Evaluating Synthetic Code-Switched Data

ABHISHEK VIJAYAKUMAR, Language Technologies Institute, Carnegie Mellon University, USA

Multilingual code-switching is the practice of mixing multiple languages within a single utterance. Modern approaches to code-switching language tasks rely on fine-tuning pretrained multilingual language models with code-switched data. As readily-available natural code-switched data is extremely limited, current approaches involve generating large amounts of synthetic code-switched data and using this data for language model fine-tuning. We seek to provide an evaluation of the quality of such synthetic code-switched data with respect to the objective of fine-tuning language models. We examine an existing benchmark, Microsoft India's GLUECoS, containing several Hindi-English (Hinglish) code-switched task evaluations. While GLUECoS indicates that fine-tuning with code-switched data can improve task performance, we observe that under a more robust framework, GLUECoS does not provide statistically significant differentiations between many fine-tuned language models. We demonstrate the difficulty of differentiating between models fine-tuned with different code-switched datasets using GLUECoS and propose modifications to make it an effective quality evaluation for synthetic code-switched data.

CCS Concepts: • **Computing methodologies** → **Natural language generation**.

Additional Key Words and Phrases: code-switching, multilingual, text generation, evaluation

ACM Reference Format:
Abhishek Vijayakumar. 2022. Evaluating Synthetic Code-Switched Data. 1, 1 (May 2022), 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Multilingual code-switching is a form of communication in which multiple natural languages are mixed within a single sentence or utterance. Multilingual speakers will embed words or phrases in an *embedding language* into a sentence primarily in a *context language* [Bhat et al. 2016]. This form of communication is especially common in informal conversations or in informal digital contexts such as on Twitter or other social media. As code-switching becomes more common, there is growing interest in developing language technologies capable of processing code-switched data and performing traditional tasks such as sentiment analysis and question answerinng.

However, existing monolingual language technologies are not equipped to handle code-switched text. Multilingual language technologies perform better, but still produce subpar results when compared to results on monolingual data, even when using state-of-the-art BERT-based models [Wang et al. 2018] [Khanuja et al. 2020]. There is thus much research around improving language technologies for code-switched tasks. Following the state-of-the-art for monolingual language technologies, such research focuses on creating language models for code-switched text which can be adapted to specific tasks with different head architectures.

Due to the lack of readily available code-switched text, stemming from its informal nature and comparatively low usage, it is not feasible to train a language model on natural code-switched text alone. A common approach to addressing this lack of data is the generation of synthetic code-switched text. Using a little to no real code-switched text as well as resources in the context and embedding language, it is possible to generate sufficient amounts of synthetic code-switched text to fine-tune a multilingual language model into a code-switching language model. However, it must be verified that such generated code-switched text accurately represents real code-switched data.

We implement and adapt several methods of generating code-switched data and evaluate generated code-switched data on several commonly used evaluations: Code-Mixing Index [Gambäck 2014], self-BLEU [Zhu et al. 2018], BERTScore [Zhang et al. 2020], and GLUECoS [Khanuja et al. 2020], a suite of task-driven evaluations developed by Microsoft India. We adapt several tasks in the GLUECos evaluation to provide a more robust and more effective discriminator of performance between different language models. We focus on the Hindi-English (Hinglish) code-switched language pair.

2 GENERATING CODE-SWITCHED DATA

We classify methods for creating code-switched data as either *substitutive* or *generative* methods. Substitutive methods involve taking monolingual sentences in the context language and replacing some phrases with phrases in the embedding language. Generative methods involve creating new sentences without explicit source sentences.

2.1 Substitutive Methods

Substitutive methods can be defined with respect to an overarching framework consisting of 3 steps: selection, translation, and reconstruction. We implement several substitutive methods by implementing components for these 3 processes.

The selection step involves choosing tokens in the input sentence to translate. Tokens may be either chosen as individual words or as longer spans. We implemented two selection methods. The first, token selection, randomly selects tokens with some probability to translate. The second, span selection, randomly selects a span length and section of the input to translate.

The translation step involves translating each selected token or span from the context language into the embedding language. We implemented two translation methods. The first uses a bilingual lexicon to convert in-vocabulary tokens to the embedding language. If a word is not in the dictionary, it remains as its context language token. The second applies Google Translate [Google 2022] to translate each given span, taking the most likely candidate for each translation.

Author's address: Abhishek Vijayakumar, Language Technologies Institute, Carnegie Mellon University, Pittsburgh, USA, abhishev@andrew.cmu.edu.

