Generating Synthetic Speech from *SpokenVocab* for Speech Translation

Jinming Zhao **Gholamreza Haffari** **Ehsan Shareghi**
Department of Data Science & AI, Monash University
firstname.lastname@monash.edu

Abstract

Training end-to-end speech translation (ST) systems requires sufficiently large-scale data, which is unavailable for most language pairs and domains. One practical solution to the data scarcity issue is to convert text-based machine translation (MT) data to ST data via text-to-speech (TTS) systems.Yet, using TTS systems can be tedious and slow. In this work, we propose *SpokenVocab*, a simple, scalable and effective data augmentation technique to convert MT data to ST data on-the-fly. The idea is to retrieve and stitch audio snippets, corresponding to words in an MT sentence, from a spoken vocabulary bank. Our experiments on multiple language pairs show that stitched speech helps to improve translation quality by an average of 1.83 BLEU score, while performing equally well as TTS-generated speech in improving translation quality. We also showcase how *SpokenVocab* can be applied in code-switching ST for which often no TTS systems exit.[1]

1 Introduction

End-to-end (E2E) speech-to-text translation (ST) models require large amounts of data to train (Sperber and Paulik, 2020). Despite the emerging ST datasets (Cattoni et al., 2021; Wang et al., 2021), their size is considerably smaller compared to text-based machine translation (MT) data. A common remedy to tackle the data scarcity issue is to leverage text-based MT data in training ST systems. Common approaches include multi-task learning (Anastasopoulos and Chiang, 2018; Ye et al., 2021), transfer learning & pretraining (Bansal et al., 2019; Wang et al., 2020) and knowledge distillation (Inaguma et al., 2021; Tang et al., 2021).

A more straightforward alternative is to convert text-based MT data to ST via text-to-speech (TTS) synthesis engines (Pino et al., 2019; Jia et al., 2019).

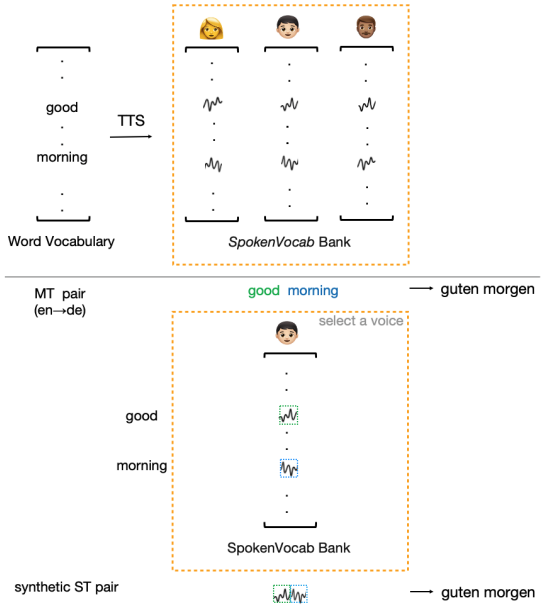

Figure 1: Overview of generating synthetic speech from SpokenVocab on-the-fly. The first step is to prepare the SpokenVocab bank offline and the second step is to retrieve and stitch audio snippets from the bank by words in a sentence.

This method is less commonly used despite its simplicity and effectiveness,[2] mainly due to practical reasons: (i) TTS models have slow inference time and may incur monetary costs; (ii) the conversion is required for each MT datasets. Recently, Lam et al. (2022) proposed to generate synthetic speech without using TTS models. However, their approach is based on real ST data, and thus cannot be extended to MT data.

In this work, we propose a simple, effective and efficient data augmentation approach to convert MT data to ST data on-the-fly. The idea is to prepare a set of spoken words, forming a spoken vocabulary (*SpokenVocab*) bank, and then generate synthetic speech by retrieving and stitching spoken

[1]Our code is available at https://github.com/mingzi151/SpokenVocab

[2]Only one work out of 8 uses TTS to augment data in the IWSLT2022 offline speech translation track.

