(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 953 970 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
03.11.1999 Bulletin 1999/44

(51) Int Cl.[6]: **G10L 5/04**

(21) Application number: **99303390.1**

(22) Date of filing: **29.04.1999**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **29.04.1998 US 67764**
29.04.1998 US 69308
30.04.1998 US 70300

(71) Applicant: **MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD.**
Kadoma-shi, Osaka 571-8501 (JP)

(72) Inventors:
- **Kuhn, Roland**
  Santa Barbara, California 93110 (US)
- **Junqua, Jean-Claude**
  Santa Barbara, California 93111 (US)
- **Contolini, Matteo**
  Santa Barbara, California 93109 (US)

(74) Representative: **Senior, Alan Murray**
J.A. KEMP & CO.,
14 South Square,
Gray's Inn
London WC1R 5LX (GB)

(54) Method and apparatus using decision trees to generate and score multiple pronunciations for a spelled word

(57) The mixed decision tree includes a network of yes-no questions about adjacent letters in a spelled word sequence and also about adjacent phonemes in the phoneme sequence corresponding to the spelled word sequence. Leaf nodes of the mixed decision tree provide information about which phonetic transcriptions are most probable. Using the mixed trees, scores are developed for each of a plurality of possible pronunciations, and these scores can be used to select the best pronunciation as well as to rank pronunciations in order of probability. The pronunciations generated by the system can be used in speech synthesis and speech recognition applications as well as lexicography applications.

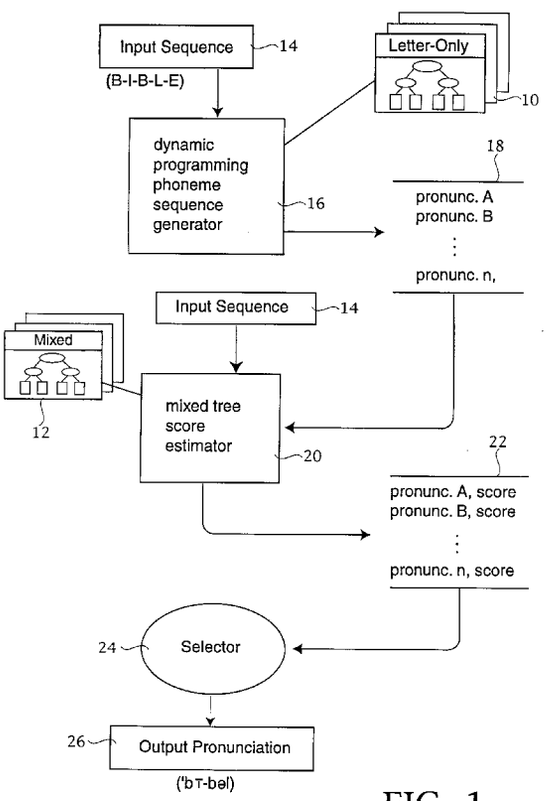

FIG. 1

EP 0 953 970 A2

Description

[0001] The present invention relates generally to speech processing. More particularly, the invention relates to a system for generating pronunciations of spelled words. The invention can be employed in a variety of different contexts, including speech recognition, speech synthesis and lexicography.

[0002] Spelled words accompanied by their pronunciations occur in many different contexts within the field of speech processing. In speech recognition phonetic transcriptions for each word in the dictionary are needed to train the recognizer prior to use. Traditionally phonetic transcriptions are manually created by lexicographers who are skilled in the nuances of phonetic pronunciation of the particular language of interest. Developing a good phonetic transcription for each word in the dictionary is time consuming and requires a great deal of skill. Much of this labor and specialized expertise could be dispensed with if there were a reliable system that could generate phonetic transcriptions of words based on their letter spelling. Such a system could extend current recognition systems to recognize words such as geographic locations and surnames that are not currently found in existing dictionaries.

[0003] Spelled words are also encountered frequently in the speech synthesis field. Present day speech synthesizers convert text to speech by retrieving digitally-sampled sound units from a dictionary and concatenating these sound units to form sentences.

[0004] As the above examples demonstrate, both the speech recognition and the speech synthesis fields of speech processing would benefit from the ability to generate accurate pronunciations from spelled words. The need for this technology is not limited to speech processing, however. Lexicographers have today completed fairly large and accurate pronunciation dictionaries for many of the major world languages. However, there still remain many hundreds of regional languages for which good phonetic transcriptions do not exist. Because the task of producing a good phonetic transcription has heretofore been largely a manual one, it may be years before some regional languages will be transcribed, if at all. The transcription process could be greatly accelerated if there were a good computer-implemented technique for scoring transcription accuracy. Such a scoring system would use an existing language transcription corpus to identify those entries in the transcription prototype whose pronunciations are suspect. This would greatly enhance the speed at which a quality transcription is generated.

[0005] Heretofore most attempts at spelled word-to-pronunciation transcription have relied solely upon the letters themselves. These techniques leave a great deal to be desired. For example, a letter-only pronunciation generator would have great difficulty properly pronouncing the word Bible. Based on the sequence of letters only the letter-only system would likely pronounce the word "Bib-l", much as a grade school child learning to read might do. The fault in conventional systems lies in the inherent ambiguity imposed by the pronunciation rules of many languages. The English language, for example, has hundreds of different pronunciation rules, making it difficult and computationally expensive to approach the problem on a word-by-word basis.

[0006] The present invention addresses the problem from a different angle. The invention uses a specially constructed mixed-decision tree that encompasses both letter sequence and phoneme sequence decision-making rules. More specifically, the mixed-decision tree embodies a series of yes-no questions residing at the internal nodes of the tree. Some of these questions involve letters and their adjacent neighbors in a spelled word sequence; other of these questions involve phonemes and their neighboring phonemes in the word sequence. The internal nodes ultimately lead to leaf nodes that contain probability data about which phonetic pronunciations of a given letter are most likely to be correct in pronouncing the word defined by its letter sequence.

[0007] The pronunciation generator of the invention uses this mixed-decision tree to score different pronunciation candidates, allowing it to select the most probable candidate as the best pronunciation for a given spelled word. Generation of the best pronunciation is preferably a two-stage process in which a letter-only tree is used in the first stage to generate a plurality of pronunciation candidates. These candidates are then scored using the mixed-decision tree in the second stage to select the best candidate.

[0008] Although the mixed-decision tree is advantageously used in a two-stage pronunciation generator, the mixed tree is useful in solving some problems that do not require letter-only first stage processing. For example, the mixed-decision tree can be used to score pronunciations generated by linguists using manual techniques.

[0009] For a more complete understanding of the invention, its objects and advantages, reference may be had to the following specification and to the accompanying drawings.

Figure **1** is a block diagram illustrating the components and steps of the invention;
Figure **2** is a tree diagram illustrating a letter-only tree;
Figure **3** is a tree diagram illustrating a mixed tree in accordance with the invention;
Figure **4** is a block diagram illustrating a presently preferred system for generating the mixed tree in accordance with the invention;
Figure **5** is a flowchart illustrating a method for generating training data through an alignment process;
Figure **6** is a block diagram illustrating use of the decision-tree in an exemplary pronunciation generator;

Figure **7** illustrates application of the Gini criterion in assessing which question to use in populating a node.
Figure **8** is a block diagram of a letter-to-sound pronunciation generator according to the invention; and
Figure **9** is a tree diagram illustrating a letter-syntax-context-dialect mixed decision tree.

[0010] To illustrate the principles of the invention the exemplary embodiment of Figure **1** shows a spelled letter-to-pronunciation generator. As will be explained more fully below, the mixed-decision tree of the invention can be used in a variety of different applications in addition to the pronunciation generator illustrated here. The pronunciation generator has been selected for illustration because it highlights many aspects and benefits of the mixed-decision tree structure.

[0011] The pronunciation generator employs two stages, the first stage employing a set of letter-only decision trees **10** and the second stage employing a set of mixed-decision trees **12**. An input sequence **14**, such as the sequence of letters B-I-B-L-E, is fed to a dynamic programming phoneme sequence generator **16**. The sequence generator uses the letter-only trees **10** to generate a list of pronunciations **18**, representing possible pronunciation candidates of the spelled word input sequence.

[0012] The sequence generator sequentially examines each letter in the sequence, applying the decision tree associated with that letter to select a phoneme pronunciation for that letter based on probability data contained in the letter-only tree.

[0013] Preferably the set of letter-only decision trees includes a decision tree for each letter in the alphabet. Figure **2** shows an example of a letter-only decision tree for the letter E. The decision tree comprises a plurality of internal nodes (illustrated as ovals in the Figure) and a plurality of leaf nodes (illustrated as rectangles in the Figure). Each internal node is populated with a yes-no question. Yes-no questions are questions that can be answered either yes or no. In the letter-only tree these questions are directed to the given letter (in this case the letter E) and its neighboring letters in the input sequence. Note in Figure **2** that each internal node branches either left or right depending on whether the answer to the associated question is yes or no.

[0014] Abbreviations are used in Figure **2** as follows: numbers in questions, such as "+1" or "-1" refer to positions in the spelling relative to the current letter. For example, "+1L=='R'?" means "Is the letter after the current letter (which in this case is the letter E) an R?" The abbreviations CONS and VOW represent classes of letters, namely consonants and vowels. The absence of a neighboring letter, or null letter, is represented by the symbol -, which is used as a filler or placeholder where aligning certain letters with corresponding phoneme pronunciations. The symbol # denotes a word boundary.

