Improving Speech Recognition with Jargon Injection

Minh-Tien Nguyen[1,2], **Phuoc-Dat Nguyen**[1], **Tuan-Hai Luu**[1], **Xuan-Quang Nguyen**[1], **Tung-Duong Nguyen**[1], **Jeff Yang**[1]
[1]Cinnamon AI, 10th floor, Geleximco building, 36 Hoang Cau, Dong Da, Hanoi, Vietnam.
{ryan.nguyen, barb, sam, albert, neo, jeff.yang}@cinnamon.is
[2]Hung Yen University of Technology and Education, Hung Yen, Vietnam.
tiennm@utehy.edu.vn

Abstract

This paper introduces a new method that improves the performance of Automatic speech recognition (ASR) engines, e.g., Whisper in practical cases. Different from prior methods that usually require both speech data and its transcription for decoding, our method only uses jargon as the context for decoding. To do that, the method first represents the jargon in a trie tree structure for efficient storing and traversing. The method next forces the decoding of Whisper to more focus on the jargon by adjusting the probability of generated tokens with the use of the trie tree. To further improve the performance, the method utilizes the prompting method that uses the jargon as the context. Final tokens are generated based on the combination of prompting and decoding. Experimental results on Japanese and English datasets show that the proposed method helps to improve the performance of Whisper, specially for domain-specific data. The method is simple but effective and can be deployed to any encoder-decoder ASR engines in actual cases. The code and data are also accessible.[1]

1 Introduction

Automatic speech recognition (ASR) is the task of automatically transcribing input audio to output text (Radford et al., 2023; O'Shaughnessy, 2024). The output of ASR systems can be used in several applications such as intelligent personal assistants (McGraw et al., 2016; He et al., 2019), voice searches (Chiu et al., 2018), or meeting analyses (Yu et al., 2020; Song et al., 2020; Jung et al., 2023; Li et al., 2023; Rennard et al., 2023). Recently, the performance of end-to-end ASR models has been improved by several approaches such as connectionist temporal classification (Graves et al., 2006; Graves and Jaitly, 2014), recurrent neural network transducer (Graves, 2012), attention-based encoder-decoder (Chorowski et al., 2015; Chan et al., 2016; Dong et al., 2018) with strong ASR engines (Gulati et al., 2020; Han et al., 2020). Among those, Whisper has shown strong performance for speech recognition (Radford et al., 2023). It was trained with 680,000 hours of labeled audio data with multitasking and multilingual learning.

Strong ASR engines such as Whisper have achieved promising results in English, yet, we observe the decent accuracy of ASR engines applied to actual business, especially for low-resource languages, e.g., Japanese. To fill the gap, there are two possible solutions for domain adaptation. The first well-known solution is to continuously fine-tune ASR engines with domain-specific data (Huang et al., 2021; Javed et al., 2022). However, creating training corpora (including speech and text) data is a non-trivial task that is time-consuming and labor-expensive. In many cases, the creation requires domain experts, especially for narrow specific domains, e.g., high-pressure gas incidents. Also, fine-tuning is a complex process that requires skilled practitioners (Radford et al., 2023). The second solution is to consider domain-specific data as a context and inject the context into the decoding phase of ASR engines (Pundak et al., 2018; Zhao et al., 2019; Alon et al., 2019; Le et al., 2021b,a; Sun et al., 2021; Han et al., 2022). Among them, biasing methods are simple and potential to inject a context into the ASR process. However, these methods are usually used with hybrid ASR (Pironkov et al., 2020) or CTC end-to-end models (Graves and Jaitly, 2014) which are behind the performance of encoder-decoder ASR models such as Whisper.

This paper addresses the problem of improving the performance of ARS engines by using jargon. The problem comes from the fact that in practical cases, only jargon (domain-specific terms) is provided by clients. The jargon only includes specific words and phrases without the availability of speech data and domain-specific text. It challenges pre-trained ASR models, e.g., Whisper, and cur-

[1]https://shorturl.at/YiBUr

rent methods of contextual speech recognition that usually require both speech and text data. To address the problem, we introduce a new method that injects domain-specific knowledge in the form of jargon into the decoding phase of ASR engines. To do that, the method uses Whisper as the backbone and jargon represented as a trie tree as the domain-specific context. By utilizing it as a form of the tree to manipulate the beam search decoding process and a prompt to give instructions to Whisper, the method improves the performance on various datasets and the appearance of jargon in the final output. The method does not require speech data for fine-tuning ASR models, so that facilitates the deployment in actual cases. In summary, the paper makes two main contributions as follows.

- It introduces a method that injects the jargon into the beam search decoding by boosting the score of the beam that includes tokens in the jargon. The method is further supported by the initial prompt method offered by Whisper. The method is simple, effective, and easy to adapt with any encoder-decoder ASR engines.
- It validates the efficiency of the method on Japanese and English datasets. Experimental results show that domain knowledge injection helps to improve the quality of ASR engines.

