# The *grapHvite* Book

For grapHvite v1.4
*Reference Release*

Kevin Power
Rachel Morton
Caroline Matheson
Dave Ollason

The *grapHvite* Book (for *grapHvite* 1.4 - reference release)



First published April 1997

# Contents


**1 *grapHvite* Overview** **1**
- 1.1 Overview of Syntax Networks . . . 3
  - 1.1.1 Sub-networks . . . 5
- 1.2 The Visual Netbuilder . . . 7
  - 1.2.1 Constructing a network . . . 7
  - 1.2.2 Testing a Network . . . 8
  - 1.2.3 Testing a Recogniser . . . 9
- 1.3 ***grapHvite*** Dictionaries and Models . . . 9
  - 1.3.1 Dictionaries . . . 9
  - 1.3.2 Acoustic Models . . . 10
- 1.4 Creating a Recognition Application . . . 10
- 1.5 What's New in Version 1.4 . . . 12
  - 1.5.1 Features Added to Version 1.1 . . . 12

**2 A Tutorial Example of Using grapHvite** **15**
- 2.1 Defining the e-mail Task . . . 16
- 2.2 Getting started: creating the network . . . 17
- 2.3 Extending the e-mail network . . . 19
  - 2.3.1 Selecting nodes . . . 20
  - 2.3.2 Entering complete sentences . . . 21
  - 2.3.3 Adding the *msg_no* sub-network . . . 22
- 2.4 Completing the e-mail network . . . 23
  - 2.4.1 Sub-network *folders*: adding new words . . . 24
  - 2.4.2 Adding silence nodes . . . 25
- 2.5 Testing the network . . . 25
- 2.6 Testing the recogniser . . . 26
  - 2.6.1 Alternative output symbols . . . 28
  - 2.6.2 Out–of–vocabulary rejection . . . 28
- 2.7 Exporting to a HAPI/JHAPI application . . . 30

**3 Reference Section** **31**
- 3.1 A Tour of the Main Window . . . 32
- 3.2 Some Common ***grapHvite*** Netbuilder Windows . . . 33
  - 3.2.1 The File Browser Window . . . 33
  - 3.2.2 The Warning Window . . . 35
  - 3.2.3 The Information Window . . . 35
- 3.3 Network Operations . . . 36
  - 3.3.1 Creating a Network . . . 36

3.3.2 Loading a Network File . . . 36
3.3.3 Saving Network Files . . . 37
3.3.4 Switching Between Networks . . . 37
3.3.5 Editing Networks on the Canvas . . . 37
3.3.6 Testing and Verifying Syntax Networks . . . 46
3.4 Dictionary Operations . . . 48
3.4.1 Looking a word up . . . 48
3.4.2 Word Pronunciation . . . 49
3.4.3 Alternative Output Symbols . . . 49
3.4.4 User Dictionary Files . . . 50
3.4.5 Creating User Dictionaries . . . 50
3.4.6 Loading User Dictionaries . . . 50
3.4.7 Saving User Dictionaries . . . 50
3.4.8 Editing User Dictionaries . . . 51
3.5 The Graphical Recogniser Interface . . . 52
3.5.1 A Tour of the Main Recogniser Window . . . 53
3.5.2 Configuring the Recogniser . . . 54
3.5.3 Loading the Recogniser . . . 55
3.5.4 Calibrating the Recogniser . . . 56
3.5.5 Using the Recogniser . . . 57
3.5.6 Rejecting Out-of-Vocabulary Speech . . . 58

**A Configuring the *grapHvite* Netbuilder** **61**
A.1 The ***grapHvite*** Netbuilder configuration file . . . 61
A.1.1 General configuration parameters . . . 61
A.1.2 System configuration parameters . . . 63
A.1.3 Example configuration file . . . 64
A.2 Configuring the ***grapHvite*** Netbuilder interactively . . . 66

**B Dictionaries** **67**
B.1 Core and Full System Dictionaries . . . 67
B.2 Dictionary Entry Format . . . 67
B.3 Special Symbols . . . 68
B.3.1 Silence and Short Pause Symbols . . . 68
B.3.2 Single Phones . . . 68
B.3.3 Semantic Markers . . . 68
B.4 English System Dictionaries . . . 69
B.4.1 English Phone Sets . . . 70
B.5 Additional Languages . . . 72
B.5.1 Accented Characters . . . 72
B.5.2 Spanish System Dictionaries . . . 73
B.5.3 Spanish Phone Set . . . 73
B.5.4 German System Dictionaries . . . 74
B.5.5 German Phone Set . . . 74

**C Standard Lattice Format - SLF** **75**

# Chapter 1

# *grapHvite* Overview

***grapHvite*** is a developer kit for building small to medium vocabulary speech recognition systems. It provides the means for creating the recogniser components for a given speech recognition task and also for incorporating a recogniser into a target application. How ***grapHvite*** achieves this is best understood in terms of the assumed structure of a speech recogniser illustrated in figure 1.1.

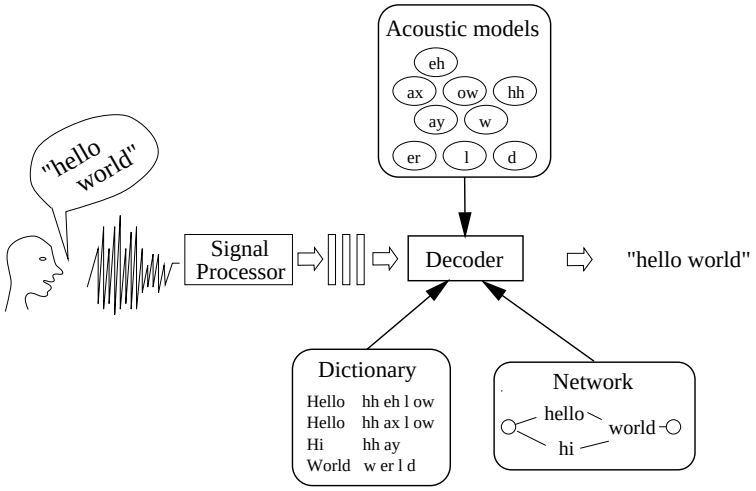


**Fig. 1.1 Speech recogniser components**

As can be seen the central element of a speech recogniser is the decoder. This transcribes continuous speech input into a textual symbol sequence which an application can directly process. It requires a syntax network to define the allowed word sequences appropriate for the recognition task, a pronunciation dictionary specifying an acoustic model sequence for each word in the task vocabulary and a set of acoustic models which model the individual speech sounds. ***grapHvite*** includes a decoder and a set of pre-trained acoustic models. To create a new recogniser it is only necessary to create a network (to define the recognition task) and its associated dictionary. These task-specific components are created by means of an easy-to-use

visual tool, the ***grapHvite*** Netbuilder.

The complete ***grapHvite*** package consists of the following items as illustrated in figure 1.2.

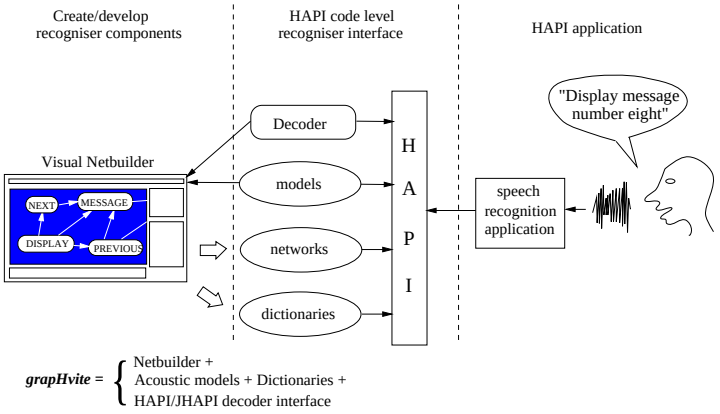


**Fig. 1.2 The *grapHvite* package**

- **Decoder** A speaker independent, continuous speech decoder, intended for small to medium vocabulary recognition systems.

- **HAPI/JHAPI decoder interface** The HTK[1] Application Programming Interface (HAPI) provides the interface to the ***grapHvite*** decoder and is available as a C library (HAPI) or alternatively as a Java library (JHAPI) which is convenient for developing portable visual applications. The HAPI/JHAPI interface provides a high degree of control over decoder operation and also allows for online processing of dictionaries and networks. The complete HAPI/JHAPI interface is discussed in depth in the accompanying documentation, *The HAPI Book*.

- **Acoustic models** Phone-level speech acoustic hidden Markov models (HMMs) are provided for each supported language. The model sets are trained on speech recorded from a wide range of speakers over a close-talking microphone in low-noise conditions.

- **Pronunciation dictionaries** There is a general purpose pronunciation dictionary for each supported language (these are described separately in Appendix B). For English language systems the dictionary covers approximately 90,000 words.

[1] Entropic's Hidden Markov model Toolkit (HTK) is a comprehensive set of tools for training and evaluating hidden Markov models (HMMs) used for modelling speech.

- **The Netbuilder** A visual tool for rapidly creating a syntax network and dictionary for a task-specific recogniser. The Netbuilder provides tools for testing that a network, once created, is syntactically correct and also for interactively testing a network with the ***grapHvite*** decoder. In addition, the Netbuilder allows pronunciations to be edited or new pronunciations to be created for words not covered by any of the supplied dictionaries.

In addition, an example speech driven e-mail application is included. This is implemented in Java and illustrates the typical usage of the JHAPI decoder interface.

The rest of *The **grapHvite** Book* is organised as follows. The remainder of this chapter provides an overview of the ***grapHvite*** development kit. The concepts of syntax networks are briefly reviewed and the main steps in building a task-specific recogniser are then outlined. Chapter 2 gives a detailed tutorial example of how to create the components needed for a task-specific recogniser using the Netbuilder. The process of using the HAPI/JHAPI interface to a recogniser within an application is described in detail in *The HAPI Book* included with the ***grapHvite*** release. Finally, chapter 3 is a reference section which documents the functionality provided by the ***grapHvite*** visual Netbuilder.

## 1.1 Overview of Syntax Networks

As noted earlier, the ***grapHvite*** decoder requires a syntax network to define the allowed word sequences to be recognised. In principle the recogniser could just take a list of the task-specific words and match them in any order to the input speech. However, in practice such an unconstrained system leads to sub-optimal recognition performance. If the recogniser is constrained to consider only those word sequences relevant to the recognition task the recognition performance can be optimised for that task. Syntax networks are the means used by the ***grapHvite*** decoder of applying task-specific constraints.

An example of a simple (but useful) recognition task is recognition of 'yes'/'no' as part of a confirmation dialog. For example the following input utterances might be allowed.

yes
no
yes please
no thank you
no thanks

This set of sentences can be represented by a network as shown in figure 1.3. A network consists of a set of nodes with interconnecting links between nodes. A network must have one (and only one) start node and one (and only one) end node. The minimum sized network consists of a start node linked to an end node. Paths through the network are traversed by following the directed links from the start node through to the end node.


**Fig. 1.3 Extended yes/no network**

Three types of network node are supported by ***grapHvite***:

- **Word nodes** A word node represents a word at a point in the network. Tracing a path from the start node to the end node defines a sentence with the word nodes along that path corresponding to the words in the sentence.
- **Null nodes** A null node is a dummy node that does not in itself represent anything. They are often used to collect and propagate links where a pattern of multiple links converging to or diverging from a node is repeated. In figure 1.3 the start and end nodes both happen to be null nodes.
- **Sub-network nodes** An entire portion of a network can be represented by a single sub-network node. This is a very powerful feature and is discussed in section 1.1.1.

Each network path in figure 1.3 corresponds to one of the above specified input sentences. In this example it is feasible to trace each path by hand and verify that *only* the desired sentences can be generated by the network. For more complex networks, various tools are needed to automatically check the syntactic correctness of a network for a given task - these are discussed later in section 1.2.

As an example, consider an application where inputs such as the following are allowed.

Can I speak to Kevin
Could I speak to Rachel
Could I please speak to Kevin
Can I speak to Kevin please
Please can I speak to Rachel
etc

For simplicity, only the prefixes 'can I' and 'could I' are included and only the two names Kevin and Rachel occur. A network that represents these inputs is shown in figure 1.4. This network uses null nodes to collate links as well as to denote the network start and end nodes. By feeding links into the word nodes 'can' and 'could' via a null node instead of directly from the start and initial please nodes, the network structure is slightly simplified.

Although not mandatory using null nodes to reduce the number of network links improves recogniser efficiency.

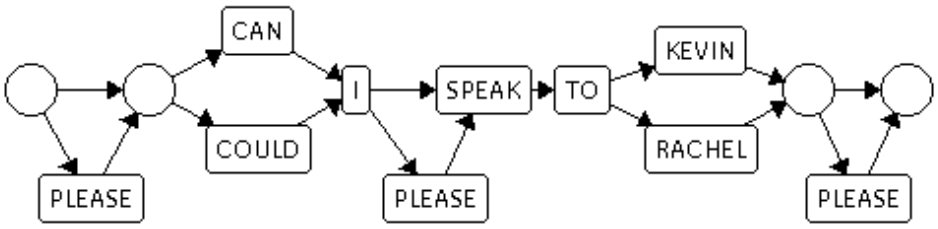


**Fig. 1.4 Example application network**

The network in figure 1.4 generates sentences with the word 'please' appearing up to three times. If it is known that 'please' can occur at most once in a sentence then this constraint should be built into the system for best results. An augmented network incorporating this constraint is shown in figure 1.5. Note that this gives rise to a more complex network. This is unavoidable and leads to maximum recognition accuracy (and potentially efficiency) for a given task.

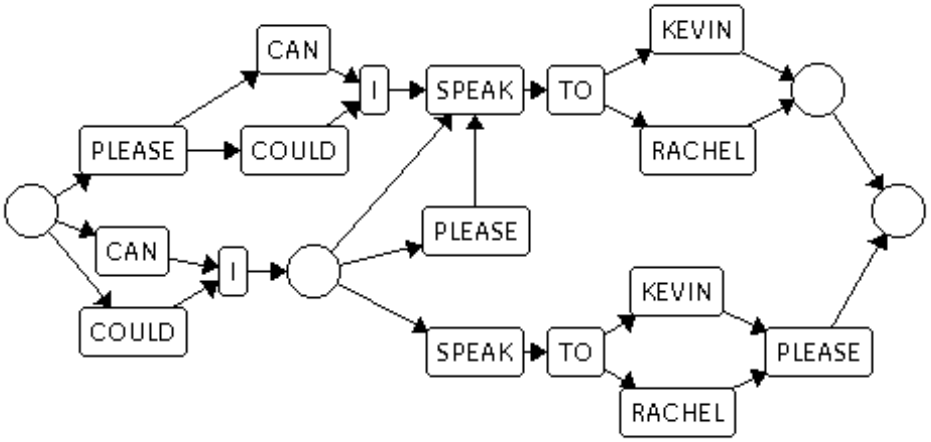


**Fig. 1.5 Refining the network**

### 1.1.1 Sub-networks

In the network shown in figure 1.5 the names portion (i.e. KEVIN, RACHEL nodes) occurs more than once. It may also be common to other networks used in the same or similar applications. This 'names' fragment of the network can be abstracted as a separate sub-network – called *names* – and referred to by a sub-network node in the

main network. The names sub-network is shown in figure 1.6 and the main network with sub-network nodes are shown in figure 1.7.

Sub-networks follow the same rules as networks. A network with sub-network nodes is syntactically equivalent to the same network with sub-networks expanded in-line. Judicious use of sub-networks helps to keep the structure of the main network clear and can also greatly increase the speed of network development as reusable sub-networks become available.

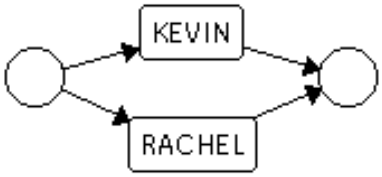


**Fig. 1.6 Names sub-network**

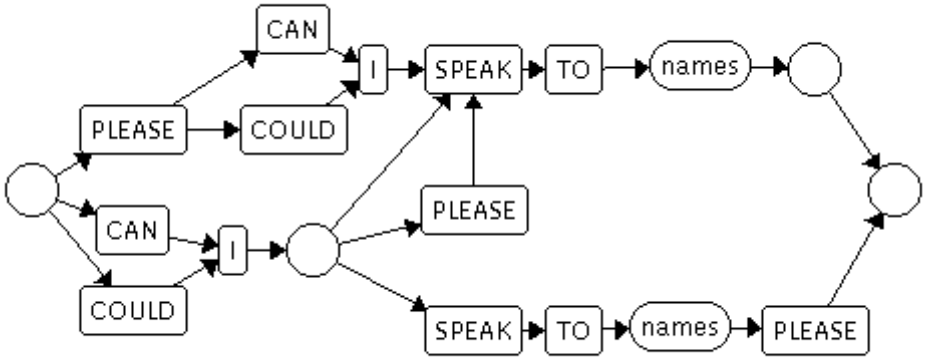


**Fig. 1.7 Using sub-network nodes**

Sub-networks can be useful for applications where parts of a recognition task change regularly. For example if an application refers to a list of names - similar to the task network in figure 1.7 - which change on a regular basis, then only the appropriate sub-network needs to be updated. This could be done manually using the Netbuilder, and the new network loaded by the application when required. However, the HAPI/JHAPI interface supports online processing of networks and the application could create the updated names sub-network (e.g. from a text file) and update the appropriate sub-network nodes of the dependent networks accordingly.

## 1.2 The Visual Netbuilder

The visual Netbuilder is a graphical tool for creating and testing the task-specific components, i.e. the syntax network and dictionary, needed for a speech recogniser. It includes tools for testing whether a network is syntactically correct for a task and also provides an interface to the ***grapHvite*** decoder, allowing interactive testing of a task recogniser. The following sections outline the process of creating a network and dictionary within the Netbuilder.

### 1.2.1 Constructing a network

An example of the main Netbuilder window is shown in figure 1.8. As can be seen the main portion of the window is a scrollable canvas area on which networks are created and developed. Networks are created on the canvas using various mouse operations in a similar manner to many graphical drawing packages, except the functionality is specific to syntax networks. At the bottom right of the Netbuilder window is a panel of buttons which control the mode of the left mouse button. For example left-clicking Insert Null Node button sets the left mouse button action so that left clicking on the canvas causes a null node to appear.

Word nodes are created in a similar manner by first selecting Insert Word Node. However, for a word node to appear on the canvas, the corresponding word must be selected in the dictionary panel to the right of the canvas. First, type the word into the dictionary search panel. A progressive match is then performed and matching entries are listed. Once one of these is selected the corresponding word node can be created by left-clicking the mouse on the canvas. New words are automatically added to a list of words used in the network displayed to the right of the canvas. This lists the word entries needed to form the task dictionary.

Links between nodes are created by first selecting Join Nodes and left-clicking and dragging the mouse from one node to another. Further functionality is available from a pop-up menu activated by right-clicking the mouse anywhere on the canvas. This pop-up menu is context sensitive, i.e. the available options vary depending on what item is currently underneath the mouse pointer. Sub-network nodes for example are created by selecting an option from the pop-up menu.

Finally, multiple nodes can be selected and various standard operations such as cutting and pasting applied to the selection. These operations on node selections can be found in the **Edit** menu. The detailed mechanics of network construction and editing are described in further detail in the tutorial and reference chapters.

