EchoTree: Engaged Conversation when Capabilities are Limited

Andreas Paepcke
Stanford University
Stanford, CA
paepcke@cs.stanford.edu

Sanjay R. Kairam
Stanford University
Stanford, CA
{skairam@cs.stanford.edu

ABSTRACT

Speech and motor-impaired individuals using assistive technologies to communicate face a common problem; the inevitable conversational 'dead space' which occurs while the user generates written or artificially-spoken sentences can dramatically reduce conversation quality and partner engagement. To address this problem, we introduce *EchoTree*, a visualization approach employing predictive language modeling to generate and display candidate utterances, thus allowing partners to make guesses about the impaired user's intended conversational direction. By including partners in the process of sentence generation, EchoTree mitigates the dead space problem by (1) engaging partners in a collaborative activity, and (2) reducing the task of word-generation to that of confirming a word choice. Our implementation is browser-based, making it suitable for face-to-face or remote communication, as multiple parties can view and interact with the same stream of EchoTrees from desktops, tablets, or smartphones. We describe the visual and system design choices and the experiments with data sources and language models which led to our current implementation.

Author Keywords

assistive technology; prolonged engagement; collaborative conversation; story telling, game

ACM Classification Keywords

H.5.2 Information Interfaces And Presentation: User Interfaces - Interaction styles

General Terms

Human Factors; Design

INTRODUCTION

Before introducing EchoTree, we start by introducing 'Henry', a motion and speech-impaired individual who acted as both a motivation and a design partner in this endeavor. While Henry enjoys conversing, his speech impairment is complete. Quadriplegia, while severely limiting, does allow Henry to move his head in affirmation or negation, and his hand can operate a mouse button.

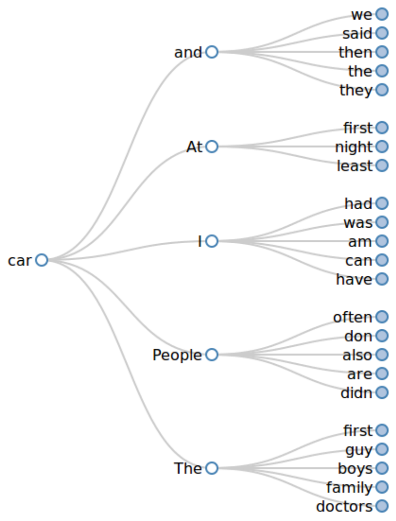

Figure 1. Screenshot of an EchoTree.

One of Henrys communication modes is a text-to-speech system (*tts*). This system works via a camera mounted on top of his laptop, which tracks a confetti-sized white dot pasted on the lower left of his glasses. The resulting cursor allows Henry to hunt down the keys of an onscreen keyboard. On a very good day, the resulting speed is 15 words per minute. The tts produces sound only once a sentence has been completed; uttering words as Henry types them could work, but this approach makes it difficult for listeners to track the very slowly evolving sentence in their mind. In addition, poor tts performance for some words would further impede comprehension, which is supported by the context available when a full sentence in pronounced together.

This slow communication channel results in very frustrating experiences during gatherings like parties. A guest will make a remark to Henry, who will go to work on an answer. To the conversation partner, Henry looks frozen, peering at his laptop screen, the back surface of which reveals nothing to the expectant partner. Often, the potential conversation partner wanders off bewildered before Henry can finish his sentence. While a second display would allow a listener to understand that a response is forthcoming, such a solution would still keep listeners passive, and likely bored.

In an attempt to ameliorate this situation for users such as Henry, we have developed a collaborative visual communication approach – *EchoTree*, an example of which is shown in Figure 1. As the impaired typist generates written words using an existing device, such as a laptop, an underlying language model predicts the conversation threads most likely to follow each typed word; these multiple possibilities are presented on a display using a WordTree[17] layout. Conversation partners can view and suggest possible sentence completions to the typist, possibly saving time and speeding the conversation.

In this paper, we present the EchoTree approach for the first time. We describe the visual design, user experience, and system architecture of our current implementation, a browser-based visualization which can be accessed from any web-enabled device, allowing for both face-to-face and remote communication. In addition, we present the results of an experiment aimed at understanding which data sources and modeling choices where best suited for generating the underlying language model. Finally, we discuss our findings, how they fit into existing knowledge about facilitating communication for users with speech and motor impairments, and opportunities for future work.

