Interventional Speech Noise Injection for ASR Generalizable Spoken Language Understanding

Yeonjoon Jung[♠,♡] **Jaeseong Lee**[♠] **Seungtaek Choi**[♣]
Dohyeon Lee[♠] **Minsoo Kim**[♠,♡] **Seung-won Hwang**[♠,♡,*]

[♠]Seoul National University [♣]Yanolja
[♡]Interdisciplinary Program in Artificial Intelligence, Seoul National University

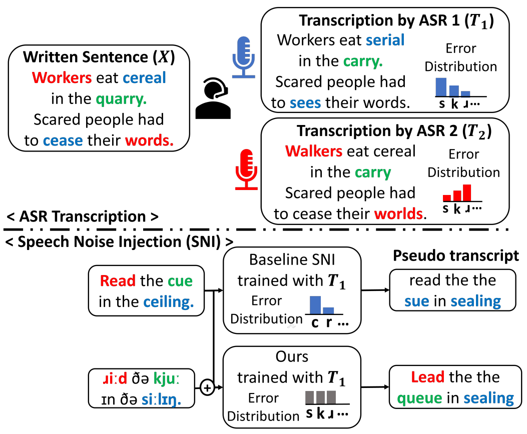
{y970120, tbvj5914, waylight, minsoo9574, seungwonh}@snu.ac.kr seungtaek.choi@yanolja.com

Abstract

Recently, pre-trained language models (PLMs) have been increasingly adopted in spoken language understanding (SLU). However, automatic speech recognition (ASR) systems frequently produce inaccurate transcriptions, leading to noisy inputs for SLU models, which can significantly degrade their performance. To address this, our objective is to train SLU models to withstand ASR errors by exposing them to noises commonly observed in ASR systems, referred to as ASR-plausible noises. Speech noise injection (SNI) methods have pursued this objective by introducing ASR-plausible noises, but we argue that these methods are inherently biased towards specific ASR systems, or ASR-specific noises. In this work, we propose a novel and less biased augmentation method of introducing the noises that are plausible to any ASR system, by cutting off the non-causal effect of noises. Experimental results and analyses demonstrate the effectiveness of our proposed methods in enhancing the robustness and generalizability of SLU models against unseen ASR systems by introducing more diverse and plausible ASR noises in advance.

1 Introduction

Pre-trained language models (PLMs) have demonstrated a robust contextual understanding of language and the ability to generalize across different domains. Thus, PLMs have gained widespread acceptance and been employed within the realm of spoken language understanding (SLU), such as voice assistants (Broscheit et al., 2022; Zhang et al., 2019). A concrete instance of this application is a pipeline where an automatic speech recognition (ASR) system first transcribes audio inputs, and the transcriptions are then processed by PLMs for downstream SLU tasks (Feng et al., 2022).



Written Sentence (X)
Workers eat cereal in the quarry. Scared people had to cease their words.
Transcription by ASR 1 (T_1)
Workers eat serial in the carry. Scared people had to sees their words.
Error Distribution
s k ɹ ...
Transcription by ASR 2 (T_2)
Walkers eat cereal in the carry Scared people had to cease their worlds.
Error Distribution
s k ɹ ...
< ASR Transcription >
< Speech Noise Injection (SNI) >
Read the cue in the ceiling.
Baseline SNI trained with T_1
Error Distribution
c r ...
Pseudo transcript
read the the sue in sealing
ɹi:d ðə kju: ɪn ðə si:lɪŋ.
Ours trained with T_1
Error Distribution
s k ɹ ...
Lead the the queue in sealing

Figure 1: Different ASR systems generate different ASR errors (ASR$_1$: blue, ASR$_2$: red, common, or, ASR$_*$: green). Biased toward a specific ASR, ASR$_1$, baseline SNI generates noises plausible only for ASR$_1$, or even some noise that are not plausible to any (cue to sue). Our distinction is 1) removing its bias to a specific ASR (Read to Lead), and 2) generating ASR$_*$-plausible noises (cue to queue).

However, such pipelines encounter challenges when faced with inaccuracies from ASR systems. We refer to these inaccuracies as ASR error words, which are phonetically similar but semantically unrelated (Ruan et al., 2020; Huang and Chen, 2020). For instance, as shown in Fig. 1, ASR systems might confuse words like "cereal" and "serial" or "quarry" and "carry", resulting in incorrect transcriptions such as "workers eat serial in the carry". Despite their phonetic resemblance, these ASR error words convey unintended meanings, thereby impeding the semantic understanding of speech in SLU tasks (Belinkov and Bisk, 2018).

A well-known solution is speech noise injection (SNI) which generates likely incorrect transcriptions, namely pseudo transcriptions, then exposes PLMs to the generated pseudo transcriptions while training SLU tasks (Heigold et al., 2018; Di Gangi et al., 2019). Thus, the injected noises should be ASR-plausible, being likely to be gen-

[*]Corresponding author.

erated by ASR systems from real audio input, for which the traditional methods attempted to replicate the ASR errors from written text. However, their ASR-plausibility is conditional on a particular ASR system, or ASR$_i$, where the error distribution from the collected transcriptions follows only the observed error distribution (Wang et al., 2020; Gopalakrishnan et al., 2020; Cui et al., 2021).

However, different ASR systems have distinct error distributions (Tam et al., 2022), which hinders the trained SLU models to be used with other ASR systems, ASR$_j$. A straightforward solution to this issue might be to employ multiple ASR systems or to build multiple SLU models, but this approach incurs significant overheads and cannot account for the diversity of real-world ASR errors, *e.g.*, errors due to environmental sounds. Instead, we investigate a novel but easier solution of introducing better generalizable noises from a single ASR system.

For this purpose, we first identify the gap between the ASR transcription in SLU and SNI from a causality perspective. First, SLU tasks aim to handle real audio input, where written ground-truths(GTs) are recorded as audio by humans, such that ASR errors are causally affected by recorded audio, as shown in Figure 2a. However, SNI, when replicating error patterns, may discard the correctly transcripted texts, biased towards the observed errors. Inspired by causality literature, we introduce two technical contributions: 1) interventional noise injection and 2) phoneme-aware generation. Specifically, we adopt *do*-calculus to intervene the “revised” in SNI to deviate from the observed error distribution, thereby broadening the error patterns in the resulting pseudo transcripts. For instance, as shown in Fig. 1, our SNI model can corrupt the word ‘Read’ with the phoneme ‘ɹ’, which was corrupted in ASR$_2$ but not in ASR$_1$, in addition to the errors introduced by baseline SNI.

Next, we ensure that the debiased noises are plausible for any ASR system, referred to as ASR$_*$. This involves making GT words and ASR noises phonetically similar based on the common characteristics shared by ASR$_*$ (Serai et al., 2022). Along with the textual input, we incorporate information on how words are pronounced. By being aware of pronunciation, we can introduce ASR noises that are plausible regardless of the specific ASR system used, making them ASR$_*$-plausible.

Experiments were conducted in an ASR zero-shot setting, where SNI models were trained on ASR$_i$ and tested on SLU tasks using another ASR system, ASR$_j$, on the DSTC10 Track2 and ASR GLUE benchmarks. Results show that our proposed methods effectively generalize across different ASR systems, with performance comparable to, or even exceeding, the in-domain setting where ASR$_j$ is used to train the SNI model.