XXXX-XXXX/2022/5-ART $15.00
https://doi.org/XXXXXXX.XXXXXXX

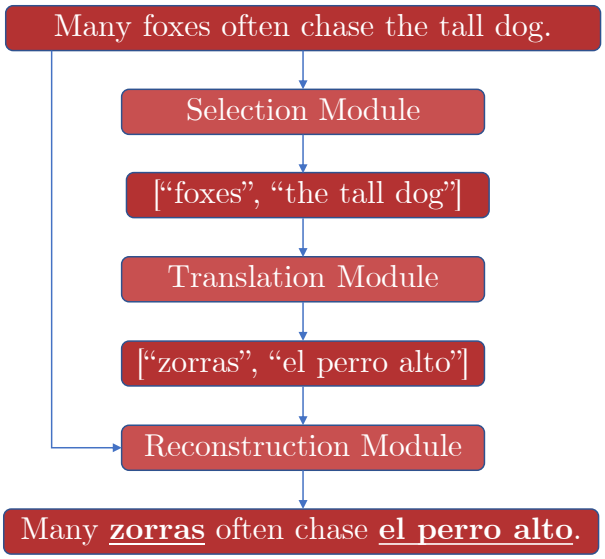

Fig. 1. Substitutive generation process

The reconstruction step involves taking the translated spans and incorporating them into the context sentence. The method we used for this step was replacing the source spans with the translated spans, then removing all punctuation from the sentence to normalize with respect to the lack of punctuation produced by the translation step.

We note that random span selection with bilingual parallel sentences is the methodology used for code-switching pretraining by Yang et al. [Yang et al. 2020]. Xu and Yvon's method of generation corresponds to random token selection and replacement [Xu and Yvon 2021].

We also evaluate data provided by Akruti Kushwaha from the Language Technologies Institute at Carnegie Mellon University. This data was generated with a machine translation approach to moving between monolingual source sentences and code-switched sentences, where the code-switching language is treated as a target language for a translation task. The source data was known as the DoG dataset, and consisted of several paired human translated code-switched and monolingual sentences and many unpaired monolingual sentences. We worked with machine-translated variants of the unpaired sentences.

2.2 Generative Methods

Generative methods train on some amount of real code-switched and monolingual data, then produce new sentences which do not correspond to specific inputs [Chandu and Black 2020] [Samanta et al. 2019] [Tarunesh et al. 2021]. These methods often involve neural generative models such as variational autoencoders (VAEs). A VAE is an encoder-decoder architecture where inputs are mapped into and out of a latent space representation. The encoder and decoder are jointly learned to minimize the reconstruction error of the training data. By randomly selecting new points in the latent space, we can use a trained decoder to generate new sentences. We use the VACS (variational autoencoder for code-switching) architecture, which encodes sentences into a 2-layer hierarchical latent representation [Samanta et al. 2019]. The encoder encodes tokens W into a latent representation z_c, then uses both this representation and per-token language tags Y to create a second-level representation z_l encoding the language switching behavior. The decoder first generates the language tags Y from latent representation z_l then uses these along with a latent word representation z_c to generate the tokens of the sentence W.

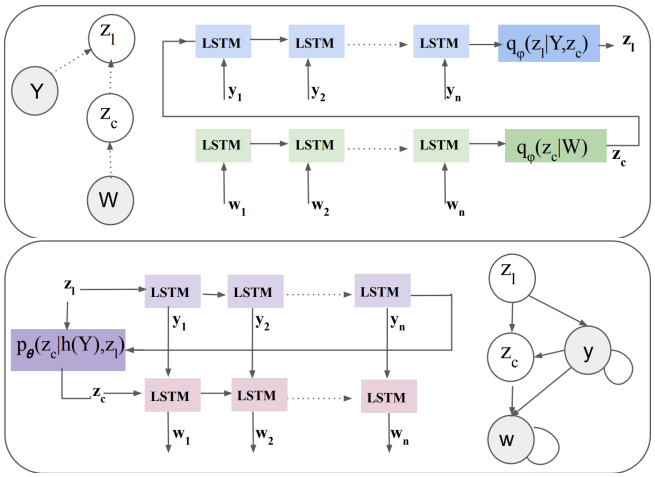

Fig. 2. The VACS architecture encoder and decoder

3 EVALUATING CODE-SWITCHED DATA

We classify evaluations of code-switched data as either *intrinsic* or *extrinsic* evaluations. Intrinsic evaluations look at statistics computed on a corpus of generated code-switched data. These can be compared to the same statistics computed on natural code-switched data. Extrinsic evaluations consider how usable the data is for some code-switched language task.

