Chapter 6
Advanced exploration of the clause cube[113]

6.1 Introduction

This chapter focuses on advanced exploration of the Hebrew linguistic data in the clause cube of Genesis 1:1-2:3. After a general discussion of the need for data mining on linguistic data and a short review of the basic concepts underlying this venture, two experiments will be conducted to illustrate the theoretical concepts in practice.

The first experiment will investigate the extraction and analysis of semantic role frameworks via a Visual Basic 6 program from the XML clause cube. The program will extract the semantic frameworks by slicing the relevant information from the threedimensional data cube. Unique semantic frameworks will be identified and their frequencies calculated. The various patterns will then be shown and discussed. Any interesting combinations will be highlighted.

The second experiment will study the mapping of syntactic and semantic functions[114] in the clause cube. After slicing off these two linguistics modules the data will be manipulated to enable the comparison of their usage in the data set. Any unexpected patterns will be discussed.

[113] A shorter and less technical, more humanities-friendly, version of this chapter ("A computer-assisted exploration of the semantic role frameworks in Genesis 1:1-2:3") has been published in Journal of Northwest-Semitic Languages, 2007, vol. 33, no. 1, pp. 55-76 (see Kroeze, 2007c). An earlier version of the article was read as a paper ("Semantic role frameworks extracted form a multidimensional XML database of Gen. 1") at the SASNES 2006 Conference, Unisa, 11-12 September 2006. This chapter has been extended extensively to include programming code and a section on the mapping of syntactic and semantic functions.

[114] "Semantic function" and "semantic role" are synonyms.

The results of this chapter may be used to indicate the value of data-mining ventures for linguistic research in general, and specifically for Biblical Hebrew syntax and semantics, and Functional Grammar.

6.2 The need for advanced, computer-assisted exploration of linguistic data

One of the most frustrating tasks of a linguist is to find ample proof for a hypothesis. For example, it may consume hours, days or even weeks of working through a corpus of texts to find enough (or even one!) examples that illustrate an interesting construction, an unusual combination of semantic roles, or to determine the valency of a verb. Linguists' lives could be a lot easier and they could work much more efficiently if they were able to make use of a proper, extensive electronic database containing raw data on the study material. Electronic data could be exploited to find unknown patterns in order to suggest new hypotheses that can further be researched (cf. Unsworth, 2001). Finding patterns and trends, and testing and suggesting hypotheses are typical data-mining applications, which should not only be restricted to economic and business data, but may also be used to optimise, *inter alia,* linguistic research.

Linguistic databases and mark-up schemes may be regarded as examples of knowledge-representation systems within humanities computing, which may enhance productivity and facilitate knowledge discovery in humanities research (Unsworth, 2001). In addition, a database may capture data from various sources and therefore "the logical statements that would flow from that ontology would necessarily exceed the knowledge of any one individual" (Ramsay, *s.a.*). One should, however, admit that such an ontology as reflected by a mark-up system is not neutral or objective and therefore limits the knowledge construction possibilities to the theoretical paradigm on which the system is based.[115] While mark-up supplied by an original author is constitutive of meaning, mark-up done to pre-existing texts is interpretive because it "... reflects the understanding of the text held by the transcriber; ... the

[115] Compare Du Plooy (1998: 54, 59) for the concept of *knowledge construction*.

mark-up *expresses a claim* about the text .... other interpreters might disagree" (Sperberg-McQueen, 2000: 216).

Petersen (2004b), for example, developed a text database engine, called Emdros, to store and retrieve analyses of any linguistic module, facilitating advanced queries to be executed regarding texts, such as searching for examples of subject inversion and agentless passives or validating rules of agreement between subject and verb in Biblical Hebrew (BH). These types of queries are usually not possible with legacy systems. Such a system "presupposes a database which is tagged with the data necessary for answering the query" (ibid.). As is the case with other data mining tools there should be interaction between the human researcher and the electronic tool since the hits found by such a search should often still be inspected to decide if they are relevant to the tested hypothesis. After all, the use of databases in humanistic sciences rather has to be regarded as a para-interpretive mechanism than a pre-interpretive one (Ramsay, *s.a.*). A multidimensional databank and processing system may enable the researcher to refine his/her queries (the more dimensions the better), thus reducing the amount of human analysis needed. More dimensions enable one to factor in more precise limitations for a specific search.

In the preceding chapters, it was illustrated that an electronic threedimensional data structure could be used to represent inherently multidimensional linguistic data regarding Biblical Hebrew clauses. The Hebrew text of Genesis 1:1-2:3 was used for the experiment. To create the data for the database, the text had to be analysed first. Clause by clause the text of Genesis 1 was phonetically transcribed, translated and analysed on the different linguistic layers of morpho-syntax, syntax and semantics in an interlinear structure. This data was represented in a table format where each table represents a clause and each column a phrase. The various levels of analysis were described in the various rows of the tables. Although this (typically paper-based) databank makes a lot of sense for someone who studies the work in a linear fashion, it does not facilitate advanced research into linguistic structures and other phenomena. Therefore the data was transferred into an electronic databank to make it more accessible and flexible (see Chapter 2). The most important challenge was to merge the data that resided in various, independent and unconnected tables (one for each clause) into one coherent data structure. The concept of the data structure used is a data cube, a threedimensional collection of rows, columns and levels, integrating all the data into one data structure. The rows of the cube represent the various clauses, the columns represent the phrases of each clause, and the depth levels represent the various linguistic modules. A threedimensional array was used in Visual Basic 6 (VB6) to capture the data, which was then programmatically tagged and saved as an XML document (see Chapter 4). The XML document can again be imported in a VB6 program for viewing and other processing. This process of importing and exporting an XML databank is called round-tripping (see Chapter 5).

The creation of the database is equally important to its eventual use because the decision on what to include and exclude and how to visualize the information patterns implies underlying hermeneutics and methodologies that should be brought to light: "As with so many similar activities in digital humanities, the act of creation [of a database - JHK] is often as vital to the experiential meaning of the scholarly endeavour as the use of the final product" (Ramsay, *s.a.*).

A hypertext-based system such as the one proposed in this study, should be regarded as an example of humanities information systems, which facilitate "the advancement of the understanding of human patterns of expression", thus contributing to the goal of the humanities (Aarseth, *s.a.*). However, the relationship between mark-up systems and the humanities is reciprocal. Not only can the humanities benefit from them, but it can and should also contribute to their creation: "Textual scholars should not relate to mark-up technology as passive recipients of products from the computing industry, but rather be actively involved in the development and in setting the agenda, as they possess insight which is essential to a successful shaping of digital text technology" (Huitfeldt, 2004).

The XML data cube described above facilitates the viewing and manipulation of the data to fulfil the needs of linguists and exegetes because the cube can be rotated, sliced and diced to reveal almost any combination of the captured data (see Chapter 3). These actions are similar to data access and manipulation facilitated by data warehousing and online analytical processing. Due to the fixed structure they may also be regarded as a form of data retrieval "where file information is precoded for specific properties and where the conceptual categories for queries must be known in advance" (Lewis & Jones, 1996). Slicing will again be used in this chapter's experiments to retrieve and link the relevant information. Various one- and twodimensional slices will be used to extract and manipulate subsets of the data cube and to capture aggregate (summarised) data. These subsets may be regarded as data marts (cf. Chapter 3). The rest of this chapter will discuss the evaluation of semantic frameworks, as well as the analysis of the mapping of semantic and syntactic functions, as textual data-mining ventures.

6.3 Text mining as a knowledge-invention venture

Once a database of linguistic data has been created, the user is empowered to do searches for specific instances to test a theoretical possibility. XML facilitates complex searches where two or more conditions are to be true (DeRose et al., 1990: 17). Without a proper database these are done partly manually: the researcher finds all texts that satisfy one condition and then searches within that data for the other conditions. One should remember that even if complex searches can be done (on more than one parameter) the researcher often still has to evaluate the hits returned by the database engine to see if they are relevant to verify or falsify a hypothesis (Petersen, 2004b).

However, the ultimate goal of a database should not only be "efficient retrieval, but interpretive insight" (cf. Ramsay, *s.a.*). Data mining or "knowledge discovery" may fulfil this role. It is the process of identifying new patterns and trends in large amounts of data. It should actually be called knowledge invention since it is a determinate process of knowledge construction. Data mining is usually done on numerical business data, and text mining is a parallel field studying the same type of activity within large amounts of text-based business data (cf. Hearst, 1999; Kroeze, Matthee & Bothma, 2004).

The type of knowledge creation illustrated in this study may be regarded as an application of text mining to humanistic resources, in this case the Hebrew text of Genesis 1:1-2:3. The data set is very small, and it should be regarded as an exploratory experiment that evaluates the applicability of these research activities for humanities computing.

Data mining, as well as text mining, forms the highest level of the knowledge creation pyramid (see Figure 6.1). The three lower levels work with explicit knowledge while the data mining level pertains to implicit (tacit) knowledge. In this project, the lowest level ("storing and exchanging of 'factual' knowledge") is represented by the initial capturing of the analyses of the Genesis 1:1-2:3 text using a series of twodimensional tables.[116] The second level (modelling of 'conceptual knowledge') refers to the links between analyses of the various linguistic layers which is automatically done and implied by the data cube in the higher level of integration. If more tables or cubes were added, for example to refer to the definitions of syntactic and semantic functions, the one-to-many relationships between these tables and the data cube could be modelled in a relational schema to show the relationships between the entity sets. The third level in the pyramid organizes and integrates information from various sources or modules in one integrated repository to facilitate online analytical processing. In the empirical work for this thesis integration is represented by the building of the data cube either in XML or in the threedimensional array in VB6. The cube may be regarded as a multidimensional data warehouse. Multidimensional interpretation is a form of data mining. Therefore, extracting patterns, such as unique semantic frameworks, from the cube is an implementation of knowledge creation. It consciously explores the integrated data[117] in order to find new combinations or, at least, to confirm existing hypotheses about known combinations. Even if no new combinations are found, knowledge is still created, namely that there are no new patterns, which may confirm an existing theory. If no examples of a known pattern are found, it may indicate that the pattern is very rare, but one would need a comprehensive data set (e.g. of the whole Hebrew Bible) to conclude that a certain pattern does not occur at all. These insights may be regarded as "the uncovering of *new, implicit* and *potentially useful* knowledge from large amounts of data" (Cannataro et al., 2002: 33-34).

[116] The analysis itself, of course, is the explication of implicit knowledge.

[117] For another example of XML being used to integrate linguistic data, compare Bird et al. (2002) who converted divergent lexicographical material from three South-west USA languages to the same format.

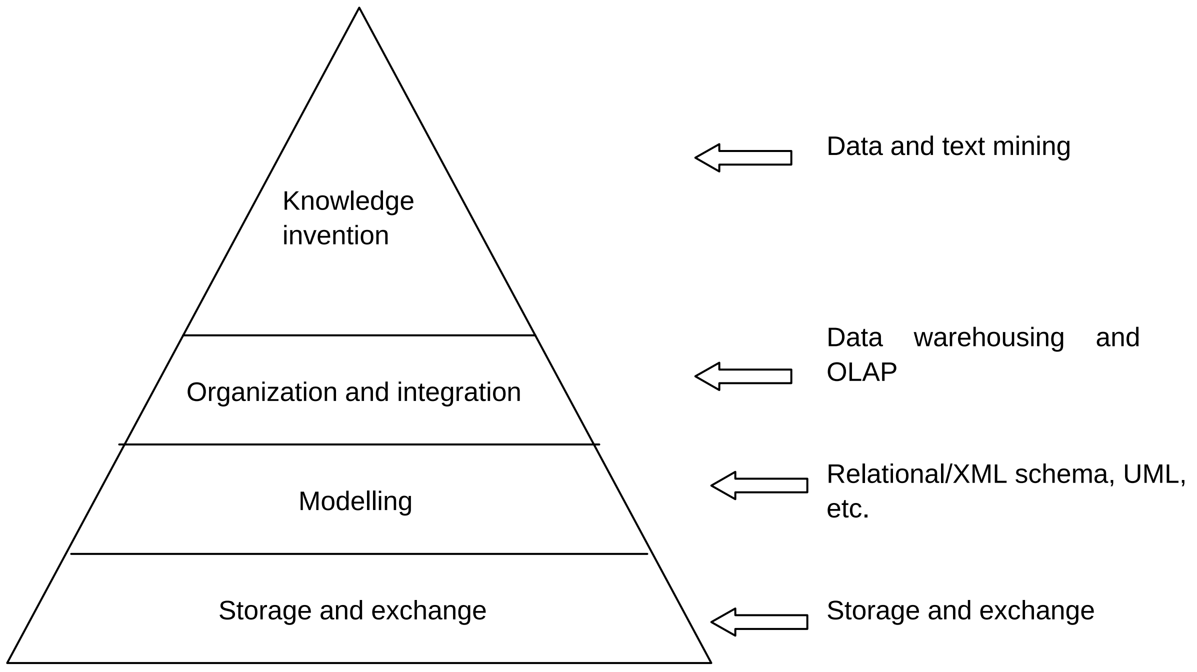


Figure 6.1. The knowledge creation paradigm (adapted from Cannataro et al., 2002: 34).

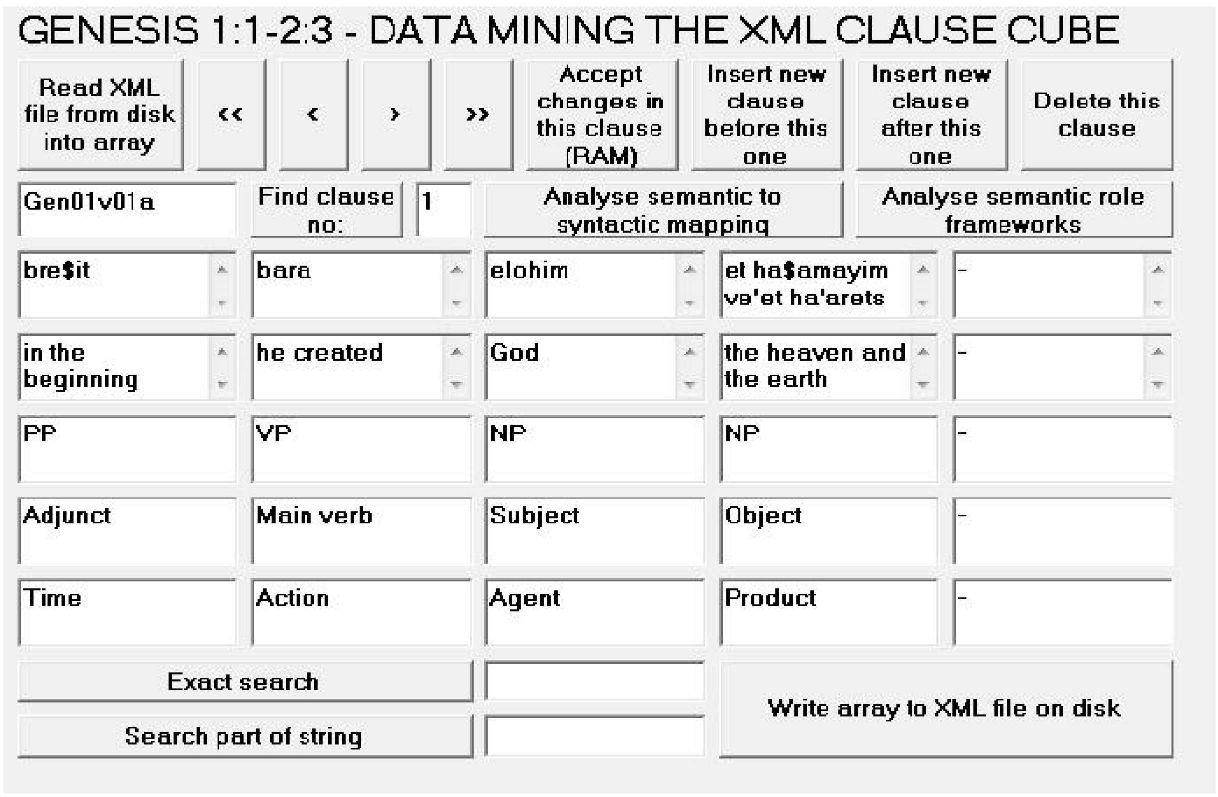
Wintner (2004: 128) identified a lack of Hebrew computational linguistic projects that deal directly with semantics. Both experiments conducted in this chapter make small contributions towards filling that gap. Embarking on such a venture is, however, no easy task, since human language is a complex phenomenon, especially in the case of Hebrew "due to its rich morphology and deficient script" (ibid.*:* 113-114). For these experiments these additional complexities were eliminated by using a phonetical transcription for the rendering of the Hebrew text and ignoring the morphological module.

The program uses, *i.a.*, for-loops and if-statements to process the data in the threedimensional array.[118] These functionalities (repetition and selection structures) are also typical of template-based XML query languages (Bourret, 2003). While the array structure itself facilitates the data storage, the array manipulation could be regarded as an application tool that enables data mining. Storage and application tools are two of the main data warehouse modules. Application tools include online

[118] Note that the size of the array (and any copies of it used for the data mining procedures – see below) should be adjusted if clauses are added or deleted to the clause cube (cf. Chapter 5).

analytical processing (OLAP), and mining and analysis tools (Wang & Dong, 2001: 49). Compare Petroustos (1999) for the relevant VB6 syntax.

Two buttons have been added to the default screen on the program interface to direct the researcher to the data-mining experiments ("Analyse semantic frameworks" and "Analyse semantic to syntactic mapping"). The interface of the completed program is shown in Figure 6.2.


Read XML file from disk into array
<<
<
>
>>
Accept changes in this clause (RAM)
Insert new clause before this one
Insert new clause after this one
Delete this clause
Find clause no:
Analyse semantic to syntactic mapping
Analyse semantic role frameworks
Exact search
Search part of string
Write array to XML file on disk

GENESIS 1:1-2:3 - DATA MINING THE XML CLAUSE CUBE

Gen01v01a | Find clause no: 1

bre$it	bara	elohim	et ha$amayim ve'et ha'arets	-
in the beginning	he created	God	the heaven and the earth	-
PP	VP	NP	NP	-
Adjunct	Main verb	Subject	Object	-
Time	Action	Agent	Product	-

Figure 6.2. The interface of the completed program with buttons added to direct the user to the data-mining experiments.

When the user clicks on the "Analyse semantic frameworks" button, a new interface is opened to facilitate the desired analysis (see 6.4 and Figure 6.3). Clicking on the "Analyse semantic to syntactic mapping" button also opens a different interface to direct the researcher through the relevant text-mining steps (see 6.5 and Figure 6.15).

6.4 Extraction and analysis of semantic role frameworks as a venture in text data mining

The first experiment investigates the unique semantic role frameworks found in the clause cube. The VB6 code to extract and analyse the relevant information, as well as the results of the undertaking, is discussed below. The following logical steps are followed:

- Slice the semantic functions
- Sort elements in each row
- Concatenate each row
- Order rows
- Identify unique frames and calculate their frequencies

The program interface reflects this logical flow (see Figure 6.3).

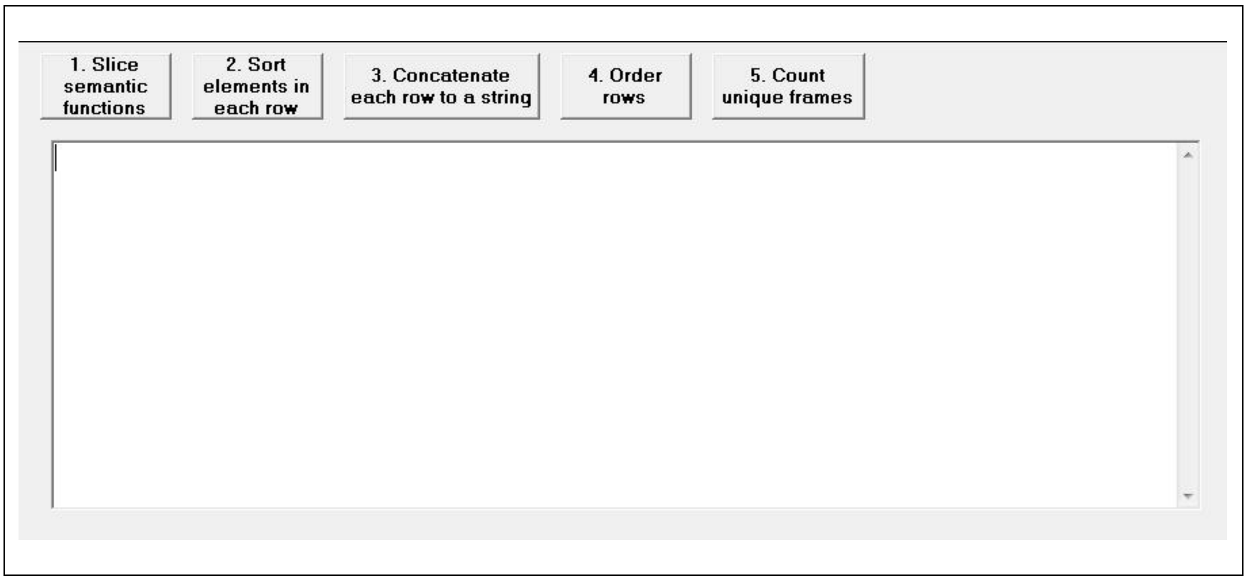

1. Slice semantic functions
2. Sort elements in each row
3. Concatenate each row to a string
4. Order rows
5. Count unique frames

Figure 6.3. The interface of the experiment that analyses semantic frameworks in the clause cube.

