46F. From Tags to Topic Maps: Using Marked-up Hebrew Text to Discover Linguistic Patterns

Jan H. Kroeze
University of Pretoria
jan.kroeze@up.ac.za

Theo J.D. Bothma
University of Pretoria
theo.bothma@up.ac.za

Machdel C. Matthee
University of Pretoria
machdel.matthee@up.ac.za

Abstract

The paper discusses a series of related techniques that prepare and transform raw linguistic data for advanced processing in order to unveil hidden grammatical patterns. It identifies XML as a suitable mark-up language to build an exploitable data bank of multi-dimensional data in the Hebrew text of the Old Testament. This concept is illustrated by tagging a transcription of Gen. 1:1-2:3 and manipulating this data bank. Transferring the data into a three-dimensional array allows advanced processing of the data in order to either confirm existing knowledge or to mine for new, yet undiscovered, linguistic features. Visualisation is discussed as a technique that enhances interaction between the human researcher and the computerised technologies supporting this process of knowledge creation. The empirical study is a small experiment that illustrates the viability and usefulness of the proposed expert devices as well as the benefits of applying information system techniques to linguistic databases.

Keywords

Text Data Mining, Data Warehousing, MOLAP, XML, Genesis

1. Introduction

Various scientific disciplines study information and communication technologies. Computer Science is a logical-mathematical science focusing on programming languages and algorithms themselves. Information Science is a humanities science exploring the relationships between data, information and knowledge.[1] Information Systems (sometimes called Informatics) is a social science that studies software used to manage information in organisations and society,[2] as well as the interaction of these software systems with people and communities. These divisions apply to the study of the use of computers in linguistics too. Natural language processing aims to simulate linguistic analysis and production by means of algorithms and may be regarded as the natural science branch of computational linguistics. Marking up texts to capture linguistic data, and the visualisation thereof, may be regarded as a humanistic venture to manage knowledge, the information science branch of computational linguistics. Building repositories of linguistic data

[1] Data refers to raw facts, information refers to data that has been processed and formatted to support decision making, and knowledge pertains to interpreted information (Rob & Coronel 2007).

[2] Information systems are "[t]he resources that enable the collection, management, control, and dissemination of information throughout an organization" (Connolly & Begg 2005).

and mining these data banks to discover patterns embedded in these data sets are related to database management and data warehousing, which form part of Information Systems. The borders between these divisions are neither absolute nor clear-cut. Electronic linguistic systems will usually comprise of a mixture of constructs from the three disciplines, the proportional scales of which will differ according to the nature and goals of each. While *electronic linguistic systems* is an umbrella term that refers to all three above-mentioned branches of computational linguistics, *linguistic information systems* refers to the discipline that facilitates "the advanced processing and analysis of linguistic data in order to find hidden patterns that are difficult for humans to uncover" (cf. Kroeze, 2007a). Linguistic information systems may be fairly simple, for example the reproduction of texts in an electronic format (allowing simple searches), or they could be highly complex by facilitating text mining and visualisation.

This paper touches on all three sub-disciplines of computational linguistics, investigating the use of XML tagging to capture linguistic categories in the Hebrew text of Gen. 1:1-2:3 and to construct a three-dimensional data bank (see Figure 1); using string processing algorithms to round-trip the data to and from the data bank and computer program; using array processing to explore the semantic patterns hidden in the marked-up text; and using a graphical visualisation to investigate the mapping of semantic and syntactic functions. The authors believe that these techniques facilitate the transformation of data to information and knowledge. The paper hopes to make a contribution by demonstrating the rigour enforced by the application of data-warehousing and data-mining concepts to a linguistic data bank. "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." (Unsworth 2001.)

2. Tagging linguistic data of the Hebrew text of Gen. 1:1-2:3

In order to create new knowledge a researcher always needs raw data. If no data is available, a data set should first be set up, which often implies the explication of tacit knowledge. In the case of languages, for example, the linguistic categories that are implicit in the text, should be spelled out in a consistent format to enable the researchers, or their computer programs, to ferret out hidden nuggets of knowledge embedded within the networks of functions on a specific level or even between various language modules.

