

LEARN TO CODE-SWITCH: DATA AUGMENTATION USING COPY MECHANISM ON LANGUAGE MODELING

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, Pascale Fung

Center for Artificial Intelligence Research (CAiRE)
Department of Electronic and Computer Engineering
Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
{giwinata, amadotto, cwuak}@connect.ust.hk, pascale@ece.ust.hk

ABSTRACT

Building large-scale datasets for training code-switching language models is challenging and very expensive. To alleviate this problem using parallel corpus has been a major workaround. However, existing solutions use linguistic constraints which may not capture the real data distribution. In this work, we propose a novel method for learning how to generate code-switching sentences from parallel corpora. Our model uses a Seq2Seq model in combination with pointer networks to align and choose words from the monolingual sentences and form a grammatical code-switching sentence. In our experiment, we show that by training a language model using the augmented sentences we improve the perplexity score by 10% compared to the LSTM baseline.

Index Terms— code-switch, bilingual, copy attention, language modeling, data augmentation

1. INTRODUCTION

Language mixing has been a common phenomenon in multilingual communities. It is motivated in response to social factors as a way of communication in a multicultural society. From a sociolinguistic perspective, individuals do code-switching in order to construct an optimal interaction by accomplishing the conceptual, relational-interpersonal, and discourse-presentational meaning of conversation [1]. In its practice, the variation of code-switching will vary due to the traditions, beliefs, and normative values in the respective communities. A number of studies [2, 3, 4, 5] found that code-switching is not produced indiscriminately, but follows syntactic constraints. Many linguists formulated various constraints to define a general rule for code-switching [2, 4, 5]. However, the constraints are not enough to make a good generalization of real code-switching constraints, and they have not been tested in large-scale corpora for many language pairs.

One of the biggest problem in code-switching is collecting large scale corpora. Speech data have to be collected from a spontaneous speech by bilingual speakers and the code-switching has to be triggered naturally during the conversation. In order to solve the data scarcity issue, code-switching data generation is useful to increase the volume and variance. A linguistics constraint-driven generation approach such as equivalent constraint [6, 7] is not restrictive to languages with distinctive grammar structure.

In this paper, we propose a novel language-agnostic method to learn how to generate code-switching sentences by using a pointer-generator network [8]. The model is trained from concatenated sequences of parallel sentences to generate code-switching sentences, constrained by code-switching texts. The pointer network copies words from both languages and pastes them into the output, generating code switching sentences in matrix language to embedded language and vice versa. The attention mechanism helps the decoder to generate meaningful and grammatical sentences without needing any sequence alignment. This idea is also in line with code-mixing by borrowing words from the embedded language [9] and intuitively, the copying mechanism can be seen as an end-to-end approach to translate, align, and reorder the given words into a grammatical code-switching sentence. This approach is the unification of all components in the work of [6] into a single computational model. A code-switching language model learned in this way is able to capture the patterns and constraints of the switches and mitigate the out-of-vocabulary (OOV) issue during sequence generation. By adding the generated sentences and incorporating syntactic information to the training data, we achieve better performance by 10% compared to an LSTM baseline [10] and 5% to the equivalent constraint.

2. RELATED WORK

The synthetic code-switching generation approach was introduced by adapting equivalence constraint on monolingual sentence pairs during the decoding step on an automatic

This work is partially funded by ITS/319/16FP of the Innovation Technology Commission, HKUST 16214415 & 16248016 of Hong Kong Research Grants Council, and RDC 1718050-0 of EMOS.AI.

