Investigating Speech Recognition for Improving Predictive AAC

Jiban Adhikary, Robbie Watling, Crystal Fletcher, Alex Stanage, Keith Vertanen
Michigan Technological University
Houghton, Michigan, USA
{jiban, rwatling, tafletch, amstanag, vertanen}@mtu.edu

Abstract

Making good letter or word predictions can help accelerate the communication of users of high-tech AAC devices. This is particularly important for real-time person-to-person conversations. We investigate whether performing speech recognition on the speaking-side of a conversation can improve language model based predictions. We compare the accuracy of three plausible microphone deployment options and the accuracy of two commercial speech recognition engines (Google and IBM Watson). We found that despite recognition word error rates of 7–16%, our ensemble of N-gram and recurrent neural network language models made predictions nearly as good as when they used the reference transcripts.

1 Introduction

People who are non-verbal often use some form of Augmentative and Alternative Communication (AAC). Common forms of speaking disorders include stuttering, cluttering, apraxia, dysarthria, aphasia, Parkinson's disease, amyotrophic lateral sclerosis (ALS), or cerebral palsy. An AAC device may let a user select letters, words, and phrases from its interface and a communication partner can read the text or hear it via text-to-speech. The rate at which an AAC user can enter text is typically slow (often less than 10 words-per-minute) (Trnka et al., 2009; Simpson et al., 2006; Higginbotham et al., 2007). That is why predictive AAC devices normally use a language model to try and make suggestions of likely upcoming text. These predictions are usually made based solely on the text entered by the AAC user. They typically ignore the two-way nature of conversation which can offer many contextual clues.

In this paper, first we investigate how to record and recognize the speech of a partner communicating with the AAC user. Then we investigate if speech recognition on partner speech improves two-sided conversational language modeling.

2 Related Work

Predictive AAC devices typically use an N-gram language model (LM). An N-gram LM calculates the probability of a token given the previous N-1 tokens. The performance of this model depends on the training data being closely matched to a user's text. But for practical, ethical, and privacy issues, there is a scarcity of text written by AAC users. Researchers have resorted to training LMs on data from news articles (Trnka et al., 2009) or phone transcripts (Wandmacher et al., 2008). Another option is the large amounts of text that can be mined from the internet, e.g. tweets, blog posts, or Wikipedia articles. While such web data may be informal or have other artifacts such as abbreviations, researchers have used filtering methods such as cross entropy difference selection (Moore and Lewis, 2010) to select training data for AAC language models (Vertanen and Kristensson, 2011).

Recently, recurrent neural network language models (RNNLMs) have achieved state-of-the-art performance over traditional N-gram language models. RNNLMs have been shown to better model long range dependencies when combined with techniques such as long short-term memory (Hochreiter and Schmidhuber, 1997) or gated units (Chung et al., 2014). Further gains have also been achieved by interpolating N-gram models (Mikolov et al., 2014) and other techniques (Mikolov et al., 2011a,b).

In addition to using textual context, previous AAC work has also investigated using face detection (Kane et al., 2012), vision (Kane and Morris, 2017), and location (Demmans Epp et al., 2012) as context for AAC predictions. But limited work has been done to predict AAC user's response based on partner speech. Wisenburn (2008; 2009) created a program called Converser and used speech recognition to identify the speaking partner's words. This input was then parsed by a

Proceedings of the Eighth Workshop on Speech and Language Processing for Assistive Technologies, pages 37–43

A:	Did you call the theater?
B:	So sorry, I forgot to call the theater.
A:	You can just go online.
B:	That's true, I'll do that now.
A:	What movie is it that you want to see?
B:	The lord of the rings.

Table 1: A dialogue created by Amazon Turk workers.

