Disambiguation and Error Resolution in Call Transcripts

Jordan Hosier*†, Vijay K. Gurbani*, Neil Milstead*
*Vail Systems, Inc.
{jhosier,vgurbani,nfnm}@vailsys.com
†Northwestern University
jhosier@u.northwestern.edu

***Abstract*—Ambiguity is inherent to human language and poses a unique challenge both to human listeners as well as natural language processing (NLP) systems. Ambiguity is understood as a type of uncertainty which allows for more than one plausible interpretation of utterances. Ambiguity can introduce problems for NLP systems designed to, for example, execute machine translation, determine sentiment, and perform automatic speech recognition (ASR). We seek to identify and resolve mis-transcriptions that arise from phonetic ambiguity and degraded acoustic signals in 87,000 call transcripts. We first present an alignment algorithm which identifies mis-transcriptions generated by ASR systems when compared against verified human transcriptions. This method not only allows for a general evaluation of ASR performance but also highlights specific areas of difficulty for such systems (e.g. "considerate" vs. "consider it"). We further present an error resolution algorithm which, given a mis-transcribed word, uses contextual cues to suggest a more likely, phonetically similar word. This work has the potential to not only evaluate existing ASR systems, but also to immediately improve their performance.**

***Index Terms*—homophone disambiguation, automatic speech recognition, text processing**

I. Introduction

Ambiguity is understood as a type of uncertainty which allows for more than one plausible interpretation of an utterance. Despite the potential for processing difficulty, ambiguity is ubiquitous in language [1]. Context plays an important role in the process of disambiguation—what is ambiguous in one context may not be ambiguous in another. Take, for example, the exchange in (1).

(1) "Mine is a long and sad tale!" said the Mouse, turning to Alice and sighing.
"It is a long tail, certainly," said Alice, looking with wonder at the Mouse's tail, 'but why do you call it sad?"
(Lewis Carroll, Alice's Adventures in Wonderland)

The exchange in (1) demonstrates a particular kind of phonetic ambiguity - homophones, which are defined as words having the same pronunciation but different meanings and/or spellings. Ambiguity can also extend beyond the word level. Take, for example, the two utterances in (2).

(2) a. The stuffy nose can lead to problems.
b. The stuff he knows can lead to problems.

978-1-7281-0858-2/19/$31.00 ©2019 IEEE

In conversational speech, pauses between discrete lexical items can be imperceptible or non-existent, which can make word segmentation difficult. The phrases "stuffy nose" and "stuff he knows" are oronyms—sequences of words that sound the same but have different meanings. Such ambiguities not only present difficulties for humans, but are particularly challenging in Natural Language Processing (NLP) settings, such as Automatic Speech Recognition (ASR). Ambiguities can result in errors in ASR generated text, which in turn affect the accuracy of downstream textual analyses. While ambiguities remain a challenge for NLP systems, there exists little work targeted at identifying and resolving homophone confusion in English (but see [2] for related approaches in Chinese).

We present an algorithm that, given an error in an ASR generated transcript, suggests higher probability, phonetically similar replacements based on training a corpus of 87,000 ASR generated customer survey transcripts. The algorithm is constructed from the following building blocks: (1) a word embedding dataset for all the words contained in the training corpus, (2) a dataset containing all tri-grams present in the corpus and their frequencies, and (3) a Phonetic Encoding algorithm which is used to determine phonetic similarity between lexical items.

The current work advances efforts in big data research in the following two ways. First, we identify potential areas of difficulty in ASR engines introduced by homophones and oronyms, and second, we present a novel error resolution algorithm that uses a number of filters to increase the accuracy of the text generated by the ASR. Our approach allows for more robust language modeling for large text corpora generated by an ASR engine. This work also demonstrates the need for a human in the loop approach, as our algorithm requires human-generated transcripts to identify mis-transcriptions.

II. Background

A. Motivation

Recent advances in the field of speech processing have led to improved performance of ASR systems. Nonetheless, such systems are far from perfect and often produce errors [3]. These ASR errors are a result of both intrinsic ambiguities as well as variation present that is not properly accounted for in the system's speech models (idiosyncratic talker properties,

differing accents, etc.) Thus, transcription errors are a result of ambiguous speech regions that contain acoustical and/or contextual confusability [4]. Recognition errors impact the validity of analyses and applications that rely on the resulting text, such as machine translation, information retrieval, etc.

It is functionally impossible to resolve all potentially ambiguous utterances without incorporating infinite world knowledge. Nonetheless, traditional lexicons in NLP settings assume that words can be defined by an enumerable set of parameters [5]. As a result, when NLP tasks are faced with ambiguity, an additional effort of disambiguation is needed.

