A Design Engineering Approach for Quantitatively Exploring Context-Aware Sentence Retrieval for Nonspeaking Individuals with Motor Disabilities

Per Ola Kristensson, James Lilley
Department of Engineering
University of Cambridge
Cambridge, United Kingdom
{pok21,jal218}@cam.ac.uk

Rolf Black, Annalu Waller
School of Science and Engineering
University of Dundee
Dundee, United Kingdom
{r.black,a.waller}@dundee.ac.uk

ABSTRACT

Nonspeaking individuals with motor disabilities typically have very low communication rates. This paper proposes a design engineering approach for quantitatively exploring context-aware sentence retrieval as a promising complementary input interface, working in tandem with a word-prediction keyboard. We motivate the need for complementary design engineering methodology in the design of augmentative and alternative communication and explain how such methods can be used to gain additional design insights. We then study the theoretical performance envelopes of a context-aware sentence retrieval system, identifying potential keystroke savings as a function of the parameters of the subsystems, such as the accuracy of the underlying auto-complete word prediction algorithm and the accuracy of sensed context information under varying assumptions. We find that context-aware sentence retrieval has the potential to provide users with considerable improvements in keystroke savings under reasonable parameter assumptions of the underlying subsystems. This highlights how complementary design engineering methods can reveal additional insights into design for augmentative and alternative communication.

Author Keywords

Augmentative and alternative communication; design engineering; text entry; context-aware text entry; sentence prediction; information retrieval

CCS Concepts

•Human-centered computing → Accessibility systems and tools;

INTRODUCTION

Nonspeaking individuals with motor disabilities typically rely on augmentative and alternative communication (AAC) technologies to communicate. In this paper we focus on the subset of the target audience that is literate. A commonly used AAC device for this user group is a keyboard, either a physical keyboard or a touchscreen keyboard, with built-in auto-complete and word predictions. When the user has typed a word, phrase or sentence the user can speak the text using speech synthesis. However, compared to speaking rates of between 125 and 185 words per minute (wpm; with a word defined as a five consecutive characters including space), aided communication rates in general are reported at 8–10 wpm without acceleration methods such as prediction (for direct selection). Current acceleration techniques range from the use of abbreviations or word and phrase encoding to letter, word and phrase prediction. However, there is only a limited increase to rates of up to 12–18 wpm and usability issues such as having to scan word prediction lists visually present challenges when using these acceleration methods. Even with acceleration methods, rates rarely exceed 20 wpm [14, 4]. In practice, individual rates span a wide range and we encourage the reader to explore this range using the AT-Node search tool.[1]

Traditionally, word prediction presents the user with a list of words that can either be selected directly (touchscreen or scanning) or via keyboard shortcuts (for example, function keys). Although several research projects have demonstrated the potential of using conversational language models to achieve rates of up to 64 wpm, these rates were obtained under particular circumstances and have not been translated into practical applications [21].

In this work we make two contributions. First, we propose a conceptual design of context-aware sentence retrieval for nonspeaking individuals with motor disabilities. Second, we explain how methods from design engineering can be used in tandem with traditional AAC design to gain design insights at the conceptual design stage of an AAC method. We will explain why this approach is sometimes even necessary in order to progress certain AAC designs, such as context-aware sentence retrieval.

Context-Aware Sentence Retrieval

We propose augmenting AAC touchscreen keyboards with sentence suggestions retrieved from the user's set of previously spoken sentences. In addition to word predictions, the system

[1] https://kpr.pythonanywhere.com/q?diagnosis=*&interface=physical_keyboard

CHI '20, April 25–30, 2020, Honolulu, HI, USA.

ACM ISBN 978-1-4503-6708-0/20/04 ...$15.00.
http://dx.doi.org/10.1145/3313831.3376525

shows sentence suggestions on the display. If the sentences are relevant, the user can save valuable keystrokes by directly selecting the desired sentence. If the sentences are not relevant, the user may ignore them. Such a system would likely be unsuitable for regular mobile text entry, as users would spend a disproportionate amount of time scanning the sentence suggestions. However, when the communication rate is very low this effort is worthwhile—assuming it results in keystroke savings.

To improve the probability of a suitable sentence suggestion, we propose leveraging *context tags*. A context tag is a label that describes aspects of the context of a conversation, such as the location, current time or conversation partner.

Design Engineering for AAC

The second contribution in this paper is a consequence of the first contribution. It is extremely difficult to validate a context-aware sentence retrieval system with AAC users and this difficulty short-circuits established user-centred design practices. First, such a system needs to be bootstrapped with sentences for an individual user. Given the rate-limited nature of AAC communication, this may demand up to six months of usage of the device by an AAC user before the benefits of the system can be felt by the user. The logistics involved in having even a single AAC user, and their assistant/family/friend, switch to a new device for the eventual benefit of being provided more relevant sentences is very difficult (and perhaps ethically wrong) to carry out. The alternative solution of prefeeding old sentences to the system might work but these would have to have context tags assigned *post-hoc*, which are unlikely to be valid and such a procedure can not be validated without ground truth data. Finally, using proxy-users (people pretending to have dexterity issues) cannot replicate issues we observe in users who have involuntary and idiosyncratic movement.

As a result of the difficulty of validating such a system we explore an approach often taken in design engineering. Design engineering (sometimes called Engineering Design) is a methodology used in product and integrated system design which spans the entire continuum from elaboration of an initial solution-neutral problem statement to design concerns in manufacturing, support and disposal of a product or system.

Specifically, we elaborate a functional design of context-aware sentence retrieval and study a conceptual design of context-aware sentence retrieval in which the key function of the system (sentence prediction) has been translated to three candidate function carriers. We then identify a set of controllable and uncontrollable system parameters of this conceptual design and explore the viable efficacy of such a system by quantitative envelope analysis.

We build a surrogate context model and identify the relevant controllable and uncontrollable parameters of the model and vary them in an envelope analysis. This allows us to demonstrate that a context-aware sentence retrieval system can provide substantial keystroke savings ranging from 50–96%, depending on assumptions on word prediction accuracy, accuracy of context tagging and the level of sentence re-use.

The design engineering approach thus serves two purposes. First, it helps define requirements for important function carriers, in particular the required accuracy of auto-complete, sentence retrieval and context-tagging. Second, it provides information on the viability of this approach without a need to first build, deploy and monitor a system over a prolonged period of time.