2 Related Work

ASR The recent success of deep neural networks has been contributed to improve the performance of ASR. Approaches range from traditional methods such as connectionist temporal classification (CTC) (Graves et al., 2006; Graves and Jaitly, 2014), recurrent neural network transducer (Graves, 2012), attention-based encoder-decoder (Chorowski et al., 2015; Chan et al., 2016; Dong et al., 2018), to sequence-to-sequence models (Chiu et al., 2018). These approaches leverage the development of strong ASR engines (Gulati et al., 2020; Han et al., 2020; Radford et al., 2023) trained by the Transformer architecture (Vaswani et al., 2017) such as SeamlessM4T (Barrault et al., 2023) or Whisper (Radford et al., 2023). We used Whisper as the main backbone of our method because of its efficiency for ASR in domain-specific Japanese data.[2]

Context-aware ASR has recently been used to improve the quality of ASR (Williams et al., 2018; Pundak et al., 2018; Zhao et al., 2019; Alon et al., 2019; Le et al., 2021a,b; Han et al., 2022; Jung et al., 2022). The context can be the text of testing data (Han et al., 2022) or a list of biasing phrases (Zhao et al., 2019; Alon et al., 2019; Pundak et al., 2018). There are two main directions. The first is to bias the decoding of ASR models by using shallow fusion methods (Zhao et al., 2019; Le et al., 2021b,a). The fusion methods create a finite state transducer (FST) created from the list of biasing phrases and use the FST to adjust the decoding process without adding any neural networks. In contrast, the second usually encodes the context by using the encoder and then uses attention to change the probability of tokens in the decoding phase (Han et al., 2022). Between the two directions, TCPGen (Sun et al., 2021) introduced a tree-constrained pointer generator that incorporates a list of biasing words into both attention encoder-decoder and transducer end-to-end ASR models.

The method of contextual speech recognition is perhaps the most relevant to our work (Williams et al., 2018). The method adjusts the output likelihoods of a neural network at each step in the beam search by a sequence probability computed from n-grams. While sharing the idea of using shallow fusion, our proposed method distinguishes two main points. First, we consider a small dictionary rather than using the text of testing data to create n-grams language models (LMs) as Williams et al. (2018). It makes our task to be more challenging. Second, in the decoding phase, we modify the probability of a token appearing in the jargon while Williams et al. (2018) just simply used shallow fusion with the probability of LMs. We follow the shallow fusion approach because it is simple but effective and is appropriate for ASR in business cases.

3 Proposed Method

3.1 Problem Statement

The problem is to improve the performance of an encoder-decoder ASR engine, e.g., Whisper by taking into account jargon when doing decoding. We define the jargon as a dictionary D that includes domain-specific tokens used in actual businesses, e.g., technical terms used in the high-pressure gas incident domain. Precisely, given the jargon D, we design a method that adjusts the decoding process of Whisper to inject D for speech recognition.

Figure 1 (right) introduces the proposed decoding method. The input speech is processed by an

[2]Whisper gives better performance than SeamlessM4T for domain-specific Japanese datasets in the internal testing.

ASR engine. The decoder adjusts the probability of tokens by considering the jargon represented in a trie tree. The initial prompt method is also combined to further improve the quality of ASR.

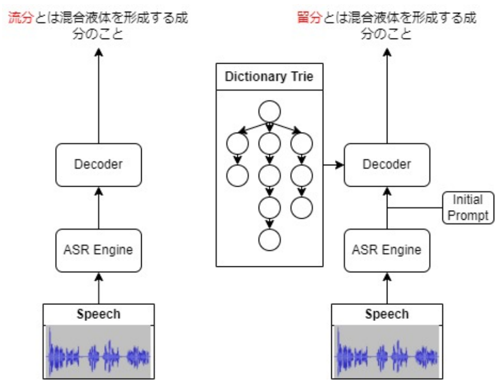

Figure 1: Whisper (left) and our proposed method (right). The inclusion of "留分" increases our model recognition. The Japanese sentence means "Part of distillation is the component that form the liquid mixture".

3.2 Whisper

There are several attention-based encode-decoder models have been released recently and achieved good results on various datasets, such as Seamless4MT (Barrault et al., 2023). While we use Whisper (Radford et al., 2023) as the backbone of our method, our method can be applicable to all models within this family. We select Whisper because of its strong performance of speech recognition in many languages, especially in Japanese. Whisper is a sequence-to-sequence model, based on the Transformer architecture. It consists of two transformers. As an encoder-decoder ASR engine, the first transformer encodes audio information and the second one encodes linguistic information. The model is jointly trained to predict a sequence of tokens by the decoder for many different speech-processing tasks including multilingual speech recognition, speech translation, spoken language identification, and voice activity detection. The decoder predicts each next token by conditioning on previously processed tokens with that token's probability of the model. During inference, Whisper generates a sequence of output labels given an input speech that maximizes the likelihood probability distribution.

$$y^* = \arg\max_y p(y|x) \tag{1}$$

with $p(y|x)$ being the output probability distribution from Whisper.