XML is a mark-up language that allows the researcher to create a unique tag set to serve his/her design purpose. XML tags contain meaningful information and therefore differ from HTML tags that are mainly used to tell an internet browser how to format and display data. To display an XML file one needs a style sheet to fulfil the same purpose. The XML tags may contain semantic information or may be used to structure the data in the file.[3] This latter function, which may be used to simulate a database's columns and rows, is used in this project to store each clause's verse number as a primary key, the phrases in the clause, their translations, word group identifiers, syntactic and semantic functions (cf. Kroeze 2006). The data file is structured as illustrated in Figure 1.

```
<Genesis1>
   <clause>
      <clauseno> ... </clauseno>
      <headers>
         <header>Level</header>
         <header>Phrase1</header>
         <header>Phrase2</header>
         <header>Phrase3</header>
         <header>Phrase4</header>
         <header>Phrase5</header>
      </headers>
      <level1>
         <leveldesc>Phonetic representation:</leveldesc>
         <phrase1> ... </phrase1>
         <phrase2> ... </phrase2>
         <phrase3> ... </phrase3>
         <phrase4> ... </phrase4>
         <phrase5> ... </phrase5>
         </level1>
      <level2>
         <leveldesc>Translation:</leveldesc>
         <phrase1> ... </phrase5>
      </level2>
      <level3>
         <leveldesc>Phrase type:</leveldesc>
         <phrase1> ... </phrase5>
      </level3>
      <level4>
         <leveldesc>Syntactic function:</leveldesc>
         <phrase1> ... </phrase5>
      </level4>
      <level5>
         <leveldesc>Semantic function:</leveldesc>
         <phrase1> ... </phrase5>
      </level5>
   </clause>
   <clause> ... </clause>
   <clause> ... </clause>
   <clause> ... </clause> ...
</Genesis1>
```

Figure 1: The Structure of an XML File Used to Store Various Layers of Linguistic Information.

[3] A unique schema of tags, which fits the relevant data set in a natural way (Flynn 2002), can be compiled to be the equivalent of a database structure.

The XML file, discussed above, is a flat representation of a three-dimensional data structure or "data cube", [4] in which one dimension represents the clauses, another the phrases and the third the linguistic modules. The concept of a data cube of linguistic data is illustrated in Figure 2. The cube shows three clauses (Gen. 1:1a, 4c and 5a) consisting of four phrases each, analysed on five linguistic levels (phonetics, translation, word groups, syntax and semantics).