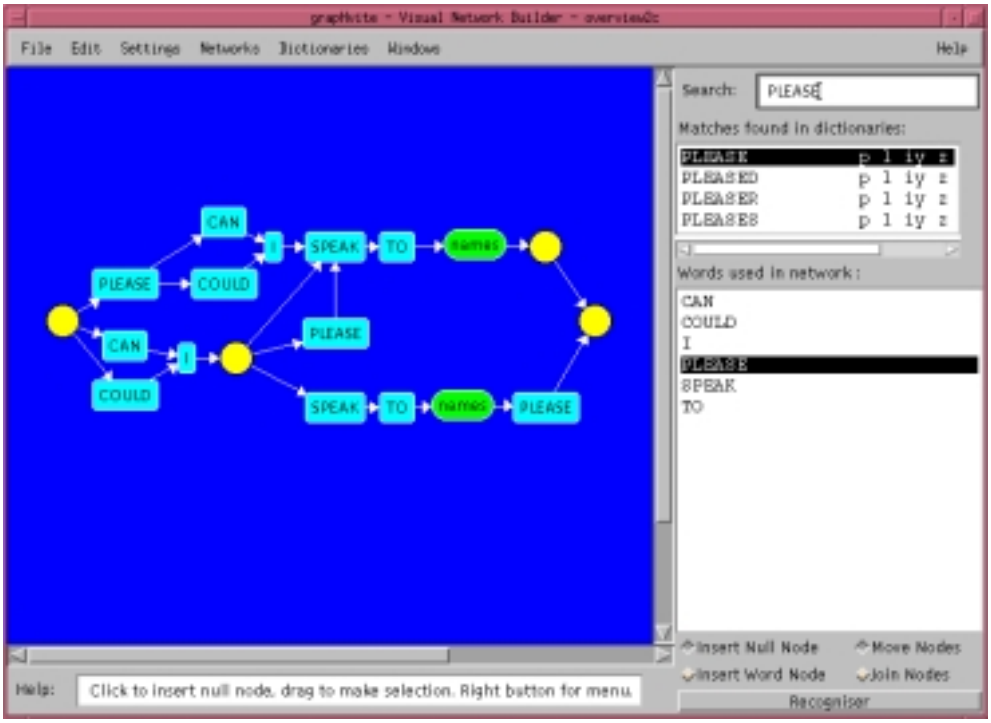


**Fig. 1.8 The main Netbuilder window**

### 1.2.2 Testing a Network

Once a network has been constructed, various checks can be performed within the Netbuilder to ensure the network is valid and that it accurately reflects the task domain. Several tools for testing networks can be invoked from the **Networks** menu in the main window. One such tool performs verification of a network and reports, in a separate dialog window, any anomalies such as unconnected nodes or more than one start or end node. This is very useful for ensuring network integrity before exporting the network to a HAPI/JHAPI application or interactively testing the recognition performance using the Netbuilder recogniser interface.

Another option from the **Networks** menu provides access to a set of tools for testing the syntactic correctness of a network. A separate dialog window for sentence testing allows a sentence to be entered into a text field and reports if the sentence can be parsed by the network. This is a convenient way to confirm that valid sentences from the task domain parse successfully and invalid sentences fail to parse. If valid sentences cannot be parsed the network needs to be extended to include these and if invalid sentences are successfully parsed this indicates that the network "over-generates" sentences for the task, for example the network in figure 1.4 generates the (potentially unwanted) sentence:

Can I please speak to Kevin please.

Further network development is needed to eliminate such behaviour in order to ensure optimal recognition performance for a particular task.

### 1.2.3 Testing a Recogniser

The Netbuilder provides a visual interface to the ***grapHvite*** decoder within which the recogniser corresponding to a syntax network can be interactively tested. Clicking the Recogniser button at the bottom right of the Netbuilder window invokes this recogniser interface. This is a useful tool for identifying weaknesses in the design of a task network, thus allowing the network to be quickly refined so that it can be used in a real system.

An optional configuration stage allows selection between the acoustic models supplied with ***grapHvite*** as well as a choice of audio source between microphone input, line input and file input. For live speech input the recogniser uses a silence detector to automatically detect when a speaker has finished speaking a sentence. The background silence levels need to be estimated for the silence detector during a separate calibration stage.

Once the recogniser has been set up the process of recognising speech input from the selected audio source is started by the click of a mouse button. As the speech is processed the most likely partial recognition hypothesis (i.e. path through the syntax network) to date is continually displayed. When all the speech has been processed the most likely recognition hypothesis is displayed. Once the results are displayed for a spoken sentence the recogniser is ready to process the next spoken input.

For simple tasks the recogniser should be correct most of the time. Sometimes, however, the recogniser consistently misrecognises certain inputs. Sometimes this is not detrimental to the task if for example 'yes' is mis-recognised as 'yes-please'. Other cases may be more critical to task performance and some fine tuning of the task network may be required to optimise the performance of the recogniser such that recognition gives a result that can be clearly interpreted by the task application. For example, there may be confusable words (which convey the key semantics for the recognised phrase) competing in parallel branches of the network. It may be possible to either slightly restructure the network or change word nodes with confusable words associated.

As well as refining the network structure to minimise potential misrecognitions, the word pronunciations can also be examined at this stage and new ones added if necessary as discussed in section 1.3.1.

## 1.3 *grapHvite* Dictionaries and Models

### 1.3.1 Dictionaries

***grapHvite*** supplies comprehensive dictionaries for each of the supported languages. Each dictionary word entry can have one or more associated pronunciations. A pronunciation specifies a sequence of the fundamental speech sounds or phones for a word, from which a corresponding sequence of acoustic models is determined for recognition.

At startup the Netbuilder loads a system dictionary for one of the supported languages, depending on how it is configured. The full system dictionary may take a while to load and a smaller 'core' dictionary of the more common words can be optionally loaded instead. In addition, users can create and load their own user dictionaries of application-specific words. The dictionary search facility applies the search to all loaded dictionaries. In the case where a word appears in more than one

dictionary, the search produces a single entry for that word with all pronunciations found listed under that entry.

As noted earlier, new words are automatically added to a list of used words in the Netbuilder dictionary panel as the corresponding word nodes are created. An option under the **File** menu allows the used words list to be exported as the task dictionary for use in a HAPI/JHAPI application. The associated pronunciations of entries in a user dictionary can be altered if needed, for example, if the existing pronunciation appears to compromise recogniser accuracy. Alternatively, new pronunciations can be added for words in the task vocabulary.

Of course, some task specific words will not have entries in the system dictionary. If this happens then a new user dictionary can be created or new words added to an existing user dictionary using options from the **Dictionaries** menu. To add new words and pronunciations, the simplest approach is to search for similar sounding words in the system dictionary and derive a pronunciation by comparison with the examples found. Examples of the corresponding sounds for each of the phones covered are given in Appendix B of the reference section.

### 1.3.2 Acoustic Models

***grapHvite*** provides a set of acoustic models for each supported language. These are trained on a large corpus of speech recorded from a close-talking microphone in a low-noise environment, collected from many speakers. These model sets give best results if the conditions during recognition match the training conditions as closely as possible.

The sets of models represent speech at the phone level (i.e. they are sub-word models). One set typically consists of context independent phones (monophones) which model the individual phones regardless of phonetic context. The other model set consists of context dependent phones (triphones) which take into account both left and right phonetic contexts in which a given phone occurs. Context dependent models are more accurate than context independent models but require substantially more memory. Any of the available model set can be selected from the recognition window when testing the recogniser.

It is worth noting that significant differences in performance may only become apparent with large test sets covering many different speakers.

The ***grapHvite*** models are used directly by the Netbuilder tool when testing the recogniser and can be used directly in a HAPI/JHAPI application.

## 1.4 Creating a Recognition Application

Once the task network and task dictionary have been created within the Netbuilder, all the elements needed to create a recognition application are in place. Applications can be implemented in either C or Java by interfacing to the HAPI or JHAPI libraries respectively. The HAPI/JHAPI interfaces provide equivalent functionality, and for convenience only the HAPI interface is discussed here.

The HAPI interface represents each of the constituent parts of a recogniser as software 'objects' which are created and accessed by an application using a set of function calls. To internally create and run a recogniser, an application must first create each of the following HAPI objects. The discussion here is very brief to give an

indication of the process involved. For an in-depth description and tutorial example of using the HAPI recogniser interface the reader is referred to the accompanying documentation, The HAPI Book.

- **Source object** Connects to a live audio source or reads audio from file. A single source object can be used throughout a session. For live audio, silence detection can optionally be enabled to signal when to start or stop recognising. The silence detector is calibrated for a session via the source object.

- **Coder object** This is created with a source object. The coder object extracts relevant features from the input speech waveform and compactly encodes these into 'frames' used for recognition.

- **HMMSet object** The HMMSet object represents the speech acoustic hidden Markov models which are matched to the incoming speech during recognition.

- **Dictionary object** This object corresponds to a task dictionary. Normally this object is used only by other HAPI objects, however, it also has a comprehensive interface for adding, editing or deleting entries directly by the application.

- **Network object** The network object corresponds to a task network and references the HMMSet and dictionary objects. For many applications the network is one that has been exported from the Netbuilder and no operations need to be directly applied to it by the application. However, HAPI also supports online network creation and processing. This is done using a HAPI *lattice* object which can be subsequently converted into a network object.

- **Recogniser object** This provides the functionality for recognition. It requires a HMMSet object and a network object for its creation. The application can have a high degree of control over the recognition process by selecting 'frame-by-frame' mode. The application specifies the number of frames for the recogniser object to process at a time. Intermediate results are accessible (via the results object) after each batch of frames is processed.

- **Results object** The results object provides access to the recognition output at different levels e.g. at the word or phone level. The word sequence along with acoustic likelihood scores and timing information can also be retrieved. Multiple recognition hypotheses can be returned from this object in the form of an N-best lattice. The returned lattice can be further processed and converted into a network object for use in a related recognition task.

For many applications the HMMSet, dictionary and network objects need not be directly accessed during recognition. Before commencing recognition the source and coder objects must be activated. The application can then request the recogniser object to process incoming speech frames. Finally, the results object can be queried for hypotheses as frames are being processed, as well as when recognition is complete. In frame-by-frame mode the application can halt the recognition process at any time, and with silence detection enabled the recogniser object can automatically detect the end of speech, at which point the processing of subsequent frames is halted.

## 1.5 What's New in Version 1.4

Version 1.4 is the final reference release of ***grapHvite***. It ships with HAPI version 1.4, which is the reference release version of HAPI. The following changes have been introduced since the previous external release (version 1.1).

1. An updated version of (J)HAPI is supplied - reference release version 1.4. This contains many fixes and enhancements to the HAPI implementation. None of these should affect the programming interface itself *i.e.* this version of (J)HAPI is backwards compatable with version 1.2, shipped with the previous ***grapHvite*** release. See the overview section of The HAPI Book for details of the HAPI versioning history.

2. The license manager has been de-activated for all libraries and tools distributed with ***grapHvite***. The Netbuilder and any (J)HAPI applications no longer perform dynamic license checking at runtime. Use of this software is still subject to the terms and conditions of the license agreement which can be found in the file LICENSE.TXT, which is located at or near the top-level of the distribution media.

3. The HAPI tutorial now comes with **HAPIVite** - a command line tool which performs recognition on either file-based or interactive speech input. This is useful for offline analysis of a system as well as providing additional examples of HAPI usage. See the "README" file in the HAPI tutorial directory for documentaion on using this tool.

4. Various fixes and minor refinements have been incorporated into the ***grapHvite*** Netbuilder tool since the last release.

### 1.5.1 Features Added to Version 1.1

As a convenient reference for users of earlier versions of ***grapHvite*** this section lists the changes to the ***grapHvite*** Netbuilder since version 1.0.

1. Penalties can now be assigned to network links within the ***grapHvite*** Netbuilder, taking effect both within the recognition window and when the network is exported to a HAPI application.

2. A facility for quickly inserting complete sentences into a network has been added. Selecting Add Sentences... from the **Networks** menu brings up a dialog in which a partial or complete sentence can be inserted into a network simply by typing it into a text field. Sentences can also be inserted from a text file.

3. Out-of-vocabulary rejection using the HAPI word-level confidence scores been incorporated. Selecting Settings... from the new **Rejection** menu in the recogniser window displays a simple dialog which allows the user to set a rejection threshold below which words are rejected when their confidence score falls below the threshold. Rejected words are highlighted in the recognition output window.

4. Support for using a *garbage model* for out-of-vocabulary rejection is now provided.

5. The test sentences dialog (select Test Sentences... from the **Networks** menu) has been updated to allow generation of a specified number of sentences and to allow these to be written to a file. The test sentence facility now correctly parses sub-networks.

6. When entering a sentence in the test sentences dialog, word nodes are skipped if their associated word is enclosed in angle brackets. This allows standard silence markers (<sil>, </sil>, etc) to be ignored and is also useful for skipping user-defined semantic markers.

7. Foreign language support for accented characters has been added. To enter an accented character into a text field (e.g. the dictionary search field) type the character sequence <META>–<accent> <character> e.g. to enter the character ñ, type <META> and ˜ together followed by the character 'n'.

8. Dictionary entries can optionally be mixed case. This is enforced by setting a configuration parameter.

9. The "Configure Recogniser" dialog in the recogniser window now features a load button to activate initial loading of the recogniser following configuration. A default setting can be specified for each item in this dialog via the configuration so that this dialog can be bypassed for a frequently used set-up.

10. Nodes can be inserted directly into network links.

11. The ***grapHvite*** Netbuilder now allows command line arguments to specify networks (-n) and dictionaries (-d) to load at startup.

12. Configuring the ***grapHvite*** Netbuilder now follows a more flexible scheme which allows new systems to be easily added to ***grapHvite***, or for new resources such as models and dictionaries to be easily added to an existing ***grapHvite*** system. In addition, some new ***grapHvite*** Netbuilder configuration parameters have been introduced and the configuration dialog updated.

13. The saving out of HAPI format networks has been made more robust.

14. The ***grapHvite*** Netbuilder and e-mail demo have been updated to run under Java version 1.1. This has caused some minor differences in the appearance of the ***grapHvite*** Netbuilder GUI when run under Unix. As a result the pop-up menu now positions correctly when the main ***grapHvite*** Netbuilder window is resized.

# Chapter 2

# A Tutorial Example of Using grapHvite

This tutorial section describes the construction of a recogniser for use in a voice driven e-mail application. The example recognition task is that of browsing a visual display of e-mail messages as illustrated in figure 2.1.

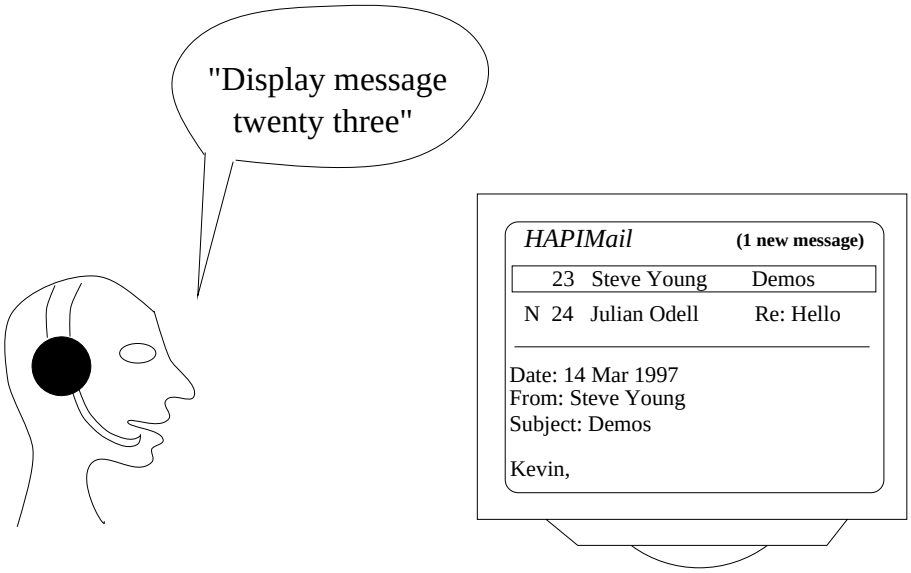


**Fig. 2.1 Example e-mail browsing task.**

Some typical inputs to the e-mail browser might be

Display message number three
Select message two
Show new messages
Display previous message
Reply to last message

Forward message number three nine four
*etc*

This tutorial focuses on constructing the e-mail browser recogniser from scratch using the ***grapHvite*** Netbuilder. Building the e-mail application itself is not discussed. Instead, a full source Java example application is supplied with ***grapHvite*** which illustrates typical use of the HAPI/JHAPI interface. *The HAPI Book*, which accompanies this documentation, gives a comprehensive tutorial on how to incorporate a task recogniser into an application.

## 2.1 Defining the e-mail Task

A recognition task is completely defined in ***grapHvite*** by a syntax network. For many non-trivial tasks the structure of the syntax network may not be immediately obvious and some design is usually required to produce a good system. A good starting point in the design process is to specify the actions to be carried out by the task. The e-mail browser actions might be broadly grouped into

- Listing new messages
- Jumping to a specified message
- Performing an operation on a specified message (e.g. forward, delete, etc.)
- General command (e.g. help, main menu, quit)

Some typical spoken inputs can then be listed for each of these broad groups of task actions. For example, possible spoken inputs to list new messages might be:

New messages
Show new messages
List any new messages
Display all new messages
*etc*

It is usually not practicable to list more than a few such inputs for each action. This is sufficient to get a feel for the task and to suggest a structure for the task network. Also, if these inputs are typed into a text file they can be later used to test the syntactic correctness of the network within the Netbuilder. Continuing this process for the other action groups suggests a possible e-mail network, shown in figure 2.2. As can be seen this includes two sub-networks (the oval-shaped boxes in the diagram): *msg_spec* which defines how the user can specify a message, and *folders* which defines a list of the application e-mail folders. A similar process can be used to arrive at a structure for these sub-networks and examples of these sub-networks are given later.

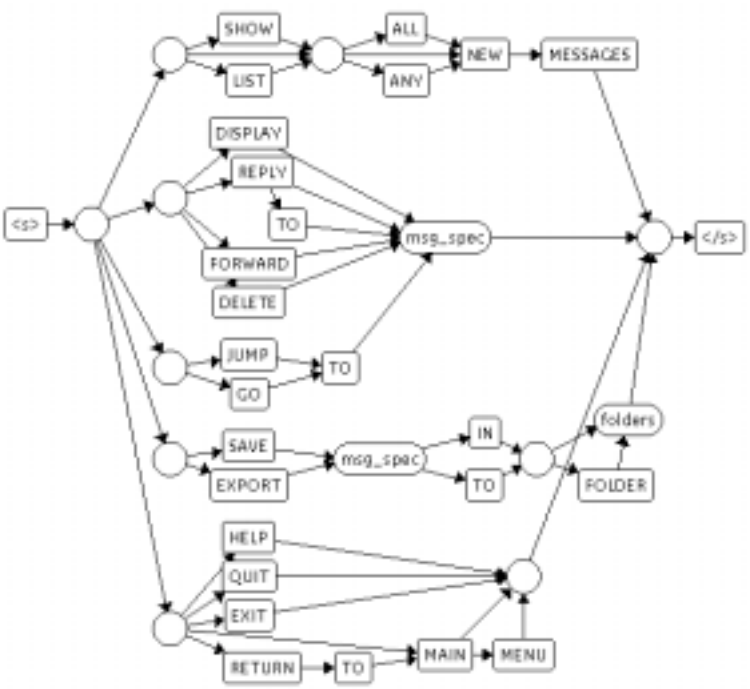


**Fig. 2.2 E-mail browser task network**

A major goal when building a task recogniser is to maximise the recognition accuracy on the given task. It is important to bear in mind that the recogniser outputs the most likely path found through the task network given the acoustic models, the dictionary and the input speech. If a spoken input such as

Display messages

has no path through the network, then the system might 'recognise' this as "DELETE MESSAGE". By accommodating as many likely inputs as possible, the system robustness will be increased. It is also important that the network does not "over generate" i.e. represent inputs that are very unlikely to occur. Referring to the top branch of the e-mail network in figure 2.2 we see that the input "ALL NEW MESSAGES" is represented. If we know this is a redundant input, that part of the network can be restructured slightly to eliminate this possibility. This avoids a valid input being misrecognised as this unlikely input, thus improving the potential efficiency of the system. Finally, defining a network based on what a user is likely to say (from experience of using the system) rather than what the developer thinks the user should say, can lead to a more natural speech interface. This is an important goal in the design of any system.

## 2.2 Getting started: creating the network

The e-mail network can now be constructed within the Netbuilder. Initially, the construction of the top portion of the e-mail network shown in figure 2.3 will be described.

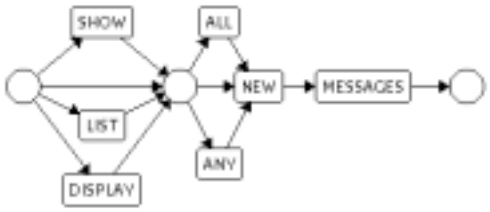


**Fig. 2.3 Fragment of e-mail browser network**

On invocation the main Netbuilder window appears, similar to that shown in figure 2.4. The main canvas area is where networks are constructed and this will initially be blank. Our goal is to construct the desired network fragment so that the main Netbuilder window shows a display similar to that shown in the figure.

To begin creating a new network select New Network from the **File** menu. Nodes and links between nodes can now be created on the canvas.