USER EXPERIENCE

A word tree[17] is read from left to right, beginning with a single word, the *root*. Branching out from the root are words that might follow the root in an underlying document collection. EchoTree provides five out-branches, which are sorted top to bottom by their likelihood of being the follower word to the root. Each follower word candidate itself features five possible followers to the candidate. The tree thereby presents a number of possible conversation threads that might begin with the root word.

The underlying collection, of course, impacts the follower relationship probabilities. We analyze three types of collections in the experiment section.

EchoTrees are browser applications that can be viewed anywhere, by multiple users. In particular, every word a typist enters on their laptop induces an EchoTree, which is made available by an EchoTree server as an interactive Web application. The typist can see the trees as well. If the word that the typist has in mind to type next is contained in the tree, they can click on the word. The word is transferred to the *sentence box* near the display bottom, which collects the evolving sentence, and saving on cumbersome text input. Once the typist finishes the next word, the previous EchoTree is replaced with a new one, rooted at the latest word. Figure 2 shows highlighted three words that were transferred to an input box by clicking on them, one after the other. Beyond providing a word source for a disabled typist to choose from, this arrangement also enables a number of scenarios where the typist communicates with conversation partners who have access to the trees. The important point about each of these scenarios is that conversation partners remain engaged in the conversation.

For example, the secondary display facing a conversation partner in the option discussed earlier could always show the typists's current EchoTree. Alternatively, a conversation partner's smartphone or tablet can *tune in* to a typist's EchoTrees. In either case, informed by the EchoTree, the partner can

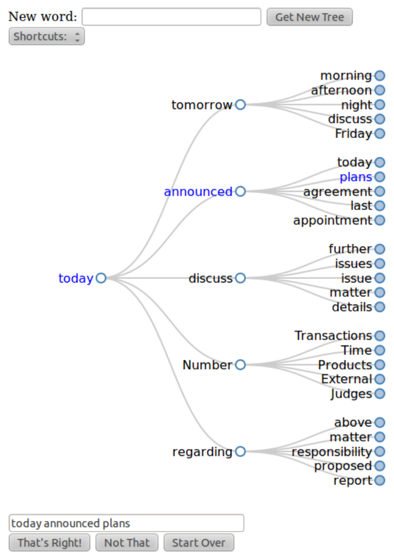

Figure 2. EchoTree re-rooted in 'today'. A click selected the blue words adding them to the sentence box.

guess future words out loud, again saving on typing and enlivening the conversation.

Or, partners can at least spend time exploring EchoTrees on their own while the typist works: The server allows multiple participants to generate their own, separate trees, visible only on their own displays, while still seeing the typist's trees as they are published. The typist's trees are pushed to the client browsers when a new tree becomes available, because the typist finished a word. The pushed trees replace any tree currently displayed on the participant's device.

EchoTree delivery to portable devices is thus subscription based. Participants can in fact subscribe to the EchoTrees of one or more other participants. The primary scenario for this paper is, however, to have conversation participants subscribed just to a disabled person.

The typist or a participant may click or tap on one of the circles in the tree that currently occupies their display. In response, the tree is *re-rooted*: the selected word becomes the root of a new tree. All follow words are recomputed, and a new tree is displayed on the participant's browser, or any other browsers that are subscribed to trees by the person who generated the new tree.

Alternatively, one may type a new word into the text box at the top, and click on the button *Get New Tree*. This action again creates a new tree, rooted at the new word, and displayed everywhere.

The EchoTree facility can be used for a number of purposes. In the context of a disabled typist interacting in a conversation, the facility is used as follows.

Collaborative Conversing

As the typist enters words, and corresponding EchoTrees in the browsers of all tuned in listeners evolve, any word that the typist completes is additionally appended to the sentence box