[0015] The leaf nodes are populated with probability data that associate possible phoneme pronunciations with numeric values representing the probability that the particular phoneme represents the correct pronunciation of the given letter. For example, the notation "iy=>0.51" means "the probability of phoneme 'iy' in this leaf is 0.51." The null phoneme, i.e., silence, is represented by the symbol '-'.

[0016] The sequence generator **16** (Fig. **1**) thus uses the letter-only decision trees **10** to construct one or more pronunciation hypotheses that are stored in list **18**. Preferably each pronunciation has associated with it a numerical score arrived at by combining the probability scores of the individual phonemes selected using the decision tree **10**. Word pronunciations may be scored by constructing a matrix of possible combinations and then using dynamic programming to select the n-best candidates. Alternatively, the n-best candidates may be selected using a substitution technique that first identifies the most probable word candidate and then generates additional candidates through iterative substitution, as follows.

[0017] The pronunciation with the highest probability score is selected first, by multiplying the respective scores of the highest-scoring phonemes (identified by examining the leaf nodes) and then using this selection as the most probable candidate or first-best word candidate. Additional (n-best) candidates are then selected by examining the phoneme data in the leaf nodes again to identify the phoneme, not previously selected, that has the smallest difference from an initially selected phoneme. This minimally-different phoneme is then substituted for the initially selected one to thereby generate the second-best word candidate. The above process may be repeated iteratively until the desired number of n-best candidates have been selected. List **18** may be sorted in descending score order, so that the pronunciation judged the best by the letter-only analysis appears first in the list.

[0018] As noted above, a letter-only analysis will frequently produce poor results. This is because the letter-only analysis has no way of determining at each letter what phoneme will be generated by subsequent letters. Thus a letter-only analysis can generate a high scoring pronunciation that actually would not occur in natural speech. For example, the proper name, Achilles, would likely result in a pronunciation that phoneticizes both ll's: ah-k-ih-l-l-iy-z. In natural speech, the second l is actually silent: ah-k-ih-l-iy-z. The sequence generator using letter-only trees has no mechanism to screen out word pronunciations that would never occur in natural speech.

[0019] The second stage of the pronunciation system addresses the above problem. A mixed-tree score estimator **20** uses the set of mixed-decision trees **12** to assess the viability of each pronunciation in list **18**. The score estimator works by sequentially examining each letter in the input sequence along with the phonemes assigned to each letter

by sequence generator **16**.

[0020] Like the set of letter-only trees, the set of mixed trees has a mixed tree for each letter of the alphabet. An exemplary mixed tree is shown in Figure **3**. Like the letter-only tree, the mixed tree has internal nodes and leaf nodes. The internal nodes are illustrated as ovals and the leaf nodes as rectangles in Figure **3**. The internal nodes are each populated with a yes-no question and the leaf nodes are each populated with probability data. Although the tree structure of the mixed tree resembles that of the letter-only tree, there is one important difference. The internal nodes of the mixed tree can contain two different classes of questions. An internal node can contain a question about a given letter and its neighboring letters in the sequence, or it can contain a question about the phoneme associated with that letter and neighboring phonemes corresponding to that sequence. The decision tree is thus mixed, in that it contains mixed classes of questions.

[0021] The abbreviations used in Figure **3** are similar to those used in Figure **2**, with some additional abbreviations. The symbol L represents a question about a letter and its neighboring letters. The symbol P represents a question about a phoneme and its neighboring phonemes. For example the question "+1L=='D'?" means "Is the letter in the +1 position a 'D'?" The abbreviations CONS and SYL are phoneme classes, namely consonant and syllabic. For example, the question "+1P==CONS?" means "Is the phoneme in the +1 position a consonant?" The numbers in the leaf nodes give phoneme probabilities as they did in the letter-only trees.

[0022] The mixed-tree score estimator rescores each of the pronunciations in list **18** based on the mixed-tree questions and using the probability data in the lead nodes of the mixed trees. If desired, the list of pronunciations may be stored in association with the respective score as in list **22**. If desired, list **22** can be sorted in descending order so that the first listed pronunciation is the one with the highest score.

[0023] In many instances the pronunciation occupying the highest score position in list **22** will be different from the pronunciation occupying the highest score position in list **18**. This occurs because the mixed-tree score estimator, using the mixed trees **12**, screens out those pronunciations that do not contain self-consistent phoneme sequences or otherwise represent pronunciations that would not occur in natural speech.

[0024] If desired a selector module **24** can access list **22** to retrieve one or more of the pronunciations in the list. Typically selector **24** retrieves the pronunciation with the highest score and provides this as the output pronunciation **26**.

[0025] As noted above, the pronunciation generator depicted in Figure **1** represents only one possible embodiment employing the mixed tree of the invention. As an alternative embodiment, the dynamic programming phoneme sequence generator **16**, and its associated letter-only decision trees **10** may be dispensed with in applications where one or more pronunciations for a given spelled word sequence are already available. This situation might be encountered where a previously developed pronunciation dictionary is available. In such case the mixed-tree score estimator **20**, with its associated mixed trees **12**, may be used to score the entries in the pronunciation dictionary, identifying those having low scores, thereby flagging suspicious pronunciations in the dictionary being constructed. Such a system may, for example, be incorporated into a lexicographer's productivity tool.

[0026] The output pronunciation or pronunciations selected from list **22** can be used to form pronunciation dictionaries for both speech recognition and speech synthesis applications. In the speech recognition context, the pronunciation dictionary may be used during the recognizer training phase by supplying pronunciations for words that are not already found in the recognizer lexicon. In the synthesis context the pronunciation dictionaries may be used to generate phoneme sounds for concatenated playback. The system may be used, for example, to augment the features of an E-mail reader or other text-to-speech application.

[0027] The mixed-tree scoring system of the invention can be used in a variety of applications where a single one or list of possible pronunciations is desired. For example, in a dynamic on-line dictionary the user types a word and the system provides a list of possible pronunciations, in order of probability. The scoring system can also be used as a user feedback tool for language learning systems. A language learning system with speech recognition capability is used to display a spelled word and to analyze the speaker's attempts at pronouncing that word in the new language, and the system tells the user how probable or improbable his or her pronunciation is for that word.

Generating the Decision Trees

[0028] The system for generating the letter-only trees and the mixed trees is illustrated in Figure **4**. At the heart of the decision tree generation system is tree generator **40**. The tree generator employs a tree-growing algorithm that operates upon a predetermined set of training data **42** supplied by the developer of the system. Typically the training data comprise aligned letter, phoneme pairs that correspond to known proper pronunciations of words. The training data may be generated through the alignment process illustrated in Figure **5**. Figure **5** illustrates an alignment process being performed on an exemplary word BIBLE. The spelled word **44** and its pronunciation **46** are fed to a dynamic programming alignment module **48** which aligns the letters of the spelled word with the phonemes of the corresponding pronunciation. Note in the illustrated example the final E is silent. The letter phoneme pairs are then stored as data **42**.

[0029] Returning to Figure **4**, the tree generator works in conjunction with three additional components: a set of

possible yes-no questions **50**, a set of rules **52** for selecting the best questions for each node or for deciding if the node should be a lead node, and a pruning method **53** to prevent over-training.

[0030] The set of possible yes-no questions may include letter questions **54** and phoneme questions **56**, depending on whether a letter-only tree or a mixed tree is being grown. When growing a letter-only tree, only letter questions **54** are used; when growing a mixed tree both letter questions **54** and phoneme questions **56** are used.

[0031] The rules for selecting the best question to populate at each node in the presently preferred embodiment are designed to follow the Gini criterion. Other splitting criteria can be used instead. For more information regarding splitting criteria reference may be had to Breiman, Friedman et al, "Classification and Regression Trees." Essentially, the Gini criterion is used to select a question from the set of possible yes-no questions **50** and to employ a stopping rule that decides when a node is a leaf node. The Gini criterion employs a concept called "impurity." Impurity is always a non-negative number. It is applied to a node such that a node containing equal proportions of all possible categories has maximum impurity and a node containing only one of the possible categories has a zero impurity (the minimum possible value). There are several functions that satisfy the above conditions. These depend upon the counts of each category within a node Gini impurity may be defined as follows. If C is the set of classes to which data items can belong, and T is the current tree node, let f(1IT) be the proportion of training data items in node T that belong to class 1, f(2IT) the proportion of items belonging to class 2, etc. Then,

$$i(T) = \sum_{j,k \in C, j \neq k} f(j|T)\, f(k|T) = 1 - \sum_{j} [f(j|T)]2 .$$

[0032] To illustrate by example, assume the system is growing a tree for the letter "E." In a given node T of that tree, the system may, for example, have 10 examples of how "E" is pronounced in words. In 5 of these examples, "E" is pronounced "iy" (the sound "ee" in "cheeze"); in 3 of the examples "E" is pronounced "eh" (the sound of "e" in "bed"); and in the remaining 2 examples, "E" is "-" (i.e., silent as in "e" in "maple").

[0033] Assume the system is considering two possible yes-no questions, Q_1 and Q_2 that can be applied to the 10 examples. The items that answer "yes" to Q_1 include four examples of "iy" and one example of "-" (the other five items answer "no" to Q_1.) The items that answer "yes" to Q_2 include three examples of "iy" and three examples of "eh" (the other four items answer "no" to Q_2). Figure **6** diagrammatically compares these two cases.

[0034] The Gini criterion answers which question the system should choose for this node, Q_1 or Q_2. The Gini criterion for choosing the correct question is: find the question in which the drop in impurity in going from parent nodes to children nodes is maximized. This impurity drop ΔT is defined as $\Delta I = i(T) - p_{yes}{}^* i(yes) - p_{no}{}^* i(no)$, where p_{yes} is the proportion of items going to the "yes" child and p_{no} is the proportion of items going to the "no" child.