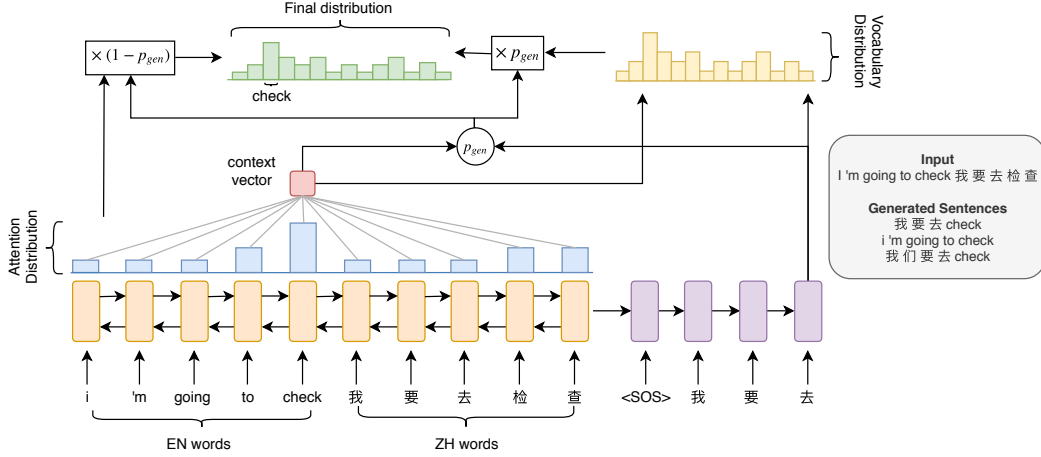


Fig. 1. Pointer Generator Networks [8]. The figure shows the example of input and 3-best generated sentences.

speech recognition (ASR) model [6]. [11] explored Functional Head Constraint, which was found to be more restrictive than the Equivalence Constraint, but complex to be implemented, by using a lattice parser with a weighted finite-state transducer. [12] extended the RNN by adding POS information to the input layer and factorized output layer with a language identifier. Then, Factorized RNN networks were combined with an n-gram backoff model using linear interpolation [13]. [14] added syntactic and semantic features to the Factorized RNN networks. [15] adapted an effective curriculum learning by training a network with monolingual corpora of both languages, and subsequently train on code-switched data. A further investigation of Equivalence Constraint and Curriculum Learning showed an improvement in language modeling [7]. A multi-task learning approach was introduced to train the syntax representation of languages by constraining the language generator [10].

A copy mechanism was proposed to copy words directly from the input to the output using an attention mechanism [16]. This mechanism has proven to be effective in several NLP tasks including text summarization [8], and dialog systems [17]. The common characteristic of these tasks is parts of the output are exactly the same as the input source. For example, in dialog systems the responses most of the time have appeared in the previous dialog steps.

3. METHODOLOGY

We use a sequence to sequence (Seq2Seq) model in combination with pointer and copy networks [8] to align and choose words from the monolingual sentences and generate a code-switching sentence. The models' input is the concatenation of the two monolingual sentences, denoted as $[w_1^{\ell_1}, \dots, w_n^{\ell_1}, w_1^{\ell_2}, \dots, w_m^{\ell_2}]$, and the output is a code-switched sentence, denoted as $y_1^{cs}, \dots, y_k^{cs}$. The main as-

sumption is that almost all, the token present in the code-switching sentence are also present in the source monolingual sentences. Our model leverages this property by copying input tokens, instead of generating vocabulary words. This approach has two major advantages: (1) the learning complexity decreases since it relies on copying instead of generating; (2) improvement in generalization, the copy mechanism could produce words from the input that are not present in the vocabulary.

3.1. Pointer-generator Network

Instead of generating words from a large vocabulary space using a Seq2Seq model with attention [18], pointer-generator network [8] is proposed to copy words from the input to the output using an attention mechanism and generate the output sequence using decoders. The network is depicted in Figure 1. For each decoder step, a generation probability $p_{gen} \in [0,1]$ is calculated, which weights the probability of generating words from the vocabulary, and copying words from the source text. p_{gen} is a soft gating probability to decide whether generating the next token from the decoder or copying the word from the input instead. The attention distribution a_t is a standard attention with general scoring [18]. It considers all encoder hidden states to derive the context vector. The vocabulary distribution $P_{vocab}(w)$ is calculated by concatenating the decoder state s_t and the context vector h_t^* .

$$p_{gen} = \sigma(w_h^T h_t^* + w_s^T s_t + w_x^T x_t + b_{ptr}) \quad (1)$$

where w_h^* , w_s , w_x are trainable parameters and b_{ptr} is the scalar bias. The vocabulary distribution $P_{vocab}(w)$ and the attention distribution a_t are weighted and summed to obtain the final distribution $P(w)$. The final distribution is calculated as follows:

$$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} a_i^t \quad (2)$$

Table 1. Data Statistics of SEAME Phase II and Generated Sequences using Pointer-generator Network [10].