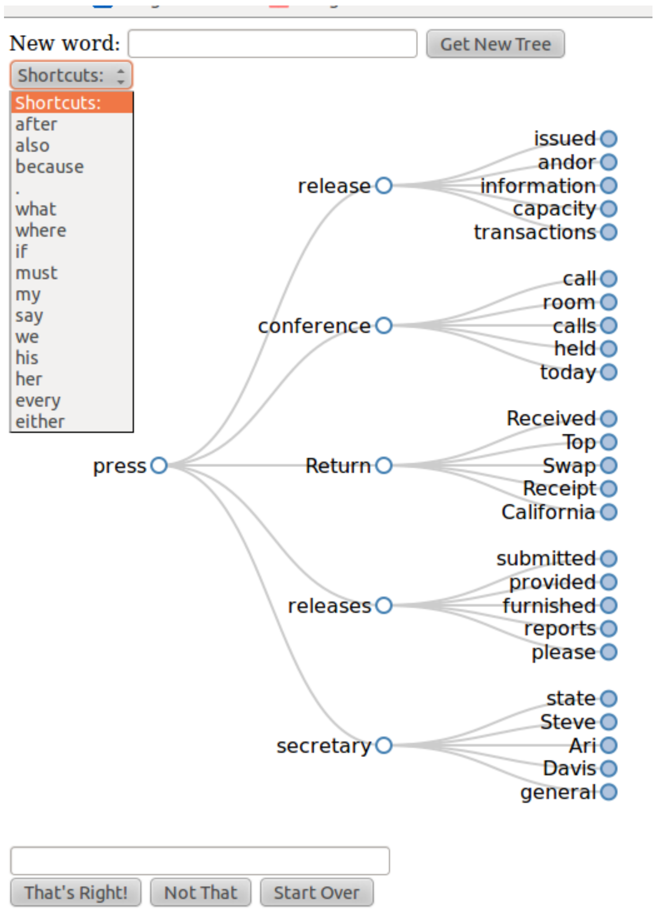

Figure 3. Shortcut words are available as fillers for the sentence box.

of all participants' screens. As listeners actively think ahead, guess where the typist might be headed, and call out a correct option, the typist can click on the *That's Right* button, or nod. After the successful guess the typist continues, skipping one of more words.

Sometimes participants or the typist may wish to enrich sentences with fill words. The pull-down menu below the *New word* field satisfies that need (Figure 3). Selecting any of these words will enter them in the sentence box. Again, in the current implementation this addition appears in all subscribing views of the. Note that the use of EchoTrees for collaborative conversation is not limited to face-to-face situations. Communication with the disabled person via the telephone are also an option. The remote participant tunes into the typist's EchoTrees, and offers guesses over the phone. Since the typist nodding assent is not an option in this scenario, the *Not That* button can serve as a negative response.

Architecture

Figure 4 shows how the EchoTree system is constructed. Central, or distributed EchoTree servers each manage some number of distinct EchoTree channels. All facilities described above operate on one channel. That is all shared EchoTree views are refreshed on that channel, and requests

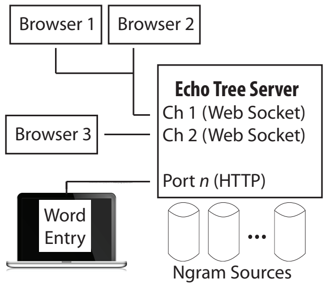

Figure 4. EchoTree architecture. Channels are implemented as Web-Socket ports. Browsers tune in to different EchoTree channels.

for re-rooting occur on that channel. The server computes trees, given a root word. Once computed, the new tree is pushed to all subscribers.

Multiple, unrelated EchoTree sequences may be served by a single server, using different ports. In Figure 4 Browser three is separated from Browsers one and two, which share mutual EchoTree transmissions.

Browsers communicate with EchoTree servers via Web-Socket connections, which are bi-directional. This bidirectionality enables the re-rooting requests from browsers back to the server.

Figure 4 also shows an HTTP port family. These ports are used to push new root words to the echo server from sources other than browsers. Non-standard, desktop based text input applications for disabled typists do not feature Web socket capabilities. The ports can be used instead via standard socket operations. The server listening on those ports interacts with the applications (or bridges to the applications) via the same protocol as the browsers use over Web sockets.

Like new trees created in response to re-rooting requests from browsers, the trees created in response to applications' requests trigger the multicast of a new EchoTree to all browsers on the respective channel. This method allows typists to focus on operating in their usual environment, not being forced to interact with a browser's *New Word* entry to push new root words.

At its bottom, Figure 4 shows a series of databases with word follower frequencies that are the basis for the generation of the trees. The resources in these databases consist of *ngrams* with associated occurrence probabilities. That is, each database holds one or more lists of entries consisting of a probability p, and two or more words, w_0, w_1, $w_2 \cdots$

For example, one snippet of such a list might look like this:

```
1.294000e-05,circumstances,like
1.294000e-05,He,just
1.050000e-04,while,I
1.294000e-05,caused,when
1.294000e-05,not,He
```

The probability informs the EchoTree construction how likely it is that the sequence w_1, $w_2 \cdots$ follows w_0. The probabilities originate from any text collection. But as we will show,