3.1 Intrinsic Evaluations

We consider 3 intrinsic, or distributional, evaluations. We further note that all 3 evaluations can be easily manipulated with rudimentary substitutive generation methods, making them unsuitable as reliable measurements of similarly between synthetic and natural code-switched data.

Code-mixing index (CMI) measures the amount of the less common language present in the code-switched text as a proxy for diversity. The CMI for real Hindi-English code-switched data is approximately 25, indicating a roughly 75% to 25% split of words in the context language (often Hindi) and the embedding language [Gambäck 2014]. This property can be set explicitly during substitutive generation, e.g. by using a selection module that selects 25% of tokens to translate, creating highly unrealistic code-switched data that scores well on this evaluation.

Self-BLEU score measures diversity by treating generated sentences as translation pairs and measuring translation similarity between different sentences in the corpus [Zhu et al. 2018]. This

property can be increased by selecting very diverse source sentences for substitutive generation. These sentences will then maintain their token dissimilarities after substitutive replacement of several tokens, creating a diverse set of generated sentences.

BERTScore is similar to self-BLEU score, but uses the contextual embeddings of sentences instead of comparing tokens for exact matches [Zhang et al. 2020]. Similar to self-BLEU, this can be increased simply by selecting more diverse input sentences with substitutive generation.

3.2 Extrinsic Evaluations

The motivation behind extrinsic, or downstream, evaluation is that our primary goal in generating code-switched data is not to create data that is similar in distribution to real code-switched data. Instead, it is to help train a language model to perform code-switched language tasks on real code-switched inputs. Therefore, even if some data does not share distributional properties with real code-switched data, as long as it is useful in training a code-switched language model, we wish to rate it highly on our evaluation. The natural way to perform such evaluations is to then use the generated data to fine-tune a language model, then use the language model to perform downstream tasks.

We construct an extrinsic evaluation by fine-tuning multilingual language models (mBERT [Devlin et al. 2018], mDistilBERT [Sanh et al. 2019], mRoBERTa [Conneau et al. 2019]) with a given generated code-switched corpus, then using this language model on the GLUECoS benchmark, which tests the language model on 6 downstream tasks: Language Identification (LID), Part-of-Speech Tagging (POS Tagging), Named Entity Recognition (NER), Sentiment Analysis (SA), Question Answering (QA), and Natural Language Inference (NLI) [Khanuja et al. 2020].

Token Tasks	Language Identification (LID)	Named Entity Recognition (NER)	Part of Speech Tagging (POS)
Sequence Tasks	Question Answering (QA)	Sentiment Analysis (SA)	Natural Language Inference (NLI)

Fig. 3. The tasks in the GLUECoS benchmark

3.3 Improving the GLUECoS Benchmark

Upon initial testing, we noticed that the GLUECoS benchmark is not an adequate differentiator of performance between code-switched language models.

The first four evaluations in the GLUECoS benchmarks are token classification tasks. These include NER, LID, and two POS evaluations, one with the Google Universal Dependency dataset and one from the ICON contest for Code-Mixed Indian Social Media Text. In general, these evaluations cannot differentiate between different models. All tested models perform extremely well these tasks, and furthermore, different models have extremely similar performance results on these tasks. It is likely that each evaluation has a set of inputs for which the task is hard, and the remaining data for each task is easily classifiable.

The QA and NLI evaluations have extremely little data. Particularly in case of the QA evaluation, this causes extremely high variance in results. Different splits of the training, validation, and testing data can produce test results both as low as 0% and over 70% accuracy. We addressed this by shuffling and re-separating the data into 10 different train-val-test splits, a process known as random sample validation. While this does not necessarily ensure the average result produced over the samples will be a much more accurate indicator of model performance, it allows us to compute the variance across splits in order to provide a numerical indicator of how reliable each result is and how significant differences between results are for different models.

Finally, the original sentiment evaluation is a sequence classification task with postitive, neutral, and negative classes. Because all models trained on this evaluation collapse to outputting a single class label, achieving very poor performance, we removed the neutral class from the data. The resulting evaluation produces extreme differences between different model architectures and potentially significant differences between different fine-tunings of the same pretrained model.

4 RESULTS

4.1 Experimental Setup

All experiments were run on a CMU LTI machine containing 2 Nvidia GTX 1070s. Experiments were run with Python 3.6.10 using an evaluation system based on the GLUECoS repository and modified according to Section 3.2.

