). All sentences in LINCE are tokenized, and each token is annotated with a language tag as well as other categories depending on the task. In our work we focus on the English-Spanish pair and filter all sentences in the data that do not contain CS, similarly discarding all the task-specific annotations. Example 2 shows a random instance from LINCE.

(2)	estaba	aquí	three	feet	away	.
	spa	spa	eng	eng	eng	eng&spa

LINCE comprises around $95,000$ train, $20,000$ development, and $33,000$ test instances for the English-Spanish pair. We deduplicate the instances among splits, and filter and pre-process the instances to ensure that they are suitable for our task by removing links, replacing usernames with the placeholder <*user*>, and detokenizing all instances with the script provided as part of the Moses toolkit (Koehn et al., 2007). After this preprocessing, we obtain a more natural version of the LINCE data. A preliminary analysis reveals that many sentences in LINCE are monolingual or contain a single word in one language that often correspond to a borrowing, as shown in Example 3. In order to ensure that all of our sentences actually contain CS, we filter sentences that do not have at least two words in each language.

(3) I need a shot of tequila or a glass of scotch to keep me warm right now.

After these pre-processing and filtering steps, we end up with $27,684$ train, $2,461$ development and $5,353$ test instances. The comparison between the original size of LINCE and the final number of sentences selected for our experiments after pre-processing is shown in Table 1.

	Train	**Dev**	**Test**
Original	94,728	19,574	33,361
Pre-processed	27,684	2,461	5,353
EN2CS	10,703	792	1,040

Table 1: Size of original LINCE (EN-ES) compared to the automatically filtered instances and the final set of parallel instances, dubbed *EN2CS*.

3.2 *EN2CS*

The next step in our method requires creating a pseudo-parallel English-CS dataset by translating the natural code-switched instances into monolingual text. As there are no available machine translation systems to convert from English-Spanish CS text to English monolingual text, we instead make use of prompt engineering, using the Command R model (Cohere For AI, 2024), one of the strongest publicly available models at the time.

We perform an initial set of experiments to determine the optimal prompt to generate monolingual English versions of the code-switched data. Ideally, we aim for a prompt that generates translations that maintain the meaning of the original sentences, are fluent and natural, whose grammar is correct and that does not contain any Spanish words or phrases. The tested prompts are listed and explained in Appendix A, and the prompt that generates outputs closest to the desired ones is: *Now convert this code-switched phrase to English. Leave the parts in English as they are, focus on translating the parts in Spanish.* Preliminary experiments also showed that a few-shot strategy helps the model

	Original	English
Silver	you just have to tell me *que como te va*.	You just have to tell me *how it's going.*
	osea i know we wanna party *pero tampoco no aya asta dallas*	like i know we want to party *but not all the way to dallas*
	fully *enamorada de mis* eyelash extensions	I'm totally *in love with my* eyelash extensions
Gold	*hasta venir a plaza se siente* like home.	*even coming to the square feels* like home.
	me siento tan pendejo right now.	*i feel so stupid* right now.
	y no te dan problemas as long as you put that it was frozen.	*and they don't give you problems* as long as you put that it was frozen.

Table 2: Examples of the *EN2CS* parallel corpus. Left: original code-switched instances, right: generated (silver) or post-edited (gold) English instances.

for the task at hand, so the prompt is enriched with 5 manually selected examples that show how the output should look like.

The generated monolingual sentences are further processed, and instances that contain profanity or that are prefaced with a direct mention of the task, such as “Of course, here's your translation:” are discarded.

In order to create a true gold standard test set to evaluate a CS generation model, we perform a manual post-edition of the the monolingual test translations for $1,040$ instances of the LINCE test set. The post-edition was carried out by three proficient speakers of English and Spanish, who were provided with specific guidelines as shown in Appendix B.

Table 1 shows the final size of the parallel corpus, which we dub *EN2CS*, after post-processing and post-edition, and Table 2 shows examples of silver and gold instances. The final version of our dataset therefore contains $10,703$ train and 792 development instances with automatically translated English sentences matched to their original CS sentences, and $1,040$ gold instances with corrected English translations.