[0035] Applying the Gini criterion to the above example:

$$i(T) = 1 - \sum_{j} [f(j/T)]^2 = 1 - 0.5^2 - 0.3^2 - 0.2^2 = 0.62$$

ΔI for Q_1 is thus:

$$i(T) - p_{yes}(Q_1) = 1 - 0.8^2 - 0.2^2 = 0.32$$

$$i(T) - p_{no}(Q_1) = 1 - 0.2^2 - 0.6^2 = 0.56$$

$$\text{So } \Delta I\ (Q_1) = 0.62 - 0.5{*}0.32 - 0.5{*}0.56 = 0.18.$$

[0036] For Q_2, we have I(yes, Q_2) = 1 - 0.5^2 - 0.5^2 = 0.5, and for i(no, Q_2) = (same) = 0.5.

$$\text{So, } \Delta I(Q_2) = 0.6 - (0.6){*}(0.5) - (0.4){*}(0.5) = 0.12.$$

[0037] In this case, Q_1 gave the greatest drop in impurity. It will therefore be chosen instead of Q_2.

[0038] The rule set **52** declares a best question for a node to be that question which brings about the greatest drop in impurity in going from the parent node to its children.

[0039] The tree generator applies the rules **52** to grow a decision tree of yes-no questions selected from set **50**. The generator will continue to grow the tree until the optimal-sized tree has been grown. Rules **52** include a set of stopping rules that will terminate tree growth when the tree is grown to a predetermined size. In the preferred embodiment the tree is grown to a size larger than ultimately desired. Then pruning methods **53** are used to cut back the tree to its desired size. The pruning method may implement the Breiman technique as described in the reference cited above.

[0040] The tree generator thus generates sets of letter-only trees, shown generally at **60** or mixed trees, shown generally at **70**, depending on whether the set of possible yes-no questions **50** includes letter-only questions alone or in combination with phoneme questions. The corpus of training data **42** comprises letter, phoneme pairs, as discussed above. In growing letter-only trees, only the letter portions of these pairs are used in populating the internal nodes. Conversely, when growing mixed trees, both the letter and phoneme components of the training data pairs may be used to populate internal nodes. In both instances the phoneme portions of the pairs are used to populate the leaf nodes. Probability data associated with the phoneme data in the lead nodes are generated by counting the number of occurrences a given phoneme is aligned with a given letter over the training data corpus.

[0041] The letter-to-pronunciation decision trees generated by the above-described method can be stored in memory for use in a variety of different speech-processing applications. While these applications are many and varied, a few examples will next be presented to better highlight some of the capabilities and advantages of these trees.

[0042] Figure **6** illustrates the use of both the letter-only trees and the mixed trees to generate pronunciations from spelled-word letter sequences. Although the illustrated embodiment employs both letter-only and mixed tree components together, other applications may use only one component and not the other. In the illustrated embodiment the set of letter-only trees are stored in memory at **80** and the mixed trees are stored in memory at **82**. In many applications there will be one tree for each letter in the alphabet. Dynamic programming sequence generator **84** operates upon input sequence **86** to generate a pronunciation at **88** based on the letter-only trees **80**. Essentially, each letter in the input sequence is considered individually and the applicable letter-only tree is used to select the most probable pronunciation for that letter. As explained above, the letter-only trees ask a series of yes-no questions about the given letter and its neighboring letters in the sequence. After all letters in the sequence have been considered, the resultant pronunciation is generated by concatenating the phonemes selected by the sequence generator.

[0043] To improve pronunciation the mixed tree set **82** can be used. Whereas letter-only trees ask only questions about letters, the mixed trees can ask questions about letters and also about phonemes. Scorer **90** may receive phoneme information from the output of sequence generator **84**. In this regard, sequence generator **84**, using the letter-only trees **80**, can generate a plurality of different pronunciations, sorting those pronunciations based on their respective probability scores. This sorted lists of pronunciations may be stored at **92** for access by the scorer **90**.

[0044] Scorer **90** receives as input the same input sequence **86** as was supplied to sequence generator **84**. Scorer **90** applies the mixed-tree **82** questions to the sequence of letters, using data from store **92** when asked to respond to a phoneme question. The resulting output at **94** is typically a better pronunciation than provided at **88**. The reason for this is the mixed trees tend to filter out pronunciations that would not occur in natural speech. For example, the proper name, Achilles, would likely result in a pronunciation that phoneticizes both ll's: ah-k-ih-l-l-iy-z. In natural speech, the second l is actually silent: ah-k-ih-l-iy-z.

[0045] If desired, scorer generator **90** can also produce a sorted list of n possible pronunciations as at **96**. The scores associated with each pronunciation represent the composite of the individual probability scores assigned to each phoneme in the pronunciation. These scores can, themselves, be used in applications where dubious pronunciations need to be identified. For example, the phonetic transcription supplied by a team of lexicographers could be checked using the mixed trees to quickly identify any questionable pronunciations.

A Letter-to-Sound Pronunciation Generator

[0046] To illustrate the principles of the invention the exemplary embodiment of Figure **8** shows a two stage spelled letter-to-pronunciation generator. As will be explained more fully below, the mixed-decision tree approach of the invention can be used in a variety of different applications in addition to the pronunciation generator illustrated here. The two stage pronunciation generator has been selected for illustration because it highlights many aspects and benefits of the mixed-decision tree structure.

[0047] The two stage pronunciation generator includes a first stage **116** which preferably employs a set of letter-syntax-context-dialect decision trees **110** and a second stage **120** which employs a set of phoneme-mixed decision trees **112** which examine input sequence **114** at a phoneme level. Letter-syntax-context-dialect decision trees examine questions involving letters and their adjacent neighbors in a spelled word sequence (i.e., letter-related questions); other questions examined are what words precede or follow a particular word (i.e., context-related questions); still other questions examined are what part of speech the word has within a sentence as well as what syntax other words have

in the sentence (i.e., syntax-related questions); still further questions examined are what dialect it is desired to be spoken. Preferably, a user selects which dialect is to be spoken by dialect selection device **150**.

[0048] An alternate embodiment of the present invention includes using letter-related questions and at least one of the word-level characteristics (i.e., syntax-related questions or context-related questions). For example, one embodiment utilizes a set of letter-syntax decision trees for the first stage. Another embodiment utilizes a set of letter-context-dialect decision trees which do not examine syntax of the input sequence.

[0049] It should be understood that the present invention is not limited to words occurring in a sentence, but includes other linguistical constructs which exhibit syntax, such as fragmented sentences or phrases.

[0050] An input sequence **114**, such as the sequence of letters of a sentence, is fed to the text-based pronunciation generator **116**. For example, input sequence **114** could be the following sentence: "Did you know who read the autobiography?"

[0051] Syntax data **115** is an input to text-based pronunciation generator **116**. This input provides information for the text-based pronunciation generator **116** to correctly course through the letter-syntax-context-dialect decision trees **110**. Syntax data **115** addresses what parts of speech each word has in the input sequence **114**. For example, the word "read" in the above input sequence example would be tagged as a verb (as opposed to a noun or an adjective) by syntax tagger software module **129**. Syntax tagger software technology is available from such institutions as the University Pennsylvania under project "Xtag." Moreover, the following reference discusses syntax tagger software technology: George Foster, "Statistical Lexical Disambiguation", Masters Thesis in Computer Science, McGill University, Montreal, Canada (November 11, 1991).

[0052] The text-based pronunciation generator **116** uses decision trees **110** to generate a list of pronunciations **118**, representing possible pronunciation candidates of the spelled word input sequence. Each pronunciation (e.g., pronunciation A) of list **118** represents a pronunciation of input sequence **114** including preferably how each word is stressed. Moreover, the rate at which each word is spoken is determined in the preferred embodiment.

[0053] Sentence rate calculator software module **152** is utilized by text-based pronunciation generator **116** to determine how quickly each word should be spoken. For example, sentence rate calculator **152** examines the context of the sentence to determine if certain words in the sentence should be spoken at a faster or slower rate than normal. For example, a sentence with an exclamation marker at the end produces rate data which indicates that a predetermined number of words before the end of the sentence are to have a shorter duration than normal to better convey the impact of an exclamatory statement.

[0054] The text-based pronunciation generator **116** examines in order each letter and word in the sequence, applying the decision tree associated with that letter or word's syntax (or word's context) to select a phoneme pronunciation for that letter based on probability data contained in the decision tree. Preferably the set of decision trees **110** includes a decision tree for each letter in the alphabet and syntax of the language involved.

[0055] Figure **9** shows an example of a letter-syntax-context-dialect decision tree **140** applicable to the letter "E" in the word "READ." The decision tree comprises a plurality of internal nodes (illustrated as ovals in the Figure) and a plurality of leaf nodes (illustrated as rectangles in the Figure). Each internal node is populated with a yes-no question. Yes-no questions are questions that can be answered either yes or no. In the letter-syntax-context-dialect decision tree **140** these questions are directed to: a given letter (e.g., in this case the letter "E") and its neighboring letters in the input sequence; or the syntax of the word in the sentence (e.g., noun, verb, etc.); or the context and dialect of the sentence. Note in Figure **9** that each internal node branches either left or right depending on whether the answer to the associated question is yes or no.

[0056] Preferably, the first internal node inquires about the dialect to be spoken. Internal node **138** is representative of such an inquiry. If the southern dialect is to be spoken, then southern dialect decision tree **139** is coursed through which ultimately produces phoneme values at the leaf nodes which are more distinctive of a southern dialect.