Paper Structure

The rest of this paper is structured as follows. First we review prior work in AAC sentence prediction. Then we explain how we used approaches for design engineering to study the system at the conceptual design stage. Then we explain the approach and modeling assumptions of the work. Thereafter, we model the underlying subsystems and observe the achievable keystroke savings via envelope analysis. Finally, we discuss the implications of this work and conclude.

RELATED WORK

Phrase prediction, that is, the prediction of more than two words, has become a ubiquitous feature in many text entry systems and is probably most prominent in the Google search window, where a number of additional search terms are suggested once a character has been typed. Google uses current searches on its platform combined with the user's location and their previous searches.[2] Another example is writing suggestions in Google's email product Gmail.

In the 1970s, phrase-based AAC systems were thought to be better equipped in enabling users to communicate at faster communication rates. In these early Speech Generating Devices[3] (SGDs), phrases were encoded using either number codes (Phonic Ear) or mnemonic encoding using icon combinations for accessing pre-stored phrases (Minspeak) [18].

However, the retrieval of pre-stored phrases places additional cognitive burden onto the user and different approaches were investigated for allowing the user to find the appropriate phrases.

The TOPIC prototype by Arnott et al. [2] added semantic tags to each phrase to allow for easier (manual) retrieval [18]. TalksBac added communication partner tags to allow easier tailoring of communication according to the conversation partner [20].

Prior research suggests that well-designed utterance-based retrieval systems which link pragmatic features and user goals can lead to faster communication without losing coherence. The TALK system [21] is based on a pragmatic model which sees the progression of a conversation as a series of gradual shifts of perspectives relating to the speaker, time (past, present, future) and event-related information (what, where, who, how, and why). By changing perspective, the system can predict possible utterances. However, these systems rely on handcrafted sentences and the user needs to remember the conversational content and the location of this content.

[2] https://www.blog.google/products/search/how-google-autocomplete-works-search/

[3] Also referred to as Voice Output Communication Aids (VOCAs)

Whole-message retrieval has also benefited from linguistic prediction. Langer and Hickey [9] developed WordKeys, a research system which allows the user to retrieve pre-stored messages by typing or selecting a key word that can be associated with any word in the target message. For instance, the message 'I enjoy watching tennis' could be retrieved by entering the word 'ball', even though 'ball' does not appear in the sentence. The associations (for example, a ball is used in tennis) are automatically generated by a large semantic lexicon, derived from the WordNet database [15].

In general, retrieval of stored sentences has proven difficult and thus individual words became the main unit to be stored in AAC systems.

Users of current commercial SGDs sometimes pre-store utterances and monologue 'talks', but sharing personal experiences (stories) interactively using SGDs is rare. The main reasons for not using pre-stored utterances and talks despite the evidence that this improves communication rates can be described in terms of pragmatics (how language is used and the usability of the user interface):

1. It is unnatural for users to anticipate experiences which may be used within future conversations, resulting in limited pre-stored material.
2. Experiences are 'shared' as interactive conversational narratives and not as monologue 'talk'.
3. The cognitive and physical effort needed to retrieve pre-stored conversational information is high, resulting in users resorting to word-for-word typing—users have to invest effort into remembering where the vocabulary is stored, or in the case of word prediction, there is a cognitive overhead in visually scanning the options to find the desired item.

The need to resort to spelling and word-by-word production affects the spontaneous conversational flow and its quality because of the slow speed and physical effort to produce text.

Nandi and Jagadish [12] describe the two main challenges of phrase prediction:

1. The number of possible phrases is considerably larger than the number of possible words.
2. A phrase has no defined boundary compared to a word.

Phrase prediction can also be seen as word prediction with larger words where commonly co-located words are coded as one word in the prediction index, for example, New York becomes New_York. However, if the phrase corpus consists of phrases typed by the user it can be expected that a phrase boundary/ending in general is defined by the use of certain function or punctuation keys, such as 'speak' and 'full stop'.

Advances in context sensing technologies have opened up opportunities to refine word and sentence prediction using information such as location and conversational partner. It is anticipated that a so-called context-aware system will predict more appropriate lexical items, thereby reducing the cognitive and physical effort required to use word and utterance-based AAC.

Very early phrase-based systems, *Floor Grabber* and *TALK* [17], gave access to a large number of phrases grouped by conversation topic, such as 'Miami Trip' or 'my car'. These systems contained approximately 500 sentences that could be accessed through a topics-based user interface on a desktop computer. In a study exploring phrase prediction Garcia et al. [3] transcribed a participant's communication paper notebooks for a corpus of 545 sentences.

It is notable that no research-based prediction systems (standard and context-aware) have as of yet been reflected in commercially available systems. Although there is a small number of systems that use GPS data to pre-select phrase collections, these are restricted to needs-based conversations, for example, ordering phrases in a cafe or restaurant (for example, Locabulary[4] or TalkRocketGo[5]). Grid 3 by Smartbox can suggest previously typed phrases filtered by the location where the phrases were typed previously.[6] There is no information available on the number of phrases stored in these systems.

Commercially available systems with manual access to phrases usually group these by context (for example, *in a cafe*, *at the bank*, etc.) for selection of a suitable phrase. Systems providing phrase prediction suggest phrases that either begin with a typed letter, contain a typed word or are identified by an acronym.

Recent systematic reviews of the literature have found little evidence for phrase prediction being used: Koester *et al.* [8] reviewed 39 papers on text entry strategies. Although prediction was not at the heart of this review, the authors conclude that word prediction in the studies seldom allowed for an increased typing rate. Phrase prediction was not available in any of the studies. Polacek *et al.* [14] reviewed 150 publications on text input for users with motor impairments. They only refer to one of their own publications for prediction of longer text elements. No indication on effectiveness is given [13].

This highlights a mismatch between theoretical and practical values in typing rate improvements. Although theoretically improvements seem apparent for using prediction they do not seem to be reflected in practical studies with users with disabilities. All these systems have in common that a user needs to scan a list visually and read suggested phrases to be able to choose a suitable one, justifying the need for improved methods for predicting correct phrases.

In addition, the idea of leveraging context has been incorporated into AAC design before. An example is the work by Kane et al. [7] and others (for example, Mahmud et al. [1]). However, our use of context in this paper is different in that the context tags are here directly incorporated into an information retrieval model for retrieval of stored sentences.