6.4.1 Slicing off the semantic functions layer from the data cube

The first step of this data-mining experiment is activated by pressing the button, "Slice semantic functions". The VB6 code written to extract (slice) the semantic role frames is provided below. While a twodimensional array is being populated with the semantic functions from the data cube, validation is performed to ensure that the terminology used is consistent.[119] Although it could be argued that another validation process is superfluous if the XML database was tested successfully against its schema, it is still recommendable, in order to ensure that clean data is used for the data-mining venture, just in case updates have been done to the XML file and the researcher has forgotten to repeat the schema validation. Another reason to do specific validation here is that the schema does validation on the whole XML file, and unless the researcher has access to the schema, one cannot assume that validation code has been included to check the parameters currently under consideration.

Numbers are prefixed to the semantic functions to facilitate a logical sorting process of the rows' elements and the rows themselves (as opposed to an alphabetical ordering according to the names of the functions). The functions are numbered more or less in the order that they are discussed by Dik (1997a), starting with predicate types, followed by arguments and satellites. If no semantic function is present, indicated by a dash, it is numbered as 98 to ensure that it will be placed after all the other semantic functions during ordering later.[120] If an invalid semantic function is found (e.g. incorrectly spelled), an error message is shown. The user should correct the source data before continuing with the ensuing steps.

The slicing, validating and numbering code is shown in Figure 6.4.

```
Option Explicit

Dim arrSem1(1 To 200, 1 To 6) As String
Dim countf21, countf22 As Integer
```

[119] Compare Chapter 4 for validation done by means of an XML schema.

[120] The number 98 was chosen at random – it could be any number higher than the numbers allocated to the unique semantic functions to facilitate the logical sorting process.

```
Dim arrSem2(1 To 200, 1 To 2) As String
Dim temp, temp2 As String

'Slice, validate and number semantic functions

Public Sub cmdSliceSem_Click() 'Load semantic role frames into arrSem1

For countf21 = 1 To arrayMax 'Slice semantic frames into new 2D array
arrSem1(countf21, 6) = Clause(countf21, 1, 1) 'Store verse no as reference
 For countf22 = 1 To 5
      arrSem1(countf21, countf22) = Clause(countf21, countf22, 6)
      'Number semantic functions to facilitate logical ordering
      If arrSem1(countf21, countf22) = "-" Then 'Number null value as last
         semantic function
       arrSem1(countf21, countf22) = "98. " & arrSem1(countf21, countf22)
      ElseIf arrSem1(countf21, countf22) = "Action" Then
       arrSem1(countf21, countf22) = "01. " & arrSem1(countf21, countf22)
      ElseIf arrSem1(countf21, countf22) = "Position" Then
       arrSem1(countf21, countf22) = "02. " & arrSem1(countf21, countf22)
      ElseIf arrSem1(countf21, countf22) = "Process" Then
       arrSem1(countf21, countf22) = "03. " & arrSem1(countf21, countf22)
      ElseIf arrSem1(countf21, countf22) = "State" Then
       arrSem1(countf21, countf22) = "04. " & arrSem1(countf21, countf22)
      ElseIf arrSem1(countf21, countf22) = "Agent" Then
       arrSem1(countf21, countf22) = "05. " & arrSem1(countf21, countf22)
      ElseIf arrSem1(countf21, countf22) = "Positioner" Then
       arrSem1(countf21, countf22) = "06. " & arrSem1(countf21, countf22)
      ElseIf arrSem1(countf21, countf22) = "Force" Then
       arrSem1(countf21, countf22) = "07. " & arrSem1(countf21, countf22)
      ElseIf arrSem1(countf21, countf22) = "Processed" Then
       arrSem1(countf21, countf22) = "08. " & arrSem1(countf21, countf22)
      ElseIf arrSem1(countf21, countf22) = "Zero" Then
       arrSem1(countf21, countf22) = "09. " & arrSem1(countf21, countf22)
      ElseIf arrSem1(countf21, countf22) = "Patient" Then
       arrSem1(countf21, countf22) = "10a. " & arrSem1(countf21, countf22)
      ElseIf arrSem1(countf21, countf22) = "Product" Then
       arrSem1(countf21, countf22) = "10b. " & arrSem1(countf21, countf22)
      ElseIf arrSem1(countf21, countf22) = "Receiver" Then
       arrSem1(countf21, countf22) = "11. " & arrSem1(countf21, countf22)
      ElseIf arrSem1(countf21, countf22) = "Location" Then
       arrSem1(countf21, countf22) = "12. " & arrSem1(countf21, countf22)
      ElseIf arrSem1(countf21, countf22) = "Direction" Then
       arrSem1(countf21, countf22) = "13. " & arrSem1(countf21, countf22)
      ElseIf arrSem1(countf21, countf22) = "Source" Then
       arrSem1(countf21, countf22) = "14. " & arrSem1(countf21, countf22)
      ElseIf arrSem1(countf21, countf22) = "Reference" Then
```

```
         arrSem1(countf21, countf22) = "15. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Beneficiary" Then
         arrSem1(countf21, countf22) = "16. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Company" Then
         arrSem1(countf21, countf22) = "17. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Instrument" Then
         arrSem1(countf21, countf22) = "18. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Manner" Then
         arrSem1(countf21, countf22) = "19. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Speed" Then
         arrSem1(countf21, countf22) = "20. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Role" Then
         arrSem1(countf21, countf22) = "21. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Path" Then
         arrSem1(countf21, countf22) = "22. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Time" Then
         arrSem1(countf21, countf22) = "23. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Circumstance" Then
         arrSem1(countf21, countf22) = "26. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Result" Then
         arrSem1(countf21, countf22) = "27. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Purpose" Then
         arrSem1(countf21, countf22) = "28. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Reason" Then
         arrSem1(countf21, countf22) = "29. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Cause" Then
         arrSem1(countf21, countf22) = "30. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Possessor" Then
         arrSem1(countf21, countf22) = "31. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Identification" Then
         arrSem1(countf21, countf22) = "32. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Classification" Then
         arrSem1(countf21, countf22) = "33. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Quality" Then
         arrSem1(countf21, countf22) = "34. " & arrSem1(countf21, countf22)
        ElseIf arrSem1(countf21, countf22) = "Existence" Then
         arrSem1(countf21, countf22) = "35. " & arrSem1(countf21, countf22)
        'Clean data if following message is shown:
        Else: MsgBox ("Semf " & arrSem1(countf21, countf22) & " in vs " &
           arrSem1(countf21, 6) & " could not be numbered")
        End If
    Next
Next

temp2 = "SLICE OF SEMANTIC FUNCTION FRAMES FROM GEN 1:1-2:3 DATA CUBE"
Call showArrSem1
```

```
End Sub
```

Figure 6.4. VB6 code used to slice off the module representative of the semantic role frameworks, while validating and numbering the semantic functions.

The code that is used to show the results in a textbox is presented in Figure 6.5.

```
'Show slice (arrSem1) in textbox

Public Sub showArrSem1()
Text1.Text = ""
temp = ""

For countf21 = 1 To arrayMax
  temp2 = temp2 & vbCrLf & temp
  temp = "Clause " & Str(countf21) & ": "
 For countf22 = 1 To 6
  temp = temp & " + " & arrSem1(countf21, countf22)
 Next
Next

temp2 = temp2 & vbCrLf & temp
Text1.Text = temp2

End Sub
```

Figure 6.5. VB6 code used to display the semantic functions slice in a textbox.

When this code is run, the semantic frames[121] of the 108 clauses are shown in a textbox (see Figure 6.6).

```
SLICE OF SEMANTIC FUNCTION FRAMES EXTRACTED FROM GEN 1:1-2:3 DATA CUBE

Clause  1:  + 23. Time + 01. Action + 05. Agent + 10b. Product + 98. - + Gen01v01a
```

[121] "Semantic frame" and "semantic frameworks" are used as synonyms.

```
Clause  2:  + 09. Zero + 04. State + 33. Classification + 98. - + 98. - + Gen01v02a
Clause  3:  + 09. Zero + 12. Location + 98. - + 98. - + 98. - + Gen01v02b
Clause  4:  + 06. Positioner + 02. Position + 12. Location + 98. - + 98. - + Gen01v02c
Clause  5:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v03a
Clause  6:  + 04. State + 09. Zero + 98. - + 98. - + 98. - + Gen01v03b
Clause  7:  + 04. State + 09. Zero + 98. - + 98. - + 98. - + Gen01v03c
Clause  8:  + 03. Process + 08. Processed + 10a. Patient + 98. - + 98. - + Gen01v04a
Clause  9:  + 09. Zero + 98. - + 34. Quality + 98. - + 98. - + Gen01v04b
Clause  10:  + 01. Action + 05. Agent + 10a. Patient + 14. Source + 98. - + Gen01v04c
Clause  11:  + 01. Action + 05. Agent + 10a. Patient + 10b. Product + 98. - + Gen01v05a
Clause  12:  + 10a. Patient + 01. Action + 10b. Product + 98. - + 98. - + Gen01v05b
Clause  13:  + 04. State + 09. Zero + 98. - + 98. - + 98. - + Gen01v05c
Clause  14:  + 04. State + 09. Zero + 34. Quality + 98. - + 98. - + Gen01v05d
Clause  15:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v06a
Clause  16:  + 04. State + 09. Zero + 12. Location + 98. - + 98. - + Gen01v06b
Clause  17:  + 01. Action + 10a. Patient + 14. Source + 98. - + 98. - + Gen01v06c
Clause  18:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v07a
Clause  19:  + 01. Action + 10a. Patient + 14. Source + 98. - + 98. - + Gen01v07b
Clause  20:  + 09. Zero + 12. Location + 98. - + 98. - + 98. - + Gen01v07c
Clause  21:  + 09. Zero + 12. Location + 98. - + 98. - + 98. - + Gen01v07d
Clause  22:  + 04. State + 19. Manner + 98. - + 98. - + 98. - + Gen01v07e
Clause  23:  + 01. Action + 05. Agent + 10a. Patient + 10b. Product + 98. - + Gen01v08a
Clause  24:  + 04. State + 09. Zero + 98. - + 98. - + 98. - + Gen01v08b
Clause  25:  + 04. State + 09. Zero + 34. Quality + 98. - + 98. - + Gen01v08c
Clause  26:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v09a
Clause  27:  + 03. Process + 08. Processed + 12. Location + 98. - + 98. - + Gen01v09b
Clause  28:  + 03. Process + 08. Processed + 98. - + 98. - + 98. - + Gen01v09c
Clause  29:  + 04. State + 19. Manner + 98. - + 98. - + 98. - + Gen01v09d
Clause  30:  + 01. Action + 05. Agent + 10a. Patient + 10b. Product + 98. - + Gen01v10a
Clause  31:  + 10a. Patient + 01. Action + 10b. Product + 98. - + 98. - + Gen01v10b
Clause  32:  + 03. Process + 08. Processed + 10a. Patient + 98. - + 98. - + Gen01v10c
Clause  33:  + 98. - + 34. Quality + 98. - + 98. - + 98. - + Gen01v10d
Clause  34:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v11a
Clause  35:  + 03. Process + 08. Processed + 15. Reference + 98. - + 98. - + Gen01v11b
Clause  36:  + 98. - + 09. Zero + 12. Location + 98. - + 98. - + Gen01v11c
Clause  37:  + 04. State + 19. Manner + 98. - + 98. - + 98. - + Gen01v11d
Clause  38:  + 03. Process + 08. Processed + 10b. Product + 19. Manner + 98. - + Gen01v12a
Clause  39:  + 98. - + 09. Zero + 12. Location + 98. - + 98. - + Gen01v12b
Clause  40:  + 03. Process + 08. Processed + 10a. Patient + 98. - + 98. - + Gen01v12c
Clause  41:  + 98. - + 34. Quality + 98. - + 98. - + 98. - + Gen01v12d
Clause  42:  + 04. State + 09. Zero + 98. - + 98. - + 98. - + Gen01v13a
Clause  43:  + 04. State + 09. Zero + 34. Quality + 98. - + 98. - + Gen01v13b
Clause  44:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v14a
Clause  45:  + 04. State + 09. Zero + 12. Location + 28. Purpose + 98. - + Gen01v14b
Clause  46:  + 04. State + 33. Classification + 98. - + 98. - + 98. - + Gen01v14c
Clause  47:  + 04. State + 33. Classification + 12. Location + 28. Purpose + 98. - + Gen01v15a
Clause  48:  + 04. State + 19. Manner + 98. - + 98. - + 98. - + Gen01v15b
Clause  49:  + 01. Action + 05. Agent + 10b. Product + 28. Purpose + 98. - + Gen01v16a
Clause  50:  + 01. Action + 10a. Patient + 05. Agent + 12. Location + 28. Purpose + Gen01v17a
Clause  51:  + 03. Process + 08. Processed + 10a. Patient + 98. - + 98. - + Gen01v18b
Clause  52:  + 98. - + 34. Quality + 98. - + 98. - + 98. - + Gen01v18d
Clause  53:  + 04. State + 09. Zero + 98. - + 98. - + 98. - + Gen01v19a
Clause  54:  + 04. State + 09. Zero + 34. Quality + 98. - + 98. - + Gen01v19b
Clause  55:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v20a
Clause  56:  + 03. Process + 08. Processed + 15. Reference + 98. - + 98. - + Gen01v20b
Clause  57:  + 05. Agent + 01. Action + 12. Location + 12. Location + 98. - + Gen01v20c
Clause  58:  + 01. Action + 05. Agent + 10b. Product + 19. Manner + 98. - + Gen01v21a
Clause  59:  + 05. Agent + 01. Action + 12. Location + 98. - + 98. - + Gen01v21b
Clause  60:  + 03. Process + 08. Processed + 10a. Patient + 98. - + 98. - + Gen01v21c
Clause  61:  + 98. - + 34. Quality + 98. - + 98. - + 98. - + Gen01v21d
Clause  62:  + 01. Action + 10a. Patient + 05. Agent + 19. Manner + 98. - + Gen01v22a
Clause  63:  + 04. State + 98. - + 98. - + 98. - + 98. - + Gen01v22b
Clause  64:  + 03. Process + 98. - + 98. - + 98. - + 98. - + Gen01v22c
Clause  65:  + 01. Action + 10a. Patient + 98. - + 98. - + 98. - + Gen01v22d
Clause  66:  + 08. Processed + 03. Process + 12. Location + 98. - + 98. - + Gen01v22e
Clause  67:  + 04. State + 09. Zero + 98. - + 98. - + 98. - + Gen01v23a
Clause  68:  + 04. State + 09. Zero + 34. Quality + 98. - + 98. - + Gen01v23b
Clause  69:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v24a
Clause  70:  + 03. Process + 08. Processed + 10b. Product + 19. Manner + 98. - + Gen01v24b
Clause  71:  + 04. State + 19. Manner + 98. - + 98. - + 98. - + Gen01v24c
Clause  72:  + 01. Action + 05. Agent + 10b. Product + 19. Manner + 98. - + Gen01v25a
Clause  73:  + 03. Process + 08. Processed + 10a. Patient + 98. - + 98. - + Gen01v25b
Clause  74:  + 98. - + 34. Quality + 98. - + 98. - + 98. - + Gen01v25c
Clause  75:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v26a
Clause  76:  + 01. Action + 10b. Product + 19. Manner + 98. - + 98. - + Gen01v26b
Clause  77:  + 05. Agent + 10a. Patient + 98. - + 98. - + 98. - + Gen01v26c
Clause  78:  + 01. Action + 05. Agent + 10b. Product + 19. Manner + 98. - + Gen01v27a
Clause  79:  + 19. Manner + 01. Action + 10b. Product + 98. - + 98. - + Gen01v27b
Clause  80:  + 34. Quality + 01. Action + 10b. Product + 98. - + 98. - + Gen01v27c
Clause  81:  + 01. Action + 10a. Patient + 05. Agent + 98. - + 98. - + Gen01v28a
Clause  82:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v28b
Clause  83:  + 04. State + 98. - + 98. - + 98. - + 98. - + Gen01v28c
Clause  84:  + 03. Process + 98. - + 98. - + 98. - + 98. - + Gen01v28d
Clause  85:  + 01. Action + 10a. Patient + 98. - + 98. - + 98. - + Gen01v28e
Clause  86:  + 01. Action + 98. - + 98. - + 98. - + 98. - + Gen01v28f
Clause  87:  + 01. Action + 10a. Patient + 98. - + 98. - + 98. - + Gen01v28g
Clause  88:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v29a
Clause  89:  + 98. - + 01. Action + 11. Receiver + 10a. Patient + 98. - + Gen01v29b
Clause  90:  + 09. Zero + 12. Location + 98. - + 98. - + 98. - + Gen01v29c
Clause  91:  + 09. Zero + 12. Location + 09. Zero + 98. - + 98. - + Gen01v29d
Clause  92:  + 16. Beneficiary + 04. State + 28. Purpose + 98. - + 98. - + Gen01v29e
Clause  93:  + 98. - + 12. Location + 09. Zero + 98. - + 98. - + Gen01v30b
Clause  94:  + 04. State + 19. Manner + 98. - + 98. - + 98. - + Gen01v30c
Clause  95:  + 03. Process + 08. Processed + 10a. Patient + 98. - + 98. - + Gen01v31a
Clause  96:  + 10b. Product + 01. Action + 98. - + 98. - + 98. - + Gen01v31b
Clause  97:  + 98. - + 34. Quality + 98. - + 98. - + 98. - + Gen01v31c
Clause  98:  + 04. State + 09. Zero + 98. - + 98. - + 98. - + Gen01v31d
Clause  99:  + 04. State + 09. Zero + 34. Quality + 98. - + 98. - + Gen01v31e
Clause  100:  + 01. Action + 10a. Patient + 98. - + 98. - + 98. - + Gen02v01a
Clause  101:  + 01. Action + 05. Agent + 23. Time + 10a. Patient + 98. - + Gen02v02a
Clause  102:  + 10b. Product + 01. Action + 98. - + 98. - + 98. - + Gen02v02b
Clause  103:  + 01. Action + 23. Time + 10a. Patient + 98. - + 98. - + Gen02v02c
Clause  104:  + 10b. Product + 01. Action + 98. - + 98. - + 98. - + Gen02v02d
Clause  105:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen02v03a
Clause  106:  + 01. Action + 10a. Patient + 29. Reason + 98. - + 98. - + Gen02v03b
Clause  107:  + 98. - + 23. Time + 01. Action + 10a. Patient + 98. - + Gen02v03c
Clause  108:  + 10b. Product + 01. Action + 05. Agent + 19. Manner + 98. - + Gen02v03d
```

Figure 6.6. The semantic frames of the 108 extracted, validated and numbered clauses, shown in a textbox.

Although these results are not very user-friendly, it is only the first, but essential step in the data mining process. The following steps will use this subset of data to organise and analyse the data. Aggregating the extracted combinations of semantic roles into groups of unique semantic frames may be regarded as the identification of a pattern or patterns, which is one of the characteristics of data mining (Thuraisingham, 2002: 191). This process is not only based on a linguistic knowledge representation (the marked-up text) but also creates, or at least enhances, a new knowledge representation (the patterns of semantic frames). The outcomes of humanistic computing is the explication of existing knowledge about a discipline by means of a knowledge representation, testing this ontology against a body of (marked-up) data and the creation of new insights: "[C]omputing a representation of large amounts of material that has been encoded and processed according to a rigorous, well thought-out knowledge representation affords opportunities for perceiving and analysing patterns, conjunctions, connections, and absences that a human being, unaided by a computer, would not be likely to find" (Unsworth, 2001).

6.4.2 Sorting the elements in each row

To facilitate aggregation functions the data should now be sorted logically. The second step of this data-mining experiment is activated by pressing the button, "Sort elements in each row". First, the numbers appended to the semantic functions are used to sort the semantic functions in each clause in the order of clause type, arguments and satellites. Although the word order is lost, the goal here is to deduct the logical structure of the semantic frameworks. If word order is important for the researcher, it could be recovered by means of the clause references.

The code used to sort the elements in each row is presented in Figure 6.7.

```
'Sorting the semantic functions in each row

Public Sub btnSortEls_Click()

Dim count1, count2, count3 As Integer

Dim temp As String
Dim ix1, ix2, ix3 As Integer
count1 = 1
count2 = 1
count3 = 1
ix1 = 1
ix2 = 1
ix3 = 1

For count1 = 1 To arrayMax 'Simple sort algorithm used
 For count2 = 1 To 5
  ix3 = count2 + 1
  For count3 = count2 To 4
   If arrSem1(ix1, ix3) < arrSem1(ix1, ix2) Then
    temp = arrSem1(ix1, ix2)
    arrSem1(ix1, ix2) = arrSem1(ix1, ix3)
    arrSem1(ix1, ix3) = temp
   End If
   ix3 = ix3 + 1
   Next
   ix2 = ix2 + 1
 Next
 ix1 = ix1 + 1
 ix2 = 1
 ix3 = 1
Next

temp2 = "SEMANTIC FUNCTION FRAMES ORDERED LOGICALLY USING NUMBERS"
Call showArrSem1

End Sub
```

Figure 6.7. VB6 code used to sort the semantic functions in each row.