Findings of the Association for Computational Linguistics: EACL 2023, pages 1975–1981

words based on a text sequence, as shown in Figure 1.[3] Our experiments show that this method is as effective as TTS-generated speech, at a much lower computational and financial cost. For instance, augmenting ST data on-the-fly with 100k of stitch-converted MT data, boosts translation quality by an average of 1.83 BLEU over 3 language pairs from Must-C (Cattoni et al., 2021) with no additional cost, memory, or speed footprints. Comparing the real ST data vs. our converted version from the same transcripts, to our positive surprise, revealed that our synthetic data outperforms its real counterpart by 0.41 BLEU score. We conduct thorough experiments to examine *SpokenVocab* in boosting translation and further showcase its use and benefit in the context of code-switching (CS) ST.

We hope this simple technique to ease the use of MT data for ST in practice as well as other tasks where synthetic speech is useful.

2 SpokenVocab

We describe our methodology in creating effective synthetic ST data based on MT data in this section. The core step is the preparation of a *SpokenVocab* bank offline and stitching sounds on-the-fly.

Concretely, we first use a TTS engine to convert items in a word vocabulary to speech to obtain a set of *SpokenVocab* offline.[4] Next, we can configure the TTS engine to generate different speaker voices and thus curate a *SpokenVocab* bank in which each set corresponds to a "speaker". The purpose is to simulate, to the greatest extent, a realistic speech dataset consisting of various speakers. At training, assume we have access to an MT dataset and each pair denoted as $< s, t >$ where s and t are source and targets sentences, respectively. Given such a pair, we choose one voice [5] from the bank, and produce synthetic speech by fetching corresponding audio snippets by words in s from the bank and stitching them together. During stitching, we deploy cross-fade, a well-known technique to smooth transitions between two independent audio clips.[6] Pairing it with t yields a synthetic ST instance.[7]

3 Experiments

We first present the ST system (§3.1) and TTS systems (§3.1.2) used in this study. We then describe the ST and MT datasets (§3.1.3), followed by providing implementation details (§3.1.4). Next we explain how *SpokenVocab* is designed (§3.2) and report translation results (§3.3). Lastly, we illustrate how our method can be applied to CS ST (§3.5).

3.1 Experimental Setup

3.1.1 Speech Translation System

Pre-trained speech encoders and text decoders have shown great performance on ST (Li et al., 2021; Zhao et al., 2022), compared to models trained from scratch. For this reason, we follow the architecture in Gállego et al. (2021) that uses Wav2vec 2 (W2V2) (Baevski et al., 2020) as the speech encoder and mBart decoder (Liu et al., 2020) as the text decoder, joint with a lightweight linear adapter and a CNN-based length adapter.

3.1.2 TTS Systems

To prepare *SpokenVocab*, we use the Google TTS service,[8] which supports a wide range of voice configurations; this allows simulating different speakers with various accents, gender and geographical background. We also use a off-the-shelf TTS toolkit, i.e., Tacotron2-DCA + Mulitband-Melgan (short for T2+Mel).[9] We use Google TTS to generate synthetic speech in raw wavforms.

3.1.3 Dataset

We conduct our major experiments on Must-C, a multilingual ST dataset curated from Ted talks. We focus on English (En)→German (De), Romanian (Ro) and Italian (It). For MT data, we use a subset of WMT14, WMT16 and OPUS100[10] for De, Ro and It, with 100k, 100k and 24k instances, respectively. For the code-switching (CS) setting, we use Prabhupadavani (Sandhan et al., 2022), multilingual CS ST dataset, and we focus on En→De, It. Its source utterances are code-mixed with English (major language), Bengali and Sanskrit; each utterance is translated manually to 25 languages. We

[3] During the writing of this manuscript we found out that Voder, the first electronic speech synthesiser developed by Bell Labs in 1939, synthesized human speeches by decomposing it into its acoustic components and combining them using human operators in real time.

[4] SpokenVocab could also be based on n-grams in a dataset.