For example, in an ASR setting, if the system encounters an ambiguous word (such as a word that has homophonous competitors), the system must select the most appropriate word available based on its perception of the word that was uttered - a process driven by matching phonetic cues with contextual information. This requires that such systems specify the contexts in which words are likely to appear, which depends greatly on the context in which such systems are trained and deployed.

To satisfactorily extract insight from ASR generated data, it is important that the content of such data be reliable. Take, for example, a speech-to-text task in which the resulting text is subjected to some further analysis, such as translation. If the word "seller" is uttered, it is possible that the ASR system will erroneously interpret the token as its homophone, "cellar" as opposed to the intended "seller". Thus, subsequent analyses performed on the generated text can potentially yield incorrect results. Developing a system that can perform ambiguity resolution, in this context, allows for more accurate text which can be used to deploy more robust language models.

B. Related Work

Our approach to ASR error detection and correction relates to other recent work in word embeddings for ASR detection, such as Ghannay et al. [6] who evaluated the performance of acoustic word embeddings to detect homophones and errors and label each word in a corpus as either *error* or *correct* using a feed-forward neural network model. This method, much like our own, included features that contain information about high frequency tri-grams, word embeddings, and lexical features (such as word length). Unlike Ghannay et al. [6], who use acoustic word embeddings to identify homophones, we use a combination of phonetic encoding and traditional word2vec embeddings to approximate both semantic and phonetic similarity of words.

There have been other, similar human-in-the-loop approaches to ASR error resolution, including Huggins-Dains and Rudnicky [7] which presented a graphical interface for correcting errors in the output of a speech recognizer. Through the application, users can "pull apart" regions to reveal word suggestions similar to the original ASR word suggestion. We suggest a similar approach, wherein users input a word and the application generates a rank-ordered list of similar words. Thus, much like [7] we present an ASR post-processing algorithm which processes and subsequently improves upon the output of an ASR engine.

III. Problem Formulation

Given some ASR generated text, such as the text in (3a), the algorithm should suggest higher probability, similar sounding alternatives for any mis-transcribed words - in this case, "consider it." The result would be a correct transcript, as in (3b).

(3) a. He was very kind and very **consider it**.
b. He was very kind and very **considerate**.

We intend to target cases such as (3) in the following ways:

- Build and evaluate an algorithm capable of resolving homophone confusion and other mis-transcriptions in ASR text derived from customer survey responses.
- Identify areas of difficulty for the ASR engine.
- Produce more accurate transcripts.

IV. Methodology

A. Data

The subsequent analyses are based on textual data derived from customer call surveys. The data, in its original form, consisted of .wav files, which were then used as input for a commercial ASR Engine. Our analyses began with the resulting textual data which consists of 87,000 call surveys varying in length from one word to approximately 300 words.

A subset of this data, which we call the verification data, was also transcribed by human listeners. The verification set consisted of a 1,900 element subset of the same 87,000 customer survey responses used to generate ASR generated transcripts. We use this verification data as an accuracy benchmark, both to highlight errors in the ASR transcripts as well as to verify the accuracy of our algorithm. Resulting is a dataset of 1,900 surveys for which we have both human verified transcripts as well as ASR generated transcripts (as in Table I). The audio files consisted of customers leaving voice survey responses about their experience with the call center agent. The survey was free form, where the customer could respond in any way they saw fit.

B. Alignment

To highlight discrepancies between the verified transcripts and the ASR generated transcripts, we estimated the best alignment of the two transcripts for each survey. We estimate the best alignment using the Needleman-Wunsch Algorithm [8], which finds the minimal set of insertions, deletions, and substitutions needed to transform the verified transcript into the ASR transcript. From these alignments, we extract the implied set of word errors made. For example, if "informal to" was transcribed instead of the correct utterance, "informative", the resulting error would be "informal to" $\rightarrow$ "informal".

The resulting error dataset contained 2,648 unique error types with a total of 6,438 errors made. There were an average of 4.4 errors per survey transcript. The highest frequency error is the deletion of filler words with the most frequent being the deletion of "uh", which occurred 705 times. The second most

Survey Number	ASR Generated Transcript	Human Verified Transcript
2265	He was tentative and quick.	He was attentive and quick.
2266	She was very nice very polite very informal to.	She was nice very polite very informative.
2267	She was very healthy.	She was very helpful.