FUNCTIONAL DESIGN AND FUNCTION CARRIERS

Design engineering is a set of methodologies for product and integrated system design. There are many specialized design

[4] http://locabulary.com/
[5] http://myvoiceaac.com/app/talkrocketgo/
[6] https://thinksmartbox.com/product/grid-3/

processes for various domains, such as medical devices or aerospace.

The relevant aspect of the design engineering process for this paper is the conceptual design stage. In this stage of the design, a product or integrated system is first described as a *functional design*. This typically means identifying the overall function (which in this case can be denoted as `Predict Text`) and the necessary sub-functions (such as `Create Context Tags`, `Auto-Complete Word`, `Retrieve Sentence`, `Display Word Predictions` and `Display Sentence Predictions`) and their interrelationships.

The second part of the conceptual design stage involves identifying solution principles for translating functions into function carriers. In this paper we focus on the conceptual design implications of a single critical function: `Retrieve Sentence`. This function is fundamental as the design of its function carrier (in other words, the implementation of sentence retrieval in software) is dependent on two other functions in particular: `Auto-Complete Word` and `Create Context Tags`. To understand the requirements of a function carrier for the function `Retrieve Sentence` it is necessary to evaluate a set of candidate function carriers to understand their relative suitability. To do this we identify the controllable and uncontrollable parameters such potential function carriers must be exposed to and vary these parameters to perform envelope analyses. These analyses provide information on some of the overall requirements the entire system must satisfy for these function carriers to be able to carry out the overall function satisfactory.

In this paper we study three function carriers, three well-established information retrieval algorithms, which are described in detail later in this paper.

What the design engineering approach does is allow us to identify critical functions and study aspects of some of them in isolation. This is normally not required in AAC design, but given the difficulty in validating context-aware sentence retrieval we alluded to in the introduction, in this particular case such an approach becomes critical to make progress.

Note in particular that we specifically are going to study the translation of a key sub-function (`Retrieve Sentence`) to a function carrier without any concern on how to translate other key functions, in particular `Auto-Complete Word` and `Create Context Tags`. Instead, an output of the analysis is a set of requirements which can later be used to make informed decisions of appropriate function carriers for these functions.

For brevity, we have only sketched a minimal functional design to motivate the quantitative parameter exploration we will carry out in the next sections in order to understand the potential of context-aware information retrieval for AAC. A complete functional design is out-of-scope for this paper, however, it is worth noting that certain functions, such as the function `Display Word Predictions`, can be studied in isolation using more traditional user-centred design methods.

QUANTITATIVE PARAMETER EXPLORATION

Our modeling approach treats the sentence retrieval problem as an information retrieval (IR) problem. We assume the AAC system contains a set of previously typed sentences, where each sentence may optionally be associated with one or more context tags. We treat both the words and the context tags as *terms* and the collections of terms for a sentence (including any associated context tags) form a *document*, where the terms *term* and *document* arise from the IR nomenclature.

In a similar vein, whenever the user desires to write a sentence, we call this a *query*. A query consists of context tags (if available) and any words the user has completed.

Both documents and queries are modeled as bags-of-words. A bag-of-words is a multiset of terms which, unlike a set, preserves multiplicity of the terms.

When investigating the performance of context-aware sentence retrieval we make an assumption that a hypothetical user has 500 prior sentences stored in their AAC system. This parameter choice is tentative but representative of AAC practice (see the related work section). We then typically select a sentence from this set of sentences (we will later in this paper also study typing sentences outside this stored sentence set) and model the user typing this sentence.

The user's typing is modeled both supported and unsupported by auto-complete assistance (word prediction). Without auto-complete assistance, a user must type each individual word to completion without aid. As a word is completed, it is added to the query as an additional term.

With auto-complete assistance, with some probability, the word is added to the query as a term for every keystroke typed that is part of the word.

Regardless of the typing method, for every keystroke typed we form a query with the terms consisting of any present context tags and the currently fully-typed, or predicted, words. We then score all sentences in the stored sentence set using three different IR algorithms (which are explained in the next section). Thereafter we rank all the sentences based on their relevance scores. If we have several sentences with equal relevance scores we choose a sentence at random. This models an interaction method where, as the user continues to type, the retrieved sentences will match the user intent better. This approach was inspired by our observations of actual AAC users retrieving stored sentences.

Unless otherwise stated, we retrieve the four best matching sentences and consider a match to be found if one of these four retrieved sentences matches the test sentence. We chose four sentences as there is sufficient space in a typical AAC touchscreen interface to display four sentences above the keyboard. We will later in the analyses in this paper also vary this number of sentences.

This typing process repeats until the system is able to retrieve a matching sentence. We then calculate the keystroke savings, KS, which is defined as:

$$KS = \left(1 - \frac{k_m}{k_c}\right) \times 100\%, \quad (1)$$

where k_m is the number of keystrokes that need to be typed before the model under investigation results in a matching sentence and k_c is the number of keystrokes in total for the test sentence. A higher keystroke savings value is better.

Unless otherwise stated, we use a set of 500 sentences from a publicly available AAC corpus [19] to model sentences stored in an AAC user's AAC device. We will select 40 sentences at random from this test set and use the above described model of typing behavior to simulate an AAC user typing the sentence on their AAC device. Before we retrieve sentences we pass all stored sentences and query terms through a porter stemmer, remove all punctuation and remove all capitalization.

SENTENCE RETRIEVAL ALGORITHMS

We investigate three algorithms for sentence retrieval: Inverse Document Frequency (IDF) [16], Best Matching (BM25) [5, 6] and a Unigram model. For completeness we define the models below. For more information on these models and how they are used in IR we refer to an IR textbook, such as for example Manning et al. [10]. We note that these IR algorithms are easily available, simple and robust. While more sophisticated algorithms exist, it is important to recall that we are performing envelope analyses and we should be conservative in our estimations.

IDF

IDF assigns each document in the collection a relevance score, and then ranks them accordingly. The relevance is given by the sum of each term's idf, defined as:

$$\sum_{t \in d} idf_t, \text{ where } idf_t = \log \frac{N}{n_t},$$

idf_t is a measure of the inverse document frequency. This is the log ratio of the size N of the collection and the number of documents n_t that contain term t. It is summed for every term in each document d to calculate the relevance score of that document.

BM25

The BM25 algorithm is a probabilistic model which also assigns each document a relevance score. The relevances are given by:

$$\sum_{t \in d} idf_t \cdot \frac{(k_1+1) \times tf_{td}}{k_1(1-b+b \times (L_d/L_{ave})) + tf_{td}},$$

where previously defined symbols remain the same as before, and tf_{td} is the term frequency of a term t in document d, L_d is the length of the document d, L_{ave} is the average document length in the collection, and k_1 and b are parameters which throughout this paper will be set to $k_1 = 1.2$ and $b = 0.75$.

