# Techniques For Editing Text For TTS Speaking

**A *Distributed*AUDIO White Paper**

by G. S. Tjaden, October, 2003

## Abstract

This paper presents techniques for editing computer readable text prior to delivery to a TTS engine so that the text will be spoken more correctly by it. The techniques are typically implemented as personal computer software. This software performs most of the editing automatically, detects situations in which the editing cannot be done automatically, notifies a human operator of these situations, and provides assistance to the operator to resolve them.

## Background

Numerous products are available today that transform written information encoded as computer readable text into speech that is audible and understandable to humans. They fall into a general classification called Text-to-Speech (TTS). Some of these products are in the form of computer software that runs on general-purpose computers, such as personal computers, while others are in the form of special purpose hardware and software. Often these TTS products are embedded in other products, forming sub-components. An example of a TTS product that is in the form of software that runs on general-purpose computers and can be embedded as a sub-component of other products is ETI-Eloquence 5, produced by Eloquent Technology, Incorporated of Ithica, New York (a division of Speechworks, Inc.).

The core component of TTS software that analyzes the text and transforms it into basic speech elements is called the speech engine.

## The Need For TTS Text Editing

While current TTS products generally do a good job of transforming textual information written in many natural languages (e.g., U.S. English, British English, French, German, etc.) into speech that is intelligible and pleasant to hear, no TTS product currently can produce completely accurate speech. The deficiencies are of several kinds:

First, it is not generally possible to pronounce every word in a given language correctly. For some words the pronunciation is determined by the context in which they are used. Homonyms, such as “close” and “record” fall into this class. Some words do not conform to the normal pronunciation rules of the language. These include special words such as jargon and acronyms (e.g., “NYSE” or “3COM”). Jargon and acronyms are particularly problematic because new words of these types are continually being created, especially within professional disciplines such as business or medicine. Therefore, the developers of the TTS products are not able to incorporate exceptional pronunciations to correct for the non-conformance to the language rules of these words.

Second, it is not generally possible to determine with perfect accuracy the end of every sentence within a paragraph. While the rules of grammar of some natural languages specify special punctuation characters to indicate a sentence end, such as a period at the end of the last word of English sentences, these indicators are not necessarily unambiguous. In the English language abbreviations also end with periods. It is sometimes the case that an abbreviation that is not the last word in a sentence will be followed by a capitalized word (as in "George W. Bush"). And it is possible that an abbreviation will indeed be the last word in a sentence (as in "Texas is the largest state in the U.S."). It is important for TTS products to determine where sentence boundaries reside so the natural rise and fall of voice pitch during speaking that is used by human speakers to indicate the position of the words can be reproduced.

Third, in order for a TTS product to produce natural sounding speech it must be able to identify the correct speech inflexion points within a sentence. For example, very long sentences should be spoken with pauses in appropriate positions, even though the inflexion points normally indicated with punctuation, such as commas in the English language, do not appear in the written form of the sentence with sufficient frequency. Some speech engines do not always identify these inflexion points, or the correct inflexion to be used.

Finally, written information intended only for reading may contain words, phrases or characters that should not be spoken, or should be replaced with alternatives more appropriate to being spoken. For example, an article written for a news magazine may have graphical information, such as charts or tables, associated with it that are referenced in the article. The references may be made with a phrase such as "Figure 1" or "Table 2." Since the figure or table will not be available to the listener, the reference phrases and some of the surrounding text may need to be removed, or modified, to be made more appropriate for speaking.

## Common Approaches

Many TTS products include capabilities that allow some of these deficiencies to be mediated by the integrators of the products. These capabilities generally comprise a phonemic annotation language and a local dictionary.

The phonemic annotation language includes certain special characters, called escape characters. These escape characters are not normally used in the written form of the language, or at least are not used in the position within a sentence or word in which they are used as annotations. When inserted in the text to be spoken they indicate to the speech engine that the characters, word, or portion of the sentence following should be interpreted differently than would normally be the case.

In addition to the escape characters, the phonemic language defines a set of special characters corresponding to each "phoneme" or speech element (e.g., vowels, consonants, diphthongs and reduced vowels) of the corresponding natural language. For example, the phonemic language of ETI-Eloquence 5 uses the character "a" to mean the vowel sounded by the letter "o" in the word "cot," and the character "A" to mean the vowel sounded by the letter "a" in the word "cat."

Included also are a set of special characters used to indicate the stress (enhanced, reduced, or none) to be placed upon vowels. For example, the vowel “a” in the word “example” always should have enhanced stress placed upon it when the word is spoken. Other special characters are used to indicate that an entire word or phrase should be stressed, and still others to change the characteristics of the speaking voice, etc.

The local dictionary is used to store word pronunciations that are substituted for those normally used by the speech engine. These replacement words can sometimes just be respellings of the word, for example replacing the word 3COM with the word “3-com,” or they may be words constructed entirely in the phonemic annotation language. TTS products may include a facility for entering words into the local dictionary. Using this facility the developers of an application in which the TTS product is embedded, or upon which it is based, can provide it with a unique vocabulary.

## Uses For TTS

A typical use for a TTS product is in a telephone voice response system. In such systems human callers are queried with speech messages produced using the TTS product. Responses to the queries are made by callers using the touch-tone keypads of their telephones, and these responses are used to determine further TTS produced queries or messages, returned to the callers as speech. Because such voice response systems usually involve a limited vocabulary of words and a limited set of phrases or sentences, very good speech accuracy can be achieved using the capabilities provided with the TTS product for locally (resident with the speech engine) correcting speech deficiencies.