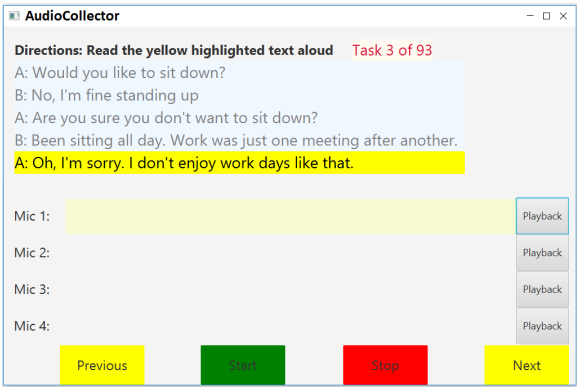

Figure 1: The application used to record dialogues.

noun phrase identification system. Identified noun phrases, along with relevant static messages, were displayed to the user. This provided users with a faster communication rate compared to a system that did not use partner speech. In our work, we also perform speech recognition on the partner's speech. However, we will use recognition results as context to our language models in hopes of better predicting an AAC user's upcoming text.

3 Speech Data Collection

Our first step was to obtain text and audio data reasonably representative of everyday person-to-person conversations. In this section, we detail how we collected this data. Further, we designed our collection to answer the practical question of how and where a microphone might be located for recording a partner's speech.

As a starting point for our spoken dialogue collection, we used the text dialogues collected by Vertanen (2017). These dialogues were invented by workers on Amazon Mechanical Turk. The dialogue started with a question invented by one of the workers. Subsequent workers then extended the dialogue by another turn until a total of six turns were completed. Table 1 shows an example dialogue. The original collection had 1,419 dialogues. We removed 265 we deemed potentially offensive, resulting in a set of 1,154 dialogues.

3.1 Audio Data Collection

The dialogue data from (Vertanen, 2017) consisted only of text. We wanted to investigate whether a partner's speech could improve an AAC device's predictions. We designed a desktop application to record audio data of participants' speaking turns in the text dialogues. The application highlighted the current turn we wanted the participant to speak. Any previous turns of the dialogue were also shown as context. The application recorded from three microphones simultaneously:

- HEADSET — A Logitech H390 USB noise cancelling headset microphone.
- LAPTOP — The built-in microphone of a 13" 2015 MacBook Pro laptop.
- CONFERENCE — A MXL AC404 USB conference microphone. This microphone was positioned behind the laptop at a distance of approximately 0.9 m from the participant.

The application allowed the participant to re-record any utterances in which they misspoke. We analyzed just the last recording for each dialogue turn. Audio was recorded at 44.1 kHz. We recruited 14 participants via convenience sampling. Four self-reported as male, ten as female. The average age was 36. Participant 5 reported having a foreign accent. Each participant took part in an approximately half-hour session and was paid $10. Participants sat at a desk with a laptop in quiet office. They were allowed to adjust their chair so they could comfortably operate the laptop.

Participants first recorded three practice dialogues. We did not analyze the practice dialogues. Each participant then completed half the turns in 28 additional dialogues. The subsequent participant completed the other half of the turns of the same 28 dialogues. In total, we collected 1,176 utterances constituting both sides of 196 dialogues. We have made our filtered text dialogues, audio recordings, recognition results, and Java audio collection application available to other researchers[1].

3.2 Speech Recognition Experiments

We performed speech recognition using two commercially available speech recognizers, Google Cloud Speech-to-Text and IBM Watson Speech-to-Text. We performed speech recognition on audio from each of the three different microphones.

[1] https://digitalcommons.mtu.edu/data-files/1

	Microphone		
	LAPTOP	HEADSET	CONF.
GOOGLE	7.3±1.0	7.0±1.0	8.9±1.2
WATSON	10.5± 1.2	10.7±1.2	16.0±1.6

Table 2: Word Error Rate (WER %) using different microphones and speech recognizers. Results are formatted as mean ± 95% bootstrap confidence intervals.

We computed the Word Error Rate (WER) of each recognition result against its reference transcript.