	SEAME Phase II			Generated Seqs	
	Train	Dev	Test	1-best	3-best
# Speakers	138	8	8	-	-
# Utterances	78,815	4,764	3,933	78,815	236,445
# Tokens	1.2M	65K	60K	-	-
# Tokens Preprocessed	978K	53K	48K	945K	2.8M
Avg. segment	4.21	3.59	3.99	4.77	4.31
Avg. switches	2.94	3.12	3.07	2.51	2.79

Table 2. Code-Switching Sentence Generation Results. Higher BLEU and lower perplexity (PPL) is better.

	Dev		Test
	BLEU	PPL	BLEU
seq2seq with attention	53.71	5.89	56.10
pointer-generator	55.19	4.61	59.68

We use a beam search to select N -best code-switching sentences and concatenate the generated sentence with the training set to form a larger dataset. The result of the generated code-switching sentences is showed in Table 1. As our baseline, we compare our proposed method with three other models: (1) We use Seq2Seq with attention; (2) We generate sequences that satisfy Equivalence Constraint [6]. The constraint doesn’t allow any switch within a crossing of two word alignments. We use FastAlign [19] as the word aligner¹; (3) We also form sentences using the alignments without any constraint. The number of the generated sentences are equivalent to 3-best data from the pointer-generator model. To increase the generation variance, we randomly permute each alignment to form a new sequence.

3.2. Language Modeling

The quality of the generated code-switching sentences is evaluated using a language modeling task. Indeed, if the perplexity in this task drops consistently we can assume that the generated sentences are well-formed. Hence, we use an LSTM language model with weight tying [20] that can capture an unbounded number of context words to approximate the probability of the next word. Syntactic information such as Part-of-speech (POS) $[p_1, \dots, p_T]$ is added to further improve the performance. The POS tags are generated phrase-wise using pretrained English and Chinese Stanford POS Tagger [21] by adding a word at a time in a unidirectional way to avoid any intervention from future information. The word and syntax unit are represented as a vector x^w and x^p respectively. Next, we concatenate both vectors and use it as an input $[x^w|x^p]$ to an LSTM layer similar to [10].

¹The code can be found at https://github.com/clab/fast_align

Table 3. Language Modeling Results (in perplexity).

	w/o syntax		with syntax	
	Dev	Test	Dev	Test
RNNLM [10]	178.35	171.27	-	-
LSTM [10]	150.65	153.06	147.44	148.38
with augmentation				
random switch	166.70	158.87	153.46	151.08
equivalent-constraint	149.72	147.03	147.48	145.05
pointer-generator 1-best	145.05	144.26	143.39	140.96
pointer-generator 3-best	144.69	143.84	142.84	138.91

4. EXPERIMENT

4.1. Corpus

In our experiment, we use a conversational Mandarin-English code-switching speech corpus called SEAME Phase II (South East Asia Mandarin-English). The data are collected from spontaneously spoken interviews and conversations in Singapore and Malaysia by bilinguals [22]. As the data preprocessing, words are tokenized using Stanford NLP toolkit [23] and all hesitations and punctuations were removed except apostrophe. The split of the dataset is identical to [10] and it is showed in Table 1.

4.2. Training Setup

In this section, we present the experimental settings for pointer-generator network and language model. Our experiment, our pointer-generator model has 500-dimensional hidden states and word embeddings. We use 50k words as our vocabulary for source and target. We evaluate our pointer-generator performance using BLEU² score. We take the best model as our generator and during the decoding stage, we generate 1-best and 3-best using beam search with a beam size of 5. For the input, we build a parallel monolingual corpus by translating the mixed language sequence using Google NMT³ to English (w^{ℓ_1}) and Mandarin (w^{ℓ_2}) sequences. Then, we concatenate the translated English and Mandarin sequences and assign code-switching sequences as the labels (y^{cs}).