More recently, TTS products have been used in electronic mail messaging systems in which users call into the system over a telephone and the system reads the messages in the user’s electronic mailbox aloud to the caller. The TTS product is used to produce speech from the text of the electronic mail messages. In these more recent applications the TTS product is given the capability to recognize the unique electronic mail characters, words and phrases (e.g., heading tags, formatting codes, etc.) that should be eliminated or replaced when the message is spoken. However, the body portions of the electronic mail messages will not generally use a limited vocabulary or a limited set of words and phrases. Thus, any words or phrases in the body of the message that the TTS product can’t speak correctly, because of the above deficiencies, will remain uncorrected.

A potential use for TTS products is in mobile computer devices, such as laptop computers, personal digital assistants, cellular telephones and pagers, wherein personalized textual information is delivered to the devices in order that they can speak the information to the user upon demand. Because only textual data (rather than digitized speech data) is delivered to the portable devices, relatively modest data communication bandwidths and smaller local memories to store the delivered data are required, thereby leading to lower costs. In this usage, as in the reading of electronic mail messages over the telephone, a generalized method for correcting for the speech deficiencies of TTS products is required.

One approach to correcting for deficiencies using TTS in portable devices would be to deliver to the devices a copy of a new local dictionary, or relevant portions thereof, along with the textual data to be spoken. This updated dictionary would them be used when speaking the text. Such a dictionary can be in general quite large, so the amount of data needing delivery would be correspondingly increased. Additionally, however, this approach would not correct for ambiguous sentence boundaries, context sensitive word pronunciations, identification of inflexion points in sentences, and removal or replacement of improper phrases in the text itself.

Methods for delivering personalized audio information to a multiplicity of remote computer devices that contain an embedded TTS capability have recently been disclosed by *Distributed*AUDIO (U.S. Patents 5,915,238 and 6,122,617 to Tjaden). These methods teach the editing of computer readable textual information for proper speaking by a TTS product before the resulting edited text is delivered to the remote devices (that contain the TTS speech engine) over a data communication network. Specifically these methods include the teaching that textual information can be pre-edited for speaking by TTS software by removing references to photos and illustrations, by replacing words with their phonetic or phonemic equivalents, and by inserting pauses into long sentences. It is also taught that human intervention may be necessary in performing such editing.

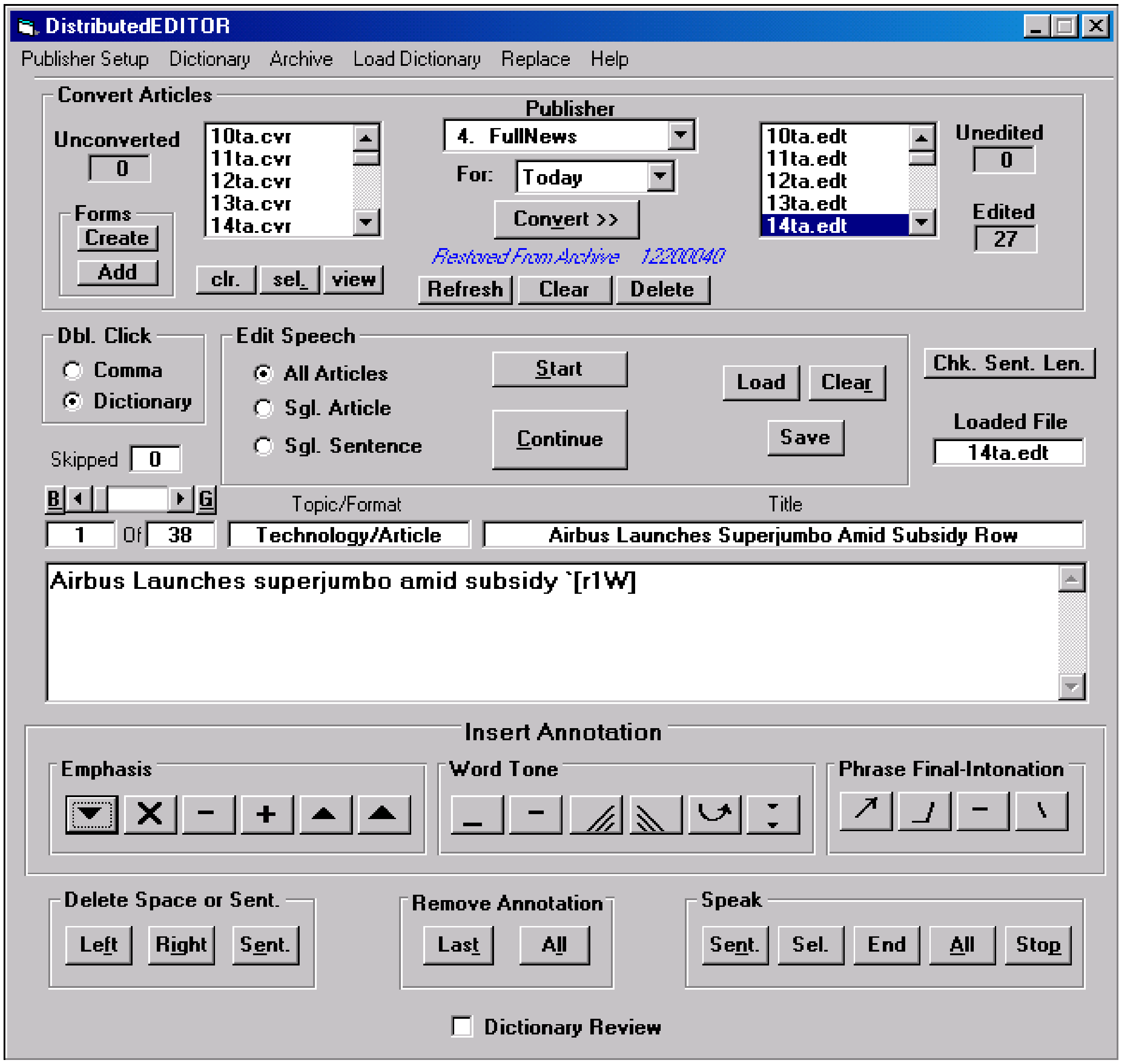
This paper describes some of the techniques developed by *Distributed*AUDIO for performing the semi-automatic editing of computer readable textual information so the information will be spoken correctly by TTS products. The techniques have been implemented in versions of the editing tool developed by *Distributed*AUDIO, called the DistributedEDITOR.