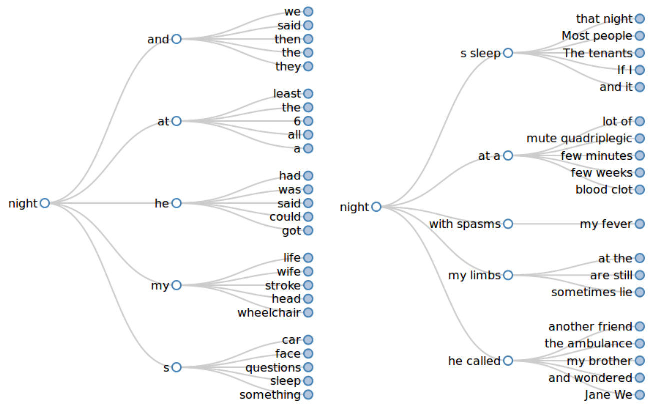

Figure 5. Comparing bigram and trigram tree displays.

the collections, and the choice of ngram arity have important impacts on the generated trees. The EchoTree of Figure 2, for example, is based on bigrams from the Enron email collection [12].

Given a new root word, the EchoTree constructor queries one of the underlying databases for the top five list entries whose w_0 is the root word. The choice of which list is consulted is determined by the channel through which the re-root request was delivered, that is on the originating subscription.

Each probability/ngram list is constructed once from one textual source, using the Good-Turing procedure [7]. This treatment smoothes the otherwise spikey ngram distribution, and sets aside some probability mass for ngrams not encountered in the underlying collection.

Cornerstones of EchoTree's system level design are the choices of underlying data sources. But other system level desicions impact the user experience as well. While we address the data source decision in this paper, the following section points to some of these additional considerations.

DESIGN CONSIDERATIONS

We now illustrate how the language model that generates EchoTrees plays into the user experience. As one more example for the details that must be considered, we then look at the issue of stopword removal, to be followed by a section on data source issues that takes us to the associated experiment.

Language Model

The EchoTree word predictions are based on a model of the English language. Working with ngrams has served us well. Both bigrams and trigrams are common choices in Natural Language Processing. Both options produce plausible word trees, but the display is by necessity more cluttered when trigrams are shown. Figure 5 shows a comparison between the two options, using the same root word, *night*. The trigram tree is more expressive than the bigram tree, but it is visually denser. To a disabled typist, finding a word in the trigram display with the intention of then transfering it to the sentence box is more time consuming than scanning the bigram version. Depending on the length of the sought word, the distraction time away from the onscreen keyboard may outweigh the savings in typing.

The distraction time not only comprises the time it takes to visually scan the tree, and to move the cursor to a found word via, for example, a head tracker. But the onscreen keyboard must then be re-acquired for further typing.

Word length, which determines how profitable a cursor excursion to an EchoTree might be, in turn is related to another design choice, the retention or elimination of stop words.

Stopwords

When designing information retrieval facilities, certain very frequently occurring words are often disregarded in user queries, and in underlying index structures. These *stopwords* include words like 'the', 'that', and 'a'. Is stopword elimination appropriate for our purpose of collaborative conversation? We experimented with both options, and included examples for both in the figures. Figure 1 was created while retaining stopwords, while Figure 2 was constructed with stopwords removed.

Clearly, stopword removal tends to display more interesting words. However, the more complete sentence structure borne from stopword retention makes it easier for conversation partners to call out the next word choice, thus saving the disabled typist time, albeit often for short words. Note that this time saving is enjoyed in full, because the typist's attention can continue to dwell on the onscreen keyboard. Short of a formal study, we have anecdotal evidence from Henry, who prefers to lay down all words, not leaving out stopwords in spite of the increased communication expense. A final decision on the stopword question awaits a study.

Data Source

As described, the probability of any word following another is derived from word occurrence statistics in an underlying data source. Which source to use? Intuitively, we might suspect that using a person's own written materials for the ngram statistics will be particularly effective in predicting that person's writing.

Three problems arise from this approach. For one, a disabled person's email messages, for example, will tend to be concise. This brevity in turn limits the amount of material over which ngrams can be computed.

A second problem with using personal writing, like email, is that obtaining the email collection is more difficult now that the material is stored on company servers, than when email was on everyone's personal disk. Not many email users would know how to download their entire email history from their email provider's server. In any event, such a startup effort is unfortunate for any computer application.