Unigram

The Unigram model builds a dictionary of all terms in the collection. For each document it finds the probability of each term by summing its frequency of occurrence in the document and dividing by the total number of terms in the document. It then smooths these probabilities by adding a small constant α to the numerator and multiplying the denominator by α. This set of probabilities attached to each document is the Unigram language model based on each document.

For a given query q the algorithm calculates the product of the probabilities of each term in the query according to each document's language model. This acts as a relevance score with the highest product corresponding to the most relevant document.

The probability of a query containing m terms is given by:

$$P(q) = \prod_{i=1}^{m} P(t_i),$$

where each term probability is:

$$P(t_i) = \frac{\text{count}(t_i) + \alpha}{m \times \alpha}.$$

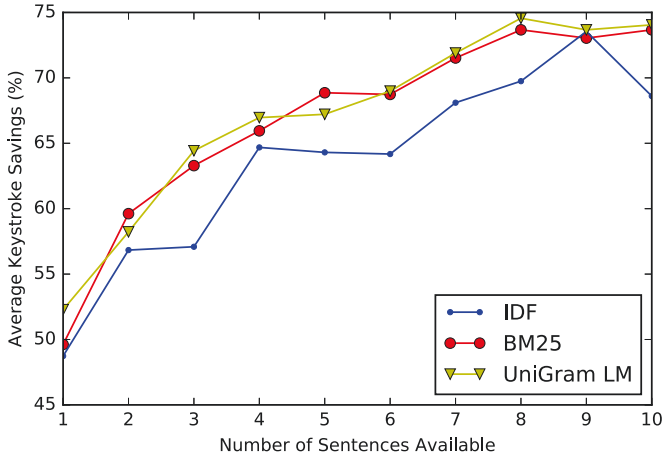


Figure 1. Average keystroke savings as a function of the number of sentence suggestions presented to the user for the three IR algorithms without using context tags and without auto-complete assistance.

BASELINE MODEL WITHOUT CONTEXT TAGS

We first investigate a baseline model without context tags where the user is typing the test sentence word by word without auto-complete assistance. Figure 1 shows the average keystroke savings obtainable as a function of the number of sentence suggestions that are shown to the user.

SURROGATE AAC CONTEXT MODEL

We then analyze if the baseline model results can be improved by assuming a context-aware AAC IR system. Since no large dataset of AAC sentences with context tags exists, real or surrogate, we need to create a surrogate AAC context model that can generate surrogate AAC sentence sets with associated context tags.

The basic form of this AAC context model is that we have a cache of context tags and these context tags are defined by the family of context tags they are part of, and their position

within the family. For example, an individual context tag may be of the form 'location1' or 'person2' and belong to a tag family, such as Location and Person, respectively.

As the user types, an actual system would provide a real-time feed of these context tags, for example by using face recognition to provide tags of the form 'person2' and GPS to provide tags of the form 'location1'. Tags could also be inputted manually by an assistant. These tags are then attached as terms to the start of the query. This means a query is pre-loaded with terms even before the user has begun typing.

To more fully define a surrogate AAC context model, which is a form of generative model, we introduce a number of parameters. We will split these parameters into two groups, *controllable* and *uncontrollable*. A parameter is controllable if it is essentially a design choice for any given system. In other words, a controllable parameter is a design parameter in the system and the identification of design parameter values capture requirements for implementing a well-functioning system. A parameter is uncontrollable if it will affect the system but cannot be directly controlled by the system. We will perform envelope analyses to both controllable and uncontrollable parameters. The envelope analyses of controllable parameters inform design objectives for their optimization. The envelope analyses of uncontrollable parameters demonstrate the likely variation in performance of any given system (a form of sensitivity analysis).

The controllable parameters are:

1. The number of context tags attached to each document. Unless otherwise stated; this parameter is set to 2.
2. How each family of context tags is represented in each document (stored sentence); this parameter is set as one context tag from each family in each document, unless otherwise stated.
3. The average number of context tags per context tag family; this parameter is set to 15, unless otherwise stated.
4. The spread of these context tags between the context tag families. Each context tag family will have the same size unless otherwise noted.

Note that the first and second parameters define the number of context tag families.

The uncontrollable parameters are:

1. The popularity of each individual context tag in each context tag family.
2. The probability of a context tag attached to a sentence being typed matching the context tag attached to the same sentence in the set of stored sentences.
3. The number of stored sentences (the size of the document collection).

ANALYZING UNCONTROLLABLE PARAMETERS

Popularity of Individual Context Tags

We first investigate how often each individual context tag will appear. This depends on how often the user, whose data we are assuming we have, is situated in different contexts and therefore we will not be able to control it and it will also most likely change between users. Therefore, we will first assume that within each context tag family the probability of each context tag being associated with each sentence follows a discrete power-law distribution. This assumption is motivated by the observation that even very simple generative typing processes will result in power law distributions [11]. We will then perform an envelope analysis of different power-law parameters in order to observe potential keystroke savings. For the rest of this paper we have set the power-law parameter to 4 as this proved a robust choice.

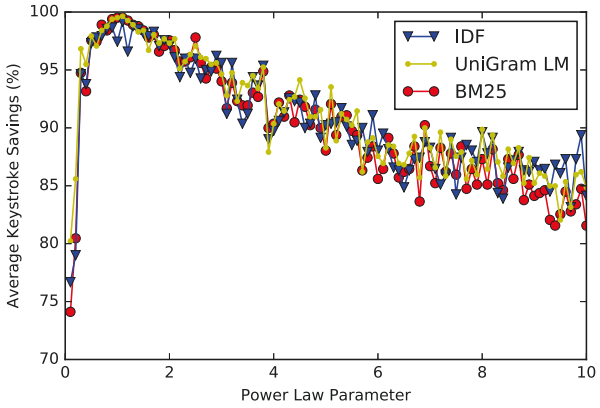


Figure 2. Average keystroke savings for three IR algorithms as a function of varying the power law parameter ω (without auto-complete assistance).