## Editing With DistributedEDITOR

The DistributedEDITOR is typically implemented as a software program written for an IBM-compatible PC (personal computer) under a Microsoft Windows operating system. The only special requirements of the PC for running the software are enough hard disk capacity to store the editing dictionary and the text files, both those to be edited and archived after editing, enough main memory to hold the editing dictionary and the speech engine, and network connectivity of some type to download text files for editing and upload edited files for subsequent distribution to remote devices where the files will be turned into speech by the end-users.

At the core of the editing program is the master dictionary. It is a table of all the words known to the editor, along with the phonetically or phonemically correct pronunciation for each word. Certain flags are also associated with each of the dictionary entries. For some words a subsidiary dictionary is also required. The size of the master dictionary is quite large (more than 80,000 entries currently). The creation and use of this dictionary is described below in more detail.

When the editing program is being used it displays to the operator on the PC's monitor the main user interface (UI), shown in Figure 1. The UI is typically presented as a window, which can be repositioned and resized by the operator. This UI window contains various controls, such as list boxes, command buttons, and text boxes, which are used to present information to the operator and accept input from the operator.



Unconverted: 0

10ta.cvr
11ta.cvr
12ta.cvr
13ta.cvr
14ta.cvr

Publisher: 4. FullNews

For: Today

10ta.edt
11ta.edt
12ta.edt
13ta.edt
14ta.edt

Unedited: 0

Edited: 27

Dbl. Click: Comma [✓ Dictionary]

Edit Speech: [✓ All Articles] Sgl. Article Sgl. Sentence

Skipped: 0

Loaded File: 14ta.edt

1 Of 38

Topic/Format: Technology/Article

Title: Airbus Launches Superjumbo Amid Subsidy Row

Airbus Launches superjumbo amid subsidy `[r1W]

Dictionary Review: ☐

**Figure 1: The DistributedEDITOR User Interface**

There are five major steps in the TTS editing process:

1. Collect files to be edited, called Source files, and store them on the PC's hard drive, or some other suitable memory, in a predetermined subdirectory accessible to the Editing program.
2. For each Source file break the text to be spoken into individual sentences stored as one sentence per record in an associated temporary file, called an Unedited Sink File.
3. For each Unedited Sink File examine every sentence and each word in the sentence for proper speaking. Store the results in a new file, called an Edited Sink File replacing the associated Unedited Sink File.
4. (Optionally,) listen to each file and insert speech annotations to produce the appropriate word emphases, word tones, and phrase-final intonations.

5. Store the collection of Edited Sink Files in a unique directory, called an Archive directory, on the PC's hard drive or other suitable memory. The Archive directory name uniquely identifies the publisher of the files and the time and date of editing.

These five steps are supported by the UI of Figure 1, as follows:

The list of Source files available for editing is shown in the Source file list box in the upper left of the UI. To perform editing the operator must first select the appropriate Publisher from the Publisher drop-down list box. The formatting of Source files will typically be different for each publisher or originator of the files, and these differences must be taken into account by the editing program as it performs Step 2 of the process. Having selected the Publisher, the operator now selects the date for publication of the edited files, typically either the current day (Today) or the next day (Tomorrow) from the For drop-down list box.

Next the operator selects one or more of the Source files from the Source file list and presses the Convert command button. This initiates Step 2 of the editing process. During the execution of this step the editing software may detect certain situations that it cannot resolve automatically. When it does it interrupts the execution of the program and displays a dialogue window on top of the UI explaining the issue and providing various options and functions for the operator to use to resolve it. After resolution by the operator the program continues from where it was interrupted. When this editing step for each selected Source file is completed the file is added to the Sink file list in the upper right of the UI. The new file is given a file name differing from the file name of the Source file only in that the file extension (last three characters of the file name) is the character string "txt". This file extension is used by the operator and editing software to determine that a Sink file has completed Step 2, but not yet completed Step 3.

Upon completion of Step 2 the operator may now proceed to Step 3 by pressing the Start command button. The editing program then selects, in turn, sink files with the "txt" extension from the Sink file list for the editing process of this step. The topic (e.g., News, Technology) and format (e.g., Greeting, Summary, Article) of the file's contents, as determined in Step 2 are displayed in the Topic/Format box. The title of the file's contents, also as determined in Step 2, is displayed in the Title box, and the text of the sentence being edited is displayed in the large text box under the Title box. As in Step 2, the editing software may detect certain issues that it cannot resolve automatically, in which case it interrupts execution and displays an appropriate operator dialogue window. When a Sink file has completed Step 3 it is given a file name extension of "edt". This edited file then replaces the corresponding ".txt" file in the Sink file list.

For many types of information files, such as news briefings, Step 4 is not necessary. The TTS speech engine automatically determines speech inflexions, such as word emphasis and word tone, and its determination is adequate for such types of information. However, for more dramatic material, such as books, it is necessary to listen to the files to make sure the automatic inflexion determination is satisfactory, and correct it if it is not. The control buttons in the Insert Annotation, Remove Annotation and Speak frames are used to assist the operator in this step. A further description of this process is included in the later Speech Prosidy section.