The reference transcripts included various numeric characters representing times or amounts. We found the recognition results on such turns were variable. Sometimes the recognizer returned numeric transcriptions and sometimes numbers were spelled out as words. For consistency, we dropped all dialogues if any of its reference turns had a number in it. This reduced the number of dialogues from 196 to 160.

As shown in Table 2, the mean WER on the three different microphones using the GOOGLE recognizer was LAPTOP 7.3%, HEADSET 7.0%, and CONFERENCE 8.9%. IBM's recognizer had higher error rates with LAPTOP at 10.5%, HEADSET at 10.7%, and CONFERENCE at 16.0%.

Figure 2 shows the WER for each participant using the GOOGLE speech recognizer and audio from the HEADSET microphone. 9 of the 14 participants had a lower mean WER of 5.5%. This was driven by the fact that 84.0% of their utterance turns were recognized with no errors.

We recorded our audio in a quiet office. We also wanted to explore how our methods might work in noisier locations. To do this, we injected a recording of street noise into our clean audio data. We used the SoX Sound eXchange utility to add in the street noise at three different volume levels: 0.1, 0.2, and 0.3. Figure 3 shows the mean word error rates on recordings with no noise and at the three noise levels. Even at noise volume level 0.3, both recognizers' mean word error rates using the HEADSET and LAPTOP microphones stayed below 40%. However, the mean word error rates using the CONFERENCE microphone started deteriorating more sharply with increasing noise.

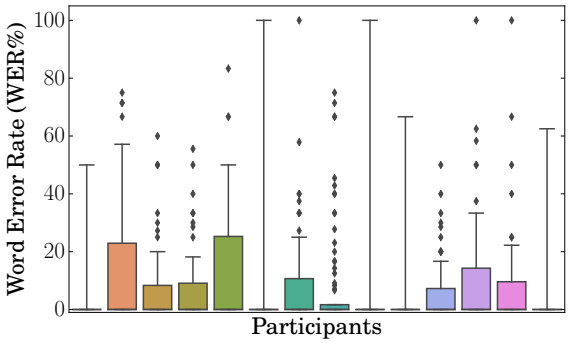

Figure 2: Participants' per utterance WER using the Google recognizer and audio from a headset mic.

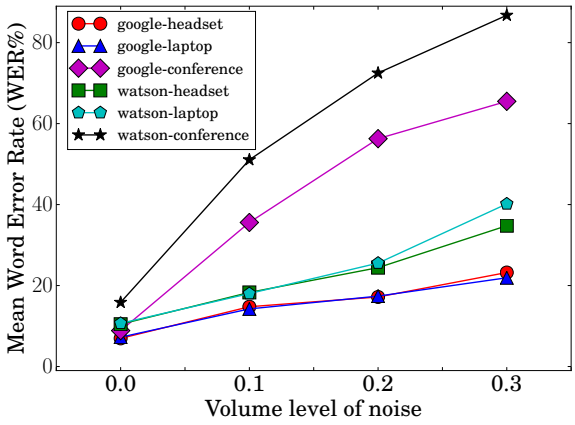

Figure 3: WER on audio dialogue turns without noise and with three different injected noise levels.

4 Language Modeling Experiments

We now investigate how to use language models to better predict turns in our dialogue collection. Recall we recorded both sides of 196 of the dialogues from our set of 1,154 dialogues. After dropping dialogues with numbers, we arrived at a test set of 160 dialogues with audio data. We created text-only training and development sets from the remaining 958 dialogues. From these dialogues, we dropped 128 that contained numbers. We randomly selected 160 from the remaining dialogues as a development set and 670 as a training set.

Our language modeling experiments used a vocabulary of 35 K words. The vocabulary consisted of the most frequent known English words occurring in 50 M words of sentences parsed from Twitter. Any words not in this vocabulary were mapped to an unknown word token. We converted text to lowercase and removed punctuation aside from apostrophe. Throughout, we report the per-word perplexity of our test set (160 dialogues, 960 turns, 7.1 K words). We excluded the sentence end pseudo-word from our calculations.