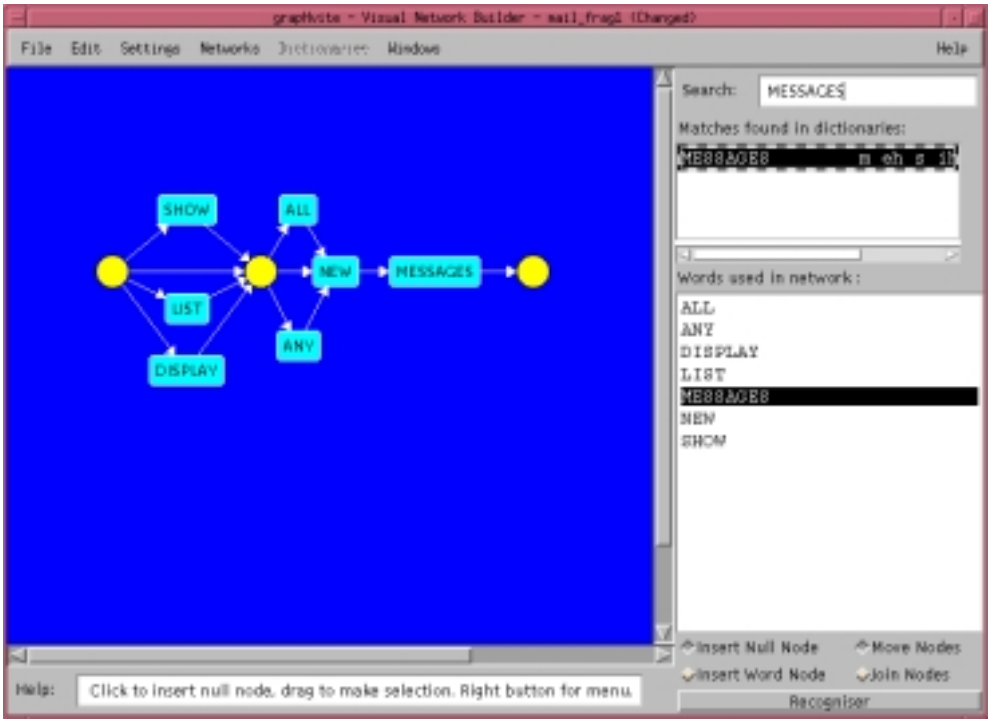


**Fig. 2.4 Typical Netbuilder display after creating the network fragment**

To 'create' the leftmost null node, first move the mouse pointer to the desired location on the canvas. Clicking the left mouse button causes a null node to appear. Repeatedly clicking the left mouse button at different canvas locations creates new

null nodes. To delete an unwanted node, first position the mouse over it (this highlights the node) and then click the right mouse button. This displays the *pop-up menu* from which Delete Node can be selected. This right mouse button activated pop-up menu is context sensitive and will be used extensively for network editing functions.

Word nodes are created in a similar fashion, except the associated word must first be selected in the dictionary panel to the right. Typing "SHOW" in the dictionary search field lists any matching word entries, and if an exact match is found it is highlighted as the currently selected dictionary entry. Clicking on any other item in the list of matching entries causes it to be selected instead. With a dictionary entry selected, a corresponding word node can now be created. To do this, select Word Node from the pop-up menu (activated by clicking the right mouse button on the canvas) and then left click the mouse on the canvas. A word node representing "SHOW" now appears. Notice how the word is also added to the used words list to the right of the canvas. Repeatedly clicking the left mouse button at different locations on the canvas causes more word nodes representing "SHOW" to appear in each location. Again, an unwanted word node is easily deleted by highlighting it and selecting Delete Node from the pop-up menu.

To link nodes, for example, the leftmost null node to the word node "SHOW", highlight the null node, depress the middle mouse button and drag the mouse away from the node. A link appears from the initial node tracking the mouse pointer on the canvas. If the mouse button is released, this link disappears. When word "SHOW" is reached the link remains when the mouse button is released[1]. Moving the mouse pointer over a link highlights it, in which case Delete Link can be selected from the pop-up menu to delete the link.

To move a node, first highlight it, and holding down the left mouse button move the mouse pointer along the canvas. The node follows the mouse pointer along the canvas. Any links to or from the node are continually redrawn to track the node. These basic operations to create null nodes, word nodes and links can be continued to create the rest of the network fragment as displayed earlier.

Before progressing further it is a good idea to save the work to date (select Save Network from the File menu). It is also instructive to observe how the pop-up menu options change depending on whether the mouse pointer is directly over a node, a link or the canvas. The check-boxes above the Recogniser button in figure 2.4 provide alternative easy-to-use controls for some pop-up menu functions. It may be worth experimenting to discover which style of control is preferable.

## 2.3 Extending the e-mail network

We are now in a position to extend the partial network further as shown in figure 2.5. A section of this new network fragment is enclosed in a rectangular box so it can be more easily referred to later on. As can be seen the extended portion of the network

[1] For a two-button mouse links can be created with the left mouse button (using the check-boxes above the Recogniser button in figure 2.4). The details of how these check-box options control mouse behaviour is described in detail in chapter 3.

has a similar structure as before except for the oval-shaped sub-network node *msg_no*, which refers to a sub-network defining message numbers consisting of sequences of up to three digits (see figure 2.7).

If starting a new Netbuilder session select Load Network... to restore the work to date.

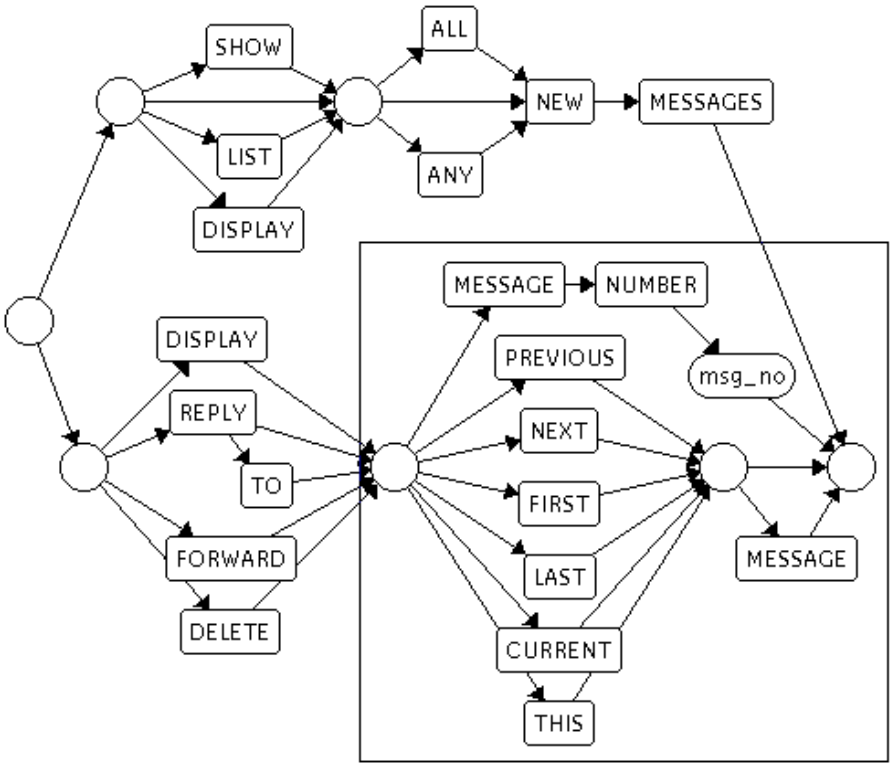


**Fig. 2.5 Extended e-mail network fragment**

Although the entire network can be constructed using the basic operations described earlier, this becomes laborious for large networks. The next section briefly describes some techniques to speed up the development process before discussing the incorporation of the *msg_no* sub-network.

### 2.3.1 Selecting nodes

The Netbuilder provides operations which can be performed on multiple nodes in a similar manner to many graphical drawing packages. Multiple nodes can be *selected* by left clicking them individually. Alternatively, positioning the mouse pointer on the canvas and dragging it across the canvas with the left button held down causes a rectangular box to appear. When the mouse button is released any nodes within this box are automatically selected. This is indicated by the nodes appearing in magenta.

Try selecting some nodes (e.g. word nodes "NEW" and "MESSAGES") and then linking any of the selected nodes to some other unselected node (e.g. "DISPLAY").

This automatically links *all* selected nodes to the target node. Selecting Undo from the Edit menu removes these links. Similarly, linking to a selection creates links to each selected node.

As well as rapid link creation, all selected nodes move in unison when any selected node is moved. Under the Edit menu several options such as Cut Selection, Copy Selection and Paste Clipboard enable rapid editing and duplication of parts of the network. These and other facilities are described in depth in the reference section.

### 2.3.2 Entering complete sentences

To speed up the entry of word nodes onto the canvas the "Add Sentence" dialog can be used to insert partial or complete sentences into a network. Selecting Add Sentences... from the **Networks** menu displays a dialog similar to that shown in figure 2.6.

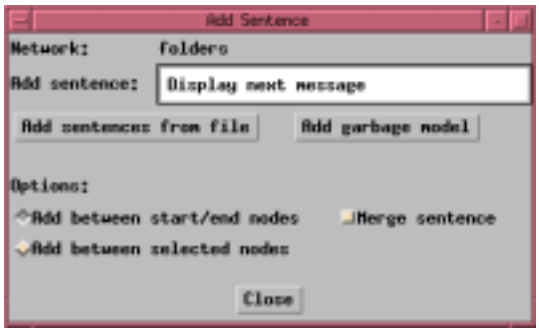


**Fig. 2.6 Add sentences dialog**

Completing the parallel words "PREVIOUS", "NEXT" *etc* in the enclosed network fragment of figure 2.5 can now be greatly speeded up. First, left-click the null nodes at each end to select them, and with option "Add between selected nodes" selected, individually enter "first", "last" *etc* into the "Add sentence:" field.

The "Merge sentences" option attempts to find a common path portion leading from the selected start node and to the end for the sentence. For example, try entering an additional sentence "Reply to the previous message" to the completed network fragment of figure 2.5 with merging enabled. Notice how this merges only the forward "Reply to" portion of the sentence. This is because adding only the word "THE" onto the canvas, and merging the remainder of the sentence by feeding into the word "PREVIOUS", implicitly adds the (possibly unwanted) sentence "Reply to the previous".

Note that this method of adding words nodes does not force the user to first ensure a dictionary entry first exists. If a word with no corresponding dictionary entry is inserted into the network by this means, then in order to use the recogniser, a dictionary entry must either be explicitly created (as described in section 2.4) or a user-dictionary containing an entry for that word loaded.

### 2.3.3 Adding the *msg_no* sub-network

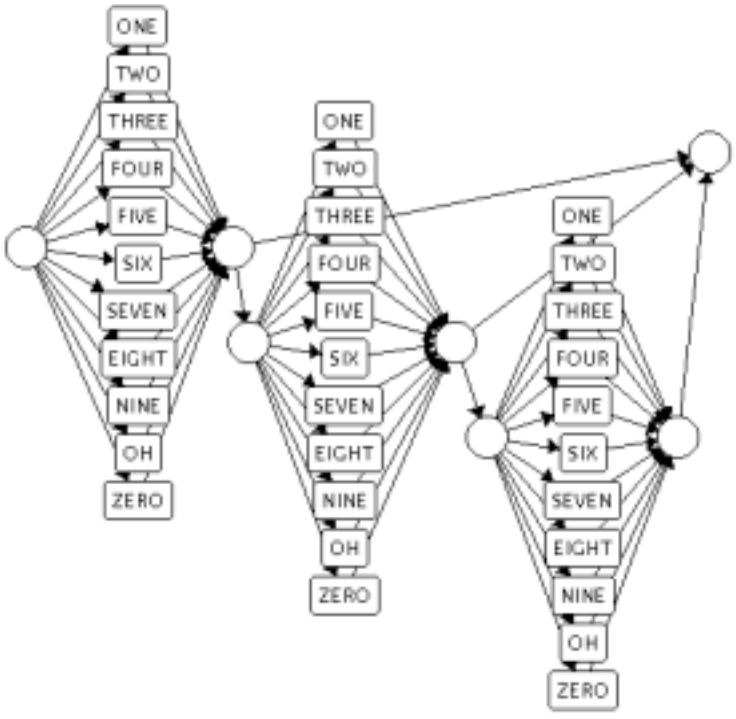


**Fig. 2.7 Sub-network** *msg_no*

To create the *msg_no* node the corresponding sub-network must first exist. A simple example of sub-network specifying numbers of up to three digits is shown in figure 2.7.

As sub-networks are just like any other network, they are created in exactly the same way by first selecting New Network from the **File** menu. This switches the display to a new blank canvas on which *msg_no* can be constructed. Once one of the digit portions has been created it can be selected, and the copy and paste operations applied to rapidly build the network. Once constructed the network should be saved as the desired sub-network name - in this case *msg_no*. This network can then be loaded independently or inserted as a sub-network into other networks.

The canvas can now be switched to the previous display of the e-mail network by selecting the latter from the Windows menu. To insert the *msg_no* sub-network node, select Insert Sub-Network from the pop-up menu. This brings up a file browser from which the sub-network file is selected before the corresponding node appears on the canvas.

Sub-networks can be nested to any depth. As the *msg_no* sub-network consists of a repeated structure, it may be better to create a more basic *digit* sub-network from which *msg_no* can be composed as shown in figure 2.8.

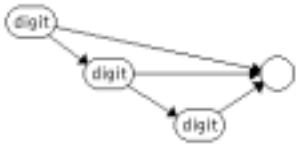


**Fig. 2.8 Building *msg_no* from a *digit* component**

It is important to note that each of the three digit nodes references *the same* instance of the *digit* sub-network, i.e. any changes to the *digit* sub-network are reflected everywhere. This can be easily verified by viewing the *digit* sub-network from, say, the leftmost node and making a change. When now viewed from the rightmost node the same change should be visible.

The Netbuilder provides a convenient facility to create a variant of a sub-network. For example, we might want the leftmost digit sub-network to include "DOUBLE". Selecting Clone Sub-network from the pop-up menu prompts for a new sub-network name, e.g. *digitdouble*. Sub-network *digit* is cloned to form this new sub-network which can be edited and inserted elsewhere as needed.

In practice the *msg_no* sub-network needs extending to allow more natural specification of three digit numbers such as "one hundred and four" etc. Using a basic building block such as the *digit* sub-network, an extended *msg_no* sub-network can be easily created which will not be discussed here. For simplicity we continue with the current *msg_no* sub-network.

There is an important distinction between *nested* sub-networks (inserting a sub-network into another sub-network) and *recursive* sub-networks (inserting a sub-network into itself). Recursive sub-networks are not supported (or needed) in ***grapHvite***. Inserting the *msg_no* sub-network into itself, or into another sub-network that has the *msg_no* sub-network higher up in its hierarchy, generates a warning dialog.

Finally, note that navigating between networks and sub-networks can be done using the hierarchical network list maintained in the Windows menu. The top level networks are listed. Sub-network *msg_no* should appear in a sub-menu to the side of the entry for the e-mail network. Alternatively, select View Sub-network from the pop-up menu when over the *msg_no* node. To return to the e-mail network select View Parent Network from the pop-up menu on the canvas.

## 2.4 Completing the e-mail network

At this stage the portion of the e-mail network shown in figure 2.5 has been created along with sub-network *msg_no* and its sub-networks. For convenience the portion of the network inside the rectangle can be encapsulated as a sub-network called *msg_spec*, defining valid ways of specifying a message. This is easily done by selecting the relevant nodes, cutting them from the network, pasting them onto a new blank canvas and saving under the appropriate name.

One more sub-network is required before we can complete the network: sub-network *folders*, which defines a list of e-mail folders. A very simple example is given in figure 2.9.


**Fig. 2.9 Example sub-network *folders***

### 2.4.1 Sub-network *folders*: adding new words

The folder sub-network is likely to contain application-specific words which may not have entries in the system dictionary. For example, "OUTBOX" is not present in the core dictionary.

Entries for new words are added to a user dictionary. Selecting New Dictionary from the File menu creates a new user dictionary. To add an entry for "OUTBOX" select Add New Word from the Dictionaries menu. This displays a dialog as shown in figure 2.10.

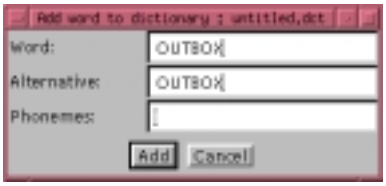


**Fig. 2.10 Adding a word to a user dictionary**

The word and alternative output symbol fields are automatically filled with the contents of the dictionary search field. The alternative output symbol is the actual text output for the word during recognition, and this defaults to being the same as the word entry (more on this later).

To form the pronunciation (phone sequence) for "OUTBOX", similar or component words in the system dictionary make a useful reference. Here, (assuming a UK English system) the word "OUT" has an entry with phone sequence "aw t sp" as does "BOX" with sequence "b oh k s sp". All pronunciations in the supplied system dictionary end with phone "sp" which represents optional post-word silence (see section 2.4.2). The new pronunciation should just have the final sp can be entered as "aw t b oh k s sp".

This method of deriving new pronunciations, although a little *ad hoc*, is usually quite effective. As a guide, a list of the phones and example pronunciations can be found in Appendix B.

Clicking Add enters the word and pronunciation in the user dictionary panel to the right of the canvas. This panel by default displays the list of used words for the

current network (selecting Show used words list from the Dictionaries menu allows this to be viewed). New entries can be added to the user dictionary or existing ones edited or deleted using the Dictionaries menu options.

Clicking entry "OUTBOX" in the user dictionary panel selects this entry allowing the creation of a corresponding word node folders sub-network.

### 2.4.2 Adding silence nodes

There is very likely to be a brief period of silence at the beginning and the end of any continuous speech input to the recogniser. This must be explicitly represented by the task network, otherwise the ***grapHvite*** decoder can only match speech acoustic models to silence at the start or end of speech, which degrades the recognition accuracy. Table 2.1 shows the dictionary entries for silence.

| Entry | Pronunciation | Usage |
|---|---|---|
| $< s >$ | sil | Pre-speech silence |
| $< /s >$ | sil | Post-speech silence |
| $< /p >$ | sil | Pause during speech |

Table 2.1: System dictionary silence entries

Silence nodes are added to a network just like any other word node, as can be seen in figure 2.2, and are normally only placed at the start and end of the top level network, i.e. any contained sub-networks should not have silence at the start and end.

Short pauses between words on the other hand are not explicitly represented in the network. This happens implicitly by the short pause (sp) specified at the end of each entry in the system dictionary. The sp phone represents optional silence, i.e. this model can be skipped during recognition if a pause does not occur after a word. Although not mandatory it is a good idea to append sp when entering new pronunciations.

## 2.5 Testing the network

Once the network is constructed, it can be tested to ensure that it reflects the task syntax as intended. Selecting Verify Network from the Networks menu reports any stray nodes or the absence of a valid start and end node. Note that this applies only to the top level network and sub-networks are not tested in turn.

With a valid network, selecting Test Sentences from the Networks menu invokes a dialog as shown in figure 2.11. Repeatedly clicking Generate random sentence appends a sentence into the text display area, thus allowing quick detection of undesirable sentences. Typing a sentence into the Test sentence: field and hitting return indicates whether the sentence can be generated by the network or not. Try typing the sentence

Show my new messages

into the test sentence field and hitting return. This outputs the sentence prefixed by "failed:" in the output area, indicating there is no corresponding path through the

network. Now try

Show new messages

This outputs the sentence and highlights the relevant network path. Alternatively selecting Test sentences from file imports a text file of sentences which are each tested against the network.


**Fig. 2.11 Testing network**

## 2.6 Testing the recogniser

Once the network has been constructed the performance of the e-mail recogniser can be informally evaluated. Clicking the Recogniser button on the bottom right of the main Netbuilder window spawns a separate dialog window which provides the interface to the recogniser. An example of this is shown in figure 2.12.

The following simple steps are required to set up the recogniser before it can be used.

- **Configuring the recogniser** Selecting Configure Recogniser from the **Config** menu brings up a simple dialog to select between the supplied monophone or triphone acoustic models and to select one of mic-in, line-in or file as the audio source for the recognition session.
- **Loading the recogniser** Simply selecting Load Recogniser from the **Config** menu now loads the acoustic models and connects to the appropriate audio source for the recognition session. Given that the e-mail network is valid and all task words have a corresponding dictionary entry, the recogniser should be successfully created.
- **Calibrating the silence detector** If either mic-in or line-in has been selected for the audio source, the ambient silence level needs to be estimated for the silence detector to detect the end of each spoken input. Selecting Calibrate Silence from **Config** displays a simple dialog which allows this to be quickly measured.