4 CS generation experiments

With *EN2CS* as our starting point, we frame CS generation as a machine translation task, with English as the source and CS as the target language, where parts of the source sentence have to be translated to Spanish. In our experiments we try four generative models, namely, Llama3, Llama3 Instruct (Dubey et al., 2024), Mistral and Mistral Instruct (Jiang et al., 2023).[3] All the models are trained with the causal language modelling objective, but we use different input formats for the base

[3]Llama3 models are used in their 8B size (Base and Instruct). Mistral models are used in their 7B size, version 0.3 (Base and Instruct).

Base model
I want to not work and make money. = quiero no trabajar and make money
Instruction-tuned model
system prompt: "*You are a bilingual speaker of English and Spanish. Translate the following English sentence into code-switched text between both languages:*"
user: "*I want to not work and make money.*"
assistant: "*quiero no trabajar and make money*"

Table 3: Examples and format of prompts used for fine-tuning base and instruction-tuned models

and instruct models. For base models we follow (Zhu et al., 2024) and use templates in the form of “`<X>=<Y>`”, where `<X>` and `<Y>` are placeholders for the input English sentence and generated CS, respectively. At inference, the second code-switched part is left empty for the model to fill. For fine-tuning instruction-tuned models, we provide them with a system prompt that contains the instruction, a query by the user in English, and an answer from the assistant with the code-switched target. At inference time, the same system prompt is used, and the user prompt contains the English sentence, so that the model generates the assistant part. Table 3 shows examples of the format used for fine-tuning base and instruction-tuned models.

All models are trained using Quantized Low-Rank Adaptation (QLoRA) (Dettmers et al., 2023), to ensure memory and parameter efficiency, with standard parameters: the model is loaded in 4 bit with NF4 quantization data type and bf16 computational data type. The LoRA rank and scaling factor are set to 16 and the dropout to 0.05. We apply the LoRA update matrices to the attention blocks and do not train bias parameters. Regarding the hyperparameters, we only tune the learning rate ($1e^{-4}$, $5e^{-4}$, $1e^{-3}$ and $5e^{-3}$) and training epoch $\in [1 \dots 10]$, choosing the parameters that give the lowest cross-entropy loss on the development set for each model.

Model	Generated Output
Original (Gold)	damm *todos se casaron* and we still single lol forever alone
English (Source)	damn everyone got married and we're still single lol forever alone
MT Baseline	damn everyone got married and we still be single lol
Llama3	damn *todos se fueron a casarse y nosotras estamos solitarias* lol forever alone
Llama3 Instruct	damm every1 got married and we're still single lol *alonso solit@o* foreverrrr lolololo
Mistral	damm everyone got married *y ni somos esa* lol ofrever alone
Mistral Instruct	dude *todo el mundo se casó y nosotros hemos quedado solteros* lol

Table 4: Example from the test set and the generated outputs of the different models.

We use the transformers package (Wolf et al., 2020) for all training experiments.

Early experiments indicated that the models' outputs are longer than expected and usually produce the desired output up to a punctuation mark and then either begin to translate the sentence again or hallucinate more content. We therefore truncate the output up to a punctuation mark where the length is closest to that of the original sentence. We additionally experimented with the *length_penalty* and *exponential_length_decay* generation parameters, as well as trying to control the length of the generation with length codes, but find that the truncation heuristic performs the best. Accordingly, all further experiments will use the truncated output. This overgeneration problem has been reported in previous papers, where similar truncation strategies have been adopted (Bawden and Yvon, 2023).

We also include a dedicated encoder-decoder model as a baseline, trained on *EN2CS* using the MarianNMT toolkit (Junczys-Dowmunt et al., 2018), with a mini-batch size automatically selected for the available memory[4]. Optimization is performed using Adam (Kingma and Ba, 2015), with $\alpha = 0.0003$, $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$, using a standard learning rate of $3e^{-4}$. Validation is conducted every 100 steps, and training stops if perplexity showed no improvement after 5 consecutive checkpoints.

Table 4 shows an example of the outputs of the different models, compared to the original code-switched sentence and the English monolingual sentence that they received as input.

[4]We used two GPUs and $8GB$ per GPU.

5 Qualitative evaluation

As a first step to assess the quality of the outputs produced by the different models, we perform a manual qualitative analysis of the results in two parts: a pairwise tournament-based human evaluation, and an in-depth analysis of the most common errors made by the models and their distribution.