Finally, a third drawback of using personal writings is that EchoTrees can be quite revealing. The frequency with which a particular word follows another in one's entire personal corpus should not necessarily be open to public view.

At the other extreme, we can use a broad source, like Google's terabyte ngram collection [14]. Its coverage across millions of Web pages is more or less topic neutral, and thereby maybe universally applicable. Between these extremes lies the option of utilizing multiple ngram sets, each from one topic specific collection of Web pages. When requesting re-rooting, users would choose a configuration that is appropriately close to their topic of conversation. Maybe the topic specificity would be beneficial.

Or, one might argue that neither email, nor Web content, nor ngrams from scanned books [11] are optimal for a conversational context. What if our word choices in conversations is very different from those of our Web pages or email messages?

We conducted an experiment that examined the EchoTree performance of three datasources, each tested under both bigram and trigram design variants.

LANGAGE MODEL EVALUATION

The EchoTree system utilizes an ngram-based model trained on a corpus of sentences. In this section, we evaluate the performance of models built under several conditions. Specifically, we test the performance of bigram vs. trigram trees trained on one of three reference corpora (one of which was drawn from a corpus of Henry's writing), giving a total of six experimental conditions.

We evaluate performance using two dependent measures relevant to the target task – prediction reliability and saved typing time – detailed in the following subsection. Despite the smaller size of the personal corpus, we hypothesized that the presence of a consistent author would lead to better performance for models trained on these data.

Data Sources

Web: 10M web pages retrieved from the *Recreation* section of the Open Directory Project (ODP) [1], a collaboratively-curated web directory. We eliminated all pages linked directly from the ODP site, retaining pages one-level deep in the Recreation target sites. HTML and JavaScript was removed from the pages and the text was tokenized using the Stanford NLP Tokenizer [9], producing 5,055,284 bigrams and 28,423,891 trigrams after setting aside 10% as a test set.

Fisher: 11,000 transcripts of ten-minute telephone conversations [3, 4]. Each conversation centers around one topic, which was assigned to the paid participants in the conversation dyad. Metadata was removed and the text was tokenized, producing 149,789 bigrams and 93,534 trigrams after setting aside 10% to test.

Personal: 1,167 sentences (96.5 MB) extracted from a blog maintained by our collaborator, Henry, captured on May 10, 2013 [6]. This corpus produced 12,222 bigrams and 16,567 trigrams after setting aside 5% to test.

Performance Metrics

The first performance component we computed for each experimental condition is the *reliability* with which EchoTrees predict follow-on words for each successive word in test sentences. The test sentences for each condition were taken from collection set-asides of the condition's data source. No ngrams were collected from those set-asides.

In measuring the reliability of an experimental condition, our automated evaluation pulled one word w after another from each testing sentence S, and constructed an EchoTree with w as its root. If w's follower appeared in the tree at the tree's first level, the sentence was assigned one point. If the follower appeared at the second tree level, one half point was awarded to the sentence. The reliability measure for S was computed as the sum of awarded points, normalized to the length of the sentence. The data source's single reliability performance measure is the average of the individual sentence performances. More formally:

```
for all sentence $S$ in test sentences do
  for all word $w_i$ in $S$ do
    tree = computeEchoTree($w_i$);
    if $w_{i+1}$ in first level of tree then
      $score_s += 1$;
    else if $w_{i+1}$ in second level of tree then
      $score_s += 0.5$;
    end if
  end for
  normalize $score$ by length($S$);
end for
$reliability = \sum_s score_s / numSentences$;
```

The second performance measure estimates the *savings* in typing. This measure is the percentage of characters that would not need to be typed in a real life situation, because the respective words were available in EchoTrees. We counted spaces between words in the grand sum of letters to be typed. We also counted the click needed to cause words in the EchoTree to replicate down into the sentence box as a cost equivalent to typing one character. We did not consider the time required for acquiring a word in the tree, and then reacquiring the onscreen keyboard.

Procedure

The six experimental conditions arise from the combination of three corpus conditions (Web, Fisher, and Personal) and two ngram conditions (bigram and trigram). 60 sentences were randomly selected from the set-aside test set and tokenized to generate bigrams.

Using the procedure described in the Performance Metrics section, word trees were computed repeatedly for each word in the test sentence and then scored. This procedure produced a *csv* file with one line for each sentence, recording the two performance scores: typing savings (expressed as a percentage) and reliability in predicting subsequent words.