These steps are usually very straightforward and more detail can be found in the reference section if required. Once loaded and calibrated successfully, the START button should be enabled indicating that the recogniser is ready to process speech input. Clicking START gives the signal to start reading from the audio source and primes the recogniser to wait for speech to begin. The status panel at the bottom left should immediately display "Recognising...", at which point a sentence can be spoken into the appropriate input channel.

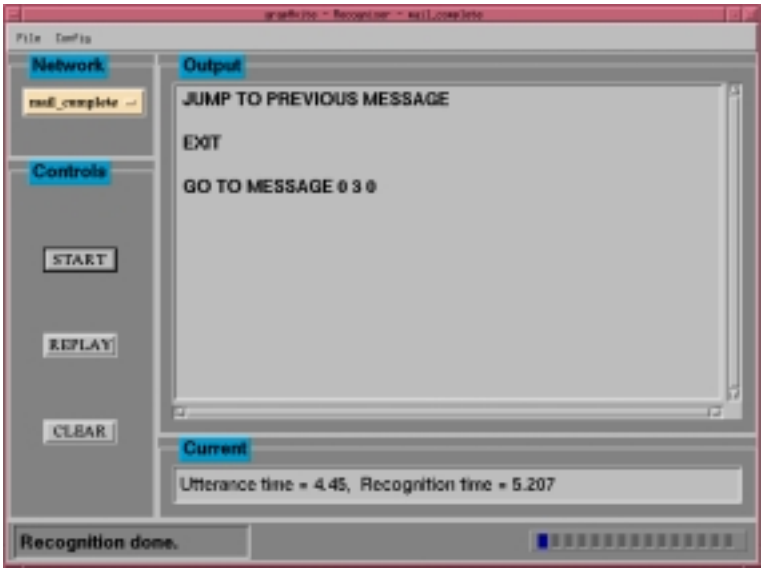


**Fig. 2.12 The Netbuilder recogniser window**

Try speaking

Go to last message

As the recogniser processes the incoming frames of speech, the Current panel continually displays the best recogniser hypothesis to date. On completion the most likely hypothesis is output to the Output panel. If recognised correctly, the output "GO TO LAST MESSAGE" should appear.

It is possible to refine the network and to experiment with the recogniser for the current task. For example we may want to add a link from "MESSAGE" to subnetwork node *msg_no* in figure 2.5 to bypass the word "NUMBER", allowing inputs such as:

display message three five four

Returning to the recogniser window after modifying the network, the START button now appears as UPDATE. Clicking this updates the recogniser with respect to any structural changes to the network and resets the button to START if the update is successful. Interactive testing of the recogniser often highlights any weaknesses in the network design which can then be fixed in the main Netbuilder window and the cycle of testing the recogniser and refining the network continued as needed.

### 2.6.1 Alternative output symbols

Up until now little consideration has been given as to how an application might process the recogniser output and hence determine an appropriate action. As can be seen from the "Output" panel in the recogniser window, each word along the decoded network path is represented literally. An alternative output symbol can be specified for each output word however. For example outputting a single character for each digit in the msg_no sub-network (e.g. "8" instead of "EIGHT") may simplify the application's job.

Alternative output symbols are specified or modified in a user dictionary. Saving the used words list ( Save Used Words As... ) and reading it in as a user dictionary ( Load User Dictionary ) via the main Netbuilder **File** menu displays the used words in the user dictionary panel[2]. Selecting "EIGHT" from this list and Edit Word Entry... from the **Dictionaries** menu displays a very similar dialog to that of figure 2.10. The Alternative: field can be set to "8" and similarly for all other digit entries. These changes to the digit output symbols are immediately visible on return to the recognition window. If the user now says

Go to message zero two oh

the output "GO TO MESSAGE 0 2 0" should be produced.

To further simplify the application's job of deciding what action to take for the given recogniser output, redundant words can be given blank output symbols. Considering again the earlier fragment of the e-mail network created (see figure 2.3). Each path has the same meaning (or semantic content) from the application's point of view, namely to display new messages. Each word could be given a blank output symbol except for the word "MESSAGES" which might be set to output something specific to the application, e.g. "C1". Any of the recognised paths through this part of the e-mail network will be decoded as "C1". However, this is likely to be problematic if any of the words in this part of the network are crucial to the semantics of some other part. To solve this problem ***grapHvite*** allows word entries to be created with blank pronunciations to be used as semantic markers. If a dictionary entry "C1" with blank pronunciation is created, a corresponding word node can be inserted after "MESSAGES" as a semantic marker. Other dictionary entries can be similarly created for use as semantic markers at other appropriate points in the network. Semantic parsing of the recognition output is discussed in more detail in *The HAPI Book*.

### 2.6.2 Out–of–vocabulary rejection

It is often desirable to detect the occurrence of out-of-vocabulary (OOV) words or phrases which may be input to a real system. Although pragmatic design of the recognition task will hopefully minimise the likelihood of OOV inputs in the first place, the system must still be able to accurately detect OOV speech in order to perform robustly. A typical example of when OOV is especially critical is where a list of items such as the *folders* sub-network is specified. If the user speaks a word or phrase not represented by the list of folders the recogniser attempts to match this to

[2]To re-assign output symbols for just the digits it is easiest to switch the canvas to the *msg_no* (or digit) sub-network so that only the digits appear in the used words list.

a valid folder name. [3]

The ***grapHvite*** Netbuilder allows the user to experiment with a simple form of rejection based on the HAPI confidence score. Selecting "Settings" from the **"Rejection"** menu in the recognition window invokes a dialog similar to that shown in figure 2.13. This allows a threshold value to be set and words with a confidence score below this threshold are rejected.

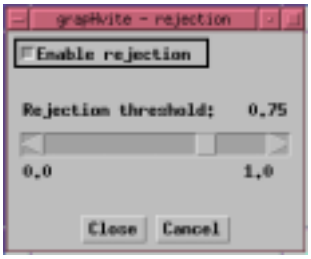


**Fig. 2.13 Rejection dialog**

Selecting "Enable rejection" actually activates this rejection mechanism and rejected words appear in red in the recogniser output window. [4]

An alternative approach to OOV rejection is to insert a *garbage model* in parallel with the list of folder names to provide an alternative path for OOV speech to match to during recognition. Clicking Add garbage model in the "Add Sentence" dialog (see figure 2.6 above) inserts a garbage model between the specified start/end nodes as can be seen in figure 2.14.

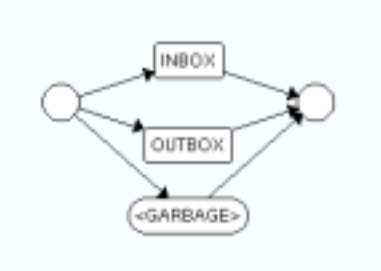


**Fig. 2.14 OOV rejection using a garbage model**

The garbage model path is effectively an alternative path to which OOV speech can be matched during recognition. This is particularly effective when the input differs acoustically from any of the in-vocabulary items. In practice, the garbage model generally scores higher than the competing in-vocabulary paths, resulting in valid inputs being frequently recognised as "<GARBAGE>". A penalty on the link into the garbage model is automatically set to counter this effect, and this can be further adjusted manually if need be to optimise the rejection rate for the given task. This is described in section 3.5.6.2 of the reference.

[3] Assuming, of course, that the rest of the command is correctly matched – an OOV word at any point may tip the balance in favour of some other competing path through the network.

[4] Note that there are certain conditions where the current confidence score estimate may not perform reliably - these are discussed in the ***grapHvite*** reference section.

## 2.7 Exporting to a HAPI/JHAPI application

Once the e-mail network has been suitably refined within the Netbuilder, the following components can be exported to allow creation of the e-mail recogniser from within a HAPI application.

- **Network** To export the e-mail network first ensure the canvas displays the top-level network (i.e. not any of the constituent sub-networks) and then select Save Network In HAPI Format from the **File** menu. The network is written to file in the required format. Before doing this it is worth checking that explicit silence nodes are attached at the start and end of the network.
- **Dictionary** The list of words used in the e-mail network also needs to be exported as the e-mail task dictionary. Again, the canvas should display the top-level e-mail network before selecting option Save Used Words As from the **File** menu to write the task dictionary to a specified file.
- **Acoustic models** ***grapHvite*** comes with sets of acoustic models which can be directly used by the task recogniser in the final application. The location of the supplied models and the accompanying model list files can be found from the Netbuilder configuration file which is described in Appendix A of the ***grapHvite*** reference.

In addition to the above components, a HAPI configuration file is needed to specify various parameters which define the operation of the HAPI decoder. The HAPI configuration used by the Netbuilder is likely to be very similar to that required for the HAPI application and serves as a useful reference. Its location is specified in the Netbuilder configuration file described in Appendix A of the ***grapHvite*** reference.

As stated at the beginning of this tutorial, the reader is referred to *The HAPI Book* for a detailed description of how to create and run a task recogniser from within an application using the HAPI/JHAPI interface.

Finally, the HAPI interface provides useful functionality in addition to a recogniser interface. For example, although the *msg_no* sub-network defines a number in the range 1 to 999, there may be far fewer current messages to select from. The application itself might create a *msg_no* network online to reflect the actual number of existing messages and avoid the possibility of allowing invalid message numbers to be specified. In practice this approach might have a complicated implementation, and an alternative may be to obtain multiple hypotheses from the recogniser results and ignore hypotheses containing illegal message numbers.

# Chapter 3

# Reference Section

This reference is organised under three topics: *network building*, *dictionary handling* and *recognition*. Network building is described in section **3.3** and details how networks are created, modified and tested using the graphical editing features of ***grapHvite***. Section 3.4 deals with the ***grapHvite*** Netbuilder interface to system and user dictionaries and describes how new entries and dictionaries are created and how dictionary entries are inserted into networks. Section 3.5 explains how networks under construction are tested with actual speech using the recogniser interface.

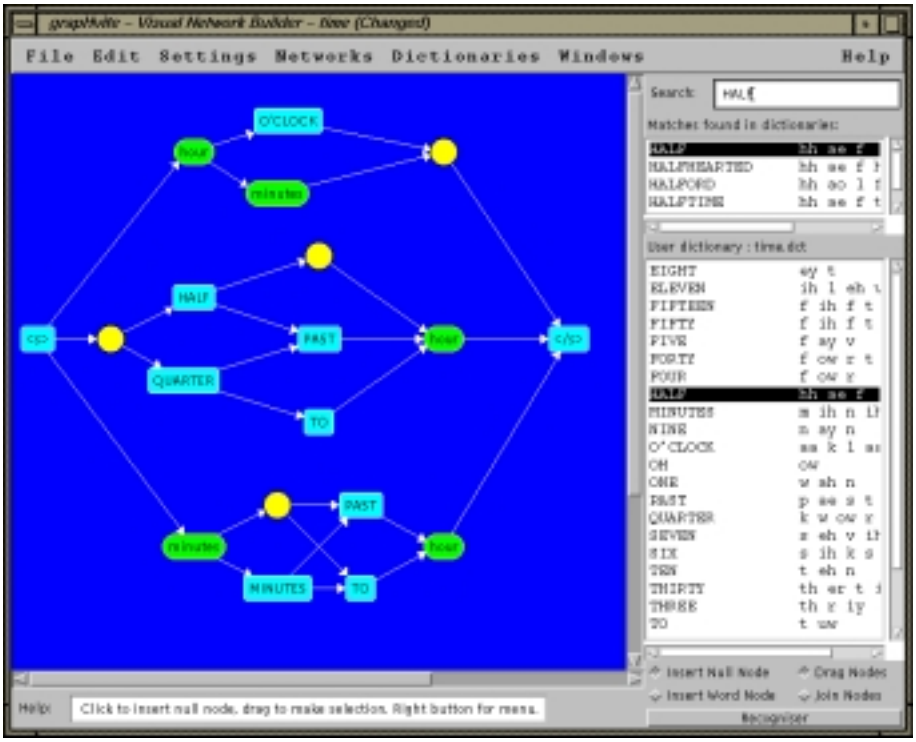


**Fig. 3.1** ***grapHvite*** **Netbuilder front panel displaying the** *time* **network**

Additionally, there are a number of appendices which cover some of the details of various ***grapHvite*** features not included in the main text. These include configuration variables, dictionaries, phone and acoustic models sets, and Standard Lattice Format, which is the format in which ***grapHvite*** networks are stored.

## 3.1 A Tour of the Main Window

Figure 3.1 shows the front panel of the ***grapHvite*** Netbuilder interface. The syntax network currently loaded is the *time* network, which can be used to recognise the time of day using the twelve hour clock.

The front panel consists of the following components:

**The Canvas** — The *canvas* takes up the major part of the ***grapHvite*** Netbuilder front panel and is the scrollable graphical drawing board on which networks are created and edited.

**The Help Status Bar** — The nodes, links and sub-networks which constitute a network are created and modified using mouse operations. The actions of the mouse depend on where the mouse pointer is moved on the canvas and the Help status bar below the canvas displays messages indicating what these are.

**The Mouse Check Boxes** — Towards the bottom right of the window are a set of four selection buttons which toggle the functionality of the left mouse button when used on the main canvas. Insert Null Node or Insert Word Node are the alternative options for a left mouse click on the canvas and Drag Nodes or Join Nodes are the options when the mouse is clicked and dragged on a node.

**The Recogniser** — Once constructed, networks may be tested immediately using the built-in ***grapHvite*** recogniser. The **recogniser** button on the bottom right of the front panel invokes a recogniser window in which networks may be tested with live or pre-recorded speech input.

**Dictionary Search Field** — The dictionary Search field to the right of the canvas is used to look up dictionary entries for insertion as word nodes in the network.

**Matches found in dictionaries** — Only the first few characters of a word need be typed in the Search field and all entries that match are displayed in the scrollable list Matches found in dictionaries.

**The User Dictionary field** — ***grapHvite*** supports both a **System Dictionary** and any number of **User Dictionaries**. Below the search engine is a scrollable display area User dictionary which is currently showing the entries in the user dictionary *time.dct*. The user dictionary displayed in this area is selected from the **Dictionaries** menu.

**Words used in network** — Not shown in Figure 3.1. The User Dictionary field may also be used to show a list of all the words in the currently loaded network, and this too is selectable from the **Dictionaries** menu.

The Menu Bar — The main window menu bar at the top has a series of seven menus which are briefly summarised below:

| | |
|---|---|
| `File` | **Create**, **Load**, **Save** and **Close** network and dictionary files. **Load Full System Dictionary**. **Exit *grapHvite*** Netbuilder. |
| `Edit` | **Undo**, **Redo** network editing operations, **Cut**, **Copy Paste**, **Delete** selected nodes on the canvas |
| `Settings` | View/modify ***grapHvite*** Netbuilder **configuration** settings, canvas **Snap to grid** and **Zoom** settings. |
| `Networks` | **Verify** network is correctly formed, **AutoArrange** network layout on canvas, test network with text (**Test Sentences**) or with speech input (**Start Recogniser**). |
| `Dictionaries` | **Display**, **Add**, **Edit** and **Delete** words from user dictionaries. This menu is only selectable when a user dictionary has been created or loaded. |
| `Windows` | Swap another network or subnetwork into the canvas. The windows menu lists all loaded networks. |
| `Help` | No online help is currently available for ***grapHvite***. This menu displays version information only. |

## 3.2 Some Common *grapHvite* Netbuilder Windows

There are some types of ***grapHvite*** windows which are used frequently. These may differ slightly in detail, but the basic structure is the same.

### 3.2.1 The File Browser Window

This window is used to select files to be loaded or saved as ***grapHvite*** networks, dictionaries, speech waveforms and so forth. It displays a scrollable list of files in the default directory for the particular file type (see Appendix **A**: *Configuring the **grapHvite** Netbuilder*).

Figure 3.2 shows an example of a **File Browser** which is being used to load a network from disk.

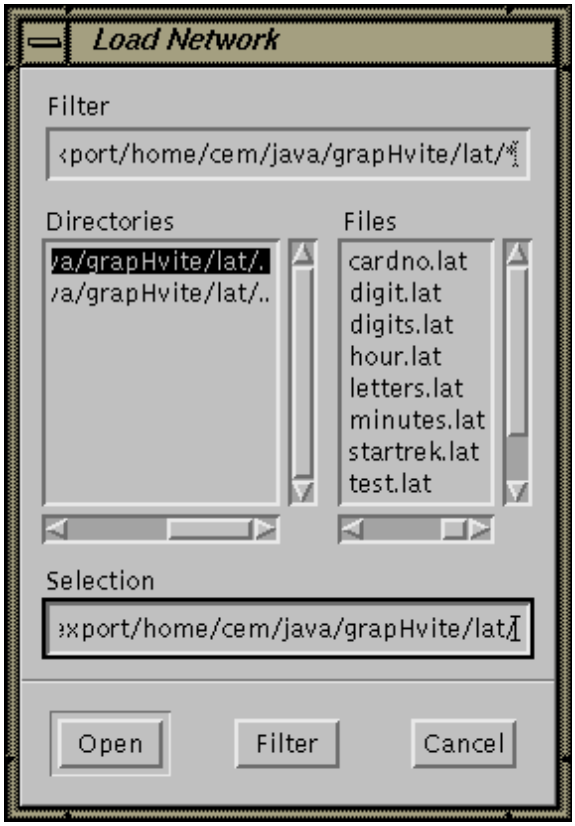


**Fig. 3.2 The File Browser Window**

The various options available from the File Browser are as follows:

Filter — Fill in this field to select a different directory and/or subset of files to view. For example, **lat/*.lat** will display all files in the sub-directory **lat** with the **.lat** extension. Press the Filter button to view your selection.

Directories — Displays an alphabetic list of directories in the current directory selected by the Filter field. Scroll this list and double click on an entry to select another directory to view. Double click on ../ to change to the parent directory.

Files — Displays an alphabetic list of files matching the Filter specification. Scroll this list and click on your selected file. Load or save this file by:

- Double Clicking
- Pressing the *Enter* key
- Pressing Open (or Save where appropriate)

Selection — Displays the currently selected file and directory. These can be typed directly into the Selection area. Alternatively the file browsing facilities available can be used to locate the file.

| | |
|---|---|
| Open | Opens the currently selected file. |
| Save | Saves to the currently selected file. |
| Cancel | Cancels the window. |

***grapHvite*** networks, user dictionaries, text input files and audio files have a default location and extension which is determined by the appropriate configuration setting (see Appendix **A** *Configuring the* ***grapHvite*** *Netbuilder*). When loading or saving, the **File Browser** will start with a view on the default directory, and filenames which are supplied *without* extensions will have default extensions appended.

### 3.2.2 The Warning Window

The warning window is generated by the ***grapHvite*** Netbuilder to warn the user about a possible error condition. Figure 3.3 below shows an example of a warning window. This consists of a warning message and one or more red buttons. In this example, the ***grapHvite*** Netbuilder is warning the user that the network about to be used for recognition does not have nodes representing silence at the beginning and end of the network. This particular window allows the user to remedy the problem directly by pressing Attach Silence or to Continue regardless.


**Fig. 3.3 Warning Window**

### 3.2.3 The Information Window

This simply displays information and the user must press the grey Ok button to continue. Figure 3.4 shows an example of an information window displayed by the network verifier.


**Fig. 3.4 Information Window**

## 3.3 Network Operations

This section describes how networks are constructed and tested using the *canvas*, the graphical drawing board, which takes up the greater part of the ***grapHvite*** Netbuilder front panel.

### 3.3.1 Creating a Network

Networks are created by selecting the New Network option from the **File** menu. This creates a blank network with the name *untitled.lat*. The user must rename this network when saving the file (see section 3.3.3).

### 3.3.2 Loading a Network File

An existing network may be loaded from disk by choosing the Load Network option from the **File** menu. This displays the **File Browser** window (see 3.2.1) which allows the user to locate the required network file and load it into the ***grapHvite*** Netbuilder.

Networks have a root file name (e.g., *time*) and default extension *.lat*. If only the root file name is supplied to the **File Browser**, the system will append the *.lat* extension.