3.3 Jargon Injection

The strong performance of pre-trained attention-based encoder-decoder ASR models, e.g., Whisper facilitates the deployment of ASR models, yet, domain adaptation makes a challenge for the deployment in actual cases. To fill this gap, a straightforward method is to fine-tune ASR models with domain-specific data. However, this method requires training data that includes both speech and its transcription. In practical cases, creating this training data is a time-consuming and non-trivial task that requires domain experts. Therefore, we come up with another direction that directly injects domain-specific jargon for domain adaptation. This section shows the proposal of injecting jargon into the decoding process of Whisper.

Jargon representation In business cases, jargon refers to a set of tokens that is uncommon with out-domain people, causing difficulty for an ASR model to recognize correctly or often be mistaken with similar tokens, e.g., 留分 ("part of distillation") and 流分 (does not exist in Japanese dictionary). In almost all cases, the jargon is usually created by humans in specific domains.

To represent the jargon D for decoding, we use the trie tree, which is one of most efficient methods for representing a collection of strings (Le et al., 2021a). Each node in a tree is associated with a character. The root node is associated with an empty string and children of a node will share a common prefix with its parent node. Figure 2 shows an example of a trie tree that represents "eat, "can", "could", "count", and "card" tokens. Representing D as a trie allows our method to efficiently

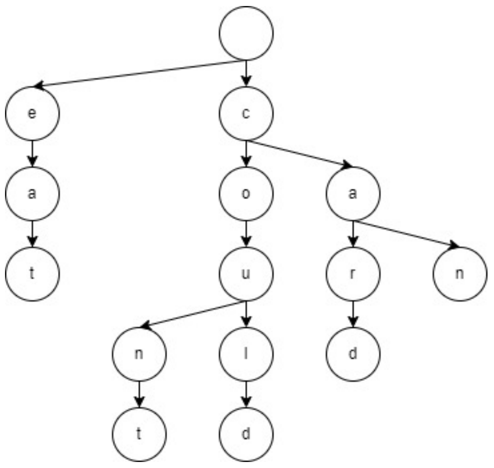

Figure 2: A trie sample made from "eat, "can", "could", "count", and "card" words.

look up jargon during decoding. The trie stores a pointer for each sequence. Whenever the decoding method extends a sequence in the beam, it moves this pointer down to the tree. If it encounters either a leaf node or a node marked as jargon-boundary, the method permanently adds a score to the total score of that sequence using Eq. (4).

Contextual beam search The conventional method in Eq. (1) provides a good way for decoding the next tokens, yet, encoder-decoder ASR engines only support generating step-wise distribution $p(y_n|y_{n-1}, ...y_0, x)$. If we brute-force this formula, checking all possible sequences and computing probabilities from them, it would take a huge amount of time. That's why, it's more common to use beam search to approximate Eq. (1) while limiting the search space. Beam search is conducted as follows. Let k be the beam size, T is the number of distinct tokens generated from ASR engines, the beam search starts with one single sequence that consists of <sos>-equivalent tokens and its scores as $p(\text{<sos>}|x)$. When decoding, the beam search extends the sequence by appending only top k tokens with the best scores among all possible T tokens that ASR models can generate, adding all k sequences back to our beam. Next, these k sequences are again appended with the next top k tokens, resulting k^2 sequences in total. The beams are sorted according to their probability to retain only the best k sequences with the highest probability. The decoding process ends when the <eos>-equivalent token appears. This process is repeated until all sequences in beams are locked.

We extend the original beam search to define our contextual beam search. The idea is to adjust the probability of generated tokens by using the jargon D. In Eq. (1), the best sequence is only generated based on the output probability distribution from ASR. To take into account the additional context D for the beam search, Eq. (1) is modified as follows.

$$y^* = \arg\max_y \frac{p(y|x)}{p_D(y)^\alpha} \tag{2}$$

where $p_D(y)$ is a distribution calculated based on the jargon D and α is a tunable hyperparameter controlling the contribution of the context D. In practice, running beam search decoding by using Eq. (2) often causes numerical instability. Therefore, we transform Eq. (2) by using the logarithm version of the equation as follows.

$$y^* = \arg\max_y log(p(y|x)) - \alpha log(p_D(y)) \tag{3}$$

Eq. (3) shows our decoding method. It includes two main parts: one from the original ASR models and another from the jargon D. The main idea of this equation is to boost the score of sequence based on prior distribution $p_D(y)$. Since D is a list of specific tokens, Eq. (3) cannot calculate prior distribution by modeling it as a language model. Instead, we artificially boost the output probability whenever the process reaches the boundary of jargon. As a result, Eq. (3) is written as follows.