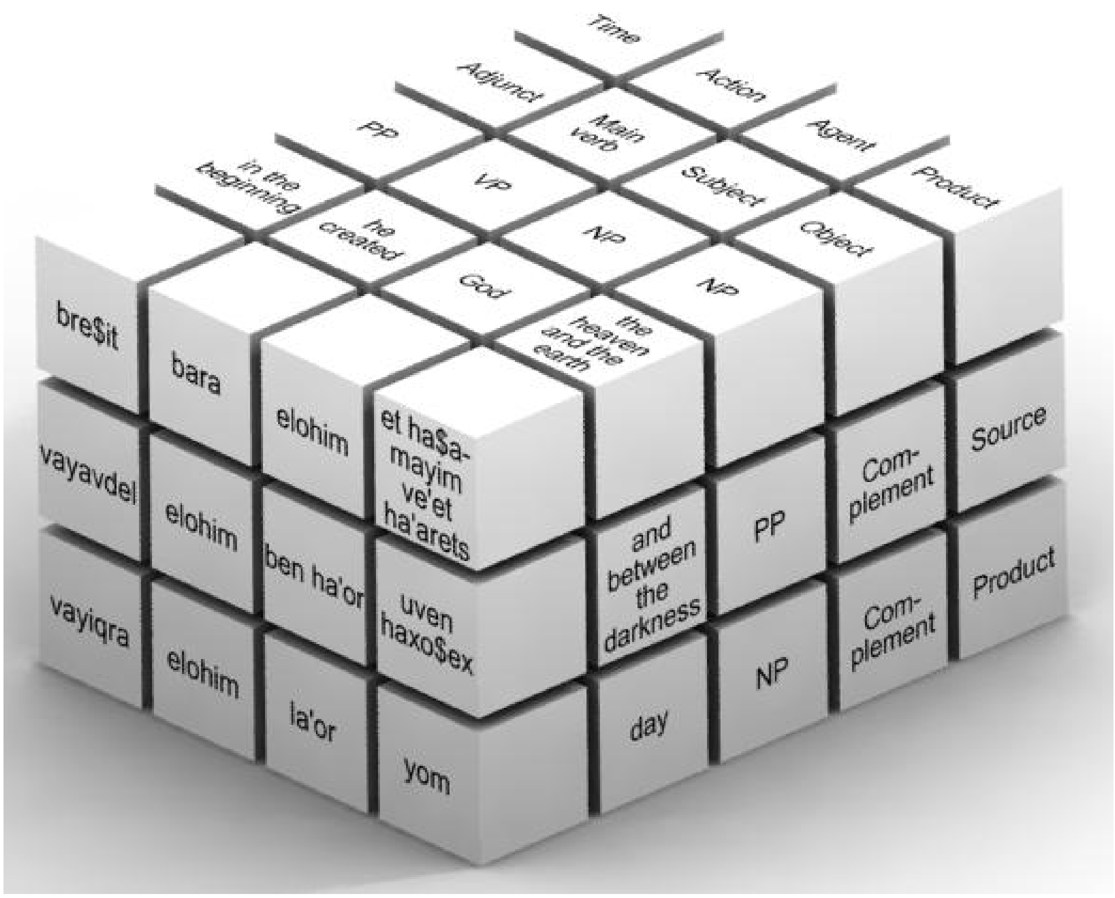


Figure 2: A Data Cube Containing Data of Three Clauses on Various Linguistic Levels.[5]

3. Using a three-dimensional array to represent the linguistic data set

Although the XML data set contains all the information needed for exploration, the verbosity caused by the mark-up makes it very difficult for a human researcher to spot any interesting patterns. In fact, the visibility of the structuring tags actually hides more than it reveals. Therefore, a computer program is needed to manipulate the data set to make the discovery process more human-friendly. Various software packages exist that are XML-enabled and some of these could certainly have been used for this purpose. However, in order to better understand and explain the various phases in the process, it was decided to write new code to represent and

[4] See Connolly & Begg (2005); Ponniah (2001). (A data cube in data warehousing does not have to be equilateral – the sizes of the various dimensions may differ according to a specific case's information structure and needs.)

[5] This orthographic representation was prepared by Christien J. Kroeze, a student at the University of Pretoria.

manipulate the data set in an efficient data structure in computer memory (RAM). Visual Basic 6 (VB6) was used for this experiment because it allows the use of three-dimensional arrays[6] and the creation of an executable file that can easily be disseminated. The array uses indexes to number and structure the various data elements stored in it, thus eliminating the need to use variables to keep track of word and clause order. After populating the array, named *Clause*, the data in the first two clauses would be represented as illustrated in Figure 3 (a dash implies a non-existing element) (cf. Kroeze 2004). The issue of scalability is not addressed in this experiment with a limited scope. Although small texts could easily be plugged in and analysed, assuming that the correct XML structure and codes are used for tagging, larger texts such as the whole Hebrew Bible could create memory and other implementation problems, issues which should be addressed in follow-up work. The focus of this paper is, however, on the contribution that the use of these technologies may make for linguistic researchers, and not primarily on the implementation problems. Therefore, issues of scalability and large file structures are excluded.

4. Converting the tagged data set to and from the array

Once XML has been identified as a suitable medium for permanent storage of the data, and a three-dimensional array for temporary, in-memory storage, one has to find a way to convert the data between the two mediums (round-tripping) (cf. Kroeze 2007b). Using the string processing facilities of VB6 the text-based XML file is opened within the program. Each line is read from the file using a loop with the same number of iterations than the number of clauses in the file. Line by line the XML tags are stripped away and the remaining linguistic data is stored into the array's variables. It is essential that the program code should reflect the structure of the XML file exactly to ensure that the linguistic data is inserted into the correct places within the array. If the data is only used for viewing purposes or as a data source for more complex manipulation, the contents of the array remain unchanged and do not have to be re-saved on permanent storage. The XML data bank may simply be opened and reused again at a later stage. When the program does save the data and the resulting version is an exact copy of the source file, the process is called ideal round-tripping (Smiljanić et al. 2002).

From a practical point of view, however, it does not make sense to replace existing data with an exact copy. But users will often feel the need to change or update the data itself. Having the data in an array in memory makes it quite easy to add other typical database functionalities (create, update and delete), which could typically be used by end-users to refine or expand the data set.

