Code-switching Sentence Generation by Generative Adversarial Networks and its Application to Data Augmentation

Ching-Ting Chang, Shun-Po Chuang, Hung-Yi Lee

Graduate Institute of Communication Engineering, National Taiwan University

`r05942066@ntu.edu.tw, f04942141@ntu.edu.tw, hungyilee@ntu.edu.tw`

Abstract

Code-switching is about dealing with alternative languages in speech or text. It is partially speaker-dependent and domain-related, so completely explaining the phenomenon by linguistic rules is challenging. Compared to most monolingual tasks, insufficient data is an issue for code-switching. To mitigate the issue without expensive human annotation, we proposed an unsupervised method for code-switching data augmentation. By utilizing a generative adversarial network, we can generate intra-sentential code-switching sentences from monolingual sentences. We applied the proposed method on two corpora, and the result shows that the generated code-switching sentences improve the performance of code-switching language models.

Index Terms: code-switching, generative adversarial networks, data augmentation, language model

1. Introduction

Code-switching (CS) is the practice that two or more languages are used within a document or a sentence. It is widely observed in multicultural areas, or countries where official language is different from native language. For example, Taiwanese tend to mix English and Taiwanese Hokkien in their text and speech besides their main language, Mandarin. Solving CS is crucial to building a general ASR system that can process both monolingual and CS speech [1, 2, 3]. In this paper, we focus on improving the language models for ASR of intra-sentential CS speech. Specifically, we only deal with words and phrases that are code-switched within a sentence.

Computational processing of CS is fundamentally challenging due to lack of data. Applying linguistic knowledge is a solution to this [4, 5]. Equivalence Constraint and Functional Head Constraint are used to build a better CS language model [6, 7, 8], and CS models with syntactic and semantic features are built to exploit more information [9, 10]. Because of a large amount of monolingual data, monolingual language models for host and guest languages are learned separately, and then combined with a probabilistic model for switching between the two [11].

Because CS is mostly used in spoken language, the most practical way of generating data is to label CS speech. However, manual transcription requires plenty of skilled labor and hours of tedious work. An alternative way is to generate CS data from existing monolingual text. Unfortunately, there are no flawless rules for predicting code-switching points within a sentence, since each person tends to code-switch in a different manner. These years, people try to synthesize more code-switching text by the models learned from data [12, 13, 14].

Generative models have been used to generate CS sentences [13], but previous work uses generative model to generate the sentences from scratch. Here the generator learns to modify monolingual sentences into CS sentences. In this way,

the generator can leverage the information from monolingual sentences.

We propose a novel CS text generation method, by using generative adversarial networks (GAN) [15] with reinforcement learning (RL) [16], to generate CS data from monolingual sentences automatically. With CS data augmented by our method, it is possible to solve the problem of sparse training data. Our proposed method has the following benefits:

- We don't use any labeled data to train the generator.
- The model learns CS rules for data generation implicitly with the help of discriminator instead of defining hand-crafted rules.
- We conduct the experiments on two Mandarin-English code-switching corpora, LectureSS and SEAME, which have very different characteristics to show that the proposed approach generalizes well in different cases.

The experimental results show that GAN can generate reasonable code-switching sentences, and the generated code-switching sentences can be used to improve language modeling.

2. Methodology

The main issue of training code-switching model is lack of adequate code-switching training sentences because code-switching mostly occurs in speech or personal messages instead of in written resources. We think that generating code-switching sentences from monolingual data may solve the above issue since we can obtain monolingual text much easier than code-switching text. In the following examples and discussion, Mandarin is the host language, and English is the guest language. Actually, the proposed approach is language independent, so it is possible to apply on other host-guest language pairs.
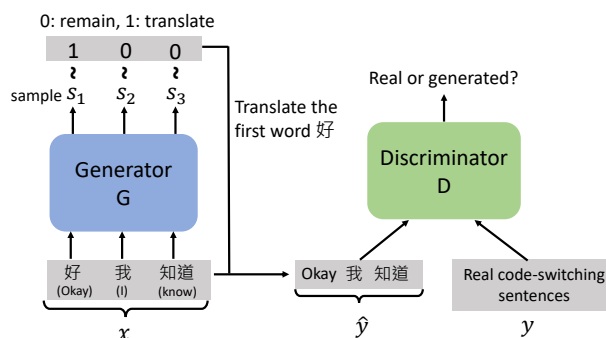