The power-law distribution is defined by the density function:

$$P(x, \omega) = \omega x^{\omega-1} \mid x \in \mathbb{Z}^+ \text{ and } x \leq \gamma,$$

where x is the position of the context tag within the family (e.g., 'location**1**' 'location**2**'), ω is a parameter that defines the distribution and γ is the number of tags in the family. The effect of varying the power law parameter is shown in Figure 2.

Probability of Matching Context Tagging

Another uncontrollable parameter is the probability of a test sentence's context tags matching the context tags of a sentence among the stored sentences. This parameter covers a range of different real-world scenarios where either the tag has been incorrectly assigned by the context tagging system (possibly due to classification errors) or the user is writing a sentence that exists among the stored sentences but in a new context and therefore the context tagging system may be either correct or incorrect but nonetheless the assigned context tag is different.

We perform an envelope analysis of this parameter as follows. Each time a test sentence is chosen from the collection, we sample from a uniform random distribution and if the sample is higher than a set probability parameter a context tag in the test sentence is randomly reassigned according to the previously defined power-law model. Otherwise, the context tag in the test sentence is left as it is. We vary the probability parameter between 0 and 1. Note that for the rest of this paper we are effectively setting the probability parameter to 1, which

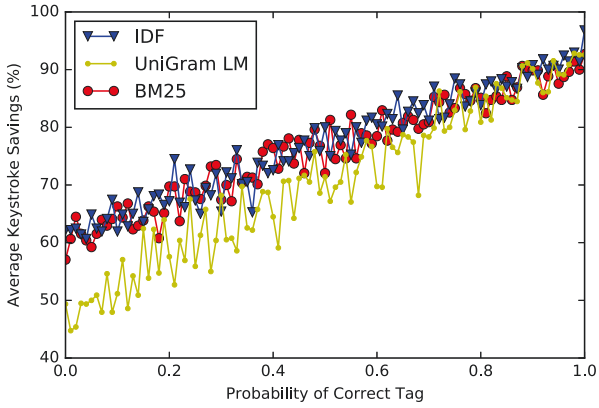


Figure 3. Average keystroke savings as a function of varying the probability of matching context tagging without auto-complete assistance.

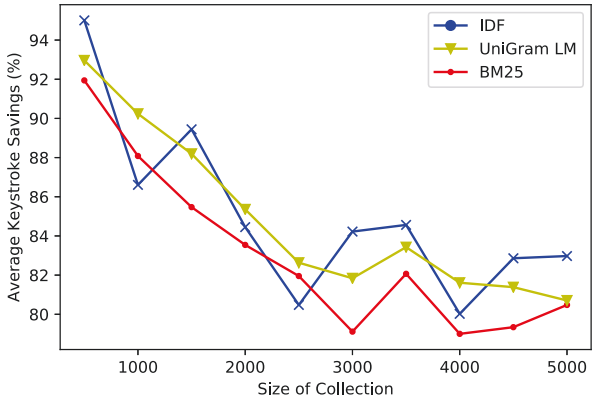


Figure 4. Average keystroke savings as a function of varying the number of stored sentences without auto-complete assistance.

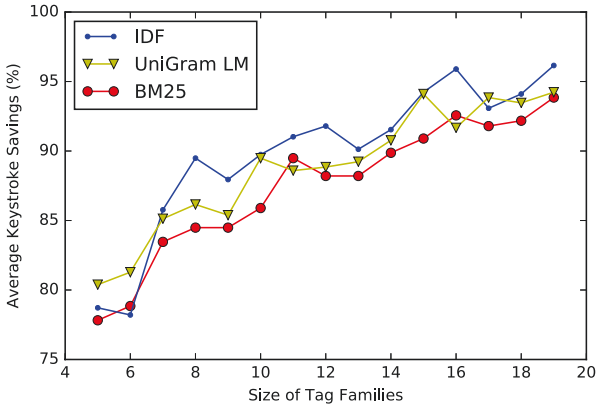


Figure 5. Average keystroke savings as a function of varying the number of context tags per context tag family without auto-complete assistance.

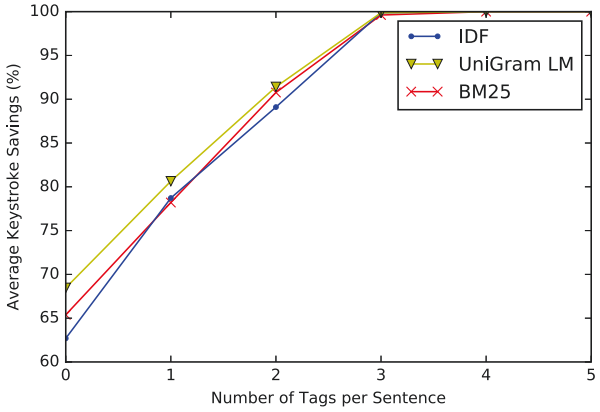


Figure 6. Average keystroke savings as a function of varying the number of context tags per sentence without auto-complete assistance.

assumes perfect context tagging. The results from varying the probability parameter are shown in Figure 3.

Stored Sentences

The final uncontrollable parameter is the number of stored sentences in the AAC system (or the number of documents using IR terminology). Figure 4 shows the effect of this parameter when varying it between 500 and 5,000 sentences.

OPTIMIZING CONTROLLABLE PARAMETERS

The analysis of the uncontrollable parameters revealed the performance gains that are likely to be obtained in an unoptimized system. However, further gains can possibly be made by optimizing the controllable parameters. We carried out four investigations:

1. The effect of varying the size of the context tag families.
2. The effect of varying the number of context tags per sentence (document).
3. The effect of unevenly assigning the context tag families to sentences. We begin by using only one context tag family and thereafter increasing the number of context tags per sentence (document), and then, using two context tag families, assigning one context tag from one of them, and then vary the number of context tags per sentence (document), adding context tags only from the other context tag family.
4. The effect of assigning unevenly sized context tag families. This is defined by varying an offset parameter with one context tag family increasing its number of context tags by the offset amount and another context tag family decreasing by the same amount.

The respective results are shown in Figures 5–8. As is evident in the envelopes, there are design parameter choices that can further improve overall system performance in terms of average keystroke savings. These insights can be captured as part of an improved requirements specification of the overall system.

EFFECT OF WORD PREDICTION

Throughout this paper the envelope analyses are based on the user typing individual words to completion without auto-complete assistance, also known as word prediction. This provides easily interpretable baseline keystroke savings and demonstrate the substantial keystroke savings that can be achieved using even modest context tagging.