$$y^* = \arg\max_y log(p(y|x)) - \alpha S_D(y, x) \tag{4}$$

with $S_D(y, x)$ is defined as follows.

$$S_D(y, x) = \sum_{0 \le i \le j < n} \sum_{k=i}^{j} log(p(y_k|x)), \text{ if } y_{i..j} \in D \tag{5}$$

with n being the length of sequence y. In short, $S_D(y, x)$ is calculated by summing the output probability of every possible jargon in D containing in the sequence y. Therefore, all sequences that contain the jargon will have their boosted scores, increasing their chance of appearing in the final beam of the beam search decoder. At each step of decoding, calculating Eqs. (4) and (5) is computationally expensive because of storing and finding candidate tokens in D. Therefore, we use the trie data structure in Figure 2 for efficient string lookup.

Initial prompt The initial prompt is one of the prominent features of Whisper. The prompt serves as the previous context of the current speech frames. By appending a context C before decoding, Whisper is indirectly biased into the prompt. The context C is composed of two parts: an instruction and the jargon. The detailed context is shown in Section 4.2. To use C, Eq. (1) is written as follows.

$$y^* = \arg\max_y log(p(y|x, C)) \tag{6}$$

Combination Since context text C is appended before decoding and does not change during the beam search, we combine the initial prompt and our method for the best result. The combination offers a global optimization that takes into account static information from the prompt (prefix) and dynamic probabilities from our decoding method (suffix). It forces Whisper to more focus on the jargon. Therefore, Eq. (4) is written as follows.

$$y^* = \arg\max_y log(p(y|x, C)) - \alpha S_D(y, x, C) \tag{7}$$

with $S_D(y, x, C)$ is defined similarly to Eq. (5).

4 Experimental Setup

4.1 Datasets

We validated the proposed method on three Japanese datasets: two in-house corpora and two benchmark dataset, and one English corpus.

HGP HGP is a smaller set of the original high-gas incident corpus published in 2022 by the High-Pressure Gas Safety Institute of Japan. From 18,171 incident cases, we extracted 1,500 incident reports in three industries: "general chemistry", "petrochemical", and "oil re- fining" (Inoue et al., 2023). HGP's dictionary was made by automatically comparing the difference between the Whisper model output and the validation ground truth. We took only tokens that exist at least one time (or higher) in the Japanese dictionary. Due to the term,[3] we can not disclose the dataset.

GPT synthesis GPT synthesis is an in-house dataset that consists of 200 audios imitating HGP dataset's style. Its content was both handcrafted and assisted with OpenAI's ChatGPT. The dictionary was made manually by a domain expert, targeting technical terms in the incident domain.

JNAS (Itou et al., 1999) is a public Japanese dataset that consists of 16679 utterances, spoken by 306 speakers, with half of them are male (16,176 by reading Mainichi Newspaper and 503 from ATR 503 PB-Sentences). The JNAS's dictionary was made with the same method as HGP. We compared the difference between the output of Whisper with the gold label of the validation dataset.

Table 1: Statistics of test sets of four databases. JA stands for Japanese and EN is English.

Dataset	Samples	Jargon	Domain	Lang
HGP	1467	150	Incident	JA
GPT-Syn	200	47	Incident	JA
JNAS	2253	157	Newspaper	JA
LibriSpeech	2620	146	Audiobook	EN

LibriSpeech (Panayotov et al., 2015) is a public English dataset that consists of approximately 1000 hours of audiobook recordings, mostly come from Project Gutenberg collection. In this paper we exclusively utilized the "clean" category of the dataset, which includes recordings from 20 male and 20 female speakers for both the "dev" and "test" subsets. The dictionary was automatically constructed using the same method applied to the HGP dataset, except that the comparison was performed at word-level instead of token-level.

[3] https://shorturl.at/fnKNO

4.2 Baselines

We compared the proposed method to strong baselines. The original **Whisper** (Radford et al., 2023) is the first baseline. It directly transcribes speech data to text without using our inject method. We used two versions: Whisper small (Whisper S) and Whisper medium (Whisper M) as strong baselines. We did not report the performance of the Whisper large model because of the tiny gap on testing datasets. **Initial prompt** uses the prompt as "はい、日本語で、*token1*、*token2*、..., *tokenN*の単語をすべて含むテキストを生成します。". ("Yes, it will generate a text containing all the words token1, token2, ..., tokenN in Japanese") for decoding. **BeamSearch + n-grams LM** (Williams et al., 2018) uses beam search decoding combined with a $n-$grams language model for decoding. **TCPGen** (Sun et al., 2021) incorporates a list of biasing words into both attention of encoder-decoder and transducer of ASR models. The probability of generated tokens is computed by using the probability distribution over a subset of output subword units conditioned by a prefix tre.

