Text-fabric: handling Biblical data with IKEA logistics

Dirk Roorda
Data Archiving and Networked Services
www.dans.knaw.nl
dirk.roorda@dans.knaw.nl

Abstract

The BHSA (Biblia Hebraica Stuttgartensia Amstelodamensis) is the BHS text plus the linguistic annotations of the Eep Talstra Centre for Bible and Computer.
The BHSA is available as a data set in Text-Fabric format. Text-Fabric is a minimalistic model to represent text: it provides addresses for all textual objects, so that it is easy to add arbitrary information at all textual levels, precisely and firmly anchored. A Text-Fabric resource resembles an IKEA ware house. The parts are nicely separated and stacked, so that they can be retrieved easily, to be combined into meaningful output later on. A consequence is that different teams with divergent purposes still can add to the same body of work, with a minimum of interference or duplication of work. Text-Fabric has helped with various types of data construction work, of which the most visible is the website SHEBANQ. We focus on two recent data combination jobs, (A) treebanks from the BHSA data and (B) a detailed comparison of the morphology in the BHSA and in the Open Scriptures effort. As the OSM is not yet finished, the comparison is repeatable.

List of terms

BHS Biblia Hebraica Stuttgartensia, a transcription of the Leningrad Codex. https://en.wikipedia.org/wiki/Leningrad_Codex http://www.bibelwissenschaft.de/startseite/wissenschaftliche-bibelausgaben/biblia-hebraica/bhs/. (Elliger 1997)

BHSA Biblia Hebraica Stuttgartensia Amstelodamensis. Text and linguistic annotations by the Eep Talstra Centre for Bible and Computer (previously called WIVU). https://github.com/ETCBC/bhsa (Peursen 2015), (Roorda and Kingham 2017)

DANS Data Archiving and Networked Services. The Netherlands Institute for permanent access to digital research resources. An institute of the Dutch Academy KNAW and funding organization NWO. https://dans.knaw.nl

Emdros Text database engine for analyzed or annotated text. Written by Ulrik Sandborg-Petersen to implement the ideas of (Doedens 1994). https://emdros.org (Petersen 2004)

ETCBC Eep Talstra Centre for Bible and Computer, formerly WIVU (Werkgroep Informatica, Vrije Universiteit). http://etcbc.nl

Jupyter Notebook Programming environment, modeled after laboratory notebooks. Originally for Python programs. Python-in-a-notebook is a popular way to do explorative data analysis and tell computing narratives. http://jupyter.org

MQL Mini Query Language. User friendly query language, used in Emdros. MQL is *topographic* as defined in (Doedens 1994).

OSM Open Scriptures Morphology. Crowd-sourced project to provide morphology tags to all words in the Hebrew Bible, resulting in a data set under an open license. http://openscriptures.github.io/morphhb/HomeFiles/Parse.html and https://github.com/openscriptures/morphhb

Text-Fabric A data model, file format and Python software package for text plus annotations. https://github.com/Dans-labs/text-fabric

SHEBANQ Website by the ETCBC to display the BHSA and let users query it and save those queries. https://shebanq.ancient-data.org . (Roorda 2017)

warp In weaving: the fixed threads through which the loom leads another thread, the weft. See https://en.wikipedia.org/wiki/Weaving

weave The result of weaving: a fabric.

weft In weaving: the thread that is pulled by the loom through the fixed threads of the warp.

WLC Westminster Leningrad Codex, an online digital version of the Leningrad Codex. https://en.wikipedia.org/wiki/Leningrad_Codex , maintained by the J. Alan Groves Center for Advanced Biblical Research at the Westminster Theological Seminary. The plain text is nearly equal to that of the BHS.

1. Data - logic - logistics

The Hebrew Bible is part of our cultural heritage. It is a complex entity, if indeed, it is a single entity. The Masoretic text at least is readily available as a unity, albeit in several manifestations and editions.

