Using Phoneme Representations to Build Predictive Models Robust to ASR Errors

Anjie Fang
Amazon
njfn@amazon.com

Simone Filice
Amazon
filicesf@amazon.com

Nut Limsopatham*
Microsoft AI
nutli@microsoft.com

Oleg Rokhlenko
Amazon
olegro@amazon.com

ABSTRACT

Even though Automatic Speech Recognition (ASR) systems significantly improved over the last decade, they still introduce a lot of errors when they transcribe voice to text. One of the most common reasons for these errors is phonetic confusion between similar-sounding expressions. As a result, ASR transcriptions often contain "quasi-oronyms", i.e., words or phrases that sound similar to the source ones, but that have completely different semantics (e.g., *win* instead of *when* or *accessible on defecting* instead of *accessible and affecting*). These errors significantly affect the performance of downstream Natural Language Understanding (NLU) models (e.g., intent classification, slot filling, etc.) and impair user experience. To make NLU models more robust to such errors, we propose novel phonetic-aware text representations. Specifically, we represent ASR transcriptions at the phoneme level, aiming to capture pronunciation similarities, which are typically neglected in word-level representations (e.g., word embeddings). To train and evaluate our phoneme representations, we generate noisy ASR transcriptions of four existing datasets - Stanford Sentiment Treebank, SQuAD, TREC Question Classification and Subjectivity Analysis - and show that common neural network architectures exploiting the proposed phoneme representations can effectively handle noisy transcriptions and significantly outperform state-of-the-art baselines. Finally, we confirm these results by testing our models on real utterances spoken to the Alexa virtual assistant.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**; **Neural networks**; *Speech recognition*; *Phonology / morphology*.

KEYWORDS

Phoneme Embeddings, Deep Learning, Natural Language Understanding, ASR Errors, Virtual Assistant

*This work was completed while Nut Limsopatham was at Amazon.

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401050

ACM Reference Format:
Anjie Fang, Simone Filice, Nut Limsopatham, and Oleg Rokhlenko. 2020. Using Phoneme Representations to Build Predictive Models Robust to ASR Errors. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401050

1 INTRODUCTION

Nowadays, voice-enabled systems are gaining more and more popularity, and virtual assistants, such as Amazon Alexa, Apple Siri or Google Home, are becoming part of our daily life. In particular, they have been used, for example, for accessing contents on the Web, controlling smart devices, and managing calendars, through different applications, e.g. voice search engine [35] and voice shopping [14].

In such systems, the Spoken Language Understanding (SLU) is usually performed in two steps: first an Automatic Speech Recognition (ASR) is used to transcribe human speech; then Natural Language Understanding (NLU) models are applied on ASR transcriptions to interpret users' requests. Different from traditional approaches, where NLU is applied on the original text, applying it on ASR transcriptions poses new challenges, as ASR systems often generate transcriptions with errors [6, 20]. These ASR errors can cause failures in downstream applications of virtual assistants, such as intention classification or slot filling [28], affecting the end-user experience.

Traditionally, researchers distinguish between three main types of ASR errors: insertions, deletions, and substitutions. However, all these errors are just an outcome of a phonetic confusion in the ASR model, causing a phrase in a human speech to be incorrectly transcribed to a "quasi-oronym", i.e., phrases with different meanings that sound very similar. Therefore, classic approaches that operate on word or even character-level representations cannot recover from such errors. In this paper we explore the usage of lower level representations, namely phoneme-based representations, to alleviate this problem. As phonemes are the smallest units of sound in a language, we expect the ASR transcription to be more similar to the correct utterance at the phoneme level than at the character or word levels. Hence, we argue that injecting phonetic information into NLU models can improve their robustness to ASR errors. More specifically, we propose to represent ASR transcriptions as sequences of phonemes. Following the deep learning approach for text processing, we map phonemes to phoneme embeddings and propose several methods to train phoneme embeddings that are able to capture pronunciation similarities. Finally, we use these

pre-trained embeddings as inputs to Neural Network architectures for solving NLU tasks.

The contribution of this paper is fourfold: (*i*) we design four methods for training phoneme embeddings using sequence-to-sequence and word2vec-based models, and evaluate them; (*ii*) we define a pipeline for contaminating existing datasets with ASR errors, and we use this pipeline to generate noisy versions of four well-known Natural Language Processing datasets[1]; (*iii*) we describe how to integrate phoneme embeddings into existing Neural Network architectures, e.g., LSTM and CNN, showing how the proposed phoneme embeddings can be jointly used with standard embeddings, i.e., character and word embeddings; (*iv*) we conduct an intensive experimental evaluation on the generated datasets, as well as on real utterances spoken to the Alexa virtual assistant; our experimental results demonstrate that models exploiting our phoneme representation can significantly improve classification performance on datasets containing ASR errors compared to models operating only on standard character or word representations.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 describes our phoneme-level representations and the proposed methods to automatically learn phoneme embeddings. Section 4 explains the data generation pipeline that is used to automatically generate datasets containing ASR errors. We run a qualitative analysis of our phoneme embedding spaces in Section 5 and report experimental results on the tasks of sentiment analysis and question classification in Section 6 using the generated datasets. In Section 7 we confirm these results by investigating the domain classification task on a real Alexa dataset. Finally, we provide concluding remarks in Section 8.

2 RELATED WORK

Some previous works have explored the possibility of using error detection systems to trigger clarification questions to users. Tam et al. [33] tackled the error detection task with a Recurrent Neural Network, while Pellegrini and Trancoso [23] used features obtained from different knowledge sources to complement an encoder-decoder model.

Other works tried to directly correct the ASR transcriptions. Sarma and Palmer [27] proposed an unsupervised method based on lexical co-occurrence statistics for detecting and correcting ASR errors. Shivakumar et al. [29] designed a noisy channel model for error correction that can learn from past ASR errors. D'Haro and Banchs [5] proposed an correction procedure using a phrase-based machine translation system.

In all the above approaches, the benefit comes at the cost of introducing additional components in the NLU pipeline. In our work, instead, we explore a different research direction: we aim to make downstream models more robust to ASR errors by using phonetic-aware text representations. In particular, we propose to adopt phoneme embeddings to replace or complement common text representations, e.g., word embeddings [18, 24, 25], or character embeddings [11].