5.1 Preference based evaluation

We perform a tournament-based evaluation that allows us to determine the ranking of models in terms of human preference. A total of 660 instances are matched against each other, corresponding to the outputs of the five models for 110 English source sentences, as well as the gold standard reference. The evaluation is conducted pairwise, requiring annotators to choose the best out of two sentences or declare a tie. When choosing the best sentence, annotators do not know the original English sentence, nor which model produced what output. This process results in $110 \cdot \binom{6}{2} = 1,650$ comparisons, and was carried out by 11 annotators, with each annotator performing 150 random comparisons.

Annotators are provided with a series of criteria to choose between the instances, devised after the error analysis described in the next section. They must take into account three main criteria, that must be applied in the following order: a) the presence and naturalness of the CS; b) the content and fluency of the sentences; and c) the orthographical errors of the instances (correct punctuation, presence of typos, etc.). Annotators are furthermore asked to avoid declaring ties, unless completely necessary (e.g., in a case where both sentences are completely monolingual and therefore equally incorrect), to compel them to develop a preference. The complete annotation guidelines are available in Appendix C.

Ranking	Model	Score
🥇	Gold Standard	392.5
🥈	Llama3	325.5
🥉	Llama3 Instruct	303.0
4	Mistral	285.5
5	Mistral Instruct	242.0
6	MT Baseline	101.5

Table 5: Ranking of models according to the human preference score.

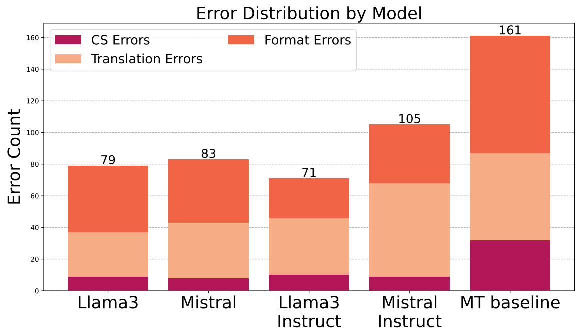

Figure 1: Error distribution by model, obtained by counting the number of instances that present errors of each type.

With the results of the pairwise comparisons, we calculate a global score for each model, as follows: every time a model is voted, it gets 1 point, and the loser gets 0 points; in case of ties, both models get 0.5 points each. Table 5 shows the global scores, as well as the ranking of human preferences according to said score. We find that the gold standard reference obtains the highest score, as expected, and that Llama3 ranks the highest among the automatic methods. Instruction-tuned models obtain worse scores compared to their base model counterparts, with a similar difference for both Llama3 and Mistral family of models. Finally, the evaluation also shows that humans clearly prefer the output of systems based on Large Language Models compared to the MT baseline, which is voted as the worst model by a large margin.

5.2 Error analysis

In order to further explore differences between model performance, we analyse the most common errors made by the CS generation models, both quantitatively and qualitatively. We adapt the methodology presented in Popović (2018), who proposes different typologies for machine translation errors, and extend it to CS generation error analysis. To do so, we randomly select a set of 100 outputs from all models and conduct a detailed examination of the types of errors present in them. This thorough analysis allows us to identify recurring patterns and propose a refined error typology specifically for automatic CS generation. This initial error analysis yields 18 total error categories, which we simplify and group into three main error types: a) CS errors, b) Translation Errors, and c) Format errors. The full error typology, along with detailed descriptions for each error type, is provided in Appendix D, while here we explain the three error categories:

CS Errors: Errors of sentences that are either completely monolingual or switch between languages in an unnatural manner, e.g., by repeating the same word in English and Spanish. In Example 4, Llama3 Instruct preserves the original meaning, but the sentence is fully monolingual.

(4) **Source** After all these things when we're done.
Output after all these things when we're finished

Translation errors: Critical errors that either change the original meaning of the sentence or introduce mistakes in fluency or grammar, for example, using the wrong tense or word order. Example 5 shows an instance where Mistral Instruct outputs a seemingly natural code-switched sentence, but the phrase "they got hurt" is not adequately translated and the meaning of the sentence is not preserved.