2 Related Work

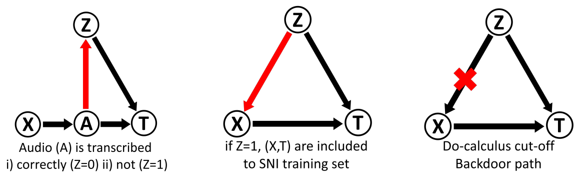
2.1 SNI

Previously proposed methods can be broadly categorized into three main approaches. The first, a Text-to-Speech (TTS)-ASR pipeline (Liu et al., 2021; Chen et al., 2017), uses a TTS engine to convert text into audio, which is then transcribed by an ASR system into pseudo transcriptions. However, this method struggles due to different error distributions between human and TTS-generated audio, making the pseudo transcriptions less representative of actual ASR errors. The second approach, textual perturbation, involves replacing words in text with noise words using a scoring function that estimates how likely an ASR system is to misrecognize the words, often employing confusion matrices (Jyothi and Fosler-Lussier, 2010; Yu et al., 2016) or phonetic similarity functions (Li and Specia, 2019; Tsvetkov et al., 2014). The third method, auto-regressive generation, utilizes PLMs like GPT-2 or BART to generate text that mimics the likelihood of ASR errors in a contextually aware manner, producing more plausible ASR-like noise (Cui et al., 2021).

We consider auto-regressive noise generation as our main baseline as it has shown superior performance over other categories (Feng et al., 2022; Kim et al., 2021) However, auto-regressive noise generation is biased to ASR$_i$, limiting the SLU model used for ASR$_i$. Our distinction is generalizing SNI so that the SLU tasks can be conducted with ASR$_*$. We provide a more detailed explanation of each category in Appendix 7.1.

2.2 ASR Correction

As an alternative to SNI, ASR correction aims to denoise (possibly noisy) ASR transcription T_i into X: Due to its similarity to SNI, similar methods, such as textual perturbation (Leng et al., 2021) and auto-regressive generation (Dutta et al., 2022; Chen et al., 2024), were used. Also, PLMs showed impressive results (Dutta et al., 2022), as the ASR noise words can be easily detected by PLMs due to their semantic irrelevance. Using such characteristics, the constrained decoding method which first detects ASR errors, then corrects detected ASR errors is proposed (Yang et al., 2022). However, the innate robustness of PLMs (Heigold et al., 2018) makes SNI outperforms ASR correction (Feng et al., 2022). In addition, it introduces additional latency in the SLU pipeline for ASR correction before the SLU model to conduct SLU tasks. In voice assistant systems like Siri, such additional latency is crucial as the SLU pipeline should deliver responses with minimal delay to ensure real-time interaction. Therefore, we focus on SNI for its effectiveness and minimal latency in SLU pipeline.



(a) Causal graph of ASR transcription. X causally influences to Z through directed path.

(b) Causal graph of SNI training data collection. X and Z are non-causally related through backdoor path.

(c) Causal graph of ISNI adopting *do*-calculus. Path between Z and X are cut-off

Figure 2: The causal graph between ASR transcription (a), SNI training data generation (b), and ISNI (c)

3 Method

Before introducing ISNI, we first formally define the problem of SNI and outline its causal diagram. Following this, we provide an overview of ISNI and detail the methods used to generate pseudo transcriptions that enhance the robustness of SLU tasks against ASR$_*$.

3.1 Problem Formulation

To robustify PLMs against ASR errors in SLU, the SNI model mimics the transcriptions of ASR$_i$ which can be defined as a functional mapping $F_i : X \rightarrow T_i$. Given the written GT $X = \{x^k\}_{k=1}^n$ with n words, the SNI model F_i outputs the pseudo transcription $T_i = \{t_i^k\}_{k=1}^n$, simulating the transcriptions produced by ASR$_i$. These pseudo transcriptions, T_i, are subsequently used to train SLU models. For each word x^k, $z^k \in Z$ indicates whether ASR$_i$ makes errors ($z^k = 1$, hence $x^k \neq t_i^k$) or not ($z^k = 0$, hence $x^k = t_i^k$). However, noises generated by this model differ from those of the other ASR system, ASR$_j$, and SLU model trained with the generated noises struggles with the errors from ASR$_j$. Therefore, we propose building an SNI model, F_*, capable of generating "ASR$_*$-plausible" pseudo transcripts T_*, which are plausible for any ASR system, ASR$_*$.

3.2 Causality in SNI

To achieve this, we compare the underlying causal relations between the transcription process in SLU and SNI training data generation, which are depicted in Fig. 2a. During ASR transcription, written GT X is first spoken as audio A, which is then transcribed as transcription T by ASR$_i$. Depending on the audio A, ASR$_i$ may make errors ($Z = 1$) or not ($Z = 0$) in the transcription T. While transcribing, X influences Z through causal paths where every edge in the path is directed toward Z, where Z acts as a mediator between X and Z.

In SNI, the X and T transcribed by ASR$_i$ are filtered based on Z since they do not exhibit the error patterns required for training. However, such filtering induces a backdoor path and a non-causal relation in SNI. To further elucidate the causal relations in SNI training data generation depicted in Fig. 2b, we outline the causal influences as follows:

- $X \rightarrow T$: There is a direct causal relationship where the clean text X influences the transcribed text T.
- $Z \rightarrow T$: If $z^k \in Z$ is 1 (an error occurs), it directly affects the corresponding transcription t_i^k, causing it to deviate from the clean text x^k.
- $Z \rightarrow X$: In the SNI training data collection process, Z determines if X is included. This means that only when the ASR system makes a mistake, indicated by any value $z^k \in Z$ being 1, the corresponding text is included in the training data. So, errors by the ASR system decide which clean texts are chosen.

The backdoor path $X \leftarrow Z \rightarrow T$, while ensuring that only instances where ASR$_i$ made errors are included, introduces bias in previous SNI models based on conventional likelihood, defined as:

$$P(t_i^k|x^k) = \sum_{z^k} P(t_i^k|x^k, z^k)P(z^k|x^k). \qquad (1)$$

In contrast to ASR transcription where z^k is a consequence of x^k and the mediator between x^k and t^k, z^k conversely influences x^k and acts as a confounder. Thus, the influence of x^k to t^k is drawn

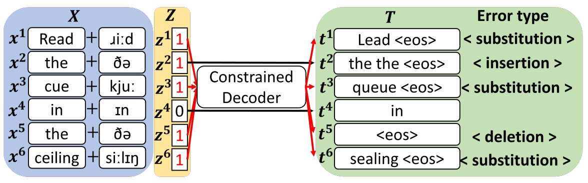


Figure 3: Overview of ISNI. ISNI generates ASR noise word t^k for the clean text x^k whose corresponding z^k is 1. The error type of t^k is determined by the generated output.

from z^k, not from x^k, thereby distorting the causal effect from X to T. Such distortion skews the prior probability $P(z^k|x^k)$ so that texts frequently incorrectly transcribed by ASR$_i$ are also noised by SNI.

