Improving Spoken Dialogue Understanding Using Phonetic Mixture Models

William Yang Wang* and **Ron Artstein** and **Anton Leuski** and **David Traum**
Institute for Creative Technologies, University of Southern California
12015 Waterfront Drive, Playa Vista, CA 90094-2536, USA

Abstract

Augmenting word tokens with a phonetic representation, derived from a dictionary, improves the performance of a Natural Language Understanding component that interprets speech recognizer output: we observed a 5% to 7% reduction in errors across a wide range of response return rates. The best performance comes from mixture models incorporating both word and phone features. Since the phonetic representation is derived from a dictionary, the method can be applied easily without the need for integration with a specific speech recognizer. The method has similarities with autonomous (or bottom-up) psychological models of lexical access, where contextual information is not integrated at the stage of auditory perception but rather later.

Introduction

A standard architecture for spoken dialogue systems employs two-tiered processing: An Automatic Speech Recognizer (ASR) transforms the user's speech into a string of words, and then a Natural Language Understanding (NLU) component extracts meanings from these words. This architecture represents an efficient way to tackle the problem of understanding human speech by splitting it into two manageable chunks. However it comes at a cost of an extremely narrow bandwidth for communication between the components: often the only information that passes from the speech recognizer to the NLU is a string of words, and typical NLU components use only word-level features (Litman et al. 2009; Raymond and Riccardi 2007; Walker, Wright, and Langkilde 2000). If the ASR output string is deficient then the NLU will experience difficulties which may cause it to ultimately misunderstand the input. The most straightforward way to address this issue is to improve ASR accuracy, and in the long term, perfect or near-perfect ASR may make the NLU problem for speech systems much more straightforward than it currently is. In the meantime, however, we need to find ways that allow NLU better recovery from speech recognition errors.

This paper addresses a particular kind of deficiency – speech recognition errors in which the ASR output has a different meaning than the actual speech input, but the two strings of words are phonetically similar. An example (taken from the experimental data described in the next section) is the question "Are you married?", which in one instance was recognized as "Are you Mary?". The particular conversational character which received this question cannot understand the word "Mary", and if he could he would probably give an inappropriate response. The character does know that he is quite likely to be asked if he is married; but since he is not aware that "Mary" and "married" sound similar, he cannot make the logical leap and infer the intended question. Such confusions in ASR output are very common, with varying levels of phonetic similarity between the speech input and ASR output. Some more subtle examples from the data include "Are all soldiers deployed?" misrecognized as "Are also just avoid", and "just tell us how you can talk" misrecognized as "does tell aside can tell".

Speech recognizers typically use language models to encode information about expected outputs, but these cannot fully eliminate this kind of close phonetic deviation without severely limiting the flexibility of expression that users can employ. A typical response to the problem is to integrate some knowledge available to NLU with the speech recognition process itself. A radical approach eschews the word-level representation altogether and interprets language directly from the phonetic representation; this has been shown to be useful in call routing applications (Alshawi 2003; Huang and Cox 2006). Milder approaches include building phonetic and semantic representations together (Schuler, Wu, and Schwartz 2009) or allowing NLU to select among competing ASR outputs (Chotimongkol and Rudnicky 2001; Gabsdil and Lemon 2004; Skantze 2007). What is common to all of these approaches is that they work with the speech signal directly, and thus incur similar costs to modifying an ASR component, including the need for a substantial amount of speech data for training, and the commitment to one particular speech recognizer with which the rest of the system is integrated.

We present a different approach: we accept the output of an off-the-shelf speech recognizer as-is (with trained domain-specific language models), and use a dictionary to endow the NLU component with a way to compute phonetic similarity between strings of words. We do not attempt to correct the ASR output through postprocessing as in Ringger (2000), and we deliberately ignore detailed information

*Now at Columbia University.

from the speech recognizer such as word and phone lattices; our approach thus avoids the costs associated with training on speech data, allows replacing one off-the-shelf speech recognizer with another, and yields performance gains even when there is little or no speech data available to train with.

