Correcting General Purpose ASR Errors using Posteriors

Sunil Kumar Kopparapu
TCS Research and Innovation - Mumbai
SunilKumar.Kopparapu@TCS.Com

C Anantaram
TCS Research and Innovation - Delhi
C.Anantaram@TCS.Com

Abstract

Speech based interfaces have gained popularity because of the advances in automatic speech recognition (ASR) technology, more recently, triggered by the use of deep neural networks for acoustic and language modeling. However, the performance of any general purpose ASR (gpASR) is poor especially for low resource languages and the performance deteriorate further for natural conversational speech. In this paper, we propose a statistical approach to learn the corrections to enable the use of a general purpose ASR for domain specific use. The proposed idea is based on the observation that there are three types of errors that occur in an ASR output, namely, insertion, deletion or substitution of a word or a phoneme. We propose to model these errors statistically and use these posteriors to develop a scheme that is able to repair the ASR output. The proposed system is able to handle ASR errors spread across lexical words also.

1 Introduction

Speech is the most natural mode of communication to query for answers (Kopparapu, 2014). Use of natural language speech to query for information is gaining practical applicability in our day to day activities. It is also believed that in the next ten years there will be a ten times increase in the number of speech interface we will be facing in our day to day life (Fuhrmann et al., 2017). However recognition of natural language speech is not always 100% accurate even when a state of the art ASR engine is employed for a resource rich language. There are several reasons, the significant among them are the mismatch in the train and the test conditions in terms of the acoustic modeling, the accent, the language, the environment etc.

There have been several attempts to improve the speech recognition accuracies (a) by fine tuning, adapting or learning the acoustic models (AM) to handle the train-test mismatch condition (Mohamed et al., 2012); and (b) by configuring the statistical language models (SLM) so that the perplexity of the search space can be reduced (Kombrink et al., 2012). However, these attempts are either far from producing accurate speech to text conversion especially for natural language speech or make the speech recognition engine constrained to perform for a very specific task or domain. Subsequently, there have been several attempts to post process (Ainsworth and Pratt, 1992; Nishizaki and Sekiguchi, 2006; Bassil and Alwani, 2012) the output of the speech recognition engine to identify and correct the erroneous output.

Most work on ASR error detection and correction has focused on using confidence measures, generally called the log-likelihood score, provided by the speech recognition engine; the text with lower confidence is assumed to be incorrect and subjected to correction (Shi, 2008; Zhou et al., 2005). Such confidence based methods are useful only when we have access to the internals of a speech recognition engine built for a specific domain. As mentioned earlier, use of domain-specific engine requires one to rebuild the interface every time the domain is updated, or a new domain is introduced. As mentioned earlier, our focus is to avoid rebuilding the interface each time the domain changes by using an existing ASR. As such our method is specifically a post-ASR system. A post-ASR system provides greater flexibility in terms of absorbing domain variations and adapting the output of ASR in ways that are not possible during training a domain-specific ASR system (Ringger and Allen, 1996). More recently,

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 283–289

there have been attempts to use a general purpose speech recognition engine and then correct the ASR output using bio-inspired evo-devo (Anantaram et al., 2015b; Anantaram et al., 2015a) and statistical techniques (Anantaram and Kopparapu, 2017) based on features extracted from the reference and the ASR output text. The machine learning based system described in (Anantaram and Kopparapu, 2017) is along the lines of (Jeong et al., 2004) but differs in the sense that they use of multiple features for training the Naive Bayes classifier instead of a single feature (syllable count) for training used in (Jeong et al., 2004) in addition to not using manual alignment between the ASR and reference text. In (Twiefel et al., 2014) the authors address the post correction of a general purpose ASR by using (a) the closeness of phoneme depending on the place and manner of articulation to identify the confusability between the actual and the recognized phoneme and (b) using the front-end and the linguist module of Sphinx. However their experiments seem to suggest that their task is that of aligning the gpASR output text with another sentence from a known finite set of sentences.