Few existing works studied phoneme embeddings. Li et al. [13] explored the application of phoneme embeddings for the task of speech-driven talking avatar synthesis to create more realistic and expressive visual gestures. Silfverberg et al. [30] proposed an approach to learn phoneme embeddings that can be used to perform phonological analogies. Toshniwal and Livescu [34] discussed the usage of phoneme embeddings for the task of grapheme-to-phoneme conversion. To our best knowledge, no previous work focused on the application of phoneme embeddings to improve NLU models operating on transcriptions containing ASR errors.

Another line of works handle ASR errors at downstream tasks. The general approach is to pass intermediate ASR results, in the forms of lattices or embeddings, to the downstream model. Lattices can be either at the word level or at the phoneme level [15]. Other solutions consist of developing end-to-end models for SLU [2]. In this case ASR and SLU models are integrated and typically need a lot of data to be trained. Conversely, our proposed approach can rely on off-the-shelf ASR systems (which typically do not give access to intermediate results) and train only SLU models, which typically require much less data.

3 PHONEME-LEVEL REPRESENTATIONS

In deep learning methods for NLP, text is typically represented as a sequence of tokens, e.g., words or characters, which are modeled using embeddings. This approach is proven to be effective for written text [10, 11, 36], however, it is not inherently robust to ASR errors. Table 1 lists three typical examples of ASR transcriptions containing quasi-oronym errors[2]. In the first example, the words *what & canadian* are incorrectly transcribed to words with completely different meanings, i.e., *well & comedian*. Since word embeddings typically reflect word semantics, the word embeddings of these misrecognized words will be very dissimilar from the reference ones. On the other hand, when an ASR model does not correctly recognize a word or a phrase, it typically confuses it with a quasi-oronym, e.g., *canadian* vs. *comedian* and *affecting* vs. *defecting* in the first and the third example in Table 1[3]. This suggests that the sequence of phonemes of an ASR transcription tends to be similar to the sequence of phonemes of the correct text.

A common metric to evaluate the performance of speech recognition or machine translation systems is Word Error Rate (WER). It measures the percentage of incorrectly transcribed words (Substitutions (S), Insertions (I), Deletions (D)). It is defined as follows:

$$WER = \frac{S + D + I}{N} \tag{1}$$

where N is the number of words in the reference text. In the same vein, we further define two additional metrics, Character Error Rate (CER) and Phoneme Error Rate (PER), as the extensions of WER to characters and phonemes, respectively. Intuitively, if ASR confuses similar sounding words or phrases, PER will be smaller than CER and WER (see Table 1). In Section 4 we further confirm this intuition on entire datasets. Hence, we argue that representing text as a sequence of phoneme embeddings can help when dealing with ASR errors.

[1] https://registry.opendata.aws/asr-error-robustness

[2] The ASR transcriptions were created by the data generation pipeline, as described in Section 4.

[3] We convert text to phonemes using the phonemizer tool (github.com/bootphon/phonemizer), which is based on the speech synthesis system Festival [1]. A phoneme is represented by 2-letter notation.

Table 1: Examples of ASR errors. WER, CER and PER are word, character and phoneme error rates, respectively.

Reference text (followed by its phonemes)	Transcribed text (followed by its phonemes)	WER	CER	PER
What Canadian city has the largest population w-ah-t k-ax-n-ey-d-iy-ax-n s-ih-t-iy hh-ae-z dh-ax l-aa-r-jh-ax-s-t p-aa-p-y-ax-l-ey-sh-ax-n	*Well, comedian city has the largest population* w-eh-l k-ax-m-iy-d-iy-ax-n s-ih-t-iy hh-ae-z dh-ax l-aa-r-jh-ax-s-t p-aa-p-y-ax-l-ey-sh-ax-n	0.285	0.205	0.142
What is amitriptyline w-ah-t ih-z ae-m-iy-t-r-ih-p-t-ax-l-ay-n	*One is amateur delete* w-ah-n ih-z ae-m-ax-t-er d-ax-l-iy-t	1.000	0.631	0.529
Remarkably accessible and affecting r-ax-m-aa-r-k-ax-b-l-iy ax-k-s-eh-s-ax-b-ax-l ae-n-d ax-f-eh-k-t-ax-ng	*Remarkably accessible on defecting* r-ax-m-aa-r-k-ax-b-l-iy ax-k-s-eh-s-ax-b-ax-l ax-n d-ax-f-eh-k-t-ax-ng	0.500	0.093	0.074

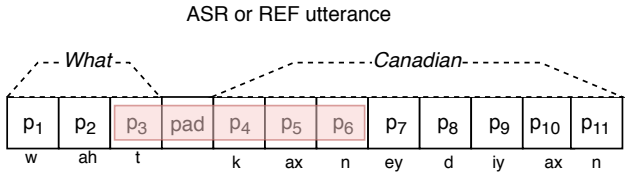

Figure 1: Context window with size 2 around p_4 in *p2vc*.

Similar to word or character embeddings, phoneme embeddings can be directly learned during the training process of a Neural Network that is designed to solve a specific task. However, these learned phoneme embeddings do not necessarily capture pronunciation aspects. Therefore, in Sections 3.1 and 3.2 we propose four methods, including a sequence-to-sequence (seq2seq) model and variants of word2vec, to train phoneme embeddings reflecting pronunciation similarity. For readability purposes, we denote the correct utterance as the reference (REF), while we denote the text transcribed by an ASR model as the ASR utterance. Note that, different from standard word embeddings, our phoneme models require both REF and ASR utterances for training.

3.1 Phoneme2Vec

Word2vec [17, 18] is widely used to learn word embeddings using a shallow neural network trained on language modeling tasks. There are two variants of word2vec, namely the continuous bag-of-words and the skip-gram model. In the skip-gram architecture, the model uses the current word to predict the surrounding words in a context window. To learn phoneme embeddings we design phoneme2vec, a modified skip-gram model that operates at the phoneme level, instead of the word level. We propose three variants by considering different definitions of context.