Figure 1: *Proposed framework. The generator learns to generate code-switching sentences from monolingual sentences.*

To generate intra-sentential code-switching sentences, we can randomly select some of the words in the Mandarin sentence and translate them into English. However, this ap-

Table 1: *Details for the corpora: LecureSS and SEAME.*

	LectureSS			SEAME		
	train	dev	test			
# speakers	-	-	-	139	8	8
# total utterances	12657	2634	1500	94055	6115	5908
# Mandarin utterances	4643	964	810	20365	1653	2211
# code-switching utterances	8014	1670	690	50421	3564	3066
# total words	160862	28063	12398	964927	65696	57620
# Mandarin words	138409	23308	10871	543399	42525	41874
# English words	22453	4755	1527	417452	22641	15370

Table 2: *Comparison between LectureSS and SEAME.*

	LectureSS	SEAME
# speakers	1	156
Nationality	Taiwan	Singapore & Malaysia
Domain	signal lecture	daily, school
ZH:EN words	1:0.16	1:0.7
Cs-rate	20%	25%

proach will generate many unreasonable code-switching sentences. For instance, given the monolingual sentence "我要介紹... " (I will introduce ...), replacing the word "我" with "I" does not generate a reasonable sentence because few speakers would code-switch in this way. Nevertheless, there are no perfect rules to predict which word or phrase in a sentence should be code-switched or not. Inspired by GAN, we propose to learn a conditional generator for code-switching sentences, so it can transform a monolingual sentence into a code-switching sentence.

Discriminator. The discriminator D takes a sentence as input, and outputs a scalar between 0 and 1. The output scalar indicates that the input sentence is generated by the generator G or from the given code-switching training sentence. For a well-trained perfect discriminator, the output is zero when the sentence is generated by generator G, and the output is one when the sentence is sampled from the training data set.

Generator. The generator G takes a monolingual (Chinese) word sequence $x = \{x_1, x_2, ..., x_N\}$ as input, where N is the length of x^1. The output of G is a sequence of values $s = \{s_1, s_2, ..., s_N\}$. s_n is a scalar between 0 and 1 corresponding to input word x_n. Each scalar s_n represents whether it is proper to replace x_n with its translated counterpart. s_n is considered as a probability, and a binary value is sampled from it. If the sampled value is 1, the word in Mandarin will be translated into English. By contrast, if 0 is sampled, the input word will remain the same. A code-switching sentence $\hat{y}$ is thus generated from G. The generator proposed here only learns which word in Mandarin can be replaced with English. It is possible to have a generator which directly generates code-switching word sequence. However, learning which word can be replaced is easier than learning to generate the words in another language directly.

Training of Discriminator. D is learned by minimizing $\mathcal{L}_D$ below,

$$\mathcal{L}_D = -(\mathbb{E}_{y \sim \mathcal{D}_{cs}}[logD(y)] + \mathbb{E}_{x \sim \mathcal{D}_{zh}, \hat{y} \sim G(x)}[log(1-D(\hat{y}))]).\tag{1}$$

In the first term of (1), the code-switching sentence y is sampled from the training data $\mathcal{D}_{cs}$, and the discriminator D learns to as-

sign a larger score $D(y)$ to y. In the second term, a monolingual sentence x is sampled from a data set $\mathcal{D}_{zh}$, and G transforms x into a code-switching sentence $\hat{y}$. D learns to assign a smaller score $D(\hat{y})$ to $\hat{y}$.