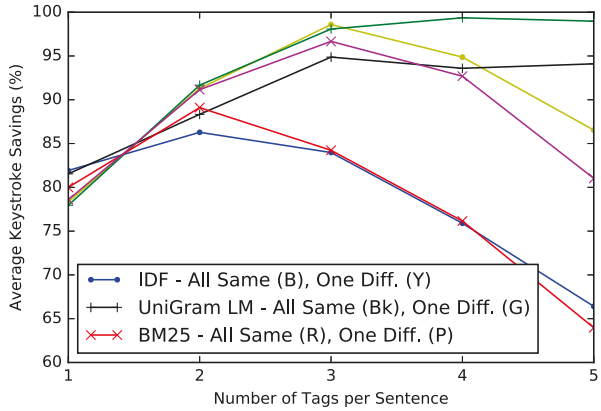


Figure 7. Average keystroke savings as a function of varying the number of context tags per sentence without auto-complete assistance.

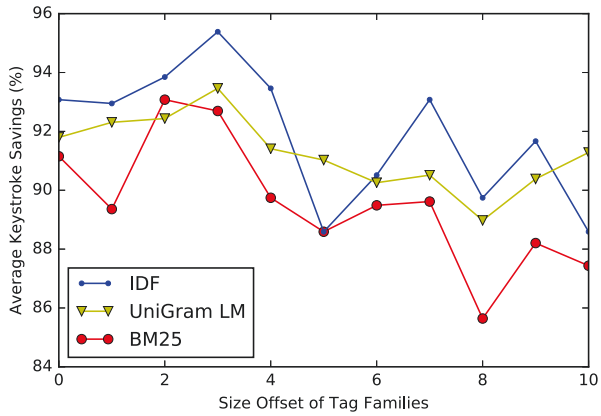


Figure 8. Average keystroke savings as a function of the size offset of the context tag families without auto-complete assistance.

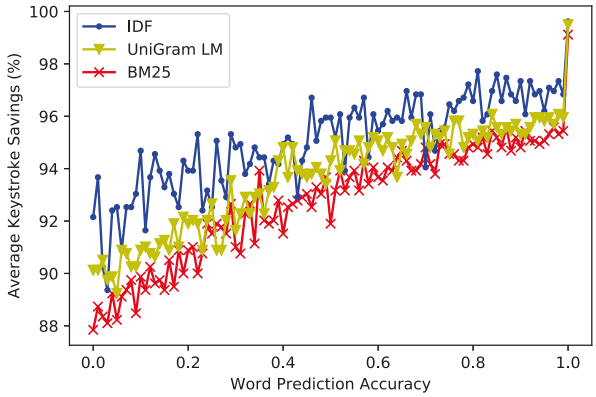


Figure 9. Average keystroke savings with auto-complete assistance as a function of word prediction accuracy for data with context tags.

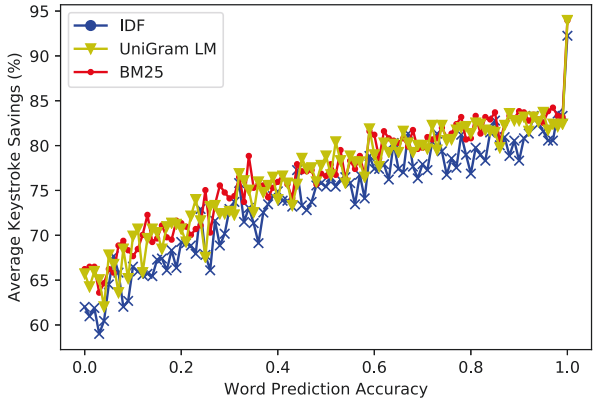


Figure 10. Average keystroke savings with auto-complete assistance as a function of word prediction accuracy for data without context tags.

However, AAC users typically rely on auto-complete[7] and we therefore also studied the impact of auto-complete on keystroke savings. This allows us to capture requirements on the important function `Auto-Complete Word`, which we identified in the conceptual design of the system earlier in this paper.

We model auto-complete with a word prediction *accuracy* parameter which determines the probability that for a given keystroke the intended word would be auto-completed. This model is a simplification of a true auto-complete system as we thereby assume the probability of auto-complete assistance predicting the intended word as being the same regardless of how many keystrokes the user has typed. In reality, the probability of an intended word being accurately predicted will increase as a function of the number of the keystrokes. However, as we perform an envelope analysis this effect will, in the limit, average out. Setting a fixed probability is a compromise between over-parameterization and simplicity (Occam's razor) and represents an "uninformative prior" that avoids us having to make elaborate distributional assumptions, which would be difficult to justify.

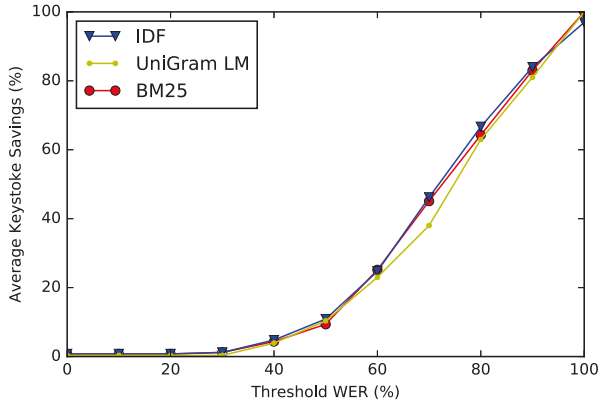
We carry out the envelope analysis both using context tags and without to assess the effect of keystroke savings with auto-complete assistance using a non-context aware system and a context-aware system. Figure 9 shows that under very reasonable estimations of word prediction accuracy (around the 80% word prediction accuracy point), the average keystroke savings are substantial at around 96%, assuming two context tags are used and they both match the context tags of the test sentence. In reality such a high performance will reduce in practice, as the context tags will inevitably not always match.

For calibration, Figure 10 repeats the analysis without the aid of context tags. At the 80% word prediction accuracy point, performance drops to around 80%.