3.1.1 ***p2vc: phoneme2vec on surrounding phonemes.*** This is the natural extension of word2vec to phonemes: given a phoneme we want to predict its surrounding phonemes. Specifically, an utterance (either a REF or an ASR utterance) is represented by its sequence of phonemes (the padding symbol is used to separate words). The traditional word2vec procedure is then applied on the phoneme sequence to predict phonemes in the same context windows. Figure 1 illustrates an example with a 2-size context window: given the central phoneme p_4, *p2vc* has to predict phonemes p_3, p_5 and p_6, as well as the padding symbol (pad).

We decided not to limit the context of a phoneme to its word as the ASR might have failed the word segmentation. However, we explicitly consider the padding symbol as it represents the "absence of sound" captured by the ASR.

Word2vec was designed to generate word embeddings reflecting semantic and syntactic aspects; however, we aim to capture pronunciation similarities. Intuitively, two phonemes are similar if the ASR often confuses them. Following this intuition we propose two variants of phoneme2vec, as well as a sequence-to-sequence model for training phoneme embeddings in the rest of this section. In *p2vc* , ASR or REF utterances are always analyzed individually. Conversely, our proposed models operate on <ASR, REF> pairs, to leverage their dissimilarity and automatically learn which sounds the ASR confuses.

3.1.2 ***p2vm: phoneme2vec on mixed REF and ASR utterances.*** In this approach we mix REF and ASR utterances at phoneme level in an alternating way, as shown in Figure 2: if $p_1^R, p_2^R, \ldots$ and $p_1^A, p_2^A, \ldots$ are the sequences of phonemes in the REF and ASR utterances, respectively, $p_1^R, p_1^A, p_2^R, p_2^A \ldots$ is the resulting mixed sequence. Given a phoneme, the model aims to predict the surrounding phonemes in the mixed sequence. For example, let us consider the REF utterance *"when (w-ih-n) iPhone 7 was released"* and its ASR counterpart *"win (w-eh-n) iPhone 7 was released"*. The underlying idea is trying to make two confused phonemes (in this case, *eh* and *ih*) appear in their reciprocal context windows. The assumption is that, if the ASR utterance contains few errors, phonemes with a similar pronunciation appear in very similar, possibly the same, positions and therefore they will be close in the mixed sequence. This means that they will occur in their reciprocal contexts, allowing phoneme2vec to learn embeddings reflecting pronunciation similarities.

3.1.3 ***p2va: phoneme2vec on aligned REF and ASR utterances.*** The previous approach relies on the hypothesis that REF and ASR utterances have a similar number of phonemes, and that the two phoneme sequences naturally align when mixed. Although this is often true, we propose a more general solution that involves an explicit alignment. We directly pair phonemes in a REF utterance with their aligned phonemes in the ASR utterance using the Needleman-Wunsch alignment algorithm [19], as shown in Figure 3. Then the context of a given phoneme in the REF [ASR] utterance is its aligned phoneme in the ASR [REF] utterance as well as the phonemes surrounding it. For example, if we consider a window

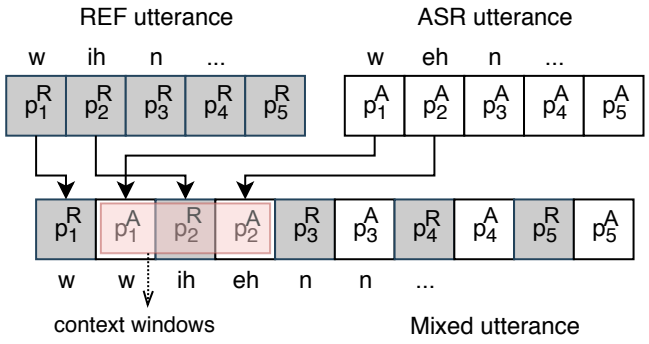

Figure 2: Mixing ⟨REF, ASR⟩ utterances in *p2vm*.

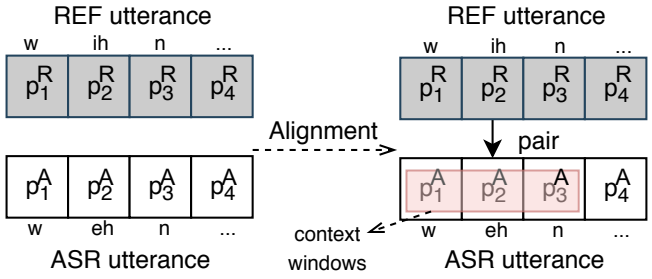

Figure 3: Aligning ⟨REF, ASR⟩ utterances in *p2va*.

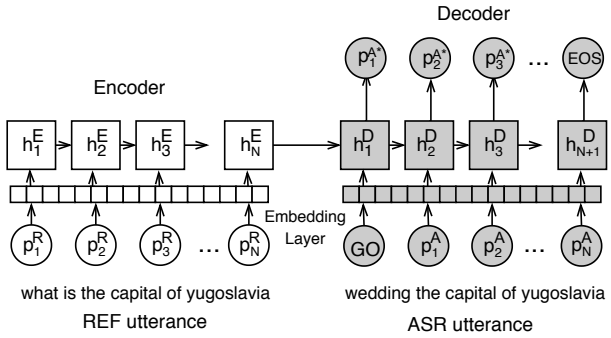

Figure 4: Training phoneme embeddings by *s2s*.

size of 1, the context phoneme of *ih* are *w, eh* and *n*, as shown in Figure 3.

3.2 Phoneme Embeddings from Seq2Seq - *s2s*

A very intuitive way of training phoneme embeddings is to use a seq2seq model [9, 32], since it can map the entire REF utterance (i.e., the input) to its ASR utterance (i.e., the output). An advantage of the seq2seq model is that it does not require any phoneme alignment procedure.