[0057] The abbreviations used in Figure **9** are as follows: numbers in questions, such as "+1" or "-1" refer to positions in the spelling relative to the current letter. The symbol L represents a question about a letter and its neighboring letters. For example, "-1L=='R' or 'L'?" means "is the letter before the current letter (which is 'E') an 'L' or an 'R'?". Abbreviations 'CONS' and 'VOW' are classes of letters: consonant and vowel. The symbol '#' indicates a word boundary. The term 'tag(i)' denotes a question about the syntactic tag of the ith word, where i=0 denotes the current word, i=-1 denotes the preceding word, i=+1 denotes the following word, etc. Thus, "tag(0)==PRES?" means "is the current word a present-tense verb?".

[0058] The leaf nodes are populated with probability data that associate possible phoneme pronunciations with numeric values representing the probability that the particular phoneme represents the correct pronunciation of the given letter. The null phoneme, i.e., silence, is represented by the symbol '-'.

[0059] For example, the "E" in the present-tense verbs "READ" and "LEAD" is assigned its correct pronunciation, "iy" at leaf node **142** with probability 1.0 by the decision tree **140**. The "E" in the past tense of "read" (e.g., "Who read a book") is assigned pronunciation "eh" at leaf node **144** with probability 0.9.

[0060] Decision trees **110** (of Figure **8**) preferably includes context-related questions. For example, context-related

question of internal nodes may examine whether the word "you" is preceded by the word "did." In such a context, the "y" in "you" is typically pronounced in colloquial speech as "ja".

[0061] The present invention also generates prosody-indicative data, so as to convey stress, pitch, grave, or pause aspects when speaking a sentence. Syntax-related questions help to determine how the phoneme is to be stressed, or pitched or graved. For example, internal node **141** (of Figure **9**) inquires whether the first word in the sentence is an interrogatory pronoun, such as "who" in the exemplary sentence "who read a book?" Since in this example, the first word in this example is an interrogatory pronoun, then leaf node **144** with its phoneme stress is selected. Leaf node **146** illustrates the other option where the phonemes are not stressed.

[0062] As another example, in an interrogative sentence, the phonemes of the last syllable of the last word in the sentence would have a pitch mark so as to more naturally convey the questioning aspect of the sentence. Still another example includes the present invention able to accommodate natural pausing in speaking a sentence. The present invention includes such pausing detail by asking questions about punctuation, such as commas and periods.

[0063] The text-based pronunciation generator **116** (Fig. **8**) thus uses decision trees **110** to construct one or more pronunciation hypotheses that are stored in list **118**. Preferably each pronunciation has associated with it a numerical score arrived at by combining the probability scores of the individual phonemes selected using decision trees **110**. Word pronunciations may be scored by constructing a matrix of possible combinations and then using dynamic programming to select the n-best candidates.

[0064] Alternatively, the n-best candidates may be selected using a substitution technique that first identifies the most probable word candidate and then generates additional candidates through iterative substitution, as follows. The pronunciation with the highest probability score is selected first, by multiplying the respective scores of the highest-scoring phonemes (identified by examining the leaf nodes) and then using this selection as the most probable candidate or first-best word candidate. Additional (n-best) candidates are then selected by examining the phoneme data in the leaf nodes again to identify the phoneme, not previously selected, that has the smallest difference from an initially selected phoneme. This minimally-different phoneme is then substituted for the initially selected one to thereby generate the second-best word candidate. The above process may be repeated iteratively until the desired number of n-best candidates have been selected. List **118** may be sorted in descending score order, so that the pronunciation judged the best by the letter-only analysis appears first in the list.

[0065] Decision trees **110** frequently produce only moderately successful results. This is because these decision trees have no way of determining at each letter what phoneme will be generated by subsequent letters. Thus decision trees **110** can generate a high scoring pronunciation that actually would not occur in natural speech. For example, the proper name, Achilles, would likely result in a pronunciation that phoneticizes both ll's: ah-k-ih-l-l-iy-z. In natural speech, the second l is actually silent: ah-k-ih-l-iy-z. The pronunciation generator using decision trees **110** has no mechanism to screen out word pronunciations that would never occur in natural speech.

[0066] The second stage **120** of the pronunciation system **108** addresses the above problem. A phoneme-mixed tree score estimator **120** uses the set of phoneme-mixed decision trees **112** to assess the viability of each pronunciation in list **118**. The score estimator **120** works by sequentially examining each letter in the input sequence **114** along with the phonemes assigned to each letter by text-based pronunciation generator **116**.

[0067] The phoneme-mixed tree score estimator **120** rescores each of the pronunciations in list **118** based on the phoneme-mixed tree questions **112** and using the probability data in the leaf nodes of the mixed trees. If desired, the list of pronunciations may be stored in association with the respective score as in list **122**. If desired, list **122** can be sorted in descending order so that the first listed pronunciation is the one with the highest score.

[0068] In many instances the pronunciation occupying the highest score position in list **122** will be different from the pronunciation occupying the highest score position in list **118**. This occurs because the phoneme-mixed tree score estimator **120**, using the phoneme-mixed trees **112**, screens out those pronunciations that do not contain self-consistent phoneme sequences or otherwise represent pronunciations that would not occur in natural speech.

[0069] In the preferred embodiment, phoneme-mixed tree score estimator **120** utilizes sentence rate calculator **152** in order to determine rate data for the pronunciations in list **122**. Moreover, estimator **120** utilizes phoneme-mixed trees that allow questions about dialect to be examined and that also allow questions to determine stress and other prosody aspects at the leaf nodes in a manner similar to the aforementioned approach.

[0070] If desired a selector module **124** can access list **122** to retrieve one or more of the pronunciations in the list. Typically selector **124** retrieves the pronunciation with the highest score and provides this as the output pronunciation **126**.

[0071] As noted above, the pronunciation generator depicted in Figure **8** represents only one possible embodiment employing the mixed tree approach of the invention. In an alternate embodiment, the output pronunciation or pronunciations selected from list **122** can be used to form pronunciation dictionaries for both speech recognition and speech synthesis applications. In the speech recognition context, the pronunciation dictionary may be used during the recognizer training phase by supplying pronunciations for words that are not already found in the recognizer lexicon. In the synthesis context the pronunciation dictionaries may be used to generate phoneme sounds for concatenated playback.

The system may be used, for example, to augment the features of an E-mail reader or other text-to-speech application.

[0072] The mixed-tree scoring system (i.e., letter, syntax, context, and phoneme) of the invention can be used in a variety of applications where a single one or list of possible pronunciations is desired. For example, in a dynamic on-line language learning system, a user types a sentence, and the system provides a list of possible pronunciations for the sentence, in order of probability. The scoring system can also be used as a user feedback tool for language learning systems. A language learning system with speech recognition capability is used to display a spelled sentence and to analyze the speaker's attempts at pronouncing that sentence in the new language. The system indicates to the user how probable or improbable his or her pronunciation is for that sentence.

[0073] While the invention has been described in its presently preferred form it will be understood that there are numerous applications for the mixed-tree pronunciation system. Accordingly, the invention is capable of certain modifications and changes without departing from the scope bf the invention as set forth in the appended claims.

[0074] The technical effect of the present invention may be realised by a suitably programmed computer and the present invention also provides a computer program product comprising a computer readable storage medium having recorded thereon computer interpretable or compilable code that, when loaded onto a suitable computer and executed, will realise the technical effect. The present invention also encompasses such code itself.

Claims

1. An apparatus for generating at least one phonetic pronunciation for an input sequence of letters selected from a predetermined alphabet, comprising:

   a memory for storing a plurality of letter-only decision trees corresponding to said alphabet,
   said letter-only decision trees having internal nodes representing yes-no questions about a given letter and its neighboring letters in a given sequence;
   said memory further storing a plurality of mixed decision trees corresponding to said alphabet,
   said mixed decision trees having a first plurality of internal nodes representing yes-no questions about a given letter and its neighboring letters in said given sequence and having a second plurality of internal nodes representing yes-no questions about a phoneme and its neighboring phonemes in said given sequence,
   said letter-only decision trees and said mixed decision trees further having leaf nodes representing probability data that associates said given letter with a plurality of phoneme pronunciations;
   a phoneme sequence generator coupled to said letter-only decision tree for processing an input sequence of letters and generating a first set of phonetic pronunciations corresponding to said input sequence of letters;
   a score estimator coupled to said mixed decision tree for processing said first set to generate a second set of scored phonetic pronunciations, the scored phonetic pronunciations representing at least one phonetic pronunciation of said input sequence.

2. The apparatus of claim **1** wherein said second set comprises a plurality of pronunciations each with an associated score derived from said probability data and further comprising a pronunciation selector receptive of said second set and operable to select one pronunciation from said second set based on said associated score.

3. The apparatus of claim **1** or **2** wherein said phoneme sequence generator produces a predetermined number of different pronunciations corresponding to given input sequence.

4. The apparatus of claim **3** wherein said phone me sequence generator produces a predetermined number of different pronunciations representing the n-best pronunciations according to said probability data.

5. The apparatus of claim **4** wherein said score estimator rescores said n-best pronunciations based on said mixed decision trees.

6. The apparatus of any one of claims **1** to **5** wherein said sequence generator constructs a matrix of possible phoneme combinations representing different pronunciations.

7. The apparatus of claim **6** wherein sequence generator selects the n-best phoneme combinations from said matrix using dynamic programming.

8. The apparatus of claim **6** wherein sequence generator selects the n-best phoneme combinations from said matrix by iterative substitution.