When the **File Browser** is first used in a ***grapHvite*** Netbuilder session, it will try and load from the default subdirectory *./lattices* in the directory from which the ***grapHvite*** Netbuilder was called. If you change to another directory to load the file, the **File Browser** will start from this new directory the next time it is used.

Both the default network directory *./lattices* and network extension *.lat* are configurable. You may change them by resetting the *Networks Directory* and *Network Suffix* configuration variables (see Appendix **A**).

#### 3.3.2.1 Loading a network file with sub-networks

The sub-networks *hour* and *minutes* in the *time* network are stored in separate network files *hour.lat* and *minutes.lat* respectively. They are in every way the same as stand-alone networks and may be used as such if required [1].

When a sub-network is first loaded (using Insert Sub-network from the **pop-up menu** - see section **3.3.5.7**) it is given the same name as its root file name and the parent network keeps a record of its full file name and the directory from which it was loaded. The user may view this by selecting Properties from the **pop-up menu** from within the sub-network (see section **3.3.5.20**).

When a network which references a sub-network is loaded, the sub-network file will be searched for in the directory in which the subnetwork was originally located. If it is not found there, the ***grapHvite*** Netbuilder will look in the directory of the parent network and then if it is still not found, in all the directories given in the configuration variable *Subnetwork Path* (see Appendix **A**). This is a colon separated list of directories (e.g., *sublats1:sublats2:sublats3*), with the default setting:

```
SubnetPath = ./lattices
```

Finally, if the file is still not found, you can specify a new location using the **File Browser** window.

[1] but they should not contain initial and final silence nodes - see Appendix **B**

### 3.3.3 Saving Network Files

Networks are saved to disk using the **File** menu options Save Network, Save Network As... and Save Network in HAPI Format.

Save Network — saves a network to its original file name in its original directory. Any embedded sub-networks which have been edited during the session are also saved in the location from which they were originally loaded.

Save Network As... — allows you to select a directory and file name for the network to be saved using the **File Browser** (see section **3.2.1**).

The **File Browser** behaves in the same way as when a file is loaded (see section **3.3.2**), selecting the default network directory *./lattices* and appending a *.lat* to any file names which have no extension.

If you Save Network As... from within a *sub-network*, the sub-network will be saved to the new directory and/or filename as above and *all instances* of this sub-network will then refer to this new location. If you change the root filename, all instances of the sub-network will be renamed to reflect this. For example, saving the *minutes* sub-network of the *time* network (see Fig **3.1** on page 31) as as *min.lat* would mean that all *minutes* nodes would be renamed *min*.

If you only want to change a specific instance of a sub-network, you must use Clone (see section **3.3.5.9**).

Save Network in HAPI format — This option merges the parent network and all its sub-networks into one file. It is used when networks have been fully developed and tested and are to be built into an application using the HAPI/JHAPI interface (see section **2.7**).

### 3.3.4 Switching Between Networks

Networks may be swapped in and out of the canvas by selecting them from the **Windows** menu. This menu lists all main networks (together with any sub-networks they employ) currently loaded. Networks (either main or sub-networks) are brought into the canvas by selecting the appropriate network from the **Windows** menu. Any network which contains sub-networks will be listed with a sub-menu underneath it, allowing the selection of the individual sub-networks connected to the main network.

Note that sub-networks can also be viewed by right-clicking on the sub-network node and choosing View Sub-Network from the **pop-up** menu. Similarly parent networks can be viewed by right-clicking on the canvas and choosing View Parent Network from the **pop-up menu**.

### 3.3.5 Editing Networks on the Canvas

The basic network editing operations of the ***grapHvite*** Netbuilder allow you to add and delete nodes and links, drag nodes around the canvas, and cut and paste them into other networks. These operations are performed by combination of mouse actions and options provided under the **Edit** menu. Below is a brief description of the general

functionality of each of the mouse buttons. Following this, each editing option is described separately.

#### 3.3.5.1 The Left Mouse Button

The action of the left mouse button depends on:

- whether the mouse pointer is over a node or not.
- the check box settings in the bottom right hand corner of the front panel.

Checkbox settings can be changed by either clicking on the boxes or using the right mouse button **pop-up menu** (see below) .


**Fig. 3.5 Mouse Action Checkboxes**

When the mouse pointer is over the **canvas**, clicking the left mouse button will either create a **null** node or a **word** node, depending on whether the Insert Null Node or Insert Word Node node checkbox is set on the front panel. Click and drag produces an extendible rectangle which is used to select a number of nodes. Note that if a node selection is currently active on the canvas then clicking the left mouse button will simply deselect the node selection and will not create a new node.

When the mouse pointer is over a **node**, click and drag will either move the node (or selection) or join it to another node (or selection), depending on the state of the Move Nodes / Join Nodes checkboxes on the front panel. Clicking without dragging on a node **selects** the node (adding it to the current selection) or alternatively **deselects** the node if it is already part of the current selection.

| Mouse Action | On a node | On the canvas |
|---|---|---|
| Click | Select/Deselect | Insert word node/null node |
| Click and drag | Move nodes/Join Nodes | Select Nodes |

Table 3.1: Left mouse action

#### 3.3.5.2 The Right Mouse Button

Clicking on the right mouse button displays a context-sensitive **pop-up menu**, with options which depend on whether the mouse is pointing to a node or a link or is over empty canvas, and whether the network being edited is a sub-network or a main network.

The following functions are available from the **pop-up menu** in the relevant context:

- Move Nodes
- Join Nodes

- Null Node
- Word Node
- Insert Sub-Network
- View Sub-Network
- Clone Sub-network
- View Parent Network
- Delete Node
- Delete Link
- Properties

The options Move Nodes / Join Nodes and Null Node / Word Node are a convenient way of toggling the front panel checkboxes which determine the action of the left mouse button. This means that selecting Word Node from the pop-up menu does **not** insert a null node immediately: it changes the action of the left mouse so the the next time it is clicked, a word node will be inserted.

Other options are described under separate headings in the remainder of this chapter.

#### 3.3.5.3 The Middle Mouse Button

Where a middle mouse button is available its action is also context sensitive depending on whether the mouse pointer is over the canvas or over a node.

When over a node the middle mouse will **move** nodes if the front panel Join Nodes checkbox is set and **join** nodes if the Move Nodes checkbox is set.

When over the canvas, click and drag allows the user to scroll around the canvas. There is no action for click alone.

#### 3.3.5.4 The Help status bar

A context-sensitive help message which explains what mouse actions are currently available is displayed in the Help status bar underneath the canvas (see figure 3.6).


**Fig. 3.6 The Help status bar**

#### 3.3.5.5 Creating Null Nodes

Ensure that the Insert Null Node check box is **on**. Left-click on the canvas where the node is to be inserted.

#### 3.3.5.6 Creating Word Nodes

Ensure that the Insert Word Node checkbox is **on**. Select a word from the dictionary panel and left-click on the canvas where the word node is to be inserted.

Words can be selected from the dictionary by either clicking on a word displayed in the two lists Matches found in dictionaries and Word used in network or entering the word in the Search field. Only the first few characters of a word need be typed, and all words which match will be displayed underneath. Select the desired word by clicking on its entry.

If the required word is not found in the loaded dictionaries, you can add it to a user dictionary of your choice by selecting Add Word from the **Dictionaries** menu. You must either create or load a user dictionary to do this. See section 3.4 for more advice on how to use the Dictionary Panel.

Note, if a dictionary entry is not highlighted, word nodes will **not** be inserted when the mouse is clicked on the canvas. See the Help status bar if this happens.

#### 3.3.5.7 Inserting Sub-network Nodes

Move the mouse pointer to where you want to insert the sub-network node. Right-click on the canvas and select Insert Sub-Network from the **pop-up menu**. Then use the **File Browser** window in the same way as if you were loading a network (see section **3.3.2**). A sub-network will be inserted at the position of the mouse with the same root name as the file you choose. For example, choosing *person.lat* as your file would create a *person* sub-network.

If *person.lat* has already been loaded in this ***grapHvite*** Netbuilder session, either as a stand-alone network or a sub-network, inserting a further sub-network node *person* will **not** cause the ***grapHvite*** Netbuilder to reload the file. The sub-network node inserted will refer to the *person* network already loaded.

#### 3.3.5.8 Viewing and Editing Sub-Networks

To view or edit a sub-network, position the mouse pointer over the sub-network node and choose View Sub-Network from the **pop-up menu**. Alternatively select the sub-network from the relevant side menu in the **Windows** menu. The sub-network can now be edited in just the same way as the parent network.

To return to the parent network, you can right-click on the canvas and select View Parent Network from the **pop-up menu**. Alternatively, you can select the parent network from the **Windows** menu.

#### 3.3.5.9 Cloning Sub-networks

*Cloning* a sub-network means that you make an identical copy of a sub-network at its original location which you can then edit without affecting any other sub-network nodes.

Select Clone Sub-Network from the **pop-up menu** (see section **3.3.5.2**). This makes a copy of the sub-network and prompts the user for a new name for the cloned sub-network. The sub-network is then saved to disk and is completely separate from the original sub-network from which it was cloned.

Sub-networks which have been created by cloning behave in just the same way as any other sub-network. They are saved with the usual file-naming conventions and they may be inserted elsewhere in the canvas, as usual.

#### 3.3.5.10 Adding Sentences

Partial or complete sentences can be quickly inserted into a network using the "Add Sentence" dialog - invoked by selecting Add Sentences... from the **Networks** menu. An example of this dialog can be seen in figure 3.7.


**Fig. 3.7 The Add Sentence Dialog**

A sentence or phrase is simply typed into the text field and the default action is for a corresponding path of word nodes to be created and appended between the network start and end nodes. If the canvas is initially blank, then start and end nodes are first automatically created.

A sentence can also be inserted between two selected nodes in the network by selecting the "Add between selected nodes" option. To select an alternative start/end node pair, left-click on the desired start and end node (in that order) on the canvas. Sentences can also be entered from a text file - clicking Add sentences from file invokes a file browser in which a file can be selected and sentences specified in this file are read in and inserted into the network.

With "Merge sentence" enabled, an attempt is made to merge the beginning and/or end portion of a sentence when an identical start/end partial path already exists in the network. For example, adding the sentence

> The cat sat on the mat

to a network which contains only the sentence

> The dog sat on the mat

merges the initial "The" and final "sat on the mat" portions of the sentence and only a new word node "dog" is created. Although this simple merge facility does not generally lead to a globally minimised network, it does reduce the build up of redundant network nodes which can potentially reduce recogniser efficiency.

Two conditions can occur where a sentence start/end path will not be fully merged (if at all) with an identical existing partial path:

- Links to be shared have different penalties. Links are created with a default penalty value and will not be shared with links having a non-default value. Note that this is easier to spot when links with non-default penalties are highlighted.

- Nodes along the existing path portion branch outwards from that path. Path sharing in this case can potentially introduce unwanted paths into the network, as for example a network contains only the two sentences

```
The-cat-sat-on-the-mats
                  \
                   mat
```

  then when merging the sentence

```
The-dog-sat-on-the-mat
```

  only the initial "The" and final "mat" are shared as sharing the "sat on the mat" portion introduces a new sentence "The dog sat on the mats" which may not be desired.

#### 3.3.5.11 Selecting and Deselecting Multiple Nodes

A set of nodes can be created in one of two ways. Individual nodes can be selected and added to the node selection by left-clicking on each node in turn. Alternatively placing the mouse pointer on the canvas and left-dragging will produce an extendible rectangle. Releasing the left mouse button selects all nodes within the rectangle. A node selection is deselected by left-clicking once on the canvas.

#### 3.3.5.12 Deleting Nodes and Selections

To delete a node, select Delete Node from the **pop-up menu** (see section **3.3.5.2**). The node and all links to and from it will be deleted. To delete a node selection choose Delete Selection from the **Edit** menu.

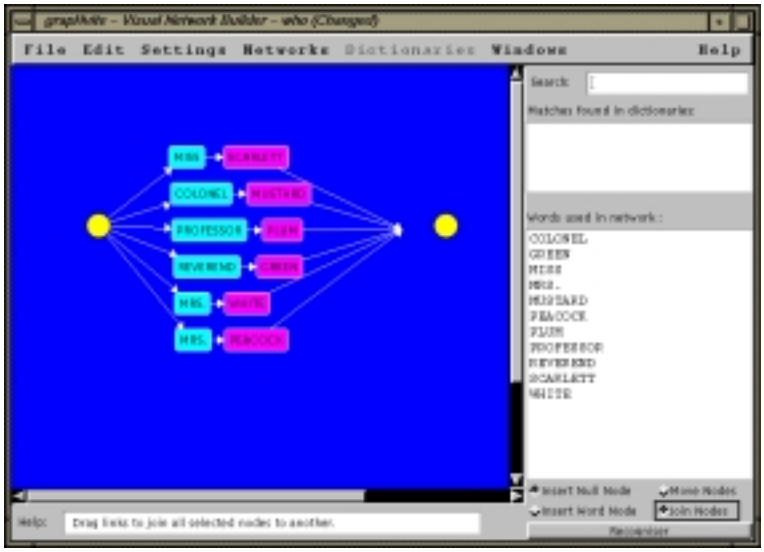


**Fig. 3.8 Linking Nodes**

#### 3.3.5.13 Adding and Removing Links

Ensure that the front panel Join Nodes checkbox is **on**, or use the middle mouse button, if you have one. Add a link between nodes by dragging from the first node to the second. Note that if either the first or the second node is **selected** then separate links are created to or from *every* node in the node selection (see Fig **3.8**).

To **delete** a link, right-click on the link and choose Delete Link from the **pop-up menu**.

#### 3.3.5.14 Moving Nodes - Canvas Layout

Ensure that Move Nodes checkbox is **on** or use the middle mouse button. Click and drag on a node to move it and all its attached links. To move a node selection click and drag on any node in the selection.

Network layout can be made easier by using the Snap Settings... options in the **Settings** menu. This creates an invisible grid on the canvas of required dimension. When this is invoked, any node that is moved will be locked to the nearest intersection point of the grid (see section **3.3.5.17**).

Finally entire networks may be automatically arranged in an orderly fashion by invoking the **Auto Arrange** facility from the **Networks** menu.

#### 3.3.5.15 The Edit Menu

The **Edit** menu provides a number of editing functions not provided by mouse operation.

| | |
|---|---|
| **Undo/Redo** | allows the user to undo and subsequently redo the previous network operation. Multiple calls to Undo progressively remove individual network operations, up to a maximum set by the *History Level* configuration variable. This defaults to 20 operations, but may be configured as desired (see Appendix **A**). |
| **Select All** | selects the entire network. |
| **Invert Selection** | makes the node selection consist of only those nodes which are currently not selected. |
| **Copy Selection** | copies the node selection into a clipboard. |
| **Cut Selection** | copies the node selection into a clipboard and removes the selection from the canvas. |
| **Paste Clipboard** | pastes the node selection in the clipboard into the canvas at the top left hand corner. Nodes may be copied between networks using cut and paste via the clipboard. |
| **Delete Selection** | removes the selection from the canvas. |

#### 3.3.5.16 Zoom Settings

***grapHvite*** networks may be displayed 2 and 3 times smaller than their normal size. Choosing the Zoom Settings option from the **Settings** menu displays the window shown in figure 3.9 below.

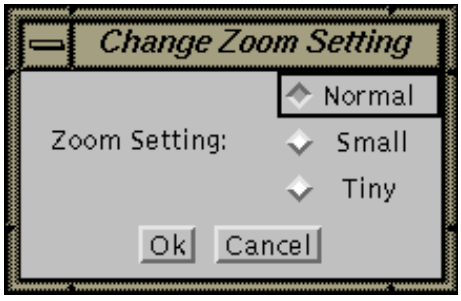


**Fig. 3.9 Zoom Settings Window**

This is very useful for viewing large networks. Normal edit operations can be performed on zoomed networks, but selecting the Tiny option means that node names will not be displayed.

#### 3.3.5.17 Snap Settings

This displays the window shown in figure 3.10 below.

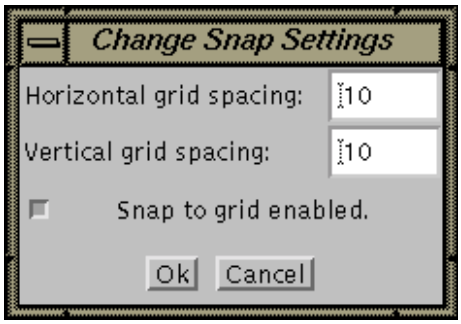


**Fig. 3.10 Snap Settings Window**

This defines an invisible grid on the canvas, with dimensionality given by the Horizontal and Vertical grid spacing fields. When **Snap to Grid** is enabled, any node that is moved on the canvas will be locked to the nearest intersection of the grid. This enables nodes to be lined up exactly horizontally/vertically or in some well-defined configuration.

The default grid spacing is 10 units in both horizontal and vertical directions. As a guide to scale, the diameter of a null node is 25 units. A snap setting of 1 unit allows a node to move freely in that direction.

It is sometimes possible to position nodes on top of each other while using snap settings. If this has happened, click and drag will move the top node away and display the others underneath.

#### 3.3.5.18 Node Properties

To display the properties of a node or a sub-network right click on the node (or sub-network) and choose Properties from the **pop-up menu**. This displays the window shown in figure 3.11 below.


**Fig. 3.11 Node Properties**

The number of the node is displayed along with an editable text field marked Extra Information. This field is normally blank, but if the user types information into it this will be saved and displayed when Node Properties is next used.

#### 3.3.5.19 Link Properties

The link properties can be displayed by moving the mouse over a link and selecting Properties from the pop-up menu (activated by right-clicking over the link). This displays a simple dialog an example of which can be seen in figure 3.12 containing information about such as the link number and the number and node type of the adjoining nodes.

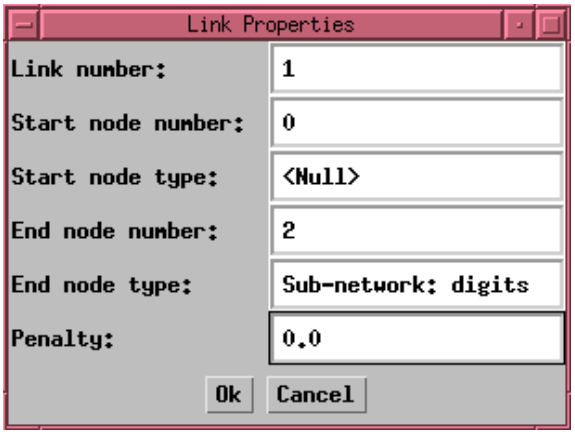


**Fig. 3.12 Link Properties**

The link penalty can be set via the "Penalty:" field. The default penalty value is 0.0 which represents no penalty and increasingly large negative values represent bigger penalties. A value of -1.0E6 would typically disable a link (assuming all other links have no penalty assigned). To quickly view a penalty value, simply move the mouse over the link and observe the value in the status bar (situated underneath the canvas).

Links with non-default penalty values are, by default, highlighted (this can be disabled using a configuration setting) which often helps when interactively testing a recogniser as non-default link penalties may give rise to otherwise unexpected recogniser behaviour.

Link penalties can be used to weight network paths which are known to be less likely than others for a recognition task - assuming such task statistics are available. For example, a link feeding into a sub-network of unusual entries may have a penalty value which biases the recogniser more in favour of a competing sub-network of more common entries. Link penalties can also be used as a convenient way to weight certain types of model. For example, they are essential to the use of garbage models in ***grapHvite***, as discussed in section 3.5.6.2. Another possible use is to apply an "insertion" penalty to the loop-back link of a word loop in cases where the recogniser tends to insert extra words.

Note that link penalty settings are preserved when a network file is exported to a HAPI application.

#### 3.3.5.20 Network Properties

To display the properties of a network right click on the canvas and choose Properties from the **pop-up menu**. This displays the window shown in figure 3.13 below.

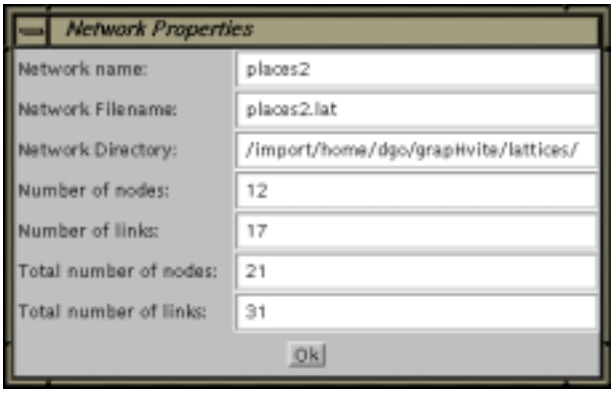


**Fig. 3.13 Network Properties**

The first three fields show the name of the network, its file name on disk and the directory on disk in which it is stored. The next two fields display the number of nodes and links contained in the visible network on the screen and the final two fields show the same information for the whole network including all sub-networks.