(5) **Source** I wasn't happy because they got hurt.
Output *no estuve* happy *porque me dieron mal*

Format errors: Errors in form that do not make the sentences unintelligible nor change their meaning, such as repetitions of a word or phrase or incorrect punctuation. Example 6, by the model Llama3, accurately preserves the original meaning and introduces CS, but removes the username and adds a smiley face.

(6) **Source** <user> old mexican remedies
Output old school *remedios mexicanos* :)

Model	BLEU	BERTScore	chrF
Llama3	**34.49**	81.64	**53.17**
Llama3 Instruct	33.42	**81.77**	52.01
Mistral	31.65	80.93	50.56
Mistral Instruct	25.98	78.66	44.58
MT Baseline	20.21	76.38	33.79

Table 6: Results of automatic metrics the *EN2CS* test set. Best results in bold, second best results underlined.

We classify 500 additional instances (100 instances per model, obtained from the same source sentences) into these kind of errors, and show the results in Figure 1. The analysis indicates that the baseline MT system has the worst performance, with nearly twice as many errors as the best models. This is in line with the results of the preference-based evaluation, where MT ranked last. The most frequent errors made by the MT system — word repetition, missing information, and incorrect translations — are particularly problematic, as they often make the output unintelligible or incomplete. Among the pre-trained models, Llama models make the fewest errors overall, with 19 fewer errors on average. Base models of both families struggle mainly with format errors, which make up 50.68% of their errors on average, whereas instruction-tuned models present more meaning-related issues, $53,45\%$. This suggests that the linguistic knowledge of the models degrade when tuned on instructions, a phenomenon that has been observed on other related areas (Fu et al., 2024). CS-related mistakes are the least common in all models, accounting for less than 15% of the overall error count. It seems that the models have effectively learned to switch between languages naturally, though they may still be prone to other types of errors.

6 Automatic Evaluation

We perform a quantitative evaluation using traditional metrics used in NLG. To that end, we use BLEU (Papineni et al., 2002), BERTScore (Zhang et al., 2020), and chrF (Popović, 2015), implemented with the evaluate library. All three are reference-based task-agnostic quality metrics that give results between 0-1, based on character-level F-score, n-gram precision and semantic similarity using contextual embeddings[5] respectively.

[5]BERTscore has been calculated using the embeddings from the model Bert Base Multilingual Cased.

The results of the evaluation can be seen in Table 6. The best two models are Llama3, with the highest BLEU and chrF, and Llama3 Instruct, with the highest BERTScore. They are closely followed by less than 2 points in all metrics by Mistral, the third best model overall. Finally, Mistral Instruct is the pre-trained model with the lowest results.

The MT baseline obtains the lowest results overall, which is consistent with the qualitative evaluation described above, where the MT baseline proves to be the worst system by a large margin. However, it is worth noting that not all metrics capture this gap in performance, since, according to BERTScore, there is only a 2 point difference between the MT baseline and Mistral, which is the pre-trained LLM that obtains the worst metrics. It is also interesting to compare these results with the error analysis in Section 5.2. For instance, Mistral Instruct yields low values for BLEU and chrF, in line with its number of translation errors, as depicted in Figure 1. However, automatic metrics fail to capture the fact that CS errors produced by language model based systems are relatively low.

6.1 Correlation With Automatic Metrics

The automatic metrics used in the section before are known to have poor correlation with human judgment in NLG tasks (Sai et al., 2022), and in this section we analyse whether this poor correlation also occurs when evaluating CS generation. To that end, we compare the automatic metrics results with the preference-based scores obtained in Section 5.1.

We calculate Pearson's (ρ) correlation coefficient at instance-level, using the 550 instances employed for the error classification and human evaluation (the output of 5 models for 110 source sentences)[6]. Each data point corresponds to the CS output of one particular model for an English source sentence, and we compute the correlation using two values: the score obtained by the model for this instance in the human preference-based evaluation of Section 5.1, and the score it attains if we apply the same strategy using the values of the automatic metrics to determine the winner.

The correlation coefficients are shown in Figure 2. The top part of the figure shows the correlation using all the instances, whereas the bottom part only considers those instances that showed

[6]We do not consider the reference CS sentence when calculating the correlations.