Finally, in Step 5, the Source and Sink files are placed in an Archive subdirectory. The process for doing so is initiated by the operator by pressing the Archive item on the menu bar of the UI.

This initiates a dialogue with the operator in which the appropriate subdirectory name is defined, the files are copied to the subdirectory, and the Source and Sink lists are cleared.

## TTS Text Editing Techniques

The fundamental objectives of the Editor are to automate as much of the editing as possible, to automatically detect where this automation is not possible, and then, in those cases, assist the operator as much as possible in manually editing quickly and correctly. From a business perspective, the total editing process should consume only a small fraction of the time it would require for a human to speak and record the text. This fraction is currently about 10%, and trending downward. Editing Steps 1,2,3 and 5 have been basic to the DistributedEDITOR from the very first version. Step 4 has been incorporated more recently, as the types of information to which the EDITOR is applied has broadened.

Three categories of TTS text editing techniques supported by the DistributedEditor are discussed below:

1. Identifying and correcting ambiguous sentence boundaries,
2. Replacing incorrectly spoken words with correctly spoken equivalents, and
3. Identifying and correcting inappropriate speech prosidy.

### *Ambiguous Sentence Boundaries*

In the majority of cases, at least for English, sentence boundaries can be automatically identified using the basic syntax rules of the language. For English, all that is required is to first identify words (strings separated with spaces) whose last character is a sentence-ending punctuation mark (e.g., ".", "?", "!", etc.). Then the next word is examined to determine if it begins with a capital letter. If so, a sentence boundary likely has been identified. The DistributedEDITOR then saves the sentence as a single line (or record) in a file, and then continues scanning the text to identify the next sentence boundary.

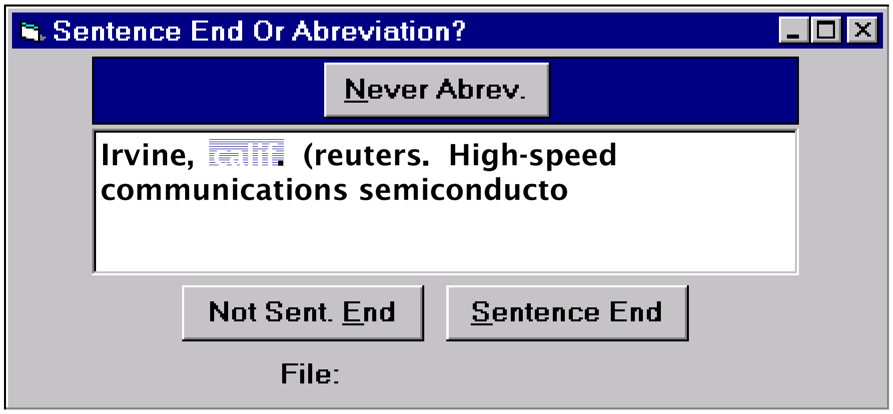
This algorithm must be tweaked a little to allow for sentences that also end in a non-sentence-ending punctuation mark, such as single or double quotes, and sentences that begin with punctuation marks (e.g., a parenthesis or quotation mark (single or double)). Further adjustment is required for sentences whose first word is or begins with a number (e.g., the word 3COM).

There is one situation where the relatively simple algorithm described above does not always produce correct results. That is when a word is an abbreviation. An abbreviation, of course, always ends in a period, so can't necessarily be distinguished from a sentence end. Sometimes abbreviations are at the end of a sentence, but sometimes they are not. Some abbreviations that are not a sentence end are also followed by capitalized words.

Abbreviations do have one common property (other than ending in a period), and that is they are relatively short in length. The reason for the existence of abbreviations is, after all, that they are shorter versions of the words they represent. This property is used to semi-automate the removal of ambiguous sentence boundaries due to abbreviations.

As sentence boundaries are being found, words that end in periods and are less then a certain length are identified as possible abbreviations. For such words a special abbreviation algorithm is used to further determine if the word is indeed a sentence end.

The first step is to check the word against the master dictionary. Each entry in the dictionary has, in addition to the correct pronunciation, two flags, called "AlwaysSentEnd" and "MaybeAbrev". The values of these flags are determined the first time the word is checked by the abbreviation algorithm (given that the word already has an entry in the dictionary). The determination is made by the operator in response to the dialog shown in Figure 2. In this example the word being checked is the highlighted word "calif." in the phrase shown.



Irvine, calif. (reuters. High-speed communications semiconducto

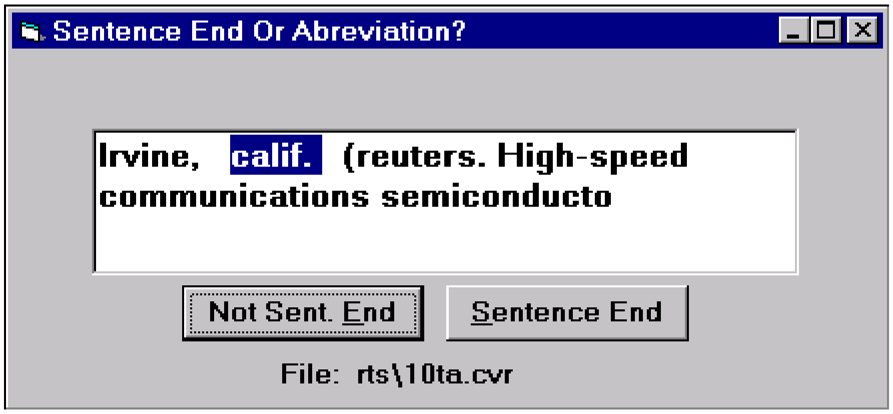
**Figure 2: Initial Abbreviation Dialog**

Upon seeing this dialog the operator first decides if the highlighted word is never an abbreviation (will always end a sentence if it occurs as the last word). An example of such a word is the last word of the last sentence two paragraphs previous to this one, "end". If so the "Never Abrev." Command button should be depressed, causing the AlwaysSentEnd flag to be turned on (set). If the Never Abrev. button were to be depressed it would then be known that the word is the end of the sentence, so the dialogue would be closed, the sentence would be saved in the sink file, and the editor software would then proceed to look for the next sentence boundary.