TABLE I: Example data for the 1,900 call survey transcripts.

common error type is the deletion of grammatical endings (i.e. "she's" $\rightarrow$ "she", "helped" $\rightarrow$ "help"). We do not consider these types of errors for resolution as filler words do not contribute to the interpretation of the utterances. We ignore errors related to grammatical endings as these morphemes result in different tenses of the same word. Often times such words are reduced to a core root as a data prepossessing step. Thus, for our purposes, we consider only substitution errors (i.e. "card" $\rightarrow$ "car", "bill" $\rightarrow$ "bail") that do not include errors related to grammatical endings.

C. The Algorithm

The algorithm takes as input a word embedding for each word in the transcript corpus, counts for all tri-grams that appear in the corpus, the mis-transcribed word, and surrounding context words. For example, in the sentence in (3), the context words are "and very" and "it" while the mis-transcribed word is "consider." The following sections further detail the various components of the algorithm.

1) Word Embedding: Word embeddings are used to identify similarities between words in a corpus by predicting the co-occurrence of words. Word embedding models can complete analogies such as "Man is to woman as king is to queen" [9]. Word embeddings are created by identifying the words that occur within a "Context Window." Word embedding models then calculate how often each word occurs next to every other word within the context window, after normalizing for overall word frequency. We built word embeddings based on context windows with a length of eight words. The resulting data set allows us to calculate vector similarity of each word in the corpus. This measure of similarity is analogous to a measure of word sense similarity. That is, it measures similarity of words based on how often they occur in similar environments.

2) N-Grams: Calculating word probability is an essential step towards identifying words in ambiguous environments. N-grams allow us to assign a probability to each possible next word in a sequence. A gram is a unit of text of any length and *n* specifies how many grams are used in the calculation. For our purposes, a gram is a word. The algorithm uses a particular type of n-gram, known as a tri-gram, or a chunk of text that contains three words.

In this context, the probability of any given word *w* can be computed given some history *h*.

$$P(w|h) \tag{1}$$

Suppose the history *h* is "and very" and we want to know the probability that the next word is "considerate", as in (3b).

$$P(considerate|and\ very) \tag{2}$$

We estimate this probability by computing relative frequency counts. In the ASR transcript data, we compute the number of times "and very" occurs and the number of times it is followed by "considerate." Formally, we calculate the following: out of the times we saw *h*, how many times was it followed by *w*, as follows:

$$P(considerate|and\ very) = \frac{C(considerate)}{C(and\ very)} \tag{3}$$

In Equation 3, *C()* is the frequency count function. The result is the probability of "considerate" appearing in this context.

3) Phonetic Encoding: Word Embeddings allow us to calculate similarity based on word co-occurrence. Another important measure of similarity for homophone resolution is phonetic similarity. We wish to do a homophone lookup on mis-transcribed words to see if there exists a homophonous word in the corpus with a higher probability in context.

We do so using a phonetic encoding algorithm. Given a word or string, the algorithm derives an encoding based on the sounds contained in the word. Here, we use the **match rating approach** - a phonetic encoding algorithm developed for the comparison of homophonous names [10]. Such algorithms are far from perfect, but serve as a good approximation and avoid having to phonetically transcribe every word in the corpus. The result of the application of this algorithm can be seen in (4), as applied to "consider it" and "considerate."

(4) a. consider it = CNSDR T.
b. considerate = CNSDRT

Given two phonetic encodings, we compare the two strings and calculate the edit distance between the two tokens. In this case, we see that the phonetic encoding of "consider it" and "considerate" are exact matches and thus have an edit distance of 0 when the space is removed. The decision to remove the space is a reasonable one, as spaces between words, while present orthographically, are rarely realized in conversational speech.

4) Parameters: The model has several parameters which are used to filter the initial candidate word list for high probability, similar words (both in terms of vector similarity and phonetic similarity). The initial candidate word list is comprised of all words in the corpus that appeared after the previous two context words. The parameters are as follows: token length, probability, vector similarity, and phonetic distance.

