ChoiceKey: A real-time speech recognition program for psychology experiments with a small response set.

Christopher Donkin, Scott D. Brown, & Andrew Heathcote
The University of Newcastle
Newcastle, Australia

Word Counts:
Body: 4,901
Abstract: 143
References: 484
Appendices: 1,338

Address correspondence to:
Chris Donkin
School of Psychology
University of Newcastle
Callaghan NSW 2308
Australia
Email: chris.donkin@newcastle.edu.au
Psychological experiments often collect choice responses using button presses. However, spoken responses are useful in many cases, for example: when working with special clinical populations, or when a paradigm demands vocalisation, or when accurate response time measurements are desired. In these cases, spoken responses are typically collected using a voice key, which usually involves manual coding by experimenters in a tedious and error-prone manner. We describe an open source speech recognition package for MATLAB, ChoiceKey, which can be optimized by training for small response sets and different speakers. We show ChoiceKey to be reliable with minimal training for most participants in experiments with two different responses. Problems presented by individual differences, and occasional atypical responses, are examined, and extensions to larger response sets are explored.
Many psychology experiments require participants to complete hundreds of trials using a small response set. For example, memory experiments might require participants to respond with only ‘old’ or ‘new’ (e.g. Rubin, Hinton & Wenzel, 1999) and two-choice response time (RT) tasks might require a participant to respond with ‘one’ and ‘two’ or ‘high’ and ‘low’ (e.g., Ratcliff & Rouder, 1998). The typical method for collecting responses for this type of experiment is a button press, usually via keyboard, mouse, specially developed button-boxes, or touch screen. There are a number of reasons that an experimenter might instead prefer to collect spoken responses, for which we offer an open source speech recognition package, ChoiceKey. We show ChoiceKey to be reliable with a small number of response alternatives, and to give precise estimates of RT. We begin by discussing a few of the circumstances in which one might prefer vocal responses over alternative data collection methods.

Response time measurement is an important aspect of many psychology experiments, and the precision and accuracy of RT estimates from different response tools has been documented extensively in this journal. Keyboard responses are often imprecise due to buffering issues (Plant, Hammond & Turner, 2004; Shimizu, 2002; Voss, Leonhart & Stahl, 2007), as are mouse-button clicks (Beringer, 1992; Crosbie, 1990; Plant, Hammond & Whitehouse, 2003). Precise RT measurements can be obtained using button-boxes connected via the PC parallel port (e.g., Stewart, 2006; Voss et al.), but these solutions require specialised hardware, which can be expensive and is not always well supported. We show that ChoiceKey also yields precise measurements of RT, but is very simple to set up and is also inexpensive, as it requires only a microphone and a sound card (which come built-in as standard for most PCs).
Aside from RT measurements, making responses via buttons can be problematic because it requires the participant to learn a response-to-button mapping. Although some of these mappings are relatively natural, such as ‘left’ and ‘right’ using the left and right arrows of the keyboard, other response sets have no obvious or intuitive button mapping. For example, Rubin et al. (2004) mapped the responses ‘old’ and ‘new’ to keys chosen by the experimenters, and participants had to learn this mapping and maintain it throughout the experiment. If participants were able to speak aloud the responses ‘old’ or ‘new’, then the learning of this mapping can be avoided.

Experimental research with clinical populations who are unable to give manual responses via keyboard or button press might also benefit from the ability to easily collect spoken responses and the associated RTs. In particular, an automatic speech recognition program might benefit experimenters working with people with schizophrenia, people with intellectual disabilities, or people with psychomotor disabilities. Trewin and Pain (1999) have shown that people with these types of psychological and/or motor disabilities display a large number and wide range of errors when using a mouse or keyboard. The use of spoken responses may help avoid some of these measurement errors.

Even if participants are able to give manual responses, Vidulich and Wickens (1985) show that under certain situations spoken responses are most appropriate, particularly when central processing is required for a task that is verbally oriented. It is also possible that the need for spoken responses is implicit given the experimental procedure or paradigm being used. For example, Stroop-interference style tasks which investigate the cause of interference due to response modality (Simon & Sudalaimuthu,
1979; Wang & Proctor, 1996). In these situations ChoiceKey offers a way of recording both response and RT.