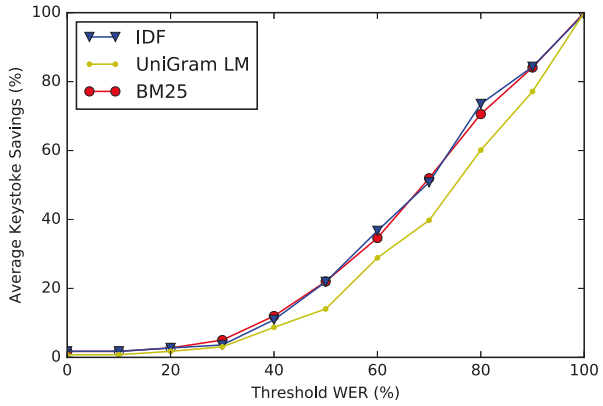
UNOBSERVED SENTENCES

We have observed that under reasonable parameter assumptions, two context tags per sentence, a stored set of 500 sentences, and each context tag family having 15 members, substantial keystroke savings can be achieved by retrieving a stored sentence using standard IR algorithms. Word prediction further improves the performance.

[7]To be precise, most AAC users rely on typing assisted by word prediction, which is not the same as typing assisted by auto-complete. We assume AAC users are willing and able to use auto-complete in this analysis.



(a) 500 stored sentences



(b) 4,500 stored sentences

Figure 11. Average keystroke savings as a function of thresholded WER when the system is requested to retrieve unobserved sentences from a) 500 stored sentences; and b) 4,500 stored sentences.

The reason the keystroke savings are substantial is because we are retrieving sentences that are already stored in the system. Such retrieval is common in AAC and therefore motivates the analysis. However, we were also interested in investigating how well a context-aware sentence retrieval system would be able to predict unobserved sentences, that is, sentences that are not stored in the system. Such unobserved sentences are by definition unlikely to perfectly match stored sentences. However, the system could find similar sentences and occasionally these sentences may be useful to the user. For example, if the user is attempting to communicate: "I want to go to school", a completion of: "Can I go to school", might be perceived as good enough by an individual user.

To get some indication of the performance of using the system in this mode, we defined a permissible word error rate (WER) which determined whether or not a retrieved sentence was considered as correct. WER is the minimum number of word insertions, deletions and substitutions necessary to transform a source sentence into a target sentence, divided by the number of words in the target sentence. We perform an envelope analysis by varying this WER threshold and observe the average keystroke savings.

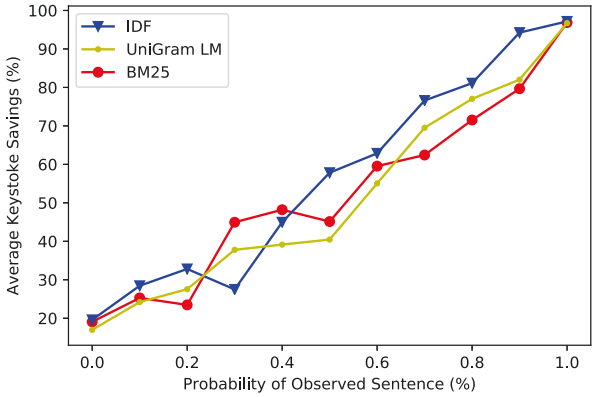


Figure 12. Average keystroke savings as a function of probability of observed sentences and correct tags with auto-complete set to 80% for sentences inside the sentence cache and 20% for sentences outside the sentence cache.

The test sentences were drawn from the same source as the other sentences in our previous analyses [19] but not present among the stored sentences. The total number of sentences in this sentence set source [19] is 5,000. While this set is small, the sentences in this set also tend to be distinct in terms of word usage and topic and therefore an unobserved sentence is unlikely to match a stored sentence well. In this sense, our analysis is conservative and erring on the side of caution.

Since the analysis is dependent on the system in some sense getting "lucky" that a test sentence happens to reasonably match a stored sentence, we carried out this investigation for two sentence sets. First, a conservative number of 500 stored sentences and, second, a large number of 4,500 stored sentences. This allowed us to observe how much performance would improve if more sentences are available. Auto-complete was turned off and two perfect context tags were used in this analysis. The results are shown in Figure 11. As expected, the threshold WER has to be relaxed to around 60–80% for the system to be able to generate meaningful keystroke savings. We also observe a slight increase in performance with the larger sentence set.

Figure 12 illustrates a more representative scenario. Figure 12 shows average keystroke savings as a function of the probability the user is typing an observed (stored) sentence for all IR algorithms. We set the threshold WER to 50%, assume perfect context tags and have added auto-complete. As with previous analyses, we set the auto-complete probability to 80% for stored sentences inside the sentence cache. However, such a high auto-complete probability is unrealistic for unobserved sentences outside the cache and we therefore lowered it to 20% for those sentences. Figure 12 shows expected keystroke savings ranging from 20% with 0% observed sentences to between 40% and 60% for 50% observed sentences depending on the IR algorithm.

Overall, a context-aware sentence retrieval system does not need a large degree of sophistication to provide tangible keystroke savings, even for sentences that are not present in the sentence cache.

DISCUSSION

Many nonspeaking individuals with motor disabilities rely on word prediction and simple sentence mechanisms to communicate. The communication rates are typically very low. In this work we have suggested a context-aware sentence retrieval model which leverages the fact that many AAC users reuse previously typed sentences. In our model the user starts typing a sentence as usual but is in addition to word predictions presented with a number of retrieved previously typed sentences.

We used a design engineering approach to identify the key functions and then decided to investigate three widely used IR algorithms which have open implementations and are easy to integrate into AAC systems: IDF, BM25 and a Unigram language model. We further assumed that very few context tags would be assigned to sentences (typically, two) and a relatively small number of retrieved sentences would be presented to the user (typically, four). These parameter choices are deliberately conservative.

We created a surrogate AAC context model and identified the controllable and uncontrollable parameters of the system. We then carried out envelope analyses that on the whole demonstrate very high keystroke savings assuming perfect context tagging. Assuming two perfect context tags, we typically obtain a keystroke savings range of 94–97% when the word prediction accuracy is assumed to reside at around 80%, which forms an important requirement on this particular sub-system.

We also investigated the sensitivity of these keystroke savings when context tagging was imperfect, and as demonstrated in Figure 3, even with a context tagging error of 50%, average keystroke savings are expected to be above 70%. This demonstrates how useful a context-aware sentence retrieval system can be for a nonspeaking individual with motor disabilities whose rate-limitation necessitates slow and perhaps inaccurate typing.

The high keystroke savings are achievable because the system is retrieving sentences that have already been typed before. For completeness, we also investigated hypothetical keystroke savings when a user typed sentences that were not stored in the system. First, we varied a threshold WER and experimented with two different sizes of the sentence sets, 500 and 4,500. We found that if a threshold WER between 60–80% is acceptable to the user, such a mode of operation can potentially result in 50% keystroke savings. Then we varied the level of sentence re-use. Figure 12 illustrates potential average keystroke savings with context tagging and auto-complete as a function of the level of sentence re-use.