Training of Generator. The parameters in G are learned from the following loss function $\mathcal{L}_G$.

$$\mathcal{L}_G = -\mathbb{E}_{x \sim \mathcal{D}_{zh}, \hat{y} \sim G(x)}[logD(\hat{y})].\tag{2}$$

With (2), G learns to generate $\hat{y}$ that can obtain large $D(\hat{y})$. Due to the output of discriminator is discrete, the model is updated by the REINFORCE algorithm [17, 18, 19].

The discriminator D and generator G are trained iteratively as typical GAN.

3. Experimental setup

3.1. Corpora

In this work, we utilized two data sets for the experiments: LectureSS and SEAME corpus [20]. The detailed statistics of these corpora are listed in Table 1. Additionally, we draw a comparison between them in Table 2. CS-rate in Table 2 is defined as below,

$$\text{CS-rate} = \frac{\text{\# English words in CS utterances}}{\text{\# total words in CS utterances}}.\tag{3}$$

LectureSS is a lecture speech corpus recorded by one Taiwanese instructor at National Taiwan University in 2006. The content of the recording is "Signal and System" (SS) course. It is spontaneous speech with highly imbalanced Mandarin-English code-switching characteristics. Mandarin is the host language and English is the guest language. Most English words in this corpus are domain-specific terminologies.

South East Asia Mandarin-English (SEAME) corpus is a conversational speech corpus recorded by Singapore and Malaysia speakers with almost balanced gender in Nanyang Technological University and Universities Sains Malaysia. There are two speaking types in the speech: conversational and interview conditions, and the content are related to daily life, school, and so on. It is also Mandarin-English code-switching while the amount of Chinese (ZH) words and English (EN) words is about equal. Not only proper nouns but also conjunctions may be used in English in this corpus. Some sentences in SEAME are completely in English, while it does not happen in NTU lecture. Nevertheless, the cs-rate of LectureSS is close to the cs-rate of SEAME.

Before using these datasets, we cleaned them first. LectureSS comprises of "Zhuyin fuhao", mathematical symbols and English alphabet which cannot be translated into English words or Chinese words. In addition, SEAME contains non-speech labels, unknown words labels, incomplete words and foreign words. We removed these words directly if the semantics of the sentences would not be influenced too much; otherwise, we ignored the utterances in the experiments.

---

[1] As typical conditional GAN, the generator G also takes a noise z sampled from Gaussian as input. We ignore z in the following formulation for simplicity.

Table 3: *Code-switching point (CSP) prediction on manually labeled sentences.*

	Precision	Recall	F-measure	BLEU-1	WER(%)	EN WER	ZH WER
	LectureSS						
ZH	0	0	0	0.76	20.56	100	0
EN	0.21	1	0.35	0.20	102.1	0	128.5
random	0.17	0.16	0.16	0.62	39.20	88.14	26.54
noun	**0.55**	0.44	0.22	0.75	**17.06**	54.02	**10.21**
proposed	0.52	0.42	0.46	0.78	22.82	54.24	14.69
proposed+pos	0.52	**0.55**	**0.53**	**0.80**	21.08	**39.83**	16.23
	SEAME						
ZH	0	0	0	0.65	33.51	100	0
EN	0.3	1	0.46	0.01	80.35	0	117.3
random	0.26	0.23	0.24	0.47	47.02	76.50	33.07
noun	**0.61**	0.19	0.29	0.49	45.44	93.99	22.48
proposed	0.55	0.35	0.43	**0.58**	**30.00**	61.20	**15.25**
proposed+pos	0.51	**0.47**	**0.49**	0.52	33.33	**48.63**	26.10

3.2. Model Setup

The inputs of both the discriminator and the generator are word sequences. There are two ways to represent a word. In the first approach, each word is first represented by one-hot encoding, and transforms into an embedding by an embedding layer. We set 8200 and 12000 vocabulary size for LectureSS and SEAME individually, and 150 as the dimension for word embedding. In the second approach, we also consider the part-of-speech (POS) tag for each word. We used Jieba[2], an Open Source Chinese segmentation application in Python language, as our POS tagger. Only Chinese words are tagged and English words are tagged as "eng." Each POS tag corresponds to a 64-dim one-hot encoding, and it is transformed into 20-dim by an embedding layer. The embedding of words and POS tags are concatenated. The embedding layer is jointly trained with the whole model, and Chinese and English word embedding are trained together.