For fine-tuning language models on code-switched text, we used Masked Language Model training with the following parameters:

- Learning rate: $5 * 10^{-5}$
- Batch masking probability: 0.15
- Maximum sequence length: 512
- Epochs: 2
- Batch size: 2
- Gradient accumulation steps: 10

We experimented with different learning rates and training epochs to attempt to have models learn more information from provided datasets, but we found these parameters from Tarunesh et al. to be the most effective [Tarunesh et al. 2021].

We used 3 pretrained models from HuggingFace.

- We refer to `bert-base-multilingual-cased` as mBERT.
- We refer to `distilbert-base-multilingual-cased` as mDistilBERT.
- We refer to `xlm-roberta-base` as mRoBERTa.

We finetuned these models with several different datasets. `-Sub` refers to models finetuned with a token substitution method. These models received 20,360 sentences of WikiQA data in which 70% of tokens were selected for replacement via a bilingual lexicon into Hindi. `-VACS` refers to models finetuned with 20,360 sentences

generated from VACS. -DOG refers to models finetuned with 48,465 sentences generated from the CMU DoG data, as discussed in Section 2.1.

4.2 Experimental Results

We first note that results for the token classification evaluations in the GLUECoS benchmarks do not vary much between different models. We found that all tested models were similar in performance on all 4 of these tasks, as seen in Table 1. The exception to this trend was the mRoBERTa model, which showed statistically significant improvement on the POS ICON task. We believe that there may be potential for fine-tunings of this model using code-switched data to achieve even greater performances, but our compute resources were insufficient to perform the fine-tuning process on this model.

We then note that for the question answering tasks, the standard deviation of results on all models is extremely high. Thus, even though there are large differences in performance (e.g. 4% between mBERT-VACS and mBERT-DOG as seen in Table 3), there is likely no statistically significant difference measured between the performance of these two models.

We tested two versions of the sentiment analysis task. The three-class sentiment analysis task (containing positive, neutral, and negative labels) has consistent results of approximately 45% accuracy. This corresponds to a mode collapse in which the model almost always outputs the neutral class label, which occurs with about 45% frequency in the data. This is likely due to two causes. First, the class imbalance in which the neutral class is by far the majority class makes it more likely for models to output this label. Second, the neutral class is commonly used in sentiment analysis tasks to label not only sentences with neutral sentiment but also sentences to which no sentiment label reasonably applies. It thus covers a much wider variety of sentences, and sentences with unfamiliar sentiment markers may be grouped by default under neutral sentiment in model predictions.

While removing the neutral class makes the sentiment analysis task less representative of the real-world problem of SA, we found that this produced a benchmark with stronger differentiating power. The result two-class version of the sentiment analysis task has significantly different results with different models. As seen in Table 3, the mRoBERTa model has much worse performance than the mBERT and mDistilBERT based models. The mDistilBERT models show small improvement after finetuning with code-switched data, though this is likely not statistically significant. We also note that the mBERT model has high stanard deviation, but that this is due to an outlier in the 10 samples of train-val-test splits rather than consistently spread out results. We conclude that the resulting 2-class sentiment analysis task is the most discriminative of our evaluations in determining what language models perform best with code-switched data.

5 DISCUSSION AND CONCLUSIONS

We conclude that the current practice of reporting benchmark results with only accuracy results can be extremely misleading and that state-of-the-art achievements reported with the same practices may not in fact be proper advancements in the field. We advocate for the inclusion of quantitative measures of significance for differences between results.

We also note that the primary focus of designing such evaluations is to be able to differentiate model performance with respect to a real-world task domain. Tasks that are too easy do not provide valuable insights into how different tested models can potentially improve task performance. Tasks that are too hard do not provide insights into which models are performing better than others. We thus note the need to design evaluations relative to the current state-of-the-art capability in a field and to continually update those evaluations to keep them relevant as models in the field improve.

6 FUTURE WORK

The primary avenue of future work for this project is to add more differentiating evaluations with additional data. While the sentiment analysis evaluation is now usable for differentiation, it is a single evaluation with only around 10,000 samples. Increasing the amount of data would ensure that models are not overfitting to this task. Adding more evaluations would ensure that models that are uniquely suited to a particular evaluation (e.g., due to the training data used) are not classified as disproportionately high performance despite poor performance on other tasks or in other domains.

Another avenue of future work is developing new methods of generating code-switched data that achieve higher performance on the evaluation. Having better generative methods could create a wider spread of values along the evaluation, giving a better picture of how well the evaluation can distinguish different thresholds of performance.