4.1 N-gram Language Models

We took each turn in the training set as an independent training example (4,020 turns, 30 K words). We trained a 4-gram interpolated modified Kneser-Ney model using SRILM (Stolcke, 2002;

Training data	Words	PPL
Twitter, small amount of data	30 K	417.3
Crowd dialogues	30 K	211.8
Twitter, large amount of data	50 M	96.0
+ CE diff. 25% dialogues	50 M	91.0
+ CE diff. 50% dialogues	50 M	86.8
+ CE diff. 75% dialogues	50 M	83.5
+ CE diff. 100% dialogues	50 M	83.5
+ Optimized CE threshold	50 M	77.4

Table 3: N-gram perplexity varying training data.

Model	PPL Sentence	PPL Dialogue
Twitter RNNLM	179.0	129.3
+ GRUs	167.8	122.8
+ NCE	172.2	111.9
+ maximum entropy	123.7	84.1
+ Twitter 4-gram LM	75.2	71.5
+ unigram cache	75.2	68.5

Table 4: Perplexities with added features. We reset the RNNLM between each sentence or after each dialogue.

Stolcke et al., 2011). As shown in Table 3, the perplexity on the test set was 211.8. For comparison, we trained a 4-gram model on 30 K words of random Twitter data collected via Twitter's streaming API between 2009–2015. The Twitter model had a much higher perplexity of 417.3.

An approach to filtering an out-of-domain training data is cross-entropy difference selection (Moore and Lewis, 2010). This approach calculates the cross-entropy of individual sentences under an in-domain and an out-of-domain model trained on similar amounts of data. We trained our in-domain model on between 25–100% of the text in our Turk dialogue training set.

We selected 50 M words of Twitter data below a certain cross-entropy difference threshold. We used an initial threshold of -0.3. The more negative the threshold, the more sentences had to resemble in-domain text in order to be selected. As shown in Table 3, using more in-domain data reduced perplexity though gains eventually flattened. Finally, we used all the in-domain data to search for the optimal cross-entropy difference threshold on the development set. The optimal threshold of -0.06 further lowered perplexity to 77.

4.2 RNN Language Models

Next, we investigated training Recurrent Neural Network Language Models (RNNLMs) on the cross-entropy difference selected Twitter data. We trained our models using the Faster RNNLM toolkit[2]. For each model type, we trained 10 RNNLMs with different random initialization seeds. We report the perplexity on the test set of the model that had the lowest perplexity on the development set. Unless otherwise noted, we used the default hyperparameters of Faster RNNLM.

[2] https://github.com/yandex/faster-rnnlm

During evaluation we reinitialized the RNNLM after every sentence or after every six-turn dialogue. This allowed us to observe how much the model was adapting to a particular dialogue while avoiding allowing the model to adapt to the general style of our Turk dialogues.

As shown in Table 4, a model trained with 250 sigmoid units had a perplexity of 129.3 on each dialogue. Switching to 250 Gated Recurrent Units (GRUs) (Chung et al., 2014) reduced perplexity to 122.8. Switching to Noise Contrastive Estimation (NCE) (Chen et al., 2015) further reduced perplexity to 111.9. Training a maximum entropy language model of size 1000 and order 4 in the RNN reduced perplexity substantially to 84.1.

We interpolated our best RNNLM with our best previous N-gram model. We optimized the mixture weights with respect to our development set. This further reduced perplexity to 71.5. We also investigated a unigram cache (Grave et al., 2016). Similar to the RNNLM, we reset the cache after each sentence or after each dialogue. The cache model provided a small reduction in perplexity to 68.5. The mixture weights were: N-gram 0.55, RNNLM 0.42, and unigram cache 0.04.