We demonstrate our approach using NPCEditor (Leuski and Traum 2010), a statistical NLU which for each input utterance selects one output out of a fixed set, based on a learned mapping between input and output language models. Instead of creating input language models based on word tokens, we translate each input word string into a string of phones using a dictionary; we then create language models which include both word information and phonetic information, which allows the NLU to select an output based on both word and phone similarities to its training data. Our experiments show that the best performance comes from mixture models, which combine and weigh separate language models for words, phones, and their respective n-grams.

The primary motivation for this work is to improve the performance of NLU in the face of a less than perfect input word string. Our method does however touch on a parallel debate in the psycholinguistic literature: autonomous models assert that word recognition happens in a bottom-up fashion, with contextual information integrated at a later stage (Marslen-Wilson 1987; Norris, McQueen, and Cutler 2000), whereas in interactive models, context affects the earliest stages of word recognition (Marslen-Wilson and Welsh 1978; McClelland and Elman 1986). Our method can be seen as an extreme implementation of an autonomous model, where all the information about the speech signal is discarded after a word is identified, and downstream interpretation processes must resort to heuristics in order to recover from errors.

The remainder of the paper describes in detail the experiment setup and results, and concludes with broader implications and directions for further study.

Method

Data

We evaluate our method using two sets of data collected from deployed virtual question-answering characters: SGT Star (Artstein et al. 2009), and the twins Ada and Grace (Swartout et al. 2010). SGT Star answers questions from potential recruits about the U.S. Army, and is contained in a mobile exhibit. Ada and Grace are part of a fixed exhibit at the Museum of Science in Boston; they answer questions from visitors about exhibits in the museum and about science in general. Characters in both systems can also answer questions about themselves, such as the example above about whether SGT Star is married. Each system has a fixed set of pre-recorded responses (283 responses for SGT Star, 148 for the Twins), and uses a statistical NLU trained on a set of example user utterances with a many-to-many mapping to appropriate responses. The NLU is designed to select the most appropriate response to variable inputs which are the result of speech recognition errors as well as variations in the phrasing of questions.

Visitor interaction with the characters is done primarily through trained handlers, who relay the visitors' questions and act as intermediaries between the visitors and the characters. The handlers are familiar with the characters, and many of their utterances are a precise word for word match of utterances in the characters' training data. It is these utterances that form the basis for our experiment, because for these utterances we know the set of correct responses; if they were sent to the NLU as uttered, the response would be perfect. But the utterances are processed by a speech recognizer, which introduces errors that sometimes lead to incorrect interpretation. The purpose of our experiments is to identify techniques for interpreting this speech recognizer output more accurately.

Data set	N	Word Error Rate (%)		
		Median	Mean	S.D.
SGT Star	3498	43	45	38
Twins	7690	20	29	36
Star Unseen	759	50	51	36

Table 1: ASR Quality for the different data sets

Our test data contain 3498 utterances from the SGT Star domain and 7690 utterances from the Twins domain, all of whose transcriptions are identical to one of the training utterances in their respective domains (transcriptions were performed manually). The data come from actual deployments of the characters, and each utterance contains the original speech recognizer output retrieved from the system logs. Speech recognition was much better for the Twins domain, with about half the word error rate (Table 1). In each domain, all of our experiments use the same training and test data – the original utterance-response links for training, and the speech recognizer output for testing; the differences are only in how the NLU constructs language models from these data. A response to a speech recognizer output is scored as correct if it is linked in the training data to the corresponding manual transcription; otherwise it is scored as incorrect.

In addition to the above data, we investigated 759 utterances from the SGT Star domain whose transcriptions were not found in the training data. This was done in order to verify that our results are also valid when the NLU has to overcome variability in phrasing on top of speech recognition errors. For these utterances there is no automatic way to determine whether responses are correct, so all responses produced by the various test conditions were rated manually as correct or incorrect.

Natural Language Understanding

In our experiments we used NPCEditor (Leuski and Traum 2010) – a text classification system which is available for download as part of the ICT Virtual Human Toolkit.[1] We summarize this work briefly here in order to describe how we modified it to accommodate phonetic information.

NPCEditor employs a statistical language modeling approach called relevance models (Lavrenko, Choquette, and

[1] http://vhtoolkit.ict.usc.edu

Croft 2002). At the center of the system is the computation of a language model for A_Q – the ideal answer to the user's question Q. This language model is the probability $P(w|A_Q)$ that a token sampled at random from the answer will be the token w, and is computed from training data in the form of question-response pairs. The language model of A_Q is compared to the language models of all available character responses R – the probability $P(w|R)$ that a token sampled at random from the response will be the token w – and NPCEditor selects the response with the closest language model to that of the ideal answer A_Q.