In this paper, we use a novel approach to repair the errors produced by a gpASR engine. We do not assume, unlike (Twiefel et al., 2014) the availability of reference pre defined sentences. In a very broad sense we try to model the performance of the gpASR engine for a certain environment and domain which is then used to repair the ASR output for all speech utterances coming from the same environment and domain. The rest of the paper is organized as follows. In Section 2 we formulate the problem and describe an approach to model the ASR for a specific domain and environment and in Section 3 we describe the dataset used to evaluate the proposed approach and give some preliminary results. We conclude in Section 4.

2 Problem Formulation

Let

$$\vec{R} = /r_1 r_2 r_3 \cdots r_m/$$

be a spoken sentence consisting of M words that is input to a speech recognition engine. For example,

$$\vec{R} = \begin{cases} \texttt{/who is the accountable} \\ \texttt{person for manufacturing} \\ \texttt{solutions/} \end{cases}$$

Let the general purpose automatic speech recognition (gpASR) output,

$$\vec{O} = \text{"}\mathtt{o_1 o_2 o_3 \cdots o_n}\text{"}$$

consisting of say N words such that $\vec{O} \neq \vec{R}$. For example,

$$\vec{O} = \begin{cases} \texttt{"who is accountable boston} \\ \texttt{for the men affecting} \\ \texttt{solutions"} \end{cases}$$

which was obtained using Kaldi (Povey et al., 2011) with Fisher acoustic models (gpAM) and language models (gpLM) (Kaldi, 2015). Namely,

$$\vec{R} \longrightarrow \underbrace{\boxed{\text{gpASR}}}_{\uparrow} \longrightarrow \vec{O}$$
$$\boxed{\text{gpLM, gpAM}}$$

Observe that M can be $\lessgtr N$. However, there are only three type of operations that are possible to transform $\vec{O}$ to $\vec{R}$, namely, a word in $\vec{O}$ is either deleted or is inserted or is substituted so that $\vec{O}$ can become identical to $\vec{R}$. Clearly, one insertion, two deletions and two substitutions to $\vec{O}$ could make it identical to $\vec{R}$, namely,

$$\begin{aligned} &\texttt{"who is (}\phi \overset{ins}{\rightarrow} \texttt{the) accountable} \\ &\texttt{(boston} \overset{sub}{\rightarrow} \texttt{person) for} \\ &\texttt{(the} \overset{del}{\rightarrow} \phi\texttt{) (men} \overset{del}{\rightarrow} \phi\texttt{)} \\ &\texttt{(affecting} \overset{sub}{\rightarrow} \texttt{manufacturing)} \\ &\texttt{solutions"} \end{aligned}$$

$$\vec{O}_p \longleftarrow \underbrace{\boxed{\text{word2phoneme}}}_{\uparrow} \longleftarrow \vec{O}$$
$$\boxed{\text{gpLexicon}}$$

Let each word in $\vec{R}$ and $\vec{O}$ be represented by its phonetic equivalent so that

$$\vec{R}_p = r_1 r_2 \cdots r_M$$

and

$$\vec{O}_p = o_1 o_2 o_3 \cdots o_N$$

such that $\{r_i, o_i\} \in I\!P$ where $I\!P$ is the set of phonemes. Note that r_1 in $\vec{R}_p$ is a phoneme while r_1 in $\vec{R}$ is a lexical word. In general there are 39 phones in $I\!P$, namely,

$$I\!P = \left\{ \begin{array}{l} \texttt{a, ae, ah, ao, aw, ay, b, ch,} \\ \texttt{d, dh, eh, er, ey, f, g, hh, ih,} \\ \texttt{iy, jh, k, l, m, n, ng, ow, oy,} \\ \texttt{p, r, s, sh, t, th, uh, uw, v,} \\ \texttt{w, y, z, zh} \end{array} \right\}$$

Subsequently, we can write

$$\vec{O}_p = \begin{cases} \texttt{hh uw ih z dh iy ah k aw n t ah b} \\ \texttt{ah l p er s ah n f r er m ae n y} \\ \texttt{ah f ae k ch er ih ng s ah l uw sh} \\ \texttt{ah n z} \end{cases} \quad (1)$$

and

$$\vec{R}_p = \begin{cases} \texttt{hh uw ih z ah k aw n t ah b ah l b} \\ \texttt{ao s t ah n f r er dh iy m eh n ah} \\ \texttt{f eh k t ih ng s ah l uw sh ah n z}. \end{cases} \quad (2)$$

Notice that even as a phonemic string, $\vec{O}_p$ can be transformed into $\vec{R}_p$ through one of the three operations, namely, deletion, insertion or substitution.