The word "calif." In Figure 2 is, however, an abbreviation, so the operator should not depress the Never Abrev. command button. Instead, in this example, the operator should depress the "Not Sent. End" button, because the word does not represent a sentence boundary. Doing so causes the MaybeAbrev flag to be set in the master dictionary indicating that this word sometimes may be a sentence boundary. The dialogue is then closed and the editing program continues to look for the sentence boundary.

After the dialogue is closed one (and only one) of the two flags will be set in the master dictionary. Whenever this word is subsequently encountered a different version of the dialogue, shown in Figure 3, will be displayed to the operator. In this case the Never Abrev. button is not displayed, since the MaybeAbrev flag is set in the dictionary. The operator then just selects one of the two options depending upon whether the word is a sentence boundary or not.

If during editing a potential abbreviation is encountered for which the AlwaysSentEnd flag is set, the dialogue is not shown at all, because no operator decision is required. Most potential abbreviations are of this nature.  Thus, only the first time a potential abbreviation is encountered is it necessary to take the time to involve the operator in identifying a sentence ending.



Irvine, calif. (reuters. High-speed communications semiconducto

File: rts\10ta.cvr

**Figure 3:  Abbreviation Dialogue For Words Previously Tested**

## *Incorrectly Spoken Words*

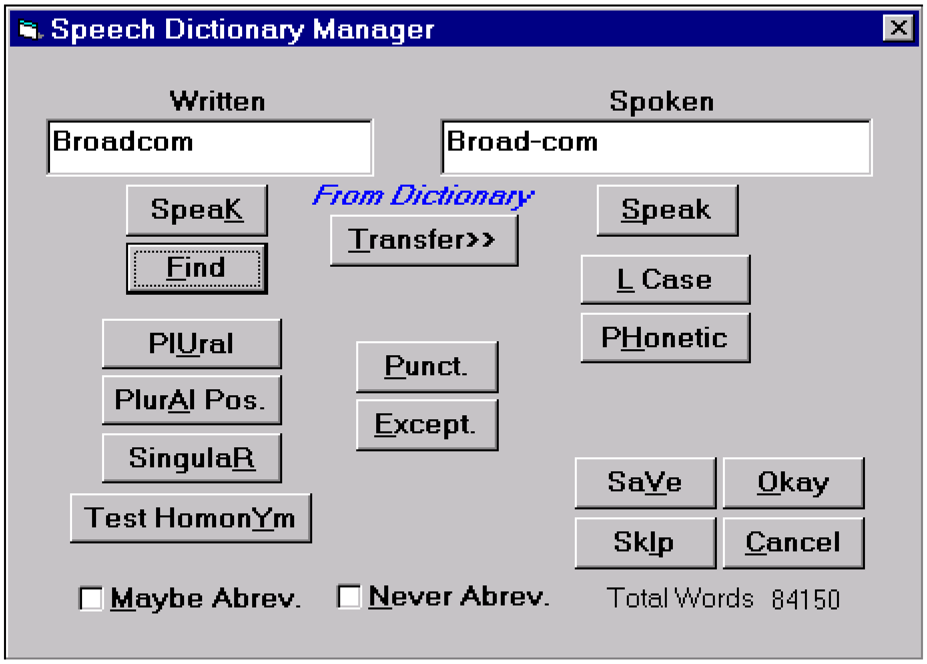
During the third step of the TTS editing process each sentence is checked in turn to make sure that each word in the sentence will be correctly pronounced.  To do so, each word as it is encountered in the sentence is looked-up in the master dictionary.  To keep this look-up from becoming a bottleneck in the editing process the dictionary look-up algorithm uses a hashing technique.

Every TTS speech engine is slightly different in the rules used to pronounce the words of a language, so it is not possible to assume that some known set of words will be always spoken correctly.  Thus, the first time a speech engine is used by the editing program none of the words will have an entry in the dictionary, and the operator will be asked to decide if the speech engine’s pronunciation is correct.  To assist in making this determination the Speech Dictionary Manger dialogue shown in Figure 4 is displayed.

The dialogue is displayed with the word to be checked, called Written, entered in the top left text box.  For words not already in the dictionary the written form of the word is also entered in the Spoken text box in the upper right.  As soon as the dialogue is displayed the speech engine is directed to speak the word in the Spoken text box.  Upon comparing the word as entered in the Spoken text box with what is heard, the operator, assuming they themselves know the correct pronunciation, can immediately decide if it was spoken correctly.  If it was spoken correctly the operator needs merely to hit the enter key on the keyboard (or depress the Okay button, which is slower) to cause both forms of the word to be saved in the dictionary and editing to proceed to the next word.  Whenever this word is again encountered the editing software will automatically replace the Written form of the word with its Spoken form (the same form in this case).  Thus, the operator will never again be asked to check the pronunciation of this word.

The Written word “Broadcom” shown in Figure 4 happens to be a word that is not spoken correctly by the speech engine used by the version of the DistributedEDITOR for this example.