1.1 Data

We use the BHS, or rather, the plain text of it, and we consider it our data. In fact, it is our working hypothesis that we can regard this text as a set of facts. The purpose is to subject the text to studies, critical and non-critical. While such studies may shed all kinds of lights on the origins of the text, it all starts by making the text in to a researchable object.

This particular text is a sequence of expressions in language, in ancient Hebrew and Aramaic, which brings with it a lot of obvious and hidden structure. We can unravel a lot of that structure by means of algorithms, and even more by algorithms that receive nudges from humans. The result is a large amount of linguistic annotations, which together form another set of data, closely linked to the plain text.

For the purpose of this article, text and annotations together form our data.

1.2 Logic

As observed, this data has a lot of structure. We can describe structure by means of a model. A data model breaks data down into bits and pieces and describes the possible relations between them. We deal with text, so we need a model that covers text and textual objects. We expect the basic properties of textual units to be expressible in the model: that they form a sequence and that they lie embedded in each other. We want to see the linguistic properties applied to text: labels for inflections, part-of-speech, constituent boundaries, discourse properties, text types.

The data model, in specifying how the text and the linguistic features hang together, embodies the logic of our data.

1.3 Logistics

As research is carried out, individuals and teams from all over the world study the text, absorb its linguistic properties, and try to answer research questions that come out of their interests. There are broad questions of origin, interpretation and reception on the one hand, but more often focused questions on syntactic variation, the attempt to connect the text with ancient geography and history, and the need to solve exegetical puzzles. When teams publish their findings, it may take the shape of articles, but more and more data sets also count as results. The result data must be linked to the original dataset. The challenge is to integrate result data into a common body of data, without ironing

out the differences of opinions that do arise. All fruits of research should be interoperable, while the provenance of all contributions is kept transparent. When that happens, researchers are may utilize unexpected combinations of data. In order to play this game, we need logistics to make data interchange efficient. One of the biggest hurdles in data science is the amount of effort that it takes to preprocess data into forms that are usable for the research question at hand. If we can factor out significant portions of the preprocessing, researchers in the field are less burdened with the mundane tasks of a field that is mostly foreign to them.

In this paper we have already said all about data that we needed to say. We shall briefly explain our choice of logic. But our focus is on the logistics part, of which we shall give two real world examples.

2. Logic: annotated graphs

There is a simple pattern that is surprisingly capable of capturing a lot of textual and linguistic structure. This model has been introduced in (Doedens 1994), implemented in Emdros, and simplified in Text-Fabric. It is known as the monad-object-feature model, but we follow Text-Fabric in calling it an annotated graph model. A graph is a structure with nodes and edges between them.

2.1 Slots

The basic entity of text is a *slot*, i.e. a textual position (*monad* in (Doedens 1994) and Emdros). A textual position is occupied by a textual unit, such as a character, a morpheme, or a word. The exact choice of granularity is mostly a matter of convenience. In Chinese it could be the character, in Babylonian the glyph, in Hebrew the morpheme, or word, or anything that is separated by spaces. Slots have numbers, from 1 onwards to the last textual position, and if we fill them with strings that represent the textual units, we just have the plain text, but with the bonus that we can address every single textual unit by a unique number. We go one step further: we identify the slots with their numbers.

2.2 Nodes

Text has more complicated objects, all built from, eventually, the textual units.

We model textual objects by the subsets of slots that they occupy. Every subset of slots that is the representation of a textual object, gets a unique number. Now we can tell what the nodes of our graph are. They are numbers. The first n numbers are given to textual units, in the order of their appearance in the text. The numbers higher than n are given to the other textual objects. In case of the BHSA, the slots correspond to *words*, we have slots going from 1 to 426,584, and the other nodes have numbers from 426,585 to 1,446,635. So, half a million words and a million other objects. Textual objects come in types: words, phrases, clauses, sentences, lexemes, books, etc. We add the concept of type to our nodes. Every node belongs to exactly one type. It is allowed to have two nodes with a distinct type and the same set of slots.