Prior to any filtering, the algorithm checks the candidate list for any exact homophones. The phonetic encoding of

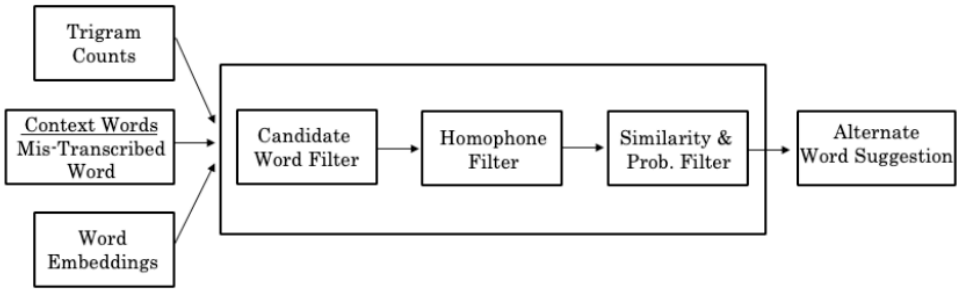

Fig. 1: Graphical model of the data pipeline and algorithm components.

the mis-transcribed word, as well as the concatenation of the mis-transcribed word and the following word, are compared against the phonetic encoding of every candidate. If there exits a highly probable exact homophone match, the match is suggested in place of the mis-transcribed word.

Token length is a parameter used to verify that the suggested replacement words are a reasonable length when compared to the mis-transcribed word. This parameter is important because some of the highest frequency words are function words and such words are high probability in many contexts. Further, these words can have overlapping sounds with the mis-transcribed words. We want to ensure that short, function words are not suggested in place of content words. For this reason, we filter the candidate list for words that are at least *.6x* the length of the mis-transcribed word.

We also filter for words that are *probability* $\geq$ .001 and *vector similarity* $\geq$ .001. These values can be thought of as hyper parameters and were approximated using a grid search. The resulting word list is a refined candidate list. We then sort the word suggestions based on three parameters: probability, vector similarity, and phonetic distance.

An example of the structure of a candidate word list is displayed in Table II, for the test case "and very consider it." Although the correct candidate, "considerate" does not have the highest probability or vector similarity, it has a phonetic distance of zero and is thus selected as the suggested replacement for the mis-transcribed word.

Candidate Token	Probability	Vector Similarity	Phonetic Distance
professional	0.053	0.00307	5
very	0.053	0.0024	5
concise	0.0013	.0018	3
considerate	0.0016	0.0028	0

TABLE II: Candiate list for "and very **consider it**," where the selected candidate is **considerate**.

V. Evaluation Results

The algorithm is deployed in the form of a Shiny Web Application[1]. In its current form, the app requires user interaction. The user inputs the previous two words, the candidate word, and the following word. It then generates word suggestion(s). As Figure 2 demonstrates, the algorithm is successful when tested on the "consider it" vs. "considerate" example.

We manually tested the app on a subset of 500 surveys for which we had verification data. Among these transcripts, the algorithm successfully resolved 60.3% of the errors of interest (95% CI [.469, .738]). Recall that we are not interested in resolving errors that involve the deletion of filler words or the dropping of grammatical endings. Table III shows examples of areas where the algorithm successfully recovered the correct utterance.

The algorithm successfully suggests alternates beyond instances of homophone confusion. The algorithm not only

[1]The Shiny Web App can be accessed here: https://jrhosier.shinyapps.io/Transcript_App/

ASR	Verified	Algorithm's Error Resolution
She was very nice very polite very **informal to**	She was very nice very polite very **informative**	informal to → informative
She was very **healthy**	She was very **helpful**	healthy → helpful
I was really **sad** thank you	I was really **satisfied** thank you	sad → satisfied
She was very **lean in** to work with me	She was very **willing** to work with me	lean in → willing
She was **person** to talk to and that's it	She was **pleasant** to talk to and that's it	person → pleasant

TABLE III: Examples of the algorithm's error resolutions.

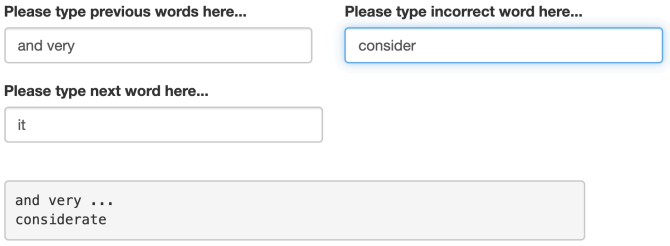

Fig. 2: Preview of the interface of the web app.

searches for probable homophones, but also suggests other probable words with overlapping phonemes. This flexibility allows the algorithm to catch instances such as "healthy" → "helpful" as these two words, while not homophonous, do share sounds. There are many such instances where an algorithm that relied solely on homophony would fail.

There are, however, instances of replacement errors that the algorithm cannot resolve. Those errors largely fall in four categories: 1) low probability replacement words, 2) highly dissimilar replacement words, 3) incomplete sentences, and 4) extremely dissimilar words. Examples of each of these errors are displayed in Table IV.