When this code is run, the following results appear in a textbox (see Figure 6.8).

```
SEMANTIC FUNCTION FRAMES LOGICALLY ORDERED USING NUMBERS

Clause  1:  + 01. Action + 05. Agent + 10b. Product + 23. Time + 98. - + Gen01v01a
Clause  2:  + 04. State + 09. Zero + 33. Classification + 98. - + 98. - + Gen01v02a
Clause  3:  + 09. Zero + 12. Location + 98. - + 98. - + 98. - + Gen01v02b
Clause  4:  + 02. Position + 06. Positioner + 12. Location + 98. - + 98. - + Gen01v02c
Clause  5:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v03a
Clause  6:  + 04. State + 09. Zero + 98. - + 98. - + 98. - + Gen01v03b
Clause  7:  + 04. State + 09. Zero + 98. - + 98. - + 98. - + Gen01v03c
Clause  8:  + 03. Process + 08. Processed + 10a. Patient + 98. - + 98. - + Gen01v04a
Clause  9:  + 09. Zero + 34. Quality + 98. - + 98. - + 98. - + Gen01v04b
Clause  10:  + 01. Action + 05. Agent + 10a. Patient + 14. Source + 98. - + Gen01v04c
Clause  11:  + 01. Action + 05. Agent + 10a. Patient + 10b. Product + 98. - + Gen01v05a
Clause  12:  + 01. Action + 10a. Patient + 10b. Product + 98. - + 98. - + Gen01v05b
Clause  13:  + 04. State + 09. Zero + 98. - + 98. - + 98. - + Gen01v05c
Clause  14:  + 04. State + 09. Zero + 34. Quality + 98. - + 98. - + Gen01v05d
Clause  15:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v06a
Clause  16:  + 04. State + 09. Zero + 12. Location + 98. - + 98. - + Gen01v06b
Clause  17:  + 01. Action + 10a. Patient + 14. Source + 98. - + 98. - + Gen01v06c
Clause  18:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v07a
Clause  19:  + 01. Action + 10a. Patient + 14. Source + 98. - + 98. - + Gen01v07b
Clause  20:  + 09. Zero + 12. Location + 98. - + 98. - + 98. - + Gen01v07c
Clause  21:  + 09. Zero + 12. Location + 98. - + 98. - + 98. - + Gen01v07d
Clause  22:  + 04. State + 19. Manner + 98. - + 98. - + 98. - + Gen01v07e
Clause  23:  + 01. Action + 05. Agent + 10a. Patient + 10b. Product + 98. - + Gen01v08a
Clause  24:  + 04. State + 09. Zero + 98. - + 98. - + 98. - + Gen01v08b
Clause  25:  + 04. State + 09. Zero + 34. Quality + 98. - + 98. - + Gen01v08c
Clause  26:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v09a
Clause  27:  + 03. Process + 08. Processed + 12. Location + 98. - + 98. - + Gen01v09b
Clause  28:  + 03. Process + 08. Processed + 98. - + 98. - + 98. - + Gen01v09c
Clause  29:  + 04. State + 19. Manner + 98. - + 98. - + 98. - + Gen01v09d
Clause  30:  + 01. Action + 05. Agent + 10a. Patient + 10b. Product + 98. - + Gen01v10a
Clause  31:  + 01. Action + 10a. Patient + 10b. Product + 98. - + 98. - + Gen01v10b
Clause  32:  + 03. Process + 08. Processed + 10a. Patient + 98. - + 98. - + Gen01v10c
Clause  33:  + 34. Quality + 98. - + 98. - + 98. - + 98. - + Gen01v10d
Clause  34:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v11a
Clause  35:  + 03. Process + 08. Processed + 15. Reference + 98. - + 98. - + Gen01v11b
Clause  36:  + 09. Zero + 12. Location + 98. - + 98. - + 98. - + Gen01v11c
Clause  37:  + 04. State + 19. Manner + 98. - + 98. - + 98. - + Gen01v11d
Clause  38:  + 03. Process + 08. Processed + 10b. Product + 19. Manner + 98. - + Gen01v12a
Clause  39:  + 09. Zero + 12. Location + 98. - + 98. - + 98. - + Gen01v12b
Clause  40:  + 03. Process + 08. Processed + 10a. Patient + 98. - + 98. - + Gen01v12c
Clause  41:  + 34. Quality + 98. - + 98. - + 98. - + 98. - + Gen01v12d
Clause  42:  + 04. State + 09. Zero + 98. - + 98. - + 98. - + Gen01v13a
Clause  43:  + 04. State + 09. Zero + 34. Quality + 98. - + 98. - + Gen01v13b
Clause  44:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v14a
Clause  45:  + 04. State + 09. Zero + 12. Location + 28. Purpose + 98. - + Gen01v14b
Clause  46:  + 04. State + 33. Classification + 98. - + 98. - + 98. - + Gen01v14c
Clause  47:  + 04. State + 12. Location + 28. Purpose + 33. Classification + 98. - + Gen01v15a
Clause  48:  + 04. State + 19. Manner + 98. - + 98. - + 98. - + Gen01v15b
Clause  49:  + 01. Action + 05. Agent + 10b. Product + 28. Purpose + 98. - + Gen01v16a
Clause  50:  + 01. Action + 05. Agent + 10a. Patient + 12. Location + 28. Purpose + Gen01v17a
Clause  51:  + 03. Process + 08. Processed + 10a. Patient + 98. - + 98. - + Gen01v18b
Clause  52:  + 34. Quality + 98. - + 98. - + 98. - + 98. - + Gen01v18d
Clause  53:  + 04. State + 09. Zero + 98. - + 98. - + 98. - + Gen01v19a
Clause  54:  + 04. State + 09. Zero + 34. Quality + 98. - + 98. - + Gen01v19b
Clause  55:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v20a
Clause  56:  + 03. Process + 08. Processed + 15. Reference + 98. - + 98. - + Gen01v20b
Clause  57:  + 01. Action + 05. Agent + 12. Location + 12. Location + 98. - + Gen01v20c
Clause  58:  + 01. Action + 05. Agent + 10b. Product + 19. Manner + 98. - + Gen01v21a
Clause  59:  + 01. Action + 05. Agent + 12. Location + 98. - + 98. - + Gen01v21b
Clause  60:  + 03. Process + 08. Processed + 10a. Patient + 98. - + 98. - + Gen01v21c
Clause  61:  + 34. Quality + 98. - + 98. - + 98. - + 98. - + Gen01v21d
Clause  62:  + 01. Action + 05. Agent + 10a. Patient + 19. Manner + 98. - + Gen01v22a
Clause  63:  + 04. State + 98. - + 98. - + 98. - + 98. - + Gen01v22b
Clause  64:  + 03. Process + 98. - + 98. - + 98. - + 98. - + Gen01v22c
Clause  65:  + 01. Action + 10a. Patient + 98. - + 98. - + 98. - + Gen01v22d
Clause  66:  + 03. Process + 08. Processed + 12. Location + 98. - + 98. - + Gen01v22e
Clause  67:  + 04. State + 09. Zero + 98. - + 98. - + 98. - + Gen01v23a
Clause  68:  + 04. State + 09. Zero + 34. Quality + 98. - + 98. - + Gen01v23b
Clause  69:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v24a
Clause  70:  + 03. Process + 08. Processed + 10b. Product + 19. Manner + 98. - + Gen01v24b
Clause  71:  + 04. State + 19. Manner + 98. - + 98. - + 98. - + Gen01v24c
Clause  72:  + 01. Action + 05. Agent + 10b. Product + 19. Manner + 98. - + Gen01v25a
Clause  73:  + 03. Process + 08. Processed + 10a. Patient + 98. - + 98. - + Gen01v25b
Clause  74:  + 34. Quality + 98. - + 98. - + 98. - + 98. - + Gen01v25c
Clause  75:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v26a
Clause  76:  + 01. Action + 10b. Product + 19. Manner + 98. - + 98. - + Gen01v26b
Clause  77:  + 05. Agent + 10a. Patient + 98. - + 98. - + 98. - + Gen01v26c
Clause  78:  + 01. Action + 05. Agent + 10b. Product + 19. Manner + 98. - + Gen01v27a
Clause  79:  + 01. Action + 10b. Product + 19. Manner + 98. - + 98. - + Gen01v27b
Clause  80:  + 01. Action + 10b. Product + 34. Quality + 98. - + 98. - + Gen01v27c
Clause  81:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v28a
Clause  82:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v28b
Clause  83:  + 04. State + 98. - + 98. - + 98. - + 98. - + Gen01v28c
Clause  84:  + 03. Process + 98. - + 98. - + 98. - + 98. - + Gen01v28d
Clause  85:  + 01. Action + 10a. Patient + 98. - + 98. - + 98. - + Gen01v28e
Clause  86:  + 01. Action + 98. - + 98. - + 98. - + 98. - + Gen01v28f
Clause  87:  + 01. Action + 10a. Patient + 98. - + 98. - + 98. - + Gen01v28g
Clause  88:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen01v29a
Clause  89:  + 01. Action + 10a. Patient + 11. Receiver + 98. - + 98. - + Gen01v29b
Clause  90:  + 09. Zero + 12. Location + 98. - + 98. - + 98. - + Gen01v29c
Clause  91:  + 09. Zero + 09. Zero + 12. Location + 98. - + 98. - + Gen01v29d
Clause  92:  + 04. State + 16. Beneficiary + 28. Purpose + 98. - + 98. - + Gen01v29e
Clause  93:  + 09. Zero + 12. Location + 98. - + 98. - + 98. - + Gen01v30b
Clause  94:  + 04. State + 19. Manner + 98. - + 98. - + 98. - + Gen01v30c
Clause  95:  + 03. Process + 08. Processed + 10a. Patient + 98. - + 98. - + Gen01v31a
Clause  96:  + 01. Action + 10b. Product + 98. - + 98. - + 98. - + Gen01v31b
Clause  97:  + 34. Quality + 98. - + 98. - + 98. - + 98. - + Gen01v31c
Clause  98:  + 04. State + 09. Zero + 98. - + 98. - + 98. - + Gen01v31d
Clause  99:  + 04. State + 09. Zero + 34. Quality + 98. - + 98. - + Gen01v31e
Clause  100:  + 01. Action + 10a. Patient + 98. - + 98. - + 98. - + Gen02v01a
Clause  101:  + 01. Action + 05. Agent + 10a. Patient + 23. Time + 98. - + Gen02v02a
Clause  102:  + 01. Action + 10b. Product + 98. - + 98. - + 98. - + Gen02v02b
Clause  103:  + 01. Action + 10a. Patient + 23. Time + 98. - + 98. - + Gen02v02c
Clause  104:  + 01. Action + 10b. Product + 98. - + 98. - + 98. - + Gen02v02d
Clause  105:  + 01. Action + 05. Agent + 10a. Patient + 98. - + 98. - + Gen02v03a
Clause  106:  + 01. Action + 10a. Patient + 29. Reason + 98. - + 98. - + Gen02v03b
Clause  107:  + 01. Action + 10a. Patient + 23. Time + 98. - + 98. - + Gen02v03c
Clause  108:  + 01. Action + 05. Agent + 10b. Product + 19. Manner + 98. - + Gen02v03d
```

Figure 6.8. The logically-ordered semantic role frameworks of the 108 clauses.

These results are still not good enough since the researcher still has to do most of the analysis himself/herself. One now needs to order the rows as units with reference to each other.

6.4.3 Concatenation of each row

The collections of semantic functions in each row now have to be concatenated into single strings to facilitate sorting of the frames (regarded as units, similar to a set of words). This is the third step of this data-mining experiment, which is activated by pressing the button, "Concatenate each row to a string". The clause numbers and primary keys must be stored in a parallel dimension of the twodimensional array for reference purposes. The sorting is done only on the semantic frames. The dummy number 98, used to mark lacking functions, is removed before sorting, since all the dashes have already been moved to the end of the collections. (Not all clauses have five phrases, nor do all phrases have semantic functions.) The code shown in Figure 6.9 performs these functions.

```
'Concatenate each row

Private Sub btnConcat_Click()
Dim count1, count2 As Integer
Dim concatString

For countf21 = 1 To arrayMax
 For countf22 = 1 To 5 'Remove number 98 inserted for sorting of zero
    elements
   If arrSem1(countf21, countf22) = "98. -" Then
    arrSem1(countf21, countf22) = "-"
  End If
 Next
Next

Text1.Text = ""
concatString = "SEMANTIC FUNCTIONS OF EACH CLAUSE/ROW CONCATENATED TO FORM
   A UNIT" & vbCrLf

For count1 = 1 To arrayMax
 arrSem2(count1, 1) = arrSem1(count1, 1) & " + " & arrSem1(count1, 2) & " +
    " & arrSem1(count1, 3) & " + " & arrSem1(count1, 4) & " + " &
    arrSem1(count1, 5)
 arrSem2(count1, 2) = arrSem1(count1, 6) & " (Clause" & Str(count1) & ")"
 concatString = concatString & vbCrLf & arrSem2(count1, 1) & ". Reference:
    " & arrSem2(count1, 2)
Next
```

```
Text1.Text = concatString

End Sub
```

Figure 6.9. Code used to concatenate the semantic functions of each clause into a unit.

When this code runs the output is shown in a textbox (see Figure 6.10).

```
SEMANTIC FUNCTIONS OF EACH CLAUSE/ROW CONCATENATED TO FORM A UNIT

01. Action + 05. Agent + 10b. Product + 23. Time + -. Reference: Gen01v01a (Clause 1)
04. State + 09. Zero + 33. Classification + - + -. Reference: Gen01v02a (Clause 2)
09. Zero + 12. Location + - + - + -. Reference: Gen01v02b (Clause 3)
02. Position + 06. Positioner + 12. Location + - + -. Reference: Gen01v02c (Clause 4)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v03a (Clause 5)
04. State + 09. Zero + - + - + -. Reference: Gen01v03b (Clause 6)
04. State + 09. Zero + - + - + -. Reference: Gen01v03c (Clause 7)
03. Process + 08. Processed + 10a. Patient + - + -. Reference: Gen01v04a (Clause 8)
09. Zero + 34. Quality + - + - + -. Reference: Gen01v04b (Clause 9)
01. Action + 05. Agent + 10a. Patient + 14. Source + -. Reference: Gen01v04c (Clause 10)
01. Action + 05. Agent + 10a. Patient + 10b. Product + -. Reference: Gen01v05a (Clause 11)
01. Action + 10a. Patient + 10b. Product + - + -. Reference: Gen01v05b (Clause 12)
04. State + 09. Zero + - + - + -. Reference: Gen01v05c (Clause 13)
04. State + 09. Zero + 34. Quality + - + -. Reference: Gen01v05d (Clause 14)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v06a (Clause 15)
04. State + 09. Zero + 12. Location + - + -. Reference: Gen01v06b (Clause 16)
01. Action + 10a. Patient + 14. Source + - + -. Reference: Gen01v06c (Clause 17)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v07a (Clause 18)
01. Action + 10a. Patient + 14. Source + - + -. Reference: Gen01v07b (Clause 19)
09. Zero + 12. Location + - + - + -. Reference: Gen01v07c (Clause 20)
09. Zero + 12. Location + - + - + -. Reference: Gen01v07d (Clause 21)
04. State + 19. Manner + - + - + -. Reference: Gen01v07e (Clause 22)
01. Action + 05. Agent + 10a. Patient + 10b. Product + -. Reference: Gen01v08a (Clause 23)
04. State + 09. Zero + - + - + -. Reference: Gen01v08b (Clause 24)
04. State + 09. Zero + 34. Quality + - + -. Reference: Gen01v08c (Clause 25)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v09a (Clause 26)
03. Process + 08. Processed + 12. Location + - + -. Reference: Gen01v09b (Clause 27)
03. Process + 08. Processed + - + - + -. Reference: Gen01v09c (Clause 28)
04. State + 19. Manner + - + - + -. Reference: Gen01v09d (Clause 29)
01. Action + 05. Agent + 10a. Patient + 10b. Product + -. Reference: Gen01v10a (Clause 30)
01. Action + 10a. Patient + 10b. Product + - + -. Reference: Gen01v10b (Clause 31)
```

03. Process + 08. Processed + 10a. Patient + - + -. Reference: Gen01v10c (Clause 32)
34. Quality + - + - + - + -. Reference: Gen01v10d (Clause 33)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v11a (Clause 34)
03. Process + 08. Processed + 15. Reference + - + -. Reference: Gen01v11b (Clause 35)
09. Zero + 12. Location + - + - + -. Reference: Gen01v11c (Clause 36)
04. State + 19. Manner + - + - + -. Reference: Gen01v11d (Clause 37)
03. Process + 08. Processed + 10b. Product + 19. Manner + -. Reference: Gen01v12a (Clause 38)
09. Zero + 12. Location + - + - + -. Reference: Gen01v12b (Clause 39)
03. Process + 08. Processed + 10a. Patient + - + -. Reference: Gen01v12c (Clause 40)
34. Quality + - + - + - + -. Reference: Gen01v12d (Clause 41)
04. State + 09. Zero + - + - + -. Reference: Gen01v13a (Clause 42)
04. State + 09. Zero + 34. Quality + - + -. Reference: Gen01v13b (Clause 43)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v14a (Clause 44)
04. State + 09. Zero + 12. Location + 28. Purpose + -. Reference: Gen01v14b (Clause 45)
04. State + 33. Classification + - + - + -. Reference: Gen01v14c (Clause 46)
04. State + 12. Location + 28. Purpose + 33. Classification + -. Reference: Gen01v15a (Clause 47)
04. State + 19. Manner + - + - + -. Reference: Gen01v15b (Clause 48)
01. Action + 05. Agent + 10b. Product + 28. Purpose + -. Reference: Gen01v16a (Clause 49)
01. Action + 05. Agent + 10a. Patient + 12. Location + 28. Purpose. Reference: Gen01v17a (Clause 50)
03. Process + 08. Processed + 10a. Patient + - + -. Reference: Gen01v18b (Clause 51)
34. Quality + - + - + - + -. Reference: Gen01v18d (Clause 52)
04. State + 09. Zero + - + - + -. Reference: Gen01v19a (Clause 53)
04. State + 09. Zero + 34. Quality + - + -. Reference: Gen01v19b (Clause 54)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v20a (Clause 55)
03. Process + 08. Processed + 15. Reference + - + -. Reference: Gen01v20b (Clause 56)
01. Action + 05. Agent + 12. Location + 12. Location + -. Reference: Gen01v20c (Clause 57)
01. Action + 05. Agent + 10b. Product + 19. Manner + -. Reference: Gen01v21a (Clause 58)
01. Action + 05. Agent + 12. Location + - + -. Reference: Gen01v21b (Clause 59)
03. Process + 08. Processed + 10a. Patient + - + -. Reference: Gen01v21c (Clause 60)
34. Quality + - + - + - + -. Reference: Gen01v21d (Clause 61)
01. Action + 05. Agent + 10a. Patient + 19. Manner + -. Reference: Gen01v22a (Clause 62)
04. State + - + - + - + -. Reference: Gen01v22b (Clause 63)
03. Process + - + - + - + -. Reference: Gen01v22c (Clause 64)
01. Action + 10a. Patient + - + - + -. Reference: Gen01v22d (Clause 65)
03. Process + 08. Processed + 12. Location + - + -. Reference: Gen01v22e (Clause 66)
04. State + 09. Zero + - + - + -. Reference: Gen01v23a (Clause 67)
04. State + 09. Zero + 34. Quality + - + -. Reference: Gen01v23b (Clause 68)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v24a (Clause 69)
03. Process + 08. Processed + 10b. Product + 19. Manner + -. Reference: Gen01v24b (Clause 70)
04. State + 19. Manner + - + - + -. Reference: Gen01v24c (Clause 71)
01. Action + 05. Agent + 10b. Product + 19. Manner + -. Reference: Gen01v25a (Clause 72)
03. Process + 08. Processed + 10a. Patient + - + -. Reference: Gen01v25b (Clause 73)
34. Quality + - + - + - + -. Reference: Gen01v25c (Clause 74)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v26a (Clause 75)
01. Action + 10b. Product + 19. Manner + - + -. Reference: Gen01v26b (Clause 76)

```
05. Agent + 10a. Patient + - + - + -. Reference: Gen01v26c (Clause 77)
01. Action + 05. Agent + 10b. Product + 19. Manner + -. Reference: Gen01v27a (Clause 78)
01. Action + 10b. Product + 19. Manner + - + -. Reference: Gen01v27b (Clause 79)
01. Action + 10b. Product + 34. Quality + - + -. Reference: Gen01v27c (Clause 80)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v28a (Clause 81)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v28b (Clause 82)
04. State + - + - + - + -. Reference: Gen01v28c (Clause 83)
03. Process + - + - + - + -. Reference: Gen01v28d (Clause 84)
01. Action + 10a. Patient + - + - + -. Reference: Gen01v28e (Clause 85)
01. Action + - + - + - + -. Reference: Gen01v28f (Clause 86)
01. Action + 10a. Patient + - + - + -. Reference: Gen01v28g (Clause 87)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v29a (Clause 88)
01. Action + 10a. Patient + 11. Receiver + - + -. Reference: Gen01v29b (Clause 89)
09. Zero + 12. Location + - + - + -. Reference: Gen01v29c (Clause 90)
09. Zero + 09. Zero + 12. Location + - + -. Reference: Gen01v29d (Clause 91)
04. State + 16. Beneficiary + 28. Purpose + - + -. Reference: Gen01v29e (Clause 92)
09. Zero + 12. Location + - + - + -. Reference: Gen01v30b (Clause 93)
04. State + 19. Manner + - + - + -. Reference: Gen01v30c (Clause 94)
03. Process + 08. Processed + 10a. Patient + - + -. Reference: Gen01v31a (Clause 95)
01. Action + 10b. Product + - + - + -. Reference: Gen01v31b (Clause 96)
34. Quality + - + - + - + -. Reference: Gen01v31c (Clause 97)
04. State + 09. Zero + - + - + -. Reference: Gen01v31d (Clause 98)
04. State + 09. Zero + 34. Quality + - + -. Reference: Gen01v31e (Clause 99)
01. Action + 10a. Patient + - + - + -. Reference: Gen02v01a (Clause 100)
01. Action + 05. Agent + 10a. Patient + 23. Time + -. Reference: Gen02v02a (Clause 101)
01. Action + 10b. Product + - + - + -. Reference: Gen02v02b (Clause 102)
01. Action + 10a. Patient + 23. Time + - + -. Reference: Gen02v02c (Clause 103)
01. Action + 10b. Product + - + - + -. Reference: Gen02v02d (Clause 104)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen02v03a (Clause 105)
01. Action + 10a. Patient + 29. Reason + - + -. Reference: Gen02v03b (Clause 106)
01. Action + 10a. Patient + 23. Time + - + -. Reference: Gen02v03c (Clause 107)
01. Action + 05. Agent + 10b. Product + 19. Manner + -. Reference: Gen02v03d (Clause 108)
```

Figure 6.10. Concatenated semantic role frameworks of the 108 clauses.