The simplest way, if it works, for the operator to correct the pronunciation is to somehow alter the spelling of the Spoken form of the word. In this case, merely inserting a hyphen between the two syllables corrects the problem. This correction is adequate for many compound words, such as this one. After inserting the hyphen, the operator depresses the Speak button under the Spoken text box, causing the speech engine to speak that word. If the word is spoken correctly the operator depresses the Okay button, which saves both forms of the word in the master dictionary, closes the dialogue, and resumes the editing process. The Spoken form of this word will henceforth be substituted automatically for the Written form by the editing program.



Written: Broadcom

Spoken: Broad-com

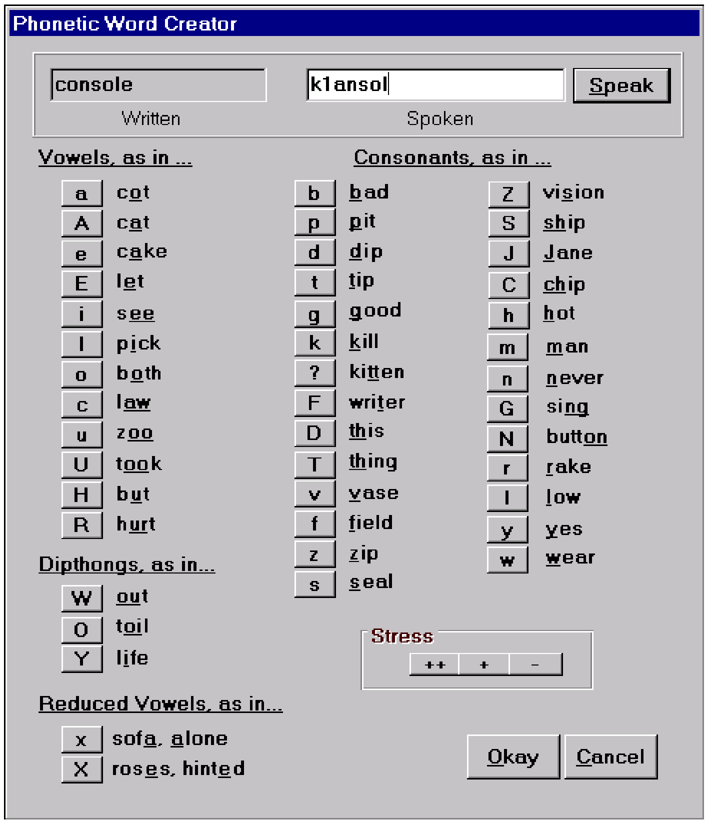
Total Words 84150

**Figure 4: Operator Correction of Word Pronunciation**

In some cases it is not possible to produce a correctly spoken form of a word by merely respelling it. The operator must then resort to using the speech engine's phonemic language. To do so the operator depresses the Phonetic button in the Speech Dictionary Manager dialogue. This opens a new dialogue, called the Phonetic Word Creator, shown in Figure 5.

When the dialogue opens, the written form of the word ("console", in this example) has already been entered into the Written text box in the upper left, but the Spoken text box is empty. In the rest of the dialogue panel all of the phonetic language elements of the speech engine being used, in this case ETI-Eloquence 5, are listed as command button labels, along with an example of the sound or effect produced by the element. The operator merely depresses the appropriate button to enter the element into the Spoken text box. Depressing the Speak button causes the phonetic form of the word as entered in the box to be spoken by the speech engine.

The operator edits the phonetic form until it is correctly spoken. If any of the phonetic rules are violated a message box describing the violation is displayed to the operator, preventing that version from being used. For example, ETI-Eloquence 5 requires that there be at least one vowel and one primary stress mark in a word. Then, when satisfied with the phonemic form of the word, the operator depresses the Okay button to cause it to be transferred to the Spoken text box in the Speech Dictionary Manger dialogue, and close the Phonetic Word Creator dialogue.



Phonetic Word Creator

| Written | Spoken |
|---|---|
| console | k1ansol |

Speak

Vowels, as in ...

| | |
|---|---|
| a | cot |
| A | cat |
| e | cake |
| E | let |
| i | see |
| I | pick |
| o | both |
| c | law |
| u | zoo |
| U | took |
| H | but |
| R | hurt |

Dipthongs, as in...

| | |
|---|---|
| W | out |
| O | toil |
| Y | life |

Reduced Vowels, as in...

| | |
|---|---|
| x | sofa, alone |
| X | roses, hinted |

Consonants, as in ...

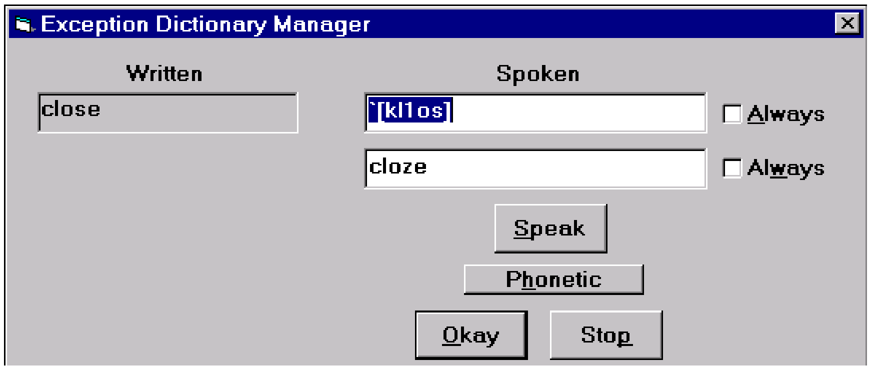
| | | | |
|---|---|---|---|
| b | bad | Z | vision |
| p | pit | S | ship |
| d | dip | J | Jane |
| t | tip | C | chip |
| g | good | h | hot |
| k | kill | m | man |
| ? | kitten | n | never |
| F | writer | G | sing |
| D | this | N | button |
| T | thing | r | rake |
| v | vase | l | low |
| f | field | y | yes |
| z | zip | w | wear |
| s | seal | | |