Amongst others, Lacouture and Marley (2004) allowed participants to respond using voice in an absolute identification experiment. Participants gave spoken responses via microphone and reaction times were obtained using a “voice key”, which is a device that measures RT by finding the time taken from stimulus onset until sound energy crosses some threshold. Aside from response times, response choices were manually coded by an experimenter, a task which is time consuming, tedious, and possibly error-prone. Speech recognition software can help alleviate this burden.

Speech recognition software is in common use; most people have had the experience of ordering a pizza, or giving personal details, by speaking into a phone. However, the accuracy of these systems can be far from perfect, and is unlikely to be acceptable for experimental measurement. Microsoft Windows and Macintosh OS X both come with inbuilt speech recognition functionality that can be adapted by training to individual users’ voices. However, we found these inbuilt speech recognition packages to be too inaccurate for use in experiments, even under ideal conditions with only two different responses and extensive training. This is likely because the programs are intended to recognise a very large number of different responses, in environments where the cost of an incorrect recognition event is low.

As an alternative, we offer an exemplar-based speech recognition program that is trained exclusively to recognise only those responses that are to be used in an experiment. The program, ChoiceKey, is an open source library for the software package MATLAB. A MATLAB script can also be used to run an experiment. Appendix A outlines an
example script for a simple experiment where one of two stimuli is presented, and the
participant is required to name it. ChoiceKey was developed using the Data Collection
and Voicebox toolboxes under MATLAB v7.5.0 (R2007b) and Reynolds, Quatieri and
Dunn’s (2000) Gaussian mixture models for speaker identification. Details about the
contents of ChoiceKey are outlined in Appendix B.

The underlying structure of ChoiceKey is based on leading models of speaker
offer a detailed and complete discussion of the extensive work in this area. A graphical
summary of how ChoiceKey works is presented in Figure 1. Sound card outputs are
captured using the inbuilt MATLAB Data Collection Toolbox. Audio capture begins
when input from a microphone reaches a threshold, and terminates one and a half seconds
later. Both the threshold and recording time can be altered by the user. The recorded data
are first passed through front-end processing, transforming the time-varying amplitude
input from the microphone into a set of features represented as a vector of cepstral
coefficients. These feature vectors are then modelled statistically creating a set of training
exemplar models. Later, during the experiment, participants’ responses are turned into
feature vectors and the likelihood that these vectors came from each training exemplar
model is calculated. The most likely model is the chosen response. We now discuss each
of these in more detail.
Speech Recognition for Psychology Experiments

Training

E.g.

"old" \rightarrow \text{Front-end Processing} \rightarrow \text{Statistical Modelling} \rightarrow \text{One training exemplar model for the response "old"}

Each response is repeated $N$ times giving that many models for each response.

"old"

"new"

Test

"old" \rightarrow \text{Front-end Processing} \rightarrow \text{Statistical Modelling}

calculate feature vectors

compare with all models and select the best fit as the chosen response

"old"

$N$

$N$

Figure 1 Graphical representation of how ChoiceKey operates. During training, the response “old” is given on a particular trial. Features are extracted and a statistical model is created for that training exemplar. The model is then stored with the rest of the models created in training. During testing, the participant has spoken the word “old” during one of the trials of the experiment. The features of the word are extracted and then compared against all the models created at training. The most likely model is chosen, which happens to be one of the trained examplars for the response “old” making ChoiceKey conclude that the participant made that response.

Front-end Processing

In the front-end processing stage, the sound input is broken up into 20ms windows using a 10ms frame-rate, ensuring 50% overlap between segments. Only those windows containing enough sound energy to be considered not silent are kept. This is done relative to the noise in the signal so as to lower the probability that speech is discarded. The Mel-scale cepstral feature vectors are then calculated for each of the 20ms
windows. This is done by first taking the fast Fourier transform of the speech segment. The resulting spectrum is smoothed using a series of band-pass frequency filters which are convolved with the spectrum to get an average value for each frequency band. These filters are spaced on the Mel scale, as it has the property of being close to the frequency scale of the human ear (Stevens, Volkman & Newman, 1937). The cosine discrete transformation is applied to the log of the values produced by the frequency filters to yield cepstral coefficients. Reynolds, Quatieri and Dunn (2000) suggest that all but the 0\textsuperscript{th} cepstral coefficient are best used in speaker recognition. However, for ChoiceKey we desire speech, not speaker, recognition, and we have achieved greater accuracy by the inclusion of the 0\textsuperscript{th} coefficient. Reynolds et al. also suggest a number of normalisation transformations be made to compensate for mismatched microphone conditions between training and testing. We did not use these transformations as we assumed that since the training of ChoiceKey takes less than five minutes, that training and testing would be done under identical conditions.