This output appears to be very similar to that of the previous step, but the underlying structure is actually quite different because the semantic frames have been merged before the printing phase into single strings, one for each clause. Each clause's frame is now similar to a single word, which can be ordered easily.

6.4.4 Ordering the rows as units with reference to each other

The fourth step of this data-mining experiment is activated by pressing the button, "Order rows". It activates a procedure that orders the rows of semantic frames, which have been merged into single strings, using the code shown in Figure 6.11.

```
' Order the rows with reference to each other to group unique frames

Private Sub cmdOrderRows_Click()

Dim smallest, smallest2, tempString As String 'temp holder
Dim curSmallPos, numEls, i, j, a As Integer

numEls = arrayMax
curSmallPos = 1 'Cf Robertson (2000: 265-266) for the selection-sort
algorithm

While curSmallPos <= (numEls - 1)
 i = curSmallPos
 smallest = arrSem2(i, 1)
 smallest2 = arrSem2(i, 2)
 j = i + 1

 While j <= numEls
  If arrSem2(j, 1) < smallest Then
   i = j
   smallest = arrSem2(j, 1)
   smallest2 = arrSem2(j, 2)
  End If
  j = j + 1
 Wend

 arrSem2(i, 1) = arrSem2(curSmallPos, 1)
 arrSem2(i, 2) = arrSem2(curSmallPos, 2)
 arrSem2(curSmallPos, 1) = smallest
 arrSem2(curSmallPos, 2) = smallest2
 curSmallPos = curSmallPos + 1

Wend

'Reorder subsets
'Read contents of arrSem2 and add a merged frame and reference into
arrSem3(1 to arrayMax, 1 to 3)
```

```
Dim rs1, rs2 As Integer
Dim arrSem3(1 To 200, 1 To 3)
For rs1 = 1 To arrayMax
 For rs2 = 1 To 2
  arrSem3(rs1, rs2) = arrSem2(rs1, rs2)
 Next
 arrSem3(rs1, 3) = arrSem2(rs1, 1) & " " & arrSem2(rs1, 2)
Next

'Repeat selection sort on merged unit
Dim smallest3, b As Integer
curSmallPos  =  1  'Cf  Robertson  (2000:  265-266)  for  the  selection-sort
algorithm


While curSmallPos <= (numEls - 1)
 i = curSmallPos
 smallest = arrSem3(i, 1)
 smallest2 = arrSem3(i, 2)
 smallest3 = arrSem3(i, 3)
 j = i + 1
 While j <= numEls
  If arrSem3(j, 3) < smallest3 Then
   i = j
   smallest = arrSem3(j, 1)
   smallest2 = arrSem3(j, 2)
   smallest3 = arrSem3(j, 3)
  End If
  j = j + 1
 Wend
 arrSem3(i, 1) = arrSem3(curSmallPos, 1)
 arrSem3(i, 2) = arrSem3(curSmallPos, 2)
 arrSem3(i, 3) = arrSem3(curSmallPos, 3)
 arrSem3(curSmallPos, 1) = smallest
 arrSem3(curSmallPos, 2) = smallest2
 arrSem3(curSmallPos, 3) = smallest3
 curSmallPos = curSmallPos + 1
Wend

'Read contents of arrSem3 back to arrSem2
For rs1 = 1 To arrayMax
 For rs2 = 1 To 2
  arrSem2(rs1, rs2) = arrSem3(rs1, rs2)
 Next
Next
```

```
'Print ordered array
Text1.Text() = ""
tempString = "SEMANTIC FUNCTION FRAMES OF ALL CLAUSES ORDERED LOGICALLY WRT
EACH OTHER" & vbCrLf
For a = 1 To arrayMax
 tempString = tempString & vbCrLf & arrSem2(a, 1) & ". Reference: " &
arrSem2(a, 2)
Next
 Text1.Text() = tempString

End Sub
```

Figure 6.11. VB6 code used to order the rows, representing individual semantic role frames, as units with reference to each other.

Running this section of code produces the output in Figure 6.12 showing that similar frames are now ordered contiguously.

```
SEMANTIC FUNCTION FRAMES OF ALL CLAUSES ORDERED LOGICALLY WRT EACH OTHER

01. Action + - + - + - + -. Reference: Gen01v28f (Clause 86)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v03a (Clause 5)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v06a (Clause 15)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v07a (Clause 18)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v09a (Clause 26)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v11a (Clause 34)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v14a (Clause 44)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v20a (Clause 55)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v24a (Clause 69)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v26a (Clause 75)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v28a (Clause 81)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v28b (Clause 82)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen01v29a (Clause 88)
01. Action + 05. Agent + 10a. Patient + - + -. Reference: Gen02v03a (Clause 105)
01. Action + 05. Agent + 10a. Patient + 10b. Product + -. Reference: Gen01v05a (Clause 11)
01. Action + 05. Agent + 10a. Patient + 10b. Product + -. Reference: Gen01v08a (Clause 23)
01. Action + 05. Agent + 10a. Patient + 10b. Product + -. Reference: Gen01v10a (Clause 30)
01. Action + 05. Agent + 10a. Patient + 12. Location + 28. Purpose. Reference: Gen01v17a (Clause
50)
01. Action + 05. Agent + 10a. Patient + 14. Source + -. Reference: Gen01v04c (Clause 10)
```

01. Action + 05. Agent + 10a. Patient + 19. Manner + -. Reference: Gen01v22a (Clause 62)
01. Action + 05. Agent + 10a. Patient + 23. Time + -. Reference: Gen02v02a (Clause 101)
01. Action + 05. Agent + 10b. Product + 19. Manner + -. Reference: Gen01v21a (Clause 58)
01. Action + 05. Agent + 10b. Product + 19. Manner + -. Reference: Gen01v25a (Clause 72)
01. Action + 05. Agent + 10b. Product + 19. Manner + -. Reference: Gen01v27a (Clause 78)
01. Action + 05. Agent + 10b. Product + 19. Manner + -. Reference: Gen02v03d (Clause 108)
01. Action + 05. Agent + 10b. Product + 23. Time + -. Reference: Gen01v01a (Clause 1)
01. Action + 05. Agent + 10b. Product + 28. Purpose + -. Reference: Gen01v16a (Clause 49)
01. Action + 05. Agent + 12. Location + - + -. Reference: Gen01v21b (Clause 59)
01. Action + 05. Agent + 12. Location + 12. Location + -. Reference: Gen01v20c (Clause 57)
01. Action + 10a. Patient + - + - + -. Reference: Gen01v22d (Clause 65)
01. Action + 10a. Patient + - + - + -. Reference: Gen01v28e (Clause 85)
01. Action + 10a. Patient + - + - + -. Reference: Gen01v28g (Clause 87)
01. Action + 10a. Patient + - + - + -. Reference: Gen02v01a (Clause 100)
01. Action + 10a. Patient + 10b. Product + - + -. Reference: Gen01v05b (Clause 12)
01. Action + 10a. Patient + 10b. Product + - + -. Reference: Gen01v10b (Clause 31)
01. Action + 10a. Patient + 11. Receiver + - + -. Reference: Gen01v29b (Clause 89)
01. Action + 10a. Patient + 14. Source + - + -. Reference: Gen01v06c (Clause 17)
01. Action + 10a. Patient + 14. Source + - + -. Reference: Gen01v07b (Clause 19)
01. Action + 10a. Patient + 23. Time + - + -. Reference: Gen02v02c (Clause 103)
01. Action + 10a. Patient + 23. Time + - + -. Reference: Gen02v03c (Clause 107)
01. Action + 10a. Patient + 29. Reason + - + -. Reference: Gen02v03b (Clause 106)
01. Action + 10b. Product + - + - + -. Reference: Gen01v31b (Clause 96)
01. Action + 10b. Product + - + - + -. Reference: Gen02v02b (Clause 102)
01. Action + 10b. Product + - + - + -. Reference: Gen02v02d (Clause 104)
01. Action + 10b. Product + 19. Manner + - + -. Reference: Gen01v26b (Clause 76)
01. Action + 10b. Product + 19. Manner + - + -. Reference: Gen01v27b (Clause 79)
01. Action + 10b. Product + 34. Quality + - + -. Reference: Gen01v27c (Clause 80)
02. Position + 06. Positioner + 12. Location + - + -. Reference: Gen01v02c (Clause 4)
03. Process + - + - + - + -. Reference: Gen01v22c (Clause 64)
03. Process + - + - + - + -. Reference: Gen01v28d (Clause 84)
03. Process + 08. Processed + - + - + -. Reference: Gen01v09c (Clause 28)
03. Process + 08. Processed + 10a. Patient + - + -. Reference: Gen01v04a (Clause 8)
03. Process + 08. Processed + 10a. Patient + - + -. Reference: Gen01v10c (Clause 32)
03. Process + 08. Processed + 10a. Patient + - + -. Reference: Gen01v12c (Clause 40)
03. Process + 08. Processed + 10a. Patient + - + -. Reference: Gen01v18b (Clause 51)
03. Process + 08. Processed + 10a. Patient + - + -. Reference: Gen01v21c (Clause 60)
03. Process + 08. Processed + 10a. Patient + - + -. Reference: Gen01v25b (Clause 73)
03. Process + 08. Processed + 10a. Patient + - + -. Reference: Gen01v31a (Clause 95)
03. Process + 08. Processed + 10b. Product + 19. Manner + -. Reference: Gen01v12a (Clause 38)
03. Process + 08. Processed + 10b. Product + 19. Manner + -. Reference: Gen01v24b (Clause 70)
03. Process + 08. Processed + 12. Location + - + -. Reference: Gen01v09b (Clause 27)
03. Process + 08. Processed + 12. Location + - + -. Reference: Gen01v22e (Clause 66)
03. Process + 08. Processed + 15. Reference + - + -. Reference: Gen01v11b (Clause 35)
03. Process + 08. Processed + 15. Reference + - + -. Reference: Gen01v20b (Clause 56)
04. State + - + - + - + -. Reference: Gen01v22b (Clause 63)

04. State + - + - + - + -. Reference: Gen01v28c (Clause 83)
04. State + 09. Zero + - + - + -. Reference: Gen01v03b (Clause 6)
04. State + 09. Zero + - + - + -. Reference: Gen01v03c (Clause 7)
04. State + 09. Zero + - + - + -. Reference: Gen01v05c (Clause 13)
04. State + 09. Zero + - + - + -. Reference: Gen01v08b (Clause 24)
04. State + 09. Zero + - + - + -. Reference: Gen01v13a (Clause 42)
04. State + 09. Zero + - + - + -. Reference: Gen01v19a (Clause 53)
04. State + 09. Zero + - + - + -. Reference: Gen01v23a (Clause 67)
04. State + 09. Zero + - + - + -. Reference: Gen01v31d (Clause 98)
04. State + 09. Zero + 12. Location + - + -. Reference: Gen01v06b (Clause 16)
04. State + 09. Zero + 12. Location + 28. Purpose + -. Reference: Gen01v14b (Clause 45)
04. State + 09. Zero + 33. Classification + - + -. Reference: Gen01v02a (Clause 2)
04. State + 09. Zero + 34. Quality + - + -. Reference: Gen01v05d (Clause 14)
04. State + 09. Zero + 34. Quality + - + -. Reference: Gen01v08c (Clause 25)
04. State + 09. Zero + 34. Quality + - + -. Reference: Gen01v13b (Clause 43)
04. State + 09. Zero + 34. Quality + - + -. Reference: Gen01v19b (Clause 54)
04. State + 09. Zero + 34. Quality + - + -. Reference: Gen01v23b (Clause 68)
04. State + 09. Zero + 34. Quality + - + -. Reference: Gen01v31e (Clause 99)
04. State + 12. Location + 28. Purpose + 33. Classification + -. Reference: Gen01v15a (Clause 47)
04. State + 16. Beneficiary + 28. Purpose + - + -. Reference: Gen01v29e (Clause 92)
04. State + 19. Manner + - + - + -. Reference: Gen01v07e (Clause 22)
04. State + 19. Manner + - + - + -. Reference: Gen01v09d (Clause 29)
04. State + 19. Manner + - + - + -. Reference: Gen01v11d (Clause 37)
04. State + 19. Manner + - + - + -. Reference: Gen01v15b (Clause 48)
04. State + 19. Manner + - + - + -. Reference: Gen01v24c (Clause 71)
04. State + 19. Manner + - + - + -. Reference: Gen01v30c (Clause 94)
04. State + 33. Classification + - + - + -. Reference: Gen01v14c (Clause 46)
05. Agent + 10a. Patient + - + - + -. Reference: Gen01v26c (Clause 77)
09. Zero + 09. Zero + 12. Location + - + -. Reference: Gen01v29d (Clause 91)
09. Zero + 12. Location + - + - + -. Reference: Gen01v02b (Clause 3)
09. Zero + 12. Location + - + - + -. Reference: Gen01v07c (Clause 20)
09. Zero + 12. Location + - + - + -. Reference: Gen01v07d (Clause 21)
09. Zero + 12. Location + - + - + -. Reference: Gen01v11c (Clause 36)
09. Zero + 12. Location + - + - + -. Reference: Gen01v12b (Clause 39)
09. Zero + 12. Location + - + - + -. Reference: Gen01v29c (Clause 90)
09. Zero + 12. Location + - + - + -. Reference: Gen01v30b (Clause 93)
09. Zero + 34. Quality + - + - + -. Reference: Gen01v04b (Clause 9)
34. Quality + - + - + - + -. Reference: Gen01v10d (Clause 33)
34. Quality + - + - + - + -. Reference: Gen01v12d (Clause 41)
34. Quality + - + - + - + -. Reference: Gen01v18d (Clause 52)
34. Quality + - + - + - + -. Reference: Gen01v21d (Clause 61)
34. Quality + - + - + - + -. Reference: Gen01v25c (Clause 74)
34. Quality + - + - + - + -. Reference: Gen01v31c (Clause 97)

Figure 6.12. Semantic role frames of all clauses ordered logically with regard to each other.

This output is already usable in a small dataset, but if a larger dataset were used it would become very verbose and clumsy due to the repetition of similar frames.

6.4.5 Identifying and counting unique semantic frameworks

The last two steps in analysing the semantic frameworks are the identification of unique combinations and the calculation of the frequencies of each unique frame. This step is activated by pressing the button, "Count unique frames". The code in Figure 6.13 uses parallel arrays to store information on the unique frames, their frequencies and references to the individual instances of each. This statistical data may be used to indicate which patterns occur frequently or rarely.

```
'Count unique frames and build reference strings - assume that frames have
    been sorted
Private Sub cmdCountFrames_Click()

'Use three parallel arrays to store unique frames, counts and references:
Dim arr2(200) As String 'Store unique strings/frames
Dim arr3(200) As Integer 'Count occurrences of unique string/frames
Dim arr4(200) As String 'Store indexes of occurrences/references
Dim a, i, j, k As Integer
Dim temp2 As String

arr2(1) = arrSem2(1, 1)
i = 1 'Value unchanged while rows match - place in new array
arr3(i) = 1
k = 1 'Value change when new row found - new value to be compared
arr4(i) = arrSem2(1, 2)

For j = 2 To arrayMax
 If arrSem2(j, 1) = arrSem2(k, 1) Then
  arr3(i) = arr3(i) + 1
  arr4(i) = arr4(i) & " " & vbCrLf & "      " & arrSem2(j, 2)
 Else
  i = i + 1
  arr2(i) = arrSem2(j, 1)
  arr3(i) = 1
  arr4(i) = arr4(i) & " " & arrSem2(j, 2)
```

```
   k = j
  End If
Next

Text1.Text() = ""
temp2 = "UNIQUE SEMANTIC FUNCTION FRAMES, NUMBER OF OCCURRENCES AND CLAUSE
   REFERENCES"
For a = 1 To i
 temp2 = temp2 & vbCrLf & vbCrLf & "Frame " & a & ": " & arr2(a) & _
 vbCrLf & "No of occurrences: " & arr3(a) & vbCrLf & "References: " &
arr4(a)
Next
Text1.Text = temp2

End Sub
```

Figure 6.13. VB6 code used to identify and count unique semantic role frames.

The results produced by running this code are shown in Figure 6.14.

```
UNIQUE SEMANTIC FUNCTION FRAMES, NUMBER OF OCCURRENCES AND CLAUSE
REFERENCES

Frame 1: 01. Action + - + - + - + -
No of occurrences: 1
References: Gen01v28f (Clause 86)

Frame 2: 01. Action + 05. Agent + 10a. Patient + - + -
No of occurrences: 13
References:  Gen01v03a (Clause 5)
             Gen01v06a (Clause 15)
             Gen01v07a (Clause 18)
             Gen01v09a (Clause 26)
             Gen01v11a (Clause 34)
             Gen01v14a (Clause 44)
             Gen01v20a (Clause 55)
             Gen01v24a (Clause 69)
             Gen01v26a (Clause 75)
             Gen01v28a (Clause 81)
             Gen01v28b (Clause 82)
             Gen01v29a (Clause 88)
             Gen02v03a (Clause 105)
```

Frame 3: 01. Action + 05. Agent + 10a. Patient + 10b. Product + -
No of occurrences: 3
References: Gen01v05a (Clause 11)
Gen01v08a (Clause 23)
Gen01v10a (Clause 30)

Frame 4: 01. Action + 05. Agent + 10a. Patient + 12. Location + 28. Purpose
No of occurrences: 1
References: Gen01v17a (Clause 50)

Frame 5: 01. Action + 05. Agent + 10a. Patient + 14. Source + -
No of occurrences: 1
References: Gen01v04c (Clause 10)

Frame 6: 01. Action + 05. Agent + 10a. Patient + 19. Manner + -
No of occurrences: 1
References: Gen01v22a (Clause 62)

Frame 7: 01. Action + 05. Agent + 10a. Patient + 23. Time + -
No of occurrences: 1
References: Gen02v02a (Clause 101)

Frame 8: 01. Action + 05. Agent + 10b. Product + 19. Manner + -
No of occurrences: 4
References: Gen01v21a (Clause 58)
Gen01v25a (Clause 72)
Gen01v27a (Clause 78)
Gen02v03d (Clause 108)

Frame 9: 01. Action + 05. Agent + 10b. Product + 23. Time + -
No of occurrences: 1
References: Gen01v01a (Clause 1)

Frame 10: 01. Action + 05. Agent + 10b. Product + 28. Purpose + -
No of occurrences: 1
References: Gen01v16a (Clause 49)

Frame 11: 01. Action + 05. Agent + 12. Location + - + -
No of occurrences: 1
References: Gen01v21b (Clause 59)

Frame 12: 01. Action + 05. Agent + 12. Location + 12. Location + -
No of occurrences: 1
References: Gen01v20c (Clause 57)

Frame 13: 01. Action + 10a. Patient + - + - + -
No of occurrences: 4
References: Gen01v22d (Clause 65)
Gen01v28e (Clause 85)
Gen01v28g (Clause 87)
Gen02v01a (Clause 100)

Frame 14: 01. Action + 10a. Patient + 10b. Product + - + -
No of occurrences: 2
References: Gen01v05b (Clause 12)
Gen01v10b (Clause 31)

Frame 15: 01. Action + 10a. Patient + 11. Receiver + - + -
No of occurrences: 1
References: Gen01v29b (Clause 89)

Frame 16: 01. Action + 10a. Patient + 14. Source + - + -
No of occurrences: 2
References: Gen01v06c (Clause 17)
Gen01v07b (Clause 19)

Frame 17: 01. Action + 10a. Patient + 23. Time + - + -
No of occurrences: 2
References: Gen02v02c (Clause 103)
Gen02v03c (Clause 107)

Frame 18: 01. Action + 10a. Patient + 29. Reason + - + -
No of occurrences: 1
References: Gen02v03b (Clause 106)

Frame 19: 01. Action + 10b. Product + - + - + -
No of occurrences: 3
References: Gen01v31b (Clause 96)
Gen02v02b (Clause 102)
Gen02v02d (Clause 104)

Frame 20: 01. Action + 10b. Product + 19. Manner + - + -
No of occurrences: 2
References: Gen01v26b (Clause 76)
Gen01v27b (Clause 79)

Frame 21: 01. Action + 10b. Product + 34. Quality + - + -
No of occurrences: 1
References: Gen01v27c (Clause 80)

Frame 22: 02. Position + 06. Positioner + 12. Location + - + -

No of occurrences: 1
References: Gen01v02c (Clause 4)

Frame 23: 03. Process + - + - + - + -
No of occurrences: 2
References: Gen01v22c (Clause 64)
Gen01v28d (Clause 84)