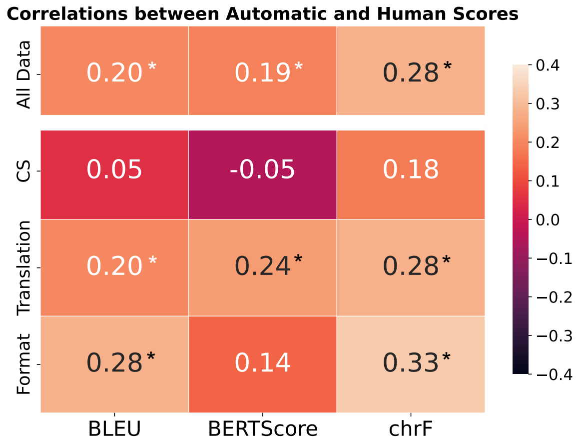

Figure 2: Heatmap of the correlations between human scores and automatic metrics, calculated using the Pearson Correlation Coefficient. The correlations are calculated for all instances, as well as for different subsets of instances, according to the type of errors they exhibit. * indicates statistical significance ($p \leq 0.05$).

some type of error, according to the error analysis described in Section 5.2. If we consider all the instances, the maximum ρ correlation value is 0.28, which indicates a low alignment with the scores assigned by automatic metrics and human judgments. The metric with highest correlation is chrF, which seems to confirm previous research that reports a higher correlation with human judgement than other metrics (Popović, 2015).

Regarding the correlations according to the error types, all metrics obtain the highest correlation on errors related to meaning and format, which is expected, as they were originally designed to evaluate MT systems. However, instances with CS errors show the lowest correlation overall, with only chrF showing ρ values above 0.05. Human evaluators were explicitly asked to never choose instances without CS, but automatic metrics are not sensitive to these nuances, and may assign high scores to instances regardless of whether they contain CS or not.

All in all, these results confirm that several of the most commonly used automatic metrics for NLG have a poor correlation with human judgments when evaluating CS generation. This underscores the need to research more specialized evaluation methods which are designed specifically to capture the nuances of this task.

7 Conclusion

In this work, we have presented a methodology to leverage LLMs in the generation of code-switched text from monolingual instances, specifically for the English-Spanish language pair.

Our framework consists of back-translating natural code-switched instances (EN-ES) into monolingual English sentences, and using the resulting parallel corpus, dubbed *EN2CS*, to fine-tune autoregressive models to turn monolingual sentences into CS. This approach has the potential to improve the naturalness of CS generation, as the gold standard CS text is not artificially generated.

We have provided an extensive evaluation of the results of our models. On the one hand, we perform a human evaluation of the outputs of the models with two parts: a human preference based evaluation and an error analysis of a subset of the test sentences, where we find out three types of errors: CS errors, errors in meaning and format errors. On the other, we employ popular NLG metrics to automatically evaluate the results of our models, and calculate the correlation between both types of evaluation.

Both the automatic metrics and the analysis of the outputs of the models, as well as the human evaluation, show that, when fine-tuning the models with LoRA, base models work better than their instruction-tuned counterparts for this task, and that the Llama3 family obtains better scores than the Mistral family. This could be an example where instruction tuning degrades the base model's linguistic ability (Fu et al., 2024) or alternatively could be related to differences in how LoRA fine-tuning affects each model type.

Our analyses show low correlation between human and automatic evaluations, particularly in cases with CS errors. This suggests that current metrics are not adequate for assessing CS generation, which would require more specialized evaluation methods.

We conclude that pre-trained models are able to yield competent results and generate satisfactory outputs, as the error analysis shows the less abundant type of errors are those related to CS, as opposed to problems in fluency, retaining the original meaning of the sentences, and errors in format. Human evaluation shows that their generations are still not on a par with the original instances.

Limitations

Our research focuses on testing the capabilities of LLMs for CS generation, a field of interest in the research of many applications, yet still in need of more research. While our findings highlight promising potential, we also identify key areas for refinement and improvement, as well as promising lines for future research in this domain.

We only perform an in-domain evaluation where the train, validation and test sets had the same origin. Additionally, we would like to test the efficiency of our models in an out-of-domain setting, since one of the use-cases of a CS generation model is to create parallel corpora to evaluate the abilities of models to perform different tasks when there is CS.