[5] One could also generate utterances by mixing speakers at the token level, with no additional cost with our technique. We leave further investigation of this to future work as it requires a test condition (i.e., including various speaker voices per utterance) which is not available to the best of our knowledge.

[6] https://github.com/jiaaro/pydub

[7] We provide a demo for stitched speeches.

[8] https://cloud.google.com/text-to-speech

[9] https://github.com/mozilla/TTS

[10] http://opus.nlpl.eu/opus-100.php

prepare ST data following the instructions in Gállego et al. (2021). We preprocess MT data with the fairseq instructions and remove pairs with the length of target sentences greater than 64 words to avoid out-of-memory issues. Minimal preprocessing is performed on the CS ST dataset.

3.1.4 Implementation Details

Similar to Li et al. (2021) and Gállego et al. (2021), training different components of W2V2 and mBart decoder yields divergent results. In our initial experiments, we note that fine-tuning the entire W2V2 except for its feature extractor and freezing mBart lead to decent translation results, and thus we use this configuration for all our experiments. To ensure Must-C to be dominant, we make the ratio of Must-C and MT data to be approximately 8:1, unless mentioned otherwise. We use sacreBLEU (Post, 2018) to evaluate translation. Please refer to *Appendix* A.1 for full training details, hyper-parameters and hardware.

3.2 SpokenVocab Preparation and Variations

Constructing the *SpokenVocab* bank is crucial, as synthetic speech produced in this manner have a direct impact on translation quality. In this section we examine *SpokenVocab* from various dimensions.

TTS Conversion. The first questions to ask are which TTS system should be used to convert a word to a spoken form and what sampling rate (SR) is appropriate.[11] To answer these questions, we conduct intrinsic evaluation on stitched speech by varying TTS engines and SR. Furthermore, as it is common to diversify raw wave forms with audio effects (Potapczyk et al., 2019), we apply the same technique to distort our stitched speech. Results in Table 1 show that using Google TTS and setting the SR to 24k are better choices, while distortion (i.e., adding the effects of tempo, speed and echo) may or may not be helpful. Contrary to the common practice of using a SR of 16k (Baevski et al., 2020), applying 16k to *SpokenVocab* alters the sound significantly, as shown in the demo in §2, and this has negative impacts on the system. Overall, we use the setting in *italic* for the rest of our experiments.

Word Vocabulary. We compile a word vocabulary, consisting of 1) a common subset of words[12], and 2) unique words with a frequency of higher than 99 from the En→X WMT subset. The purpose is to construct an approximated version of *SpokenVocab* that is ready to convert any sentence to synthetic speech. For words that are not covered by the list, we employ a fuzzy matching mechanism where the most similar word at the surface level is returned. For instance, an out-of-vocabulary (OOV) word "apples" is replaced by its closest match in the vocabulary "apple", and the speech snippet for "apple" is retrieved. When no match is found, a default filter word, "a", is returned. To investigate the effect of this approximation which would inevitably lead to mispronounced words, we prepare another set of *SpokenVocab* containing the full set of spoken words in the WMT data (eliminating the need for fuzzy matching). In controlled experiments on En→De, the BLEU scores with the approximated and full *SpokenVocab*s, with the size of 35k and 460k respectively, are 28.02 and 27.91. The negligible difference indicates the effectiveness of using an approximated *SpokenVocab*. Additional ablation studies on using 50% and 10% of the full vocabulary yield scores of 27.79 and 27.94, further validating the insensitivity of W2V2 to nuanced mispronunciation, perhaps due to the presence of powerful pre-trained auto-regressive decoder.[13]

Data	TTS	SR	Distort.	BLEU
ST	-	-	-	26.91
ST+MT$_{\texttt{stitched}}$	T2+Mel	-	-	OOM
	Google	*24k*	-	**28.02**
		24k	✓	27.72
		16k	-	26.77
		16k	✓	27.47

Table 1: Comparison of different TTS conversions in terms of TTS engine, sampling rate (SR) and distortion (Distort.) Top row: baseline. Bottom rows: MT data is converted to ST data with *SpokenVocab*. OOM: out-of-memory with 24k and 16k SRs. *italic*: best setting.