2.1 Computing Posteriors

We define an extension $\mathbb{P}' = \{\mathbb{P}, \phi\}$, where the element ϕ represents a null-phoneme. Given a corpus of $\{\vec{O}_p^i, \vec{R}_p^i\}_{i=1}^K$ pairs (K is large). Note that the elements of $\vec{O}_p$ and elements of $\vec{R}_p \in \mathbb{P}$ and can take one of the 39 unique phones. Represent o and r as $p \in \mathbb{P}$. Now we compute for all $p_i \in \mathbb{P}$

$$P_{sub} = P(p_i \overset{sub}{\rightarrow} p_j) = \frac{\#((p_i \in \vec{O}_p)\&(p_j \in \vec{R}_p))}{\#(p_i \in \vec{O}_p)} \quad (3)$$

where (a) $\#(p_i \in \vec{O}_p)$ is the count of the phone p_i occurring in $\{\vec{O}_p^i\}_{i=1}^K$ and (b) $\#((p_i \in \vec{O}_p)\&(p_j \in \vec{R}_p))$ is the count of the phone p_i which occurs in $\{\vec{O}_p^i\}_{i=1}^K$ when p_j occurs in $\{\vec{R}_p^i\}_{i=1}^K$. Similarly, we find

$$P_{ins} = P(\phi \overset{sub}{\rightarrow} p_j) \quad (4)$$

where p_j occurs in $\vec{R}_p$ but does not occur in $\vec{O}_p$ and

$$P_{del} = P(p_i \overset{sub}{\rightarrow} \phi) \quad (5)$$

where p_i occurs in $\vec{O}_p$ but does not occur in $\vec{R}_p$. Note that we can compute $P_*(p_i \overset{sub}{\rightarrow} p_j)$ for all $p_i, p_j \in \mathbb{P}'$, clearly P_* is a 40×40 matrix. The last column corresponds to P_{del} while the last row corresponds to P_{ins}. We conjecture that these posterior probabilities $P_*(p_i \overset{sub}{\rightarrow} p_j)$ can be used to model the ASR engine (see Figure 1) which in turn can be used to repair the ASR output, namely,

$$\vec{O}_p \longrightarrow \underbrace{\boxed{\text{repair}}}_{\uparrow P_*} \longrightarrow \vec{R}_p$$

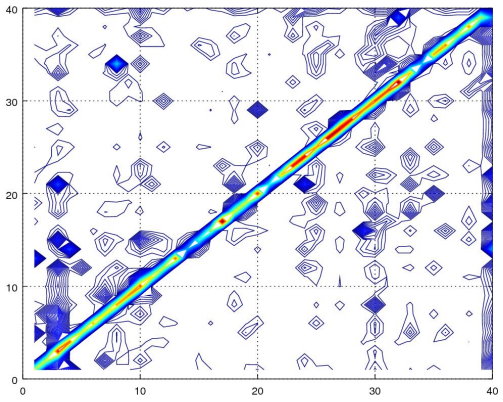

Figure 1: P_* for Kaldi (Povey et al., 2011) with Fischer acoustic and language models. Note that there are higher P_{ins}.

2.2 Repair Approach

Now given an general purpose ASR output, say, $O_1O_2O_3O_4O_5 \cdots O_N$ identify the words O_i which are not part of the domain along the lines of (Anantaram et al., 2015a), using a domain ontology. The domain ontology (dLexicon) helps in identifying all the O_i's such that $O_i \notin$ dLexicon.