3.3 *do*-calculus for ISNI

To mitigate biases in SNI toward the ASR$_i$, we propose interventional SNI (ISNI) where the non-causal relations are cut off by adopting *do*-calculus. Adopting *do*-calculus for conventional likelihood in Eq. 1, ISNI can be formulated as follows:

$$P(t^k|do(x^k)) = \sum_{z^k} P(t^k|x^k, z^k) \cdot P(z^k). \quad (2)$$

Compared to Eq.1, the prior probability $P(z^k|x^k)$ is replaced with $P(z^k)$. This difference implies that the non-causal path $Z \rightarrow X$ is cut off, as the influence of x^k on t^k is not drawn from z^k in Eq. 2 as prior probability $P(z^k)$ is estimated independently from the non-causal path $Z \rightarrow X$. Thus, the influence of x^k to t^k is induced solely from x^k. We provide more detailed proof of Eq. 2 in Appendix 7.2.

3.4 Overview of Our Proposed Approach

In this section, we briefly overview how ISNI is implemented to generate ASR$_*$-plausible noises of different error types, as illustrated in Fig. 3. To achieve this, ISNI implements two distinct terms in Eq. 2: $P(z^k)$ and $P(t^k|x^k, z^k)$.

For debiased noise generation $P(z^k)$, it is essential to cut off the influence of x^k on z^k so that z^k becomes independent of x^k. ISNI explicitly controls z^k and removes the dependence on x^k by implementing SNI with constrained decoding, following Yang et al. (2022). Specifically, by utilizing *do*-calculus? to determine z, ISNI introduces corruption in tokens that ASR$_i$ transcribes correctly but ASR$_j$ may mistranscribe, thereby adding these as noise in the pseudo transcripts. For example, the token x with $z = 1$, such as 'cue' in Fig. 3, is fed to the constrained decoder. The constrained decoder then generates the noise word t to replace x.

The noise word t can be categorized as three conventional ASR error types: substitution, insertion, and deletion (Jurafsky and Martin, 2019), which is determined by the generation output of ISNI. Substitution errors occur when the generated output is a new token that replaces x. Insertion errors occur when the constrained decoder outputs multiple tokens for x. For example, in Fig. 3, the constrained decoder outputs two 'the' words for x^2, which means the insertion of the newly generated 'the' beside the given word 'the'. Deletion errors occur when the constrained decoder outputs only the *eos* token, representing the end of generation. If the generation ends without any tokens, as in t^5, the empty string replaces x^5. ISNI generates these error types synthetically, as demonstrated in the quantitative study in Appendix 7.6 and the generated examples in Appendix 7.7.

For ASR$_*$-plausible generation $P(t^k|x^k, z)$, we provide the phonetic information of x, such as 'kju' in Fig. 3, to ISNI to understand how x is pronounced. The constrained decoder evaluates the generation probability based on phonetic similarity, recognizing that 'c' is pronounced as the phoneme 'k'. Consequently, ISNI generates the ASR$*$-plausible noise word 'queue' which is phonetically similar to 'cue'.

3.5 Interventional SNI for Debiasing

To mitigate bias from dependence on ASR$_i$, we propose an intervention on the non-causal effect $X \leftarrow Z$. This approach diversifies the noise words in the pseudo transcripts, including those rarely corrupted by ASR$_i$. By applying *do*-calculus as shown in Eq. 2, we derive a following equation for SNI:

$$P(t^k|do(x^k)) = \sum_{z} P(t^k|x^k, z) \cdot P(z). \quad (3)$$

This intervention ensures that the corruption of words x^k in the pseudo transcripts does not depend on any particular ASR system. Consequently, it allows for the corruption of words that are typically less susceptible to errors under ASR$_i$.

For each word x^k, the prior probability $P(z)$ in Eq. 3 represents the probability that any given word x^k will be corrupted. To simulate $P(z)$, we assume that the average error rates of any ASR system, ASR$_*$ would be equal for all words and sample a random variable a^k for each word x^k from a continuous uniform distribution over the interval $[0,1]$. If $a^k \leq P(z)$, we set z to 1, indicating an incorrect transcription of x^k and generating its noise word t^k. Otherwise, z is set to 0, and t^k remains identical to x^k, mimicking an accurate transcription. We set the prior probability $P(z)$ as a constant hyperparameter for pseudo transcript generation.

To generate a noise word t^k whose corruption variable z^k is determined independently of any biases, we adopt a constrained generation technique that outputs t^k to input x^k (Yang et al., 2022). To implement constrained generation, we use BERT (Devlin et al., 2019) for encoding the written GT X into the vector representation E as follows:

$$E_{encoder} = \mathit{BERT}(M_{word}(X) + M_{pos}(X)), \quad (4)$$

where $e^k_{encoder} \in E_{encoder}$ is the encoder representation of the token $x^k \in X$ and M_{word} and M_{pos} denote word and position embeddings.

The transformer decoder (Vaswani et al., 2017) generates t^k constrained on x^k. To encompass all ASR error types, the transformer decoder generates multiple tokens (see Section 3.4), denoted as m tokens, $\{\tilde{t}^k_l\}^m_{l=1}$. These tokens replace x^k in the pseudo transcripts. The error type of t^k is contingent on m:

- $m = 1$ corresponds to deletion, generating only eos is generated; x^k is substituted by the empty string.
- If $m = 2$, one token in addition to eos is generated; this token substitutes x^k.
- When $m > 2$, additional token will be inserted to x^k, to simulate insertion errors.

To generate $\tilde{t}^k_l$, the input of the decoder comprises the encoder representation of the written word $e^k_{encoder}$ and the special embedding vector bos (beginning of sequence) and the tokens generated so far, $\{\tilde{t}^k_0, ..., \tilde{t}^k_{l-1}\}$ as follows:

$$E^k_{decoder} = H_{decoder} \cdot [e^k_{encoder}; bos, \tilde{t}^k_0, ..., \tilde{t}^k_{l-1}], \quad (5)$$

where $H_{decoder}$ is the weight matrix of the hidden layer in the decoder. The transformer decoder then computes the hidden representation d^k by processing the input through its layers as follows:

$$d^k = TransformerDecoder(Q, K, V), \quad (6)$$

$$Q = E^k_{decoder}, K, V = E_{encoder}, \quad (7)$$

where $E^k_{decoder}$ serves as query Q, and $E_{encoder}$ is used for both key K and value V in the transformer decoder. Finally, the probability of generating each token $\tilde{t}^k_l$ is calculated using a softmax function applied to the product of the word embedding matrix M_{word} and the hidden representation d^k added to the trained bias b_n:

$$P_n(\tilde{t}^k_l | x^k, \tilde{t}^k_0, ..., \tilde{t}^k_{l-1}) = softmax(M_{word} \cdot d^k + b_n), \quad (8)$$

where b_n are trained parameters. Here, we note that ISNI generates different error types depending on the output as we explained in Sec. 3.4.