Number of Speakers. Despite the artificial nature of the stitched speech sounds, one still can tell the speaker's information (e.g., gender, accent). To examine whether diverse voices would be helpful for translation, we set n to 1, 5 and 10 and train models with the same amount of data. These sys-

[11] SR is defined as the number of samples taken from a continuous signal per second.

[12] The list comes from Official Scrabble Players Dictionary and Wiktionary's word frequency lists, and can be found at https://github.com/dolph/dictionary/blob/master/popular.txt

[13] Optionally, one can dynamically call a TTS system to generate an audio on OOV words.

Data	Cost			BLEU		
	⏲	$	⛁	De	Ro	It
ST		-		26.91	24.66	22.13
ST + MT_{TTS}	900	90	25	**28.20**	24.71	**26.46**
ST + $MT_{stitched}$	9	0	0	28.02	**25.05**	26.13

Table 2: Translation quality on Must-C and the average costs associated for generating synthetic speech for *every 100k sentences* in terms of inference time in minutes (⏲), USD value ($) and storage required in GB (⛁). Preparing *SpokenVocab* took 2 hours, free of charge, with Google TTS, and stitched speeches are discarded.

tems display similar translation performance with 28.02, 27.73 and 27.80 BLEU scores respectively, suggesting that having a single speaker is sufficient. Our conjecture to this phenomenon is that speech representations produced by W2V2 have removed speaker information, as demonstrated in Nguyen et al. (2020) where analysis was conducted on wav2vec (Schneider et al., 2019), the predecessor to W2V2. This could be further examined with using dialect- or pronunciation-focused translation settings, which we leave to future work.

3.3 Translation Performance on Must-C

Producing synthetic speech from *SpokenVocab* on-the-fly makes the conversion from text to speech highly scalable in terms of time and monetary costs, and it also avoids the need of storing speech. Table 2 reports the time, dollar value and space required to produce every 100k speech with Google TTS, while these numbers are negligible for *SpokenVocab* due to its re-usability.[14] Apart from scalability, it is more important to see the translation performance difference between unnatural speech produced by *SpokenVocab* and fluent speech generated by state-of-the-art TTS systems. Table 2 summarises results for 3 Must-C language pairs, with stitched speech and TTS-generated speech. As expected data augmentation of ST with MT data method boosts translation quality, using our method by 1.83 BLEU score on average. Our stitched speech performs equally well as TTS-generated counterpart, showing no loss of quality during conversion.

[14]For fair comparison between TTS which operates on the full vocabulary, we report the cost under the full vocabulary version of our method.

Data	Nature of Speech	BLEU
Must-C	real	26.91
Must-C + Europarl	real + real	27.5
Must-C + $Europarl_{TTS}$	real + synthetic	27.76
Must-C + $Europarl_{stitched}$	real + synthetic	**27.91**

Table 3: BLEU scores under different augmentations.

		ST_{CS}	ST_{CS}+$MT_{CS-stitched}$
BLEU	En-Be→De	26.11	28.09
	En-Be→It	26.41	26.90

Table 4: Translation quality for CS ST dataset.

3.4 Stitched Speech vs. Real Speech

An alternative approach to augmentation is to leverage real ST data from any other existing domains. To assess whether our approach as another augmentation technique is still competitive, we conduct an experiment on En→De by augmenting Must-C with 35k training instances from the Europarl-ST (Iranzo-Sánchez et al., 2020). Table 3 reports the results. To our positive surprise, our stitched speech (generated from the transcripts of eurorparl-ST counterpart) works even better than the real Europarl-ST speech.