The generator G is made up of embedding layer, one bidirectional long short-term memory (BLSTM) [21] layer, one fully connected (FC) layer. It outputs one value with sigmoid for each time step to determine whether this word will be translated into English. Gaussian noise is 10-dim vector concatenated with the output of BLSTM. The parameter of G is updated by policy gradient with the output of D as reward. Translator is merely a mapping table which contains a list of Chinese vocabulary with each comparing English word translated by Google translator.

The discriminator D shares the same embedding layer and BLSTM with G. However, it is updated only when G is training and fixed when D is training. The output of BLSTM is passed into a FC layer with dropout rate 0.3. It ends in a one-dimension vector with sigmoid.

The whole optimization process is based on Adam optimizer [22] and we train 100 epochs for all experiments. The input data of G is all Chinese training sentences, and D is trained by all code-switching sentences in the training set and fake code-switching sentences generated by G in respective corpora.

4. Results

We evaluate our proposed method in three aspects: code-switching point (CSP) prediction, quality of generated text, and performance of language modeling with augmented text.

4.1. Code-switching Point Prediction

We selected 50 code-switching sentences y in testing set as ground truth, and manually translated them into fully Chinese sentences x. The generator then generates code-switching sentences $\hat{y}$ conditioned on x. We consider the positions of English words in y as CSPs that we want to detect, and use precision, recall and F-measure to evaluate the accuracy of detected CSPs in $\hat{y}$. Additionally, we also apply BLEU score [23] and word error rate (WER) in (4) to evaluate $\hat{y}$,

$$\text{Word Error Rate} = \frac{\sum_i \text{Edit Distance}(y_i, \hat{y}_i)}{\# \text{ total words}}, \quad (4)$$

where i indicates the ith selected code-switching sentences.

The proposed approach is compared with four baselines: (1) *ZH*: fully Chinese sentences, that is, $\hat{y} = x$. (2) *EN*: fully English sentences. (3) *random*: words are randomly translated into English. The translation probability for each word is the same as the cs-rate of the corpus considered. (4) *noun*: translate all words that are tagged as nouns (common nouns and proper nouns) [24] by POS tagger into English.

According to Table 3, we observe that *ZH* can have good performance in BLEU score and total WER. This is attributed to the fact that Chinese words occur more frequently than English words in code-switching sentences in both corpora. *Random* gets poor performance because people don't code-switch arbitrarily. *Noun* has high precision owing to high exactness, but it does not predict CSPs other than noun. Our method obtains better recall, F-measure, BLEU-1 and English WER than *noun* on all the corpora because it detects not only nouns but other CSPs like conjunctions, discourse particles, filled pauses, and so on.

4.2. Generated Text Quality

Next, we demonstrate the quality of our generated text by validating them with language model trained on training text. We calculate the perplexity (PPL) of our generated text.

Two types of language models were used to evaluate our results: n-gram model [25] and neural language model [26]. N-gram language model is a word-level tri-gram with Kneser-Ney (KN) smoothing [27] trained by SRILM [28]. Recurrent neural networks based language model (RNNLM) is a two-layer character-level LSTM [29] language model. Because the two corpora have different scales of training data, we used 32-dimensional LSTM for LectureSS and 64-dimensional LSTM for SEAME. We used Adam optimizer with initial learning rate

---

[2] Jieba toolkit from: https://github.com/fxsjy/jieba

Table 4: *Code-switching examples from different methods.*

Ground Truth	Causality 這個也是讀過的就是指我 output at-any-time 只 depend-on input
Input	因果性 這個也是讀過的就是指我輸出在任意時間只取決於輸入
	(Causality, this is also what you have read, that means what I output at any time only depends on input)
Random	因果性 this 也是讀過的就是指我 output 在任意時間只取決於輸入
Noun	Causality 這個也是讀過的就是指我輸出在任意時間只取決於 input
Proposed	Causality this also 你所 read 過的就是指我 output 在任意時間只取決於輸入
Proposed+pos	Causality 這個也是讀過的就是指我 output at-any-time 只 depend-on 輸入