The baseline language model is trained using RNNLM [24]⁴. Then, we train our 2-layer LSTM models with a hidden size of 500 and unrolled for 35 steps. The embedding size is equal to the LSTM hidden size for weight tying. We optimize our model using SGD with initial learning rates of $\{10, 20\}$. If there is no improvement during the evaluation, we reduce the learning rate by a factor of 0.75. In each time step, we apply dropout to both embedding layer and recurrent network. The gradient is clipped to a maximum of 0.25. Perplexity measure is used in the evaluation.

²BLEU is computed using multi_bleu.perl from MOSES package

³Google NMT Translate API

⁴RNNLM toolkit from <http://www.fit.vutbr.cz/~imikolov/rnnlm/>

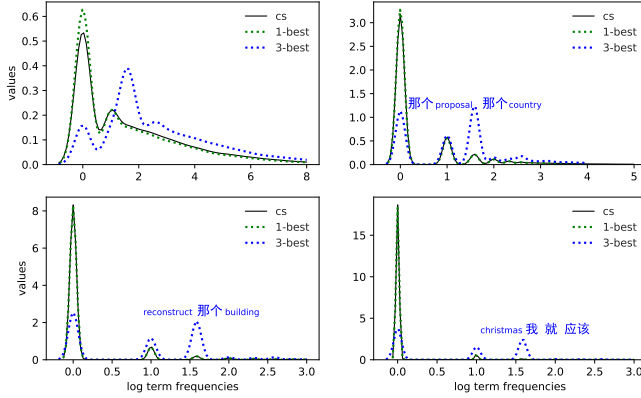


Fig. 2. Univariate data distribution for unigram (top-left), bigram (top-right), trigram (bottom-left), and fourgram (bottom-right). The showed n-grams are sampled from 3-best data pointer-generator model.

5. RESULTS

The pointer-generator significantly outperforms the Seq2Seq with attention model by 3.58 BLEU points on the test set as shown in Table 2. Our language modeling result is given in Table 3. Based on the empirical result, adding generated samples consistently improve the performance of all models with a moderate margin around 10% in perplexity. After all, our proposed method still slightly outperforms the heuristic from linguistic constraint. In addition, we get a crucial gain on performance by adding syntax representation of the sequences.

Change in data distribution: To further analyze the generated result, we observed the distribution of real code-switching data and the generated code-switching data. From Figure 2, we can see that 1-best and real code-switching data have almost identical distributions. The distributions are left-skewed where the overall mean is less than the median. Interestingly, the distribution of the 3-best data is less skewed and generates a new set of n-grams such as “那个 (that) proposal” which was learned from other code-switching sequences. As a result, generating more samples effects the performance positively.

Importance of Linguistic Constraint: The result in Table 3 emphasizes that linguistic constraints have some significance in replicating the real code-switching patterns, specifically the equivalence constraint. There is a slight reduction in perplexity around 6 points on the test set. In addition, when we ignore the constraint, we lose performance because it still allows switches in the inversion grammar cases.

Does the pointer-generator learn how to switch? We found that our pointer-generator model generates sentences that have not been seen before. The example in Figure 1 shows that our model is able to construct a new well-formed sentence such as “我们要去 (We want to) check”. It is also shown that the pointer-generator model has the capability

to learn the characteristics of the linguistic constraints from data without any word alignment between the matrix and embedded languages. On the other hand, training using 3-best data obtains better performance compared to 1-best data. We found a positive correlation from Table 1, where 3-best data is more similar to the test set in terms of segment length and number of switches compared to 1-best data. Adding more samples N may improve the performance, but it will be saturated at a certain point. One way to solve this is by using more parallel samples.

6. CONCLUSION

We introduce a new learning method for code-switching sentence generation using a parallel monolingual corpus that is applicable to any language pair. Our experimental result shows that adding generated sentences to the training data, effectively improves our model performance. Combining the generated samples with code-switching dataset reduces perplexity. We get further performance gain after using syntactic information of the input. In future work, we plan to explore reinforcement learning for sequence generation and employ more parallel corpora.