3.5 Code-switching Speech Translation

Development in CS ST is constrained by the availability of relevant datasets (Sandhan et al., 2022) and using TTS systems to augment data is practically difficult. To this end, our method provides a high degree of flexibility in that it can stitch audio clips of different languages freely. To produce a code-switched utterance, we further prepare *SpokenVocab* for Bengali (Google TTS does not support Sanskrit) based on an English-Bengali dictionary.[15] We maintained the ratio of code-switching in the real data (i.e., 0.35 probability of CS occurring, and 2 as the average number of code-switched words in a sentence). Please see Algorithm 1 in *Appendix* A.2 for the detailed utterance generation process. Results in Table 4 suggest that the models trained with additional 100k and 24k instances (for De and It respectively.) from *SpokenVocab* outperform those only trained with the original data.

[15]https://github.com/MinhasKamal/BengaliDictionary

4 Conclusion

In this work, we proposed a simple, fast and effective data augmentation technique, *SpokenVocab* for ST. This provides an alternative for converting MT data to ST data with TTS systems which comes with monetary and computation costs in practice. Our approach generates synthetic speech on-the-fly during training, with no cost or footprint. We have shown that speech stitched from *SpokenVocab* works as effective as TTS-generated speech, and unlike TTS system, it could directly be applied as a data augmentation tool in code-switching ST. Our approach can be used in other content-driven speech processing tasks as an uncompromising and easy-to-use augmentation technique.

Limitations

CS ST exbihit difficulties (Huber et al., 2022; Weller et al., 2022), exposing several limitations in this study: 1) Bengali and Sanskrit (another minority language) are treated without difference, as they originate from the same script and Sanskrit is not supported by the Google TTS service. 2) We use a open-source language detection tool to calculate the oracle hyper-parameters in the dev set; yet, imperfection of the detector on token-level prediction and the fact that source sentences are written in Latin regardless of the language deviate the scores from true values.

References

Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 82–91.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *arXiv preprint arXiv:2006.11477*.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2019. Pre-training on high-resource speech recognition improves low-resource speech-to-text translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 58–68.

Roldano Cattoni, Mattia Antonino Di Gangi, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2021. Must-c: A multilingual corpus for end-to-end speech translation. *Computer Speech & Language*, 66:101155.

Gerard I Gállego, Ioannis Tsiamas, Carlos Escolano, José AR Fonollosa, and Marta R Costa-jussà. 2021. End-to-end speech translation with pre-trained models and adapters: Upc at iwslt 2021. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 110–119.

Christian Huber, Enes Yavuz Ugan, and Alexander Waibel. 2022. Code-switching without switching: Language agnostic end-to-end speech translation. *arXiv preprint arXiv:2210.01512*.

Hirofumi Inaguma, Tatsuya Kawahara, and Shinji Watanabe. 2021. Source and target bidirectional knowledge distillation for end-to-end speech translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1872–1881.

Javier Iranzo-Sánchez, Joan Albert Silvestre-Cerda, Javier Jorge, Nahuel Roselló, Adria Giménez, Albert Sanchis, Jorge Civera, and Alfons Juan. 2020. Europarl-st: A multilingual corpus for speech translation of parliamentary debates. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8229–8233. IEEE.

Ye Jia, Melvin Johnson, Wolfgang Macherey, Ron J Weiss, Yuan Cao, Chung-Cheng Chiu, Naveen Ari, Stella Laurenzo, and Yonghui Wu. 2019. Leveraging weakly supervised data to improve end-to-end speech-to-text translation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7180–7184. IEEE.

Tsz Kin Lam, Shigehiko Schamoni, and Stefan Riezler. 2022. Sample, translate, recombine: Leveraging audio alignments for data augmentation in end-to-end speech translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 245–254.

Xian Li, Changhan Wang, Yun Tang, Chau Tran, Yuqing Tang, Juan Pino, Alexei Baevski, Alexis Conneau, and Michael Auli. 2021. Multilingual speech translation from efficient finetuning of pretrained models. 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 827–838.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Ha Nguyen, Fethi Bougares, Natalia Tomashenko, Yannick Estève, and Laurent Besacier. 2020. Investigating self-supervised pre-training for end-to-end speech translation. In *Interspeech 2020*.