The language model of each available response R is computed using Maximum Likelihood Estimation (MLE) with Jelinek-Mercer smoothing (Bahl, Jelinek, and Mercer 1990):

$$P(w|R) \cong \pi_R(w) = \lambda_\pi \frac{\#_R(w)}{|R|} + (1-\lambda_\pi)\frac{\sum_R \#_R(w)}{\sum_R |R|}$$

where $\#_R(w)$ is the number of times token w appears in sequence R, $|R|$ is the length of sequence R, and λ_π is a parameter that can be determined from the training data. The language model of the idealized answer A_Q is estimated using a cross-lingual relevance model estimation

$$\begin{aligned} P(w|A_Q) &\cong P(w|Q) = \frac{P(w,q_1,...,q_n)}{P(q_1,...,q_n)} \\ &\cong \phi_Q(w) = \frac{\sum_j \pi_{R_j}(w)\prod_{i=1}^m \pi_{Q_j}(q_i)}{\sum_j \prod_{i=1}^m \pi_{Q_j}(q_i)} \end{aligned}$$

where the sum is over all linked question-response pairs $\{Q_j, R_j\}$ in the character database.

To compare the answer to the user's question with a character response, NPCEditor compares the corresponding distributions $\phi_Q(w)$ and $\pi_R(w)$ by applying Kullback-Leibler (KL) divergence:

$$D(A_Q||R) = \sum_{w \in V_R} \phi_Q(w) \log \frac{\phi_Q(w)}{\pi_R(w)}$$

where the sum is over all tokens observed in character responses. The KL-divergence is a dissimilarity measure, so NPCEditor uses $-D(A_Q||R)$ as the confidence score.

So far this discussion assumed that we are working with a single token type (e.g. words). These tokens can be different for questions and responses, but for a single utterance type we assumed that the tokens have the same statistical properties. NPCEditor supports mixture models where the same text string can be represented as, for example, a sequence of words and a sequence of word pairs (bigrams). Leuski and Traum (2010) call these individual sequences "fields." NPCEditor implements a mixture language modeling approach that calculates probability distributions for each individual field and then combines them using a weighted mixture:

$$D(A_Q||R) = \sum_l \alpha_l \sum_{w \in V_{R(l)}} \phi_{Q(l)}(w) \log \frac{\phi_{Q(l)}(w)}{\pi_{R(l)}(w)}$$

Here the outer summation goes over every field l of interest in responses, while the inner summation iterates over vocabulary for that field. $R(l)$ denotes the lth field in a response sequence. The distribution $\phi_{Q(l)}(w)$ is similarly defined as a mixture of probability distributions for the question fields:

$$\phi_{Q(l)}(w) = \frac{\sum_j \pi_{R_j(l)}(w)\prod_k (\prod_{i=1}^m \pi_{Q_j(k)}(q_{k,i}))^{\beta_k}}{\sum_j \prod_k (\prod_{i=1}^m \pi_{Q_j(k)}(q_{k,i}))^{\beta_k}}$$

where $Q_j(k)$ is the kth field in the question from the jth question-response pair in the character database and $q_{k,i}$ is the ith word or token in the kth field of the input question. Parameters α_l and β_k allow us to vary the importance of different fields in the mixture and are determined from the training data.

Simple	Bag		Mixture	
word	w12	w12p1	w12	w12p1
phone		w12p2	w1p2	w12p2
biphone		w12p12	w1p12	w12p12
triphone		w12p123	w1p123	w12p123

Table 2: Tokenizers used in the experiments

Tokenization

NPCEditor comes with default tokenizers for words and word bigrams, which constitute separate fields as described in the previous section. We extended NPCEditor by creating custom tokenizer plugins that parse the utterance text and produce additional fields that represent phonetic information. Each string of words was transformed into phones using the CMU dictionary,[2] as in the following example.

word:	does	the	army	pay	well
phone:	d ah z	dh ah	aa r m iy	p ey	w eh l

We then created tokens from single words, word bigrams, single phones, phone bigrams, and phone trigrams.

bigram: does_the the_army army_pay pay_well

biphone: d_ah ah_z z_dh dh_ah ah_aa aa_r r_m m_iy iy_p p_ey ey_w w_eh eh_l

triphone: d_ah_z ah_z_dh z_dh_ah dh_ah_aa ah_aa_r aa_r_m r_m_iy m_iy_p iy_p_ey p_ey_w ey_w_eh w_eh_l

The phone n-grams deliberately ignore word boundaries, in order to allow recovery from errors in boundary placement by the speech recognizer (as in the example from the introduction, where "all soldiers" was misrecognized as "also just").