After editing, the data should be stored on the hard disk again for retrieval at a later stage. This process is a reversal of the method discussed above. The text file is opened and the old data overwritten. The algorithm adds the relevant XML tags to the linguistic elements before they are appended to the new file. One could also give the resulting file a different name to ensure the retention of the original data that may be used as a backup when necessary. Even if the data is changed by the user, the basic structure of the XML file should remain exactly the same. Adding or removing clauses, will, however, change the number of loop iterations needed when reading or writing the data from and to the XML file. Currently, a new XML version of all the data is

[6] A data cube can easily be created in many computer languages by declaring a multidimensional array (cf. Banchoff 1996).

created when data is edited by end-users, and a more elegant solution should be researched in order to allow incremental updates.

```
Clause(1, 1, 1) = "Gen01v01a"           Clause(2, 1, 1) = "Gen01v02a"
Clause(1, 1, 2) = "bre$it"              Clause(2, 1, 2) = "veha'arets"
Clause(1, 1, 3) = "in the beginning"    Clause(2, 1, 3) = "and the earth"
Clause(1, 1, 4) = "PP"                  Clause(2, 1, 4) = "NP"
Clause(1, 1, 5) = "Adjunct"             Clause(2, 1, 5) = "Subject"
Clause(1, 1, 6) = "Time"                Clause(2, 1, 6) = "Zero"
Clause(1, 2, 1) = "-"                   Clause(2, 2, 1) = "-"
Clause(1, 2, 2) = "bara"                Clause(2, 2, 2) = "hayta"
Clause(1, 2, 3) = "he created"          Clause(2, 2, 3) = "was"
Clause(1, 2, 4) = "VP"                  Clause(2, 2, 4) = "VP"
Clause(1, 2, 5) = "Main verb"           Clause(2, 2, 5) = "Copulative verb"
Clause(1, 2, 6) = "Action"              Clause(2, 2, 6) = "State"
Clause(1, 3, 1) = "-"                   Clause(2, 3, 1) = "-"
Clause(1, 3, 2) = "elohim"              Clause(2, 3, 2) = "tohu vavohu"
Clause(1, 3, 3) = "God"                 Clause(2, 3, 3) = "an emptiness and
Clause(1, 3, 4) = "NP"                     void"
Clause(1, 3, 5) = "Subject"             Clause(2, 3, 4) = "NP"
Clause(1, 3, 6) = "Agent"               Clause(2, 3, 5) = "Copula-predicate"
Clause(1, 4, 1) = "-"                   Clause(2, 3, 6) = "Classification"
Clause(1, 4, 2) = "et ha$amayim ve'et   Clause(2, 4, 1) = "-"
   ha'arets"                            Clause(2, 4, 2) = "-"
Clause(1, 4, 3) = "the heaven and the   Clause(2, 4, 3) = "-"
   earth"                               Clause(2, 4, 4) = "-"
Clause(1, 4, 4) = "NP"                  Clause(2, 4, 5) = "-"
Clause(1, 4, 5) = "Object"              Clause(2, 4, 6) = "-"
Clause(1, 4, 6) = "Product"             Clause(2, 5, 1) = "-"
Clause(1, 5, 1) = "-"                   Clause(2, 5, 2) = "-"
Clause(1, 5, 2) = "-"                   Clause(2, 5, 3) = "-"
Clause(1, 5, 3) = "-"                   Clause(2, 5, 4) = "-"
Clause(1, 5, 4) = "-"                   Clause(2, 5, 5) = "-"
Clause(1, 5, 5) = "-"                   Clause(2, 5, 6) = "-"
Clause(1, 5, 6) = "-"
```