RESULTS

We used a one-way ANOVA to evaluate the differences among mean typing savings (first dependent variable) and among mean prediction reliabilities (second dependent variable) for the six experimental conditions. Although Levenes test found variances among the six groups to be

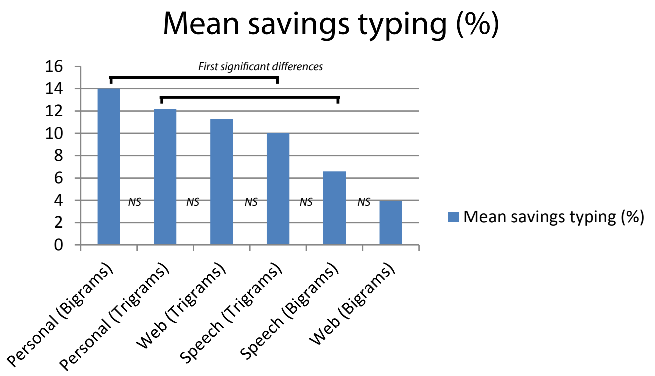

Figure 6. Mean savings in typing. No neighboring differences are significant (*ns*)

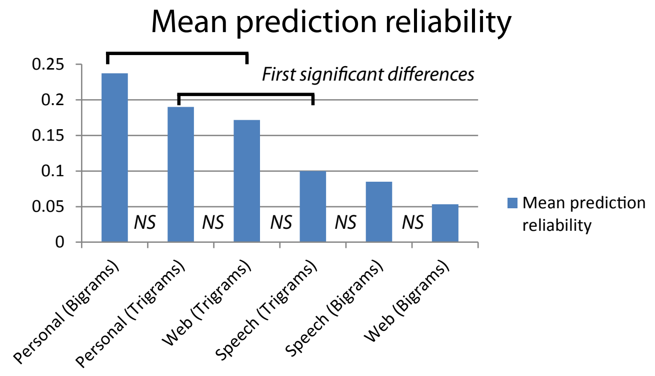

Figure 7. Mean prediction reliability. No neighboring differences are significant (*ns*)

different, we nevertheless opted for an ANOVA because our sample sizes were all equal (60 sentences).

We found a highly significant effect of the data source on savings in typing, $F(5, 354) = 16.0$, $p < .001$. These results are shown in Figure 6. The effect of data source on prediction reliability was also highly significant, $F(5, 354) = 25.4$, $p < .001$. These results are shown in Figure 7. Tukey HSD post-hoc tests revealed a number of significant between-group differences; these are shown in Tables 1 and 2.

As evaluated by both performance metrics, ngrams drawn from the personal corpus performed best. However, as shown in Figure 6, most of the differences between models were not significant; $Personal_{bi}$ significantly outperformed $Speech_{tri}$, $Speech_{bi}$, and Web_{bi}, while $Personal_{tri}$ only outperformed $Speech_{bi}$ and Web_{bi}. Generally, the results show that bigrams may not be the best option, unless there is a personal reference corpus on which to train. For trigrams, results were similar for Web_{tri} and $Personal_{tri}$, arguing for the use of trigrams in the case that a personal reference corpus is not available.

We were surprised by the relative strength of Web_{tri}. (Note that the difference between Web_{tri} and Web_{bi} is significant.). We did not expect the Web Recreation crawl to do well for either variant, because the raw results from a crawl are very 'messy.' Even after HTML tag removal, a large amount of formatting, and styling information is left over. Numerous menu headings, and boilerplate materials are present as well. We made no effort to clean the corpus beyond HTML tag removal.

Just to ensure that the random draw of Web_{tri} test sentences did not produce an unusual outcome, we repeated Web_{tri} with a different set of random 60 sentences from the respective set-aside. The result was comparable to Figure 6.

The prediction reliability (Figure 7) shows that the order of the six conditions by decreasing performance is as per the savings measure. A fundamental difference between the two measures, however, is that reliability does not depend on individual word lengths.

The best performer in reliability was $Personal_{bi}$ with just under .25 on the scale from 0 to 1. Again, for bigram trees no choice is better than $Personal_{bi}$. The next bigram choice is $Speech_{bi}$, which is a significantly worse performer for prediction. For trigrams, however, the equivalent choice of Web_{tri} is available.

In summary, the optimal choice for ngram data source is always the personal collection. This result holds for both bigram and trigram trees.

For bigrams, the next best choice is to use the Fisher conversation transcripts, albeit at large cost in typing savings and prediction reliability.

If trigrams are acceptable to end users, then the recreation Web crawl is an acceptable choice for both savings and reliability.