4.3 Evaluation Metrics

Character and word error rate (CER and WER) We used CER and WER (Rix et al., 2001; Hu and Loizou, 2006), well-known metrics for evaluating ASR models, as the main metrics for evaluation. The CER was used for Japanese datasets and WER was used for the English dataset.

Dictionary recognition rate The CER or WER metrics can measure the improvement in terms of corrected predicted tokens over the gold label. However, the number of corrected characters is small compared to the total number of characters in a testing set. As a result, a small improvement in CER or WER may not represent the efficiency of the proposed method. We, therefore, define a new metric called **Dictionary Recognition Rate (DRR)** as follows.

$$DRR = \frac{\text{\#words in dictionary correctly recognized}}{\text{\#words in dictionary should be recognized}}$$

4.4 Implementation

The α parameter was selected in the range of [0.1, 0.6] shown in Figure 3. The beam size was fixed at five. Whisper small and medium versions were used as the main backbone for transcription. We did not report the performance of the Whisper large version due to very tiny improvements.

The method requires a single Tesla P100 GPU for running the backbone ASR model.

5 Results and Discussion

Performance comparison Table 2 shows the comparison of the proposed method to strong baselines. We can observe that the combination of our decoding with the initial prompting method obtains the best results in almost all cases on two evaluation metrics. The improvement may come from two possible reasons. First, our decoding method forces Whisper to more focus on domain-specific tokens in the dictionary. By adjusting the probability of tokens generated by the decoder of Whisper, the decoding process can replace generated tokens with those in the dictionary, e.g., 流分 (does not exist in Japanese dictionary) and 留分 (part of distillation). Second, the initial prompt method also provides a good indicator for the decoding. It uses a prompt that includes a short instruction and a set of tokens in the jargon for guiding the decoding process. This prompt forces the decoder of Whisper to more focus on tokens in the prompt when doing speech recognition. By using both methods, the combination can achieve the best results.

An important point is that the performance of initial prompting and our decoding methods is better than Whisper, the strong baseline of ASR on the four testing datasets. As mentioned, the initial prompt method directly embeds the jargon in the prompt that serves as the prefix when decoding audio segments. However, this method has a main limitation. The prompt is short (less than 250 tokens) and faces a challenge with a long dictionary in practical cases. In contrast, our decoding method directly adjusts the decoding process by using the trie tree representation. Tokens appearing in the jargon receive higher probabilities than those that are not in the jargon. As a result, our decoding method is more general and can work with any ASR engines and dictionaries. In addition, the initial prompt method does not show the efficiency on Librispeech. This is because this corpus is public and already included in the pre-training of Whisper, therefore, adding additional context in form of a dictionary is not necessary. The inclusion of Librispeech in the pre-training of Whisper also results the high score of DRR on this dataset. Despite the limitation, in actual cases, the initial prompt method still can be used by using matching or selection methods for dealing with a large dictionary. For example, the selection method can select relevant tokens given an input speech for recognition to reduce the number of tokens in the prompt. In this case, the combination of our decoding and initial prompt methods can retain promising results. In the case that the initial prompt method is not available, our method can still output competitive scores. For example, it is the best on Librispeech and the second best of DRR on HGP. It confirms the efficiency of our proposed decoding method.

For the baselines, Whisper (small and medium versions) obtains promising results. As mentioned, by using a huge amount of data for pre-training, Whisper can work well on various domains in the multilingual setting. TCPGen does not show the efficiency on GPT synthesis and JNAS datasets because these datasets have only testing sets while TCPGen requires training data for adaptation. For Japanese, only HGP provides both training and testing sets. Therefore, we fine-tuned TCPGen on HPC and directly used the model for GPT Synthesis and JNAS testing sets. The Beam Seach+n-grams LM method achieves competitive results even it is a quite simple method. It shows the role of beam search and language modeling for the contextual basing task of ASR. However, the performance of this method is still behind our methods.

Among the four datasets, the improvement on the two in-house datasets is larger than JNAS and Librispeech, two public corpora. There are two possible reasons. First, the two in-house datasets contain more domain-specific knowledge and terms that may not appear in the training data of Whisper. This, therefore, challenges the transcription of Whisper in domain-specific data. Second, JNAS and Librispeech are benchmark datasets that are easy to collect and use as the training data of Whisper. It explains the reason why Whisper operates well on benchmark ASR corpora. The proposed method is simple and effective. It does not increase much the running time due to the efficient searching method on a small dictionary using a trie tree.

Human vs. automatic dictionary creation Dictionaries created by humans are good at biasing, however, it's creation is costly and requires domain knowledge. We, therefore, investigated the behavior of injection methods using dictionaries created

Table 2: Performance comparison on three datasets. **Bold** text is the best and underline text is the second best. S (small) and M (medium) stand for the two versions of Whisper. This notation is also used for Table 3.