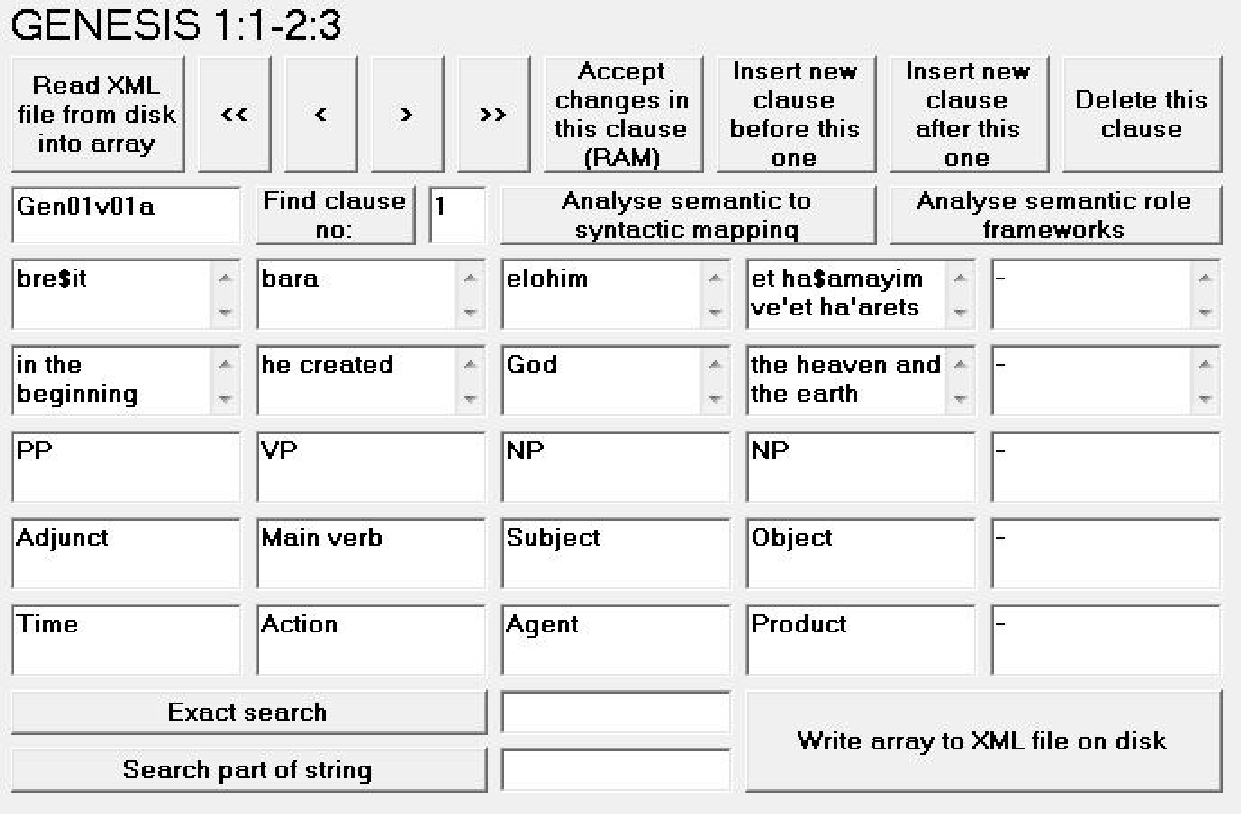
Figure 3: The Contents of a Three-Dimensional Array Containing Data of the First Two Clauses in Gen. 1. (The size of the second dimension is five to facilitate storage of longer clauses. No clause in the data set contains more than five phrases.)

5. Slicing and dicing the data set in the array

To view the data in a human-friendly way relevant subsets should be extracted from the data cube in the computer's memory. The concepts of slicing and dicing have been borrowed from data warehousing theory and practice. Although a typical data warehouse contains aggregated data, while our linguistic data cube contains detailed data, the data structure is also three-dimensional. Slicing off one layer from the Gen. 1:1-2:3 data cube reveals a two-dimensional subset that can easily be represented on paper or a computer screen. The most obvious slices[7] of

[7] A slice is a 'two-dimensional plane of the cube' (Ponniah 2001). Cf. Kroenke (2005), who discusses two-dimensional projections of three dimensions of student data.

a linguistic data cube are those that show one clause's analyses each (see Figure 4). Stacking all of these slices onto each other will rebuild the cube. Again, array processing is used to slice each clause's integrated analyses as a two-dimensional table. Dicing refers to the unveiling of a specific piece of data within the data set. It is used in this project for search functions. The program can search the array to find a required clause number, a whole phrase or a part of a string; when found, all the information related to the relevant clause is displayed (cf. Kroeze, Bothma & Matthee 2008).


GENESIS 1:1-2:3
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

Gen01v01a

1

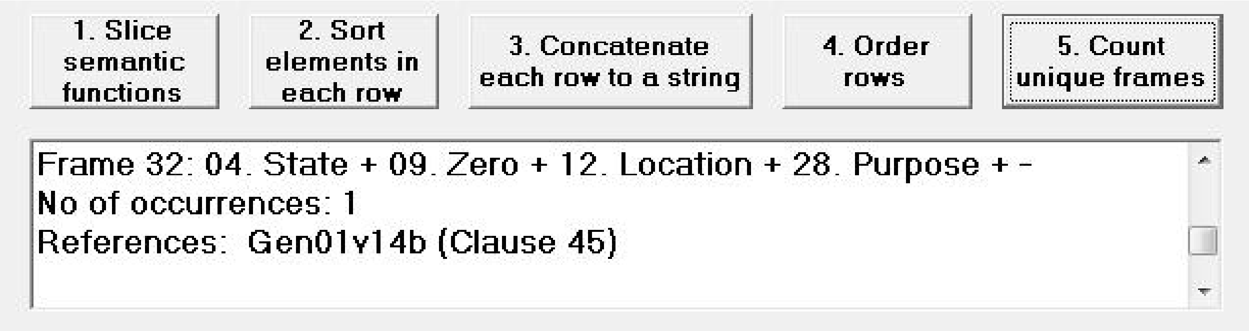
bre$it	bara	elohim	et ha$amayim ve'et ha'arets	-
in the beginning	he created	God	the heaven and the earth	-
PP	VP	NP	NP	-
Adjunct	Main verb	Subject	Object	-
Time	Action	Agent	Product	-