ACKNOWLEDGMENTS

Thanks to Alan W. Black for his mentorship throughout the project and detailed feedback and direction on research objectives. We thank Khyathi Chandu for her assistance in determining project direction and providing resources and related work to explore. We thank Akruti Kushwaha for providing the CMU DoG data. Thanks to Leila Wehbe for providing additional feedback on this project in bi-weekly meetings.

REFERENCES

Gayatri Bhat, Monojit Choudhury, and Kalika Bali. 2016. Grammatical Constraints on Intra-sentential Code-Switching: From Theories to Working Models. arXiv:1612.04538 [cs.CL]

Khyathi Raghavi Chandu and Alan W. Black. 2020. Style Variation as a Vantage Point for Code-Switching. In *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, Helen Meng, Bo Xu, and Thomas Fang Zheng (Eds.). ISCA, 4761–4765. https://doi.org/10.21437/Interspeech.2020-2574

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised Cross-lingual Representation Learning at Scale. *CoRR* abs/1911.02116 (2019). arXiv:1911.02116 http://arxiv.org/abs/1911.02116

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

Björn Gambäck. 2014. On Measuring the Complexity of Code-Mixing.

Google. 2022. Google Translate. https://translate.google.com/.

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, ACL 2020, Online, July 5-10, 2020*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.). Association for Computational Linguistics, 3575–3585. https://doi.org/10.18653/v1/2020.acl-main.329

Table 1. Token Evaluation Results

Model	LID Mean	LID SD	NER Mean	NER SD	POS UD Mean	POS UD SD	POS ICON Mean	POS ICON SD
mBERT	0.9092	0.0058	0.9652	0.0020	0.8099	0.0074	0.7014	0.0064
mBERT-Sub	0.9101	0.0060	0.9645	0.0026	0.8101	0.0063	0.7020	0.0085
mBERT-VACS	0.9105	0.0046	0.9650	0.0012	0.8113	0.0089	0.7015	0.0069
mBERT-DOG	0.9083	0.0058	0.9642	0.0017	0.8110	0.0079	0.7012	0.0076
mDistilBERT	0.9088	0.0050	0.9638	0.0028	0.8015	0.0089	0.6955	0.0063
mDistilBERT-DOG	0.9068	0.0036	0.9636	0.0019	0.8067	0.0116	0.6944	0.0085
mRoBERTa	0.9097	0.0196	0.9644	0.0067	0.8199	0.0080	0.7295	0.0070

Table 2. Question Answering Results

Model	QA Mean	QA SD
mBERT	.6857	.0878
mBERT-Sub	.6798	.1126
mBERT-VACS	.6577	.1012
mBERT-DOG	.6953	.1344

Table 3. Sentiment Analysis Results

Model	3-Class SA Mean	3-Class SA SD	2-Class SA Mean	2-Class SA SD
mBERT	0.4550	0.0142	0.7224	0.0586
mBERT-Sub	0.4556	0.0151	-	-
mBERT-VACS	0.4550	0.0142	-	-
mBERT-DOG	0.4550	0.0142	-	-
mDistilBERT	-	-	0.7403	0.0184
mDistilBERT-DOG	-	-	0.7494	0.0149
mRoBERTa	-	-	0.5911	0.0464

Bidisha Samanta, Sharmila Reddy, Hussain Jagirdar, Niloy Ganguly, and Soumen Chakrabarti. 2019. A Deep Generative Model for Code Switched Text. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 5175–5181. https://doi.org/10.24963/ijcai.2019/719

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv* abs/1910.01108 (2019).

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)*. Association for Computational Linguistics, Online, 3154–3169. https://doi.org/10.18653/v1/2021.acl-long.245

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics, Brussels, Belgium, 353–355. https://doi.org/10.18653/v1/W18-5446

Jitao Xu and François Yvon. 2021. Can You Traducir This? Machine Translation for Code-Switched Input. *CoRR* abs/2105.04846 (2021). arXiv:2105.04846 https://arxiv.org/abs/2105.04846

Zhen Yang, Bojie Hu, Ambyera Han, Shen Huang, and Qi Ju. 2020. CSP:Code-Switching Pre-training for Neural Machine Translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 2624–2636. https://doi.org/10.18653/v1/2020.emnlp-main.208

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. *ArXiv* abs/1904.09675 (2020).

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A Benchmarking Platform for Text Generation Models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (Ann Arbor, MI, USA) *(SIGIR '18)*. Association for Computing Machinery, New York, NY, USA, 1097–1100. https://doi.org/10.1145/3209978.3210080