We used the ASR transcription T to train ISNI model. ISNI is supervised to maximize the log-likelihood of the l-th token $\hat{t}^k_l$ from the ASR transcription as follows:

$$\mathcal{L}_n = -\sum log(P_n(\hat{t}^k_l | x^k, \hat{t}^k_0, ..., \hat{t}^k_{l-1})). \quad (9)$$

3.6 Phoneme-aware Generation for Generalizability

The next step is to ensure the noise word generation $P(t^k|x^k, z)$ ASR$_*$-plausible. We adjust the generation probability P_n with the phoneme-based generation probability P_{ph} so that the ASR$_*$-plausible noise words can be generated as followed:

$$P_{gen}(\tilde{t}^k_l | x^k, \tilde{t}^k_0, ..., \tilde{t}^k_{l-1}) = P_n(\tilde{t}^k_l | x^k, \tilde{t}^k_0, ..., \tilde{t}^k_{l-1}) \cdot P_{ph}(\tilde{t}^k_l | x^k, \tilde{t}^k_0, ..., \tilde{t}^k_{l-1}). \quad (10)$$

Our first proposal for P_{ph} is to provide the phonetic characteristics of each token via phoneme embedding M_{ph}. M_{ph} assigns identical embeddings to tokens sharing phonetic codes, thereby delivering how each word is pronounced. Phoneme embedding M_{ph} is incorporated into the input alongside word and position embeddings as follows:

$$E_{encoder} = BERT(\lambda_w \cdot M_{word}(X) + (1 - \lambda_w) \cdot M_{ph}(X) + M_{pos}(X)), \quad (11)$$

where λ_w is a hyperparameter that balances the influence of word embeddings and phoneme embeddings. The input for the decoder is formulated similarly to Eq. 5.

Fed both the word and phoneme embedding, the decoder then can understand the phonetic information of both the encoder and decoder input. Aggregating such information, the decoder would yield the hidden representation d^k as in Eq. 7. Then, we feed the hidden representation d^k to classification head to evaluate the phoneme based generation probability P_{ph} as follows:

$$P_{ph}(\tilde{t}_l^k|x^k, \tilde{t}_0^k, ..., \tilde{t}_{l-1}^k) = softmax(M_{ph} \cdot d^k + b_{ph}), \tag{12}$$

where classification head has the phoneme embedding matrix same as in Eq. 11 and bias b_{ph}. Using the phoneme embedding M_{ph} instead of the word embedding M_{word} in Eq. 8, $P_{ph}(\hat{t}^k|x^k)$ can be evaluated based on the phonetic information.

Our second proposal for P_{ph} is phonetic similarity loss $\mathcal{L}_{ph}$ supervising the phonetic information using phonetic similarity. This approach aims to generate ASR$_*$-plausible noise words by assessing the phonetic resemblance of each token to x^k. P_{ph} should evaluate how phonetically similar each token is to the x^k. Phonetic similarity is advantageous as it quantifies the differences in phonetic codes, thereby allowing for objective supervision of $P_{ph}(\hat{t}^k|x^k)$. We utilize the phoneme edit distance D, as outlined in Ahmed et al. (2022), to calculate phonetic similarity. The phoneme edit distance $D(w^p, w^q)$ measures the minimum edit distance between the phonetic codes of two words, reflecting how closely one word is pronounced to another. Notably, $D(w^p, w^q)$ leverages articulatory features to compute similarity, which incrementally increases as the phonetic resemblance between the word pair enhances.

Phonetic similarity, S, is defined as follows:

$$S(w^p, w^q) = max(|C_p| - D(w^p, w^q), 0), \tag{13}$$

where $|C_p|$ is the length of the phonetic code of the word w^p. This formulation ensures that $S(w^p, w^q)$ attains higher values when w^p and w^q are phonetically similar, and approaches zero when there is no phonetic resemblance.

To supervise P_{ph}, phonetic similarity should be formulated as a probability distribution. For such purpose, we normalize phonetic similarity and compute the supervision $R(\hat{t}^k)$ as follows:

$$R(\hat{t}^k) = \frac{S(t^k, w)}{\sum_{w' \in W} S(t^k, w')}. \tag{14}$$

Then, P_{ph} is supervised by loss defined as follows:

$$\mathcal{L}_{ph} = KL(P_{ph}(\tilde{t}_l^k|x^k, \tilde{t}_0^k, ..., \tilde{t}_{l-1}^k)|R(\hat{t}^k)), \tag{15}$$

where KL is the KL divergence loss. Finally, ISNI is optimized to jointly minimize the total loss $\mathcal{L}_{tot}$ which is defined as follows:

$$\mathcal{L}_{tot} = \mathcal{L}_n + \lambda_{ph} \cdot \mathcal{L}_{ph}, \tag{16}$$

where λ_{ph} is the weight of $\mathcal{L}_{ph}$. Supervised to evaluate the generation probability based on the phoneme information, the phoneme generation probability P_{ph} ensures the ASR$_*$-plausible noise words and M_{ph} contains phonetic information.

4 Experiments

In this section, we delineate our experimental setup, datasets, and baseline, for SNI and SLU. Our experiments, conducted in an ASR zero-shot setting, train SNI models on one ASR system (ASR$_i$) and test SLU models on another (ASR$_j$), to evaluate generalizability across different ASR systems.

4.1 Dataset

SNI Training For SNI training, we selected datasets consistent with our primary baseline, Noisy-Gen (Cui et al., 2021), utilizing three popular corpora: Common Voice, Tatoeba audio, and LJSpeech-1.1, with the MSLT corpus serving as the validation set. Audio recordings from these sources were processed using DeepSpeech to collect transcriptions that exhibit ASR errors. This approach yielded approximately 930,000 pairs of ground-truth and ASR-noised transcriptions.

SLU Training To demonstrate ASR generalizability across diverse SLU tasks, we utilize two benchmarks: ASR GLUE and DSTC10 Track 2. ASR GLUE, an adaptation of the widely recognized NLU benchmark GLUE, includes two natural language inference tasks, QNLI and RTE, and one sentiment classification task, SST2. To simulate real-world background noises, ASR GLUE benchmark randomly injected background noise audios into speech. Noises are injected in 4 levels, high, medium, low, and clean. Randomly injected noises introduce ASR errors across a broader range of phonemes. DSTC10 Track 2, tailored for spoken language dialogue systems, comprises three subtasks: Knowledge-seeking Turn Detection (KTD), Knowledge Selection (KS), and Knowledge-grounded Response Generation (RG). Details of the datasets for SNI training and SLU testing are available in Appendix 7.4.

4.2 Baselines

In our study, we utilized diverse SNI models, including the GPT2-based auto-regressive SNI model, NoisyGen, as our primary baseline, given its proven efficacy in various spoken language tasks (Cui et al., 2021; Feng et al., 2022). For the

ASR GLUE benchmark, we also included Noisy-Gen (In Domain) (Feng et al., 2022), another GPT2-based SNI model that, unlike our standard approach, uses the same ASR system for both training and testing, thereby not adhering to the ASR zero-shot setting. Also, PLMs trained only with written GT are used to show the necessity of exposure to ASR noises. Demonstrating that SNI models can match or surpass NoisyGen (In Domain) will confirm their ASR generalizability.

Additionally, for the DSTC10 Track 2, we incorporated the state-of-the-art TOD-DA (Tian et al., 2021) as a baseline, selected for its inclusion of both TTS-ASR pipeline and textual perturbation techniques, which are absent in NoisyGen. We selected TOD-DA because it covers two distinct categories of SNI which were not covered by Noisy-Gen: TTS-ASR pipeline and textual perturbation.

4.3 ASR system for SNI training and SLU testing

We provide the ASR systems used for SNI training and SLU testing in Table 1.