Our tokenizers fall into three groups: simple tokenizers use just one kind of token, "bag" tokenizers lump two or more kinds into a single field, and mixture models combine two or more token types as distinct fields in a mixture model. We use mnemonics of the form **w[12]p[123]** to designate the bag and mixture model tokenizers – for example, w12p2 combines word unigrams, word bigrams and phone bigrams. Altogether we used 17 tokenizers, shown in Table 2.

[2] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Results

Accuracy/return tradeoff

The Natural Language Understanding component, NPCEditor, is more than a classifier: it also employs dialogue management logic designed to avoid the worst responses. NPCEditor calculates the classifier's confidence in the appropriateness of the selected response, and if it falls below a certain threshold, the selected response is replaced with an "off-topic" utterance that asks the user to repeat the question or takes initiative and changes the topic (Leuski et al. 2006); such failure to return a response (also called non-understanding) is usually preferred over returning an inappropriate one (misunderstanding). The capability to not return a response is crucial in keeping conversational characters coherent, but it is not captured by standard classifier evaluation methods such as accuracy, recall (proportion of correct responses that were retrieved), or precision (proportion of retrieved responses that are correct). We therefore evaluate the different tokenizers in a way that takes into account the ability to avoid responses for certain questions.

A response from the system falls into one of three categories – correct, incorrect, or off-topic, which happens when the best response is not good enough. NPCEditor calculates a response threshold that finds an optimal balance between false positives (inappropriate responses above threshold) and false negatives (appropriate responses below threshold) on the training data; however, it turns out that the various tokenizers yield very different return rates, making it impossible to compare them directly. Instead, we compare the tokenizers across all possible return rates. A better tokenizer will give fewer errors at all return levels, or at least at all the relevant ones (typically, it is acceptable to not return 10%–30% of the responses).

We compared the various tokenizers by looking at the trade-off between returns and error levels. For each test utterance we logged the top-ranked response together with its confidence score, and then we plotted the rate of off-topics against errors at each possible threshold; this was done separately for each tokenizer (since confidence scores are based on parameters learned during training, they are not comparable across tokenizers). Figure 1 shows the curves for 5 representative tokenizers: non-returns are plotted on the horizontal axis and corresponding error rates on the vertical axis; at the extreme right, where no responses are returned, error rates are necessarily zero for all tokenizers. Lower curves indicate better performance.

We note a number of observations from these charts. First of all, the scales are different: when no off-topics are returned we get around 30% errors in the SGT Star domain, 10% errors in the twins domain, and 45% errors for the SGT Star unseen utterances. Nevertheless, the relations between the curves are rather consistent between the three plots, which suggests that the results may generalize to other domains. As a baseline we take the simple word tokenizer. Other simple tokenizers, represented here by biphones, are usually worse than word tokens, though there is some variability – for example, we see that the biphone tokenizer on the SGT Star data is better than words at low non-return rates