One of the key points of our research is using open-weight LLMs, however, the use of bigger, more powerful restricted-weights LLMs could very plausibly yield better results, even in a zero-shot scenario (Huzaifah et al., 2024).

We want to acknowledge the fact that our approach is dependent on having an initial set of code-switched sentences, which may not be available for all pairs of languages, especially in a low-resource scenario. We believe that it would be interesting to explore the possibility of a cross-lingual approach using our methodology, with English and/or Spanish as pivot languages, that could be useful for transfer knowledge into other less-resourced language pairs.

Finally, as we have pointed out, we are aware of the problems of the automatic metrics that we have used to evaluate the outputs of our models, that do not capture the nuances of our task. In the future, we would like to investigate how to improve this evaluation by designing new methods to automatically evaluate CS generation, focusing on a more linguistic approach able to capture the linguistic and social intricacies of CS.

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.

Rene Appel and Pieter C. Muysken. 2005. *Language Contact and Bilingualism*. Amsterdam University Press.

Kalika Bali, Jatin Sharma, Monojit Choudhury, and Yogarshi Vyas. 2014. "I am borrowing ya mixing ?" an analysis of English-Hindi code mixing in Facebook. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 116–126, Doha, Qatar. Association for Computational Linguistics.

Anshul Bawa, Pranav Khadpe, Pratik Joshi, Kalika Bali, and Monojit Choudhury. 2020. Do multilingual users prefer chat-bots that code-mix? let's nudge and find out! *Proc. ACM Hum.-Comput. Interact.*, 4(CSCW1).

Rachel Bawden and François Yvon. 2023. Investigating the translation performance of a large multilingual language model: the case of BLOOM. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 157–170, Tampere, Finland. European Association for Machine Translation.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Cohere For AI. 2024. c4ai-command-r-v01 (revision 8089a08).

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms.

Jean-Marc Dewaele and Li Wei. 2014. Attitudes towards code-switching among adult mono- and multilingual language users. *Journal of Multilingual and Multicultural Development*, 35(3):235–251.

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.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Tingchen Fu, Deng Cai, Lemao Liu, Shuming Shi, and Rui Yan. 2024. Disperse-then-merge: Pushing the limits of instruction tuning via alignment tax reduction. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2967–2985, Bangkok, Thailand. Association for Computational Linguistics.

Hila Gonen and Yoav Goldberg. 2019. Language modeling for code-switching: Evaluation, integration of monolingual data, and discriminative training. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4175–4185, Hong Kong, China. Association for Computational Linguistics.

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.

Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2020. A semi-supervised approach to generate the code-mixed text using pre-trained encoder and transfer learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2267–2280, Online. Association for Computational Linguistics.

Maite Heredia, Marianela Fernández Trinidad, and Miguel Jiménez-Bravo. 2025. Actitudes lingüísticas hacia el cambio de código entre valenciano y castellano. *Revista Española de Lingüística Aplicada/Spanish Journal of Applied Linguistics*.

Muhammad Huzaifah, Weihua Zheng, Nattapol Chanpaisit, and Kui Wu. 2024. Evaluating code-switching translation with large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 6381–6394, Torino, Italia. ELRA and ICCL.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Simran Khanuja, Sandipan Dandapat, Anirudh Srinivasan, Sunayana Sitaram, and Monojit Choudhury. 2020. GLUECoS: An evaluation benchmark for code-switched NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3575–3585, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, CA, USA.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona Diab, Veselin Stoyanov, and Xian Li. 2022. Few-shot learning with multilingual generative language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9019–9052, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Giovanni Molina, Fahad AlGhamdi, Mahmoud Ghoneim, Abdelati Hawwari, Nicolas Rey-Villamizar, Mona Diab, and Thamar Solorio. 2016. Overview for the second shared task on language identification in code-switched data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 40–49, Austin, Texas. 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.

Dwija Parikh and Thamar Solorio. 2021. Normalization and back-transliteration for code-switched data. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 119–124, Online. Association for Computational Linguistics.

Parth Patwa, Gustavo Aguilar, Sudipta Kar, Suraj Pandey, Srinivas PYKL, Björn Gambäck, Tanmoy Chakraborty, Thamar Solorio, and Amitava Das. 2020. SemEval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 774–790, Barcelona (online). International Committee for Computational Linguistics.