SNI Training We chose the open-source Mozilla DeepSpeech ASR system (Hannun et al., 2014) primarily because it aligns with the use of commercial ASR systems in previous SLU studies, including our main baseline, NoisyGen[1]. We specifically selected DeepSpeech because, as an open-source system, it provides transparency and flexibility while demonstrating a word error rate comparable to other closed commercial ASR systems. Moreover, our decision to use DeepSpeech reflects a practical scenario where SNI models trained on one ASR system need to demonstrate robustness and adaptability when applied to newer or different ASR systems, such as LF-MMI TDNN for Noisy-Gen (In Domain) (Feng et al., 2022) and Wave2Vec for TOD-DA (Yuan et al., 2017).

ASR	DeepSpeech	LF-MMI TDNN	Wave2vec	Unknown
SNI	ISNI, NoisyGen	NoisyGen (In Domain)	TOD-DA	-
SLU	-	ASR GLUE	-	DSTC10 Track2

Table 1: ASR systems for SNI training and SLU testing.

SLU Testing To evaluate the generalizability of SNI across various ASR systems, we employed distinct ASR systems for SLU testing in the ASR GLUE and DSTC10 Track 2 benchmarks. As detailed in Table 1, the ASR GLUE test set was transcribed using an LF-MMI TDNN-based ASR system, while an unknown ASR system was used for the DSTC10 Track 2 validation and test sets.

4.4 Experimental Settings

SNI Training For ISNI implementation, we utilized BERT-base (Devlin et al., 2019) as an encoder and a single Transformer decoder layer (Vaswani et al., 2017), aligning with established methodologies (Yang et al., 2022). The balance between word and phoneme embeddings was set with λ_w at 0.5 in Eq. 11, and the phoneme generation loss weight λ_{ph} was also adjusted to 0.5 (Eq. 16). We provide further details in Appendix 7.5.

SLU Training Utilizing the trained ISNI, we convert the written GTs into pseudo transcripts. During the generation of pseudo transcripts, we set the prior probability $P(z)$ for ASR GLUE at 0.15 and for DSTC10 Track2, at 0.21, based on validation set result of downstream SLU tasks[2]. To ensure a fair comparison, we set phonetic similarity thresholds in our baseline, NoisyGen, for filtering out dissimilar pseudo transcripts, based on the validation set result of downstream SLU tasks. In terms of downstream task models, we implemented BERT-base for ASR GLUE and GPT2-small for DSTC10 Track 2, consistent with baseline configurations (Kim et al., 2021; Feng et al., 2022).

5 Results

We now present our experimental results, addressing the following research questions:
RQ1: Is the ASR zero-shot setting valid and how effective are ISNI in the ASR zero-shot setting?
RQ2: Can ISNI robustify the various SLU tasks in the ASR zero-shot setting?
RQ3: Does each of methods contribute to robustification?

5.1 RQ1: Validity of ASR Zero-shot Setting and Effectiveness of ISNI.

To demonstrate the ASR generalizability of SNI models, we compare them with NoisyGen (In Domain) on ASR GLUE in Table 2. Unlike Noisy-Gen and our models tested in an ASR zero-shot setting,

[1]For a detailed comparison of commercial ASR systems, see Appendix 7.3.

[2]Directly matching $P(z)$ to the word error rate (WER) of the ASR system is not feasible in an ASR zero-shot setting where WER for unknown ASR systems is not available. Additionally, training with arbitrary word error rates might bias the SLU model towards certain ASR systems.

Task	QNLI				RTE				SST2			
Noise Level	High	Medium	Low	Clean	High	Medium	Low	Clean	High	Medium	Low	Clean
Written GT	70.22	73.00	81.45	90.00	45.84	48.82	60.71	78.57	79.11	80.19	81.41	93.51
NoisyGen	71.00	73.34	79.73	**86.00**	46.43	50.00	58.33	**60.71**	80.85	87.34	81.84	**92.86**
ISNI (Ours)	**76.89**	**77.89**	**82.56**	**86.00**	**56.55**	**58.93**	**60.12**	**60.71**	**82.36**	**88.12**	**85.17**	91.56
Noisy-Gen (In Domain)	74.00	77.39	83.44	88.67	53.57	58.93	59.52	60.71	83.98	88.75	84.20	92.86

Table 2: Accuracy on QNLI and RTE, SST2 of ASR GLUE benchmark.

Task	KTD			KS			RG							
Metric	P	R	F1	MRR@5	R@1	R@5	B@1	B@2	B@3	B@4	M	RG@1	RG@2	RGL
TOD-DA	88.58	89.75	89.16	67.29	60.51	76.75	8.31	4.74	2.33	0.92	13.02	16.45	6.49	15.13
NoisyGen	**89.42**	90.34	89.88	65.80	57.10	76.62	9.88	5.42	2.47	0.97	13.62	17.19	6.42	15.77
ISNI (Ours)	88.19	**92.97**	**90.52**	**72.61**	**66.43**	**81.11**	**15.32**	**9.79**	**5.10**	**2.36**	**20.18**	**24.42**	**10.65**	**22.64**
- phoneme-aware generation	88.29	90.48	89.37	71.12	64.49	79.68	14.61	9.22	4.59	2.36	18.84	22.55	9.66	21.33
-intervention	88.62	91.22	89.90	69.82	63.2	78.79	13.77	8.66	4.17	1.99	18.66	22.23	9.36	20.89

Table 3: Results of DSTC10 Track2. Precision (P) and Recall (R), F1 is used to evaluate KTD. Mean Reciprocal Rank at 5 (MRR@5) and Recall at 1, 5 (R@1,5) are used to evaluate KS. To evaluate RG, Bleu at 1,2,3,4 (B@1,2,3,4) and Meteor (M) and Rouge at 1,2,L (RG@1,2,L) are used.

NoisyGen (In Domain) was trained and tested using the identical ASR system, which is incompatible with the ASR zero-shot setting.

Our findings indicate that auto-regressive SNI lacks generalizability for diverse ASR systems. If different ASR systems have similar error distributions, existing auto-regressive generation SNIs would generalize in an ASR zero-shot setting. However, NoisyGen is consistently outperformed by NoisyGen (In Domain) in every task. This result validates the ASR zero-shot setting where existing auto-regressive generation-based SNI models struggle to generalize in the other ASR systems.

Results of ISNI suggest that ISNI can robustify SLU model in the ASR zero-shot setting. ISNI surpassed NoisyGen in every task in every noise level even NoisyGen (In Domain) in high and medium noise levels for QNLI and in low noise levels for SST2. Such results might be attributed to the diversified ASR errors in the ASR GLUE benchmark, which ISNI is specifically designed to target.

ISNI demonstrated robust performance across varying noise levels compared to baselines, maintaining its efficacy as noise levels escalated. This result highlights that ISNI can better robustify against phoneme confusions, which increase under noisy conditions (Wu et al., 2022; Petkov et al., 2013).

5.2 RQ2: Robustification of ISNI on Various SLU Tasks.

We demonstrate that ISNI significantly enhances robustness across various SLU tasks in an ASR zero-shot setting, particularly in KS, where lexical perturbation heavily influences retrieval (Penha et al., 2022; Chen et al., 2022).

Results from the DSTC10 Track 2 dataset, in Table 3, reveal that while baseline models struggle in the ASR zero-shot setting, showing R@1 score below 60, ISNI consistently outperforms these models across all metrics. This superior performance, especially notable in KS, validates ISNI's effectiveness against errors from unknown ASR systems, unlike previous models that require identical ASR systems for both training and testing (Feng et al., 2022; Cui et al., 2021).