2.3 Edges

There are various relationships between textual objects. We represent a relationship with a set of *edges* between nodes. An edge is just a pair of numbers. In case of the BHSA, we have three sets of edges: functional parent, distributional parent, and mother. They indicate embedding and dependencies of objects, according to several ways of linguistic analysis. Note in passing that this model deals with multiple hierarchies.

2.4 Annotations

The graph of our model is now clear: we know what the nodes and the edges are. We add *annotations* to the graph. Every node and every edge can be annotated by a piece of information, typically a string or a number. In Text-Fabric and Emdros we call annotations *features*. A feature has a name, and it maps nodes or edges to values. While the graph is an abstract system of numbers and sets of pairs of numbers, the features embody the tangible aspects of the text. For example, the feature part-of-speech maps each slot to the part-of-speech of the word occupying that slot. People familiar with the statistical package R recognize that our features are their vectors. Indeed, a feature in the BHSA is a vector consisting of one and a half million values.

This is really it. These few simple, generic concepts allow us to express a wealth of textual and linguistic data in such a way that they can be retrieved easily.

2.5 Emdros and Text-Fabric

There are two tools available that work with the graph and features model of text.

2.5.1 Emdros

Emdros is a database engine with a natural query language, MQL, in which you can express syntactical queries quite easily. It stores data in the monad-object-feature model.

It can be run from the command line, and it comes with a graphical user interface.

MQL has a complete set of features, and Emdros implements it efficiently.

2.5.2 Text-Fabric

Text-Fabric is a tool that helps you explore the graph by walking over it in high speed, grabbing the bits of information that you need as you go. It has no interface, except for an API, a programmer's interface. It is written in Python, and the most intuitive way to interact with it is in a Jupyter Notebook. Text-Fabric has a built-in search tool, similar to MQL, but lacks a few key primitives of MQL (NOTEXIST). It is also less efficient than MQL. On the other hand, TF-searches run inside your programs, and you can easily post-process the results without friction. That is more powerful than MQL queries.

2.5.3 SHEBANQ

The website SHEBANQ displays the BHS text with annotated BHSA data and offers users to query that data by means of MQL and save those queries so that they can be cited and published. SHEBANQ has been built with Text-Fabric as the main construction tool. Now, Text-Fabric can import features from MQL and export them to MQL. When researchers contribute new data, Text-Fabric picks up the new features, wraps it into MQL, which is fed to SHEBANQ, so that people can use the new data in their queries. Seen this way, SHEBANQ is a good example of how MQL and Text-Fabric have their complementary qualities.

At this point, however, we are talking *logistics.* We'd better start a new section for that.

3. Logistics: (re)combining data

We draw inspiration from unexpected corners.

3.1 Ikea

Consider this narrative:

> You need a new piece of furniture, a kitchen cabinet. You make a round of visits to your local furniture shops, and in shop A you find a cabinet with just the right number of drawers. In shop B you find a cabinet built from materials that appeal to you. And in shop C you find a cabinet with design elements that you really like. You decide you want the best of three worlds. You go to shop A, B and C and buy the three cabinets. You strip cabinet A bare by removing all non-structural parts. You break up cabinet B into its components, which you chisel until they fit unto the frame of cabinet A. You cherry-pick the fancy grips and knobs from cabinet C and fix them unto your new creation.

Few people adopt this way of working when they install a kitchen. Yet in data preprocessing, this strategy is all too familiar. In the furniture world, Ikea has solved the problem by offering series of parts that you can choose independently from each other. I propose to borrow Ikea's solution when we deal with textual resources, especially Biblical resources.

3.2 Modular Biblical resources

For example, you want the syntax of the BHSA, the morphology from the OSM, and the prosody of Leuven (hypothetically). For your research, you need this all in one set, meaningfully organized. What you tend to get, is: all things from all sources, organized as local wisdom prescribes. So, you will have to ignore the morphology of the BHSA, the full-text of the OSM, and the syntax of Leuven. Before you can start your research proper, you must write three programs that peel off the packaging, and a few other programs to align the treasures that come out of it.