### 3.3.6 Testing and Verifying Syntax Networks

The **Networks** menu provides options which allow the user to verify the *correctness* of the network and ensure that the legal paths through the network are as intended.

A network is correct if:

- There is only one **start node**. A start node is a node which only has links going out of it.
- There is only one **end node**. An end node is a node which only has links going into it.
- There are no unconnected nodes.

- There are no loops of null nodes. The user is normally prevented from doing this in the editing stage.

#### 3.3.6.1 Verifying the Network

Select Verify Network from the **Networks** menu to verify the current network displayed on the canvas. If any problems are found the culprit nodes are highlighted and a window explaining the problem is displayed.

Verify Network only checks the the visible network displayed on the screen and does not recursively check any sub-networks.

The random sentence generator in Test Sentences (see section **3.3.6.2**) is a convenient way of testing the whole network.

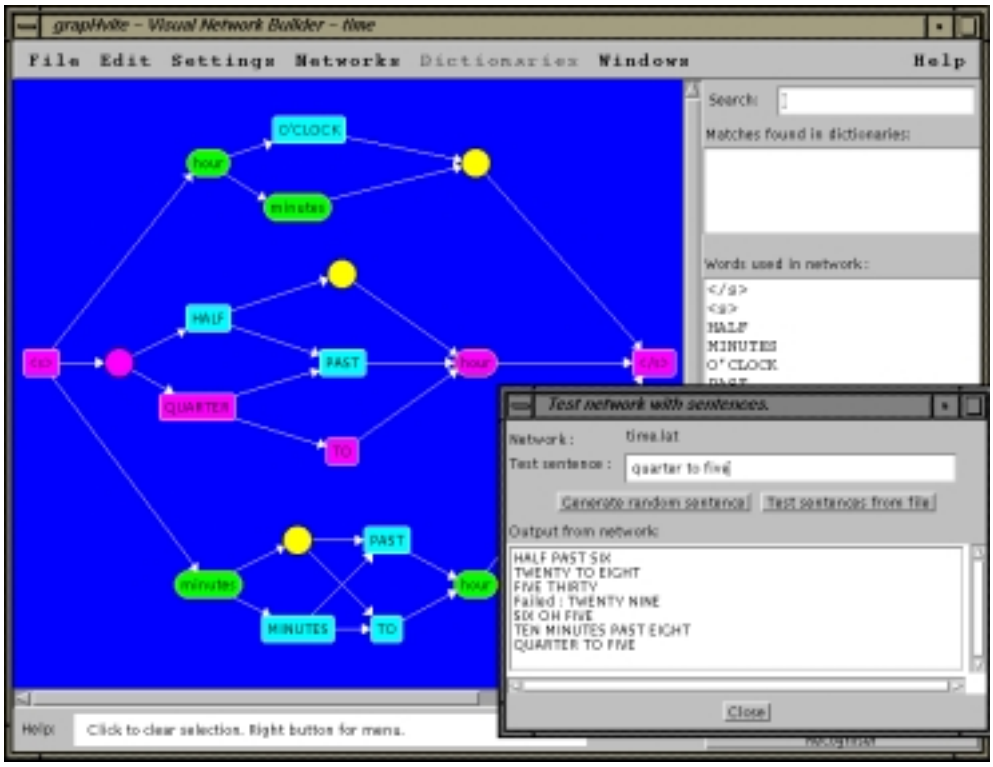


**Fig. 3.14 Test Sentence Output**

#### 3.3.6.2 Generating and Testing Sentences

Use Test Sentences in the **Networks** menu to generate random text sentences from the network and see if test sentences are accepted by the network. The Test Sentences window is displayed in figure 3.14.

Generate random sentence can be repeatedly pressed to generate examples of legal sentences by randomly traversing the nodes and sub-networks in the network. Example sentences are shown in the display area Output from Network and for each sentence

generated the nodes traversed through the network are highlighted. If highlighted sub-networks are swapped into the canvas (see section **3.3.5.8**) the individual path traversed through the sub-network also appears highlighted.

To test to see if a single sentence or phrase is accepted by the network, type it into the Test sentence box and press return. Note that in figure 3.14 `Twenty nine` has been rejected by the network and is marked `Failed`.

To test an entire file of user generated test sentences, press the button Test sentences from file. This displays the **File Browser** (see section **3.2.1**) window from which the file of test sentences can be selected. The **File Browser** looks in the directory *./sentences* for the file (this is configurable - see Appendix **A**). Sentences must be prepared one per line, and any that are rejected by the network are marked `Failed` in the display area Output from network.

## 3.4 Dictionary Operations

The **dictionary panel** on the right-hand side of the front panel provides access to ***grapHvite*** dictionaries. Dictionaries are used to provide a phonetic pronunciation for words used in **word nodes**, and to define *output symbols*. An output symbol is the literal string that is output by the recogniser when a particular word is recognised (e.g. possibly "13" for word "thirteen"). The ***grapHvite*** Netbuilder uses two types of dictionary:

- The **System dictionary**

  This is loaded when the ***grapHvite*** Netbuilder starts up and will depend on what language is to be used and whether a *full* (comprehensive vocabulary, but large and slower to load) or *core* (a subset of the most commonly used words) dictionary has been selected.

  System dictionaries supplied with this version of ***grapHvite*** are documented in Appendix **B**, and the ***grapHvite*** Netbuilder can be configured to load whichever dictionary is required at the start of each session (see Appendix **A** - *Configuring* ***grapHvite***).

  The system dictionary may not be modified.

- **User Dictionaries**

  User dictionaries are created by the user either from scratch or from subsets of words used in the System Dictionary. New words can be added, and the pronunciation and output symbols of existing entries modified.

### 3.4.1 Looking a word up

Click on the Search field of the dictionary panel and type the word into it. As you type, all currently loaded dictionaries are searched for the characters you have already entered, and matches found are displayed in Matches found in dictionary. The highlighted word (e.g., *pipe* in figure 3.15) is the word currently selected and will be the word associated with any word nodes inserted in the network.

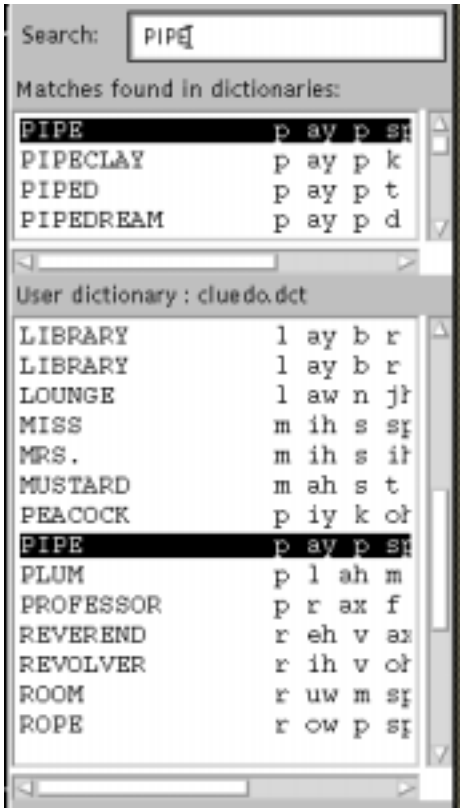


**Fig. 3.15 The Dictionary Panel**

If a word is not available in the dictionary, it may be that you are using a core vocabulary for the language. Reset the *System Dictionary* configuration setting (see Appendix **A**) to **full** and restart the ***grapHvite*** Netbuilder to load the full System Dictionary.

### 3.4.2 Word Pronunciation

The pronunciation of words used in word nodes is recorded using one of the phone sets tabulated in Appendix **B**. Several pronunciations may be recognised for one word, and these are displayed in the "/" separated list in the Matches and User Dictionary fields.

As explained in Appendix **B**, Each *word* pronunciation in the dictionary should terminated by the short pause symbol *sp* to allow for optional short pauses between words.

### 3.4.3 Alternative Output Symbols

When a word is recognised, the recogniser outputs an *output symbol* specific to that word. System dictionary entries always output the same symbol as displayed by the word node (except for word nodes representing pauses - see Appendix **B**), but user dictionaries may define an alternative symbol (e.g., `4` for `FOUR`). If the output symbol differs from the entry name, it is displayed in square brackets alongside the entry in the User Dictionary panel (see Fig **3.16**).


**Fig. 3.16 Dictionary entry for *four* with *4* as output symbol**

### 3.4.4 User Dictionary Files

The default directory and extension for dictionary files is *./dicts/* and *.dct* respectively. These are both configurable and governed by the *Dictionary Directory* and *Dictionary Suffix* settings (see Appendix **A** *Configuring* ***grapHvite*** *Netbuilder*).

### 3.4.5 Creating User Dictionaries

To create a new user dictionary from scratch, select the New Dictionary option from the **File** menu. The new dictionary has the default name *untitled.dct* (you can save this dictionary to a new filename by selecting the Save Dictionary As... option from the **File** menu - see section **3.4.7**). The dictionary is now ready for use.

New task-specific dictionaries may also be created from the subset of words that have been used in network by selecting Save used words as.... from the **File** menu. The **File Browser** allows you to select a directory and file name to which the dictionary entries for all the words used in the current network and its sub-networks will be saved as a new dictionary.

### 3.4.6 Loading User Dictionaries

To load an existing user dictionary select the Load Dictionary option from the **File** menu. This displays a **File Browser** window (see section **3.2.1**) from which the desired user dictionary file can be selected. The current working dictionary is now an amalgamation of the system and user dictionaries.

The ***grapHvite*** Netbuilder supports the loading and use of multiple user dictionaries for the same network. Currently loaded dictionaries are listed at the bottom of the **Dictionaries** menu and may be individually selected for display in the dictionary display panel.

If there are duplicate word entries in the currently loaded set of dictionaries with different word pronunciations, the differences are preserved as multiple pronunciations of the same word. User dictionaries can therefore be used to augment the system dictionary with new words or with alternative pronunciations for existing words.

If duplicate entries differ in their *output symbols* alone then the system dictionary default output symbol for that entry will be overridden without warning. Any subsequent attempts to change this output symbol (which is now no longer the system default) will result in the user being asked for confirmation of the change.

### 3.4.7 Saving User Dictionaries

To save the currently selected user dictionary choose the option Save Dictionary As... from the **File** menu. This invokes a **File Browser** window (see section **3.2.1**) which allows the user to define the directory and filename to which the dictionary is saved.

You will also be given the option to save any modified loaded dictionaries when the ***grapHvite*** Netbuilder is shutdown.

### 3.4.8 Editing User Dictionaries

The **Dictionaries** menu allows you to add, edit and delete words in the currently selected user dictionary. If several user dictionaries are loaded, the *current* dictionary is the one selected from the list of user dictionaries at the bottom of the **Dictionaries** menu. Note that editing of the system dictionary is not permitted.

#### 3.4.8.1 Adding New Word Entries

To add a completely new word entry to the currently selected user dictionary, or to add an alternative pronunciation for an existing word, select the option Add New Word... from the **Dictionaries** menu. This displays the window shown in figure 3.17.

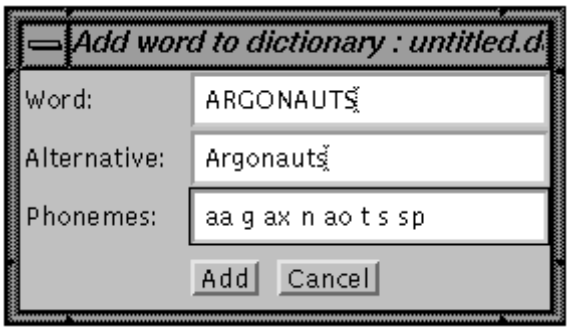


**Fig. 3.17 Adding Words to User Dictionary**

- Type the word to be added into the Word field (in this case, `ARGONAUTS`).

- Type the *output symbol* (the string to be output when this word is recognised - see section **3.4.3**) into the Alternative field . This will normally be the same as the Word field, in which case you should enter the word again here. *If left blank, the recogniser will output* nothing *when the word is recognised.*

- Type the blank-separated string of phones which define the pronunciation into the Phonemes field (see section **3.4.2**), ending with the short pause symbol *sp* to allow an optional pause after the word.

  An explanatory listing of the relevant phones for the supplied language is given in Appendix B. Currently, there is no on-line listing of legal phones, but you can always look up a word in the system dictionary with a similar pronunciation. Illegal phones will be flagged by a warning window and may then be corrected.

- Click the Add button. Your new word will appear in the User Dictionary entries in the dictionary panel, and the **Add Word** window will be blank again, ready for you to enter another word.

- Press Cancel to abort the entry you are editing, or on a blank entry to terminate the list of new words.

#### 3.4.8.2 Deleting Word Entries

To delete a word entry from the currently selected user dictionary, select the word by either left clicking on the word in the scrollable dictionary display areas or by typing the word into the Search field in the main window. Now choose the option Delete Word... from the **Dictionaries** menu. The word is removed from the user dictionary without any warning request for confirmation. Note that the Undo/Redo facilities available under the **Edit** menu apply only to graphical edits made to networks on the canvas and have no effect on dictionary edit operations.

#### 3.4.8.3 Editing Word Entries

To edit a word entry from the currently selected user dictionary, select the word by either left clicking on the word in the scrollable dictionary display areas or by typing the word into the editable text field Search in the main window. Now choose the option Edit Word Entry from the **Dictionaries** menu. This makes the window shown in figure 3.18 below.

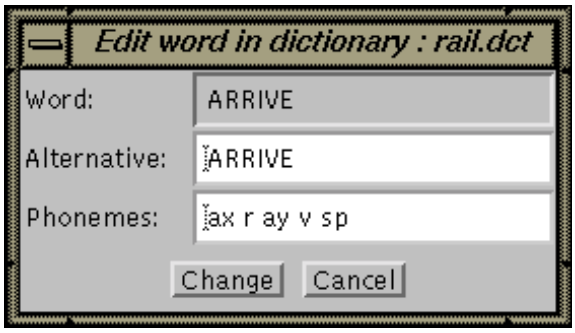


**Fig. 3.18 Editing Words in the User Dictionary**

As in the Add word window (see section **3.4.8.1**), Alternative contains the current output symbol (see section **3.4.3**) and Phonemes the phonetic pronunciation (see Appendix **B**). Press Change when you have finished editing these fields.

## 3.5 The Graphical Recogniser Interface

Networks may be tested immediately using the graphical interface to the continuous speech, speaker independent speech recogniser that is supplied with ***grapHvite***. This is started up by pressing the Recogniser button in the lower right hand corner of the front panel (see Fig **3.1**), or by selecting Start Recogniser... from the **Networks** menu.

The first time the recogniser is used during a ***grapHvite*** Netbuilder session, it must be:

**Configured** The recogniser needs to know what sort of speech models to use (see Appendix A) and what the test speech source type is.

**Loaded** The recogniser is started up with the selected configuration.

**Calibrated** If the recogniser is configured to accept live audio, it must be calibrated against current speech levels.

After this has been done, users may move freely between editing networks and testing with live speech.

### 3.5.1 A Tour of the Main Recogniser Window

The main recogniser window consists of a **File** and **Config** menu at the top, four panels ( Network , Controls , Output and Current ) covering the greater part of the window, and a status area at the bottom.

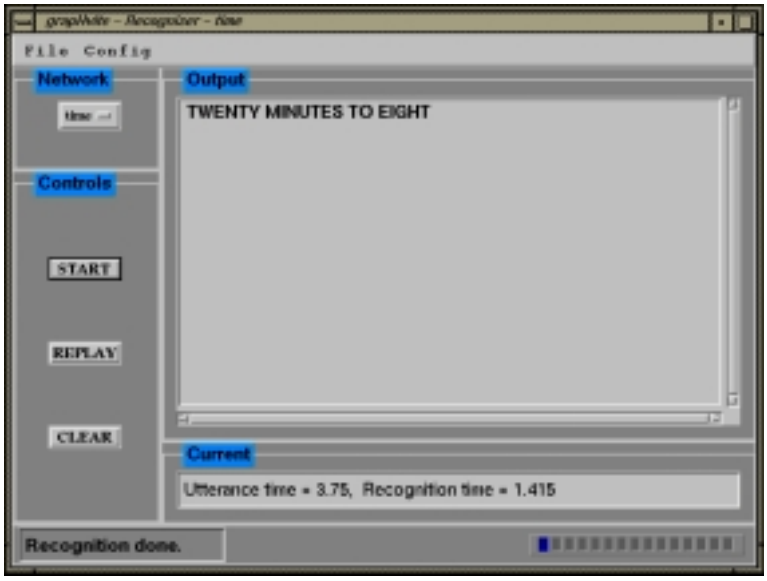


**Fig. 3.19 The Recogniser Interface**

**File** Currently, the **File** menu consists of the single option Quit to exit the recogniser.

**Config** The configuration menu has three options: Configure Recogniser , Load Recogniser and Calibrate Recogniser . Each of these options is explained in detail in later sections.

Network The Network button in the top left hand corner lists all currently loaded networks and swaps them in and out of the recogniser. In figure 3.19 it is set to the *time* network.

Controls The controls panel has three buttons, START , REPLAY and CLEAR . START may also become STOP (when the recogniser is recognising) or UPDATE (when recognition constraints are changed).

**START** Start recognition. Press START when you want to begin speaking or play recorded speech into the recogniser. The recogniser will start listening for speech and begin processing it. During this phase, START will be replaced by STOP .

**STOP** Stop recognition. Under normal circumstances the recogniser will stop when the input speech ends (that is, the recogniser detects a silence), but pressing STOP causes it to terminate immediately.

**UPDATE** This button is displayed when one or more of the recogniser's components (Network, model set etc) has been modified. Press UPDATE to update the recogniser. This button then reverts back to START.

**REPLAY** Replay the last utterance recognised.

**CLEAR** Clear the Output and Current panel.

Output The Output panel is a scrollable display area in which the final recognition results for each utterance are displayed. All text output is preserved in this display area until the CLEAR button on the controls panel is pressed.

Current The Current panel is a fixed size display area into which the intermediate recognition results (results so far) for each utterance are displayed. At the end of each utterance this area is automatically cleared of intermediate results and instead displays the length of the utterance in seconds and the time taken to recognise the utterance, also in seconds.

Status The status area is located along the lower edge of the recogniser window and consists of a text display area to the left and a progress bar to the right. The display area is used to display status messages (for example, `Recognising...`, `Recognition done...` etc.) to indicate the current state of the recogniser. The progress bar depicts two separate quantities depending on the context. When time consuming loading of recogniser components is underway, the progress bar displays a series of red squares continually filling the bar from left to right. During recognition, the progress bar acts as a volume meter and is filled with a number of blue squares proportional to the current volume of the input signal.

### 3.5.2 Configuring the Recogniser

Configuring the recogniser involves choosing the source of speech input and the type of acoustic models (HMMs) that will be used for recognition (see Appendix A).

Selecting the option Configure Recogniser from the **Config** menu starts up the window shown in Figure 3.20 below.

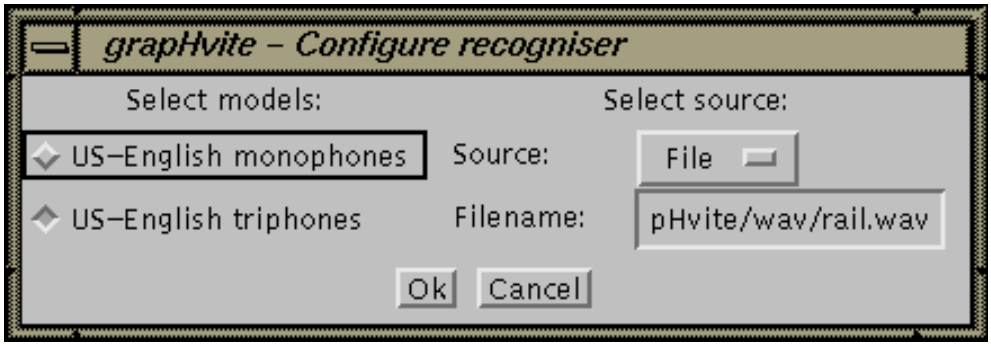


**Fig. 3.20 Recogniser Configuration Window**

Select Models, to the left of the window, contains two option checkboxes which allow you to select the acoustic models for recognition. Select Source, on the right hand side, is used to define the source of audio data for the recogniser.