**Statistical Modelling**

The cepstral coefficients are modelled using Gaussian mixture models (GMM), which have been shown to be successful in the domain of speaker recognition (Reynolds, 1992). GMMs have the required properties of being able to capture the behaviour of a distribution without assuming a very specific form (e.g. Gaussian), while also being computationally simple, to allow for real time processing. A GMM’s density is the weighted linear combination of $M$ Gaussian densities, each parameterized by a mean and variance term for each cepstral coefficient vector. The number of Gaussian densities to use, $M$, can be altered by the user. During development we found that five Gaussian
densities gave the best overall performance. However, individual differences in the optimal value of $M$ did exist, and hence, improved individual accuracy may be obtained by setting $M$ based on an individual’s data.

**The Decision**

During the training of ChoiceKey, the participant will speak aloud each of $k$ response words $N$ times. Each of these training words is modelled using a GMM, giving $N$ exemplar models for each of the $k$ responses at the end of training. On any particular trial of the experiment proper, the participant will make a new and unknown response. The mel-scale cepstral feature vectors are calculated for this new response. ChoiceKey then calculates the log-likelihood of observing the cepstral feature vectors given the parameters of the GMM for each of the $N$ exemplars and $k$ responses. In ChoiceKey, the exemplar with the largest log-likelihood is selected as the given response. During development we tried a range of alternative response selection rules, such as selecting the response set with the largest summed log-likelihood across all exemplars, and more sophisticated classifiers, such as backpropogation neural networks and support vector machines. For the small response set sizes in our experiments the more sophisticated selection rules did not provide any benefit, but this may not be the case for larger response set sizes. The simple selection rule used by ChoiceKey has the advantage of reduced computational cost, particularly during training. In other settings, ChoiceKey can be easily adapted by the user to implement alternative training and decision algorithms.
Using ChoiceKey

A typical experiment using ChoiceKey involves a short training session (less than five minutes), where participants speak aloud the words of the response set (e.g. old/new) a number of times (typically between 10 and 30), training ChoiceKey to identify their voice. The experiment then proceeds as normal, with responses made by voice, using ChoiceKey to return the response that it calculates to be the one most likely spoken by the participant, and the response time (RT). We report results examining the accuracy of these measurements.

We first investigate the accuracy of RT as measured by ChoiceKey by comparing it with reaction times manually measured from audio waveforms recorded in real time. In our second experiment we investigate how accurately ChoiceKey identifies responses from a variety of typical response sets.

Experiments

Response Time

An AMD computer with an AthlonXP 64-bit 2.33 Ghz processor, 2Gb of RAM, running Windows XP SP2 with a SoundBlaster Live! v5.10 sound card was set up to play a loud tone through external speakers. After the tone played, data capture within ChoiceKey was initialised, and after some interval the word ‘respond’ appeared on screen and the participant was to speak aloud the number ‘one’ into a headset-mounted SONY DR-220 microphone. The speaking of the word ‘one’ was intended to trigger recording, and for all 200 trials the triggering worked as required. The intervals between the tone and response were varied from 250 to 2000ms in intervals of 250ms, with each interval
occuring 25 times, yielding 200 response times that spanned the range of RTs usually observed in simple psychological tasks.

Throughout the course of the experiment, a second laptop was set up nearby with its external microphone making a real-time recording of the entire proceedings under Adobe CS3 Soundbooth. This recording was later opened in Soundbooth as a waveform and the time between the tone and response of “one” was manually determined by measurement of the time elapsed between the start of the tone and the start of the voice response. This process gave us an accurate estimate of response time that should correlate highly with ChoiceKey’s RT measurement.