Then using the domain lexicon we construct for each O_i the phoneme sequence $o_{i1}, o_{i2}, o_{i3}, o_{i4}, o_{i5}, \cdots, o_{im}$ where each $o_{ij} \in \mathbb{P}$. We then expand the phonemes that require correction as

$$\begin{aligned} expand(o_i, \phi) \quad = \quad & \phi, o_{i1}, \phi, o_{i2}\phi, o_{i3}, \phi, o_{i4}, \\ & \phi, o_{i5}, \phi, \cdots, \phi, o_{im}, \phi \end{aligned} \quad (6)$$

by inserting ϕ between the phonemic representation of O_i. This extended phoneme string is corrected using the posterior (P_*). Now for all identified o_{ij} and $\phi \in \epsilon$, find all $p_k \in \mathbb{P}'$ such that the posterior

$$P_*(o_j \overset{sub}{\rightarrow} p_k) > \tau \text{ and } P_*(\phi \overset{sub}{\rightarrow} p_k) > \tau. \quad (7)$$

We form a lattice ($\mathcal{L}$) using p_k obtained in (7) , namely,

$$\epsilon \longrightarrow \boxed{P_*} \longrightarrow \mathcal{L}$$

We search through this lattice, $\mathcal{L}$, to identify the best set of p_k that results in the highest probability score such that the resulting phoneme string represents a word in the domain, namely,

$$\mathcal{L} \longrightarrow \underbrace{\boxed{\text{search, phoneme2word}}}_{\uparrow \boxed{\text{dLexicon}}} \longrightarrow O'$$

where O' is the corrected word corresponding to O_i. Algorithm 1 captures this approach in greater details.

Algorithm 1 Repair $\vec{O}$.

Given $\vec{O} = O_1 O_2 O_3 O_4 O_5 \cdots O_N$
Given τ, P_*/* Posterior */
Given $cnt = 0$, dLexicon /* Domain Ontology */

for $k = 1, 2 \cdots N$ **do**
  if $O_k \notin$ dLexicon **then**
    $\alpha[cnt++] \longleftarrow O_k$
    /* α contains all words $\notin$ dLexicon */
  end if
end for

for $l = 1, 2, \cdots, cnt$ **do**
  $\vec{o} \leftarrow O_l$ /* Using gpLexicon */
  $\vec{o}_l \leftarrow expand(\vec{o}, \phi)$ /* Expand using (6)*/
  for $p_j \in \vec{o}_l$ **do**
    $knt = 0$
    for $(p_k \in I\!P')$ **do**
      if $(P(p_j \overset{sub}{\rightarrow} p_k) > \tau)$ **then**
        $\mathcal{L}_{j,knt++} = \left\{p_k, P(p_j \overset{sub}{\rightarrow} p_k)\right\}$
      end if
    end for
    /* $\mathcal{L}$ is the lattice */
  end for

  Search through $\mathcal{L}$ to find the repaired $\vec{o}\,'_l$
end for

$O'_l \leftarrow \vec{o}\,'_l$ /* Using Lexicon */
/* O'_l is the corrected word */

3 Experimental Results

3.1 Data setup

Experiments were carried out on a database of 700 spoken utterances by 7 different people, each of them speaking 100 sentences. Each of the 7 speakers were given a set of 100 different queries which they were required to speak into a data collection application built in-house. Each of them spoke 10 queries in a session and the 10 sessions were recorded over a period of one or two days. The spoken utterance was recorded in the wave format with 16 bits resolution and a sampling rate of 16 kHz. These 700 spoken utterances were first converted to text using a general purpose speech recognition engine, in our case Kaldi (Povey et al., 2011) using the Fischer acoustic (Kaldi, 2015) and Fischer language models. We built the gpLexicon using the 700 decoded text outputs using an online tool (CMU, 2017). Note that the general purpose ASR (Kaldi) can output words which need not always be part of the domain Ontology (in our case the domain was related to software industry). In all there were 1426 unique words in the 700 ASR text output (gpLexicon) and there were 372 unique words in the domain lexicon (dLexicon). We divided the 700 into 5 sets of 140 utterances each and carried out a 5 fold validation, namely we used 4 sets, consisting of 560 sentences to compute the posterior and 1 set consisting of 140 sentences to test. In this paper, we present a sample example to demonstrate the effectiveness of our approach.