Table 5: *Quality of generated code-switching text from testing text evaluated by PPL on n-gram LM and neural-based LM (RNNLM).*

	random	noun	proposed	proposed+pos
LectureSS				
n-gram	1022.95	337.713	**330.028**	400.275
RNNLM	82.292	75.515	82.802	**74.287**
SEAME				
n-gram	177.039	154.28	159.103	**145.3**
RNNLM	79.338	84.081	78.335	**69.345**

0.5 to optimize the network, and 0.7 dropout is also applied to avoid over-fitting.

We used *random*, *noun* and the proposed approaches to generate some sentences to evaluate text quality. These sentences are generated from all the fully Chinese sentences in the testing set (810 sentences for LectureSS and 2211 sentences for SEAME as shown in Table 1). The results are shown in Table 5. In the result of both language models, we observe that the performance of our methods with POS tagging (*proposed+pos*) is far better than *random* on both corpora. It shows that our model has the capability to transform a Chinese sentence into a code-switching sentence with the similar pattern as the training data.

Table 6: *PPL of neural based language model (RNNLM) trained on code-switching training text and data augmented from Chinese training text. The last +pos column indicates considering POS features in proposed method.*

		train	random	noun	proposed	+pos
LectureSS	dev	110.35	107.28	105.37	109.58	**103.37**
	test	73.394	71.779	70.038	71.974	**69.185**
SEAME	dev	75.295	75.307	75.307	**74.474**	75.119
	test	86.088	83.873	85.366	**83.819**	84.358

4.3. Language Modeling

To see whether the data augmentation methods help language modeling, we trained RNNLM which is introduced in Section 4.2 on training data, and evaluated them on the same set of development data and testing data. We do not show the results of n-gram-based LM here because its performance is not comparable with RNNLM as shown in Table 5. Lower perplexity represents better performance on language modeling. We form the augmented training set by combining the generated code-switching sentences with the original training set. The generated code-switching sentences are from the Chinese sentences in original training set (4643 sentences in LectureSS and 20365 sentences in SEAME as shown in Table 1).

Table 6 shows our experimental results. The *train* column in the table represents the perplexity of language model without the augmented code-switching sentences, which is the baseline

of the experiment. As shown in this table, *random* surpasses the baseline on both LectureSS and SEAME testing set. It shows that augmented code-switching text helps language modeling even if the CSPs are randomly selected. *Noun* improves the results only on LectureSS. This may be due to the fact that LectureSS contains lots of CSPs on domain-specific noun, while SEAME has more complicated CSPs.

Proposed+pos performs the best on LectureSS, while *proposed* performs the best on SEAME[3]. It indicates that POS features help generator generate more useful code-switching sentences on LectureSS, but not SEAME. This is because in LectureSS domain-specific terminologies which tend to be code-switched into English are nouns. Meanwhile, SEAME comes from daily life conversation where CSPs are not focused on nouns, resulting in better performance without POS information. Based on Table 6, we demonstrated that the augmented data automatically generated by our method helps language modeling on CS text, and by adding POS features to generator input, our generated data further improves RNNLM on some of the data domains.

4.4. Examples

Some generated code-switching examples are demonstrated in Table 4. The first row is the original code-switching sentence (*ground truth*). We translated it into fully Chinese (*input*). Then, we compared the generated results to *random* and *noun*. The rule-based approach is accurate, but cannot find out all CSPs. The proposed method with POS tagging can find out more CSPs. More examples are in the following link: http://goo.gl/KdBYSy. The examples show that the proposed approach usually generates reasonable code-switching sentences. However, it also generates some terrible sentences. We found that most of them stem from bad translation from Chinese to English.