![Graph showing correlation between ChoiceKey's and manually calculated RTs](image)

*Figure 2* Both the RT as recorded by ChoiceKey (y-axis) and the RT as calculated manually from sound waveforms (x-axis) are plotted together. The solid diagonal line represents perfect measurement of RT. ChoiceKey gives a precise, but slightly biased estimate of RT.
Figure 2 shows the response times recorded by ChoiceKey plotted against response times from the waveform. The two measures of response time were highly correlated ($r = .999$). Response times from ChoiceKey were used as the response variable in a linear regression, with true response time used as the predictor variable. The slope of the regression line was 0.999, $t(199) = 344.83$, $p < .001$, and the intercept was 90ms $t(199) = 18.875$, $p < .001$, suggesting that ChoiceKey gives a precise, but slightly biased estimate of response time. Rastle and Davis (2002) discuss biases in voice-key RT measurement as a function of the onset characteristics of different response waveforms. If experimenters are concerned about obtaining absolutely unbiased measurements of RT, the above procedure can be carried out for all responses separately. More likely to be of concern to users of ChoiceKey, however, are differences between biases in RT measurement for different responses. Unless this issue is addressed, differences in RT may be attributed to differences between stimuli, when the real cause is differences in the time taken for ChoiceKey to trigger the onset of recording.

To address this issue, ChoiceKey includes a function called `callib.m` which, on each trial, presents participants with a “+” sign and ask them to respond with one response repeatedly for a block of trials of length determined by the experimenter. The process is then repeated for each response. The function returns the mean RT for each response. Any differences in RT due to onset biases for the different responses can then be identified and controlled without requiring real time recordings to be scored by hand.

**Identification**

Identification accuracy was investigated using data from 24 participants who read aloud the following three sets: {1,2,3,4}, {old,new} and {high,low}. Participants were 12
male and 12 female first year psychology students. Words in each set were spoken 50 times in a random order and the order of each set was counterbalanced across subjects. Responses were collected using the same hardware used in the previous experiment. As in a standard psychological experiment, the first 10 responses spoken by participants were used to train ChoiceKey and the remaining 40 responses were used to test ChoiceKey’s identification accuracy. Results from the response set consisting of numbers one to four were partitioned into six different response sets of size two. These sets, along with \{old,new\} and \{high,low\} were used to test ChoiceKey’s identification accuracy. Table 1 shows the distributions of the number of errors out of the 80 identifications made by ChoiceKey for each response set of size two.

Table 1 The number of errors (out of 80 identifications) made by ChoiceKey for individual participants for each response set. For example, the first cell indicates that for 16 out of 24 participants there were zero errors in identification for the word set \{old,new\}. The mean proportion of correct identifications for each response set is reported in the rightmost column.

<table>
<thead>
<tr>
<th>Response Set</th>
<th>Number of errors (out of 80 identifications)</th>
<th>Mean Correct Id’s</th>
</tr>
</thead>
<tbody>
<tr>
<td>old,new</td>
<td>16 0 3 3 2 0</td>
<td>.99</td>
</tr>
<tr>
<td>2,4</td>
<td>15 5 1 1 2 2</td>
<td>.98</td>
</tr>
<tr>
<td>1,2</td>
<td>14 5 0 3 2 2</td>
<td>.96</td>
</tr>
<tr>
<td>1,3</td>
<td>11 6 1 4 2 2</td>
<td>.97</td>
</tr>
<tr>
<td>3,4</td>
<td>12 4 4 3 1 2</td>
<td>.98</td>
</tr>
<tr>
<td>1,4</td>
<td>9 2 2 9 2 2</td>
<td>.95</td>
</tr>
<tr>
<td>high,low</td>
<td>6 4 5 8 1 1</td>
<td>.96</td>
</tr>
<tr>
<td>2,3</td>
<td>6 2 4 7 5 5</td>
<td>.92</td>
</tr>
</tbody>
</table>

Very few errors in identifications were observed for the majority of participants and responses. For most sets of responses ChoiceKey made zero or one error out of 80 for almost all participants. For the response set \{old,new\}, ChoiceKey made no errors in
identifying responses for 16 out of 24 participants. The response sets \{2,4\} and \{1,2\} were also identified with very few errors, with either zero or one error being made for 20 and 19 participants, respectively. The average accuracy of ChoiceKey’s identification was highest for the response sets \{old,new\}, followed closely by \{2,4\} and \{3,4\}. Individual differences in identification far outweighed any differences observed between age or gender groups.