Juan Pino, Liezl Puzon, Jiatao Gu, Xutai Ma, Arya D McCarthy, and Deepak Gopinath. 2019. Harnessing indirect training data for end-to-end automatic speech translation: Tricks of the trade. In *Proceedings of the 16th International Conference on Spoken Language Translation*.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.

Tomasz Potapczyk, Paweł Przybysz, Marcin Chochowski, and Artur Szumaczuk. 2019. Samsung's system for the iwslt 2019 end-to-end speech translation task. In *Proceedings of the 16th International Conference on Spoken Language Translation*.

Jivnesh Sandhan, Ayush Daksh, Om Adideva Paranjay, Laxmidhar Behera, and Pawan Goyal. 2022. Prabhupadavani: A code-mixed speech translation data for 25 languages. *arXiv preprint arXiv:2201.11391*.

Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. 2019. wav2vec: Unsupervised pre-training for speech recognition. In *INTERSPEECH*.

Matthias Sperber and Matthias Paulik. 2020. Speech translation and the end-to-end promise: Taking stock of where we are. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7409–7421.

Yun Tang, Juan Pino, Xian Li, Changhan Wang, and Dmitriy Genzel. 2021. Improving speech translation by understanding and learning from the auxiliary text translation task. 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 4252–4261.

Changhan Wang, Anne Wu, Jiatao Gu, and Juan Pino. 2021. Covost 2 and massively multilingual speech translation. In *Interspeech*, pages 2247–2251.

Chengyi Wang, Yu Wu, Shujie Liu, Ming Zhou, and Zhenglu Yang. 2020. Curriculum pre-training for end-to-end speech translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3728–3738.

Orion Weller, Matthias Sperber, Telmo Pires, Hendra Setiawan, Christian Gollan, Dominic Telaar, and Matthias Paulik. 2022. End-to-end speech translation for code switched speech. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1435–1448.

Rong Ye, Mingxuan Wang, and Lei Li. 2021. End-to-end speech translation via cross-modal progressive training. *Proc. Interspeech 2021*, pages 2021–1065.

Jinming Zhao, Hao Yang, Ehsan Shareghi, and Gholamreza Haffari. 2022. M-adapter: Modality adaptation for end-to-end speech-to-text translation. *arXiv preprint arXiv:2207.00952*.

A Appendix

A.1 Implementation Details

We implement and train all models with fairseq[16] on 4 A40 GPUs, using 16 floating point precision, for $25k$ updates. WAV2VEC 2[17] and the mBart50[18] decoder are used. We employ an Adam optimizer with $\beta_1 = 0.99$, $\beta_2 = 0.98$, while setting the dropout to 0.1, clip norm to 20 and label smoothing to 0.2. For the baseline models, we use a learning rate of 5e-04 and reduce it at plateau. For models trained with additional data, we use the same learning scheduler with a learning rate of 3e-04.

A.2 Code-switching Speech Translation

Algorithm 1 Code-switching Utterance Generation

Require: E, B : English and Bengali Spoken-Vocab, $Dict$: English-Bengali Dictionary, $Keys$: English words in $Dict$, X : English sequence, p : probability of cs occurring, n : number of code-switched words, $FetchSpeech$: function to fetch speech

Output: U: CS utterance

$q =$ NormDist$(0, 1)$
if $q > p$ **then**
  // Select words to be code-switched
  $words, indices =$ Random(X, n)
  for $word, i$ in $words, indices$ **do**
    // Only switch words in the dictionary
    if $word$ in $Keys$ **then**
      // Replace with the Bengali word
      $X[i] = Dict[word]$
    end if
  end for
end if
$U =$ FetchSpeech(E, B, X)
return U

[16] https://github.com/facebookresearch/fairseq

[17] https://dl.fbaipublicfiles.com/fairseq/wav2vec/wav2vec_vox_960h_pl.pt

[18] https://dl.fbaipublicfiles.com/fairseq/models/mbart50/mbart50.ft.1n.tar.gz