9. The apparatus of any one of claims **1** to **8** further comprising a speech recognition system having a pronunciation dictionary used for recognizer training and wherein at least a portion of said second set populates said dictionary to supply pronunciations for words based on their spelling.

10. The apparatus of any one of claims **1** to **9** further comprising a speech synthesis system receptive of at least a portion of said second set for generating an audible synthesized pronunciation of words based on their spelling.

11. The apparatus of claim **10** wherein said speech synthesis system is incorporated into an e-mail reader.

12. The apparatus of claim **10** wherein said speech synthesis system is incorporated into a dictionary for providing a list of possible pronunciations in order of probability.

13. The apparatus of any one of claims **1** to **10** further comprising a language learning system that displays a spelled word and analyzes a speaker's attempt at pronouncing that word using at least one of said letter-only decision tree and said mixed decision tree to tell the speaker how probable his or her pronunciation was for that word.

14. A method for processing spelling-to-pronunciation data, comprising the steps of:

providing a first set of yes-no questions about letters and their relationship to neighboring letters in an input sequence;
providing a second set of yes-no questions about phonemes and their relationship to neighboring phonemes in an input sequence;
providing a corpus of training data representing a plurality of different sets of pairs each pair containing a letter sequence and a phoneme sequence, said letter sequence selected from an alphabet;
using said first and second sets and said training data to generate decision trees for at least a portion of said alphabet, said decision trees each having a plurality of internal nodes and a plurality of leaf nodes;
populating said internal nodes with questions selected from said first and second sets; and
populating said leaf nodes with probability data that associates said portion of said alphabet with a plurality of phoneme pronunciations based on said training data.

15. The method of claim **14** further comprising providing said corpus of training data as aligned letter sequence-phoneme sequence pairs.

16. The method of claim **14** or **15** wherein said step of providing a corpus of training data further comprises providing a plurality of input sequences containing sequences of phonemes representing pronunciation of words formed by said sequence of letters; and aligning selected ones of said phonemes with selected ones of said letters to define aligned letter-phoneme pairs.

17. The method of claim **14, 15** or **16** further comprising supplying an input string of letters with at least one associated phoneme pronunciation and using said decision trees to score said pronunciation based on said probability data.

18. The method of claim **14, 15** or **16** further comprising supplying an input string of letters with a plurality of associated phoneme pronunciations and using decision trees to select one of said plurality of pronunciation based on said probability data.

19. The method of claim **14, 15** or **16** further comprising supplying an input string of letters representing a word with a plurality of associated phoneme pronunciations and using said decision trees to generate a phonetic transcription of said word based on said probability data.

20. The method of claim **19** further comprising using said phonetic transcription to populate a dictionary associated with a speech recognizer.

21. The method of claim **14, 15** or **16** further comprising supplying an input string of letters representing a word with a plurality of associated phoneme pronunciations and using decision trees to assign a numerical score to each one of said plurality of pronunciations.

22. An apparatus for generating at least one phonetic pronunciation for an input sequence of letters selected from a predetermined alphabet, said sequence of letters forming words which substantially adhere to a predetermined

syntax, said apparatus comprising:

an input device for receiving syntax data indicative of the syntax of said words in said input sequence;
a computer storage device for storing a plurality of text-based decision trees having questions indicative of predetermined characteristics of said input sequence,
said predetermined characteristics including letter-related questions about said input sequence, said predetermined characteristics also including characteristics selected from the group consisting of syntax-related questions, context-related questions, dialect-related questions or combinations thereof,
said text-based decision trees having internal nodes representing questions about predetermined characteristics of said input sequence;
said text-based decision trees further having leaf nodes representing probability data that associates each of said letters with a plurality of phoneme pronunciations; and
a text-based pronunciation generator connected to said text-based decision trees for processing said input sequence of letters and generating a first set of phonetic pronuciations corresponding to said input sequence of letters based upon said text-based decision trees.

23. The apparatus of claim **22** further comprising:
a phoneme-mixed tree score estimator connected to said text-based pronunciation generator for processing said first set to generate a second set of scored phonetic pronunciations, the scored phonetic pronunciations representing at least one phonetic pronunciation of said input sequence.

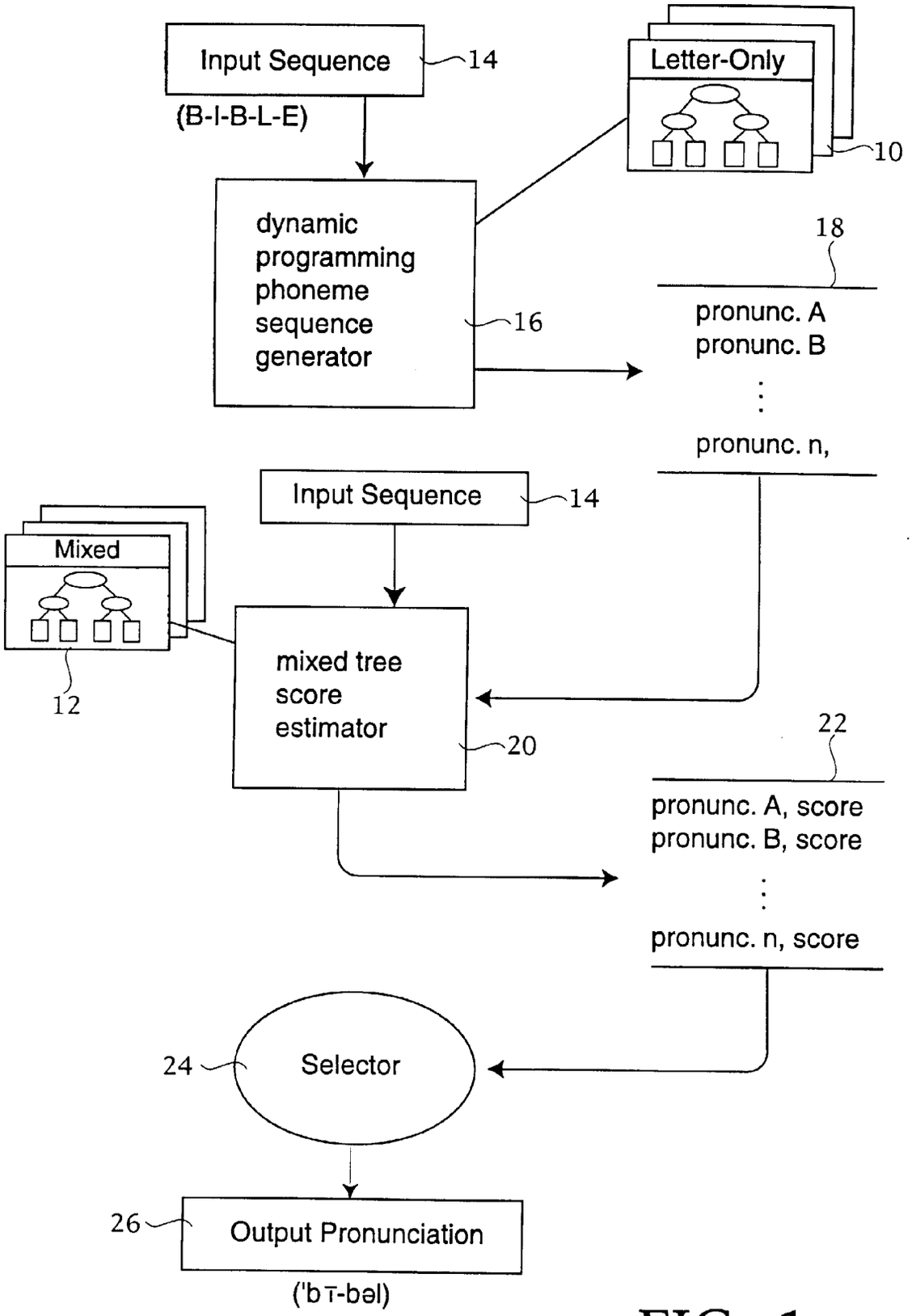
Input Sequence
14
(B-I-B-L-E)
Letter-Only
10
dynamic programming phoneme sequence generator
16
18
pronunc. A
pronunc. B
pronunc. n,
Input Sequence
14
Mixed
12
mixed tree score estimator
20
22
pronunc. A, score
pronunc. B, score
pronunc. n, score
24
Selector
26
Output Pronunciation
('bī-bəl)