The results of Table 1 indicate that when using only 10 training examplars, ChoiceKey is able to perform well for some response sets, but others are less discriminable. For example, in the response sets \{high,low\} and \{2,3\}, ChoiceKey was able to identify all responses correctly for only a quarter of the participants. Not only were some response sets less discriminable, but even for the responses sets in which ChoiceKey was almost perfectly accurate for the majority of participants, a small proportion of participants remained whose responses were difficult to discriminate. For example, the response set \{1,2\} leads to either one or no errors for 19 out of 24 participants, suggesting it as a good candidate response set for use with ChoiceKey. However, for one participant 35 responses out of 80 were identified incorrectly. This suggests that with minimal training ChoiceKey is not viable for some participants.

One approach to solving this problem is to first screen participants based on a pre-experimental test of ChoiceKey’s accuracy. The function `traintest` provides an estimate of ChoiceKey’s identification accuracy. Interestingly, for the participant with very low accuracy for the response set \{1,2\} all other combinations of number responses accuracy were also low. However, no errors in identification were observed for the response set
{high/low} for this participant, suggesting that speaker identification accuracy varies substantially as a function of response set.

An alternative approach to dealing with low identification accuracy, for either particular participants or particular word sets, is to use more than the ten training exemplars used above. To evaluate this strategy, 12 of the 24 participants completed an extra 50 responses for the word sets {old,new} and {high,low}. These extra responses were used to test the effect of varying the number of responses per word used in training ChoiceKey. The number of training exemplars used was varied from 1 to 30, always with the last 70 exemplars used to test ChoiceKey’s accuracy. Figure 3 shows the average percentage of accurate identifications by ChoiceKey as a function of the number of training exemplars. For the response set {old,new}, increasing the number of training exemplars beyond 10 did not increase accuracy. This is likely due to a ceiling effect, as discrimination between the words was already close to perfect when just 10 training exemplars were used. For the response set {high,low}, improvement was less rapid, but the same accuracy as for {old,new} was achieved with a set of 30 exemplars, suggesting that even difficult-to-discriminate word sets can be used with ChoiceKey, as long as sufficient training is provided.
Figure 3 Accuracy for identifications made by ChoiceKey averaged over participants and plotted as a function of how many exemplars are used in training. Accuracy is higher for the word set \{old,new\} when fewer training exemplars are used, however, ChoiceKey becomes more able to discriminate between \{high,low\} as more training exemplars are used.

Increasing the number of training exemplars improved accuracy for all participants, even those who initially had very low accuracy. The left panel of Figure 4 compares individual participant accuracy for the response set \{high,low\} with 10 and 30 training exemplars. Participants are ordered along the x-axis by accuracy for the 10 training exemplar case, and the same order used for the 30 exemplar case, to highlight individual improvement. For 23 out of 24 participants, accuracy either increased or remained perfect with the increase in training set size. Participants whose accuracies were lowest with 10 training exemplars showed the largest increase, bringing performance for almost all participants up to acceptable levels.
Figure 4 Recognition accuracy by ChoiceKey for individual participants for the response set \{\text{high/low}\} when 10 and 30 exemplars were used in training (left panel), or when the response sets \{1,2,3\} or \{1,2,3,4\} were used with 10 training exemplars. Participants are ordered according to accuracy in the 10 exemplar and \{1,2,3\} cases for left and right panels, respectively.

With only ten training examplars, accuracy was much worse for responses sets of more than two words. The right panel of Figure 4 shows accuracy for individual participants for the response sets \{1,2,3\} and \{1,2,3,4\} when ChoiceKey was trained with ten exemplars. Participants are ordered by accuracy for the smaller response set on the x-axis. Average accuracy was roughly equivalent for both response sets, although some participants were noticeably less accurate for the larger response set. Only a quarter of participants were more than 95% accurate, suggesting that a larger training set is required for the remaining participants. Experimenters who wish to use ChoiceKey for response sets larger than two can use the \texttt{train/test} function to screen participants and/or calibrate training set size to achieve the desired level of accuracy. The latter strategy should be
used with caution, however, as we did not test whether larger set sizes display the same improvement in accuracy with training set size as we found with set size two. A second limitation related to the use of larger set sizes is the associated computational cost, which increases as a polynomial function of both the response set size and the number of training exemplars. Sufficient time must be available between collecting the training data and commencing the experiment to process the training data. For example, when response set size was increased from two to four, the time taken to identify a single response, given 10 training exemplars, increased from 50 to 100 milliseconds. When 40 training exemplars were used, however, identifying a single response took 100 and 300 milliseconds for two and four responses, respectively.

