Viterbi Search Visualization using Vista: A Generic Performance Visualization Tool

Robert Halstead Jr., Benjamin Serridge[*], *Jean-Manuel Van Thong, and William Goldenthal*
email: halstead@crl.dec.com, ben@mama-bear.lcs.mit.edu, jmvt@crl.dec.com, thal@crl.dec.com

Digital Equipment Corporation
Cambridge Research Lab

ABSTRACT

The capability to view detailed events taking place inside of a speech search engine can be essential for reducing the computational complexity and enhancing the accuracy of a speech recognition system. This paper describes a generic search visualization and performance tuning tool, Vista, which permits a user to interactively examine data within the search module of a speech recognition system. Vista's key attributes include the ability to display a large number of active search paths simultaneously, to display data relevant to these paths, and to update the paths and data as the search progresses in time.

1. Introduction

Although conceptually straightforward, the development of a speech recognition search engine is a complex process. Search requires the integration of multiple sources of information and the maintenance of a large number of hypotheses. All of this is done within the framework of a *lattice* which incorporates pronunciation constraints and temporal information.

To effectively tune a search engine, it is important to gain insight into a large number of details that are generally hidden within the system. It can be extremely valuable to the developer to understand the dominant factors behind why a particular path wins or loses, or at what time and for what reasons a particular path (hypothesis) was pruned.

Despite the apparent simplicity of the Viterbi algorithm, it is often extremely difficult to gain insight into what is taking place inside of a speech recognition search engine. This is because the number of parameters and paths that might be examined is potentially huge.

This is especially true for a segment-based search which is inherently more complex than an HMM Viterbi search [1]. In an HMM search, all active paths can be examined by only looking a single step back in time. A segment-based search requires that segment start and end times be explicitly hypothesized. Therefore, the terminal points of active paths tend to exist at many different times in the past.

Therefore, to understand and study the behavior of search algorithms, we need tools that go beyond code debuggers or the depiction of a single path [3]. With a visualization tool, it is possible to determine how multiple active paths are interacting with lattice points residing at the current time. Since search needs to be implemented in a computationally efficient manner, proper visualization can also provide insights into implementing the search efficiently.

This paper describes a search visualization tool, *Vista*, which we utilize to gain insight into multiple aspects of our segment-based search engine. Vista allows the user to examine any active path in the past, and displays all the relevant scores which are used in computing a path's cost. Vista also allows the user to step through the search at any pace desired and actively updates its display after each step. The next section describes Vista's architecture and original implementation as a tool for viewing the execution of parallel programs. Section 3 discusses the adaptation of Vista to the domain of search and Section 4 describes Vista's capabilities in detail. Future Vista extensions are in discussed in Section 5.

2. The Architecture of Vista

Vista is a generic performance visualization tool originally designed for understanding the performance of parallel programs [2]. Many other data-visualization programs built for this purpose are specialized to particular parallel-programming systems, but Vista aims to provide a useful set of *generic* performance analysis and visualization modules, plus an open infrastructure that enables the development of customized performance analysis and visualization tools both by interconnecting existing modules and by adding new modules. In addition to communicating data sets between modules, this infrastructure communicates view boundaries and other interactively changing properties between modules, allowing data exploration from several different but coordinated viewpoints.

[*]Currently a member of MIT's Spoken Language Systems Group

Modules' input and output data sets are represented using SDF, a self-describing format that allows a flexible representation of information. As a result, the information contained in data sets is self-documenting and can be changed easily without modifying Vista. Any desired information can be dumped, including (in our application to search visualization) acoustic-phonetic, duration, and language scores, accumulated path scores, etc.

A crucial and unique part of Vista's architecture is a set of specified conventions for using SDF to represent common types of performance data sets, such as event histories and histograms. These conventions are specified in terms of generic concepts such as "time base" and "binary relationship" rather than specific concepts such as "cycle count," "processor ID," and "send/receive pair." Because they support these generic concepts, Vista's modules are flexible and can be applied in many different situations.

This architecture has proven to be flexible enough that we were able to apply it without modification to the task of visualizing the operation of a Viterbi search engine. For this purpose, only Vista's event-history data type was used. An event history is a collection of *events*. Each event has an *event name* and a set of named *fields*, each of which contains a numerical, character-string, or array value. An event history has one or more *time bases*, which provide ways to arrange events in time, and one or more *threading fields*, which provide ways to group related events together into threads. Events come in three flavors:

- *Transient* events are not considered as changing the state of any thread. The main item of interest about a transient event is the time at which it occurred, along with the auxiliary information recorded in the event's fields.
- *State-changing* events are viewed as changing the state of the thread in which they occur. They are typically displayed as changes in color, with the color change marking the time when the event occurred.
- *Interval* events include both a starting and ending time and are typically displayed as segments of color whose beginning and ending points correspond to the interval's starting and ending times.