DISCUSSION

Our experiments with various data sources find that a bigram model based on the user's own communication data performs best with respect to prediction reliability and overall typing saved. In the absence of such a personal corpus, we have found that a trigram model trained on a web corpus of recreation pages performs reasonably well. Starting with such a web corpus and capturing communication through continued use of the system may serve as a valauble bootstrapping method for developing a personal corpus.

Our findings suggest possible design choices for improving our language model. We first note the high performance of the Web_{tri} corpus, biased towards a single topic, compared to the Fisher corpus, which ranges over many topics. Allowing the typist to choose from several topic-specific corpora based on conversation topic could lead to performance gains. For this reason, the EchoTree architecture (Figure 4) anticipates multiple ngram sources that can be switched dynamically. EchoTree could also be improved by techniques draw from the domain of natural language processing. Word stemming, for instance, would likely lead to improvements in prediction reliability, as well as identification of syntactic structures. Adding a learning component, such that ngram weights are adjusted for each user over time, could provide long-term gains, as well.

Clearly, end-user experiments are required as well to resolve

	$Personal_{bi}$ M=14.0 SD=9.8	$Personal_{tri}$ M=12.1 SD=8.4	$Speech_{bi}$ M=6.6 SD=7.5	$Speech_{tri}$ M=10.1 SD=8.0	Web_{bi} M=4.0 SD=3.5	Web_{tri} M=11.3 SD=5.3
$Personal_{bi}$;M=14.0,SD=9.8			p<.001	p<.045	p<.001	
$Personal_{tri}$;M=12.1,SD=8.4			p<.05		p<.001	
$Speech_{bi}$;M=6.6,SD=7.5	p<.001	p<.05				p<.05
$Speech_{tri}$;M=10.1,SD=8.0	p<.05				p<.001	
Web_{bi};M=4.0,SD=3.5	p<.001	p<.001		p<.001		p<.001
Web_{tri};M=11.3,SD=5.3			p<.05		p<.001	

Table 1. Post-hoc test results for savings in typing.

	$Personal_{bi}$ M=.237 SD=.145	$Personal_{tri}$ M=.190 SD=.119	$Speech_{bi}$ M=.085 SD=.143	$Speech_{tri}$ M=.010 SD=.091	Web_{bi} M=.053 SD=.052	Web_{tri} M=.172 SD=.082
personal bi M=.237,SD=.145			p<.001	p<.001	p<.001	p<.05
personal tri M=.190,SD=.119			p<.001	p<.001	p<.001	
speech bi M=.085,SD=.143	p<.001	p<.001				p<.001
speech tri M=.010,SD=.091	p<.001	p<.001				p<.05
web bi M=.053,SD=.052	p<.001	p<.001				p<.001
web tri M=.172,SD=.082	p<.05		p<.001	p<.05	p<.001	

Table 2. Post-hoc test results for prediction reliability.

at least two questions. First, are bigram or trigram trees the preferred visual representation for supporting conversations with the impaired typist? Improving comprehensibility on the conversation partners end will surely improve the quality of predictions and generation of rapport. Second, do conversation partners generate correct guesses that capture the typists intention but which differ from the literal suggestions given by EchoTree? If such associative guesses do occur, then the overall system – machine plus human(s) – could exceed the performance measures seen in our experiment.

RELATED WORK

Though our work was motivated by a desire to improve Henry's ability to converse, he is not alone in his needs as an individual with impaired speech. In the United States alone, roughly 2.8M people (1.2%) reported difficulty with speech in 2010, of which 523,000 (0.2

Advances in language modeling and prediction have been fruitful for AAC researchers; anticipating imminent letters, words, or even sentences can vastly reduce text input demands of impaired users. Systems such as Humsher [10], for example, have adapted letter-based text-entry systems developed for able-bodied users to significantly accelerate text entry for severely motor-impaired users.

Past research on n-gram prediction-based AAC systems has shown that the additional cognitive overhead imposed by having to monitor predictions can actually outweigh the typing savings, leading to slower communication overall [16, 8]. In a study aimed at addressing this issue, Trnka et al. stress the importance of the quality of the predictive model in overcoming this tradeoff [15]. For this reason, we focus our evaluation in this paper on optimizing aspects related to the language model, before conducting the user evalautions planned for future work.

EchoTree builds not only on prior research in n-gram word prediction, but also on prior work in technology-assisted co-construction of sentences. Prior efforts have shown how involving an able-bodied communication partner in the conversation loop to make guesses about the intended messages can complement the benefits offered by ngram language models [13].