Another important aspect we have demonstrated in this paper is the usefulness of functional design, envelope analysis and modeling in AAC. It would be very difficult to carry out traditional A/B testing or user-centred designs around a context-aware sentence retrieval system without first gaining an understanding of the controllable and uncontrollable parameters of the underlying model. The heterogeneity among the AAC user population and the fact that a context-aware sentence retrieval system would need to be field tested over a period of several months further necessitates a new approach in AAC text entry design. We hope this design engineering approach will inspire further research in this direction as a complementary method to traditional AAC user-centred design.

CONCLUSIONS

Many nonspeaking individuals with motor disabilities have difficulties communicating due to low communication rates. In this paper we suggest context-aware sentence retrieval as a potentially useful complementary technology alongside a touchscreen keyboard with word predictions. We have built a surrogate AAC context model and identified both controllable and uncontrollable parameters and visualized the design space using envelope analysis. We have shown that such a system can realize substantial keystroke savings ranging from 50–96%, depending on assumptions on word prediction accuracy, accuracy of context tagging and the level of sentence re-use.

The IR algorithms investigated in this paper are relatively simple and openly available, thereby making it easy for AAC device manufacturers to implement context-aware sentence retrieval functionality. Further, the context tags need not rely on overly sophisticated sensing. Context tags generated by, for example, GPS, the time of day and face identification are examples of context sensing which is widely available today. Our next step is to perform a longitudinal observational study with an AAC user group to study the nature of sentences stored, the accuracy of context tagging in practice and the realizable gains achieved in the field. We hope this work will find its way into AAC devices as a complementary technology that can provide substantial keystroke savings when the user simply wants to retype a previously typed sentence.

ACKNOWLEDGMENTS

This work was supported by EPSRC (grant EP/N014278/1). Supporting code for this paper is available in the ACM Digital Library.

REFERENCES

[1] Abdullah Al Mahmud, Rikkert Gerits, and Jean-Bernard Martens. 2010. XTag: designing an experience capturing and sharing tool for persons with aphasia. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*. ACM, 325–334.

[2] John L Arnott, Norman Alm, and Alan F Newell. 1988. A text database as a communication prosthesis. June (1988), 76–77.

[3] Luís Garcia, Luis de Oliveira, and David de Matos. 2014. Word and sentence prediction: Using the best of the two worlds to assist AAC users. *Technology and Disability* 26, 2, 3 (2014), 79–91.

[4] D. Jeffery Higginbotham, Gregory W. Lesher, Bryan J. Moulton, and Brian Roark. 2012. The Application of Natural Language Processing to Augmentative and Alternative Communication. *Assistive Technology* 24, 1 (mar 2012), 14–24. DOI: http://dx.doi.org/10.1080/10400435.2011.648714

[5] K Sparck Jones, Steve Walker, and Stephen E. Robertson. 2000a. A probabilistic model of information retrieval: development and comparative experiments: Part 1. *Information processing & management* 36, 6 (2000), 779–808.

[6] K Sparck Jones, Steve Walker, and Stephen E. Robertson. 2000b. A probabilistic model of information retrieval: development and comparative experiments: Part 2. *Information processing & management* 36, 6 (2000), 809–840.

[7] 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*. ACM, 49–56.

[8] Heidi Horstmann Koester and Sajay Arthanat. 2018. Text entry rate of access interfaces used by people with physical disabilities: A systematic review. *Assistive Technology* 30, 3 (2018), 151–163. DOI: http://dx.doi.org/10.1080/10400435.2017.1291544

[9] Stefan Langer and Marianne Hickey. 1998. Using semantic lexicons for full text message retrieval in a communication aid. *Natural Language Engineering* 4, 1 (1998), 41–55. DOI: http://dx.doi.org/10.1017/S1351324998001855

[10] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge University Press.

[11] George A Miller, E Newman, and E Friedman. 1957. Some effects of intermittent silence. *American Journal of Psychology* 70, 2 (1957), 311–314.

[12] Arnab Nandi and H. V. Jagadish. 2007. Effective Phrase Prediction. *Proceedings of the 33rd International Conference on Very Large Data Bases* (2007), 219–230. http://dl.acm.org/citation.cfm?id=1325851.1325879

[13] Ondrej Polacek, Zdenek Mikovec, Adam J. Sporka, and Pavel Slavik. 2011. Humsher: A Predictive Keyboard Operated by Humming. *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility - ASSETS '11* (2011), 75. DOI:http://dx.doi.org/10.1145/2049536.2049552

[14] Ondrej Polacek, Adam J. Sporka, and Pavel Slavik. 2017. Text input for motor-impaired people. *Universal Access in the Information Society* 16, 1 (2017). DOI: http://dx.doi.org/10.1007/s10209-015-0433-0

[15] R. Richardson, A. Smeaton, and J. Murphy. 1994. Using WordNet as a Knowledge Base for Measuring Semantic Similarity between Words. *Technical Report Working Paper CA-1294* (1994). DOI: http://dx.doi.org/10.1.1.49.6027

[16] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* 28, 1 (1972), 11–21.

[17] John Todman, Norman Alm, and Leona Elder. 1994. Computer-Aided Conversation: A Prototype System for Nonspeaking People with Physical Disabilities. *Applied Psycholinguistics* 15, 1 (1994), 45–73. DOI: http://dx.doi.org/10.1017/S0142716400006974

[18] John Todman, Norman Alm, Jeff Higginbotham, and Portia File. 2008. Whole utterance approaches in AAC. *AAC: Augmentative and Alternative Communication* 24, 3 (2008), 235–254. DOI: http://dx.doi.org/10.1080/08990220802388271

[19] 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)*. ACL, 700–711.

[20] Annalu Waller, Fiona Dennis, Janet Brodie, and Alistair Y. Cairns. 1998. Evaluating the use of TalksBac, a predictive communication device for nonfluent adults with aphasia. *International Journal of Language and Communication Disorders* 33, 1 (1998), 45–70. DOI: http://dx.doi.org/10.1080/136828298247929

[21] Timothy Walsh. 2010. Utterance-Based Systems: Organization and Design of AAC Interfaces. *Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility* (2010), 327–328. http://src.acm.org/binaries/content/assets/src/2010/timothy-walsh.pdf