**Discussion**

ChoiceKey is a measurement tool for the MATLAB environment that allows the collection of vocal choice responses. It is designed for use in experiments with a small number of possible responses. ChoiceKey provides precise estimates of vocal onset time, and can be easily calibrated to eliminate onset differences between responses (Rastle & Davis, 2002). For a variety of common response pairs, it can reliably identify most participants’ responses with high accuracy after only minimal training (e.g., 10 training exemplars per response, which takes around two minutes for a binary choice task).

However, we found that some response pairs are identified with lower accuracy than others (e.g., {high, low}). Although experimenters could simply use response pairs which are reliably identified with high accuracy (e.g., {old,new}), doing so removes the potential benefit of reduced response learning offered by spoken responses. An
alternative strategy is to use a larger training set, which improves accuracy (e.g., for 30 exemplars accuracy was equally good for \{old,new\} and \{high,low\}).

A second problem is that identification accuracy is low for some participants, suggesting that there may be a need for pre-test screening of ChoiceKey’s accuracy for each participant. This problem can be limited through the use of more training exemplars, at least for the participants in our experiment. Increasing the number of responses used to train ChoiceKey from 10 to 30 not only increased identification accuracy for one of the response sets with the poorest performance, \{high,low\}, it also improved identification accuracy for those participants whose responses were most poorly identified when only 10 training exemplars were used.

Even after extended training, ChoiceKey did not perfectly identify all responses from all participants. These errors appeared to be asymptotic (i.e., they did not disappear with increased training). Such asymptotic errors are likely due to atypical responses, background noise (in the environment or in computer hardware), or both. We minimized the latter by using a quiet testing environment and a high quality sound card and microphone. However, it is likely that even when hardware errors are minimized, participants will sometimes say words in a way that was not encountered in training, causing mis-identification. Fortunately, the proportion of such asymptotic errors in testing was low (around 1%). It is arguable that this low rate may not be too different from the rate of errors due to participants pressing the wrong response button, and it may even be less than the button-press error rate when the response mapping is unfamiliar or insufficiently practiced. Similarly, ChoiceKey’s low rate may be comparable to errors made by the experimenter manually coding responses in real-time. Where perfect vocal
choice identification is required we recommend that responses be recorded and scored off-line. Even where an error rate of 1-2% is acceptable it may be prudent to record some responses and perform an off-line check of ChoiceKey’s scoring as a quality control measure.

Apart from lower background noise, there appears to be no other clear practice that will reliably increase ChoiceKey’s accuracy. For example, there appears to be no clear reason why certain word pairs show lower discriminability than others. We therefore advise users that they choose the most natural response set for the task, and if necessary, increase the number of training exemplars until the desired level of identification accuracy is achieved. Similarly, there appears to be no clear pattern to the type of voice that ChoiceKey is able to identify with high accuracy (e.g. male or female voices). We suggest the same course of action – pick the most natural methodology and if pre-test screening shows a high error rate with any individual participant either use more training exemplars or exclude that participant’s results from analysis.

Vocal responses are particularly advantageous with larger response sets, where button responding may be more error prone due to the greater difficulty of learning a larger response mapping. Large response sets can also be more likely to introduce differences in RT due to differences in response production time (e.g., differences between fingers). When more than 10 responses are required a unique finger cannot be assigned to each response, requiring either finger combinations, further increasing learning difficulty, or a movement response (e.g., moving a finger or mouse cursor from a “home” button to a response button). In both cases, RT variability, and the potential for differences in production time, are increased. However, we found that creating a speech
recognition system which is highly accurate for large response sets is very difficult. When the response set was extended beyond two alternatives, we observed a large drop in accuracy, to about 90% on average for three or four different responses. We also found large individual differences, and that a majority of participants had low accuracy when ChoiceKey was trained with only 10 exemplars per response. These results indicate that ChoiceKey should be used cautiously with greater than two response alternatives, and that likely extended training will be required to obtain high accuracy. Fortunately, because ChoiceKey is open source and implemented in the flexible MATLAB language, users may easily explore such extensions.