Figure 4 shows our seq2seq model, where LSTM layers are used in both the encoder and decoder. A REF utterance is represented as a sequence, i.e., $\{p_1^R, p_2^R, ..., p_N^R\}$. During the encoding phase, at each time step t, the LSTM reads a phoneme of the sentence and updates the hidden states h_t^E:

$$h_t^E = f(h_{t-1}^E, p_t^R) \tag{2}$$

where f represents LSTM operations [8], and p_t^R indicates the current phoneme in the REF utterance. After reading the entire utterance, the last hidden state of the LSTM, h_N^E, is passed to decoder. The initial state h_0^E of the LSTM encoder is a zero vector. At step t, the hidden state of the LSTM decoder h_t^D is calculated as:

$$h_t^D = f(h_{t-1}^D, p_{t-1}^A) \tag{3}$$

where $h_0^E = h_N^D$, p_0^A is the start symbol "GO", and p_{t-1}^A is the $(t-1)$th phoneme of the ASR utterance. We train the seq2seq model to predict the next correct phoneme of the ASR utterance given the REF utterance and the previous ASR phonemes. The next phoneme p_t^{A*} (i.e., output of the LSTM decoder) is predicted using conditional distribution:

$$P(p_t^{A*}|p_{t-1}^A, p_{t-2}^A, ..., p_1^A, h_N^E) = g(h_t^D) \tag{4}$$

where g is the softmax activation function. As a loss function, we use the categorical cross-entropy between the prediction $g(h_t^D)$ and the one-hot encoding of p_t^A.

In this sequence-to-sequence architecture, we add phoneme embedding layers before the encoder and decoder. During the training process, the REF utterances and their corresponding ASR utterances are transformed into sequences of phonemes that are given as inputs to the encoder and decoder, respectively[4]. Finally, the embedding layer of the decoder is used as pre-trained phoneme embeddings[5].

4 GENERATION OF NOISY DATASETS

The proposed training procedures for learning phoneme embeddings require a corpus of corresponding REF and ASR utterances. In the ASR literature there are several corpora [e.g., in 3, 21] containing text and the associated human speech, which could be provided as inputs to an ASR system to obtain the required ⟨REF, ASR⟩ utterance pairs. However, to verify the impact of the proposed phoneme embeddings on specific prediction tasks, e.g., classification tasks, we also need such data to be annotated according to a desired class taxonomy. Unfortunately, to the best of our knowledge, human annotated speech corpora are not publicly available. Since speech transcription and annotation are expensive and labor-intensive processes, we propose an automatic data generation pipeline, as shown in Figure 5. In particular, we use this pipeline to automatically generate ASR transcriptions of existing annotated datasets.

First, we use a Text To Speech (TTS) tool to generate speech audios of sentences from a textual corpus (REF utterances). To produce realistic speech, we inject different types of synthetic noise into the audio. By using SSML tags[6], it is possible to directly apply several effects to the produced speech, such as changing the prosody, emphasizing or pausing. Moreover, we add 20 types of ambient noise[7] to the audio, e.g., traffic noise, or restaurant noise. Overall, for

[4]We also tried the opposite, but the results did not show significant differences.
[5]We also tried the phoneme embeddings from the encoder layer, but we did not observe substantial differences.
[6]www.w3.org/TR/speech-synthesis11/
[7]We use the ambient noise from www.pacdv.com/sounds/ambience_sounds.html.

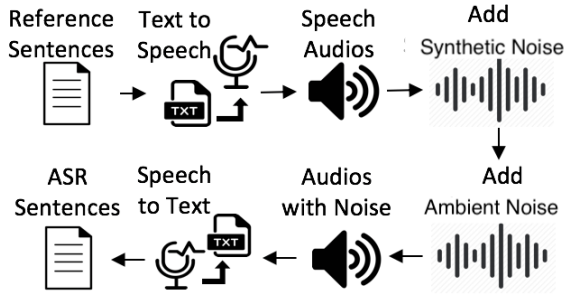

Figure 5: Our data generation pipeline.

a given audio file, we add one random synthetic noise using SSML tags and one random ambient noise. Based on our manual analysis, we set the volume of the ambient noise to -5 dB and that of the audio speech to 0 dB, to obtain reasonably understandable audios. Lastly, the noisy audios are passed to an ASR tool to generate the transcriptions. We keep only transcriptions containing ASR errors, by repeating the process for correctly transcribed utterances.

Even though the above pipeline is generic, in our experiments we use two standard off-the-shelf tools for TTS and ASR, namely Amazon Polly[8] and Amazon Transcribe[9]. We invoke the proposed pipeline to obtain noisy versions of four datasets:

- **SST**. The Stanford Sentiment Treebank (SST) dataset [31] contains sentences with their labels from a five-point sentiment scale.
- **TQ**. The TREC Question classification dataset contains questions with 6 (TQ-6) or 50 (fine-grained, TQ-50) question types [12].
- **SQuAD**. This dataset contains approximately 150k crowdsourced questions regarding a number of Wikipedia articles [26]. We randomly select 20 questions from each of a total of 442 Wikipedia articles[10].
- **SUBJ**. This is the subjectivity dataset (10k sentences) from Pang and Lee [22]. We randomly select 5k sentences out of 10k[10].

We list some statistics of the four datasets in Table 2. The PER in the four datasets is lower than their WER and CER, confirming the intuition that a phoneme-based text representation is the least affected by ASR errors.

5 QUALITATIVE ANALYSIS

As discussed in Section 3, we defined four different models for pre-training phoneme embeddings: the seq2seq model *s2s* and three phoneme2vec variants, i.e., *p2vc*, *p2vm*, and *p2va*. As pre-training examples, we use ⟨REF, ASR⟩ utterance pairs from the union of SQuAD and SUBJ datasets (a total of 13,840 pairs). ASR transcriptions contain errors from the specific ASR system adopted in the data generation pipeline. Therefore, the resulting phoneme embeddings should link the phonemes that this particular ASR system often confuses. We denote such embeddings with the subscript *asr*, e.g., $s2s_{asr}$ or $p2vm_{asr}$.

[8] aws.amazon.com/polly
[9] aws.amazon.com/transcribe
[10] We did not transcribe the entire datasets due to budget constraints.

Additionally, we also employ CMU Pronouncing Dictionary[11] to extract roughly 8000 words having multiple accepted pronunciations. For each word, we couple all its alternative pronunciations and we consider the resulting pairs as ⟨REF, ASR⟩ utterances for training our phoneme embeddings. In this case the data generation pipeline is not used, and the phoneme embeddings we generate express general pronunciation aspects. We denote such embeddings with the subscript *dict*, e.g., $s2s_{dict}$ or $p2vm_{dict}$. Note that the CMU Pronouncing Dictionary was also adopted by Hixon et al. [7] to study phoneme similarities.