Shana Poplack. 1980. Sometimes i'll start a sentence in spanish y termino en espaÑol: toward a typology of code-switching 1. *Linguistics*, 18:581–618.

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.

Maja Popović. 2018. *Error Classification and Analysis for Machine Translation Quality Assessment*, pages 129–158. Springer International Publishing, Cham.

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.

Sai Krishna Rallabandi and Alan W. Black. 2017. On building mixed lingual speech synthesis systems. In *Interspeech*.

Ananya B. Sai, Akash Kumar Mohankumar, and Mitesh M. Khapra. 2022. A survey of evaluation metrics used for nlg systems. *ACM Comput. Surv.*, 55(2).

E. Sarkisov. 2021. Interlingual interference as a linguistic and cultural characteristic of the current online communication. *Russian Journal of Bilingualism Studies*, 3:16–21.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Ghoneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, and Pascale Fung. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 62–72, Doha, Qatar. 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.

G Richard Tucker. 2001. A global perspective on bilingualism and bilingual education. *Georgetown University Round table on Languages and Linguistics 1999*.

Genta Winata, Alham Fikri Aji, Zheng Xin Yong, and Thamar Solorio. 2023. The decades progress on code-switching research in NLP: A systematic survey on trends and challenges. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2936–2978, Toronto, Canada. Association for Computational Linguistics.

Genta Indra Winata, Samuel Cahyawijaya, Zihan Liu, Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2021. Are multilingual models effective in code-switching? In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 142–153, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Jitao Xu and François Yvon. 2021. Can you traducir this? machine translation for code-switched input. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 84–94, Online. Association for Computational Linguistics.

Zheng Xin Yong, Ruochen Zhang, Jessica Forde, Skyler Wang, Arjun Subramonian, Holy Lovenia, Samuel Cahyawijaya, Genta Winata, Lintang Sutawika, Jan Christian Blaise Cruz, Yin Lin Tan, Long Phan, Long Phan, Rowena Garcia, Thamar Solorio, and Alham Fikri Aji. 2023. Prompting multilingual large language models to generate code-mixed texts: The case of south East Asian languages. In *Proceedings of the 6th Workshop on Computational Approaches to Linguistic Code-Switching*, pages 43–63, Singapore. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert.

Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, and

Lei Li. 2024. Multilingual machine translation with large language models: Empirical results and analysis. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2765–2781, Mexico City, Mexico. Association for Computational Linguistics.

A Prompt Tuning

We test the prompts in Table 7, combined with 0-, 1- and 5-shot strategies. The prompts include the instructions explained in different ways, including more or less information.

For the few-shot strategies, the prompt includes the following template at the beginning, alongside a set of manually selected examples:

Here are {n} examples of a code-switched text that has been converted to {lang}:

Testing the different prompts, we are able to choose the one whose outputs are closest to our needs, taking into consideration the trade-off between including too little and too much level of specificity in the instructions to the models.

Regarding the few-shot strategies, we find out that giving some examples to the models results in outputs that are more aligned with the expected output, which is logical, since this allows the models to more faithfully replicate the examples provided. The more examples given, the more the model is able to comply to leaving the punctuation marks as they are and not standardizing the spelling, but also it tends to add more colloquial terms and alternate spellings.

B Post-edition Guidelines

The original sentence should contain **CS** and be **translatable**. The main reasons to **remove** an instance altogether are:

- If the sentence is very clearly monolingual and the CS has been detected incorrectly (eg, the case of interlingual homographs such as *has*).
- When the sentence is bilingual for metalinguistic reasons, because it makes the translation tricky and hard to understand, and in most cases it's not even CS.
- The part that is in the other language is a named entity, such as a title, a name, . . .
- If the code-switched part is not translatable or very hard to translate, probably because it's a borrowing. Ambiguous and a little bit up to the annotator.
- If the tweet is saying the same thing in both languages (making it monolingual doesn't make sense).
- Some instances are tweets that are part of a conversation or thread and taken out of context are very hard to understand/intelligible.
- Some tweets are not translatable because of wordplay that doesn't transfer to monolingual speech.

The result should be a **monolingual** sentence that has roughly the **same meaning** as the original sentence. The main reasons to edit a translation are:

- If the meaning changes or the model has hallucinated extra information that wasn't present in the original sentence.
- If there are still some words in the Spanish.
- Attempts to translate named entities.
- Remove "meta comments" from the model about the task.