FIG. 1

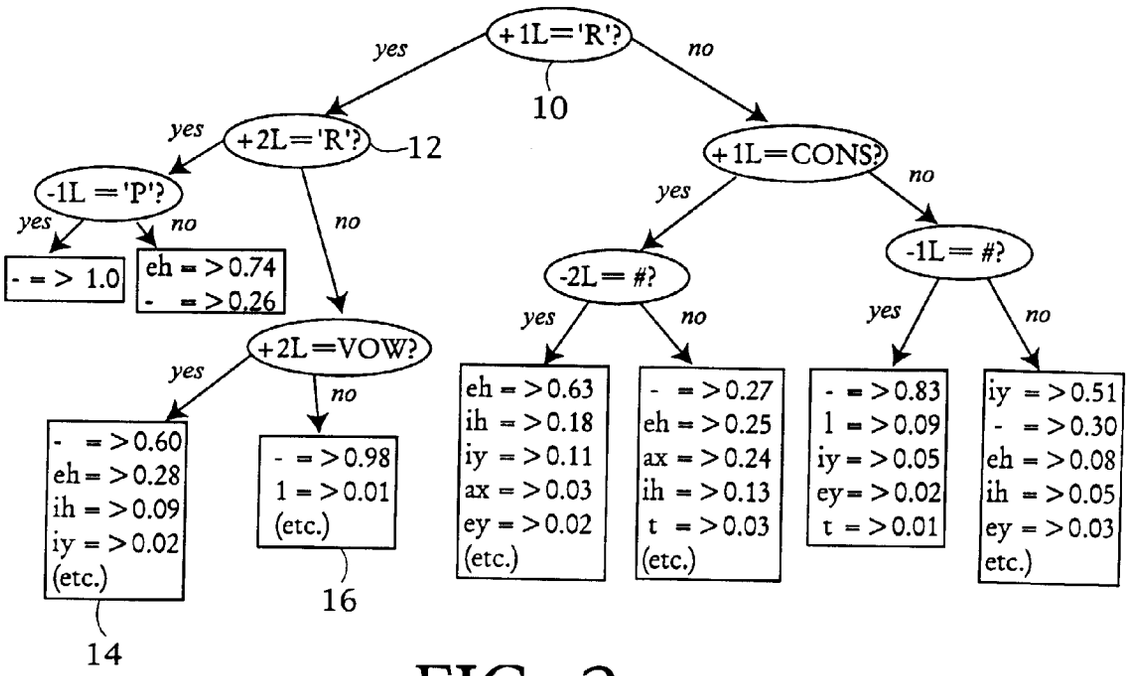
+1L='R'?
yes
no
10
+2L='R'?
12
-1L ='P'?
yes
no
- = > 1.0
eh = > 0.74
- = > 0.26
+2L=VOW?
yes
no
- = > 0.60
eh = > 0.28
ih = > 0.09
iy = > 0.02
(etc.)
14
- = > 0.98
1 = > 0.01
(etc.)
16
+1L=CONS?
yes
no
-2L = #?
yes
no
eh = > 0.63
ih = > 0.18
iy = > 0.11
ax = > 0.03
ey = > 0.02
(etc.)
- = > 0.27
eh = > 0.25
ax = > 0.24
ih = > 0.13
t = > 0.03
(etc.)
-1L = #?
yes
no
- = > 0.83
l = > 0.09
iy = > 0.05
ey = > 0.02
t = > 0.01
iy = > 0.51
- = > 0.30
eh = > 0.08
ih = > 0.05
ey = > 0.03
etc.)

FIG. 2

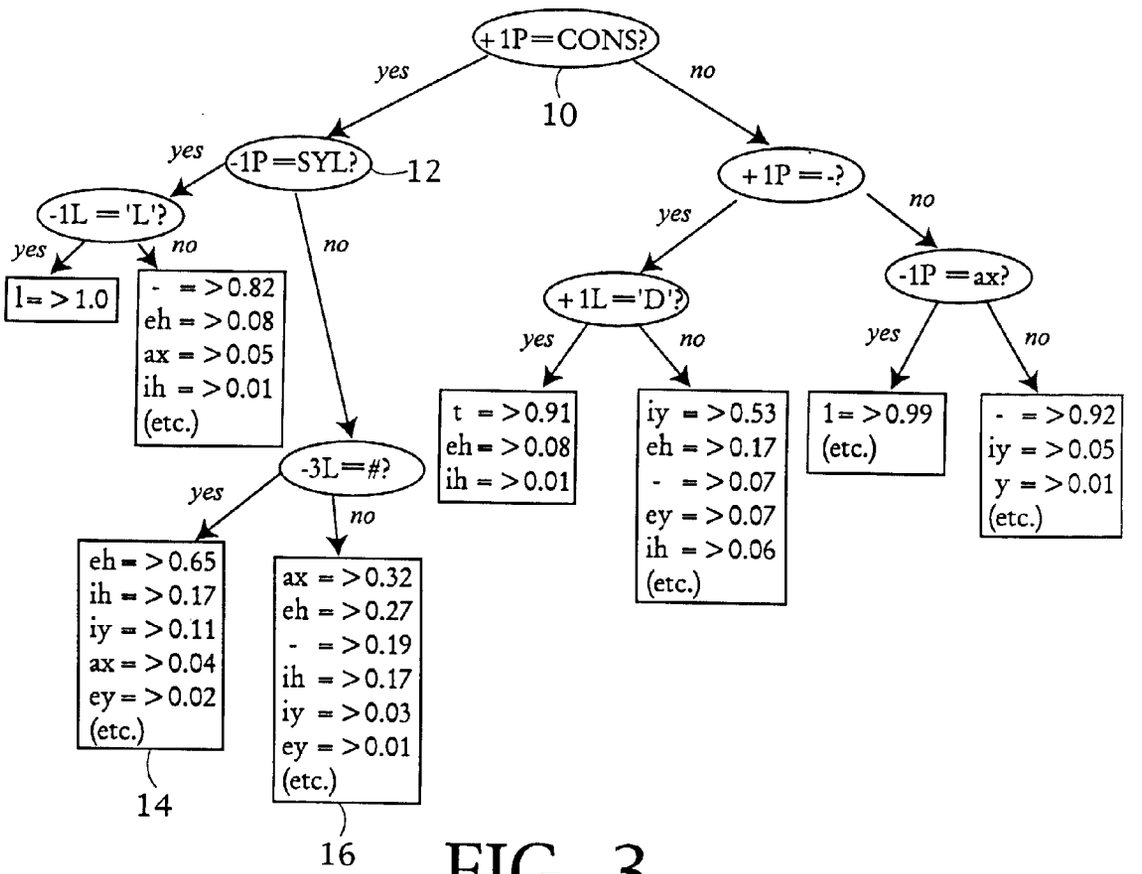
+1P=CONS?
yes
no
10
-1P=SYL?
12
yes
no
-1L ='L'?
yes
no
l= > 1.0
- = > 0.82
eh = > 0.08
ax = > 0.05
ih = > 0.01
(etc.)
-3L=#?
yes
no
eh = > 0.65
ih = > 0.17
iy = > 0.11
ax = > 0.04
ey = > 0.02
(etc.)
14
ax = > 0.32
eh = > 0.27
- = > 0.19
ih = > 0.17
iy = > 0.03
ey = > 0.01
(etc.)
16
+1P = -?
yes
no
+1L='D'?
yes
no
t = > 0.91
eh = > 0.08
ih = > 0.01
iy = > 0.53
eh = > 0.17
- = > 0.07
ey = > 0.07
ih = > 0.06
(etc.)
-1P =ax?
yes
no
1= > 0.99
(etc.)
- = > 0.92
iy = > 0.05
y = > 0.01
(etc.)

FIG. 3

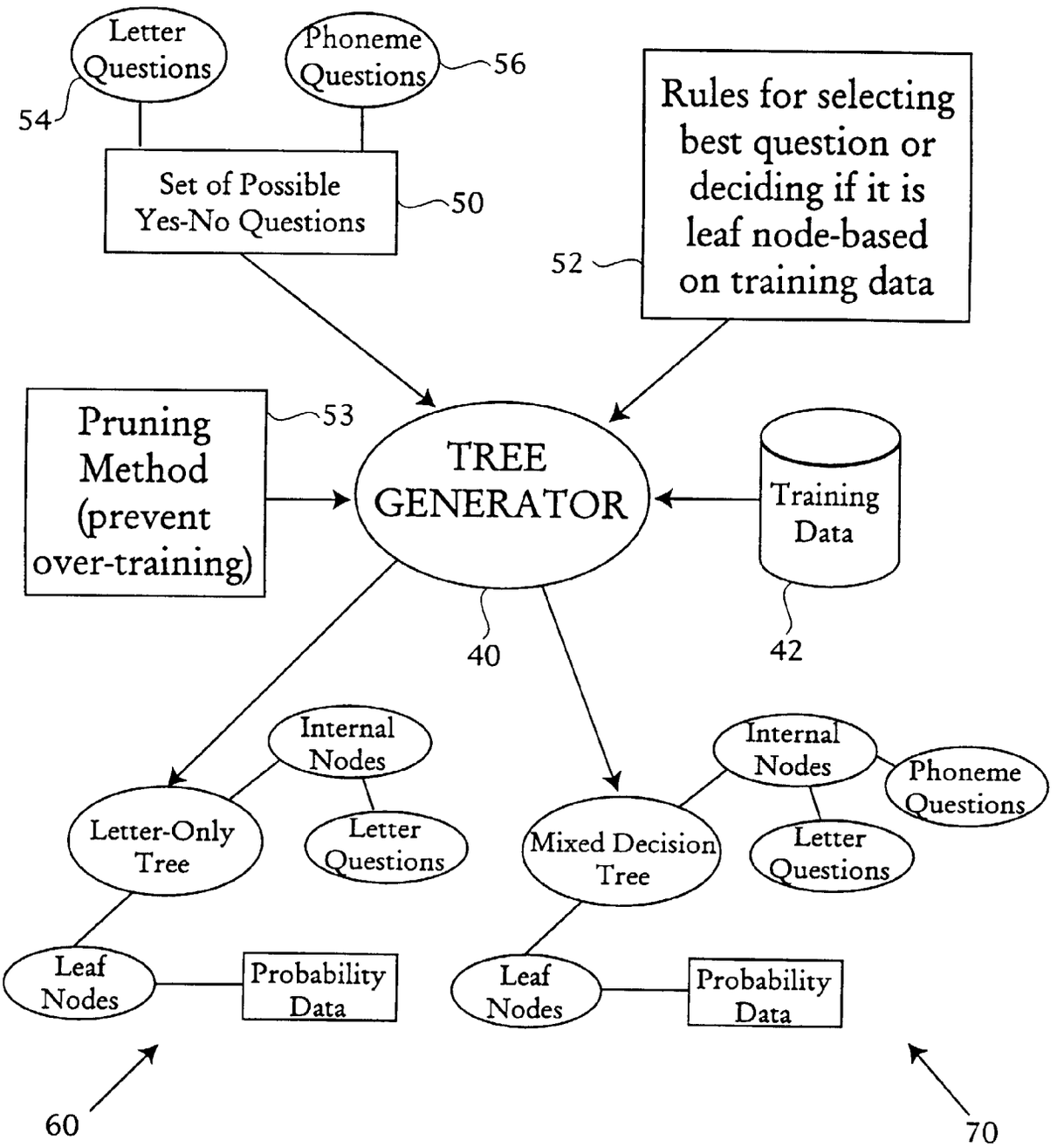
Letter Questions
54
Phoneme Questions
56
Set of Possible Yes-No Questions
50
Rules for selecting best question or deciding if it is leaf node-based on training data
52
Pruning Method (prevent over-training)
53
TREE GENERATOR
Training Data
40
42
Internal Nodes
Letter-Only Tree
Letter Questions
Leaf Nodes
Probability Data
60
Internal Nodes
Phoneme Questions
Mixed Decision Tree
Letter Questions
Leaf Nodes
Probability Data
70