Method	HGP		GPT Synthesis		JNAS		LibriSpeech	
	CER	DRR	CER	DRR	CER	DRR	WER	DRR
Beam Search+n-grams LM	14.98	78.76	15.61	16.95	9.92	28.19	2.92	98.85
TCPGen (Whisper S)	22.90	68.59	24.61	13.60	19.98	24.77	3.23	99.03
TCPGen (Whisper M)	19.02	78.89	15.55	16.80	15.17	29.15	2.92	98.30
Whisper (S)	23.50	66.42	23.08	13.22	17.27	23.52	3.67	98.19
Whisper (M)	14.97	78.82	15.60	16.95	9.92	28.35	2.82	**99.25**
Initial prompt (Whisper S)	23.18	66.60	20.67	51.86	19.34	32.40	3.48	98.45
Initial prompt (Whisper M)	12.05	81.09	13.09	59.32	9.22	39.56	3.34	98.77
Ours (Whisper S)	22.82	71.86	22.40	29.49	16.96	31.15	3.65	98.21
Combined (Whisper S)	22.40	71.91	20.40	79.66	20.24	41.31	3.49	98.45
Ours (Whisper M)	14.66	82.76	14.86	25.76	9.60	34.74	**2.81**	**99.25**
Combined (Whisper M)	**11.57**	**85.08**	**12.24**	**82.37**	**9.06**	**46.26**	3.34	98.77

by humans or automation. As mentioned in Section 3.3, the dictionaries used in the experiments were created by humans. To assess the setting of using automatic dictionary creation, we used a method as follows. First, the speech data of testing sets of the datasets were transcribed by using Whisper. The transcription was then aligned with the gold texts to obtain wrong words based on word segmentation (we used Mecab[4] for Japanese). Finally, common words were removed by using Japanese or English dictionaries. Note that we could not create the dictionary by humans for Librispeech due to the large number of testing samples. Therefore, we did not report the observation on Librispeech.

Table 3 reports the scores of the injection methods using dictionaries created by humans or automation. The general trend shows that dictionaries created by humans help to improve the performance of ASR. For example, methods using dictionaries created by humans output better performance than those using dictionaries created automatically in four cases. It is easy to understand that domain experts can create closer and more correct domain-specific words that the original Whisper models can not transcribe correctly. In this sense, incorporating these words into the decoding of Whisper can improve CER. In contrast, the automatic creation method may create wrong words due to the accumulated error of ASR. It then affects the final scores of the injection methods. An interesting observation is that the gap between the two setting is small. It suggest that we can reduce human effort in creating domain-specific dictionaries by applying appropriate automatic methods.

[4]https://github.com/elisa-aleman/MeCab-python

Table 3: The performance of the injection methods using automatic dictionary creation or human-involved dictionary creation. **IP** stands for Initial Prompt.

Method	HGP	GPT Syn	JNAS	Avg
Human creation				
IP (S)	23.18	20.67	19.34	21.06
IP (M)	12.05	13.09	9.22	**11.45**
Ours (S)	22.82	22.40	16.96	**20.72**
Combined (S)	22.40	20.40	20.24	21.01
Ours (M)	14.66	14.86	9.60	**13.04**
Combined (M)	11.57	12.24	9.06	**10.95**
Automatic creation				
IP (S)	21.20	21.13	19.86	**20.73**
IP (M)	16.01	14.28	15.99	15.42
Ours (S)	23.35	22.20	17.34	20.96
Combined (S)	21.33	20.34	20.22	**20.63**
Ours (M)	18.57	14.48	14.37	15.80
Combined (M)	12.60	13.08	14.79	13.49

Parameter fine-tuning Eqs. (4) and (7) use the parameter α to control the contribution of the jargon, so we investigated the behavior of the model with different α values. To do that, we tuned the parameter α in the range of [0.1, 0.2, 0.3, 0.4, 0.5, 0.6]. Figure 3 shows the performance with different α values using the Whisper medium version. We can observe that the CER slowly decreases until 0.4 and then quickly increases. The reason is that with a small value of α the contribution of our decoding is tiny. When increasing α, the decoding process is hallucinated by the jargon that leads to high CERs and WER. The change of performance on Librispeech is tiny compared to other Japanese datasets. As mentioned, the Librispeech corpus may be included in the pre-training of Whisper, that mitigates the contribution of our decoding method. Based on the obversation, we therefore selected $\alpha = 0.2$ for HGP, JNAS, and Librispeech corpora and $\alpha = 0.4$ the GPT-Syn dataset.