It is not necessary to correct things like:

- Punctuation marks.
- Different spellings of the same word.
- Words of phrases that the model has changed for synonyms.

C Pairwise Annotation Guidelines

The main objective of this task is two evaluate a pair of sentences that **should contain code-switching between English and Spanish**. It should be noted that models have been trained with texts extracted from social media and informal conversations, so **the outputs of the models are expected to present traits of informality**, such as common typos, that at first should not be considered errors, because they are within the expected behaviour of the models. The criteria to choose between both sentences is to be applied **in the following order**:

1. **Code-switching**

Convert this code-switched phrase to English.
Convert this code-switched phrase to English without correcting the original spelling, focus on translating the parts in Spanish.
Convert this code-switched phrase to English. Leave the parts in Spanish as they are, focus on translating the parts in Spanish.
Convert this code-switched phrase to English. Directly output the translation and don't correct the original spelling, focus on translating the parts in Spanish.

Table 7: Different prompts that have been used to convert the code-switched instances into English, with different levels of specificity. Final prompt in bold.

- 1.1. **Presence of code-switching**: For a sentence to be a suitable candidate it must have tokens in both languages. A completely monolingual sentence will always be wrong.
- 1.2. **Naturalness of the code-switching**: A switch between both languages can be unnatural. There are different linguistic constraints. For example, a switch is only possible at a point in a sentence where it does not violate the syntactic rules of either language.

2. **Content and fluency**
   - 2.1. **Content**: Sentences must have meaning as a whole, they have to be understandable, without extra content disconnected from the rest of the message or abrupt interruptions.
   - 2.2. **Agreement**: Sentences must have the right gender and number agreement.
   - 2.3. **Conjugation**: Verbs have to be correctly conjugated.

3. **Form**: Additional errors that can be used in case none of the above are applicable.
   - 3.1. **Repetitions** of the same word or phrase.
   - 3.2. **Misspelled words / uncommon typos**
   - 3.3. **Wrong punctuation marks**
   - 3.4. **Extra characters**

Ties are only contemplated in two situations:

- Two sentences that are **equally wrong**, that is to say, they are both either completely monolingual or unintelligible.
- Two sentences that are **exactly the same** and thus no criteria can be used to break the tie.

In case no criteria is applicable to a pair, we ask the annotators to choose their preferred sentence, using their own judgement o additional criteria they might observe in the specific pair of sentences.

D Error Typology

1. **CS errors**
   - 1.1. **No CS** - the sentence is entirely monolingual.
   - 1.2. **Unnatural CS** - the sentence contains unnatural CS, either due to unnatural switching points, or unnatural register.
   - 1.3. **Repetition in both languages** - the sentence contains the same information repeated in both languages, rather than CS.

2. **Translation errors**
   - 2.1. **Made-up words** - the words in the output look like English or Spanish but do no actually exist.
   - 2.2. **Wrong translation** - the translation of a word or phrase is incorrect.
   - 2.3. **Wrong conjugation** - a verb is translated with the right lexeme but a seemingly made-up conjugation.
   - 2.4. **Wrong agreement** - there is a mistake in agreement in gender or number.
   - 2.5. **Wrong meaning** - a word or phrase has been translated into a sense that does not fit into the context.
   - 2.6. **Wrong order** - the words are right but they are written in the wrong order.
   - 2.7. **Wrong tense** - the verbal tense is not consistent through the sentence.
   - 2.8. **Unintelligible** - it is not possible to understand the sentence in English nor in Spanish.

2.9. **Instruction misunderstanding** - the task has been misunderstood, e.g., the model makes a "comment" about the content of the output or explains a word.

3. **Format errors**

   3.1. **Extra words** - the sentence contains seemingly random extra words that do not affect its meaning.

   3.2. **Extra characters** - the sentence contains more non-word characters than the original, e.g., ‘???’ instead of ‘??’.

   3.3. **Hallucinations** - the sentence contains new words or phrases not derived from the original text.

   3.4. **Start over** - the sentence is finalized, but the model begins a second translation of the same sentence.

   3.5. **Duplications** - some words or phrases of the sentence are duplicated.