7. REFERENCES

- [1] Rakesh M Bhatt and Agnes Bolonyai, “Code-switching and the optimal grammar of bilingual language use,” *Bilingualism: Language and Cognition*, vol. 14, no. 4, pp. 522–546, 2011.
- [2] Shana Poplack, *Syntactic structure and social function of code-switching*, vol. 2, Centro de Estudios Puertorriqueños,[City University of New York], 1978.
- [3] Carol W Pfaff, “Constraints on language mixing: intrasentential code-switching and borrowing in spanish/english,” *Language*, pp. 291–318, 1979.
- [4] Shana Poplack, “Sometimes i’ll start a sentence in spanish y termino en espanol: toward a typology of code-switching1,” *Linguistics*, vol. 18, no. 7-8, pp. 581–618, 1980.
- [5] Hedi M Belazi, Edward J Rubin, and Almeida Jacqueline Toribio, “Code switching and x-bar theory: The functional head constraint,” *Linguistic inquiry*, pp. 221–237, 1994.
- [6] Ying Li and Pascale Fung, “Code-switch language model with inversion constraints for mixed language speech recognition,” *Proceedings of COLING 2012*, pp. 1671–1680, 2012.
- [7] Adithya Pratapa, Gayatri Bhat, Monojit Choudhury, Sunayana Sitaram, Sandipan Dandapat, and Kalika Bali,

- “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)*, 2018, vol. 1, pp. 1543–1553.
- [8] Abigail See, Peter J. Liu, and Christopher D. Manning, “Get to the point: Summarization with pointer-generator networks,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2017, pp. 1073–1083, Association for Computational Linguistics.
- [9] John M Lipski, “Code-switching or borrowing? no sé so no puedo decir, you know,” in *Selected proceedings of the second workshop on Spanish sociolinguistics*. Cascadilla Proceedings Project Somerville, MA, 2005, pp. 1–15.
- [10] Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung, “Code-switching language modeling using syntax-aware multi-task learning,” in *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*. 2018, pp. 62–67, Association for Computational Linguistics.
- [11] LI Ying and Pascale Fung, “Language modeling with functional head constraint for code switching speech recognition,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 907–916.
- [12] Heike Adel, Ngoc Thang Vu, Franziska Kraus, Tim Schlippe, Haizhou Li, and Tanja Schultz, “Recurrent neural network language modeling for code switching conversational speech,” in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8411–8415.
- [13] Heike Adel, Ngoc Thang Vu, and Tanja Schultz, “Combination of recurrent neural networks and factored language models for code-switching language modeling,” in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2013, vol. 2, pp. 206–211.
- [14] Heike Adel, Ngoc Thang Vu, Katrin Kirchhoff, Dominic Telaar, and Tanja Schultz, “Syntactic and semantic features for code-switching factored language models,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 431–440, 2015.
- [15] Ashutosh Baheti, Sunayana Sitaram, Monojit Choudhury, and Kalika Bali, “Curriculum design for code-switching: Experiments with language identification and language modeling with deep neural networks,” *Proceedings of ICON*, pp. 65–74, 2017.
- [16] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly, “Pointer networks,” in *Advances in Neural Information Processing Systems*, 2015, pp. 2692–2700.
- [17] Andrea Madotto, Chien-Sheng Wu, and Pascale Fung, “Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 1468–1478, Association for Computational Linguistics.
- [18] Thang Luong, Hieu Pham, and Christopher D Manning, “Effective approaches to attention-based neural machine translation,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1412–1421.
- [19] Chris Dyer, Victor Chahuneau, and Noah A. Smith, “A simple, fast, and effective reparameterization of ibm model 2,” in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2013, pp. 644–648, Association for Computational Linguistics.
- [20] Ofir Press and Lior Wolf, “Using the output embedding to improve language models,” in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. 2017, pp. 157–163, Association for Computational Linguistics.
- [21] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer, “Feature-rich part-of-speech tagging with a cyclic dependency network,” in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, 2003, pp. 173–180.
- [22] Universiti Sains Malaysia Nanyang Technological University, “Mandarin-english code-switching in south-east asia ldc2015s04. web download. philadelphia: Linguistic data consortium,” 2015.
- [23] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky, “The Stanford CoreNLP natural language processing toolkit,” in *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, pp. 55–60.
- [24] Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Cernocky, “Rnnlm-recurrent neural network language modeling toolkit,” in *Proc. of the 2011 ASRU Workshop*, 2011, pp. 196–201.