Frame 24: 03. Process + 08. Processed + - + - + -
No of occurrences: 1
References: Gen01v09c (Clause 28)

Frame 25: 03. Process + 08. Processed + 10a. Patient + - + -
No of occurrences: 7
References: Gen01v04a (Clause 8)
Gen01v10c (Clause 32)
Gen01v12c (Clause 40)
Gen01v18b (Clause 51)
Gen01v21c (Clause 60)
Gen01v25b (Clause 73)
Gen01v31a (Clause 95)

Frame 26: 03. Process + 08. Processed + 10b. Product + 19. Manner + -
No of occurrences: 2
References: Gen01v12a (Clause 38)
Gen01v24b (Clause 70)

Frame 27: 03. Process + 08. Processed + 12. Location + - + -
No of occurrences: 2
References: Gen01v09b (Clause 27)
Gen01v22e (Clause 66)

Frame 28: 03. Process + 08. Processed + 15. Reference + - + -
No of occurrences: 2
References: Gen01v11b (Clause 35)
Gen01v20b (Clause 56)

Frame 29: 04. State + - + - + - + -
No of occurrences: 2
References: Gen01v22b (Clause 63)
Gen01v28c (Clause 83)

Frame 30: 04. State + 09. Zero + - + - + -
No of occurrences: 8
References: Gen01v03b (Clause 6)
Gen01v03c (Clause 7)

Gen01v05c (Clause 13)
Gen01v08b (Clause 24)
Gen01v13a (Clause 42)
Gen01v19a (Clause 53)
Gen01v23a (Clause 67)
Gen01v31d (Clause 98)

Frame 31: 04. State + 09. Zero + 12. Location + - + -
No of occurrences: 1
References: Gen01v06b (Clause 16)

Frame 32: 04. State + 09. Zero + 12. Location + 28. Purpose + -
No of occurrences: 1
References: Gen01v14b (Clause 45)

Frame 33: 04. State + 09. Zero + 33. Classification + - + -
No of occurrences: 1
References: Gen01v02a (Clause 2)

Frame 34: 04. State + 09. Zero + 34. Quality + - + -
No of occurrences: 6
References: Gen01v05d (Clause 14)
Gen01v08c (Clause 25)
Gen01v13b (Clause 43)
Gen01v19b (Clause 54)
Gen01v23b (Clause 68)
Gen01v31e (Clause 99)

Frame 35: 04. State + 12. Location + 28. Purpose + 33. Classification + -
No of occurrences: 1
References: Gen01v15a (Clause 47)

Frame 36: 04. State + 16. Beneficiary + 28. Purpose + - + -
No of occurrences: 1
References: Gen01v29e (Clause 92)

Frame 37: 04. State + 19. Manner + - + - + -
No of occurrences: 6
References: Gen01v07e (Clause 22)
Gen01v09d (Clause 29)
Gen01v11d (Clause 37)
Gen01v15b (Clause 48)
Gen01v24c (Clause 71)
Gen01v30c (Clause 94)

Frame 38: 04. State + 33. Classification + - + - + -

No of occurrences: 1
References: Gen01v14c (Clause 46)

Frame 39: 05. Agent + 10a. Patient + - + - + -
No of occurrences: 1
References: Gen01v26c (Clause 77)

Frame 40: 09. Zero + 09. Zero + 12. Location + - + -
No of occurrences: 1
References: Gen01v29d (Clause 91)

Frame 41: 09. Zero + 12. Location + - + - + -
No of occurrences: 7
References: Gen01v02b (Clause 3)
Gen01v07c (Clause 20)
Gen01v07d (Clause 21)
Gen01v11c (Clause 36)
Gen01v12b (Clause 39)
Gen01v29c (Clause 90)
Gen01v30b (Clause 93)

Frame 42: 09. Zero + 34. Quality + - + - + -
No of occurrences: 1
References: Gen01v04b (Clause 9)

Frame 43: 34. Quality + - + - + - + -
No of occurrences: 6
References: Gen01v10d (Clause 33)
Gen01v12d (Clause 41)
Gen01v18d (Clause 52)
Gen01v21d (Clause 61)
Gen01v25c (Clause 74)
Gen01v31c (Clause 97)

Figure 6.14. The final results of the semantic role framework experiment.

These results provide the researcher with good information, which may now be used to confirm or falsify existing hypotheses, or to create new ones.

The following frames were to be expected and confirm existing definitions of semantic functions: Frames 1, 3, 4, 5, 6, 8, 11, 22, 24, 27, 28, 30, 31, 33, 41, 42. In Frame 2, Gen. 1:28a and Gen. 2:3a were tagged correctly.

The following frames are interesting but do not contradict existing definitions of semantic functions:

- Two location satellites: Frame 12
- Patient after process verb: Frame 25
- Zero assumed (elliptic): Frame 43

The following frames are interesting but at closer inspection reveal deficiencies in the tagging scheme and prompts better (more detailed) coding:

- Agent included in action verb: Frames 13-21
- Agent and product included in action verb: Frame 1
- Processed included in process verb: Frame 23
- Zero included in state verb: Frame 29, 35-38

While working through the data according to the identified patterns a number of analysis and tagging mistakes or inconsistencies were found. These inconsistencies have been retained in order to demonstrate the rigour that is enforced by a computer-assisted analysis:[122]

- Frame 39 is incorrectly tagged: "Agent" should be replaced with "Action". Verse 1:26c is actually another example of Frame 13, which itself should be tagged in more detail.
- The semantic function of the direct object as patient should be changed to product in verses 1:7a, 2:2a (cf. Frames 2, 7).
- The semantic function of direct speech as direct object should be changed from patient to product in verses 1:3a, 1:6a, 1:9a, 1:11a, 1:14a, 1:20a, 1:24a, 1:26a, 1:28b, 1:29a (cf. Frame 2).
- Inconsistencies regarding the tagging format of embedded phrases should be corrected: see verses 1:12a, 1:16a, 1:21a, 1:24b, 1:25a (cf. Frame 8, 10, 26).

[122] If these inconsistencies were corrected in the XML clause cube, the results of the data mining experiment would have been different.

- Inconsistencies regarding the tagging format of embedded clauses and clauses spanning two verses should be corrected: see verses 1:3a, 1:4a, 1:17a-18a, 1:29b (cf. Frames 2, 4, 15, 25).
- Processed should be replaced with force in verses 1:12a and 1:24b (Frame 26).
- Zero should be changed to classification in verses 1:5d, 1:8c, 1:13b, 1:19b, 1:23b, 1:31e (Frame 34).
- Purpose should be changed to classification in verses 1:16a, 1:29e (cf. Frame 10, 36).
- The zero allocated to a relative pronoun should be changed to a dash (-) in verse 29d (cf. Frame 40).

The most important outcome of the data-mining venture is that a small number of frames were found that contradict existing definitions of semantic functions:

- **Purpose in state:** Frames 32, 35 (vs. 1:14b, 1:15a)
  According to Dik (1997a:244) a purpose satellite can only be found in a controlled predication. The examples found in Genesis 1:14b and 15a ("to be/exist in order to separate/shine") clearly shows that in BH a state predicate may contain an embedded predication with the semantic function of purpose.
- **Quality after product:** Frame 21 (vs. 1:27c)
  According to Dik (1997a:214) the semantic function of "property assignment" (here called "quality") is restricted to non-verbal predicates (adjectives or bare nominals). One should, however, consider assigning this role also to some adjectival phrases, such as in this example ("*male and female* he created them"). (Cf. the discussion in 6.5.)
- **Quality after classification:** Frame 34 (vs. 1:5d, 1:8c, 1:13b, 1:19b, 1:23b, 1:31e)
  These examples are similar to the previous case, since they contain phrases in apposition that are regarded as adjectival modifiers. Dik (1997a:62) would regard these examples as term operators or restrictors. More loosely coupled than mere attributes, they highlight the need to assign the quality role to phrases in apposition (e.g. "and it was morning, *day one*"). (Cf. the discussion in 6.5.)

- **Manner in state:** Frame 37 (vs. 1:7e, 1:9d, 1:11d, 1:15b, 1:24c, 1:30c)
  According to Dik (1997a:230), manner satellites occur in actions, positions and processes. Although this recurring example ("and it was so") is not a very strong one, it does prompt the grammarian to reconsider the possibility of manner occurring in states as well. (Cf. the discussion in 6.5.)

These instances should be reconsidered and either the tagging should be changed or the definitions should be adjusted. It may be concluded that the patterns identified by the data-mining process proved to be very useful in the knowledge construction process of checking and refining the definitions of semantic functions.

6.5 Analysis of the mapping of syntactic and semantic functions as another linguistic data-mining venture

The study of the mapping of various linguistic modules has been highlighted by a number of scholars as research that may significantly benefit from a computer-assisted approach. Compare Witt (2002), who highlights the importance of being able to discover the relations between different tiers of annotation as a part of the computation of a linguistic knowledge representation. "But the interrelations of annotation layers are of interest for many persons concerned with structuring and modelling of information" (Witt, 2005: 64). Bayerl et al. (2003) compared, for example, the structural, thematic and rhetorical levels of a corpus of scientific texts. According to Burnard (2004) studying the interplay of analyses of the various language modules is crucial for many grammatical and literary research studies, for example "the extent to which syntactic structure and narrative structure mesh, or fail to mesh, ... or the extent to which phonological structures reflect morphology". These authors all identified mark-up systems, such as XML, as serviceable technology for the study of linguistic mapping.

The levels may be differentiated due to various perspectives on the same data, such as phonology, morphology, syntax, semantics, or the annotations may differ, even on the same language module, due to different linguistic or philosophical theories underlying the analyses (cf. Witt, 2005: 55, 64). For example, the annotation of the syntactic level may differ depending on whether it is based on Lexical Functional Grammar or Head-Driven Phrase Structure Grammar (Witt et al., 2005: 103).

Witt (2005:79) also indicated the importance of using the power of computers to test hypotheses. He suggested the use of the basic text structure to associate various layers of linguistic analyses to discover meta-relations between the tiers: "Moreover, with these meta-relations, generalizations stated by researchers or inferred automatically on a small empirical basis can be falsified."

In this section the mapping of the semantic layer onto the syntactic layer in Genesis 1:1-2:3 will be explored. A VB6 program extracts the relevant data from the clause cube and manipulates the data into coherent, human-friendly information. The researcher then uses this information to test some existing assumptions and hypotheses about Biblical Hebrew syntax and semantics.

When the user presses the button, "Analyse semantic to syntactic mapping", on the main form (Figure 6.2), he/she is directed to a new window having five buttons that simulate the logical steps of this data-mining experiment (see Figure 6.15):

- Clean data and number the syntactic functions according to a logical order
- Clean data and number the semantic functions according to a logical order
- Calculate frequencies of possible combinations and store references
- List tested combinations
- List combinations found with their frequencies

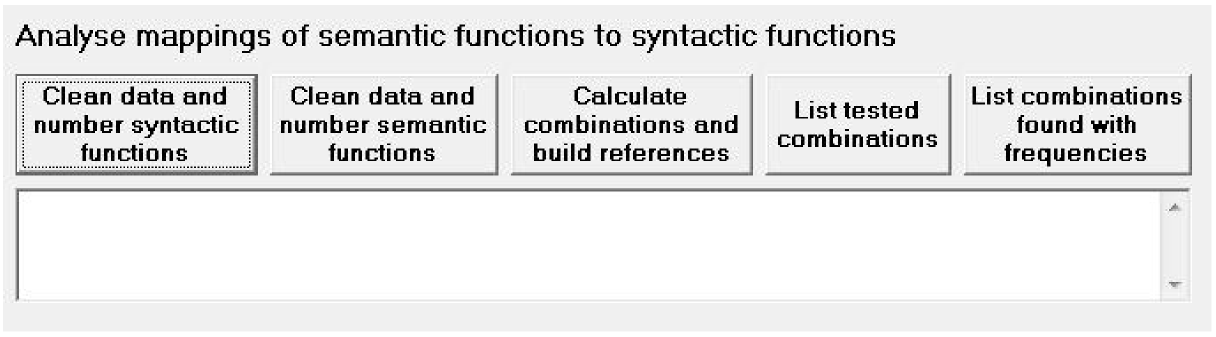

Analyse mappings of semantic functions to syntactic functions
Clean data and number syntactic functions
Clean data and number semantic functions
Calculate combinations and build references
List tested combinations
List combinations found with frequencies

Figure 6.15. The interface of the experiment that analyses the mapping of semantic functions on syntactic functions.

When the researcher presses the "Clean data and number syntactic functions" button, the code, shown in Figure 6.16, is run. It validates and replaces the names of syntactic functions with logical numbers in order to fit the output on a small screen and, more importantly, to facilitate the procedure that will test whether a theoretically possible combination occurs within the data set or not. This procedure makes use of repetition structures and compares the loops' indexes to the logical numbers of the semantic and syntactic functions.

```
Option Explicit

Dim count11, count12, count13, count14, count15, count16 As Integer
Dim arrSyn1(1 To 200, 1 To 5, 1 To 6)
Dim combSemSyn(1 To 37, 1 To 14) As Integer
Dim refSemSyn(1 To 37, 1 To 14, 1 To 2) As String 'Store types and
   references
Dim m, n, o, p, totCombs, totUniqueCombs, tucFlag As Integer
Dim toets11 As String

'Clean data - syntactic functions
Private Sub cmdCleanData_Click()

Dim toets11 As String
toets11 = ""
```

```
For count11 = 1 To arrayMax 'Copy array
 For count12 = 1 To 5
  For count13 = 1 To 6
   arrSyn1(count11, count12, count13) = Clause(count11, count12, count13)
    Next
 Next
Next

For count11 = 1 To arrayMax 'Check and number syntactic functions
 For count12 = 1 To 5
  For count13 = 5 To 5 'Check only syntactic function dimension
   'Here limited to those functions occurring in Gen 1:1-2:3
   If arrSyn1(count11, count12, 5) = "Main verb" Then
      arrSyn1(count11, count12, 5) = "01"
   ElseIf arrSyn1(count11, count12, 5) = "Copulative verb" Then
      arrSyn1(count11, count12, 5) = "02"
   ElseIf arrSyn1(count11, count12, 5) = "Subject" Then
      arrSyn1(count11, count12, 5) = "03"
   ElseIf arrSyn1(count11, count12, 5) = "Object" Then
      arrSyn1(count11, count12, 5) = "04"
   ElseIf arrSyn1(count11, count12, 5) = "Object clause" Then
      arrSyn1(count11, count12, 5) = "05"
   ElseIf arrSyn1(count11, count12, 5) = "Object cluster" Then
      arrSyn1(count11, count12, 5) = "06"
   ElseIf arrSyn1(count11, count12, 5) = "IndObj" Then
      arrSyn1(count11, count12, 5) = "07"
   ElseIf arrSyn1(count11, count12, 5) = "Complement" Then
      arrSyn1(count11, count12, 5) = "08"
   ElseIf arrSyn1(count11, count12, 5) = "Copula-predicate" Then
      arrSyn1(count11, count12, 5) = "09"
   ElseIf arrSyn1(count11, count12, 5) = "Adjunct" Then
      arrSyn1(count11, count12, 5) = "10"
   ElseIf arrSyn1(count11, count12, 5) = "Disjunct" Then
      arrSyn1(count11, count12, 5) = "11"
   ElseIf arrSyn1(count11, count12, 5) = "Attribute" Then
      arrSyn1(count11, count12, 5) = "12"
   ElseIf arrSyn1(count11, count12, 5) = "Conj" Then
      arrSyn1(count11, count12, 5) = "13"
   ElseIf arrSyn1(count11, count12, 5) = "-" Then
      arrSyn1(count11, count12, 5) = "14" 'To count null synf values
   'Clean data if following message is shown:
   Else: MsgBox ("Synf " & arrSyn1(count11, count12, 5) & " in vs " &
      arrSyn1(count11, 1, 1) & " could not be numbered")
   End If
  Next
 Next
```

```
Next

For count11 = 1 To arrayMax 'Show syntactic frames as combinations of
   numbers
 For count12 = 1 To 5
  For count13 = 5 To 5
   toets11 = toets11 & " " & arrSyn1(count11, count12, 5)
  Next
 Next
 toets11 = toets11 & " (" & arrSyn1(count11, 1, 1) & ", Clause " & count11
    & ")" & vbCrLf
Next
Text1.Text = toets11

End Sub
```

Figure 6.16. VB6 code used to validate and number the syntactic function slice.

The output of this first step is shown in Figure 6.17.

```
10 01 03 04 14 (Gen01v01a, Clause 1)
03 02 09 14 14 (Gen01v02a, Clause 2)
03 09 14 14 14 (Gen01v02b, Clause 3)
03 09 08 14 14 (Gen01v02c, Clause 4)
01 03 05 14 14 (Gen01v03a, Clause 5)
02 03 14 14 14 (Gen01v03b, Clause 6)
02 03 14 14 14 (Gen01v03c, Clause 7)
01 03 05 14 14 (Gen01v04a, Clause 8)
03 13 09 14 14 (Gen01v04b, Clause 9)
01 03 08 08 14 (Gen01v04c, Clause 10)
01 03 07 08 14 (Gen01v05a, Clause 11)
07 01 08 14 14 (Gen01v05b, Clause 12)
02 03 14 14 14 (Gen01v05c, Clause 13)
02 03 12 14 14 (Gen01v05d, Clause 14)
01 03 06 14 14 (Gen01v06a, Clause 15)
02 03 09 14 14 (Gen01v06b, Clause 16)
02 08 08 14 14 (Gen01v06c, Clause 17)
01 03 04 14 14 (Gen01v07a, Clause 18)
01 04 08 14 14 (Gen01v07b, Clause 19)
03 09 14 14 14 (Gen01v07c, Clause 20)
03 09 14 14 14 (Gen01v07d, Clause 21)
02 08 14 14 14 (Gen01v07e, Clause 22)
01 03 07 04 14 (Gen01v08a, Clause 23)
```

```
02 03 14 14 14 (Gen01v08b, Clause 24)
02 03 12 14 14 (Gen01v08c, Clause 25)
01 03 06 14 14 (Gen01v09a, Clause 26)
01 03 08 14 14 (Gen01v09b, Clause 27)
01 03 14 14 14 (Gen01v09c, Clause 28)
02 08 14 14 14 (Gen01v09d, Clause 29)
01 03 07 04 14 (Gen01v10a, Clause 30)
07 01 04 14 14 (Gen01v10b, Clause 31)
01 03 05 14 14 (Gen01v10c, Clause 32)
13 09 14 14 14 (Gen01v10d, Clause 33)
01 03 05 14 14 (Gen01v11a, Clause 34)
01 03 08 14 14 (Gen01v11b, Clause 35)
13 03 09 14 14 (Gen01v11c, Clause 36)
02 08 14 14 14 (Gen01v11d, Clause 37)
01 03 04 10 14 (Gen01v12a, Clause 38)
13 03 09 14 14 (Gen01v12b, Clause 39)
01 03 05 14 14 (Gen01v12c, Clause 40)
13 09 14 14 14 (Gen01v12d, Clause 41)
02 03 14 14 14 (Gen01v13a, Clause 42)
02 03 12 14 14 (Gen01v13b, Clause 43)
01 03 06 14 14 (Gen01v14a, Clause 44)
02 03 09 10 14 (Gen01v14b, Clause 45)
01 09 14 14 14 (Gen01v14c, Clause 46)
01 09 10 10 14 (Gen01v15a, Clause 47)
02 08 14 14 14 (Gen01v15b, Clause 48)
01 03 04 10 14 (Gen01v16a, Clause 49)
01 04 03 08 10 (Gen01v17a, Clause 50)
01 03 05 14 14 (Gen01v18b, Clause 51)
13 09 14 14 14 (Gen01v18d, Clause 52)
02 03 14 14 14 (Gen01v19a, Clause 53)
02 03 12 14 14 (Gen01v19b, Clause 54)
01 03 06 14 14 (Gen01v20a, Clause 55)
01 03 08 14 14 (Gen01v20b, Clause 56)
03 01 08 10 14 (Gen01v20c, Clause 57)
01 03 04 10 14 (Gen01v21a, Clause 58)
03 01 10 14 14 (Gen01v21b, Clause 59)
01 03 05 14 14 (Gen01v21c, Clause 60)
13 09 14 14 14 (Gen01v21d, Clause 61)
01 04 03 10 14 (Gen01v22a, Clause 62)
01 14 14 14 14 (Gen01v22b, Clause 63)
01 14 14 14 14 (Gen01v22c, Clause 64)
01 04 14 14 14 (Gen01v22d, Clause 65)
03 01 10 14 14 (Gen01v22e, Clause 66)
02 03 14 14 14 (Gen01v23a, Clause 67)
02 03 12 14 14 (Gen01v23b, Clause 68)
01 03 05 14 14 (Gen01v24a, Clause 69)
```

```
01 03 04 10 14 (Gen01v24b, Clause 70)
02 08 14 14 14 (Gen01v24c, Clause 71)
01 03 04 10 14 (Gen01v25a, Clause 72)
01 03 05 14 14 (Gen01v25b, Clause 73)
13 09 14 14 14 (Gen01v25c, Clause 74)
01 03 06 14 14 (Gen01v26a, Clause 75)
01 04 10 14 14 (Gen01v26b, Clause 76)
01 08 14 14 14 (Gen01v26c, Clause 77)
01 03 04 10 14 (Gen01v27a, Clause 78)
10 01 04 14 14 (Gen01v27b, Clause 79)
12 01 04 14 14 (Gen01v27c, Clause 80)
01 04 03 14 14 (Gen01v28a, Clause 81)
01 03 06 14 14 (Gen01v28b, Clause 82)
01 14 14 14 14 (Gen01v28c, Clause 83)
01 14 14 14 14 (Gen01v28d, Clause 84)
01 04 14 14 14 (Gen01v28e, Clause 85)
01 14 14 14 14 (Gen01v28f, Clause 86)
01 08 14 14 14 (Gen01v28g, Clause 87)
01 03 06 14 14 (Gen01v29a, Clause 88)
11 01 07 04 14 (Gen01v29b, Clause 89)
03 09 14 14 14 (Gen01v29c, Clause 90)
13 08 03 14 14 (Gen01v29d, Clause 91)
07 01 09 14 14 (Gen01v29e, Clause 92)
13 09 03 14 14 (Gen01v30b, Clause 93)
02 08 14 14 14 (Gen01v30c, Clause 94)
01 03 04 14 14 (Gen01v31a, Clause 95)
04 01 14 14 14 (Gen01v31b, Clause 96)
11 09 14 14 14 (Gen01v31c, Clause 97)
02 03 14 14 14 (Gen01v31d, Clause 98)
02 03 12 14 14 (Gen01v31e, Clause 99)
01 03 14 14 14 (Gen02v01a, Clause 100)
01 03 10 04 14 (Gen02v02a, Clause 101)
04 01 14 14 14 (Gen02v02b, Clause 102)
01 10 08 14 14 (Gen02v02c, Clause 103)
04 01 14 14 14 (Gen02v02d, Clause 104)
01 03 04 14 14 (Gen02v03a, Clause 105)
01 04 10 14 14 (Gen02v03b, Clause 106)
13 10 01 08 14 (Gen02v03c, Clause 107)
04 01 03 10 14 (Gen02v03d, Clause 108)
```

Figure 6.17. The syntactic function slice represented by a logical numbering system.