FIG. 4

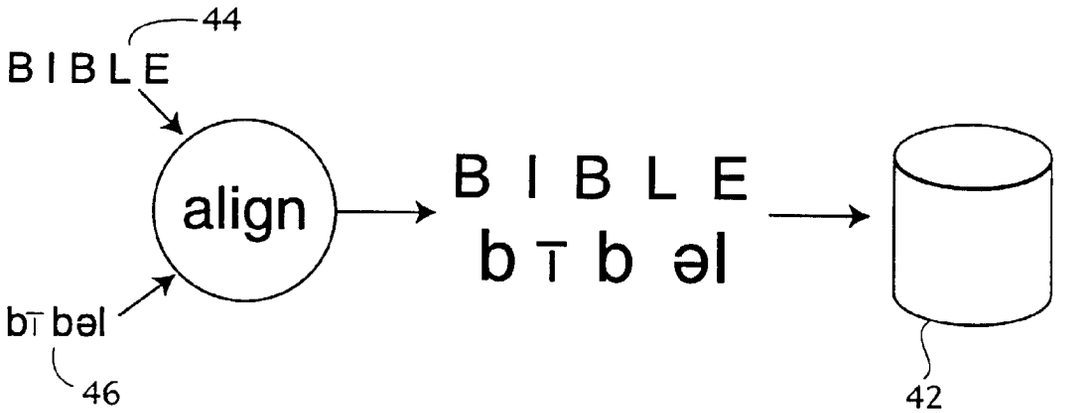
44
B I B L E
align
B I B L E
b ɪ b əl
bɪ bəl
46
42

FIG. 5

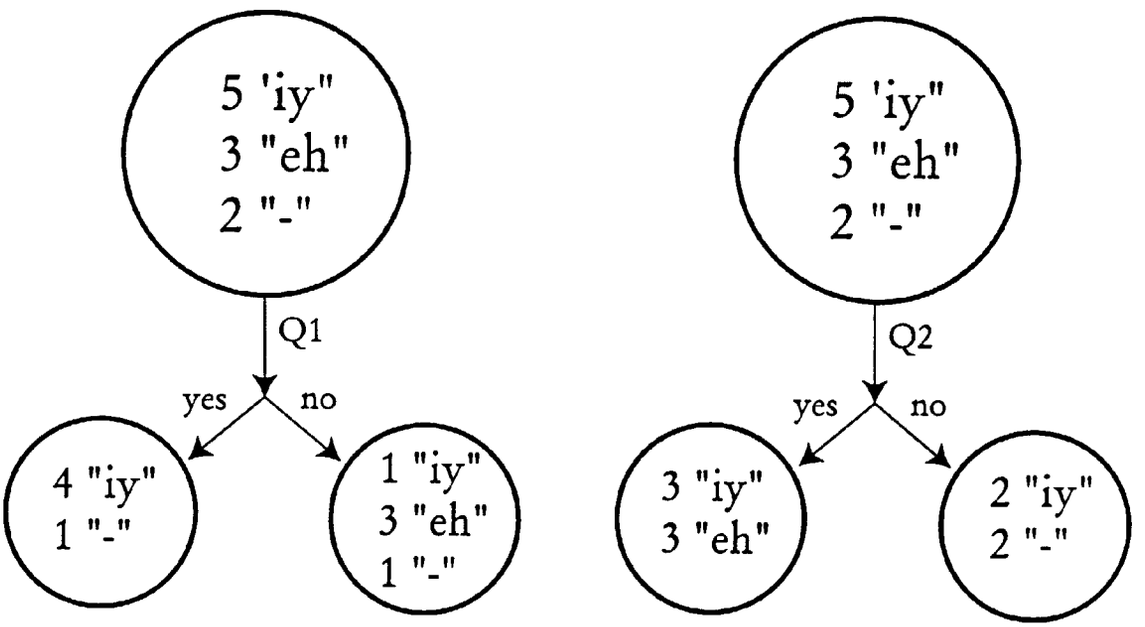
5 'iy"
3 "eh"
2 "-"
Q1
yes
no
4 "iy"
1 "-"
1 "iy"
3 "eh"
1 "-"
5 'iy"
3 "eh"
2 "-"
Q2
yes
no
3 "iy"
3 "eh"
2 "iy"
2 "-"

FIG. 7

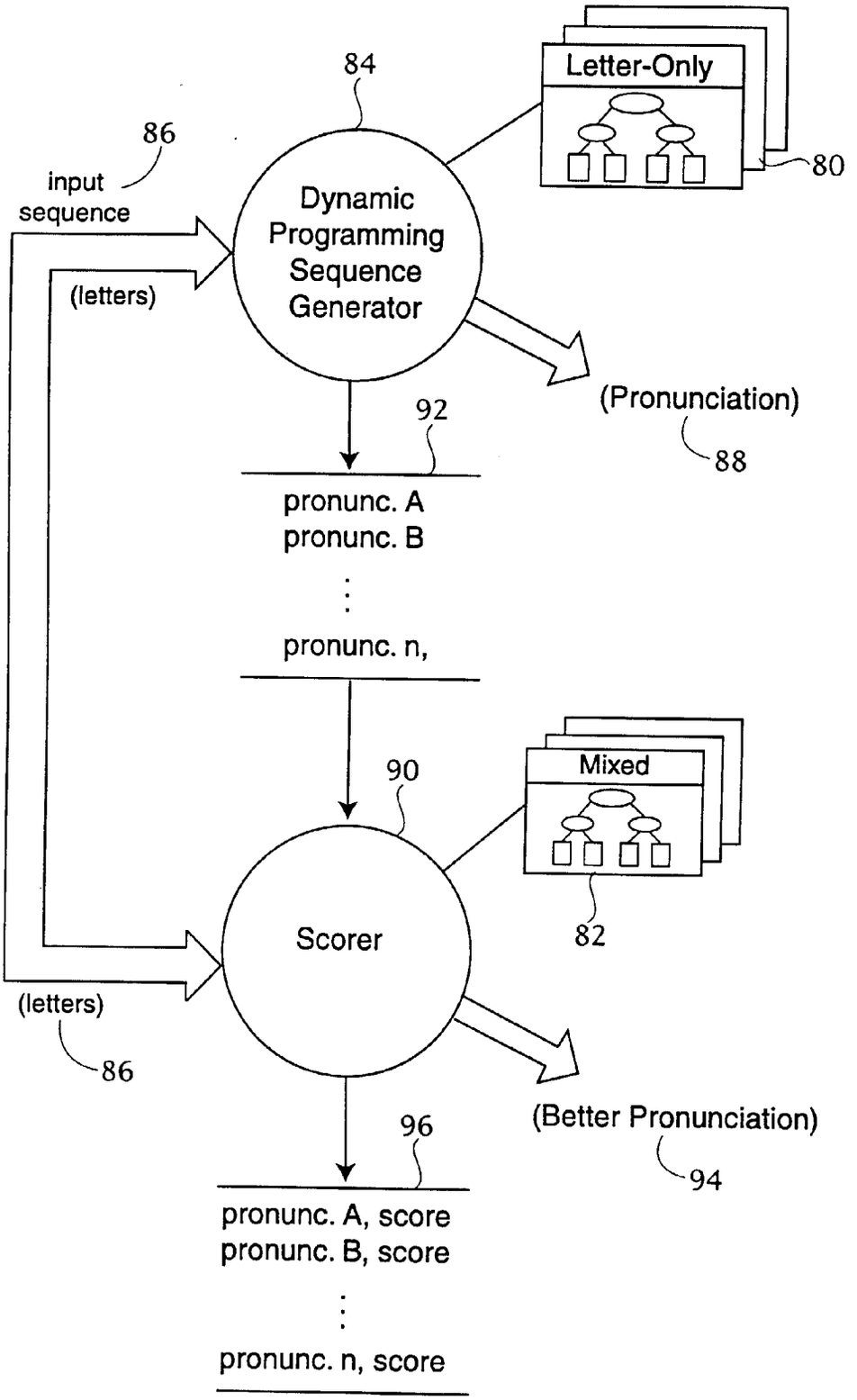
84
Letter-Only
80
86
input
sequence
Dynamic
Programming
Sequence
Generator
(letters)
(Pronunciation)
88
92
pronunc. A
pronunc. B
pronunc. n,
90
Mixed
82
Scorer
(letters)
86
(Better Pronunciation)
94
96
pronunc. A, score
pronunc. B, score
pronunc. n, score