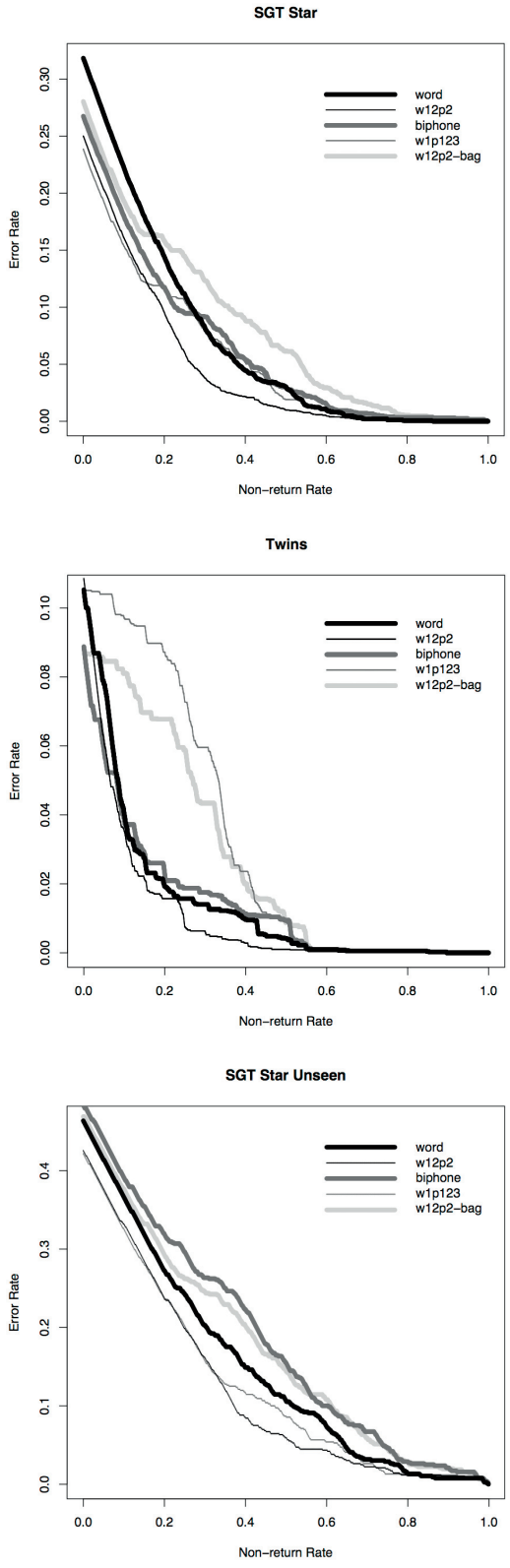

Figure 1: Trade-off between errors and non-returns

but worse at higher non-return rates. A peculiar case is the simple phone tokenizer (not shown), which is substantially worse than words across the entire range on the SGT Star data, but better than words on the twins data; we do not have an explanation for this behavior. Bag-of-features tokenizers, represented here by w12p2-bag, are also usually worse than word tokens, especially at higher non-return rates (above 20% non-return for SGT Star and above 10% for twins).

Where we do get substantial performance improvements is the mixture models. The best performing tokenizer, for all 3 datasets and across the entire range of return rates, is w12p2. For the SGT Star domain it beats the word tokenizer by 5%–7% until the latter's error level drops to 10%, and continues to provide more modest gains at higher non-return rates. The other mixture models which include the same features, w12p12 and w12p123 (not shown), come fairly close. Other mixture models do not show a consistent improvement over word tokens. For example, w1p123 is better than words on the SGT Star domain, but much worse than words on the twins domain; similar mixture models, containing word unigrams and a variety of phonetic features but not word bigrams, display similar behavior. The mixture model w12 (not shown), containing words and bigrams without any phonetic features, is very similar to word tokens on all 3 domains.

It turns out, then, that the necessary features for high performance are words, word bigrams, and phone bigrams. At this point we can only conjecture about the reason for this. The phonetic features allow the NLU to recover from certain speech recognition errors; phone unigrams probably do not carry sufficient information, which is why bigrams are required. However, phonetic features alone might cause too much confusion, which is why word information is also needed. Apparently, both word unigrams and bigrams are required to offset the phonetic confusion, though it is not exactly clear why, especially given that without phonetic features, words and bigrams are practically equivalent to words alone. At any rate, the experiments demonstrate that when used appropriately, phonetic features derived from a dictionary can improve the performance of NLU in the face of speech recognition errors.

Full ranking of responses

The preceding discussion assumed that NPCEditor always chooses the top-ranked option as the character response. For some applications, however, it may be desirable to return all of the appropriate responses rather than only the best one, for instance if the Natural Language Understanding component passes its result to a dialogue manager downstream. NPCEditor evaluates and ranks all of the responses, and certain utterances may have more than one correct response. If multiple responses are returned, one wants to maximize the discrimination between correct and incorrect responses, so that a maximal number of correct responses are returned with a minimal number of incorrect ones. Discrimination between correct and incorrect responses can be viewed with a Detection Error Tradeoff (DET) curve (Martin et al. 1997).