Comparing the result columns in Table 4, we see consistently higher perplexities when the RNNLM was evaluated on sentences instead of on entire dialogues. In particular, the RNNLM was substantially worse with a perplexity of 123.7 on sentences versus 84.1 on dialogues. This demonstrates the ability of the RNNLM to adapt to aspects of the text over a longer time horizon.

4.3 Two-sided Dialogue Language Models

We now turn to training language models on two-sided dialogues. Since our Amazon Turk dialogue collection is relatively small, we instead used dialogues from movies (Danescu-Niculescu-Mizil and Lee, 2011). We created a training set

Model	PPL
Movie dialogue 7-gram	138.5
Movie dialogue RNNLM	129.1
Turk dialogue RNNLM	185.5
Mixture, dialogue models	104.3
Mixture, Twitter + dialogue + cache	66.3

Table 5: Perplexity of models trained on two-sided dialogues and mixtures of dialogue and twitter models.

of 83 K dialogues consisting of 305 K turns and 3.2 M words. We introduced a pseudo-word to denote speaker changes. We excluded this speaker change word from our perplexity calculations. We treated the set of turns making up a dialogue as a single "sentence" during training and testing. We evaluated models on each dialogue in our Turk test set (the same set used previously). In the case of RNNLMs, we reset the model after each dialogue.

We first tested 4-gram through 8-gram N-gram models. The 4-gram had the highest perplexity of 139.2 The 7-gram model had the best perplexity of 138.5 (Table 5). Next we trained a RNNLM on the movie dialogues using 300 GRU units, NCE, and with a maximum entropy model of size 1000, order 4. The RNNLM had a lower perplexity of 129.1. This again highlights the ability of the RNNLM to better model long-range dependencies and/or topics compared to the N-gram model.

We also trained an RNNLM on just the Turk dialogues. We used 100 GRU units, NCE, and a maximum entropy model with 100 units and an order of 4. This model had a perplexity of 185.5. We think this model's worse performance reflects the substantially smaller amount of training data. By interpolating these three dialogue models, we obtained an even lower perplexity of 104.3. The mixture weights were: Movie 7-gram 0.24, Movie RNNLM 0.40, and Turk RNNLM 0.37.

Our two-sided models were trained on modest amounts of data. To see if they still offered gains in combination with models trained on substantially more Twitter data, we interpolated all our models. The mixture weights were: Twitter N-gram 0.43, Twitter RNNLM 0.32, movie dialogue N-gram 0.05, movie dialogue RNNLM 0.10, Turk dialogue RNNLM 0.06, and unigram cache 0.04. The mixture model's perplexity was 66.3, a modest gain compared to the 68.5 obtained using a mixture of the Twitter models and unigram cache. It does however represent a more substantial gain compared to the 77.4 of the best N-gram only model. This shows that having access to both sides of a dialogue combined with the adaptive nature of RNNLMs may offer improved predictive AAC.

4.4 Impact of Speech Recognition Errors

In real-time person-to-person conversations, we cannot expect to have a perfect transcript of the other side of the conversation. We now investigate the impact of speech recognition errors on the performance of our language models. We did this by measuring the perplexity on two copies of the test set. In the first copy, we replaced the transcript of the even number dialogue turns with the speech recognition result of one of our participants speaking that turn. In the second copy, we replaced the odd number turns. We report the perplexity calculated from the odd turns from the first copy and the even turns from the second copy.

The entire six turns were provided to the language models for both copies to allow the model to condition on prior turns (including any speech errors). We reset the RNNLMs and unigram cache model between each dialogue. We used the previous best ensemble of six models which had a perplexity on the test dialogues of 66.3. We tested injecting the speech recognition results from the three microphones, two recognition engines, and four noise levels (none, 0.1, 0.2, and 0.3).

As shown in Figure 4, the perplexity of our ensemble of models only increased slightly when we replaced the reference transcripts with speech recognition results based on noise-free audio. For example, the far-field conference microphone had a WER of 8.9%. However, the errors introduced by recognition only slightly increased the perplexity of the dialogues from 66.3 to 66.6. Similar to WER in Figure 3, as the level of injected noise increased, perplexities also increased.