3.2 Construction of P_*

We constructed the posteriors using the decoded text (example, (1)) and the actual spoken text (example, (2)) after converting both the text strings into phonemes using a phonetic lexicon (we used gpLexicon for ASR output text and dLexicon for the actual spoken text). We aligned the two phonetic strings using edit distance algorithm which came with (Povey et al., 2011). For (1) and (2) we get

"hh, uw, ih, z, dh, (ae $\overset{sub}{\rightarrow}$ ah), (t $\overset{del}{\rightarrow}$ ϕ), ah, k, aw, n, t, ah, b, ah, l, (b $\overset{sub}{\rightarrow}$ p), (aa $\overset{sub}{\rightarrow}$ er), s, (t $\overset{del}{\rightarrow}$ ϕ), ah, n, f, ao, r, (dh $\overset{del}{\rightarrow}$ ϕ), (ah $\overset{del}{\rightarrow}$ ϕ), m, (eh $\overset{sub}{\rightarrow}$ ae), n, (ϕ $\overset{ins}{\rightarrow}$ y), ah, f, (eh $\overset{sub}{\rightarrow}$ ae), k, (t $\overset{sub}{\rightarrow}$ ch), (ih $\overset{sub}{\rightarrow}$ er), (ϕ $\overset{ins}{\rightarrow}$ ih), ng, s, ah, l, uw, sh, ah, n, z"

which are used to compute the posterior as mentioned earlier. As seen there are 7 substitutions, 3 deletions and 2 insertions. A sample posterior P_* is shown in Figure 1 as a contour plot. The x-axis represents the phonemes (from the actual text) while the y-axis is the phonemes (in the words recognized by gpASR).

3.3 Sample Repair

As an example the general purpose ASR returned

```
"who is that accountable
boston for the men affecting
solutions"
```

when

```
/who is the accountable
person for manufacturing
solutions/
```

	"boston"												
		b		ao		s		t		ah		n	
1	$\boldsymbol{\phi}$	b	$\boldsymbol{\phi}$	ao	$\boldsymbol{\phi}$	**s**	$\boldsymbol{\phi}$	t	$\boldsymbol{\phi}$	**ah**	$\boldsymbol{\phi}$	**n**	$\boldsymbol{\phi}$
2	ah	**p**	ah	ah	ah	sh	ah	$\boldsymbol{\phi}$	ah	$\boldsymbol{\phi}$	ah	ϕ	ah
3	ih	ϕ	ih	ϕ	ih	ϕ	ih	r	ih	er	ih	l	ih
4	k	r	k	**er**	k	z	k	ah	k	r	k	ah	k
5	t	w	t	aa	t	ah	t	er	t	aa	t	k	t
6	l	l	l	ow	l	er	l	l	l	ae	l	r	l
7	r	ah	r	iy	r	ey	r	d	r	ih	r	ae	r
8	s	dh	s	sh	s	r	s	th	s	eh	s	ih	s
9	d	d	d	p	d	ih	d	dh	d	ey	d	m	d
10	w	t	w	v	w	ch	w	eh	w	uw	w	ng	w
		p		**er**		**s**				**ah**		**n**	
	"person"												

Table 1: Lattice $\mathcal{L}$ constructed using the posterior P_*, used to correct 'b ao s t ah n' to 'p er s ah n'. The phones marked in bold are the ones that are picked during the lattice search.

was given as the audio input to the gpASR. The output text "boston" and "men affecting" are not part of the domain ontology (dLexicon).

We first converted the word "boston" into its phoneme equivalent using the general purpose phonetic lexicon (gpLexicon), namely 'b ao s t ah n', then using the posterior P_* we constructed the lattice $\mathcal{L}$ (see Table 1) which shows the top 10 phonemes that could replace the phoneme output by the ASR based on the posterior P_*. For example, the column associated with b shows that $P(\texttt{b} \overset{sub}{\rightarrow} \texttt{p}) \geq P(\texttt{b} \overset{del}{\rightarrow} \phi) \geq P(\texttt{b} \overset{sub}{\rightarrow} \texttt{r})$ etc. Note that we have captured the possible insertion by appending ϕ between the phones, namely, ϕ b ϕ ao ϕ s ϕ t ϕ ah ϕ n (6). Now traversing the lattice $\mathcal{L}$ in Table 1 from left to right we can get $\boldsymbol{\phi}$ **p** $\boldsymbol{\phi}$ **er** $\boldsymbol{\phi}$ **s** $\boldsymbol{\phi}$ $\boldsymbol{\phi}$ $\boldsymbol{\phi}$ **ah** $\boldsymbol{\phi}$ **n** $\boldsymbol{\phi}$ which is nothing but 'p er s ah n' which is part of our domain.