Events can also contain arbitrary application-dependent information. Further information about the semantics and representation of Vista's data set types can be found in an earlier paper [2].

For our application, we used Vista's generic event-history viewer. In a parallel-processing context, this viewer displays threads of execution (defined as the sets of events containing the same value in the selected threading field) and interactions between those threads (such as the creation of a child thread by a parent thread). Each thread is displayed as horizontal band of color. Transient events are displayed as vertical lines that cross this band, while state-changing and interval events are displayed as color changes within the band. Binary relationships are thought of as involving a "causer" and a "causee" thread and are displayed as arrows pointing from the "causer" to the "causee."

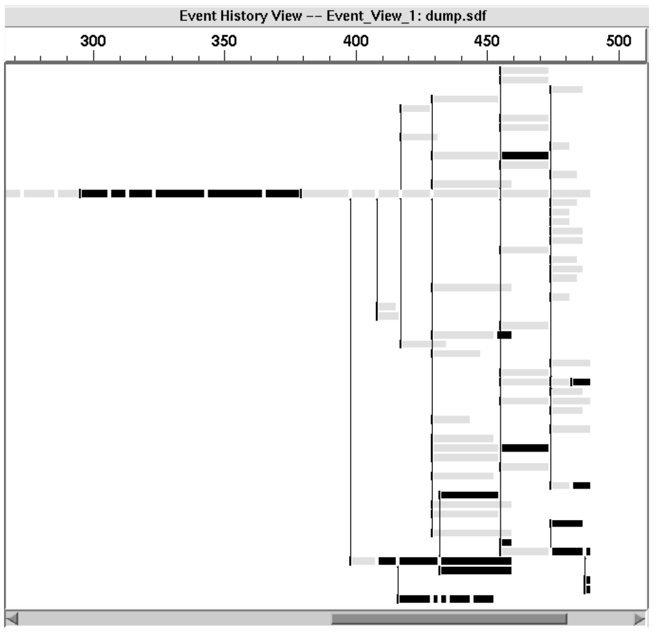
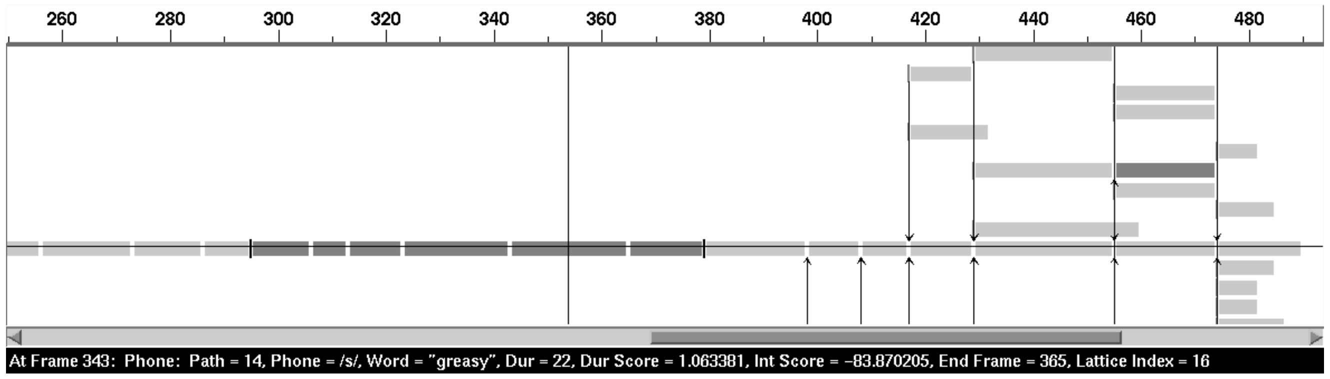


Figure 1: Event history view of all paths active at frame 490. The dark sequences represent full or partially completed word hypotheses. White boundaries (lines) correspond to the transitions between phonetic units. The dark vertical lines denote binary relationships representing the dependence between alternate hypotheses, that is, the point at which *child* hypotheses split off from a *parent* which represents a common past. Note that many hypotheses do not extend to the current time. These hypotheses are still active, but are "hanging" in the past, awaiting future phonetic units to connect back to them. Duration constraints permit a path to be pruned if it is determined that none of its allowed phonetic transitions is still possible.