5 Discussion and Limitations

In this paper, we conducted an initial investigation into the feasibility of performing speech recognition on an AAC user's speaking partner. We found that whether audio was captured from a wired headset or from a far-field microphone, we could recognize conversational-style utterances with error rates between 7–16%. We found Google's speech engine provided more accurate recognition than the IBM Watson recognizer. However, IBM's engine offers other benefits such as exposing probabilistic information about recognition re-

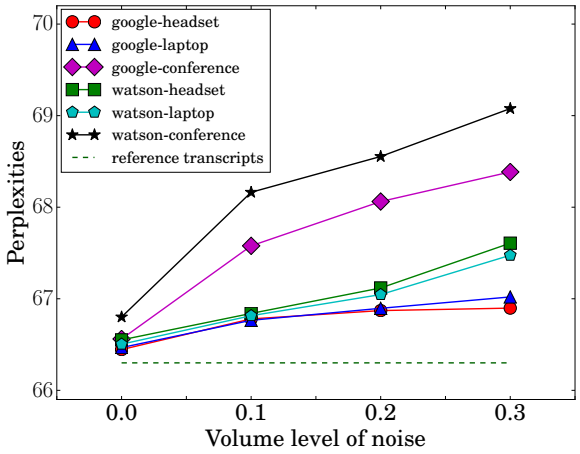

Figure 4: Perplexities using speech recognition on partner turns rather than reference transcripts. Results for no added noise, and for three levels of injected noise.

sults (e.g. a word confusion network). Such information might be leveraged to help avoid conditioning a predictive AAC interface on erroneous regions of a partner's speech recognition result.

Our participants were given the verbatim text for each of their dialogue turns. As such, we can expect they spoke more fluently than one could expect in a spontaneous conversation. Further, we only collected audio in a quiet environment. While our results seem robust to artificially added noise, it remains to be seen if this holds for real-world noisy environments. As such, our error rates probably represent a lower-bound of what could be expected. Nonetheless, it is reassuring that our language models predict the non-speaking side's text with only minimal perplexity loses despite relying on text obtained via speech recognition.

Thus far we have focused on ascertaining whether there is a potential advantage to conditioning on recognition of the speaking side. Whether the perplexity gains we showed will result in actual practical improvements in the auspices of a predictive AAC interface remain to be seen. Further work is needed to understand whether these language model gains will result in, for example, better word predictions that actually save a user keystrokes. Even more work is needed to validate if end-user performance improves.

The use of speech recognition by an AAC device also has obvious privacy implications. This may require the AAC device or user to allow partners to opt-in to having their voice recognized. Further, our current work used cloud-based speech recognition. Users may prefer to have their speech recognized locally on device. Local recognition may also be necessary to avoid network latency or to allow use without network connectivity.

Our goal here was to demonstrate some of the building blocks necessary for modeling everyday conversational-style text. While we made some effort to optimize our models (e.g. tuning mixture weights on development data), further improvements are certainly possible. For example, we did not conduct an extensive search for the best hyperparameters used during RNNLM training. Further, we need to investigate whether our methods and results scale to substantially more training data.

Our results show the benefits of language models based on recurrent neural networks. In particular, we found even when trained on non-dialogue data, RNNLMs adapted to the content of our short dialogues, providing good gains compared to an N-gram model. Further, we showed how a small in-domain corpus can be used to optimize models for everyday conversations. Despite our relatively small amount of two-sided dialogues data (3.2 M words of movie dialogues), we obtained improvements compared to using models trained only on much more non-dialogue data (50 M words of Twitter). In the end, we found an ensemble of N-gram and RNNLMs trained on sentence and dialogues combined with a unigram cache model provided the best performance.