5. Conclusion

In this work, we try to generate code-switching sentences from monolingual sentences by GAN. The generator can learn to predict CSPs to a great degree without any linguistic knowledge. Moreover, our generated code-switching sentences are better than random generation and rule-based generation. Last but not least, the augmented data by our methods improves RNNLM. For the future work, there is still room for improvement in the translator in this work since wrong translation may lead to terrible generated code-switching sentences. We will further analyze the generator to learn more mechanism about code-switching.

---

[3]We notice that the influence of POS tags in Table 6 and Table 5 are different. However, we believe the results in Table 6 is more crucial because it is the goal of this task. In Table 5, even the model decides not to code-switch any word in the input sentences, it may still obtains reasonable number, but this model would not be very helpful in Table 6.

6. References

[1] Ö. Çetinoğlu, S. Schulz, and N. T. Vu, "Challenges of computational processing of code-switching," *arXiv preprint arXiv:1610.02213*, 2016.

[2] Z. Zeng, H. Xu, T. Y. Chong, E.-S. Chng, and H. Li, "Improving n-gram language modeling for code-switching speech recognition," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2017*. IEEE, 2017, pp. 1596–1601.

[3] H. Xu, T. Chen, D. Gao, Y. Wang, K. Li, N. Goel, Y. Carmiel, D. Povey, and S. Khudanpur, "A pruned rnnlm lattice-rescoring algorithm for automatic speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5929–5933.

[4] R. M. Bhatt, "Code-switching and the functional head constraint," in *Janet Fuller et al. Proceedings of the Eleventh Eastern States Conference on Linguistics. Ithaca, NY: Department of Modern Languages and Linguistics*, 1995, pp. 1–12.

[5] C. W. Pfaff, "Constraints on language mixing: intrasentential code-switching and borrowing in spanish/english," *Language*, pp. 291–318, 1979.

[6] Y. Li and P. Fung, "Code-switch language model with inversion constraints for mixed language speech recognition," *Proceedings of COLING 2012*, pp. 1671–1680, 2012.

[7] ——, "Improved mixed language speech recognition using asymmetric acoustic model and language model with code-switch inversion constraints," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7368–7372.

[8] L. Ying and P. 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.

[9] H. Adel, N. T. Vu, K. Kirchhoff, D. Telaar, and T. 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.

[10] C.-F. Yeh and L.-S. Lee, "An improved framework for recognizing highly imbalanced bilingual code-switched lectures with cross-language acoustic modeling and frame-level language identification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 7, pp. 1144–1159, 2015.

[11] S. Garg, T. Parekh, and P. Jyothi, "Dual language models for code mixed speech recognition," *arXiv preprint arXiv:1711.01048*, 2017.

[12] E. Yilmaz, H. Heuvel, and D. van Leeuwen, "Acoustic and textual data augmentation for improved asr of code-switching speech," in *Proc. Interspeech*. Hyderabad, India: ISCA, 2018, pp. 1933–1937.

[13] S. Garg, T. Parekh, and P. Jyothi, "Code-switched language models using dual rnns and same-source pretraining," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3078–3083.

[14] E. Yılmaz, A. Biswas, E. van der Westhuizen, F. de Wet, and T. Niesler, "Building a unified code-switching asr system for south african languages," *arXiv preprint arXiv:1807.10949*, 2018.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[16] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient." in *AAAI*, 2017, pp. 2852–2858.

[17] R. Williams, "A class of gradient-estimation algorithms for reinforcement learning in neural networks," in *Proceedings of the International Conference on Neural Networks*, 1987, pp. II–601.

[18] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[19] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[20] D.-C. Lyu, T.-P. Tan, E. S. Chng, and H. Li, "Seame: a mandarin-english code-switching speech corpus in south-east asia," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[21] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[23] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 311–318.

[24] C. Wei-Yu Chen, "The mixing of english in magazine advertisements in taiwan," *World Englishes*, vol. 25, no. 3-4, pp. 467–478, 2006.

[25] P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Computational linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[26] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[27] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, vol. 1. IEEE, 1995, pp. 181–184.

[28] A. Stolcke, "Srilm-an extensible language modeling toolkit," in *Seventh international conference on spoken language processing*, 2002.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.