A similar repair mechanism described in Algorithm 1 helps in correcting "men affecting" to the domain word "manufacturing" as shown in Figure 2. As a result the general purpose ASR output

"who is that accountable boston for the men affecting solutions"

is corrected to

"who is that accountable person for the manufacturing solutions"

Note that this is not exactly what was spoken, namely,

/who is the accountable person for manufacturing solutions/

Notice that the repair mechanism is unable to correct "that" to "the" and delete "the" however the repair mechanism is able to correct the non-domain words by identifying phonetically equivalent words in the domain. In this example and all our experiments we choose τ in a manner that we had the top 10 phonemes to form the lattice $\mathcal{L}$.

Also notice that because the repair mechanism operates in the phoneme space it is able to operate even if the error is spread across words (example, "men affecting" $\longrightarrow$ "manufacturing"). This example sentence clearly demonstrates the ability of the proposed repair approach to correct the output of a general purpose automatic speech recognition engine based on the computed posteriors.

We measured the accuracy of improvement in the ASR output, by looking at the edit distance between ($\vec{O}$, $\vec{R}$) and the edit distance between ($\vec{O}'$, $\vec{R}$). Here $\vec{R}$ is the reference out (perfect ASR output), while $\vec{O}$ is the output of the general purpose ASR (in our case Kaldi ASR output) and $\vec{O}'$ is the repaired output based on Algorithm 1. It is observed that the $dis(\vec{O}', \vec{R}) \leq dis(\vec{O}, \vec{R})$. Namely, the repaired output was always closer to the reference that the output of the general purpose ASR engine ($\vec{O}$).

"men"							"affecting"													
	m		eh		n		ah		f		eh		k		t		ih		ng	
ϕ	**m**	ϕ	eh	ϕ	**n**	ϕ	**ah**	ϕ	**f**	ϕ	eh	ϕ	**k**	ϕ	t	ϕ	**ih**	ϕ	**ng**	ϕ
ah	ϕ	ah	ah	ah	ϕ	ah	ϕ	ah	v	ah	ah	ah	ϕ	ah	ϕ	ah	ey	ah	n	ah
ih	n	ih	**ae**	ih	l	ih	er	ih	ah	ih	**ae**	ih	t	ih	r	ih	ah	ih	ϕ	ih
k	l	k	ϕ	k	ah	k	r	k	ϕ	k	ϕ	k	ah	k	ah	k	eh	k	d	k
t	ih	t	ih	t	k	t	aa	t	p	t	ih	t	f	t	**er**	t	ϕ	t	v	t
	m		**ae**		**n**		**ah**		**f**		**ae**		**k**		**er**		**ih**		**ng**	
"manufacturing"																				

Table 2: Lattice $\mathcal{L}$ constructed using the posterior P_*, used to correct "men affecting".

4 Conclusions

Speech based applications are being increasingly deployed for self help in enterprises. However for resource deficient languages (including Indian English) the performance of ASR engine is poor especially for natural spoken utterances. While there are two ways of getting over the poor performance of ASR engine, namely (a) fine tuning the AR engine in terms of acoustic and language models; thereby making the ASR engine domain specific or (b) using a general purpose ASR engine and then correcting the ASR output using domain ontology and in some way modeling the ASR behavior. The advantage of the second approach is that of being able to use a readily available state of the art ASR engine without having to build a unique speech recognition engine for every application. In this paper, we approach the problem of correcting the general purpose ASR output using posteriors. The main contribution of this paper is the formulation of a posterior approach to repair the output of a general purpose ASR engine as depicted in detail in Algorithm 1. We also showed that the approach is able to repair the errors that might be spread across lexical words.