Some directions for these future extensions have already been discussed. For example, it may be possible to optimise certain parameters affecting identification accuracy individually for each participant (e.g., the number of Gaussian density mixtures to use in the GMM). Future improvement may also lie in an alternative form of statistical modelling of the features of the recorded speech segment. Our use of GMM as a model of these features was based on their success in the field of text-independent speaker recognition. Text-independent speaker identification involves the recognition of a voice irrespective of the spoken utterance. Text-dependent speaker identification involves recognising voices based on a particular set of spoken words. Hidden Markov models (HMMs) are often used to model the features of the spoken response in text-dependent speaker identification, as they incorporate temporal information from the sound segment, where GMMs do not (Bimbot et al., 2004). It is possible that the lower accuracy observed for certain word pairs might be due to the time-independent modelling of spoken features. For example, the words “high” and “low” certainly sound different when we hear them.
However, when their features are collapsed into a single point in time, as in a GMM, perhaps there is substantial overlap between the two. This might make it such that Gaussian mixture modelling cannot reliably discriminate between these two word pairs. This may also explain why there appears to be no clear reason why certain word pairs are discriminated with lesser accuracy than others. Using HMMs instead of GMMs in ChoiceKey might lead to higher identification accuracy for word pairs whose features overlap significantly on a time-independent scale, or for larger sets of words.

To summarise, ChoiceKey offers the ability to easily collect spoken responses and precise response times without the need for manual coding of responses associated with voice keys. We have shown that when 30 training exemplars are used, ChoiceKey is inaccurate on around 1% of trials for roughly a quarter of participants. We doubt whether this error rate is much different from errors made using other forms of response collection (i.e. pressing the wrong button when using a keyboard or mouse). Some may worry that the time taken to train ChoiceKey using 30 exemplars might be restrictive or offer no benefit over the time taken for participants to learn response-button mappings; however, with only two responses this training would take only two minutes (two seconds per response for 30 exemplars). We also note that once trained, unlike a participant, ChoiceKey will not forget its training. At present, the only major downfall of ChoiceKey is that it is limited to highly accurate measurement when using two responses.
References


Stewart, N. A PC parallel port button box provides millisecond response time accuracy under Linux. *Behavior Research Methods, 38*, 170-173.


Appendix A

Example Experiment

We provide MATLAB code for a mock experiment (example.m), where participants are asked to determine on each trial which of two tones differing in loudness is presented. The purpose of this code is not only to help the user collect responses using ChoiceKey, but also to show how MATLAB can be used to control a simple experiment. To begin the experiment, “example” (the name of the .m file) should be entered into the MATLAB Command Window. The experiment starts with the calling of the train function which allows for the recording of responses and the subsequent training of ChoiceKey. After the training, a test of ChoiceKey’s identification accuracy is performed using the traintest function. After the accuracy and number of errors are presented the enter key must be pressed to continue to the experiment.

When the experiment begins, participants are prompted to press any key to continue. The function getkeywait, has been included as it is a handy way to get MATLAB to wait to accept and then return keyboard presses. A fixation cross is then presented for 300 ms, followed by the presentation of the stimulus. In this experiment a tone is played; however, this can be easily adapted to any other stimuli, such as strings of characters, using code similar to that used to display the fixation cross in the example code. In a similar way, images can be displayed using the imread and image functions in MATLAB.