#### 3.5.2.1 Models Selection

The recogniser uses acoustic models which have been trained on speech from a wide variety of native speakers of the language for which ***grapHvite*** has been configured. For each language, two sets of models (monophones or triphones) are available and the check boxes in the Select models area allows selection between these two sets (see Appendix A). Essentially, monophones occupy less memory but are less accurate than triphone models.

#### 3.5.2.2 Source Selection

The menu option button marked Source provides the user with the options Mic, Line and File.

Mic Audio input is via the computer's microphone socket.

Line Audio input is via the computer's line-in socket. This option is normally chosen when using a wide-band microphone with a pre-amplifier.

File Audio input is taken from a pre-recorded waveform file. Selecting this option invokes a **File Browser** (see section **3.2.1**). By default the **File Browser** expects to find waveform files in the directory *./waveforms* and will add a *.wav* extension if none is given. These defaults are configurable (See Appendix **A**).

The source of audio input can also be changed mid-session. In this case, the UPDATE button will be displayed instead of START, because the recogniser will need to be updated to reflect this change.

### 3.5.3 Loading the Recogniser

After the recogniser is configured, the message `Not Loaded` is displayed in the status area. Before the recogniser can be used it needs to be constructed from the various components: the acoustic models, the dictionary and the recognition network. This

task is performed by selecting the Load Recogniser option from the **Config** menu. The time taken for loading depends on the size and complexity of the components. Progress during loading is indicated by the red dots on the progress bar in the status area.

Loading is complete when the message `Loading Done` is displayed in the status area and the moving red progress dot stops and changes colour to blue.

### 3.5.4 Calibrating the Recogniser

Although the recogniser itself is speaker independent and able to cope with a variety of background and speaking levels, it depends upon a silence detector to automatically determine when to stop processing live speech input, and hence when to stop the recognition process. For each new session this silence detector needs to measure the ambient background noise level in order to function correctly. As well as initialising the silence detector, the calibration phase is also useful for setting up input levels and amplifier gains.

Selecting Calibrate Recogniser from the **Config** menu starts up the window shown in figure 3.21.


**Fig. 3.21 Calibrating the Recogniser**

#### 3.5.4.1 The Calibration Window

This consists of a Calibrate speech button, three display areas: Silence level, Peak Speech and Mean Speech, into which feedback regarding the levels of the input audio signal will be displayed, and a Done button, which should be pressed when satisfactory calibration has been completed.

#### 3.5.4.2 The Calibration Process

Press the Calibrate button and speak a sentence into the microphone. The example sentence `Short term treasury bills fell slightly` displayed below the button is only a suggestion. Calibration is complete when numbers are displayed in the three display areas.

#### 3.5.4.3 Interpreting the Audio Levels Feedback

During the calibration process measurements are made of the various parameters of the input signal and the results are displayed in the Silence level, Peak speech and Mean speech fields.
The definition of the displayed values are as follows:

| | |
|---|---|
| Silence level | Ambient volume level of the periods of silence, measured in dB (decibels) |
| Peak Speech | The peak-to-peak speech volume level measured as a percentage of the highest volume within the portions of speech. |
| Mean Speech | The average volume within the portions of speech, measured in dB (decibels). |

These levels should be checked to ensure that the audio system has been configured correctly and that the user is speaking at an appropriate volume. Below are some guidelines to ensure optimum performance.

i The silence levels should be between 15-30dB.

ii A peak-to-peak value should be between 25-75% (to ensure that the signal is not clipped).

iii The mean speech level should significantly higher (ideally 35-40dB) than the silence level.

If the mean speech level is close to the silence level then check that the microphone is connected correctly or that the microphone pre-amplifier (if any) is switched on.

### 3.5.5 Using the Recogniser

Once the silence detector has been calibrated and input levels checked, the recogniser is ready for use.

The **Controls** panel contains three mouse operated buttons controlling the recogniser.

START — Start recognition.

Press the START button to activate the recogniser and begin speaking or play a selected audio file to the recogniser. The message `Recognising` appears in the status display area at the bottom of the window and the recogniser waits for the user to begin speaking. When speech has been detected the volume meter (progress bar) in the status area will indicate the current speech volume level whilst the Current panel will show the current recognition hypothesis so far.

The button will change to STOP and if pressed will cause the recognition processing to stop immediately. Otherwise the recogniser will continue processing the speech until the silence detector determines that the utterance is finished (after 1 second of silence). The volume meter will drop to its minimum level and the recogniser will finish processing the utterance.

The sentence which best matches the input is displayed in the Output panel, and the length of the utterance and the time taken to recognise it are shown in the Current panel (see Fig **3.19**).

REPLAY — Replay the last recorded/recognised utterance.

Click this button to hear that part of the last utterance that the recogniser matched. This is a useful check to see that the silence detector did not cut off the start or end of the utterance.

CLEAR — Clear **Current** and **Output** panels for new speaker session.

Clicking this button clears all text from the **Output** and **Current** panels.

### 3.5.6 Rejecting Out-of-Vocabulary Speech

Two mechanisms for out-of-vocabulary (OOV) rejection are now supported by the ***grapHvite*** Netbuilder. The first is based on the HAPI confidence scoring mechanism, which has been made available with this release of ***grapHvite***. The second form uses a garbage model which can be inserted into a network at suitable points in order to detect OOV speech at those points in the spoken input.

#### 3.5.6.1 Using HAPI confidence scores

A simple form of OOV rejection, based on the confidence score returned from the HAPI results object, can be enabled by selecting Settings... from the **Rejection** menu. This displays a dialog similar to that shown in figure 3.22.

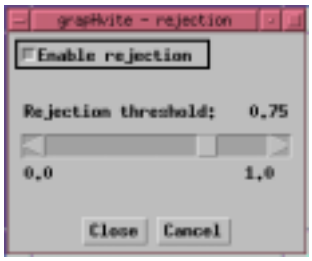


**Fig. 3.22 Setting the rejection threshold**

When Enable rejection is set, the confidence score for each recognised word is compared with the threshold setting and words with a confidence score below this threshold are rejected. Confidence scores range in value between 0 and 1, and the threshold setting can be adjusted along this range, so that, with a threshold setting of 1 all words are rejected, while a setting of 0 triggers no rejects. The threshold

can be adjusted in between recognitions (i.e. the rejection dialog remains open as desired) until a suitable value is found for the given recognition task. Default settings of the Enable rejection option and threshold value can be specified in the ***grapHvite*** Netbuilder configuration.

Words rejected by this means are displayed in red in the recognition output window. Note that rejection by this means is performed on *all* words irrespective of context. For some applications, detection of OOV words is only of interest at certain critical points in the network, for instance where a name is specified, and the application would typically compare the HAPI confidence score with the pre-set threshold only for the relevant words returned by the HAPI results object.

The HAPI confidence scores should be used with care under the following two conditions which can sometimes result in an unreliable confidence score estimate.

1. Very short words. In this case the confidence score is estimated over a short duration and sometimes this results in a low confidence score value when the word is correctly recognised.

2. Very few competing words. When there are very few (i.e. one or two) alternative words being evaluated at a point during recognition – either because the network has a very low branching factor at that point, or because very tight beam pruning is being applied during recognition – then the confidence score estimate may be misleadingly high (i.e. approaching 1).

#### 3.5.6.2 Using a garbage model

An alternative form of OOV rejection can be performed by inserting a garbage model at an appropriate point in a network. Clicking Add garbage model in the "Add Sentences" dialog (described earlier in section 3.3.5.10) inserts an orange-coloured 'garbage' node in between the specified network start and end nodes. A garbage model is typically added at specific points in a network, for instance in a sub-network of names, in parallel to the desired vocabulary nodes, to provide an alternative recognition path for OOV occurrences. During recognition, for portions of speech whose best match is to the garbage model, the garbage node output symbol (by default "<GARBAGE>") is written to the recognition window.

The garbage model typically tends to match better to valid speech than the competing in-vocabulary paths and may need to be weighted to find the right balance between OOV accepts (garbage model does not best match OOV speech) and in-vocabulary rejects (garbage model matches best to valid speech). A simple and effective way to do this is by assigning a penalty to the link into the garbage model. Selecting View Sub-network from the pop-up menu displays the actual garbage model and, by default, the entry link and loop-back link have associated penalties and appear highlighted. Setting the entry link to a suitable negative value reduces the garbage model tendency to "false trigger" on valid speech. Setting a penalty on the loop-back link allows further fine tuning for cases where the garbage model tends to be re-entered too often. For some forms of garbage model the loop-back may be implemented in the model itself and the loop-back penalty can be set to a large negative value to disable this link. Default settings of both these link penalties can be specified in the ***grapHvite*** Netbuilder configuration.

To use a non-supplied garbage model, for example one built by the developer on task-specific data, simply set the configuration variable "GbgeModelName" in the

***grapHvite*** Netbuilder to the actual name of the garbage model as it appears in the model set and model list files, for example,

GbgeModelName = garbage

if a model of the name "garbage" has been trained. In order for this model to be accessed within the ***grapHvite*** Netbuilder the dictionary loaded at start-up must be a user-created dictionary containing a phone of that name. This can be done by adding the phone-name "garbage" to the phone list entry appearing at the top of any saved ***grapHvite*** Netbuilder dictionaries. Alternatively, a dummy word entry with that phone as its pronunciation can be added to the dictionary.

# Appendix A

# Configuring the *grapHvite* Netbuilder

## A.1 The *grapHvite* Netbuilder configuration file

The operation of the ***grapHvite*** Netbuilder is controlled by a set of parameters stored in the configuration file *grapHvite.cfg*. At startup, the ***grapHvite*** Netbuilder searches for a file with this name in the following directory order.

Current directory

User's home directory

Directory from which ***grapHvite*** Netbuilder was invoked (installation directory)

The first occurence of the file *grapHvite.cfg* is read to determine the configuration for the ***grapHvite*** Netbuilder session.

The configuration file consists of entries, one per line, each defining a configuration parameter. In addition, comments can be placed anywhere in the configuration file and are preceded by the character '#'. Any text following the comment specifier on a given line is ignored. Blank lines are also ignored.

### A.1.1 General configuration parameters

General parameters, which take effect regardless of what model sets and dictionaries are loaded, are defined on separate lines using the format

Keyword = value

where *Keyword* is a configuration parameter and *value* is its associated setting. Many of the following general parameters are directory path names and these should end with a '/' e.g.

SomePathnameParameter = /usr/local/grapHvite/somedirectory/

Directory pathnames can be prefixed with '!' which is internally expanded to the directory pathname from which the ***grapHvite*** Netbuilder was invoked. For example,

AnotherPathname = !dicts/

becomes internally expanded to /usr/local/grapHvite/dicts/ if the ***grapHvite*** Netbuilder was invoked from /usr/local/grapHvite.

The following table lists the general parameters: the characters '.' and '!' in default pathnames represent current and ***grapHvite*** Netbuilder-invocation directories respectively. Parameters with a 'Y' entry in the "Sys" column can be overriden for a particular system (see Sub-section A.1.2 below).

| Keyword | Sys | Default | Description |
|---|---|---|---|
| DictionaryDir | Y | ./dicts/ | Default user dictionary directory. |
| WaveformsDir | Y | !/wav/ | Default speech waveform file directory. |
| SentencesDir | Y | !/sentences/ | Default test sentence file directory. |
| SentencesSuf | N | .snt | Default suffix for sentence files. |
| NetworkDir | Y | ./lattices/ | Default network file directory. |
| SubnetPath | Y | ./lattices/ | Colon-separated list of directories to search for sub-networks when the SUBREF FILE field in a network file becomes outdated. |
| SystemType | N | UK_ENGLISH | Selects a system for the ***grapHvite*** Netbuilder session. This can be assigned to a valid SYSID label or the value of a 'System' parameter (see Sub-section A.1.2). |
| HistoryLevel | N | 20 | Maximum number of network editing 'undo' operations. |
| DictionarySuf | N | .dct | Default user dictionary file suffix. |
| NetworkSuf | N | .lat | Default network file suffix. |
| HTKNetSuf | N | .htk | Default HAPI format network file suffix. |
| AllowMixedCase | Y | FALSE | Enable/disable mixed case dictionary entries. |
| ShowLinkPenalties | N | TRUE | Highlight links that have non-default penalty values. |
| NumSentences | N | 1 | Default number of random sentences generated by "Test Sentences" dialog. |
| DefAudioSource | Y | mic | Default audio source setting when the "Configure Recogniser" dialog first displays. |
| DefAudioFile | Y | | Default filename setting if a file-based source is specified when the "Configure Recogniser" dialog first displays. |
| UseReject | N | FALSE | Default "enable rejection" setting when "Rejection" dialog first displays. |
| RejThresh | N | 0.75 | Default value of rejection threshold when "Rejection" dialog first displays. |
| GbgeModelName | N | | Actual name of garbage model - if none specified the grapHvite garbage model is used. |
| GbgeLoopPenalty | N | -20.0 | Default garbage model loopback penalty. |
| GbgePenalty | N | -50.0 | Default garbage model penalty. |
| UseProgBar | N | TRUE | Enable/disable progress dialog displayed when networks are loaded and saved. |

### A.1.2 System configuration parameters

Configuration parameters specific to a system (e.g. US-English) use the extended format

SYSID: [RESID:] Keyword = value

where the label SYSID specifies the system to which parameter *Keyword* applies. System parameters for resources such as model sets or dictionaries require the additional RESID label to distinguish between multiple resources. For example,

ENG_UK: CORE: Dictionary = coredict
ENG_UK: FULL: Dictionary = fulldict

associates a 'core' and 'full' dictionary with the system labelled "ENG_UK". Additional dictionaries can be added to this system by specifying new RESID labels.

The label fields SYSID and RESID (optionally followed by a colon) can be any string value not conflicting with the configuration parameter names and, of course, not beginning with a '#'. All parameters for a given system must use the same SYSID, and similarly, all parameters for a particular resource within a system must use the same RESID. The RESID labels need only be unique for the different resources within a system and so can be reused for other systems.

The ***grapHvite*** Netbuilder configuration is easily updated to include a new system by assigning a unique SYSID label for the system and adding the new set of system configuration parameters each prefixed with the new SYSID label. Similarly, configuring a system for new resources is easily done by adding the appropriate parameter set each with a new RESID label.

In the following list of system configuration parameters the 'Id' column specifies whether a RESID label is required. In addition to the following list, general parameters with a "Y" entry in the "Sys" column above can also be set to system-specific value which overrides the general setting for that system, for example

ENG_UK: DictionaryDir = !dicts/dicts_uk/

sets the default directory where user-dictionaries for the UK-English system are located.

| Keyword | Id | Default | Description |
|---|---|---|---|
| System | N | UK_ENGLISH | Descriptive name for the system to appear in Netbuilder windows. Either the value of this parameter or its SYSID label can be assigned to 'SystemType'. |
| HapiConfigFile | N | ./hapi.cfg | HAPI configuration file for recogniser operation. |
| SelectModels | N | First RESID for system | Optional. Default model set loaded by the ***grapHvite*** Netbuilder recogniser. |
| ModelsName | Y | SYSID | Optional. Descriptive model set name to appear in Netbuilder dialogs. |
| ModelsFile | Y | None | Name of model set file. |
| ModelsDir | N | None | Models directory for system. Overrides ModelsDir for system if a RESID label is given. |
| ModelList | Y | None | List file for model set. |
| SelectDictionary | N | First RESID for system | Optional. Select system dictionary to load at startup. |
| Dictionary | Y | None | System dictionary filename. |

### A.1.3 Example configuration file

The following is an example configuration file which has system configurations for UK English and US English, with UK English selected.

```
# grapHvite v1.1 configuration file
# Entropic Cambridge Research Laboratory, April 1997
# In the following settings, any occurrence of ! will be replaced
# by the value of the property GVHOME if specified. Note that the
# grapHvite startup script (graphvite) automatically determines
# the installation directory and passes it to the java interpreter

#
# General parameters to select UK-English system and undo stack size
#
SystemType = ENG_01
HistoryLevel = 20

#
# General directory defaults
#
WaveformsDir = /usr/local/grapHvite/waveforms/
SentencesDir = /usr/local/grapHvite/sentences/
SubnetPath = /usr/local/grapHvite/lattices/
NetworkDir = /usr/local/grapHvite/lattices/
DictionaryDir = /usr/local/grapHvite/dicts/

#
```

```
# General filename suffix defaults
#
DictionarySuf=.dct
NetworkSuf=.lat
HTKNetSuf=.htk

#
# System configuration for UK English
#
ENG_01:            System = UK_English
ENG_01:            HapiConfigFile = /usr/local/grapHvite/hapi_UK.cfg
ENG_01:            ModelsDir = /usr/local/grapHvite/models/
ENG_01:            SelectModels = MODS_02
ENG_01: MODS_01:   ModelsName = Monophones-kjp
ENG_01: MODS_01:   ModelsFile = ECRL_UK_MONO.mmf
ENG_01: MODS_01:   ModelList = ECRL_UK_MONO.list
ENG_01: MODS_02:   ModelsName = Triphones
ENG_01: MODS_02:   ModelsFile = ECRL_UK_WTRI.mmf
ENG_01: MODS_02:   ModelList = ECRL_UK_WTRI.list

# Following overrides default ModelsDir for MODS_02
ENG_01: MODS_02: ModelsDir = /usr/local/grapHvite/models/

ENG_01:            SelectDictionary = CORE
ENG_01: CORE:      Dictionary = /usr/local/grapHvite/sys/ECRL_UK_CORE.dict
ENG_01: FULL:      Dictionary = /usr/local/grapHvite/sys/ECRL_UK_FULL.dict
ENG_01:            DictionaryDir = /usr/local/grapHvite/

#
# System 2 - US English
#
ENG_02:            System = US_English
ENG_02:            HapiConfigFile = /usr/local/grapHvite/hapi_US.cfg
ENG_02:            SelectModels = MODS_01
ENG_02: MODS_01:   ModelsName = Monophones
ENG_02: MODS_01:   ModelsFile = ECRL_US_MONO.mmf
ENG_02: MODS_01:   ModelList = ECRL_US_MONO.list
ENG_02: MODS_01:   ModelsDir = /usr/local/grapHvite/models/
ENG_02: MODS_02:   ModelsName = Triphones
ENG_02: MODS_02:   ModelsFile = ECRL_US_WTRI.mmf
ENG_02: MODS_02:   ModelList = ECRL_US_WTRI.list
ENG_02: MODS_02:   ModelsDir = /usr/local/grapHvite/models/
ENG_02:            SelectDictionary = CORE
ENG_02: CORE:      Dictionary = /usr/local/grapHvite/sys/ECRL_US_CORE.dict
ENG_02: FULL:      Dictionary = /usr/local/grapHvite/sys/ECRL_US_FULL.dict
```

## A.2 Configuring the *grapHvite* Netbuilder interactively

Configuration settings may also be modified from within the ***grapHvite*** Netbuilder by selecting the Config Settings option of the **Settings** menu. Figure A.1 shows a typical window.

For general parameter settings, clicking on its entry invokes a simple dialog with an editable textfield in which the parameter value can be updated. An exception to this is item "Show System Configuration:" which displays a simple dialog with a choice menu for the available system configurations. When a new system is selected the displayed system configuration parameters are updated for the new choice of system.

System configuration parameters are displayed for only one system at a time. Similarly, only one model set configuration is displayed for the currently selected system. Clicking on "Show Models Configuration:" displays a simple dialog with a choice list of the available model configurations and selecting a new item from the list updates the displayed model configuration. The desired system dictionary configuration is selectable from a choice list activated by clicking "Show Dictionary Configuration:".

When editing is complete, press Ok to keep settings for the current session only. Press Save Settings to retain for future use. This will cause the ***grapHvite*** Netbuilder to create or over-write *grapHvite.cfg* in the current directory. The next time the ***grapHvite*** Netbuilder is invoked from this directory, it will be configured using this file.

Some settings must be saved to have any effect (for example an updated system dictionary configuration or choice of system) since they are only read by the ***grapHvite*** Netbuilder upon initialisation. A warning box is displayed in these circumstances.