Similarly, the visual design of EchoTree adapts the interactive "keyword-in-context" visualization technique embodied in the Word tree [17] from its original purpose of retrospective corpus studies to the new problem of collaborative text generation. The use of visual metaphors, direct interaction, and output-as-input techniques allow for tight coupling between conversation participants and the information display [2]. EchoTree's sentence box and confirmation loop further afford the process of grounding, enabling tighter coupling between the conversation partners themselves as they negotiate the evolving sentence together. [5].

CONCLUSION

In this paper, we introduced EchoTrees, a language modeling and visualization approach designed to engage conversation partners and faciltiate the speed and quality of communication for motion and speech-impaired users. As the impaired user types words, the past sentence context is displayed along with a WordTree visualizing likely possible future words. Our implementation of this approach as a browser-based application allows co-located or remote participants to view and select possible upcoming words, thus saving the typist time and effort. We discussed the design considerations and implementation details of our system, as well as the results of empirical evaluation of various data sources aimed at identifying the best means for constructing language models in this interactive conversation scenario.

The problem of social isolation for communication-impaired individuals is clearly difficult to solve in the general case.

However, through the design of EchoTree, we demonstrate that specialized computer-supported collaborative solutions can ameliorate some of the problems associated with this isolation. Computers are patient where conversation partners are not. By connecting the individual, the computer, and the conversation partners in an interactive loop, EchoTree can facilitate a more efficient, more enjoyable, and less isolating experience for users such as Henry.

REFERENCES

1. The Open Directory Project. http://www.dmoz.org/, April 2013.
2. Ahlberg, C., and Shneiderman, B. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. Human Factors in Computing Systems (CHI 1994), ACM (Boston, MA, USA, April 1994), 313–317.
3. Cieri, C., Graff, D., Kimball, O., Miller, D., and Walker, K. Fisher english training speech part 1 transcripts. http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2004T19, December 2004.
4. Cieri, C., Graff, D., Kimball, O., Miller, D., and Walker, K. Fisher english training part 2, transcripts. http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2005T19, April 2005.
5. Clark, H. H., and Brennan, S. A. *Perspectives on socially shared cognition*. No. 7. APA Books, 1991, ch. Grounding in communication.
6. Evans, H. Hevans. http://hevans-hevans.blogspot.com/, May 2013.
7. Gale, W. A. Good-turing smoothing without tears. *Journal of Quantitative Linguistics 2* (1995).
8. Koester, H.H.and Levine, S. Modeling the speed of text entry with a word prediction interface. *IEEE Transactions on Rehabilitation Engineering 2*, 3 (1994), 177–187.
9. Manning, C., Grow, T., Grenager, T., Finkel, J., and Bauer, J. Stanford tokenizer. http://nlp.stanford.edu/software/, 2013.
10. Poláček, O., Míkovec, Z., Sporka, A. J., and Slavík, P. Humsher: A predictive keyboard operated by humming. ASSETS '11, ACM (Dundee, Scotland, UK, October 2011), 75–82.
11. Project, T. A. N. C. The american national corpus. World-Wide Web, 2012. Archived by WebCite® at http://www.webcitation.org/6AnivCgHX. Accessed: 2012-09-19.
12. Project, T. C. The enron email dataset. World-Wide Web, August 2009. Archived by WebCite® at http://www.webcitation.org/6AniNSXC6. Accessed: 2012-09-19.
13. Roark, B., Folwer, A., Sproat, R., Gibbons, C., and Fried-Oken, M. Towards technology-assisted co-construction with communication partners. 2nd Workshop on Speech and Language Processing for Assistive Technologies (SLPAT '11), ACL (Edinburgh, Scotland, UK, July 2011), 22–31.
14. Thorsten Brants, A. F. Web 1t 5-gram version 1. http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T13, September 2006.
15. Trnka, K., McCaw, J., Yarrington, D., McCoy, K. F., and Pennington, C. User interaction with word prediction: The effects of prediction quality. *ACM Transactions on Accessible Computing 1*, 3 (February 2009), Article 17.
16. Venkatagiri, H. Efficiency of lexical prediction as a communication acceleration technique. *Augmentative and Alternative Communication 9* (1993), 161–167.
17. Wattenberg, M., and Viégas, F. B. The word tree, an interactive visual concordance. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (Nov/Dec 2008), 1221–1228.