Figure 4: One Slice of the Data Cube that Shows the Various Analyses of Gen. 1:1a.

This feature, however useful, could easily have been simulated by a simple HTML or word processing implementation as well. The real power of the data cube is only demonstrated when one realises that the same data set may now be used to perform other and more advanced processing functions. Finding new patterns in the linguistic data, which may be regarded as an example of text (data) mining (Hearst 1999; Kroeze, Matthee & Bothma 2004; Kroeze & Matthee 2008), will be discussed in the following sections as examples of these functions.

6. Exploring semantic role frameworks in the data set

Another meaningful slice is one that isolates the semantic layer. This allows the researcher to study the various combinations of semantic functions or semantic role frameworks. In this experiment a procedure is used to identify unique frameworks and to count their frequencies. It first slices off the semantic layer and sorts the elements in each row; after concatenating each row to a string, the rows are ordered and the unique frames counted. The researcher then uses this information to compare existing definitions of semantic functions (cf. Kroeze 2007c). Even though the experimental data set is very small, a number of interesting combinations indicates that the definitions of some of the semantic functions should be revised, at least for Biblical Hebrew. For example, one instance was found of the semantic function of purpose which is embedded in a state predication (see Figure 5). According to the current definition of purpose (Dik 1997) this should not have been possible since purpose may only occur in controlled predications (actions and positions). Also compare verse 15a where the same phenomenon is repeated in almost exactly the same manner.

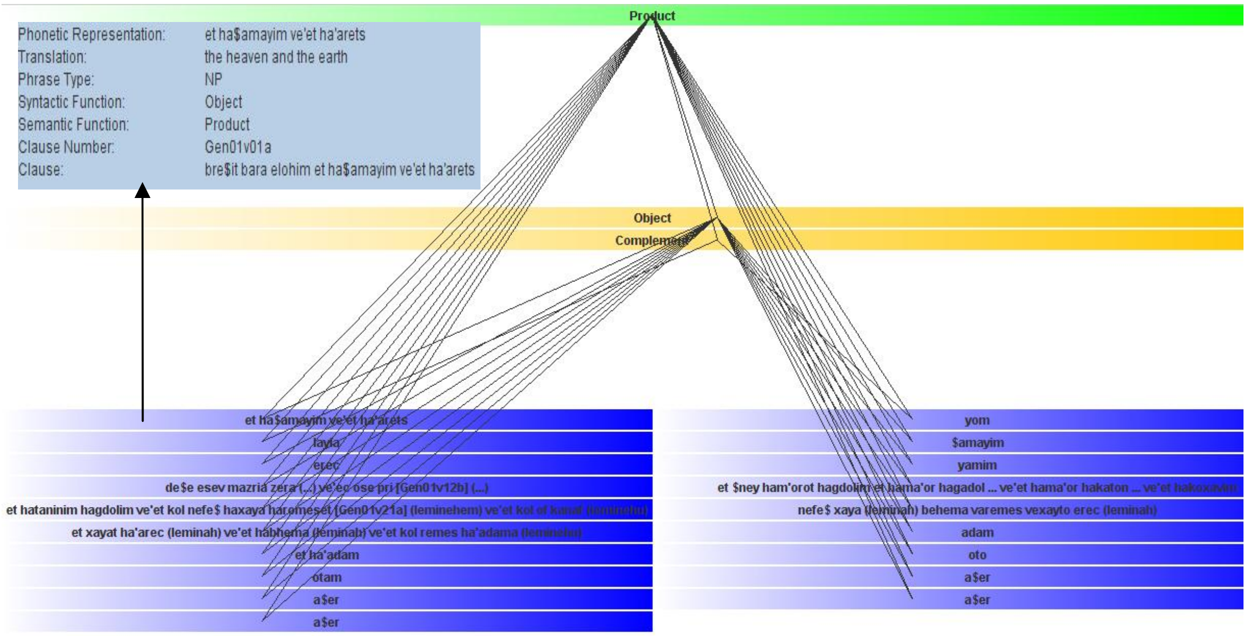

1. Slice semantic functions
2. Sort elements in each row
3. Concatenate each row to a string
4. Order rows
5. Count unique frames