The function test then allows the participant to speak their response, and will return the response which ChoiceKey calculates to be most probable given its training, as
well as the response time. Feedback is displayed for one second, as either the word “Correct” or the correct response, depending on whether the participant was correct or incorrect, respectively. At the end of each block the block number, trial number, stimulus presented, response time and given response are all recorded to a text file. Participants are either given a break of fixed duration, or thanked for their participation if they have completed all blocks. The following is pseudocode for the example experiment:

```
#Train ChoiceKey using the train function

train()

#Test the trained version of ChoiceKey

traintest()

#The experiment

For (k in 1:number of trials)
    #Show the stimuli
    showstimuli()
    #Collect the response and use ChoiceKey to get RT and response
    test()
end
```
Appendix B

The file sr.zip is available from the Psychonomic Society’s electronic archive www.psychonomic.org/archive. Contained within this zip file are the functions, and their respective .m files, required for ChoiceKey to run. The end user will normally only be concerned with the following four of these functions:

The *train* function

The first function to be used is *train*. Participants are first given the complete response set, followed by a series of presentations of each word individually. During each presentation participants are asked to read aloud the presented word. These initial utterances form an exemplar set which ChoiceKey uses to make all future identifications. Only one parameter must be set for the *train* function, the response set: “words”. Optional parameters are the number of responses per word to use for training, “ex”, the duration of recording, “duration”, the frequency of recording, “Fs”, and “trigger”, the amount of input required before audio capture begins. The default number of exemplars that ChoiceKey uses is 10. After the initial exemplars are recorded there will be a short pause of around 30 seconds to a minute, depending on the size of the response set, while ChoiceKey extracts the features and builds a Gaussian mixture model for each exemplar (Reynolds et al., 2000). We experimented with different numbers of features (Gaussians) and found 5 (the default value) to be best with our response sets. Both a smaller and larger number of Gaussians decreased accuracy, and larger values increasing computational time.
The **traintest function**

The *testtrain* function provides a test of ChoiceKey’s identification accuracy for each participant. An extra set of exemplars for each word in the response set are recorded and then used to provide an estimate of expected identification accuracy for the participant. The *testtrain* function requires the response set, “words”, the number of responses per word to use in testing, “ntest”, and the results of the training, “mu”, “sigma”, and “c” to be given. The optional parameters are the same as for the *train* function as well as an additional parameter “silent”, which defaults to “F” (false). The *traintest* function returns the proportion of correctly identified responses and displays it and the number of errors in identification in the MATLAB Command Window if “silent” is not set to “T” (true).

The experimenter may choose to use this feedback to decide if the expected identification accuracy is too low, and whether or not to use ChoiceKey with this participant, or possibly use the extra responses just recorded as additional training exemplars. After the accuracy is displayed onscreen (if “silent” is set to “F”) the experimenter is asked to decide whether or not to use the additionally recorded responses as the exemplars in training ChoiceKey. If silent is set to “T”, the participant’s accuracy is written to a text document called acc.txt. If “yes” is chosen after the prompt then a pause will occur while ChoiceKey is trained on the new responses.

The **callib function**

The *callib* function offers the experimenter a method for estimating the differences in identifying onset time for different responses. Participants are instructed to respond with one of the words from the response set for a block of $n$ trials each time they
Speech Recognition for Psychology Experiments

are presented with a neutral stimulus (a “+” sign). The process is repeated for each word in the response set. The function writes to a text file, callib.txt, the average RT for each response. The callib function requires as input the response set, “words” and the number of recordings per response, “nrec”. The optional parameters “duration”, “Fs” and “trigger”, default to 1.5 seconds, 44100Hz and 0.05, respectively.

The test function

After ChoiceKey has been trained for a participant’s voice, the test function can be called whenever a response is required. This function will record the participant’s response and return the most likely spoken response, given the set of exemplars recorded in the training stage. The time taken to make the response is also returned. As input, the test function needs the three parameters “mu”, “sigma”, and “c” returned by the train function. Optional parameters also include “duration”, if different from 1.5 seconds, “Fs”, if different from 44100Hz, and “trigger”, if different from 0.05. Once the function is called, the audio capture device is activated and waits until audio input reaches the “trigger” threshold, after which it records audio signal for a set duration. The time taken from stimulus onset to the audio signal reaching threshold is recorded as the response time, and returned to the user. The trigger threshold value default of 0.05 worked well in our testing; however, this value can be changed by the user. For example, if background noise is present (e.g., from a computer fan), the threshold may be increased to reduce the false alarm rate due to triggering by background noise. However, setting this value too high may result in responses being missed by ChoiceKey. The features of the audio signal recorded by ChoiceKey are compared to the exemplar sets for each response.
gathered in the training stage. The response selected has the largest individual exemplar log-likelihood value.