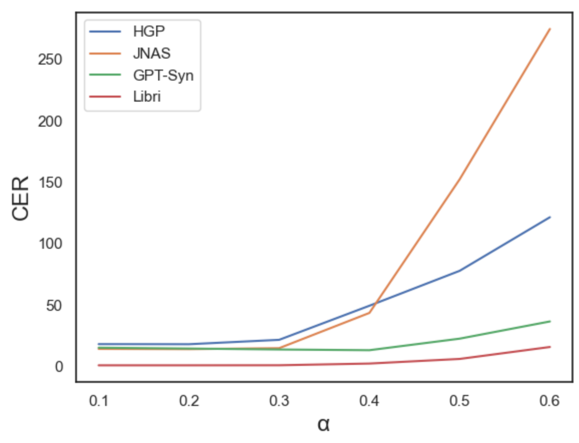

Figure 3: Parameter tuning using Whisper medium.

Output observation Table 4 shows the output observation of methods compared to the ground truth (translated as "It is presumed that the corrosion gap was made due to aging of refrigerant piping") on the HGP dataset using the Whisper medium version. 冷媒 means "refrigerant", and 腐食孔 means "corrosion gap". We did not show the output of the Beam Search+n-grams LM and TCPGen due to low CER and WER as in Table 2.

Table 4: The outputs from decoding methods. Blue text is correct and red text denotes incorrect tokens.

Method	Sample
Ground-truth	冷媒配管の経年劣化による腐食孔と推定
Whisper	霊媒配管の経年劣化による腐食効と推定
Initial prompt	例外配管の経年劣化による腐食孔と推定
Ours	冷媒配管の経年劣化による腐食効と推定
Combination	冷媒配管の経年劣化による腐食孔と推定

Whisper predicts two similar tokens marked by red. This is because these terms are in the high-pressure gas incident domain that does not appear in the training of Whisper. Instead, it tries to generate similar tokens used in the training process. The initial prompt can correctly recognize one token due to the use of the jargon in the prompt. It is similar to our decoding method. The combined method shows the best result that can correctly predict two tokens. It supports the results in Table 2 in which our methods output better scores than others.

6 Conclusion

This paper introduces a new method for improving the performance of ASR engines, i.e., Whisper by taking into account jargon. To do that, the method considers the jargon as the context injected into the decoding process. Domain-specific tokens receive more attention by adjusting the score of tokens in the beam search. Experimental results on three Japanese and one English datasets confirm two important points. First, the jargon can provide useful domain knowledge to improve the quality of Whisper. It shows that the improvement on domain-specific corpora is higher than public datasets due to the lack of domain knowledge of Whisper. Second, the combination of our decoding and the initial prompt methods achieves the best results. The proposed method provides a simple but effective way for domain adaptation of Whisper without accessing speech data for fine-tuning ASR models. Future work will confirm the efficiency of the method on other datasets and improve the decoding process using graph neural networks.

Limitations

Even achieving promising results, the proposed method has some limitations. First, it can fail to correct tokens even if they are included in the jargon. The possible reason comes from the fact that the beam that contains these tokens receives a low score. Even boosting the score of these tokens, the beam could not receive the highest probability. Second, the initial prompt method helps to improve the overall performance. In this case, if this method is not available (ASR models to not offer it) or the number of biasing words is larger than 244 tokens, it may challenge our proposed method.

Ethics Statement

All datasets and baseline models experimented with in this work have no unethical applications or risky broader impacts. The evaluation uses one public dataset that is widely used for ASR. For the HGP dataset, we followed the term that we could not publish the data. We really acknowledge the understanding of audiences for data publication. The dataset does not include any confidential or personal information of workers or companies. The baseline methods used for evaluation can be publicly accessed with GitHub links. There is no bias for the re-implementation or parameter selection that can affect the final results.

References

Uri Alon, Golan Pundak, and Tara N Sainath. 2019. Contextual speech recognition with difficult negative training examples. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6440–6444. IEEE.

Loïc Barrault, Yu-An Chung, Mariano Cora Meglioli, David Dale, Ning Dong, Paul-Ambroise Duquenne, Hady Elsahar, Hongyu Gong, Kevin Heffernan, John Hoffman, et al. 2023. Seamlessm4t-massively multilingual & multimodal machine translation. *arXiv preprint arXiv:2308.11596*.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4960–4964. IEEE.

Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. 2018. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4774–4778. IEEE.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. *Advances in neural information processing systems*, 28.

Linhao Dong, Shuang Xu, and Bo Xu. 2018. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5884–5888. IEEE.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376.

Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*, pages 1764–1772. PMLR.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. 2020. Conformer: Convolution-augmented transformer for speech recognition.

Minglun Han, Linhao Dong, Zhenlin Liang, Meng Cai, Shiyu Zhou, Zejun Ma, and Bo Xu. 2022. Improving end-to-end contextual speech recognition with fine-grained contextual knowledge selection. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8532–8536. IEEE.

Wei Han, Zhengdong Zhang, Yu Zhang, Jiahui Yu, Chung-Cheng Chiu, James Qin, Anmol Gulati, Ruoming Pang, and Yonghui Wu. 2020. Contextnet: Improving convolutional neural networks for automatic speech recognition with global context.

Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, et al. 2019. Streaming end-to-end speech recognition for mobile devices. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6381–6385. IEEE.

Yi Hu and Philipos C Loizou. 2006. Evaluation of objective measures for speech enhancement. In *Ninth international conference on spoken language processing*.

Wen-Chin Huang, Chia-Hua Wu, Shang-Bao Luo, Kuan-Yu Chen, Hsin-Min Wang, and Tomoki Toda. 2021. Speech recognition by simply fine-tuning bert. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7343–7347. IEEE.

Shumpei Inoue, Minh-Tien Nguyen, Hiroki Mizokuchi, Tuan-Anh Nguyen, Huu-Hiep Nguyen, and Dung Le. 2023. Towards safer operations: An expert-involved dataset of high-pressure gas incidents for preventing future failures. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 509–521.

Katunobu Itou, Mikio Yamamoto, Kazuya Takeda, Toshiyuki Takezawa, Tatsuo Matsuoka, Tetsunori Kobayashi, Kiyohiro Shikano, and Shuichi Itahashi. 1999. Jnas: Japanese speech corpus for large vocabulary continuous speech recognition research. *Journal of the Acoustical Society of Japan (E)*, 20(3):199–206.

Tahir Javed, Sumanth Doddapaneni, Abhigyan Raman, Kaushal Santosh Bhogale, Gowtham Ramesh, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh M Khapra. 2022. Towards building asr systems for the next billion users. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10813–10821.

Jeesu Jung, Hyein Seo, Sangkeun Jung, Riwoo Chung, Hwijung Ryu, and Du-Seong Chang. 2023. Interactive user interface for dialogue summarization. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, pages 934–957.

Namkyu Jung, Geonmin Kim, and Joon Son Chung. 2022. Spell my name: keyword boosted speech recognition. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6642–6646. IEEE.

Duc Le, Mahaveer Jain, Gil Keren, Suyoun Kim, Yangyang Shi, Jay Mahadeokar, Julian Chan, Yuan Shangguan, Christian Fuegen, Ozlem Kalinli, et al. 2021a. Contextualized streaming end-to-end speech recognition with trie-based deep biasing and shallow fusion. *arXiv preprint arXiv:2104.02194*.

Duc Le, Gil Keren, Julian Chan, Jay Mahadeokar, Christian Fuegen, and Michael L Seltzer. 2021b. Deep shallow fusion for rnn-t personalization. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 251–257. IEEE.

Daniel Li, Thomas Chen, Alec Zadikian, Albert Tung, and Lydia B Chilton. 2023. Improving automatic summarization for browsing longform spoken dialog. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–20.

Ian McGraw, Rohit Prabhavalkar, Raziel Alvarez, Montse Gonzalez Arenas, Kanishka Rao, David Rybach, Ouais Alsharif, Haşim Sak, Alexander Gruenstein, Françoise Beaufays, et al. 2016. Personalized speech recognition on mobile devices. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5955–5959. IEEE.

Douglas O'Shaughnessy. 2024. Trends and developments in automatic speech recognition research. *Computer Speech Language*, 83:101538.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE.

Gueorgui Pironkov, Sean UN Wood, and Stephane Dupont. 2020. Hybrid-task learning for robust automatic speech recognition. *Computer Speech & Language*, 64:101103.

Golan Pundak, Tara N Sainath, Rohit Prabhavalkar, Anjuli Kannan, and Ding Zhao. 2018. Deep context: end-to-end contextual speech recognition. In *2018 IEEE spoken language technology workshop (SLT)*, pages 418–425. IEEE.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR.

Virgile Rennard, Guokan Shang, Julie Hunter, and Michalis Vazirgiannis. 2023. Abstractive meeting summarization: A survey. *Transactions of the Association for Computational Linguistics*, 11:861–884.

Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra. 2001. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*, volume 2, pages 749–752. IEEE.

Yan Song, Yuanhe Tian, Nan Wang, and Fei Xia. 2020. Summarizing medical conversations via identifying important utterances. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 717–729.

Guangzhi Sun, Chao Zhang, and Philip C Woodland. 2021. Tree-constrained pointer generator for end-to-end contextual speech recognition. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 780–787. IEEE.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Ian Williams, Anjuli Kannan, Petar S Aleksic, David Rybach, and Tara N Sainath. 2018. Contextual speech recognition in end-to-end neural network systems using beam search. In *Interspeech*, pages 2227–2231.

Dian Yu, Kai Sun, Claire Cardie, and Dong Yu. 2020. Dialogue-based relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4927–4940.

Ding Zhao, Tara N Sainath, David Rybach, Pat Rondon, Deepti Bhatia, Bo Li, and Ruoming Pang. 2019. Shallow-fusion end-to-end contextual biasing. In *Interspeech*, pages 1418–1422.