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

Figure 5: A Screen Shot Showing a Semantic Frame Found in Gen. 1:14b that Suggests that the Semantic Function of Purpose Could also be Embedded in a State Predication (*let there be lights in the firmament of the heaven to separate the day from the night*).

Another interesting result is found in verses 7e, 9d, 11d, 15b, 24c and 30c ("and it was so"). According to Dik (1997) manner satellites occur in actions, positions and processes. Although this recurring example is not a very strong one, it does prompt the grammarian to reconsider the possibility of manner occurring in states as well.

These procedures illustrate how the process of text mining may catch up on gaps in existing linguistic knowledge, for example when a general linguistic theory is applied to and tested on a specific language. The results of this experiment represent new information that may be used to create knowledge, but they are rather static. The user is presented with one set of semantic frameworks in a text-based format and (s)he should work through the document in a linear way. The next paragraph will discuss another manipulation strategy that allows the user to search for linguistic patterns by means of a graphical tool that allows one to change or fine-tune the parameters (see Figure 6).



Phonetic Representation:	et ha$amayim ve'et ha'arets
Translation:	the heaven and the earth
Phrase Type:	NP
Syntactic Function:	Object
Semantic Function:	Product
Clause Number:	Gen01v01a
Clause:	bre$it bara elohim et ha$amayim ve'et ha'arets

Product
Object
Complement
et ha$amayim ve'et ha'arets
layla
erec
de$e esev mazria zera (...) ve'ec ose pri [Gen01v12b] (...)
et hataninim hagdolim ve'et kol nefe$ haxaya haromeset [Gen01v21a] (leminehem) ve'et kol of kanaf (leminehu)
et xayat ha'arec (leminah) ve'et habhema (leminah) ve'et kol remes ha'adama (leminehu)
et ha'adam
otam
a$er
a$er
yom
$amayim
yamim
et $ney ham'orot hagdolim et hama'or hagadol ... ve'et hama'or hakaton ... ve'et hakoxavim
nefe$ xaya (leminah) behema varemes vexayto erec (leminah)
adam
oto
a$er
a$er

Figure 6: A Screen Shot of a Visualisation of the Network Linking the Semantic Function of Product to the Syntactic Functions of Either Complement or Object, as Found in Various Clauses in the Data Set.

7. Visualising the mapping of semantic roles on syntactic functions in the data set

An interactive visualising tool enables the researcher to look at a data set from various perspectives. An example of such a tool is a graphical topic map[8] that shows all the relationships between phrases, semantic roles and syntactic functions in the data set. A new program, using the same XML data set, was created to facilitate link analysis between these nodes (cf. Kroeze, Bothma, Matthee & Kroeze 2008).[9] By adding or deleting filters the user may focus on subsets within the numerous relations. For the creation of Figure 6 a filter was selected to isolate all phrases with the semantic function of product. It shows that, in Gen. 1:1-2:3, the semantic function of product is realised by the syntactic functions of object and complement. When the user hovers with the mouse over one of the phrases, more clause detail is shown in a tool-tip.

8. Conclusion

This experiment demonstrated the process of creating knowledge by means of the interpretation of information extracted by digital tools from raw linguistic data. It proved that XML is a suitable mark-up language to build a three-dimensional data bank capturing information of

[8] Compare Bradley (2003).

[9] This tool was created by Jan C.W. Kroeze, a student at the University of Pretoria.

various language modules. It showed that array technology provides an efficient way, not only to round-trip this data to and from computer memory, but also to view and manipulate it in order to reveal linguistic relations embedded in the marked-up data. It explored graphical visualisation as a powerful, experimental way to search for patterns in the data set. The authors trust that it made a small but significant contribution to linguistic information systems as a humanistic endeavour. Although these findings may only be valid with reference to the small experiment of this study, we may conclude by expressing our trust and hope that there are promising indications that the use of linguistic information systems in larger texts could lead to a variety of knowledge-creation ventures.