Figure 2 shows DET curves for the same 5 tokenizers of Figure 1. Each curve, corresponding to a single tokenizer,

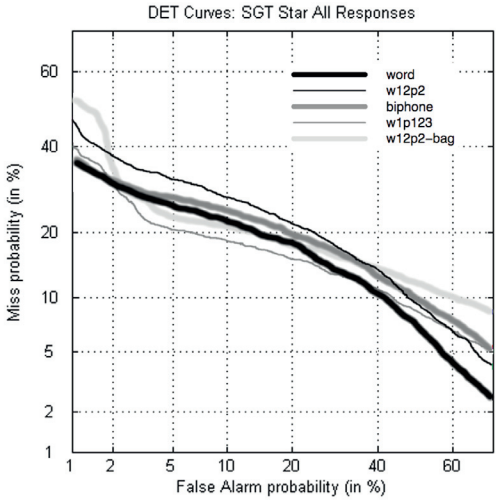

Figure 2: SGT Star discrimination among all responses

shows the false alarm rate (errors above threshold as a percentage of all errors) plotted against the miss rate (correct responses below threshold as a percentage of all correct responses), based on the scores of all responses to all test utterances. Lower curves indicate better discrimination. The best discrimination at very low (under 2%) and very high (over 40%) false alarm rates is achieved by the word tokenizer, while in the middle range, better discrimination is achieved by w1p123. Tokenizer w12p2, the consistent top performer on the task of picking the best single response, is among the worst in discriminating among the full set of responses. Explaining this observation would require a more detailed investigation.

Discussion

Our experiments show that adding phonetic features, derived from a dictionary, can substantially improve the performance of a Natural Language Understanding component. This is because the phonetic features allow the NLU to recover from certain kinds of speech recognition errors, where the actual words uttered by the speaker are misrecognized as distinct but phonetically similar words.

The phonetic dictionary used in tokenizing the ASR output gives the NLU access to information that would otherwise not be available, allowing it to reason about phonetic similarity between strings. However, the transformation from words to phones also loses information, most importantly about word boundaries. This highlights the importance of the word level, and explains why the best performers are mixture models, which make use of both word and phone-level information.

The contrast in relative performance of w12p2 in Figures 1 and 2 points out that it is important to evaluate the NLU in the manner that it will be actually used – one might pick a different tokenizer depending on whether one wants to maximize accuracy of n-best or 1-best, or whether one cares more about false positives or false negatives.

Our method is extremely practical: it is easy to implement, does not require any communication overhead, is not tied to a specific speech recognizer, and can be applied to the output of an off-the-shelf speech recognizer without access to its code: all of the experiments presented in this paper were performed on the text output of a speech recognizer, without ever accessing the original audio. It is possible, however, that the actual phonetic content of the utterance, as determined by the speech recognizer, would be even more useful. This is worthy of further investigation.

We chose to experiment with an NLU component to which adding phonetic information is fairly straightforward, because it treats its input as tokens without additional structure. Our method would probably transfer to other applications that use language modeling of spoken utterances, such as Machine Translation. However, NLU architectures such as parsing assign more meaning and structure to the word tokens, so our method would not transfer easily. Nevertheless we believe that any components that process spoken language (or, more specifically, ASR output) would benefit from an ability to reason about phonetic similarities between words or strings. Our method may also be useful for systems that process text interaction, to the extent that users make phonetically based errors (e.g. homophone substitutions).

We end with an observation about the application of this research beyond Natural Language Processing. As we mentioned in the introduction, our method is intended to improve machine performance and not as a model of human cognition. Nevertheless, humans do possess the ability to reason about phonetic relatedness of text strings. Passonneau et al. (2010) show that human wizards are good at recovering from ASR output errors, and in a pilot study we found that a human annotator presented with ASR output and simply guessing the wording of the original utterances was able to reduce word error rate from 59% to 42%, on a sample of Twins data specifically biased towards higher word error rates. We as humans are familiar with the feeling of failing to understand the words of an utterance, only to make sense of it a few seconds later. So the ability to perform phonetic reasoning and post-processing of an utterance should form some part of a model of human language comprehension.