**Figure 5: Creating A Phonemic Word Form**

The operator then completes the editing of this word by accepting its phonemic form in the Speech Dictionary Manger, causing it to be saved in the dictionary. Once entered into the dictionary the word will automatically be replaced with its phonemic equivalent during editing step 3. For example, the sentence shown in the editing text box in Figure 1 contains the word "row". The edited form of the sentence is shown in the figure, with the word "row" replaced by its phonemic equivalent. The symbols on either end of the phonemic form are the escape symbols that tell the speech engine that the form is the phonemic, not the natural language form.

The word "close" (and the word "row" in the above example) belongs to a class of words called homonyms. These are words that have a single spelling, but different pronunciations depending upon context. For example, in the sentence "The market close was close to a high." the word "close" is pronounced two different ways. Some speech engines attempt to recognize homonyms and to adjust their pronunciation automatically. Experience has shown that the algorithms used to do this are not perfect, and sometimes are far from perfect. Thus, the DistributedEDITOR provides special support for editing homonyms for speaking.

When a word is being entered into the master dictionary for the first time the operator should recognize if it is a homonym. If so, the Except button in the Speech Dictionary Manager dialogue of Figure 4 is depressed. This opens a new dialogue, called the Exception Dictionary Manager, shown in Figure 6.



Written: close

Spoken: `[kl1os] ☐ Always

cloze ☐ Always

**Figure 6: Dialogue For Words With Multiple Pronunciations**

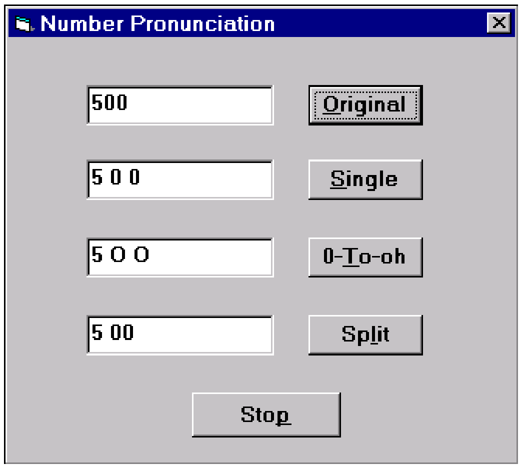
In Figure 6 the two forms of the word "close" are shown. One form is a purely phonemic form. The other is simply a (phonetic) respelling of the word which causes the speech engine to speak it correctly. The operator either edits the words in the Exception Dictionary Manager dialogue, or depresses the Phonetic button to open the Phonetic Word Creator dialogue described above.

Words that can have more than one pronunciation are kept in a secondary dictionary, called the exception dictionary. The master dictionary stores a special string in place of the spoken form of the word for such "exception" words. If this string is encountered during editing the Exception Dictionary Manager dialogue is displayed. Editing is paused while the operator selects the correct form of the word to use. Thus, while editing homonyms for TTS speaking cannot be fully automated, the DistributedEDITOR allows for semi-automatic editing to be performed very quickly.

There are still other classes of words that can have multiple pronunciations. These are numbers, hyphenated words, and slashed (/) words. Such words also always require operator intervention. However, it is not possible (or necessary) to store the various forms in a dictionary for lookup. Instead, the DistributedEDITOR computes the possible forms each time such a word is encountered, and displays the alternatives to the operator for selection. Not all of these computed forms will always apply to the specific context in which the word is used.

While some speech engines have the capability to speak numbers in different ways, they cannot in general determine which is always the correct version to use. Rather, a parameter is set in the speech engine if, for example, it is known somehow that telephone numbers or zip codes will be spoken. All numbers encountered will then be spoken according to the parameter setting until it is changed. General information, such as news, will have a mixture of the different forms, and the correct form cannot be predetermined in a general way.

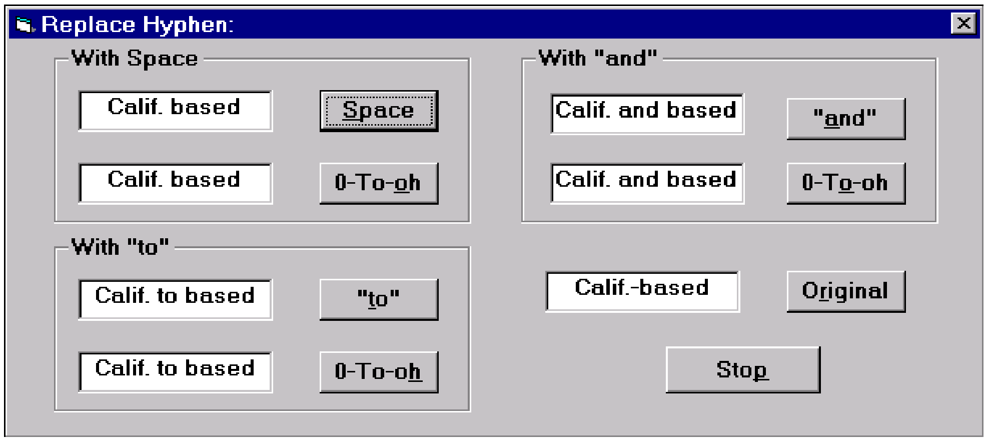
Figure 7 shows the Number Pronunciation dialogue box that is displayed when a number is encountered. In this example the number "500" is shown in its four possible forms. Depressing the command button for the desired form causes it to be inserted into the text and editing to resume.