The second step is to clean and number the semantic-function data slice. The code is shown in Figure 6.18. The names of the semantic functions are also replaced by logical numbers.

```
'Clean data - semantic functions
Private Sub cmdNoSemf_Click()

For count11 = 1 To arrayMax 'Check and number semantic functions
 For count12 = 1 To 5
  For count13 = 5 To 5 'Check only semantic function dimension
   If arrSyn1(count11, count12, 6) = "Action" Then
    arrSyn1(count11, count12, 6) = "01"
   ElseIf arrSyn1(count11, count12, 6) = "Position" Then
    arrSyn1(count11, count12, 6) = "02"
   ElseIf arrSyn1(count11, count12, 6) = "Process" Then
       arrSyn1(count11, count12, 6) = "03"
   ElseIf arrSyn1(count11, count12, 6) = "State" Then
       arrSyn1(count11, count12, 6) = "04"
   ElseIf arrSyn1(count11, count12, 6) = "Agent" Then
       arrSyn1(count11, count12, 6) = "05"
   ElseIf arrSyn1(count11, count12, 6) = "Positioner" Then
      arrSyn1(count11, count12, 6) = "06"
   ElseIf arrSyn1(count11, count12, 6) = "Force" Then
       arrSyn1(count11, count12, 6) = "07"
   ElseIf arrSyn1(count11, count12, 6) = "Processed" Then
       arrSyn1(count11, count12, 6) = "08"
   ElseIf arrSyn1(count11, count12, 6) = "Zero" Then
       arrSyn1(count11, count12, 6) = "09"
   ElseIf arrSyn1(count11, count12, 6) = "Patient" Then
       arrSyn1(count11, count12, 6) = "10"
   ElseIf arrSyn1(count11, count12, 6) = "Product" Then
       arrSyn1(count11, count12, 6) = "11"
   ElseIf arrSyn1(count11, count12, 6) = "Receiver" Then
       arrSyn1(count11, count12, 6) = "12"
   ElseIf arrSyn1(count11, count12, 6) = "Location" Then
       arrSyn1(count11, count12, 6) = "13"
   ElseIf arrSyn1(count11, count12, 6) = "Direction" Then
       arrSyn1(count11, count12, 6) = "14"
   ElseIf arrSyn1(count11, count12, 6) = "Source" Then
       arrSyn1(count11, count12, 6) = "15"
   ElseIf arrSyn1(count11, count12, 6) = "Reference" Then
       arrSyn1(count11, count12, 6) = "16"
   ElseIf arrSyn1(count11, count12, 6) = "Beneficiary" Then
       arrSyn1(count11, count12, 6) = "17"
   ElseIf arrSyn1(count11, count12, 6) = "Company" Then
       arrSyn1(count11, count12, 6) = "18"
   ElseIf arrSyn1(count11, count12, 6) = "Instrument" Then
       arrSyn1(count11, count12, 6) = "19"
   ElseIf arrSyn1(count11, count12, 6) = "Manner" Then
       arrSyn1(count11, count12, 6) = "20"
   ElseIf arrSyn1(count11, count12, 6) = "Speed" Then
       arrSyn1(count11, count12, 6) = "21"
   ElseIf arrSyn1(count11, count12, 6) = "Role" Then
       arrSyn1(count11, count12, 6) = "22"
   ElseIf arrSyn1(count11, count12, 6) = "Path" Then
       arrSyn1(count11, count12, 6) = "23"
   ElseIf arrSyn1(count11, count12, 6) = "Time" Then
       arrSyn1(count11, count12, 6) = "24"
   ElseIf arrSyn1(count11, count12, 6) = "Circumstance" Then
       arrSyn1(count11, count12, 6) = "27"
   ElseIf arrSyn1(count11, count12, 6) = "Result" Then
       arrSyn1(count11, count12, 6) = "28"
   ElseIf arrSyn1(count11, count12, 6) = "Purpose" Then
       arrSyn1(count11, count12, 6) = "29"
   ElseIf arrSyn1(count11, count12, 6) = "Reason" Then
       arrSyn1(count11, count12, 6) = "30"
   ElseIf arrSyn1(count11, count12, 6) = "Cause" Then
       arrSyn1(count11, count12, 6) = "31"
   ElseIf arrSyn1(count11, count12, 6) = "Possessor" Then
       arrSyn1(count11, count12, 6) = "32"
   ElseIf arrSyn1(count11, count12, 6) = "Identification" Then
       arrSyn1(count11, count12, 6) = "33"
   ElseIf arrSyn1(count11, count12, 6) = "Classification" Then
       arrSyn1(count11, count12, 6) = "34"
   ElseIf arrSyn1(count11, count12, 6) = "Quality" Then
       arrSyn1(count11, count12, 6) = "35"
   ElseIf arrSyn1(count11, count12, 6) = "Existence" Then
       arrSyn1(count11, count12, 6) = "36"
   ElseIf arrSyn1(count11, count12, 6) = "-" Then
       arrSyn1(count11, count12, 6) = "37" 'To count null semantic function
          values
   'Clean data if following message is shown:
   Else: MsgBox ("Semf " & arrSyn1(count11, count12, 6) & " in vs " &
      arrSyn1(count11, 1, 1) & " could not be numbered")
   End If
  Next
 Next
Next

toets11 = ""
```

```
For count14 = 1 To arrayMax 'Show semantic frames
 For count15 = 1 To 5
  For count16 = 5 To 5
   toets11 = toets11 & " " & arrSyn1(count14, count15, 6)
  Next
 Next
  toets11 = toets11 & " (" & arrSyn1(count14, 1, 1) & ", Clause " & count14
     & ")" & vbCrLf
Next
Text1.Text = toets11

End Sub
```

Figure 6.18. VB6 code used to validate and number the semantic function slice.

When the user presses the "Clean data and number semantic functions" button, the code runs and produces the output shown in Figure 6.19.

```
24 01 05 11 37 (Gen01v01a, Clause 1)
09 04 34 37 37 (Gen01v02a, Clause 2)
09 13 37 37 37 (Gen01v02b, Clause 3)
06 02 13 37 37 (Gen01v02c, Clause 4)
01 05 10 37 37 (Gen01v03a, Clause 5)
04 09 37 37 37 (Gen01v03b, Clause 6)
04 09 37 37 37 (Gen01v03c, Clause 7)
03 08 10 37 37 (Gen01v04a, Clause 8)
09 37 35 37 37 (Gen01v04b, Clause 9)
01 05 10 15 37 (Gen01v04c, Clause 10)
01 05 10 11 37 (Gen01v05a, Clause 11)
10 01 11 37 37 (Gen01v05b, Clause 12)
04 09 37 37 37 (Gen01v05c, Clause 13)
04 09 35 37 37 (Gen01v05d, Clause 14)
01 05 10 37 37 (Gen01v06a, Clause 15)
04 09 13 37 37 (Gen01v06b, Clause 16)
01 10 15 37 37 (Gen01v06c, Clause 17)
01 05 10 37 37 (Gen01v07a, Clause 18)
01 10 15 37 37 (Gen01v07b, Clause 19)
09 13 37 37 37 (Gen01v07c, Clause 20)
09 13 37 37 37 (Gen01v07d, Clause 21)
04 20 37 37 37 (Gen01v07e, Clause 22)
01 05 10 11 37 (Gen01v08a, Clause 23)
04 09 37 37 37 (Gen01v08b, Clause 24)
```

```
04 09 35 37 37 (Gen01v08c, Clause 25)
01 05 10 37 37 (Gen01v09a, Clause 26)
03 08 13 37 37 (Gen01v09b, Clause 27)
03 08 37 37 37 (Gen01v09c, Clause 28)
04 20 37 37 37 (Gen01v09d, Clause 29)
01 05 10 11 37 (Gen01v10a, Clause 30)
10 01 11 37 37 (Gen01v10b, Clause 31)
03 08 10 37 37 (Gen01v10c, Clause 32)
37 35 37 37 37 (Gen01v10d, Clause 33)
01 05 10 37 37 (Gen01v11a, Clause 34)
03 08 16 37 37 (Gen01v11b, Clause 35)
37 09 13 37 37 (Gen01v11c, Clause 36)
04 20 37 37 37 (Gen01v11d, Clause 37)
03 08 11 20 37 (Gen01v12a, Clause 38)
37 09 13 37 37 (Gen01v12b, Clause 39)
03 08 10 37 37 (Gen01v12c, Clause 40)
37 35 37 37 37 (Gen01v12d, Clause 41)
04 09 37 37 37 (Gen01v13a, Clause 42)
04 09 35 37 37 (Gen01v13b, Clause 43)
01 05 10 37 37 (Gen01v14a, Clause 44)
04 09 13 29 37 (Gen01v14b, Clause 45)
04 34 37 37 37 (Gen01v14c, Clause 46)
04 34 13 29 37 (Gen01v15a, Clause 47)
04 20 37 37 37 (Gen01v15b, Clause 48)
01 05 11 29 37 (Gen01v16a, Clause 49)
01 10 05 13 29 (Gen01v17a, Clause 50)
03 08 10 37 37 (Gen01v18b, Clause 51)
37 35 37 37 37 (Gen01v18d, Clause 52)
04 09 37 37 37 (Gen01v19a, Clause 53)
04 09 35 37 37 (Gen01v19b, Clause 54)
01 05 10 37 37 (Gen01v20a, Clause 55)
03 08 16 37 37 (Gen01v20b, Clause 56)
05 01 13 13 37 (Gen01v20c, Clause 57)
01 05 11 20 37 (Gen01v21a, Clause 58)
05 01 13 37 37 (Gen01v21b, Clause 59)
03 08 10 37 37 (Gen01v21c, Clause 60)
37 35 37 37 37 (Gen01v21d, Clause 61)
01 10 05 20 37 (Gen01v22a, Clause 62)
04 37 37 37 37 (Gen01v22b, Clause 63)
03 37 37 37 37 (Gen01v22c, Clause 64)
01 10 37 37 37 (Gen01v22d, Clause 65)
08 03 13 37 37 (Gen01v22e, Clause 66)
04 09 37 37 37 (Gen01v23a, Clause 67)
04 09 35 37 37 (Gen01v23b, Clause 68)
01 05 10 37 37 (Gen01v24a, Clause 69)
03 08 11 20 37 (Gen01v24b, Clause 70)
```

```
04 20 37 37 37 (Gen01v24c, Clause 71)
01 05 11 20 37 (Gen01v25a, Clause 72)
03 08 10 37 37 (Gen01v25b, Clause 73)
37 35 37 37 37 (Gen01v25c, Clause 74)
01 05 10 37 37 (Gen01v26a, Clause 75)
01 11 20 37 37 (Gen01v26b, Clause 76)
05 10 37 37 37 (Gen01v26c, Clause 77)
01 05 11 20 37 (Gen01v27a, Clause 78)
20 01 11 37 37 (Gen01v27b, Clause 79)
35 01 11 37 37 (Gen01v27c, Clause 80)
01 10 05 37 37 (Gen01v28a, Clause 81)
01 05 10 37 37 (Gen01v28b, Clause 82)
04 37 37 37 37 (Gen01v28c, Clause 83)
03 37 37 37 37 (Gen01v28d, Clause 84)
01 10 37 37 37 (Gen01v28e, Clause 85)
01 37 37 37 37 (Gen01v28f, Clause 86)
01 10 37 37 37 (Gen01v28g, Clause 87)
01 05 10 37 37 (Gen01v29a, Clause 88)
37 01 12 10 37 (Gen01v29b, Clause 89)
09 13 37 37 37 (Gen01v29c, Clause 90)
09 13 09 37 37 (Gen01v29d, Clause 91)
17 04 29 37 37 (Gen01v29e, Clause 92)
37 13 09 37 37 (Gen01v30b, Clause 93)
04 20 37 37 37 (Gen01v30c, Clause 94)
03 08 10 37 37 (Gen01v31a, Clause 95)
11 01 37 37 37 (Gen01v31b, Clause 96)
37 35 37 37 37 (Gen01v31c, Clause 97)
04 09 37 37 37 (Gen01v31d, Clause 98)
04 09 35 37 37 (Gen01v31e, Clause 99)
01 10 37 37 37 (Gen02v01a, Clause 100)
01 05 24 10 37 (Gen02v02a, Clause 101)
11 01 37 37 37 (Gen02v02b, Clause 102)
01 24 10 37 37 (Gen02v02c, Clause 103)
11 01 37 37 37 (Gen02v02d, Clause 104)
01 05 10 37 37 (Gen02v03a, Clause 105)
01 10 30 37 37 (Gen02v03b, Clause 106)
37 24 01 10 37 (Gen02v03c, Clause 107)
11 01 05 20 37 (Gen02v03d, Clause 108)
```

Figure 6.19. The semantic function slice represented by a logical numbering system.

Next, all possible combinations of syntactic and semantic functions are tested against the clause cube. The code, shown in Figure 6.20, calculates the total number of all theoretically possible combinations, but only of those syntactic and semantic functions that do occur in the clause cube. In a larger data set there would be many more possibilities. All these combinations are tested against the data in the cube and the frequencies of those combinations that do occur, are calculated. The number of occurrences of each combination is then shown in a matrix. This procedure is activated by pressing the button, "Calculate combinations and build references".

```
'Calculate combinations and build references
Private Sub cmdCalcComb_Click()

For m = 1 To 37 'Start with each semantic function
 For n = 1 To 14 'Combine with each syntactic function
  combSemSyn(m, n) = 0 'Reset all counts to zero
 Next
Next

toets11 = ""
totCombs = 0
totUniqueCombs = 0
tucFlag = 0 'Boolean to signal if new unique combination is found

For m = 1 To 37 ' Start with each semantic function – outer loop
 For n = 1 To 14 ' Combine with each syntactic function – inner loop
  tucFlag = 0
  For o = 1 To arrayMax 'Work through all rows/clauses
   For p = 1 To 5 'Work through all columns/constituents
    If Val(arrSyn1(o, p, 6)) = m And Val(arrSyn1(o, p, 5)) = n Then
     combSemSyn(m, n) = combSemSyn(m, n) + 1 'Add 1 to count if combination
        is found
     totCombs = totCombs + 1
     tucFlag = 1
     refSemSyn(m, n, 2) = refSemSyn(m, n, 2) & "    " & arrSyn1(o, 1, 1) & "
        - " & "Clause " & o & vbCrLf 'Add verse no to reference string
    End If
   Next
  Next
   If tucFlag = 1 Then
    totUniqueCombs = totUniqueCombs + 1
   End If
  Next
 Next
```

```
toets11 = "Total combinations found: " & totCombs
toets11 = toets11 & vbCrLf & "Total unique combinations found: " &
totUniqueCombs & vbCrLf & vbCrLf

For m = 1 To 37 'Show table
 For n = 1 To 14
  toets11 = toets11 & "  " & combSemSyn(m, n) 'Create output table
 Next
  toets11 = toets11 & vbCrLf
Next
Text1.Text = toets11

End Sub
```

Figure 6.20. VB6 code used to test all possible combinations of semantic and syntactic functions.

The output, shown in Figure 6.21, is not human-friendly, and should still be "translated" into language that is accessible to humans (see next step below). Each row represents a semantic function and each column represents the list of syntactic functions; each cell represents a possible combination and the number of hits in the data set. For example, the first cell means that there are 46 combinations of the mapping of action predicates on main verbs in Genesis 1:1-2:3.

```

Total combinations found: 540
Total unique combinations found: 39

 46  1  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  1  0  0  0  0  0
 16  0  0  0  0  0  0  0  0  0  0  0  0  0
 5  23  0  0  0  0  0  0  0  0  0  0  0  0
 1  0  28  0  0  0  0  0  0  0  0  0  0  0
 0  0  1  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  14  0  0  0  0  0  0  0  0  0  0  0
 0  0  26  0  0  0  0  0  0  0  0  0  1  0
 0  0  1  12  9  7  5  6  0  0  0  0  0  0
 0  0  0  17  0  0  0  2  0  0  0  0  0  0
 0  0  0  0  0  0  1  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  5  9  4  0  0  0  0
```

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 3 0 0 0 0 0 0
0 0 0 0 0 0 0 2 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 6 0 9 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 4 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 4 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 3 0 0 0 0 0
0 0 0 0 0 0 0 0 7 0 0 7 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 2 0 10 239
```

Figure 6.21. Initial, human-unfriendly, results of the algorithm that tests for all possible combinations of semantic and syntactic functions in the data set.

The second-last step is an intermediate step that builds a human-readable list of all the theoretical combinations that have been tested and displays these to the researcher. The code is shown in Figure 6.22.

```
'Build slice in refSemSyn to capture combination types

Private Sub cmdOutput_Click()

Dim combTypeA, combTypeB, combType As String

For m = 1 To 37 'Start with each semantic function
 combTypeA = "semf"
  Select Case m
```

```
Case 1
 combTypeA = "Action"
Case 2
 combTypeA = "Position"
Case 3
 combTypeA = "Process"
Case 4
 combTypeA = "State"
Case 5
 combTypeA = "Agent"
Case 6
 combTypeA = "Positioner"
Case 7
 combTypeA = "Force"
Case 8
 combTypeA = "Processed"
Case 9
 combTypeA = "Zero"
Case 10
 combTypeA = "Patient"
Case 11
 combTypeA = "Product"
Case 12
 combTypeA = "Receiver"
Case 13
 combTypeA = "Location"
Case 14
 combTypeA = "Direction"
Case 15
 combTypeA = "Source"
Case 16
 combTypeA = "Reference"
Case 17
 combTypeA = "Beneficiary"
Case 18
 combTypeA = "Company"
Case 19
 combTypeA = "Instrument"
Case 20
 combTypeA = "Manner"
Case 21
 combTypeA = "Speed"
Case 22
 combTypeA = "Role"
Case 23
 combTypeA = "Path"
```

```
  Case 24
   combTypeA = "Time"
  Case 25
   combTypeA = "Duration"
  Case 26
   combTypeA = "Frequency"
  Case 27
   combTypeA = "Circumstance"
  Case 28
   combTypeA = "Result"
  Case 29
   combTypeA = "Purpose"
  Case 30
   combTypeA = "Reason"
  Case 31
   combTypeA = "Cause"
  Case 32
   combTypeA = "Possessor"
  Case 33
   combTypeA = "Identification"
  Case 34
   combTypeA = "Classification"
  Case 35
   combTypeA = "Quality"
  Case 36
   combTypeA = "Existence"
  Case 37
   combTypeA = "-"
  Case Else
  MsgBox ("Semantic function no " & m & " not listed.")
 End Select

 For n = 1 To 14 'Combine with each syntactic function
  combTypeB = "synf"
  Select Case n
  Case 1
   combTypeB = "Main verb"
  Case 2
   combTypeB = "Copulative verb"
  Case 3
   combTypeB = "Subject"
  Case 4
   combTypeB = "Object"
  Case 5
   combTypeB = "Object clause"
  Case 6
```

```
    combTypeB = "Object cluster"
   Case 7
    combTypeB = "IndObj"
   Case 8
    combTypeB = "Complement"
   Case 9
    combTypeB = "Copula-predicate"
   Case 10
    combTypeB = "Adjunct"
   Case 11
    combTypeB = "Disjunct"
   Case 12
    combTypeB = "Attribute"
   Case 13
    combTypeB = "Conj"
   Case 14
    combTypeB = "-"
   Case Else
   MsgBox ("Syntactic function no " & n & " not listed.")
  End Select

  combType = combTypeA & (" = ") & combTypeB 'Combine semantic and syntactic
     function
  refSemSyn(m, n, 1) = combType
  Next
 Next

'Show table - list of all possible combinations
toets11 = "Possible combinations of semantic functions and syntactic
    functions tested:" & vbCrLf & vbCrLf

For m = 1 To 37
 For n = 1 To 14
  toets11 = toets11 & "  " & refSemSyn(m, n, 1) & vbCrLf ' Create output -
     combination type
 Next
 toets11 = toets11 & vbCrLf
Next
Text1.Text = toets11

End Sub
```

Figure 6.22. VB6 code used to build and display a human-readable list of combinations of semantic and syntactic functions.

The output of this intermediate step, activated by pressing the "List tested combinations" button, is shown in Figure 6.23.