Additionally, the robustification of ISNI extends beyond KS. ISNI excels across all evaluated tasks, markedly improving the BLEU score by 2-3 times for Response Generation (RG). These findings affirm ISNI's capacity to substantially mitigate the impact of ASR errors across diverse SLU tasks.

5.3 RQ3: Importance of Each Proposed Method to the Robustification.

Noise Level	High	Medium	Low	Clean
NoisyGen	46.43	50.00	58.33	60.71
NoisyGen (In Domain)	53.57	58.93	59.52	60.71
ISNI (Ours)	56.55	58.93	60.12	60.71
- phoneme aware generation	53.57	55.35	60.12	61.31
- intervention	52.38	54.17	60.12	60.43
-phoneme similarity loss	51.19	51.78	58.93	60.72

Table 4: Ablation study on RTE of ASR GLUE benchmark.

To evaluate the contribution of each component in ISNI, we performed ablation studies on both the DSTC10 Track2 dataset in Table 3 and the RTE task of the ASR GLUE benchmark in Table 4.

The results without the phoneme-aware generation are presented in the fourth row of Table 3 and Table 4. Removing the phoneme-aware generation led to a drop in performance across all tasks in DSTC10 Track2, as well as at high and medium noise levels in the RTE task of the ASR GLUE benchmark. This result demonstrates how phoneme-aware generation improves the robustness of SLU models in noisy environments. By accounting for word pronunciation, it ensures pseudo transcripts are ASR-plausible.

We also conducted an ablation study on the intervention by removing the do-calculus-based random corruption. For such goal, we trained a constrained decoding-based SNI model without *do*-calculus-based random corruption as in Eq. 3. For the constrained decoding method in ASR correction (Yang et al., 2022), the corruption module, which determines whether the word x^k will be corrupted, is jointly learned with the generation decoder. Such module can be considered as implementing $P(z|x^k)$ in Eq. 1.

The results, shown in the fifth row of Table 3 and Table 4, indicate a performance decrease, particularly in the KS subtask of DSTC10 Track2 and in the RTE task, which are largely influenced by lexical perturbations due to ASR errors. A similar decline is observed in the RG subtask, further emphasizing the importance of the intervention in generating diverse and generalized ASR errors. However, we noticed a slight performance increase in the KTD task. We hypothesize that this improvement may be attributed to the nature of text classification tasks, where robustness against minor lexical changes may not be as critical. Despite this, previous research suggests that abundant noise in the training set can degrade text classification performance over time (Agarwal et al., 2007), which ISNI is designed to mitigate.

Finally, we performed an ablation study on the phonetic similarity loss by training ISNI without phonetic similarity loss, relying solely on phoneme embeddings. The results, presented in the last row of Table 4, show a further reduction in performance across most noise levels. By supervising the phonetic information, phonetic similarity loss ensures that the generated noise words remain phonetically realistic, which is essential for improving model robustness in noisy conditions.

6 Conclusion

In this paper, we address the challenge of ASR generalizability within the context of SNI. We focus on enhancing the robustness of SLU models against ASR errors from diverse ASR systems. Our contributions are two-fold: Firstly, ISNI significantly broadens the spectrum of plausible ASR errors, thereby reducing biases. Second, not to lose the generality to any ASR, we generate noises that are universally plausible, or ASR$_*$-plausible, which is empirically validated through extensive experiments across multiple ASR systems.

Limitations. One limitation of ISNI is reducing the chance of making substitution errors for entire sequences of 2-3 words. As in previous works in SNI, ISNI consumes one token at a time when producing outputs. While such consumption does not restrict to make substitution errors for entire sequences, it may reduce the chance of making substitution errors for such long sequences, as ISNI decides to corrupt one token at a time. Secondly, as ISNI is trained with speech corpora collected for academic purposes, they may face challenges when adopted for real-world applications, including diverse spoken language variations such as dialects and accents. These variations can introduce noises that are not phonetically similar, which are different from the speech data used during ISNI training. This discrepancy may cause ISNI to fail in robustifying SLU models as ISNI is not prepared to handle ASR errors from those speech variations. Addressing this limitation may require enlarging the training dataset for ISNI to cover the diverse noises from spoken language variations.

Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) [NO.RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)] and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT)(No. 2022-0-00077/RS-2022-II220077, AI Technology Development for Commonsense Extraction, Reasoning, and Inference from Heterogeneous Data).

References

Sumeet Agarwal, Shantanu Godbole, Diwakar Punjani, and Shourya Roy. 2007. How much noise is too much: A study in automatic text classification. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 3–12.

Tafseer Ahmed, Muhammad Suffian, Muhammad Yaseen Khan, and Alessandro Bogliolo. 2022. Discovering lexical similarity using articulatory feature-based phonetic edit distance. *IEEE Access*, 10:1533–1544.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*.

Samuel Broscheit, Quynh Do, and Judith Gaspers. 2022. Distributionally robust finetuning BERT for covariate drift in spoken language understanding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1970–1985, Dublin, Ireland. Association for Computational Linguistics.

Chen Chen, Yuchen Hu, Chao-Han Huck Yang, Sabato Marco Siniscalchi, Pin-Yu Chen, and Eng-Siong Chng. 2024. Hyporadise: An open baseline for generative speech recognition with large language models. *Advances in Neural Information Processing Systems*, 36.

Pin-Jung Chen, I-Hung Hsu, Yi-Yao Huang, and Hung-Yi Lee. 2017. Mitigating the impact of speech recognition errors on chatbot using sequence-to-sequence model. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 497–503. IEEE.

Xuanang Chen, Jian Luo, Ben He, Le Sun, and Yingfei Sun. 2022. Towards robust dense retrieval via local ranking alignment. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 1980–1986. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Tong Cui, Jinghui Xiao, Liangyou Li, Xin Jiang, and Qun Liu. 2021. An approach to improve robustness of nlp systems against asr errors.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Matti Di Gangi, Robert Enyedi, Alessandra Brusadin, and Marcello Federico. 2019. Robust neural machine translation for clean and noisy speech transcripts. In *Proceedings of the 16th International Conference on Spoken Language Translation*, Hong Kong. Association for Computational Linguistics.

Samrat Dutta, Shreyansh Jain, Ayush Maheshwari, Ganesh Ramakrishnan, and Preethi Jyothi. 2022. Error correction in ASR using sequence-to-sequence models. *CoRR*, abs/2202.01157.

Lingyun Feng, Jianwei Yu, Yan Wang, Songxiang Liu, Deng Cai, and Haitao Zheng. 2022. ASR-Robust Natural Language Understanding on ASR-GLUE dataset. In *Proc. Interspeech 2022*, pages 1101–1105.

Karthik Gopalakrishnan, Behnam Hedayatnia, Longshaokan Marshall Wang, Yang Liu, and Dilek Hakkani-Tür. 2020. Are neural open-domain dialog systems robust to speech recognition errors in the dialog history? an empirical study. In *Interspeech 2020*.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

Georg Heigold, Stalin Varanasi, Günter Neumann, and Josef van Genabith. 2018. How robust are character-based word embeddings in tagging and MT against wrod scramlbing or randdm nouse? In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 68–80, Boston, MA. Association for Machine Translation in the Americas.