6 Conclusions

AAC users often face challenges in taking part in everyday conversations due to their typically slow text entry rates. Predictions can provide an opportunity to accelerate their communication rate, but it is crucial these predictions be as accurate as possible. Leveraging real-world contextual clues offers one route to improving these predictions. In this paper, we found speech can be accurately recognized with a variety of microphone configurations that might be deployed on an AAC device. Further, we found the error rates of current state-of-the-art recognizers allowed predictions nearly as good as having the verbatim text of the partner's turn. We think this work provides promising results showing a partner's speech can provide context to improve an AAC device's predictions.

7 Acknowledgements

This material is based upon work supported by the NSF under Grant No. IIS-1750193.

References

Xie Chen, Xunying Liu, Mark J.F. Gales, and Philip C. Woodland. 2015. Recurrent neural network language model training with noise contrastive estimation for speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP '15, pages 5411–5415.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 76–87. Association for Computational Linguistics.

Carrie Demmans Epp, Justin Djordjevic, Shimu Wu, Karyn Moffatt, and Ronald M Baecker. 2012. Towards providing just-in-time vocabulary support for assistive and augmentative communication. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, pages 33–36. ACM.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2016. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*.

D. Jeffery Higginbotham, Howard Shane, Susanne Russell, and Kevin Caves. 2007. Access to AAC: Present, past, and future. *Augmentative and Alternative Communication*, 23(3):243–257.

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

Shaun K Kane, Barbara Linam-Church, Kyle Althoff, and Denise McCall. 2012. What we talk about: designing a context-aware communication tool for people with aphasia. In *Proceedings of the 14th international ACM SIGACCESS conference on Computers and accessibility*, pages 49–56. ACM.

Shaun K Kane and Meredith Ringel Morris. 2017. Let's talk about x: Combining image recognition and eye gaze to support conversation for people with ALS. In *Proceedings of the 2017 Conference on Designing Interactive Systems*, pages 129–134. ACM.

Tomáš Mikolov, Anoop Deoras, Stefan Kombrink, Lukas Burget, and Jan Cernockỳ. 2011a. Empirical evaluation and combination of advanced language modeling techniques. In *Proceedings of the International Conference on Spoken Language Processing*, pages 605–608.

Tomáš Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. 2014. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2011b. Extensions of recurrent neural network language model. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP '11, pages 5528–5531.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 220–224, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard Simpson, Heidi Koester, and Ed LoPresti. 2006. Evaluation of an adaptive row/column scanning system. *Technology and disability*, 18(3):127–138.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Seventh International Conference on Spoken Language Processing*, pages 901–904, Denver, CO.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. 2011. SRILM at sixteen: Update and outlook. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*, volume 5 of *ASRU '11*.

Keith Trnka, John McCaw, Debra Yarrington, Kathleen F McCoy, and Christopher Pennington. 2009. User interaction with word prediction: The effects of prediction quality. *ACM Transactions on Accessible Computing (TACCESS)*, 1(3):17.

Keith Vertanen. 2017. Towards improving predictive aac using crowdsourced dialogues and partner context. In *ASSETS '17: Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility (poster)*, pages 347–348.

Keith Vertanen and Per Ola Kristensson. 2011. The imagination of crowds: Conversational AAC language modeling using crowdsourcing and large data sources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP'11, pages 700–711. Association for Computational Linguistics.

Tonio Wandmacher, Jean-Yves Antoine, Franck Poirier, and Jean-Paul Départe. 2008. Sibylle, an assistive communication system adapting to the context and its user. *ACM Transactions on Accessible Computing (TACCESS)*, 1(1):6.

Bruce Wisenburn and D. Jeffery Higginbotham. 2008. An AAC application using speaking partner speech recognition to automatically produce contextually relevant utterances: objective results. *Augmentative and Alternative Communication*, 24(2):100–109.

Bruce Wisenburn and D. Jeffery Higginbotham. 2009. Participant evaluations of rate and communication efficacy of an AAC application using natural language processing. *Augmentative and Alternative Communication*, 25(2):78–89.