The context window parameter in phoneme2vec (see Section 3) reflects how many phonemes we take into account when predicting the current phoneme. We set this value to 2 according to some preliminary experiments. For *p2va* we also use a 0 context window (we refer to the resulting models as $p2va^0_{dict}$ and $p2va^0_{asr}$), to force the model to acquire only pronunciation similarities between phonemes confused by the ASR. In fact, using a 0 window size implies that in the context of a given phoneme from an ASR [REF] utterance, there is only the aligned phoneme in the corresponding REF [ASR] utterance. Overall, we create 10 different pre-trained phoneme embeddings. We set the dimension size of the embedding vector to 20 for all the methods. This setting is reasonable since the total number of phonemes is only 40.

To visually assess different phoneme embeddings, Figures 6 and 7 show a 2D projection of $s2s_{asr}$ and $p2vc_{asr}$ obtained by using t-SNE [16]. We use different colours to highlight three groups of phonemes. The phonemes in each group are similar in terms of pronunciation. It is clear that the pre-trained embeddings using seq2seq (i.e., $s2s_{asr}$) can reasonably cluster these similar phonemes together, such as "ay", "ey", "iy" and "oy" (in red cycle in Figure 6). We observe similar outcomes by using *p2va* and *p2vm*. However, as shown in Figure 7 these similar phonemes are not relatively close when the training model is $p2vc_{asr}$. This suggests that *p2vc*, i.e., the adaptation of the classic word2vec to phonemes, is not suited for learning pronunciation aspects, while the other proposed models more effectively capture these desired properties.

6 EXPERIMENTAL EVALUATION ON GENERATED DATA

In this section, we evaluate the impact the proposed phoneme-based representations in classification tasks. In addition to the 10 different embedding spaces introduced in the previous section, we also use randomly initialized vectors (denoted as *rnd*) to explore the un-pretrained case.

6.1 Neural Models on Phonemes

We integrate the proposed phoneme embeddings in standard neural network models for sentence classification, i.e., Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). For the CNN-based model, we extend the CNN proposed in Kim [10] to operate on multiple inputs, as shown in Figure 8. We consider different types of inputs, i.e., the sequences of word, character and phoneme embeddings; for each input sequence a convolutional layer and a max pooling layer are applied to create a sentence

[11] speech.cs.cmu.edu/cgi-bin/cmudict

Table 2: Datasets. "C" is the number of classes. "V" is the vocabulary size. "L" is the average sentence length.

	C	V	L	Train	Dev	Test	WER	CER	PER
SST	5	17,836	17.38	8,534	1,099	2,210	0.244	0.113	0.113
TQ	6/50	9,492	8.89	5,452	-	500	0.319	0.187	0.148
SQuAD	-	13,240	12.28	8,840	-	-	0.241	0.118	0.107
SUBJ	2	14,410	18.66	5,000	-	-	0.229	0.100	0.097

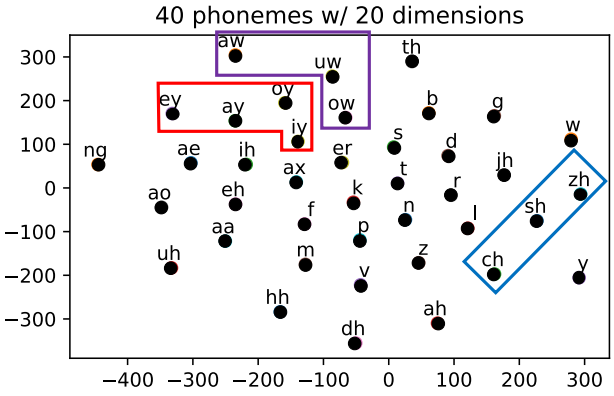

Figure 6: 2D visualization of $s2s_{asr}$ embeddings.

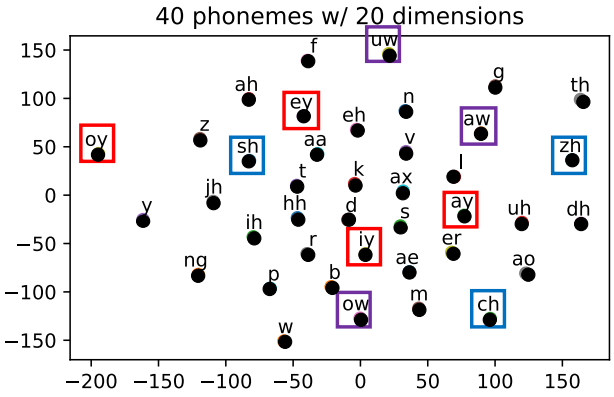

Figure 7: 2D visualization of $p2vc_{asr}$ embeddings.

vector; finally, the sentence vectors from different inputs are concatenated and classified using a fully connected layer with the softmax activation function.

In addition, we employ a multi-input variant of the character-level neural network proposed in Kim et al. [11]. As shown in Figure 9, this is an LSTM-based model, where the information extracted from different input types (i.e., word, character and phoneme) is aggregated at the word level. This architecture consists of four main steps: (*i*) the sequences of character and/or phoneme embeddings are passed to the corresponding inputs in groups of words; (*ii*) each group[12] is processed by a convolution layer followed by a max pooling layer that creates word vectors; (*iii*) these word vectors are

[12]Basically each group corresponds to a word and it is represented as a matrix whose rows are character/phoneme embeddings.

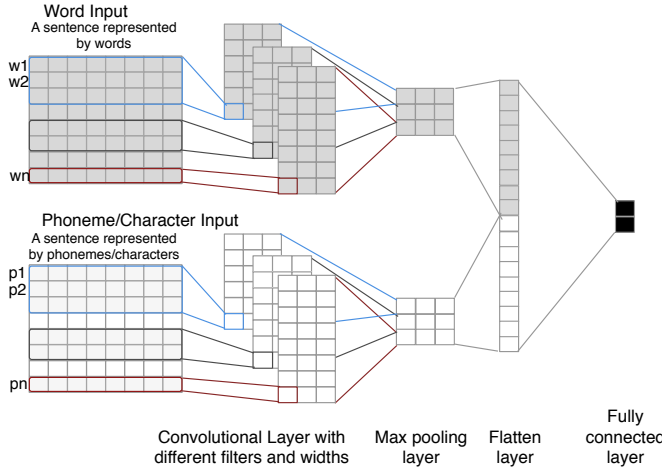

Figure 8: The multi-input CNN.

concatenated with the word embeddings provided by the word input to create a sequence of enriched word embeddings; (*iv*) an LSTM operates on this sequence, and its final hidden state is classified by a dense layer with the softmax activation function.