5 Acknowledgements

The authors would like to thank Chirag Patel who assisted in performing some experiments in the earlier stages of this work.

References

W.A. Ainsworth and S.R. Pratt. 1992. Feedback strategies for error correction in speech recognition systems. *International Journal of Man-Machine Studies*, 36(6):833 – 842.

C. Anantaram and Sunil Kumar Kopparapu. 2017. Adapting general-purpose speech recognition engine output for domain-specific natural language question answering. *CoRR*, abs/1710.06923.

C. Anantaram, Rishabh Gupta, Nikhil Kini, and Sunil Kumar Kopparapu. 2015a. Adapting general-purpose speech recognition engine output for domain-specific natural language question answering. In *Workshop on Replicability and Reproducibility in Natural Language Processing: adaptive methods, resources and software at IJCAI 2015*, Buenous Aires.

C Anantaram, Nikhil Kini, Chirag Patel, and Sunil Kopparapu. 2015b. Improving ASR recognized speech output for effective NLP. In *The Ninth International Conference on Digital Society ICDS 2015*, pages 17–21, Lisbon, Portugal.

Youssef Bassil and Mohammad Alwani. 2012. Post-editing error correction algorithm for speech recognition using bing spelling suggestion. *CoRR*, abs/1203.5255.

CMU. 2017. The CMU pronouncing dictionary. http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

Ferdinand Fuhrmann, Anna Maly, Christina Leitner, and Franz Graf. 2017. Three experiments on the application of automatic speech recognition in industrial environments. In Nathalie Camelin, Yannick Estève, and Carlos Martín-Vide, editors, *Statistical Language and Speech Processing - 5th International Conference, SLSP 2017, Le Mans, France, October 23-25, 2017, Proceedings*, volume 10583 of *Lecture Notes in Computer Science*, pages 109–118. Springer.

Minwoo Jeong, Byeongchang Kim, and G Lee. 2004. Using higher-level linguistic knowledge for speech recognition error correction in a spoken q/a dialog. In *HLT-NAACL special workshop on Higher-Level Linguistic Information for Speech Processing*, pages 48–55.

Kaldi. 2015. Kaldi fisher english. http://kaldi-asr.org/downloads/build/2/sandbox/online/egs/fisher_english/.

Stefan Kombrink, Tomas Mikolov, Martin Karafiát, and Lukás Burget. 2012. Improving language models for ASR using translated in-domain data. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2012, Kyoto, Japan, March 25-30, 2012*, pages 4405–4408. IEEE.

S.K. Kopparapu. 2014. *Non-Linguistic Analysis of Call Center Conversations*. SpringerBriefs in Electrical and Computer Engineering. Springer International Publishing.

A. Mohamed, G.E. Dahl, and G. Hinton. 2012. Acoustic modeling using deep belief networks. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):14 –22, jan.

Hiromitsu Nishizaki and Yoshihiro Sekiguchi. 2006. Word error correction of continuous speech recognition using web documents for spoken document indexing. In Yuji Matsumoto, RichardW. Sproat, Kam-Fai Wong, and Min Zhang, editors, *Computer Processing of Oriental Languages. Beyond the Orient: The Research Challenges Ahead*, volume 4285 of *Lecture Notes in Computer Science*, pages 213–221. Springer Berlin Heidelberg.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, number Idiap-RR-04-2012, Rue Marconi 19, Martigny, December. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

E.K. Ringger and J.F. Allen. 1996. Error correction via a post-processor for continuous speech recognition. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 427–430 vol. 1, May.

Yongmei Shi. 2008. *An Investigation of Linguistic Information for Speech Recognition Error Detection*. Ph.D. thesis, University of Maryland, Baltimore County, October.

Johannes Twiefel, Timo Baumann, Stefan Heinrich, and Stefan Wermter. 2014. Improving domain-independent cloud-based speech recognition with domain-dependent phonetic post-processing. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1529–1535. AAAI Press.

Lina Zhou, Jinjuan Feng, A. Sears, and Yongmei Shi. 2005. Applying the naive bayes classifier to assist users in detecting speech recognition errors. In *System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 183b–183b, Jan.