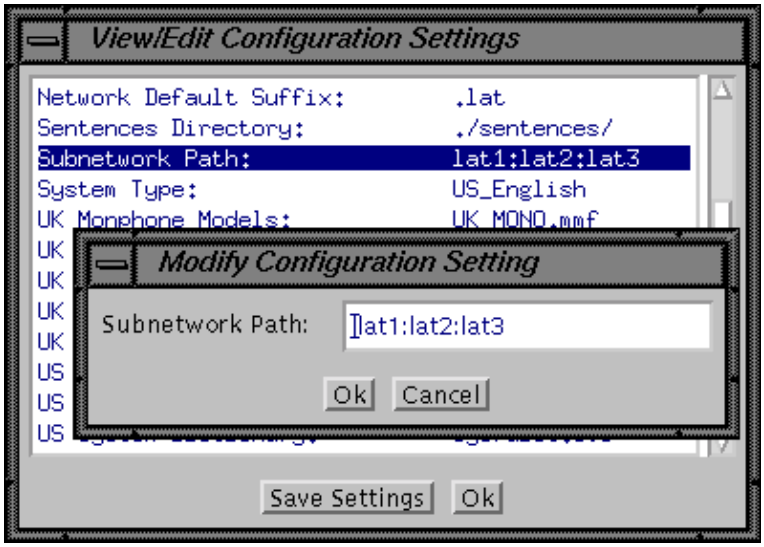


**Fig. A.1 Configuration Settings Menu**

# Appendix B

# Dictionaries

## B.1 Core and Full System Dictionaries

***grapHvite*** is supplied with a variety of systems for different languages or acoustic conditions. The dictionaries and acoustic models for each system are configured as described in section A.1 on configuration. To identify which systems are installed, search for the field **System=** in the grapHvite.cfg file. Before running a new session, the configuration variable **SystemType** can be set to load any of these available systems.

Typically systems are supplied with two sizes of system dictionary - a smaller **core** dictionary which is generally faster to load, and the extended **full** dictionary which covers a much wider vocabulary. ***grapHvite*** provides the facility to switch between supplied dictionaries via the **SelectDictionary** configuration variable.

The following section describes features general to all system dictionaries. Sections B.4 and B.5 describe the dictionary of each language supported by ***grapHvite*** together with their related phone sets.

## B.2 Dictionary Entry Format

Section 3.4 described how ***grapHvite*** dictionary entries map a word to one or multiple pronunciations, which may each display a different output symbol in the recognition window.

By default all word entries in ***grapHvite*** are forced to upper case to make the dictionary search facility more efficient. This can be overridden by setting the configuration parameter **AllowMixedCase** set to TRUE. Note however that the output symbol can always be mixed case, as they treated as a raw strings and are unaffected by the **AllowMixedCase** variable.

During dictionary handling, the output symbol is assumed to be identical to the main word entry unless specified otherwise. An exception to this is the group of reserved special symbols for silence, short pauses and single phones whose output symbols differ from the main entry.

## B.3 Special Symbols

### B.3.1 Silence and Short Pause Symbols

Optional short pauses between words are represented at the end of each dictionary pronunciation by the short pause phone **sp**. Silence, or longer pauses which occur at the beginning and end of complete utterances *must* be explicitly included as network start and and end markers $< s >$ and $< /s >$. These markers should only be used in top level networks. They should not appear at the start and end of sub-networks. If a longer pause is required within a network or sub-network, then the long pause marker $< /p >$ should be used.

The word entries $< s >$, $< /s >$ and $< /p >$ map to silence, and have no output in the results string as the output symbol is set to the empty string. This is illustrated in Table B.1.

| Symbol | Output Symbol | Pronunciation |
|---|---|---|
| $< /p >$ | [ ] | sil |
| $< /s >$ | [ ] | sil |
| $< s >$ | [ ] | sil |

Table B.1: Dictionary entries with null output symbols

### B.3.2 Single Phones

All ***grapHvite*** system dictionaries contain entries prefixed by **P_** which represent a single phone. By searching for P_ the entire phone set can be viewed quickly. Examples of phone entries for **sp b** and **s** are shown below.

| Symbol | Output Symbol | Pronunciation |
|---|---|---|
| P_sp | [sp] | sp |
| P_B | [b] | b |
| P_S | [s] | s |

Table B.2: Single Phone Dictionary Entries

Note that the single phone pronunciations are not terminated by the phone sp, as sp has its own single phone entry.

### B.3.3 Semantic Markers

In some cases semantic markers may be inserted into a network to allow for more efficient parsing of the recognition results string. See the graphvite tutorial Chapter 2 for examples. These markers have an output symbol which appears in the results string of a valid path, but no pronunciation.

## B.4 English System Dictionaries

UK-English and US-English systems come supplied with 90,000 word **full** system dictionaries and 5000 word **core** dictionaries.

| System Key | Core | Full |
|---|---|---|
| US-English | ECRL_US_CORE.dct | ECRL_US_FULL.dct |
| UK-English | ECRL_UK_CORE.dct | ECRL_UK_FULL.dct |

Table B.3: English Dictionary Keys

UK-English and US-English system dictionaries cover the same word entries except for spelling variants. They also differ in pronunciations for certain words. The full UK system dictionary contains approximately 7,000 more pronunciations than the US dictionary, mainly due to multiple pronunciations in UK-English for vowel+r contexts. For example,Scottish speakers or Northern English speakers would pronounce the r in the word *ABOARD*, whereas speakers of Southern English would not pronounce an 'r' in this context.

| UK English | | US English | |
|---|---|---|---|
| Word | Pronunciation | Word | Pronunciation |
| LEVELLED | l eh v ax l d sp | LEVELED | l eh v ax l d sp |
| PROSPERED | p r oh s p er d sp | PROSPERED | p r aa s p er d sp |
| ABOARD | ax b ao r d sp | ABOARD | ax b ao r d sp |
| ABOARD | ax b ao d sp | | |
| DRAUGHTS | d r aa f t s sp | DRAFTS | d r aa f t s sp |
| DRAUGHTS | d r ae f t s sp | | |

Table B.4: Example US and UK Dictionary Entries

Table B.4 shows some example dictionary entries highlighting how UK and US entries may differ in pronunciation, spelling, or both.

### B.4.1 English Phone Sets

Both US-English and UK-English dictionaries use a phone set of 41 distinct speech phones plus the silence phone **sil** and short pause phone **sp**. The US and UK English phone sets are shown below in Tables B.5 and B.6

| Symbol | Example | Symbol | Example |
|---|---|---|---|
| Vowels | | Plosives | |
| aa | balm | b | bet |
| aa | box | d | debt |
| ae | bat | g | get |
| ah | but | k | cat |
| ao | bought | p | pet |
| aw | bout | t | tat |
| ax | about | Fricatives | |
| ay | bite | dh | that |
| eh | bet | th | thin |
| er | bird | f | fan |
| ey | bait | v | van |
| ih | bit | s | sue |
| iy | beet | sh | shoe |
| ow | boat | z | zoo |
| oy | boy | zh | measure |
| uh | book | Affricates | |
| uw | boot | ch | cheap |
| Semi-Vowels | | jh | jeep |
| l | led | Nasals | |
| r | red | m | met |
| w | wed | n | net |
| y | yet | en | button |
| hh | hat | ng | thing |
| | | Silence | |
| | | sil | silence |
| | | sp | short pause |

Table B.5: ***grapHvite*** US Phone Set

| Symbol | Example | Symbol | Example |
|---|---|---|---|
| Vowels | | Plosives | |
| aa | balm | b | bet |
| aa | barn | d | debt |
| ae | bat | g | get |
| ah | but | k | cat |
| ao | bought | p | pet |
| aw | bout | t | tat |
| ax | about | Fricatives | |
| ay | bite | dh | that |
| eh | bet | th | thin |
| er | bird | f | fan |
| ey | bait | v | van |
| ih | bit | s | sue |
| iy | beet | sh | shoe |
| oh | box | z | zoo |
| ow | boat | zh | measure |
| oy | boy | Affricates | |
| uh | book | ch | cheap |
| uw | boot | jh | jeep |
| Semi-Vowels | | Nasals | |
| l | led | m | met |
| r | red | n | net |
| w | wed | ng | thing |
| y | yet | Silence | |
| hh | hat | sil | silence |
| | | sp | short pause |

Table B.6: ***grapHvite*** UK Phone Set

In contrast with the US phone set, the UK dictionary does not use the phone **en**, this is expanded to the phone sequence **ax n**. The UK dictionary uses an additional phone **oh** as found in *hot*. This is distinct from the vowel **aa** in *heart*, whereas the US dictionary makes no distinction, and uses the same vowel **aa** in both cases.

## B.5 Additional Languages

### B.5.1 Accented Characters

System dictionaries for additional languages may be case-sensitive and use extended character sets. ***grapHvite*** is able to read dictionaries containing the full ISO-Latin1 8-bit character set, and also allows a subset of these to be entered into the dictionary search window. The control sequences are shown in Table B.7 .

| Character | Control Sequence | Character | Control Sequence |
|---|---|---|---|
| Á | ALT-' A | Ó | ALT-' O |
| á | ALT-' a | ó | ALT-' o |
| À | ALT-` A | Ò | ALT-` O |
| à | ALT-` a | ò | ALT-` o |
| Â | ALT-^ A | Ô | ALT-^ O |
| â | ALT-^ a | ô | ALT-^ o |
| Ä | ALT-" A | Ö | ALT-" O |
| ä | ALT-" a | ö | ALT-" o |
| Í | ALT-' I | Ú | ALT-' U |
| í | ALT-' i | ú | ALT-' u |
| Ì | ALT-` I | Ù | ALT-` U |
| ì | ALT-` i | ù | ALT-` u |
| Î | ALT-^ I | Û | ALT-^ U |
| î | ALT-^ i | û | ALT-^ u |
| Ï | ALT-" I | Ü | ALT-" U |
| ï | ALT-" i | ü | ALT-" u |
| É | ALT-' E | Ñ | ALT-~ N |
| é | ALT-' e | ñ | ALT-~ n |
| È | ALT-` E | Ç | ALT-, C |
| è | ALT-` e | ç | ALT-, c |
| Ê | ALT-^ E | ¡ | ALT-! |
| ê | ALT-^ e | ¿ | ALT-? |
| Ë | ALT-" E | | |
| ë | ALT-" e | | |

Table B.7: Accented character control sequences

The ALT (or META) key should be pressed simultaneously with the accent, and followed by the character to be accented. Alternatively any of these characters can also be pasted into the dictionary handling windows using the mouse.

### B.5.2 Spanish System Dictionaries

The ***grapHvite*** Spanish system dictionary is supplied with a 90,000 word upper-case full system dictionary and a 5,000 word core dictionary. The Configuration keys are as follows

| System Key | Core | Full |
|---|---|---|
| Spanish | ECRL_SP_CORE.dct | ECRL_SP_FULL.dct |

Table B.8: Spanish Dictionary Keys

### B.5.3 Spanish Phone Set

The Spanish phone set contains 25 distinct speech phones plus the 2 silence phones **sil** and **sp**. Note that the vowel phones **a**,**e**,**i**,**o** and **u** represent both unaccented and accented vowels.

| Symbol | Example | Symbol | Example |
|---|---|---|---|
| Vowels | | Plosives | |
| a | casa | b | boca |
| a | está | d | dolor |
| e | dijé | g | gota |
| i | vino | k | cama |
| o | habló | p | peso |
| u | sucre | t | techo |
| Semi-Vowels | | Fricatives | |
| w | güera | f | fino |
| y | ayer | j | gel |
| l | lago | s | asa |
| ll | llave | z | azote |
| r | aro | Nasals | |
| rr | arroz | m | mano |
| Silence | | n | nombre |
| sil | silence | | año |
| sp | short pause | | |

Table B.9: ***grapHvite*** Spanish Phone Set

### B.5.4 German System Dictionaries

The ***grapHvite*** German system is supplied with a 90,000 word mixed-case full system dictionary and a 10,000 word core dictionary. The Configuration keys are as follows

| System Key | Core | Full |
|---|---|---|
| German | ECRL_G_CORE.dct | ECRL_G_FULL.dct |

Table B.10: German Dictionary Keys

### B.5.5 German Phone Set

| Symbol | Example | Symbol | Example |
|---|---|---|---|
| Vowels | | Plosives | |
| a | Kampf | b | Ball |
| a: | Kahn | d | Deutch |
| eh | wenn | g | gern |
| eh: | Affäre | k | Kind |
| ey | wen | p | Paar |
| oe | Hölle | t | Tafel |
| oe: | Flösse | Fricatives | |
| o | offen | f | fern |
| o: | Ofen | v | wer |
| u | Mutter | s | fassen |
| u: | Mus | z | singen |
| y | System | sh | Stein |
| y: | Kübel | zh | genieren |
| au | Haus | x | Loch |
| ai | weit | x | mich |
| oy | freut | Affricates | |
| ax | mache | jh | Joystick |
| i | Kiste | Nasals | |
| i: | Ziel | m | matt |
| Semi-Vowels | | n | Nest |
| j | Million | ng | lang |
| l | links | Silence | |
| r | rennen | sil | silence |
| h | Hand | sp | short pause |

Table B.11: ***grapHvite*** German Phone Set

# Appendix C

# Standard Lattice Format - SLF

***grapHvite*** networks are stored in plain text using the notation described in this section.

A network consists of a list of definitions defining the sub-networks, nodes and links used in the network. Each one is written on a single line and consists of a number of *name=value* pairs, such as:

W=HALF x=95

All lines which start with # are comments and are disregarded.

Definitions must be given in a fixed order: networks start with a list of definitions of any subsyntaxes they may be using, followed by a *header*, giving the total number of nodes and arcs in the network, and lastly a list of the actual nodes and links in the grammar.

The following extracts from *time.lat* should illustrate the basic format of the network file.

```
#
# Lattice file Time.lat
#
# Sub-network definitions: SUBREF= name (eg. hour),
#                          FILE= filename (eg. hour.lat)
#                          DIR = directory (eg. ./lattices/)
SUBREF=hour FILE=hour.lat DIR=./lattices/

#
# Define size of Network: N = number of nodes, L = number of links.
#
N=19 L=27
#
# List of nodes: I=node number, W=word node, L=subnet node
#                x = x co-ordinate, y = y co-ordinate
#                W=!NULL indicates null node
I=0 W=!NULL x=95 y=255
I=1 W=HALF x=175 y=215
```

```
I=2 W=PAST x=295 y=255
I=3 W=TO x=295 y=335
I=4 L=hour x=415 y=255
I=5 W=!NULL x=295 y=175

# Link definitions: J=arc number, S=start node, E=end node
J=0 S=0 E=1
J=1 S=0 E=18
J=2 S=1 E=2
J=3 S=1 E=5
```

# Index


accented characters
  iso-Latin1, 72
acoustic models, 2
  overview, 10
Add New Word, 24
Adding sentences, 12, 21, 29, 41
applications
  creating, 10, 30
  HAPI/JHAPI, 30
  network file format, 37
Auto-arrange, 43

canvas, 7, 32, **36**
  scrolling, 39
  snap to grid, 44
  switching networks, 37, 40
  zooming out, 43
Clone Sub-network, 40
Coder object, 11
Confidence scoring, 29, 58
Config Menu, 66
configuration, 61
  file, 61
  interactive, 66
Copy Selection, **43**
Cut Selection, **43**

decoder, *see* recogniser
Delete Selection, **43**
Dictionaries menu, **33**
dictionary, 2, 7, **48**
  adding words to, 24, 51
  core system dictionary, 9
  creating, 7, 24, 50
  deleting words from, 52
  editing, 51
  editing words, 52
  files, 33, 50
  for use in applications, 30
  full system dictionary, 9
  null output symbols, 68
  output symbols, 67
  overview, 9
  search field, 32
  short pauses, 68
  system, 9, 48
    accented characters, 72
    core, 48, 69
    full, 48, 69
    UK English, **67**
    US English, **67**
  user, 24, 32, 48
    loading, 50
    saving, **50**
Dictionary object, 11
Digits network, 23

E-mail browser network, 18
Edit menu, **33**
Enquiry network, 5

File Browser, **33**, 36, 37, 48, 50, 55
File menu, **33**
Folder network, 24

Garbage model, 12, 29, 59
Generate random sentences, 25, **47**
grapHvite.cfg, **61**

HAPI Book, *16*
HAPI/JHAPI, 2, 3, 8, 10, 37
  configuring, 30
help
  Help Menu, 33
  help status bar, 32, **39**
hidden Markov model, 2
HMMSet object, 11

Information Window, 35
Invert Selection, **43**

lattice, 11
Link properties, 45

links
creating, 7, 19, **43**
deleting, 19
removing, **43**
long pauses, 68
looking things up, 48

Merging sentences, 21, 41
monophone models, 10
mouse buttons, 7, **37**
check boxes, 38
checkboxes, 32
inserting nodes, 18
left mouse, 38
middle mouse, 39
right mouse, 38
selecting actions, 39
Msg_no Network, 23

Netbuilder, 3, 7–9, 36–47
overview, 7
Network object, 11
networks
correctness, 46
creating, 7, 17, **36**
editing, **36**
example
e-mail browser, 16
examples
Enquiry, 5
Folder, 24
Msg_no, 23
time, 31
Yes/No, 4
exporting to HAPI/JHAPI, 30
files, 33
format, 75
for use in applications, 30, 37
loading, 20, **36**
overview, 3
properties, 46
saving, 19, 37
start and end pauses, 68
testing
using audio input, 9
using text input, 8
testing with audio input, 26
testing with text, 25
used as sub-networks, 36, 68
verifying, 46, 47

Networks menu, **33**
node
selections
cut and paste, 21
Node Properties, 45
nodes
Auto-arrange, **43**
creating, 38
null, 18, **39**
word, 19, **40**
deleting, 19, **42**
deselecting, **42**
inserting, 7
joining, 19, 38, **43**
moving, 19, 38, **43**
properties, 44
selecting, 20, **42**
selections
copying, 43
cutting, 43
deleting, **42**, 43
inverting, 43
joining, 43
moving, 43
pasting, 43
null nodes
creating, **39**
inserting, 18
overview, 4

output symbols, 28, **49**

Paste Clipboard, **43**
Penalties, on links, 12, 41, 45
phone sets used, 24, 70
pop-up menu, 7, 19, **38**
cloning sub-networks, 40
deleting nodes, 42
editing sub-networks, 40
inserting sub-networks, 36, 40
invoking with right mouse button, 38
mouse action checkboxes, 38
removing links, 43
sub-network properties, 36
viewing sub-networks, 37

recogniser, 32, **52**
audio input, 55
calibrating, 26, 53, **56**

CLEAR button, 54, 58
Config Menu, **53**
configuration, 52, 54
configuring, 26
File Menu, **53**
loading, 26, 52, **55**
Network button, **53**
overview, 1
REPLAY button, 54, 58
silence detection, 9, 57
speech models, 55
START button, **53**
starting recognition, 57
STOP button, **54**, 58
testing a network, 9
UPDATE button, 54
recogniser button, 26
Recogniser Object, 11
recognition accuracy, 17
Redo, **43**
Rejection, out-of-vocabulary, 28, 58
Results object, 11

Save Network, 37
Save Network As..., 37
from sub-networks, 37
Save Network in HAPI format, 30, **37**
Search field, **48**
Select All, **43**
Settings Menu, **33**
short pauses, 24, 49, 68
silence nodes, 25, 49, 51, 68
Snap Settings, 43, **44**
Source object, 11
Standard Lattice Format, 75
sub-networks, 22
cloning, 23, **40**
creating, 22
editing, 40
examples
digit, 23
hour, 31
minutes, 31
msg_no, 22
names, 5
inserting sub-network nodes, **40**
loading, 36, 40
nesting, 23
nodes, 4, 6
creating, 7
properties, 46
recursion, 23
saving, 37
search path, 36
use of, 6
using HAPI/JHAPI, 6
viewing, 23
Switching networks, 37

Test Sentences, 25, **47**
Test sentences from file, 48
The HAPI Book, 2, 28
Time network, 31
triphone models, 10

Undo, **43**

Verify Network, 25, **47**

Warning Window, **35**
Windows Menu, 37
Windows menu, 23
bb, 33
word nodes
adding words to dictionary, 24
creating, **40**
inserting, 19
looking up words in dictionary, **48**
overview, 4
pronunciation, 49
word pronunciation, 49

Yes/No network, 4

Zoom Settings, **43**