Chao-Wei Huang and Yun-Nung Chen. 2020. Learning asr-robust contextualized embeddings for spoken language understanding. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8009–8013.

Dan Jurafsky and James H Martin. 2019. Speech and language processing (3rd (draft) ed.).

Preethi Jyothi and Eric Fosler-Lussier. 2010. Discriminative language modeling using simulated asr errors. In *Eleventh Annual Conference of the International Speech Communication Association*.

Seokhwan Kim, Yang Liu, Di Jin, Alexandros Papangelis, Karthik Gopalakrishnan, Behnam Hedayatnia, and Dilek Hakkani-Tur. 2021. "how robust r u?": Evaluating task-oriented dialogue systems on spoken conversations.

Yichong Leng, Xu Tan, Rui Wang, Linchen Zhu, Jin Xu, Wenjie Liu, Linquan Liu, Xiang-Yang Li, Tao Qin, Edward Lin, and Tie-Yan Liu. 2021. FastCorrect 2: Fast error correction on multiple candidates for automatic speech recognition. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4328–4337, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhenhao Li and Lucia Specia. 2019. Improving neural machine translation robustness via data augmentation: Beyond back-translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 328–336, Hong Kong, China. Association for Computational Linguistics.

Jiexi Liu, Ryuichi Takanobu, Jiaxin Wen, Dazhen Wan, Hongguang Li, Weiran Nie, Cheng Li, Wei Peng, and Minlie Huang. 2021. Robustness testing of language understanding in task-oriented dialog. 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 2467–2480, Online. Association for Computational Linguistics.

Gustavo Penha, Arthur Câmara, and Claudia Hauff. 2022. Evaluating the robustness of retrieval pipelines with query variation generators. In *Advances in Information Retrieval: 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10–14, 2022, Proceedings, Part I*, page 397–412, Berlin, Heidelberg. Springer-Verlag.

Petko N Petkov, Gustav Eje Henter, and W Bastiaan Kleijn. 2013. Maximizing phoneme recognition accuracy for enhanced speech intelligibility in noise. *IEEE transactions on audio, speech, and language processing*, 21(5):1035–1045.

Weitong Ruan, Yaroslav Nechaev, Luoxin Chen, Chengwei Su, and Imre Kiss. 2020. Towards an ASR Error Robust Spoken Language Understanding System. In *Proc. Interspeech 2020*, pages 901–905.

Prashant Serai, Vishal Sunder, and Eric Fosler-Lussier. 2022. Hallucination of speech recognition errors with sequence to sequence learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:890–900.

Yik-Cheung Tam, Jiacheng Xu, Jiakai Zou, Zecheng Wang, Tinglong Liao, and Shuhan Yuan. 2022. Robust unstructured knowledge access in conversational dialogue with asr errors. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Xin Tian, Xinxian Huang, Dongfeng He, Yingzhan Lin, Siqi Bao, Huang He, Liankai Huang, Qiang Ju, Xiyuan Zhang, Jian Xie, Shuqi Sun, Fan Wang, Hua Wu, and Haifeng Wang. 2021. Tod-da: Towards boosting the robustness of task-oriented dialogue modeling on spoken conversations. *arXiv preprint arXiv:2112.12441*.

Yulia Tsvetkov, Florian Metze, and Chris Dyer. 2014. Augmenting translation models with simulated acoustic confusions for improved spoken language translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 616–625, Gothenburg, Sweden. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Longshaokan Marshall Wang, Maryam Fazel-Zarandi, Aditya Tiwari, Spyros Matsoukas, and Lazaros Polymenakos. 2020. Data augmentation for training dialog models robust to speech recognition errors. In *ACL 2020 Workshop on NLP for Conversational AI*.

Xueyang Wu, Rongzhong Lian, Di Jiang, Yuanfeng Song, Weiwei Zhao, Qian Xu, and Qiang Yang. 2022. A phonetic-semantic pre-training model for robust speech recognition. *CAAI Artificial Intelligence Research*, 1(1):1–7.

Jingyuan Yang, Rongjun Li, and Wei Peng. 2022. ASR Error Correction with Constrained Decoding on Operation Prediction. In *Proc. Interspeech 2022*, pages 3874–3878.

Lang-Chi Yu, Hung-yi Lee, and Lin-Shan Lee. 2016. Abstractive headline generation for spoken content by attentive recurrent neural networks with asr error modeling. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 151–157. IEEE.

Ye Yuan, Guangxu Xun, Qiuling Suo, Kebin Jia, and Aidong Zhang. 2017. Wave2vec: Learning deep representations for biosignals. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 1159–1164. IEEE.

Zhichang Zhang, Zhenwen Zhang, Haoyuan Chen, and Zhiman Zhang. 2019. A joint learning framework with bert for spoken language understanding. *IEEE Access*, 7:168849–168858.

7 Appendix

7.1 Related Works

Previously proposed methods can be categorized into three categories. First, TTS (Text-to-Speech)-ASR pipeline (Liu et al., 2021; Chen et al., 2017) adopts the TTS engine, the reverse of ASR, to convert written text into audio. The ASR$_i$ transcribes it into pseudo transcription. However, the human recording and TTS-generated audio differ in their error distributions, which makes the resulting pseudo transcriptions different from ASR transcriptions (Feng et al., 2022).

Second is textual perturbation, replacing the words $x^k \in X$ to the noise word t_i^k as follows:

$$t_i^k = argmax_{w \in W} S_i(w|x^k), \tag{17}$$

where $S_i(w|x^k)$ evaluates the score that ASR$_i$ would corrupt the written word x^k into each word $w \in W$ in the vocabulary set W. A widely adopted type of $S_i(w|x^k)$ is a confusion matrix built from the paired written and ASR transcribed corpora (Jyothi and Fosler-Lussier, 2010; Yu et al., 2016) or phonetic similarity function (Li and Specia, 2019; Tsvetkov et al., 2014).

A representative of the above two categories is TOD-DA which embodies both categories.

Third is the auto-regressive generation, where auto-regressive PLMs such as GPT-2 or BART are supervised to maximize the likelihood of the ASR transcription T_i given its written text X. Adopting PLMs, the auto-regressive generation can consider contextual information and generate more ASR$_i$-plausible noise words (Cui et al., 2021).

Auto-regressive generation is biased to ASR$_i$, limiting the SLU model used for ASR$_i$.

Our distinction is generalizing SNI so that the SLU tasks can be conducted with ASR$_*$.

7.2 Proof of Eq. 2

Using the do operator to the conventional likelihood, $P(Y|do(X))$ is transformed as follows:

$$P(Y|do(X)) = \sum_z P(Y, z|do(X)) \tag{18}$$

$$= \sum_z P(Y|do(X), z) \cdot P(z|do(X)) \tag{19}$$

$$= \sum_z P(Y|X, z) \cdot P(z|do(X)) \tag{20}$$

Then, further transition is conducted by applying Rule 3 of do-calculus. Rule 3 states that we can remove the do-operator if X and Z are independent in $G_{\bar{X}}$, a modified version of the causal graph of SNI where all arrows incoming to X are removed. In $G_{\bar{X}}$, where there are two paths, $X \rightarrow Y$ and $Z \rightarrow Y$, X, and Z are independent, $X \perp Z$, as there is no valid path between X and Z. Removing the do-operator, Eq. 20 is evaluated as follows:

$$P(Y|do(X)) = \sum_z P(Y|X, z) \cdot P(z). \tag{21}$$

7.3 ASR systems for SNI and Downstream Tasks.

For training the SNI model, we used the open-source commercial ASR system, Mozilla DeepSpeech. Mozilla DeepSpeech, which we adopted for SNI model training, is an RNN-based end-to-end ASR system trained on a 1700-hour audio dataset. DeepSpeech shows similar word error rates to famous commercial ASR systems such as Google Translate's speech-to-text API and IBM Watson Speech-to-text, in various benchmark datasets such as Librispeech clean test and Commonvoice as the table below shows. This similarity in performance makes it a relevant and practical choice for our study, providing a realistic and challenging testbed for our methods.