For the sake of simplicity, we denote the CNN-based and the LSTM-based models as *cnn* and *lstm*, respectively. To prevent confusion, lowercase is used for referring to the entire architectures and the uppercase for the CNN and LSTM layers.

We test the proposed Neural Models with various combinations of the three available inputs, i.e., words (*w*), characters (*c*) and phonemes (*p*). We specify which input is used in the model prefix, e.g., *wp-lstm* means that the model is the LSTM-based architecture operating on words and phonemes, while *c-cnn* is the CNN-based model using only the character input.

6.2 Experimental Settings

We run an extensive experimental evaluation on the SST and TQ datasets. On the TQ dataset we performed experiments with both the 6-class and 50-class settings. We use word embeddings (300 dimensions) trained on Wikipedia from GloVe [24]. The character embeddings (20-dimensional) are randomly initialized and trainable, whereas the pre-trained phoneme embeddings are not trainable[13], except the randomly initialized ones (*rnd* model).

For all the convolutional layers used in the *cnn* and *lstm* models, we set the filter windows (*w*) to {1, 2, 3, 4} with 256 as the filter size. The pooling size is $l - w$, where l is the length of the sentence

[13]In our preliminary experiments, we verified that making them trainable hurts the performance.

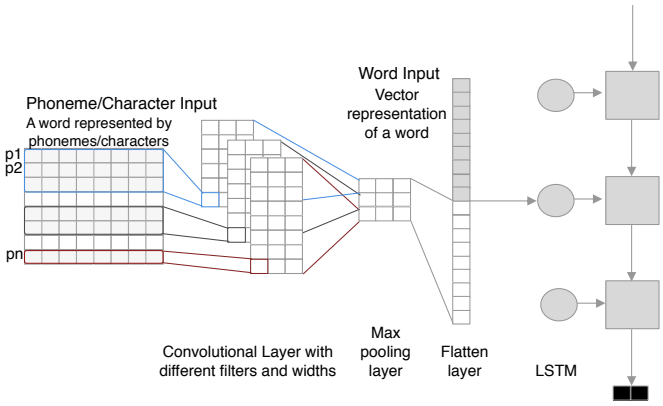

Figure 9: The multi-input LSTM.

Table 3: The upper-bound (↑) and lower-bound (↓) accuracy of classification models.

	SST		TQ-50		TQ-6	
	↑	↓	↑	↓	↑	↓
w-cnn	0.436	0.406	**0.844**	0.627	0.899	0.751
c-cnn	0.378	0.350	0.693	0.549	0.829	0.662
wc-cnn	0.439	0.400	0.842	**0.648**	0.910	0.752
w-lstm	**0.452**	**0.414**	0.804	0.628	**0.917**	0.743
c-lstm	0.288	0.293	0.704	0.564	0.842	0.678
wc-lstm	0.430	0.409	0.832	0.632	0.910	**0.756**
[15]	0.233		0.450		0.558	

sequence. In the *lstm* model, we set 128 as the number of units. We set the dropout rate using a tuning stage on the development set[14]. Each experiment is repeated 5 times for conducting significance tests (i.e., Student's *t*-test).

6.3 Results and Analysis

Upper- and Lower-bounds: As a first experiment, we assess the upper-bound and the baseline (lower-bound) for the evaluated datasets. The upper-bound, indicated as ↑, is defined as the accuracy of the models on clean data, i.e., REF utterances. Models are also trained and tuned on REF utterances. The baseline (indicated as ↓) is the accuracy that the model achieves on noisy data, i.e., ASR utterances. In this case, ASR utterances are used during both the training and tuning stages. We also tried to use REF utterances for training the model to be tested on ASR utterances, however, we observed that this was generally hurting the performance, due to the data distribution mismatch between the train and test data. Table 3 reports the upper- and lower-bound performance of the two types of neural models in the three tasks. It is clear that there are rather large margins between the baseline and the upper-bound accuracy. The margin is 3.8% for the SST dataset, while it is much higher for the TQ-50 and TQ-6, i.e., 19.6% and 16.1%, respectively. The larger margin in the question type classification tasks can be explained by the higher WER that affects the TQ dataset (see Table 2). Using our phoneme-based approach we aim to reduce this margin.

Finally, we also compare with a recent approach of Lugosch et al. [15], which is an end-to-end speech model that directly operates on the audio signals. We use their pre-trained model with *no unfreezing* setting, as the authors reported that it achieves high performance. We fine tune such model on our training sets. As shown in Table 3, the results achieved by this model are fairly low. A possible explanation is that it was trained on simple instructions and very clean speeches, while our datasets include more elaborated utterances and the audios contain ambient noise.

Impact of Phoneme Embeddings: In the following experiments, we adopt the word embedding models as baselines (i.e., *w-cnn* and *w-lstm*). For each individual task we select the neural architecture achieving the best accuracy in Table 3, i.e., *lstm* for SST and TQ-6, and *cnn* for TQ-50. Table 4 reports the results of these models after activating the phoneme embedding input.

Table 4: Accuracy comparison of pre-trained phoneme embeddings. The bold scores suggests a higher accuracy compared to the model without using phoneme embedding (i.e., *None*) and "*" indicates that the improvement is statistically significant (p<0.05). "†" means that the accuracy is significantly higher than the accuracy provided by the *rnd* embeddings.

	SST	TQ-50	TQ-6
(1) *None*	0.414	0.627	0.743
(2) *rnd*	0.409	**0.648***	**0.762***
(3) $s2s_{asr}$	0.409	**0.653***	**0.765***
(4) $s2s_{dict}$	**0.417**	**0.651***	**0.758**
(5) $p2vc_{asr}$	0.402	**0.643**	**0.759***
(6) $p2vc_{dict}$	0.395	**0.638**	**0.754***
(7) $p2vm_{asr}$	0.399	**0.664***†	**0.746**
(8) $p2vm_{dict}$	0.403	**0.655***	**0.753**
(9) $p2va^{0}_{asr}$	0.408	**0.655***	**0.751**
(10) $p2va^{0}_{dict}$	**0.419***	**0.646***	**0.759***
(11) $p2va_{asr}$	0.408	**0.643**	**0.757***
(12) $p2va_{dict}$	0.407	**0.629**	**0.758***

We observe that phoneme embeddings (rows 2-12 in Table 4), even without pre-training (row 2 in Table 4), generally lead to a better accuracy. The positive impact is more pronounced in TQ-50/6, because, as reported in Table 3, there is more margin for improvement. As the results suggest, using phoneme representations is beneficial across different neural architectures and tasks (i.e., *cnn* for TQ-50 and *lstm* for SST and TQ-6); we argue that the proposed approach is useful for any task operating on noisy ASR input.