```
Possible combinations of semantic functions and syntactic functions tested:

 Action = Main verb
 Action = Copulative verb
 Action = Subject
 Action = Object
 Action = Object clause
 Action = Object cluster
 Action = IndObj
 Action = Complement
 Action = Copula-predicate
 Action = Adjunct
 Action = Disjunct
 Action = Attribute
 Action = Conj
 Action = -

 Position = Main verb
 Position = Copulative verb
 Position = Subject
 Position = Object
 Position = Object clause
 Position = Object cluster
 Position = IndObj
 Position = Complement
 Position = Copula-predicate
 Position = Adjunct
 Position = Disjunct
 Position = Attribute
 Position = Conj
 Position = -

 Process = Main verb
 Process = Copulative verb
 Process = Subject
 Process = Object
 Process = Object clause
 Process = Object cluster
 Process = IndObj
 Process = Complement
```

Process = Copula-predicate
Process = Adjunct
Process = Disjunct
Process = Attribute
Process = Conj
Process = -

State = Main verb
State = Copulative verb
State = Subject
State = Object
State = Object clause
State = Object cluster
State = IndObj
State = Complement
State = Copula-predicate
State = Adjunct
State = Disjunct
State = Attribute
State = Conj
State = -

Agent = Main verb
Agent = Copulative verb
Agent = Subject
Agent = Object
Agent = Object clause
Agent = Object cluster
Agent = IndObj
Agent = Complement
Agent = Copula-predicate
Agent = Adjunct
Agent = Disjunct
Agent = Attribute
Agent = Conj
Agent = -

Positioner = Main verb
Positioner = Copulative verb
Positioner = Subject
Positioner = Object
Positioner = Object clause
Positioner = Object cluster
Positioner = IndObj
Positioner = Complement
Positioner = Copula-predicate

Positioner = Adjunct
Positioner = Disjunct
Positioner = Attribute
Positioner = Conj
Positioner = -

Force = Main verb
Force = Copulative verb
Force = Subject
Force = Object
Force = Object clause
Force = Object cluster
Force = IndObj
Force = Complement
Force = Copula-predicate
Force = Adjunct
Force = Disjunct
Force = Attribute
Force = Conj
Force = -

Processed = Main verb
Processed = Copulative verb
Processed = Subject
Processed = Object
Processed = Object clause
Processed = Object cluster
Processed = IndObj
Processed = Complement
Processed = Copula-predicate
Processed = Adjunct
Processed = Disjunct
Processed = Attribute
Processed = Conj
Processed = -

Zero = Main verb
Zero = Copulative verb
Zero = Subject
Zero = Object
Zero = Object clause
Zero = Object cluster
Zero = IndObj
Zero = Complement
Zero = Copula-predicate
Zero = Adjunct

Zero = Disjunct
Zero = Attribute
Zero = Conj
Zero = -

Patient = Main verb
Patient = Copulative verb
Patient = Subject
Patient = Object
Patient = Object clause
Patient = Object cluster
Patient = IndObj
Patient = Complement
Patient = Copula-predicate
Patient = Adjunct
Patient = Disjunct
Patient = Attribute
Patient = Conj
Patient = -

Product = Main verb
Product = Copulative verb
Product = Subject
Product = Object
Product = Object clause
Product = Object cluster
Product = IndObj
Product = Complement
Product = Copula-predicate
Product = Adjunct
Product = Disjunct
Product = Attribute
Product = Conj
Product = -

Receiver = Main verb
Receiver = Copulative verb
Receiver = Subject
Receiver = Object
Receiver = Object clause
Receiver = Object cluster
Receiver = IndObj
Receiver = Complement
Receiver = Copula-predicate
Receiver = Adjunct
Receiver = Disjunct

Receiver = Attribute
Receiver = Conj
Receiver = -

Location = Main verb
Location = Copulative verb
Location = Subject
Location = Object
Location = Object clause
Location = Object cluster
Location = IndObj
Location = Complement
Location = Copula-predicate
Location = Adjunct
Location = Disjunct
Location = Attribute
Location = Conj
Location = -

Direction = Main verb
Direction = Copulative verb
Direction = Subject
Direction = Object
Direction = Object clause
Direction = Object cluster
Direction = IndObj
Direction = Complement
Direction = Copula-predicate
Direction = Adjunct
Direction = Disjunct
Direction = Attribute
Direction = Conj
Direction = -

Source = Main verb
Source = Copulative verb
Source = Subject
Source = Object
Source = Object clause
Source = Object cluster
Source = IndObj
Source = Complement
Source = Copula-predicate
Source = Adjunct
Source = Disjunct
Source = Attribute

Source = Conj
Source = -

Reference = Main verb
Reference = Copulative verb
Reference = Subject
Reference = Object
Reference = Object clause
Reference = Object cluster
Reference = IndObj
Reference = Complement
Reference = Copula-predicate
Reference = Adjunct
Reference = Disjunct
Reference = Attribute
Reference = Conj
Reference = -

Beneficiary = Main verb
Beneficiary = Copulative verb
Beneficiary = Subject
Beneficiary = Object
Beneficiary = Object clause
Beneficiary = Object cluster
Beneficiary = IndObj
Beneficiary = Complement
Beneficiary = Copula-predicate
Beneficiary = Adjunct
Beneficiary = Disjunct
Beneficiary = Attribute
Beneficiary = Conj
Beneficiary = -

Company = Main verb
Company = Copulative verb
Company = Subject
Company = Object
Company = Object clause
Company = Object cluster
Company = IndObj
Company = Complement
Company = Copula-predicate
Company = Adjunct
Company = Disjunct
Company = Attribute
Company = Conj

Company = -

Instrument = Main verb
Instrument = Copulative verb
Instrument = Subject
Instrument = Object
Instrument = Object clause
Instrument = Object cluster
Instrument = IndObj
Instrument = Complement
Instrument = Copula-predicate
Instrument = Adjunct
Instrument = Disjunct
Instrument = Attribute
Instrument = Conj
Instrument = -

Manner = Main verb
Manner = Copulative verb
Manner = Subject
Manner = Object
Manner = Object clause
Manner = Object cluster
Manner = IndObj
Manner = Complement
Manner = Copula-predicate
Manner = Adjunct
Manner = Disjunct
Manner = Attribute
Manner = Conj
Manner = -

Speed = Main verb
Speed = Copulative verb
Speed = Subject
Speed = Object
Speed = Object clause
Speed = Object cluster
Speed = IndObj
Speed = Complement
Speed = Copula-predicate
Speed = Adjunct
Speed = Disjunct
Speed = Attribute
Speed = Conj
Speed = -

```
Role = Main verb
Role = Copulative verb
Role = Subject
Role = Object
Role = Object clause
Role = Object cluster
Role = IndObj
Role = Complement
Role = Copula-predicate
Role = Adjunct
Role = Disjunct
Role = Attribute
Role = Conj
Role = -

Path = Main verb
Path = Copulative verb
Path = Subject
Path = Object
Path = Object clause
Path = Object cluster
Path = IndObj
Path = Complement
Path = Copula-predicate
Path = Adjunct
Path = Disjunct
Path = Attribute
Path = Conj
Path = -

Time = Main verb
Time = Copulative verb
Time = Subject
Time = Object
Time = Object clause
Time = Object cluster
Time = IndObj
Time = Complement
Time = Copula-predicate
Time = Adjunct
Time = Disjunct
Time = Attribute
Time = Conj
Time = -
```

Duration = Main verb
Duration = Copulative verb
Duration = Subject
Duration = Object
Duration = Object clause
Duration = Object cluster
Duration = IndObj
Duration = Complement
Duration = Copula-predicate
Duration = Adjunct
Duration = Disjunct
Duration = Attribute
Duration = Conj
Duration = -

Frequency = Main verb
Frequency = Copulative verb
Frequency = Subject
Frequency = Object
Frequency = Object clause
Frequency = Object cluster
Frequency = IndObj
Frequency = Complement
Frequency = Copula-predicate
Frequency = Adjunct
Frequency = Disjunct
Frequency = Attribute
Frequency = Conj
Frequency = -

Circumstance = Main verb
Circumstance = Copulative verb
Circumstance = Subject
Circumstance = Object
Circumstance = Object clause
Circumstance = Object cluster
Circumstance = IndObj
Circumstance = Complement
Circumstance = Copula-predicate
Circumstance = Adjunct
Circumstance = Disjunct
Circumstance = Attribute
Circumstance = Conj
Circumstance = -

Result = Main verb

Result = Copulative verb
Result = Subject
Result = Object
Result = Object clause
Result = Object cluster
Result = IndObj
Result = Complement
Result = Copula-predicate
Result = Adjunct
Result = Disjunct
Result = Attribute
Result = Conj
Result = -

Purpose = Main verb
Purpose = Copulative verb
Purpose = Subject
Purpose = Object
Purpose = Object clause
Purpose = Object cluster
Purpose = IndObj
Purpose = Complement
Purpose = Copula-predicate
Purpose = Adjunct
Purpose = Disjunct
Purpose = Attribute
Purpose = Conj
Purpose = -

Reason = Main verb
Reason = Copulative verb
Reason = Subject
Reason = Object
Reason = Object clause
Reason = Object cluster
Reason = IndObj
Reason = Complement
Reason = Copula-predicate
Reason = Adjunct
Reason = Disjunct
Reason = Attribute
Reason = Conj
Reason = -

Cause = Main verb
Cause = Copulative verb

Cause = Subject
Cause = Object
Cause = Object clause
Cause = Object cluster
Cause = IndObj
Cause = Complement
Cause = Copula-predicate
Cause = Adjunct
Cause = Disjunct
Cause = Attribute
Cause = Conj
Cause = -

Possessor = Main verb
Possessor = Copulative verb
Possessor = Subject
Possessor = Object
Possessor = Object clause
Possessor = Object cluster
Possessor = IndObj
Possessor = Complement
Possessor = Copula-predicate
Possessor = Adjunct
Possessor = Disjunct
Possessor = Attribute
Possessor = Conj
Possessor = -

Identification = Main verb
Identification = Copulative verb
Identification = Subject
Identification = Object
Identification = Object clause
Identification = Object cluster
Identification = IndObj
Identification = Complement
Identification = Copula-predicate
Identification = Adjunct
Identification = Disjunct
Identification = Attribute
Identification = Conj
Identification = -

Classification = Main verb
Classification = Copulative verb
Classification = Subject

Classification = Object
Classification = Object clause
Classification = Object cluster
Classification = IndObj
Classification = Complement
Classification = Copula-predicate
Classification = Adjunct
Classification = Disjunct
Classification = Attribute
Classification = Conj
Classification = -

Quality = Main verb
Quality = Copulative verb
Quality = Subject
Quality = Object
Quality = Object clause
Quality = Object cluster
Quality = IndObj
Quality = Complement
Quality = Copula-predicate
Quality = Adjunct
Quality = Disjunct
Quality = Attribute
Quality = Conj
Quality = -

Existence = Main verb
Existence = Copulative verb
Existence = Subject
Existence = Object
Existence = Object clause
Existence = Object cluster
Existence = IndObj
Existence = Complement
Existence = Copula-predicate
Existence = Adjunct
Existence = Disjunct
Existence = Attribute
Existence = Conj
Existence = -

- = Main verb
- = Copulative verb
- = Subject
- = Object

```
- = Object clause
- = Object cluster
- = IndObj
- = Complement
- = Copula-predicate
- = Adjunct
- = Disjunct
- = Attribute
- = Conj
- = -
```

Figure 6.23. Human-readable list of all possible combinations of syntactic and semantic mappings that may appear in Genesis 1:1-2:3.

The last step transforms the results into human-friendly output. The code is shown in Figure 6.24. It is activated by pressing the "List combinations with frequencies" button.

```
'Prepare and show human-readable output - list combinations with
    frequencies

Private Sub cmdOutput2_Click()
Dim flagRefFound As Integer

toets11 = "Combinations of semantic functions and syntactic functions found
   in Genesis 1:1-2:3:" & vbCrLf & vbCrLf
toets11 = toets11 & "Total of possible combinations explored: " & totCombs
toets11 = toets11 & vbCrLf & "Total unique combinations found: " &
   totUniqueCombs & vbCrLf & vbCrLf

For m = 1 To 37 'Show table - show only occurring mappings
 For n = 1 To 14
  If refSemSyn(m, n, 2) <> "" Then
   toets11 = toets11 & "" & refSemSyn(m, n, 1) 'Create output - combination
     type
   toets11 = toets11 & " (" & combSemSyn(m, n) & "x)" & vbCrLf  'Create
     output - frequency
   toets11 = toets11 & "" & refSemSyn(m, n, 2) & vbCrLf  'List reference
     string after combination type
  End If
 Next
 toets11 = toets11 & vbCrLf
Next

Text1.Text = toets11

End Sub
```

Figure 6.24. VB6 code used to prepare and show human-readable output, listing all occurring mappings of semantic and syntactic functions and their frequencies.

The final results of this data-mining venture are shown in Figure 6.25. Summarised results are shown by listing all occurring combinations and their frequencies; drilled-down data is also provided by listing references to the clauses that represent the various combinations. The reseacher may now use these results to compare the linguistic theory regarding the mapping of these two modules with the examples found in the clause cube. This may either confirm or falsify existing hypotheses or prompt new ones.

```
Combinations of semantic functions and syntactic functions found in Genesis 1:1-2:3