The algorithm at times fails to suggest words that are lower probability than the mis-transcribed word, particularly when the words are not exact homophones as in the case of "efficient" → "sufficient". The word "efficient" is high frequency across the dataset, and is highly probable in this context. Thus, in these instances, the algorithm suggests a number of other, lower probability candidates and does not always suggest the correct alternative.

The algorithm also has difficulty when the correct replacement word and the mis-transcribed word are highly dissimilar. In the case of "make a payment/human", the training data did not have enough instances of "make a payment" to have this tri-gram be high enough probability to suggest "payment" in this case.

The third example in Table IV demonstrates a case where the context is ill-defined due to the use of a sentence fragment. Such instances are also difficult for the algorithm to capture. Finally, there exist some instances where the ASR transcription is almost entirely incorrect. In these instances, where the context is not only ill-defined, but also inaccurate, the algorithm is unable to generate word suggestions.

VI. Conclusion

This work has three primary contributions in BigData efforts. First, we present an objective evaluation of an ASR engine through the alignment of ASR generated transcripts with verified human transcriptions. The result is a well-defined vocabulary of ASR errors which provides insight into areas of difficulty for the ASR engine and allows for the evaluation of our algorithm.

Our algorithm not only successfully performs homophone resolution, but a more general error resolution based on word vector similarity, probability in context, and phonetic similarity. This algorithm can be implemented such that error resolution might be performed automatically or with the assistance of human approval or selection. In this way, our algorithm can be used in a speech-to-text pipeline to yield more reliable transcripts.

In the future, we intend to expand the algorithm to include automatic error detection. The current work serves as a first step towards automatically identifying and resolving errors in transcripts. A necessary next step involves identifying likely errors in these transcripts such that the algorithm can suggest likely replacements.

We would also like to expand the training set to include more data. We are also interested in testing the algorithm's performance in highly constrained contexts. For instance, if the transcript data is exclusively made up of individuals paying their bills over the phone, could this constrained context bolster the algorithms ability to generate correct replacements? This algorithm can be used as a complement to the output of any ASR engine. If the training vocabulary is relatively well-defined, the algorithm has the potential to increase the accuracy of transcripts generated by an open source ASR engine.

Acknowledgment

We acknowledge the effort of Neetu Gurbani, Ann Smetana, and Benjamin Burkhardt for generating transcripts from audio files used in our work.

References

[1] Davis, Matthew H., William D. Marslen-Wilson, and M. Gareth Gaskell. "Leading up the lexical garden path: Segmentation and ambiguity in spoken word recognition." Journal of Experimental Psychology: Human Perception and Performance 28.1 (2002): 218.

ASR	Verified	Correct Resolution	Algorithm Resolution
He was **efficient**	He was **sufficient**	efficient → sufficient	efficient → patient
She helped me make a **human**	She helped me make a **payment**	human → payment	human → change
Good customer service **detective**	Good customer service **attentive**	detective → attentive	detective → active
Male oh **male ambulance**	**Billing problems** oh **billing problems**	male → billing problems male ambulance → billing problems	NA

TABLE IV: Examples of transcript errors that the algorithm does not resolve.

[2] Chang, Chao-Huang. "Corpus-based adaptation mechanisms for Chinese homophone disambiguation." VERY LARGE CORPORA: ACADEMIC AND INDUSTRIAL PERSPECTIVES. 1993.

[3] Vasilescu, Ioana, Martine Adda-Decker, and Lori Lamel. "Cross-lingual studies of ASR errors: paradigms for perceptual evaluations." LREC. 2012.

[4] Vasilescu, Ioana, et al. "Cross-lingual study of ASR errors: on the role of the context in human perception of near-homophones." Twelfth Annual Conference of the International Speech Communication Association. 2011.

[5] McShane, Marjorie, Sergei Nirenburg, and Stephen Beale. "An NLP lexicon as a largely language-independent resource." Machine Translation 19.2 (2005): 139-173.

[6] Ghannay, Sahar, et al. "Acoustic Word Embeddings for ASR Error Detection." INTERSPEECH. 2016.

[7] Huggins-Daines, David, and Alexander I. Rudnicky. "Interactive asr error correction for touchscreen devices." Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Demo Session. Association for Computational Linguistics, 2008.

[8] Needleman, Saul B., and Christian D. Wunsch. "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.

[9] Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig. "Linguistic regularities in continuous space word representations." Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2013.

[10] Moore, Gwendolyn B. Accessing individual records from personal data files using non-unique identifiers. Vol. 13. US Department of Commerce, National Bureau of Standards, 1977.