3. Applying Vista to Search

Viterbi search produces a tree describing all the possible hypothesized sequences of phonetic units (phones). A *path* within the tree describes a hypothesized sequence of phones along with the starting and ending times of each hypothesized phone. Two or more such paths may share a common initial phone subsequence; we think of this situation as a *branch* in which child paths branch from a common ancestor path at the point where their phone sequences diverge. A path may also become *inactive*, in other words, be pruned out from further consideration by the search.

We are using Vista to monitor and debug a keyword spotting



At Frame 343: Phone: Path = 14, Phone = /s/, Word = "greasy", Dur = 22, Dur Score = 1.063381, Int Score = -83.870205, End Frame = 365, Lattice Index = 16

Figure 2: A close-up view of a piece of the search lattice displayed in Figure 1. The cursor has been moved over an instance of the phonetic unit [s]. Information related to this phone is displayed in a text window at the bottom of the figure. Transition information is obtained by moving the cursor to a transition point in the lattice.

application. This application has the capability to stop at pre-specified increments in time and dump a self-describing file that contains information relating to all active paths. The active paths include paths which are "hanging" in the sense that they terminate somewhere in the past. Thus, a *snapshot* is obtained which describes the state of all relevant hypotheses at the current time.

A correspondence between search paths and execution threads allowed us to apply Vista's generic event-history viewer, without modification, for our purposes. In visualizing a search, each active path is mapped to a thread. A branch point in the search then corresponds to creation of new threads. As a time base, we use the frame number in the input speech signal, so the phone boundaries and path branch points will be displayed in terms of their positions in the utterance being analyzed. In this way, the viewer can display the search tree's large-scale structure in a highly intuitive manner: see Figure 1.

Each hypothesized phone along a path is mapped to an interval event giving the beginning and ending times that have been hypothesized for that phone; different colors are used for intervals hypothesized to belong to garbage (filler) words and those hypothesized to belong to a keyword. Transient events represent transitions between phones. The color of a transition indicates whether or not it is at a boundary between words. Finally, binary relationships are used to represent points in time where paths branch. Each phonetic event includes information identifying which phone it is, which word it belongs to, its duration, and relevant scores of interest. Each transition event includes information identifying the left and right phones and the acoustic-phonetic transition score.

Details about a particular event can be accessed by moving the mouse to the corresponding point on the display, which causes the self-describing record for that search node to be displayed textually in a separate window pane as illustrated at the bottom of Figure 2. We have also exploited Vista's linking capabilities to connect it to an external spectrogram viewer, providing a coordinated view of the search behavior and the input speech signal.

4. Vista as a Microscope on Search

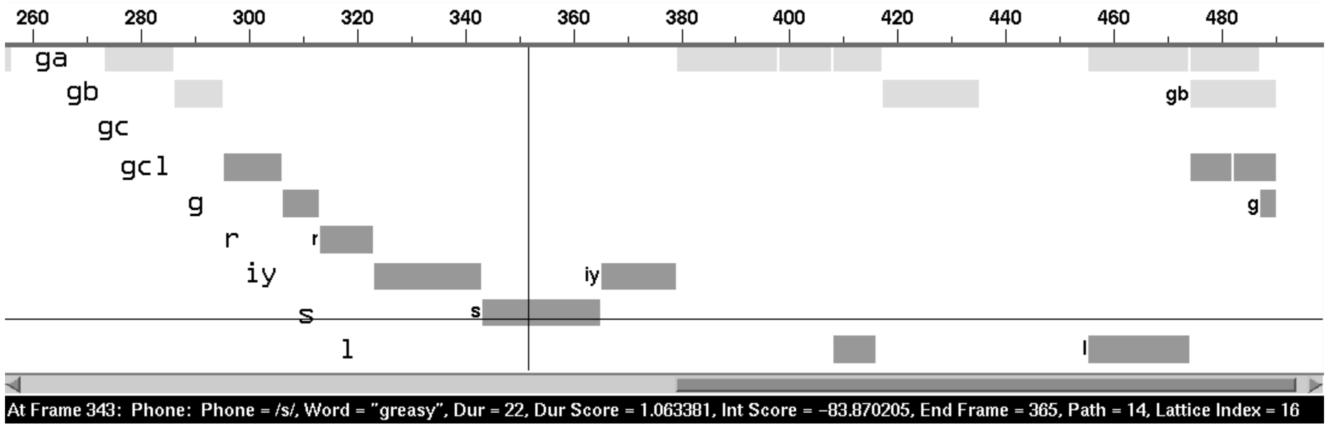
In practice, Vista is utilized as a type of "microscope" for viewing the internal workings of the search engine. Vista's event history viewer is used to view a snapshot of the search lattice hypotheses at any given moment in time. Figure 1 shows a complete view of the search lattice at frame 490. The view covers the period from frame 250 to frame 490 for an example utterance. A single frame corresponds to 5 milliseconds.