References

Banchoff, T.F., Beyond the Third Dimension: Geometry, Computer Graphics, and Higher Dimensions, 2nd edition, Scientific American Library, New York, NY, 1996.

Bradley, J., “Finding a Middle Ground Between ‘Determinism’ and ‘Aesthetic Indeterminacy’: a Model for Text Analysis Tools”, Literary and Linguistic Computing, 18(2): 185-207, 2003.

Connolly, T.M., and Begg, C.E., Database Systems: a Practical Approach to Design, Implementation, and Management, 4th edition, Pearson, Essex, England, 2005.

Dik, S.C., The Theory of Functional Grammar, Part 1, The Structure of the Clause (edited by Kees Hengeveld), 2nd edition, Mouton de Gruyter, Berlin, Germany, 1997.

Flynn, P., “Is there Life beyond the Web?”, Literary and Linguistic Computing, 17(1): 49-59, 2002.

Hearst, M.A., “Untangling Text Data Mining”, Proceedings of ACL’99: the 37th Annual Meeting of the Association for Computational Linguistics, University of Maryland, June 20–26 (invited paper), 1999. [Online]. Available:
http://www.ai.mit.edu/people/jimmylin/papers/Hearst99a.pdf [Cited 20 August 2002].

Kroenke, D.M., Database Concepts, 2nd edition, Pearson, Upper Saddle River, NJ, 2005.

Kroeze, J.H., “Towards a Multidimensional Linguistic Database of Biblical Hebrew Clauses”, Journal of Northwest Semitic Languages (JNSL), 30(2): 99-120, 2004.

Kroeze, J.H., “Building and Displaying a Biblical Hebrew Linguistics Data Cube using XML”, Paper read at Israeli Seminar on Computational Linguistics (ISCOL), Haifa, Israel, 29 June 2006. [Onine.] Available: http://mila.cs.technion.ac.il/english/events/ISCOL/2006/index.html [Cited 27 February 2008].

Kroeze, J.H., “Linguistic Information [Systems] - a Humanistic Endeavour”, Innovate, 02: 38-39, 2007a. (Published by University of Pretoria, EBIT.)

Kroeze, J.H., “Round-tripping Biblical Hebrew Linguistic Data”, In: Managing Worldwide Operations and Communications with Information Technology (Proceedings of 2007 Information Resources Management Association, International Conference, Vancouver, British Columbia, Canada, May 19-23, 2007), edited by M. Khosrow-Pour, IGI Publishing, Hershey, PA, 1010-1012, 2007b.

Kroeze, J.H., “A Computer-assisted Exploration of the Semantic Role Frameworks in Genesis 1:1-2:3”, Journal of Northwest Semitic Languages (JNSL), 33(1): 55-76, 2007c.

Kroeze, J.H., Bothma, T.J.D., and Matthee, M.C., “Slicing and dicing a linguistic data cube”, Accepted for publication in Handbook of Research on Text and Web Mining Technologies, edited by M. Song, 2008 (Forthcoming).

Kroeze, J.H., Bothma, T.J.D., Matthee, M.C., and Kroeze, J.C.W., "Visualizing Mappings of Semantic and Syntactic Functions", 2008 (Forthcoming). (Paper accepted for INFOS2008.)

Kroeze, J.H., Matthee, M.C., and Bothma, T.J.D., "Differentiating Between Data-mining and Text-mining Terminology", South African Journal of Information Management (SAJIM), 6(4), 2004. [Online]. Available: http://www.sajim.co.za/ [Cited 1 March 2008].

Kroeze, J.H., and Matthee, M.C., "Discovering Unknown Patterns in Free Text" (updated version), Accepted for publication in Encyclopedia of Data Warehousing and Mining - 2nd Edition, edited by J. Wang, 2008 (Forthcoming).

Ponniah, P., Data Warehousing Fundamentals: a Comprehensive Guide for IT Professionals, John Wiley, New York, NY, 2001.

Rob, P., and Coronel, C., Database Systems: Design, Implementation, and Management, 7th edition, Course Technology, Boston, MA, 2007.

Smiljanić, M., Blanken, H., Van Keulen, M., and Jonker, W., "Distributed XML Database Systems", 2002. [Online.] Available: http://www.purl.org/utwente/38064 [Cited 27 July 2005].

Unsworth, J., "Knowledge Representation in Humanities Computing", Essays in Humanities Computing, 2001. [Online.] Available: http://www.digitalhumanities.org/Essays/ [Cited 23 November 2005].