Several pre-training models (rows 3-12 in Table 4) lead to better accuracy. $p2vc_{asr}$ and $p2vc_{dict}$ achieve the worst results. As discussed in Section 5, this is expected, since *p2vc* is not designed to capture pronunciation similarities of phonemes. In contrast, the other phoneme pre-training models focus on acoustic aspects, and achieve more competitive results. In particular, the neural network models with $p2va^{0}_{dict}$ embeddings perform significantly better than the baseline models (i.e., *None*) in the three tasks. This indicates

[14]In TQ, the dev set is a random sample of the train set without replacement.

that the acoustic information captured by our models helps neural models to deal with ASR transcription errors. The results for pre-training the phoneme embeddings on the CMU dictionary data, rather then on the ASR generated utterances, are not conclusive as they alternate across different models and tasks.

Table 5: Accuracy of classification models using different inputs. The best scores are bold.

	SST	TQ-50	TQ-6
w	0.414	0.627	0.743
c	0.293	0.549	0.678
p	0.318	0.570	0.702
wc	0.408	0.648	0.756
wp	**0.419**	0.646	**0.758**
wcp	0.405	**0.650**	0.754

Impact of Different Input Types: Next, we examine the impact of using different combinations of inputs (i.e., words, characters and phonemes). For the phoneme embeddings, we use $p2va^0_{dict}$. Table 5 reports the results of this set of experiments. The best models always include phoneme embeddings, indicating that, when dealing with ASR errors, using phoneme representations is more effective than the sole use of word and character representations. Although there is usually high correlation between representing text in terms of its character sequence or its phoneme sequence, our results show that phoneme-based representations are significantly more effective compared to character-based representations.

Finally, we run a complete grid search which explores all 6 input combinations, all 11 phoneme embeddings models and both neural architectures. The models that provided the best accuracy on the development sets are *wp-lstm* using $s2s_{dict}$ phoneme embeddings, *wp-lstm* using $s2s_{asr}$ and *wp-cnn* using $p2va^0_{dict}$ for the SST, TQ-50 and TQ-6 tasks, respectively. Such models achieve 41.7%, 66.7% and 75.9% accuracy on the test sets. This means that they provide 0.3%, 1.9% and 0.7% absolute accuracy improvement w.r.t. the best models not using phoneme embeddings.

Table 6 reports some examples in which the baseline model (i.e., *w-cnn*) cannot predict the correct class due to ASR errors, while the model using phoneme embeddings (i.e., *wp-cnn*) can actually produce the expected output.

7 EXPERIMENTAL EVALUATION ON AMAZON ALEXA DATA

In this section, we evaluate our models using a real-world dataset from the Amazon Alexa virtual assistant. Virtual assistants, e.g., Cortana, Alexa, or Google Assistant, perform a wide range of tasks in different domains, such as *Music*, *Weather*, *Calendar*, etc. Typically, a sentence is first classified into one of the supported domains, then domain dependent intent analysis and slot filling (i.e., entity extraction) are carried out. In our evaluation, we consider live utterances spoken to the top 11 domains of Alexa and we experiment on the domain classification task. To create a complex scenario and effectively test the model robustness to ASR errors, in our datasets we upsample utterances affected by ASR errors. In particular, we apply a stratified sampling procedure with four different strata, each one corresponding to a different average WER. In addition to collecting utterances not affected by ASR errors (i.e., 0 WER), we use a stratum corresponding to 0.33 WER, a value similar to the one observed in the TQ dataset (see Table 2). For the other two strata we select a lower WER, namely 0.24 and a higher WER, namely 0.45. From each stratum we select the same number of samples, therefore the average WER is 0.25[15]. The training and development dataset portions contain 60k and 20k utterances, respectively. In testing, to better study the impact of the ASR errors to the model accuracy we keep separated the data sampled from the four different strata: we have four different test sets with increasing WER, i.e., $\{0.00, 0.24, 0.33, 0.45\}$. Each of the four test sets has 5k utterances. The overall dataset statistics are reported in Table 7, where $Test_{0.24}$ indicates a test set with an average WER of 0.24. For all these datasets (except $Test_{0.00}$), their PER is lower than CER, which is consistent with what we observe in our generated datasets shown in Table 2.

We compared the CNN model which uses only the word embedding channel, with the same model where the phoneme embedding channel is also active (we used the $s2s_{asr}$ phoneme embedding space, as it was performing effectively in the experiments reported in Section 6). We selected the hyperparameters of the two models on the development set (i.e., DEV in Table 7). Table 8 reports the results. All the experiments are repeated five times in order to conduct the statistical significance test and the average accuracy is reported.

Results are consistent with what we observed on the generated data: the phoneme embeddings allow the model to improve its robustness to ASR errors. As expected, the gap between the baseline model and the proposed one increases when ASR errors are more frequent and the accuracy difference becomes significant when the word error rate increases.

8 CONCLUSIONS

In this paper, we explored the usage of phoneme-based representations to make classification models more robust to ASR errors. We proposed several approaches to learn phoneme embeddings capturing pronunciation similarities of phonemes. We then showed how to integrate phoneme embeddings into existing Neural Network architectures, and we demonstrated that their adoptions can improve classification models when dealing with data containing ASR errors.

To support our experiments, we introduced a data generation pipeline that is able to automatically produce noisy ASR transcriptions of textual data. Using this pipeline we created noisy versions of four NLU datasets that we plan to make available to the community.

We experimented the proposed models on these generated datasets, as well as on a real-world dataset from the Alexa virtual assistant. Results are consistent and we show that our models can increase the robustness of NLU models to ASR errors.