Note that all of the active paths extend to frames in the range [415, 490]. This range corresponds to the maximum duration of any phonetic unit in the pronunciation network. Paths which terminate at points prior to this maximum duration cannot possibly be extended, and are therefore pruned out of the search. It is possible to narrow the active window in time, displaying only active paths that are hanging within a given time range. The paths are sorted by accumulated score at the current frame, with the best path appearing at the top.

Vista allows the user to dynamically adjust the view and also to zoom in on any desired part of the lattice. This capability permits the view in Figure 1 to be transformed into the close-up view shown in Figure 2.

Information relative to a phone, a transition or a branch can be displayed interactively by moving the cursor to the corresponding graphical element. A text window at the bottom displays this information to the user.

An example of this can be seen in Figure 2 which shows the



At Frame 343: Phone: Phone = /s/, Word = "greasy", Dur = 22, Dur Score = 1.063381, Int Score = -83.870205, End Frame = 365, Path = 14, Lattice Index = 16

Figure 3: View that results from setting the threading field to the phonetic unit option. Each instance of an individual phonetic unit is shown in a given row. For example, the seventh row shows two instances where the phone [iy] has been hypothesized within the keyword *greasy*. The top rows are reserved for hypothesized garbage units [ga], [gb], and [gc]. The alignment of the units in time is maintained. The gaps in time are occupied by phonetic units which do not appear in this zoomed view. The large-type labels have been added to the figure to indicate the phonetic unit that corresponds to each row.

information displayed when the cursor is moved over a phonetic unit. This information includes the phone name and its lattice index, the word to which it belongs, its duration, time boundaries, and associated acoustic-phonetic scores. When the cursor is moved onto a phonetic transition, the display shows the pair of phones, the transition score, the score increment, and the best score.

Vista permits the same information to be displayed in various different organizations. The threading field can be changed from the path ID to the index within the lattice, the phonetic units, or the words in the lexicon. The information is then displayed such that the hypothesized phones or words overlap in time. It is also possible to change the sort key. In particular, the phones and words can be sorted by best score or by label.

Figure 3 shows another view of the same search lattice and time frame. This view is the result of changing the threading field to the phonetic unit option. Here, each instance of a particular phonetic unit is shown along an individual row. Now the hypothesized sequence of phones in time is not explicitly shown. Instead, alternative overlapping phonetic hypotheses at different points in time can be seen. The units maintain their proper temporal alignments.

Finally, it is possible to utilize Vista to step through the search process in time. This is currently accomplished by dumping the contents of the self-describing file at each point in time when the search is to be examined. The result is that the dynamics of the active paths can be displayed, but the procedure is somewhat clumsy. It would be desirable for Vista to support this mode of operation more smoothly.

5. Conclusions and Future Work

The Vista viewing tool provides a powerful method for examining the internal workings of a search engine. By combining Vista with other tools such as a spectrogram viewer, the developer is able to obtain a comprehensive picture of a speech system. This picture is an invaluable aid in tuning system performance, understanding complex interactions that can occur during search, and locating possible computational inefficiencies.

In the future, we would like to enhance Vista's capabilities by adding the ability to handle incremental updates. This task is currently accomplished by dumping the complete tree description at each step into a file. This enhancement would allow us to interactively track the structure of the search tree as it evolves over time.

6. REFERENCES

1. W. Goldenthal. *Statistical Trajectory Models for Phonetic Recognition.* PhD thesis, Massachusetts Institute of Technology, Department Of Aeronautics and Astronautics, September 1994.
2. R. Halstead. Self-describing files + smart modules = parallel program visualization. *Springer-Verlag Lecture Notes in Computer Science*, 907:253–283, November 1994.
3. Zissman M.A. Seward D.C. Graphical analysis of hidden markov models, speech recognition experiments. *MIT Lincoln Lab Technical Report 1009*, October 1995.