| | |
|---|---|
| 500 | Original |
| 5 0 0 | Single |
| 5 O O | 0-To-oh |
| 5 00 | Split |

**Figure 7: Alternative Number Pronunciations**

Hyphenated words, an example dialogue for which is shown in Figure 8, can have three possible forms, each of which has two variants. Sometimes the hyphen should just be replaced with a space, forming two individual words. Other times the hyphen should be replaced with the word "to" or the word "and". If the number 0 is included in one or both of the words, that number sometimes should be pronounced as the letter "O" and sometimes just as the number itself.



| | |
|---|---|
| Calif. based | Space |
| Calif. based | 0-To-oh |
| Calif. and based | "and" |
| Calif. and based | 0-To-oh |
| Calif. to based | "to" |
| Calif. to based | 0-To-oh |
| Calif.-based | Original |

**Figure 8: Spoken Forms of Hyphenated Words**

Slashed words, an example dialogue for which is shown in Figure 9, can have three forms. Either the slash is replaced with a space, with the word "to", or the slash is retained. The result of speaking a slashed word in which the slash is retained will depend on the speech engine implementation.

After the operator selects the correct form of a hyphenated or slashed word by depressing the command button for it, the form is inserted into the text being edited. The editing program than rechecks the text from the point at which the insertion was made so that any resulting new individual words will themselves be checked against the master dictionary.

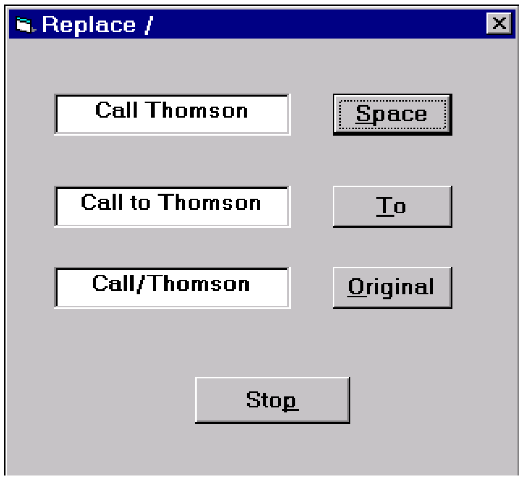


**Figure 9: Spoken Forms of Slashed Words**

## *Speech Prosidy*

The rise and fall of speech pitch and the pacing of speaking is called prosody. While most speech engines do a good job of inserting prosidy into the speech they produce, their performance is not always adequate.

One type of prosodic deficiency is encountered in long sentences in which no or few prosodic punctuation marks (e.g., commas) appear. A special editing technique that semi-automatically corrects for such situations is sometimes useful, and has been implemented in some versions of the DistributedEDITOR. This technique is based on the fact that natural pauses in speaking a sentence often occur just before a prepositional phrase.

During editing the EDITOR keeps track of the number of words since the last comma encountered. If this count grows larger than a set limit the editing program looks for the first previous preposition after the most previous comma. It finds such prepositions by checking each word against a list of know prepositions. If a preposition is found a comma is automatically inserted after the word preceding the preposition. If not, editing is halted, the editing cursor is inserted at the stopped position in the sentence displayed in the editor's user interface, and a warning message is displayed to the operator instructing them to manually insert a comma in the sentence.

With usage over time the list of prepositions has grown. It also now includes some words which are not actual prepositions, but which usually will result in natural sounding speech if a comma is inserted after the preceding word.

In addition to pauses there are at least three other types of speech inflexions, word emphasis, word tone, and the final intonation of phrases. As mentioned previously. for certain types of information files, such as news briefings, more recent speech engines do an adequate job of automatically producing these types of inflexions. However, for more dramatic material, such as books, it is necessary to listen to the files to make sure the automatic determination is satisfactory, and correct it if it is not.

To perform this prosodic editing with the DistributedEDITOR, the user loads each file in turn by selecting the edited file from the file list and pressing the Load button (see Figure 1). To then listen to the file the users presses the All button in the Speak frame at the bottom of the screen. Or, presses Sent. to speak just the current sentence. Pressing Stop at any time will stop the speech. If an inflexion is not correct, the user stops the speech and inserts the needed annotations into the sentence by placing the cursor at the appropriate place in the sentence and pressing the appropriate button in the Insert Annotation frame under the text box.

As shown in Figure 1, for the ETI-Eloquence 5 speech engine the EDITOR supports five word emphasis annotations. They are called, from left to right in the figure, Reduced, None, Normal, Added and Heavy. Six word tone annotations are supported. Again from left to right, they are called Low, High, Falling, Rising, Scooped and Downstepped. Finally, three phrase-final annotations, called Small Rise, Continuation Rise, Flat-High and Large-Fall, are supported.

After inserting an annotation the user should listen to the sentence to make sure the needed effect is produced. If not, the annotation can be removed using the buttons in the Remove Annotation frame. Multiple annotations may be inserted into a sentence. Emphasis and Tone annotations are inserted immediately preceding the word to which they apply, while Phrase-Final annotations must be inserted immediately preceding a punctuation mark.

Some speech engines also support the insertion of annotations to control pauses. While the current version of the DistributedEditor does not support these, it could easily do so using the same approach as for word emphases, word tones and phrase-final inflexions. This could be in place of or in addition to the above described automatic pause insertion technique.

## Conclusion

Since each speech engine is somewhat different, the implementation of the DistributedEDITOR for it must usually be somewhat customized. Contact us at info@DistributedAUDIO.com to find out how we can help you with your TTS text editing needs.