Total of possible combinations explored: 540
Total unique combinations found: 39

Action = Main verb (46x)
  Gen01v01a - Clause 1
  Gen01v03a - Clause 5
  Gen01v04c - Clause 10
  Gen01v05a - Clause 11
  Gen01v05b - Clause 12
  Gen01v06a - Clause 15
  Gen01v07a - Clause 18
  Gen01v07b - Clause 19
  Gen01v08a - Clause 23
  Gen01v09a - Clause 26
  Gen01v10a - Clause 30
  Gen01v10b - Clause 31
  Gen01v11a - Clause 34
  Gen01v14a - Clause 44
  Gen01v16a - Clause 49
```

Gen01v17a - Clause 50
Gen01v20a - Clause 55
Gen01v20c - Clause 57
Gen01v21a - Clause 58
Gen01v21b - Clause 59
Gen01v22a - Clause 62
Gen01v22d - Clause 65
Gen01v24a - Clause 69
Gen01v25a - Clause 72
Gen01v26a - Clause 75
Gen01v26b - Clause 76
Gen01v27a - Clause 78
Gen01v27b - Clause 79
Gen01v27c - Clause 80
Gen01v28a - Clause 81
Gen01v28b - Clause 82
Gen01v28e - Clause 85
Gen01v28f - Clause 86
Gen01v28g - Clause 87
Gen01v29a - Clause 88
Gen01v29b - Clause 89
Gen01v31b - Clause 96
Gen02v01a - Clause 100
Gen02v02a - Clause 101
Gen02v02b - Clause 102
Gen02v02c - Clause 103
Gen02v02d - Clause 104
Gen02v03a - Clause 105
Gen02v03b - Clause 106
Gen02v03c - Clause 107
Gen02v03d - Clause 108

Action = Copulative verb (1x)
Gen01v06c - Clause 17

Position = Copula-predicate (1x)
Gen01v02c - Clause 4

Process = Main verb (16x)
Gen01v04a - Clause 8
Gen01v09b - Clause 27
Gen01v09c - Clause 28
Gen01v10c - Clause 32
Gen01v11b - Clause 35

Gen01v12a - Clause 38
Gen01v12c - Clause 40
Gen01v18b - Clause 51
Gen01v20b - Clause 56
Gen01v21c - Clause 60
Gen01v22c - Clause 64
Gen01v22e - Clause 66
Gen01v24b - Clause 70
Gen01v25b - Clause 73
Gen01v28d - Clause 84
Gen01v31a - Clause 95

State = Main verb (5x)
Gen01v14c - Clause 46
Gen01v15a - Clause 47
Gen01v22b - Clause 63
Gen01v28c - Clause 83
Gen01v29e - Clause 92

State = Copulative verb (23x)
Gen01v02a - Clause 2
Gen01v03b - Clause 6
Gen01v03c - Clause 7
Gen01v05c - Clause 13
Gen01v05d - Clause 14
Gen01v06b - Clause 16
Gen01v07e - Clause 22
Gen01v08b - Clause 24
Gen01v08c - Clause 25
Gen01v09d - Clause 29
Gen01v11d - Clause 37
Gen01v13a - Clause 42
Gen01v13b - Clause 43
Gen01v14b - Clause 45
Gen01v15b - Clause 48
Gen01v19a - Clause 53
Gen01v19b - Clause 54
Gen01v23a - Clause 67
Gen01v23b - Clause 68
Gen01v24c - Clause 71
Gen01v30c - Clause 94
Gen01v31d - Clause 98
Gen01v31e - Clause 99

Agent = Main verb (1x)
  Gen01v26c - Clause 77

Agent = Subject (28x)
  Gen01v01a - Clause 1
  Gen01v03a - Clause 5
  Gen01v04c - Clause 10
  Gen01v05a - Clause 11
  Gen01v06a - Clause 15
  Gen01v07a - Clause 18
  Gen01v08a - Clause 23
  Gen01v09a - Clause 26
  Gen01v10a - Clause 30
  Gen01v11a - Clause 34
  Gen01v14a - Clause 44
  Gen01v16a - Clause 49
  Gen01v17a - Clause 50
  Gen01v20a - Clause 55
  Gen01v20c - Clause 57
  Gen01v21a - Clause 58
  Gen01v21b - Clause 59
  Gen01v22a - Clause 62
  Gen01v24a - Clause 69
  Gen01v25a - Clause 72
  Gen01v26a - Clause 75
  Gen01v27a - Clause 78
  Gen01v28a - Clause 81
  Gen01v28b - Clause 82
  Gen01v29a - Clause 88
  Gen02v02a - Clause 101
  Gen02v03a - Clause 105
  Gen02v03d - Clause 108

Positioner = Subject (1x)
  Gen01v02c - Clause 4

Processed = Subject (14x)
  Gen01v04a - Clause 8
  Gen01v09b - Clause 27
  Gen01v09c - Clause 28
  Gen01v10c - Clause 32
  Gen01v11b - Clause 35
  Gen01v12a - Clause 38
  Gen01v12c - Clause 40

Gen01v18b - Clause 51
Gen01v20b - Clause 56
Gen01v21c - Clause 60
Gen01v22e - Clause 66
Gen01v24b - Clause 70
Gen01v25b - Clause 73
Gen01v31a - Clause 95

Zero = Subject (26x)
Gen01v02a - Clause 2
Gen01v02b - Clause 3
Gen01v03b - Clause 6
Gen01v03c - Clause 7
Gen01v04b - Clause 9
Gen01v05c - Clause 13
Gen01v05d - Clause 14
Gen01v06b - Clause 16
Gen01v07c - Clause 20
Gen01v07d - Clause 21
Gen01v08b - Clause 24
Gen01v08c - Clause 25
Gen01v11c - Clause 36
Gen01v12b - Clause 39
Gen01v13a - Clause 42
Gen01v13b - Clause 43
Gen01v14b - Clause 45
Gen01v19a - Clause 53
Gen01v19b - Clause 54
Gen01v23a - Clause 67
Gen01v23b - Clause 68
Gen01v29c - Clause 90
Gen01v29d - Clause 91
Gen01v30b - Clause 93
Gen01v31d - Clause 98
Gen01v31e - Clause 99

Zero = Conj (1x)
Gen01v29d - Clause 91

Patient = Subject (1x)
Gen02v01a - Clause 100

Patient = Object (12x)
Gen01v07a - Clause 18

Gen01v07b - Clause 19
Gen01v17a - Clause 50
Gen01v22a - Clause 62
Gen01v22d - Clause 65
Gen01v28a - Clause 81
Gen01v28e - Clause 85
Gen01v29b - Clause 89
Gen01v31a - Clause 95
Gen02v02a - Clause 101
Gen02v03a - Clause 105
Gen02v03b - Clause 106

Patient = Object clause (9x)
Gen01v03a - Clause 5
Gen01v04a - Clause 8
Gen01v10c - Clause 32
Gen01v11a - Clause 34
Gen01v12c - Clause 40
Gen01v18b - Clause 51
Gen01v21c - Clause 60
Gen01v24a - Clause 69
Gen01v25b - Clause 73

Patient = Object cluster (7x)
Gen01v06a - Clause 15
Gen01v09a - Clause 26
Gen01v14a - Clause 44
Gen01v20a - Clause 55
Gen01v26a - Clause 75
Gen01v28b - Clause 82
Gen01v29a - Clause 88

Patient = IndObj (5x)
Gen01v05a - Clause 11
Gen01v05b - Clause 12
Gen01v08a - Clause 23
Gen01v10a - Clause 30
Gen01v10b - Clause 31

Patient = Complement (6x)
Gen01v04c - Clause 10
Gen01v06c - Clause 17
Gen01v26c - Clause 77
Gen01v28g - Clause 87
Gen02v02c - Clause 103
Gen02v03c - Clause 107

Product = Object (17x)
Gen01v01a - Clause 1
Gen01v08a - Clause 23
Gen01v10a - Clause 30
Gen01v10b - Clause 31
Gen01v12a - Clause 38
Gen01v16a - Clause 49
Gen01v21a - Clause 58
Gen01v24b - Clause 70
Gen01v25a - Clause 72
Gen01v26b - Clause 76
Gen01v27a - Clause 78
Gen01v27b - Clause 79
Gen01v27c - Clause 80
Gen01v31b - Clause 96
Gen02v02b - Clause 102
Gen02v02d - Clause 104
Gen02v03d - Clause 108

Product = Complement (2x)
Gen01v05a - Clause 11
Gen01v05b - Clause 12

Receiver = IndObj (1x)
Gen01v29b - Clause 89

Location = Complement (5x)
Gen01v02c - Clause 4
Gen01v09b - Clause 27
Gen01v17a - Clause 50
Gen01v20c - Clause 57
Gen01v29d - Clause 91

Location = Copula-predicate (9x)
Gen01v02b - Clause 3
Gen01v06b - Clause 16
Gen01v07c - Clause 20
Gen01v07d - Clause 21
Gen01v11c - Clause 36
Gen01v12b - Clause 39
Gen01v14b - Clause 45
Gen01v29c - Clause 90

Gen01v30b - Clause 93

Location = Adjunct (4x)
Gen01v15a - Clause 47
Gen01v20c - Clause 57
Gen01v21b - Clause 59
Gen01v22e - Clause 66

Source = Complement (3x)
Gen01v04c - Clause 10
Gen01v06c - Clause 17
Gen01v07b - Clause 19

Reference = Complement (2x)
Gen01v11b - Clause 35
Gen01v20b - Clause 56

Beneficiary = IndObj (1x)
Gen01v29e - Clause 92

Manner = Complement (6x)
Gen01v07e - Clause 22
Gen01v09d - Clause 29
Gen01v11d - Clause 37
Gen01v15b - Clause 48
Gen01v24c - Clause 71
Gen01v30c - Clause 94

Manner = Adjunct (9x)
Gen01v12a - Clause 38
Gen01v21a - Clause 58
Gen01v22a - Clause 62
Gen01v24b - Clause 70
Gen01v25a - Clause 72
Gen01v26b - Clause 76
Gen01v27a - Clause 78
Gen01v27b - Clause 79
Gen02v03d - Clause 108

Time = Adjunct (4x)
Gen01v01a - Clause 1

Gen02v02a - Clause 101
Gen02v02c - Clause 103
Gen02v03c - Clause 107

Purpose = Copula-predicate (1x)
Gen01v29e - Clause 92

Purpose = Adjunct (4x)
Gen01v14b - Clause 45
Gen01v15a - Clause 47
Gen01v16a - Clause 49
Gen01v17a - Clause 50

Reason = Adjunct (1x)
Gen02v03b - Clause 106

Classification = Copula-predicate (3x)
Gen01v02a - Clause 2
Gen01v14c - Clause 46
Gen01v15a - Clause 47

Quality = Copula-predicate (7x)
Gen01v04b - Clause 9
Gen01v10d - Clause 33
Gen01v12d - Clause 41
Gen01v18d - Clause 52
Gen01v21d - Clause 61
Gen01v25c - Clause 74
Gen01v31c - Clause 97

Quality = Attribute (7x)
Gen01v05d - Clause 14
Gen01v08c - Clause 25
Gen01v13b - Clause 43
Gen01v19b - Clause 54
Gen01v23b - Clause 68
Gen01v27c - Clause 80
Gen01v31e - Clause 99

- = Disjunct (2x)
Gen01v29b - Clause 89

Gen01v31c - Clause 97

- = Conj (10x)
Gen01v04b - Clause 9
Gen01v10d - Clause 33
Gen01v11c - Clause 36
Gen01v12b - Clause 39
Gen01v12d - Clause 41
Gen01v18d - Clause 52
Gen01v21d - Clause 61
Gen01v25c - Clause 74
Gen01v30b - Clause 93
Gen02v03c - Clause 107

- = - (239x)
Gen01v01a - Clause 1
Gen01v02a - Clause 2
Gen01v02a - Clause 2
Gen01v02b - Clause 3
Gen01v02b - Clause 3
Gen01v02b - Clause 3
Gen01v02c - Clause 4
Gen01v02c - Clause 4
Gen01v03a - Clause 5
Gen01v03a - Clause 5
Gen01v03b - Clause 6
Gen01v03b - Clause 6
Gen01v03b - Clause 6
Gen01v03c - Clause 7
Gen01v03c - Clause 7
Gen01v03c - Clause 7
Gen01v04a - Clause 8
Gen01v04a - Clause 8
Gen01v04b - Clause 9
Gen01v04b - Clause 9
Gen01v04c - Clause 10
Gen01v05a - Clause 11
Gen01v05b - Clause 12
Gen01v05b - Clause 12
Gen01v05c - Clause 13
Gen01v05c - Clause 13
Gen01v05c - Clause 13
Gen01v05d - Clause 14
Gen01v05d - Clause 14
Gen01v06a - Clause 15
Gen01v06a - Clause 15

Gen01v06b - Clause 16
Gen01v06b - Clause 16
Gen01v06c - Clause 17
Gen01v06c - Clause 17
Gen01v07a - Clause 18
Gen01v07a - Clause 18
Gen01v07b - Clause 19
Gen01v07b - Clause 19
Gen01v07c - Clause 20
Gen01v07c - Clause 20
Gen01v07c - Clause 20
Gen01v07d - Clause 21
Gen01v07d - Clause 21
Gen01v07d - Clause 21
Gen01v07e - Clause 22
Gen01v07e - Clause 22
Gen01v07e - Clause 22
Gen01v08a - Clause 23
Gen01v08b - Clause 24
Gen01v08b - Clause 24
Gen01v08b - Clause 24
Gen01v08c - Clause 25
Gen01v08c - Clause 25
Gen01v09a - Clause 26
Gen01v09a - Clause 26
Gen01v09b - Clause 27
Gen01v09b - Clause 27
Gen01v09c - Clause 28
Gen01v09c - Clause 28
Gen01v09c - Clause 28
Gen01v09d - Clause 29
Gen01v09d - Clause 29
Gen01v09d - Clause 29
Gen01v10a - Clause 30
Gen01v10b - Clause 31
Gen01v10b - Clause 31
Gen01v10c - Clause 32
Gen01v10c - Clause 32
Gen01v10d - Clause 33
Gen01v10d - Clause 33
Gen01v10d - Clause 33
Gen01v11a - Clause 34
Gen01v11a - Clause 34
Gen01v11b - Clause 35
Gen01v11b - Clause 35
Gen01v11c - Clause 36

Gen01v11c - Clause 36
Gen01v11d - Clause 37
Gen01v11d - Clause 37
Gen01v11d - Clause 37
Gen01v12a - Clause 38
Gen01v12b - Clause 39
Gen01v12b - Clause 39
Gen01v12c - Clause 40
Gen01v12c - Clause 40
Gen01v12d - Clause 41
Gen01v12d - Clause 41
Gen01v12d - Clause 41
Gen01v13a - Clause 42
Gen01v13a - Clause 42
Gen01v13a - Clause 42
Gen01v13b - Clause 43
Gen01v13b - Clause 43
Gen01v14a - Clause 44
Gen01v14a - Clause 44
Gen01v14b - Clause 45
Gen01v14c - Clause 46
Gen01v14c - Clause 46
Gen01v14c - Clause 46
Gen01v15a - Clause 47
Gen01v15b - Clause 48
Gen01v15b - Clause 48
Gen01v15b - Clause 48
Gen01v16a - Clause 49
Gen01v18b - Clause 51
Gen01v18b - Clause 51
Gen01v18d - Clause 52
Gen01v18d - Clause 52
Gen01v18d - Clause 52
Gen01v19a - Clause 53
Gen01v19a - Clause 53
Gen01v19a - Clause 53
Gen01v19b - Clause 54
Gen01v19b - Clause 54
Gen01v20a - Clause 55
Gen01v20a - Clause 55
Gen01v20b - Clause 56
Gen01v20b - Clause 56
Gen01v20c - Clause 57
Gen01v21a - Clause 58
Gen01v21b - Clause 59
Gen01v21b - Clause 59

Gen01v21c - Clause 60
Gen01v21c - Clause 60
Gen01v21d - Clause 61
Gen01v21d - Clause 61
Gen01v21d - Clause 61
Gen01v22a - Clause 62
Gen01v22b - Clause 63
Gen01v22b - Clause 63
Gen01v22b - Clause 63
Gen01v22b - Clause 63
Gen01v22c - Clause 64
Gen01v22c - Clause 64
Gen01v22c - Clause 64
Gen01v22c - Clause 64
Gen01v22d - Clause 65
Gen01v22d - Clause 65
Gen01v22d - Clause 65
Gen01v22e - Clause 66
Gen01v22e - Clause 66
Gen01v23a - Clause 67
Gen01v23a - Clause 67
Gen01v23a - Clause 67
Gen01v23b - Clause 68
Gen01v23b - Clause 68
Gen01v24a - Clause 69
Gen01v24a - Clause 69
Gen01v24b - Clause 70
Gen01v24c - Clause 71
Gen01v24c - Clause 71
Gen01v24c - Clause 71
Gen01v25a - Clause 72
Gen01v25b - Clause 73
Gen01v25b - Clause 73
Gen01v25c - Clause 74
Gen01v25c - Clause 74
Gen01v25c - Clause 74
Gen01v26a - Clause 75
Gen01v26a - Clause 75
Gen01v26b - Clause 76
Gen01v26b - Clause 76
Gen01v26c - Clause 77
Gen01v26c - Clause 77
Gen01v26c - Clause 77
Gen01v27a - Clause 78
Gen01v27b - Clause 79
Gen01v27b - Clause 79

Gen01v27c - Clause 80
Gen01v27c - Clause 80
Gen01v28a - Clause 81
Gen01v28a - Clause 81
Gen01v28b - Clause 82
Gen01v28b - Clause 82
Gen01v28c - Clause 83
Gen01v28c - Clause 83
Gen01v28c - Clause 83
Gen01v28c - Clause 83
Gen01v28d - Clause 84
Gen01v28d - Clause 84
Gen01v28d - Clause 84
Gen01v28d - Clause 84
Gen01v28e - Clause 85
Gen01v28e - Clause 85
Gen01v28e - Clause 85
Gen01v28f - Clause 86
Gen01v28f - Clause 86
Gen01v28f - Clause 86
Gen01v28f - Clause 86
Gen01v28g - Clause 87
Gen01v28g - Clause 87
Gen01v28g - Clause 87
Gen01v29a - Clause 88
Gen01v29a - Clause 88
Gen01v29b - Clause 89
Gen01v29c - Clause 90
Gen01v29c - Clause 90
Gen01v29c - Clause 90
Gen01v29d - Clause 91
Gen01v29d - Clause 91
Gen01v29e - Clause 92
Gen01v29e - Clause 92
Gen01v30b - Clause 93
Gen01v30b - Clause 93
Gen01v30c - Clause 94
Gen01v30c - Clause 94
Gen01v30c - Clause 94
Gen01v31a - Clause 95
Gen01v31a - Clause 95
Gen01v31b - Clause 96
Gen01v31b - Clause 96
Gen01v31b - Clause 96
Gen01v31c - Clause 97
Gen01v31c - Clause 97

```
Gen01v31c - Clause 97
Gen01v31d - Clause 98
Gen01v31d - Clause 98
Gen01v31d - Clause 98
Gen01v31e - Clause 99
Gen01v31e - Clause 99
Gen02v01a - Clause 100
Gen02v01a - Clause 100
Gen02v01a - Clause 100
Gen02v02a - Clause 101
Gen02v02b - Clause 102
Gen02v02b - Clause 102
Gen02v02b - Clause 102
Gen02v02c - Clause 103
Gen02v02c - Clause 103
Gen02v02d - Clause 104
Gen02v02d - Clause 104
Gen02v02d - Clause 104
Gen02v03a - Clause 105
Gen02v03a - Clause 105
Gen02v03b - Clause 106
Gen02v03b - Clause 106
Gen02v03c - Clause 107
Gen02v03d - Clause 108
```

Figure 6.25. Final results of the experiment that analyses the mapping of semantic and syntactic functions in Genesis 1:1-2:3.

Although most of the mappings found confirm existing knowledge about syntactic and semantic functions (e.g., the main verb most often expresses an action, and location can function as argument or satellite), a number of interesting results was found. These will be briefly discussed below, some of which need more in-depth study.

- **Action expressed by a copulative verb**
  In Gen. 1:6c the phrase *vihi mavdil* (and let it be separating) has been tagged as a copulative verb. This is a tagging error, since the phrase should have been split up. *vihi* (and let it be) should have been tagged as a copulative verb (with no semantic function) and the participle *mavdil* (separating) as a copula-predicate

having the semantic function of attribute (and force, since it is an embedded predication.) (Cf. Gen. 1:2c for a similar construction.)

- **Position expressed by copula-predicate**
  In Gen. 1:2c a participle (*meraxefet* – hovering) has been tagged as a copula-predicate expressing the semantic function of position. This is correct, but the tagging of embedded verbs (participles and infinitives) and nominal clauses is problematic and needs to be refined and researched in more detail.

- **State expressed by copulative verbs and main verbs**
  The category of copulative verb is a sub-category of main verb (see Addendum E). In this project main verbs have not been subcategorised, except some copulative verbs. The tagging of main verbs, therefore, is inconsistent and should be corrected. See, for example, Gen. 1:2a (*hayta* – was – copulative verb), but Gen. 1:14c (*vehayu* – and let them be – main verb).

- **Agent expressed by main verb**
  In Gen. 1:26c the main verb *veyirdu* (and let them govern) has been coded as an agent due to the fact that the Hebrew prefix conjugation is synthetic, expressing both subject and verb. However, all other similar instances have been coded as actions. The tagging of synthetic Hebrew verbs is problematic and further research should look for a more elegant solution.

- **Zero expressed by conjunction**
  In Gen. 1:29d (*a$er bo pri ec zorea zara* – in which (there is) fruit of seed bearing trees) the relative particle was incorrectly coded as having the semantic function of zero. Although the particle could, in fact, be used independently as a subject having this semantic function (cf. Gen. 1:29c), it is here used only as a conjunction since the phrase *ec zorea zara* (fruit of seed bearing trees) functions as the subject having the zero semantic function.

- **Location expressed either by complement or copula-predicate**
  The researcher suspected that some copula-predicates could have been tagged as complements since it is a specific subtype of complement. However, working through all the hits listed revealed that the tagging was done consistently.[123] With reference to location, copula-predicate was used as the second argument in a nominal clause, while complement was used as the third argument in nominal or verbal clauses. Location may also be expressed by adjuncts. This confirms the hypothesis of Functional Grammar that location may be expressed by arguments or satellites (cf. Dik, 1997a: 120-122, 214-215, 243).

- **Patient expressed by indirect object**
  Various examples occur in the data set where a preposition phrase expresses the patient, e.g. Gen. 1:5a: *vayikra elohim la'or yom* (God called (to) the light day). Since it is unusual to regard a preposition phrase as direct object, these phrases have been tagged as indirect objects. However, this is incompatible with the definition of an indirect object supplied in Addendum E. The simplest solution will be to allow preposition phrases like these to be regarded and tagged as direct objects. Alternatively, the definition of indirect object could be changed to allow this syntactic function as second argument. More in-depth research is needed to explore these hypotheses prompted by the data-mining venture.

- **Manner expressed by complement**
  In a number of identical clauses (*vayehi xen* - and it was so; see, e.g., Gen. 1:7e) the adverb *xen* is used as a complement. Although this is supported by the syntax proposed in Addendum E, it suggests that the Functional Grammar theory should be adjusted. Dik (1997a: 230) defines manner as a satellite that occurs in actions, positions and processes. If the tagging as manner is correct in this experiment, the theory should be adapted to include manner as an argument in states. Alternatively one could reconsider the tagging – maybe *xen* could be tagged as quality, but even this would prompt an adjustment in Functional Grammar's description of semantic relations in non-verbal predications – "Property

[123] In an older version of the databank, however, Gen. 1:29c had initially been coded incorrectly (*al pney kol ha'arec* - on the whole earth - complement), but this error was corrected.

Assignment" is allocated only to adjectival and bare nominal predicate types (Dik, 1997a: 214).

- **Purpose expressed by copula-predicate**
  In Gen.1: 29e (*laxem yiheyeh le'oxla* – to you it will be as food) a copula-predicate (*le'oxla* – as food) is expressing a purpose satellite. Since purpose satellites should be constructions embedded within controlled predications, one should rather consider tagging *le'oxla* as classification, which, however, in turn prompts further research into the type of predicates, which may express "Class Inclusion". Dik (1997a: 202-206, 214) only mentions an "indefinite term", but it is not clear whether this should include preposition phrases.

- **Quality expressed by attribute**
  In various clauses (e.g. Gen. 1:5d) the semantic function of quality is allocated to attributes. For example, in the clause *vayehi voker yom exad* (and it was morning, day one), *yom exad* (day one) is a noun in apposition to the subject and functions as an adjectival modifier. Also compare Gen. 1:27c (*zaxar unkeva bara otam* – male and female, he created them). *Zaxar unkeva* (male and female) is an adjectival phrase consisting of two adjectives that describe the direct object in the clause. In both examples, however, the attributes are rather loosely coupled to the main clause and cannot simply be regarded as part of the noun clauses that they describe. Although the construction is slightly different from normal "property assignment" constituents – they are not predicates – they do seem to fit Dik's (1997a: 214) requirement of being adjectival or bare nominal elements. Dik (1997b: 401) discusses similar extra-clausal constituents on a pragmatic level and calls them "tails". The function of these "loosely adjoined constituents" is to "add a further specification to a term which is already contained in the clause". Since pragmatics is excluded from this study, these cases have provisionally been tagged as attributes with the semantic function of quality, but the analysis and tagging of this type of phrases should be researched in more detail.

- **Empty syntactic and semantic functions**
  Quite a number of phrases do not have any syntactic and/or semantic function. Empty functions are represented by a dash and will not be discussed further.

Although some of these "interesting" mappings may be ascribed to tagging errors, the data-mining process did demonstrate the rigour enforced by computer-assisted research. In addition, a number of cases have been identified that challenge existing hypotheses and suggest possibilities for further research. This demonstrates the idea that text mining not only helps linguists to test hypotheses but that they can also prompt new ones: "The computer can deal with far more information than you can, and even though it can't (yet) reason, it can show you opportunities for reasoning you would never find without it" (Unsworth, 2001).

6.6 Conclusion

After a short review of underlying concepts, such as threedimensional arrays, data cube operations and XML-based linguistic databases, this chapter focused on the benefits of text data mining facilitated by these technologies. Data mining concepts were introduced in a theoretical overview, some of which have then been applied in two experiments investigating aspects of the semantic and syntactic modules tagged in Genesis 1:1-2:3.

A computer-assisted exploration of the semantic data captured in the XML database of Genesis 1:1-2:3 illustrated the significance of such a text-mining venture. By enabling the researcher to look at the data from a different perspective, inconsistencies in the tagging of Genesis 1:1-2:3 were discovered. This proves the usability of using a computer to assist the human researcher by enforcing a certain level of rigour that is difficult or even impossible for humans to acquire. This rigour not only helps researchers to discover their own mistakes, but could also lead to new insights. In the words of Unsworth (2001): "The process that one goes through in order to develop, apply and compute these knowledge representations is unlike anything that humanities scholars, outside of philosophy, have ever been required to do. This method, or perhaps we should call it a heuristic, discovers a new horizon for humanities scholarship, a paradigm as powerful as New Criticism, New Historicism, or Deconstruction—indeed, very likely more powerful, because the rigor it requires will bring to our attention undocumented features of our own ideation, and coupled with enormous storage capacity and computational throughput, this method will present us with patterns and connections in the human record that we would otherwise never have found or examined."

Even in this small experiment some new patterns regarding combinations of semantic functions emerged, for example, a purpose embedded within a state predication. Although the results of the data-mining process provided overall support for the applicability of Dik's theory for a Hebrew text, it also revealed possible deficiencies in the definitions of some semantic functions, at least when these are applied to BH. It is rather surprising that, in the limited dataset used for this experiment, the discovery was made of four semantic role frameworks that contradict existing definitions of semantic functions. The discovery of such deficiencies could help grammarians to formulate and research new hypotheses, such as revising the definitions of semantic functions for specific languages. For example, the deficiencies referred to above should be researched using other languages as source data to determine if they are Hebrew specific or more general in nature.

The second experiment investigated the mapping of semantic and syntactic functions revealed. This endeavour reinforced the finding of a rigorous research method referred to above. Not only did it point out more tagging errors, but is also challenged existing assumptions about the syntactic and semantic theories that have been applied to the Hebrew text. Taking into regard that only two modules have been compared, one cannot help but wonder what the implications would be if other modules were involved. Some examples indicated the need to tag and integrate the pragmatics level into a clause cube.

Were the dataset larger one could probably expect many similar discoveries to prompt processes of knowledge invention. A more extensive database, including morphological information, could, for example, be very useful to programmatically study the valencies of Hebrew verbs. This again highlights the need for extensive linguistic databases. Humanities scholars should regard the encoding of their deep knowledge of texts to build XML software as "an investment in the longer-term future of their discipline" – in order to leave a persistent and usable legacy for succeeding generations (Flynn, 2002:59).

The programs used to execute the two text data mining ventures have been custom-made, using VB6. However, more research is needed to investigate the use of XML query languages and other data mining approaches and technologies to find more elegant, customisable ways to facilitate *ad hoc* explorations. Visualisation, for example, may provide a solution for easier navigation through selected subsets of the data, and this will be discussed in the following chapter.