ASR	DeepSpeech	Google Translate	IBM Watson
Librispeech Clean	0.07	0.11	0.11
CommonVoice	0.32	0.32	0.39

Table 5: WER of commercial ASR systems.

Specifically, for the DSTC Track2 dataset, an unknown ASR system is used to generate transcriptions (Kim et al., 2021). For the ASR GLUE benchmark, the LF-MMI TDNN ASR system is adopted to generate transcriptions (Feng et al., 2022). ASR GLUE benchmark adopts an LF-MMI time-delayed neural network-based ASR syetem trained on a 6000-hour dataset.

7.4 Dataset Details

SNI Training For training SNI, we used the same datasets with our main baseline, Noisy-Gen (Cui et al., 2021), which used popular speech corpora, Common Voice, tatoeba audio, and LJSpeech-1.1 and MSLT, to collect ASR error transcriptions. Specifically, the audio recordings of the above corpora are fed to the DeepSpeech and get ASR transcriptions. Then, we compare ASR transcription with ground-truth transcription, ignoring punctuation and casing errors, so that only erroneous transcriptions would remain. Finally, we obtain about 930k pairs of ground-truth transcription and the ASR-noised transcription pair. Among the resulting pairs, those from Common Voice, Tatoeba audio, and LJSpeech-1.1 are used for the training set and the others from MSLT are used for the validation set.

SLU Testing To show the ASR generalizability of the SNI models, we adopt two distinct SLU benchmarks, ASR GLUE and DSTC10 Track 2. DSTC10 Track2 dataset models task-oriented dialogue systems with unstructured knowledge access in a spoken language. We adopt the DSTC10 Track2 dataset as it covers various tasks in NLP in the dialogue domain where SLU is required frequently. Specifically, it consists of three successive subtasks covering various SLU tasks: Knowledge-seeking Turn Detection (KTD), Knowledge Selection (KS), and Knowledge-grounded Response Generation (RG). First, KTD aims to determine whether the dialogue turn requires external knowledge access or not as a classification. Once determined to require external knowledge, the second step is KS, which aims to retrieve the appropriate knowledge snippet by estimating the relevance between the given dialogue context and each knowledge snippet in the knowledge base. Finally, the response is generated in RG, based on the dialogue context and the selected knowledge snippet.

In the DSTC10 Track2 dataset, human responses are transcribed by an unknown ASR system.

Another benchmark, ASR GLUE is an SLU version of the widely adopted NLU benchmark, GLUE. It provides the written GTs for the training set and the transcriptions of 3 noise levels spoken by 5 human speakers for the development set and test set. As the ground-truth label for the test set is unavailable, we report the results on the development set and sample the validation set from the pseudo transcripts generated from the training set. Among various subtasks, we provide the results of two NLI tasks, QNLI and RTE, and 1 sentiment classification task, SST, which the DSTC10 Track2 dataset does not contain.

7.5 Details of ISNI model

We used Transformer decoder (Vaswani et al., 2017) with 1 layer which has 12 attention heads and 768 hidden layer dimensions. We trained ISNI for 20 epochs with Adam optimizer with a learning rate of 0.00005. Also, we set λ_w and $\lambda_{phoneme}$ as 0.5 to balance the semantic and phonetic information.

7.6 Quantitative study on the generated pseudo-transcripts.

In this section, we quantitatively study the characteristics of the pseudo-transcripts generated by our ISNI. For this study, We generated pseudo transcripts in MSLT, our development set for SNI training. We set $P(z) = 0.45$ for pseudo transcripts generation, which is similar to the word error rate of ASR transcription as we will show in Table 7. We analyze the following characteristics:

i) How phonetically similar are our generated pseudo transcripts?
ii) The word error rate.
iii) Which error types composes the noise in the generated pseudo transcripts?

First, we show that PLMs are insufficient to generate the ASR$_*$-plausible pseudo transcriptions. The noise words would be ASR$_*$-plausible if it is phonetically similar to its written form as an ASR system would incorrectly transcribe into phonetically similar noise words. Therefore, we compared the phoneme edit distance $D(w^p, w^q)$ between the written GT and the pseudo transcriptions. For a fair comparison, we set all tokens to have identical z for the noise word generation to ensure that the identical words are corrupted.

Model	PD (↓)
Ours	62.02
- phoneme-aware generation	72.52

Table 6: Phoneme edit distance between generated pseudo transcriptions and written GTs (Lower is better).

Table 6 shows that the pseudo transcriptions without phoneme-aware generation show 17% larger phonetic distance. This result shows that PLMs are not enough to generate the ASR$_*$-plausible pseudo transcriptions and ignorance of phonetic information is the obstacle to generating ASR$_*$-plausible pseudo transcriptions.

Then, we study the word error rate of the generated pseudo transcripts. The results in Table 7 show

Model	WER
DeepSpeech	0.46
Ours	0.66
Noisy-Gen	0.76

Table 7: Word error rate of the DeepSpeech transcriptions and the pseudo transcriptions generated by ours and Noisy-Gen.

that pseudo transcripts generated by our ISNI contain more errors than P(z). However, compared to the baselines, the word error rate generated by our methods is more controlled, as we control whether to corrupt the word by P(z).

Then, we break down word error rate results by error types of insertion/deletion/substitutions. Table 8 shows that three error types take up similarly in DeepSpeech transcription and pseudo transcriptions by our ISNI. This result shows that our ISNI can well handle three error types.

error type	insertion	deletion	substituion
DeepSpeech	0.20	0.29	0.51
Ours	0.25	0.16	0.59

Table 8: Error type of the DeepSpeech transcriptions and pseudo transcripts generated by our ISNI.

7.7 Generated Pseudo Transcripts Examples

We provide the examples of the generated pseudo transcripts in Table 9 and 10. Among the tokenized input, z of $\#\#ial$ was set to 1, thus the constrained decoder generates its noise word. The constrained

input	as bestial		
tokenized input	as	best	##ial
z	0	0	1
generated output	-	-	at ##ial <eos>
pseudo transcript	as best atial		

Table 9: Examples of pseudo transcripts generated by our ISNI.

decoder made 1 substitution error $(bestial \rightarrow best)$ and 1 insertion error $(atail)$.

input	only labored the gags					
tokenized input	only	larbor	##ed	the	gag	##s
z	0	0	1	1	0	1
generated output	-	-	##ed labor <eos>	the ##s <eos>	-	<eos>
pseudo transcript	only labored labor thes gag					

Table 10: More examples of pseudo transcripts generated by our ISNI.

7.8 Use of AI Assistants

We used ChatGPT for grammatical corrections.