FIG. 6

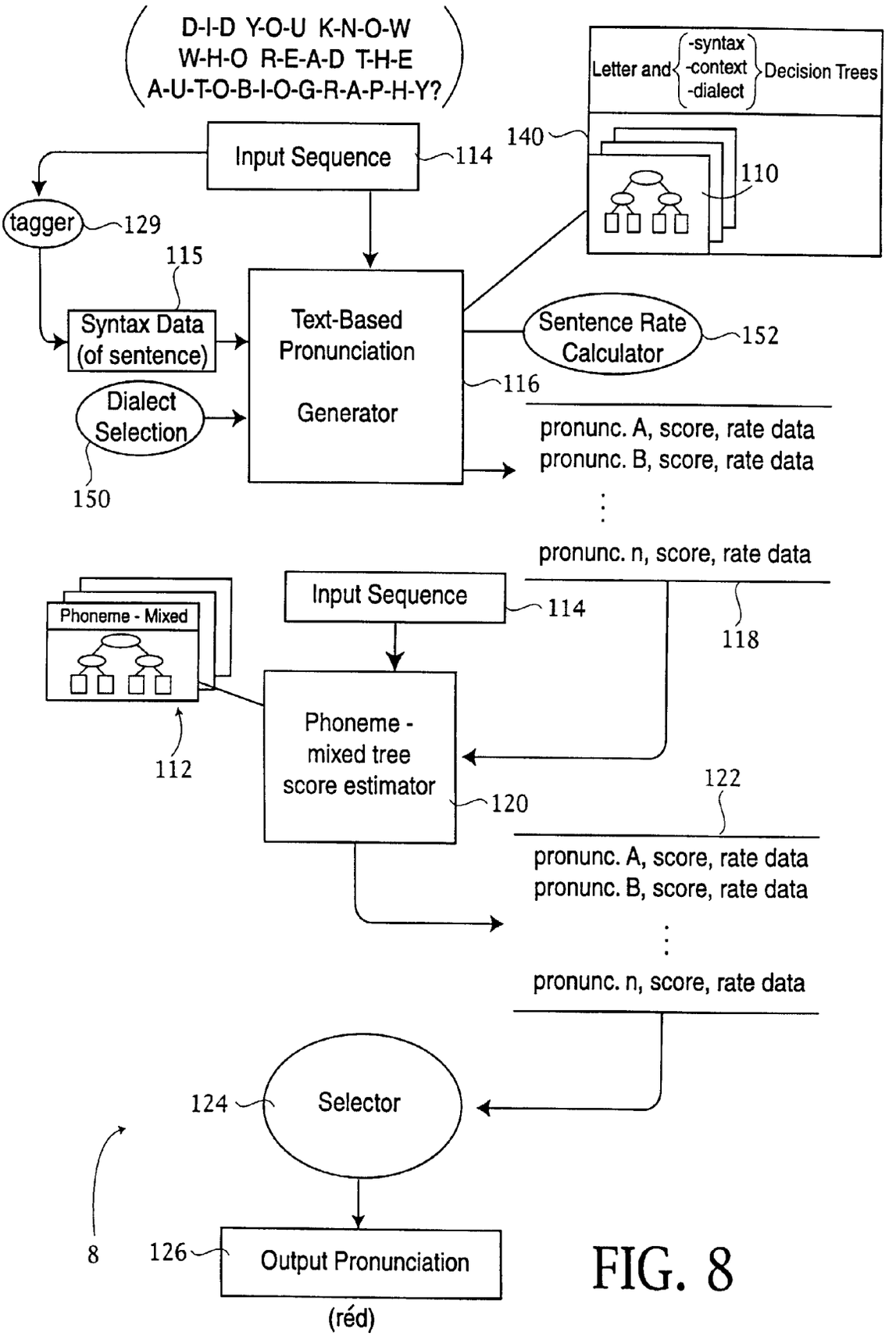
D-I-D Y-O-U K-N-O-W
W-H-O R-E-A-D T-H-E
A-U-T-O-B-I-O-G-R-A-P-H-Y?
Input Sequence
114
Letter and -syntax -context -dialect Decision Trees
140
110
tagger
129
115
Syntax Data (of sentence)
Text-Based Pronunciation Generator
116
Sentence Rate Calculator
152
Dialect Selection
150
pronunc. A, score, rate data
pronunc. B, score, rate data
pronunc. n, score, rate data
Phoneme - Mixed
Input Sequence
114
118
112
Phoneme - mixed tree score estimator
120
122
pronunc. A, score, rate data
pronunc. B, score, rate data
pronunc. n, score, rate data
124
Selector
8
126
Output Pronunciation
(réd)

FIG. 8

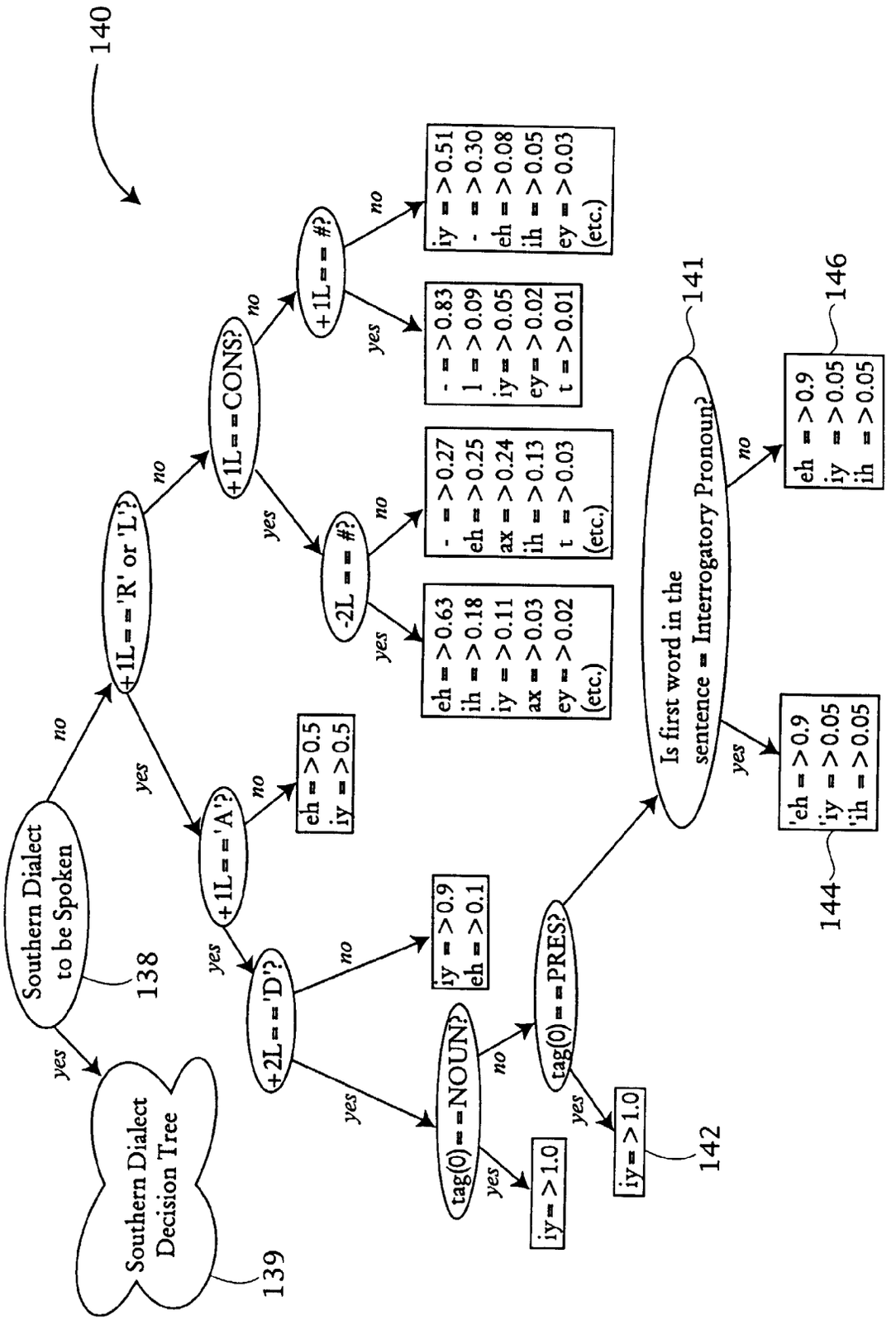
140
Southern Dialect to be Spoken
138
Southern Dialect Decision Tree
139
+1L = = 'R' or 'L'?
+1L = = 'A'?
+2L = = 'D'?
eh = > 0.5
iy = > 0.5
iy = > 0.9
eh = > 0.1
tag(0) = = NOUN?
iy = > 1.0
+1L = = CONS?
-2L = = #?
eh = > 0.63
ih = > 0.18
iy = > 0.11
ax = > 0.03
ey = > 0.02
(etc.)
- = > 0.27
eh = > 0.25
ax = > 0.24
ih = > 0.13
t = > 0.03
(etc.)
+1L = = #?
- = > 0.83
l = > 0.09
iy = > 0.05
ey = > 0.02
t = > 0.01
iy = > 0.51
- = > 0.30
eh = > 0.08
ih = > 0.05
ey = > 0.03
(etc.)
tag(0) = = PRES?
iy = > 1.0
142
Is first word in the sentence = Interrogatory Pronoun?
141
'eh = > 0.9
'iy = > 0.05
'ih = > 0.05
144
eh = > 0.9
iy = > 0.05
ih = > 0.05
146
yes
no

FIG. 9