Acknowledgments

The project or effort described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

References

Alshawi, H. 2003. Effective utterance classification with unsupervised phonotactic models. In *HLT-NAACL 2003*, 1–7.

Artstein, R.; Gandhe, S.; Gerten, J.; Leuski, A.; and Traum, D. 2009. Semi-formal evaluation of conversational characters. In Grumberg, O.; Kaminski, M.; Katz, S.; and Wintner, S., eds., *Languages: From Formal to Natural. Essays Dedicated to Nissim Francez on the Occasion of His 65th Birthday*, volume 5533 of *LNCS*. Springer. 22–35.

Bahl, L. R.; Jelinek, F.; and Mercer, R. L. 1990. A maximum likelihood approach to continuous speech recognition. In Waibel, A., and Lee, K.-F., eds., *Readings in Speech Recognition*. San Francisco: Morgan Kaufmann. 308–319.

Chotimongkol, A., and Rudnicky, A. I. 2001. N-best speech hypotheses reordering using linear regression. In *EuroSpeech 2001*.

Gabsdil, M., and Lemon, O. 2004. Combining acoustic and pragmatic features to predict recognition performance in spoken dialogue systems. In *ACL 2004, Main Volume*, 343–350.

Huang, Q., and Cox, S. 2006. Task-independent call-routing. *Speech Communication* 48(3–4):374–389.

Lavrenko, V.; Choquette, M.; and Croft, W. B. 2002. Cross-lingual relevance models. In *25th ACM SIGIR*, 175–182.

Leuski, A., and Traum, D. 2010. Practical language processing for virtual humans. In *IAAI 2010*, 1740–1747.

Leuski, A.; Patel, R.; Traum, D.; and Kennedy, B. 2006. Building effective question answering characters. In *7th SIGdial Workshop on Discourse and Dialogue*.

Litman, D.; Moore, J.; Dzikovska, M. O.; and Farrow, E. 2009. Using natural language processing to analyze tutorial dialogue corpora across domains and modalities. In *Artificial Intelligence in Education*, 149–156. Amsterdam: IOS Press.

Marslen-Wilson, W. D., and Welsh, A. 1978. Processing interactions and lexical access during word recognition in continuous speech. *Cognitive Psychology* 10(1):29–63.

Marslen-Wilson, W. D. 1987. Functional parallelism in spoken word-recognition. *Cognition* 25(1–2):71–102.

Martin, A.; Doddington, G.; Kamm, T.; Ordowski, M.; and Przybocki, M. 1997. The DET curve in assessment of detection task performance. In *Eurospeech 1997*, 1895–1898.

McClelland, J. L., and Elman, J. L. 1986. The TRACE model of speech perception. *Cognitive Psychology* 18(1):1–86.

Norris, D.; McQueen, J. M.; and Cutler, A. 2000. Merging information in speech recognition: Feedback is never necessary. *Behavioral and Brain Sciences* 23(3):299–370.

Passonneau, R.; Epstein, S. L.; Ligorio, T.; Gordon, J. B.; and Bhutada, P. 2010. Learning about voice search for spoken dialogue systems. In *HLT-NAACL 2010*, 840–848.

Raymond, C., and Riccardi, G. 2007. Generative and discriminative algorithms for spoken language understanding. In *INTERSPEECH 2007*.

Ringger, E. K. 2000. *Correcting Speech Recognition Errors*. Ph.D. Dissertation, University of Rochester, Rochester, NY.

Schuler, W.; Wu, S.; and Schwartz, L. 2009. A framework for fast incremental interpretation during speech decoding. *Computational Linguistics* 35(3):313–343.

Skantze, G. 2007. *Error Handling in Spoken Dialogue Systems: Managing Uncertainty, Grounding and Miscommunication*. Ph.D. Dissertation, KTH, Stockholm, Sweden.

Swartout, W.; Traum, D.; Artstein, R.; Noren, D.; et al. 2010. Ada and Grace: Toward realistic and engaging virtual museum guides. In *Intelligent Virtual Agents*, volume 6356 of *LNAI*, 286–300. Springer.

Walker, M.; Wright, J.; and Langkilde, I. 2000. Using natural language processing and discourse features to identify understanding errors in a spoken dialogue system. In *17th ICML*, 1111–1118.