As future work, we plan to (*i*) experiment on other tasks such as slot filling or intent classification and (*ii*) extend other state-of-the-art models, e.g., ELMo [25] or BERT [4], in order to allow them to take advantage of phoneme representations.

[15]Note that 0.25 WER does not reflect the real WER of Alexa.

Table 6: Examples of cases in which the usage of phoneme embeddings can correct the prediction.

Example	Gold Label	*w-cnn* prediction	*wp-cnn* prediction
REF: What Canadian city has the largest population	Location	Number	Location
ASR: Well, comedian city has the largest population			
REF: What is the conversion rate between dollars and pounds ?	Number	Description	Number
ASR: What is the conversion rain between dollars and pounds ?			

Table 7: Alexa dataset for domain classification.

Dataset	#utterances	WER	CER	PER
Train	60k	0.25	0.12	0.10
Dev	20k	0.25	0.12	0.10
$\text{Test}_{0.00}$	5k	0	0	0
$\text{Test}_{0.24}$	5k	0.24	0.11	0.10
$\text{Test}_{0.33}$	5k	0.33	0.14	0.12
$\text{Test}_{0.45}$	5k	0.45	0.20	0.17

Table 8: Accuracy results on the Alexa domain classification task. "*" indicates that the improvement is statistically significant (p<0.05).

Model	$\text{Test}_{0.00}$	$\text{Test}_{0.24}$	$\text{Test}_{0.33}$	$\text{Test}_{0.45}$
w-cnn	0.900	0.895	0.900	0.824
wp-cnn	**0.903**	**0.896**	**0.904**	**0.831***

REFERENCES

[1] Alan W Black, Paul Taylor, Richard Caley, and Rob Clark. 1999. The festival speech synthesis system. *University of Edinburgh* (1999).

[2] Yuan-Ping Chen, Ryan Price, and Srinivas Bangalore. 2018. Spoken language understanding without speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. 6189–6193.

[3] Chris Cieri, David Graff, Mark Liberman, Nii Martey, Stephanie Strassel, et al. 1999. The TDT-2 text and speech corpus. In *Proceedings of the DARPA Broadcast News workshop*. 57–60.

[4] 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 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 4171–4186.

[5] Luis Fernando D'Haro and Rafael E Banchs. 2016. Automatic correction of ASR outputs by using machine translation. In *Proceedings of the Annual Conference of the International Speech Communication Association*. 3469–3473.

[6] Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*. 1764–1772.

[7] Ben Hixon, Eric Schneider, and Susan L Epstein. 2011. Phonemic similarity metrics to compare pronunciation methods. In *Proceedings of the Annual Conference of the International Speech Communication Association*. 825–828.

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[9] Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1700–1709.

[10] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1746–1751.

[11] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-Aware Neural Language Models.. In *Proceedings of the Association for the Advancement of Artificial Intelligence*. 2741–2749.

[12] Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the International Conference on Computational linguistics*. 1–7.

[13] Xu Li, Zhiyong Wu, Helen M Meng, Jia Jia, Xiaoyan Lou, and Lianhong Cai. 2016. Phoneme Embedding and its Application to Speech Driven Talking Avatar Synthesis.. In *Proceedings of the International Speech Communication Association*. 1472–1476.

[14] Nut Limsopatham, Oleg Rokhlenko, and David Carmel. 2018. Research Challenges in Building a Voice-based Artificial Personal Shopper-Position Paper. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI*. 40–45.

[15] Loren Lugosch, Mirco Ravanelli, Patrick Ignoto, Vikrant Singh Tomar, and Yoshua Bengio. 2019. Speech Model Pre-Training for End-to-End Spoken Language Understanding. In *Proceedings of the Annual Conference of the International Speech Communication Association*. 814–818.

[16] Laurens van der Maaten and Geoffrey Hinton. 2008. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Advances in neural information processing systems*. 3111–3119.

[19] Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology* 48, 3 (1970), 443–453.

[20] Atsunori Ogawa and Takaaki Hori. 2017. Error detection and accuracy estimation in automatic speech recognition using deep bidirectional recurrent neural networks. *Speech Communication* 89 (2017), 70–83.

[21] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an ASR corpus based on public domain audio books. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*. 5206–5210.

[22] Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the annual meeting on Association for Computational Linguistics*. 271–278.

[23] Thomas Pellegrini and Isabel Trancoso. 2010. Improving ASR error detection with non-decoder based features. In *Proceedings of the Annual Conference of the International Speech Communication Association*. 1950–1953.

[24] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1532–1543.

[25] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2227–2237.

[26] Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. 784–789.

[27] Arup Sarma and David D Palmer. 2004. Context-based speech recognition error detection and correction. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 85–88.

[28] Raphael Schumann and Pongtep Angkititrakul. 2018. Incorporating ASR Errors with Attention-Based, Jointly Trained RNN for Intent Detection and Slot Filling. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*. 6059–6063.

[29] Prashanth Gurunath Shivakumar, Haoqi Li, Kevin Knight, and Panayiotis Georgiou. 2018. Learning from Past Mistakes: Improving Automatic Speech Recognition Output via Noisy-Clean Phrase Context Modeling. *arXiv preprint arXiv:1802.02607* (2018).

[30] Miikka P Silfverberg, Lingshuang Mao, and Mans Hulden. 2018. Sound Analogies with Phoneme Embeddings. *Proceedings of the Society for Computation in Linguistics*, 136–144.

[31] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1631–1642.

[32] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Advances in neural information*

processing systems. 3104–3112.

[33] Yik-Cheung Tam, Yun Lei, Jing Zheng, and Wen Wang. 2014. ASR error detection using recurrent neural network language model and complementary ASR. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*. 2312–2316.

[34] Shubham Toshniwal and Karen Livescu. 2016. Jointly learning to align and convert graphemes to phonemes with neural attention models. In *Proceedings of the IEEE Spoken Language Technology Workshop*. 76–82.

[35] Alexandra Vtyurina, Adam Fourney, Meredith Ringel Morris, Leah Findlater, and Ryen W White. 2019. Bridging Screen Readers and Voice Assistants for Enhanced Eyes-Free Web Search. In *Proceedings of The World Wide Web Conference*. 3590–3594.

[36] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine* 13, 3 (2018), 55–75.