Exactly at the point that you succeeded in doing this, it is worthwhile to intercept the data.

Because now you have the clean parts, in an interoperable shape. This is what everybody after you also wants! It would have saved you time if the BHSA, the OSM and the Leuven group would have delivered the data in this modular form. That would be Ikea-logistics for Biblical resources. The good news is, that the BHSA has already adopted it.

3.3 Text-Fabric and MQL (revisited)

Both MQL and Text-Fabric work with the same model for text. But they handle it a bit differently. Because I needed a workhorse that could shift data around, rather than query it, I had to pay more attention to the efficiency of combining data. The basic difference is that an MQL dump has all features in one file, and Text-Fabric has just one feature per file. MQL adheres to the record-by record-view, Text-Fabric adopts the column-by-column view. That has some profound consequences.

3.4 Separation of concerns

An important principle in software design is *separation of concerns*. Textual display is one concern, morphology another, syntax and semantics are again different concerns. Linking to outside sources comes on top. Separation of concerns means that you can pursue any of these goals without interfering with the other goals. Essentially, that means: without even touching the data belonging to those other goals.

For example, say the BHSA is good in syntax, and it exports 10 syntactical features. Whereas the OSM is good in morphology, exporting 5 morphology features. Likewise, Leuven exports 3 prosody features. The three groups create them in their own time, update them with the frequency they find

feasible. In MQL, every time something changes in one of the features, the whole database must be updated. Somebody must coordinate all those updates into one system.

In Text-Fabric, each feature sits in its own file. A Text-Fabric data set is a collection of files, which may sit in multiple directories, everywhere on your system. When you call Text-Fabric, you point to the locations where it has to look for feature files. It then collects all features found, and lets the user load them into memory on demand. In the example above, the BHSA syntax is just a bunch of 10 files, the OSM morphology is a bunch of 5 files, and the Leuven prosody is a set of three files. Only when a user wants to use them in conjunction, they will be drawn into the same runtime environment, from the diverse places on the filesystem where they happen to be downloaded to. This prevents a significant amount of friction in the chain from data producer to data consumer.

4. Example A: trees

The BHSA offers tree structures for each sentence. The tree for Job 3:16 looks like this

> (S(C(Ccoor(CP(cj 0))(PP(pp 1)(U(n 2))(U(vb 3)))(NegP(ng 4))(VP(vb 5)))(Ccoor(PP(pp 6)(n 7)(Cattr(NegP(ng 8))(VP(vb 9))(NP(n 10)))))))

The concrete words have been left out, we see sequence numbers instead. It may seem that this makes it more difficult to use the tree fruitfully, but we show some scenarios where the reverse is true. Suppose *you* want tree structures of sentences and a fully pointed Hebrew representation of the words inside. But your semantic colleague only wants a lexeme identifier in ASCII notation. And your linguistic co-worker would really be helped by a phonetic approximation of the words, plus a gloss.

Separation of concerns dictates that the feature tree is only concerned with the tree structure, not with the concrete representation of the words. For the latter we choose other features, possible from other sources, created by other teams. We frown upon a marriage between the tree structures and the word representations. The provider of the trees does not have to make choices about word representation, nor does he have to multiply the trees for each possible set of desiderata. Conversely, the phonetic data is the result of a completely independent enterprise. Maybe a future project will deliver better phonetic strings. We need to be able to switch to them without interfering with the programs that deliver the trees.

In order to satisfy the various user requirements, we weave the trees and the representations on demand. This is the specialty of Text-Fabric. In order to get the trees, we first parse the string into a nested data structure (this is generic programming), then we pick the word numbers, add the starting position of the sentence, and use the resulting numbers to look up the desired feature values for those words. In that way, we can get easily an output like this:

```
1  S
2   C
3    Ccoor
4     CP
5      cj {HC} "וְאִ֫" [ˈʔô] "<or>"
4     PP
5      pp {HR} "כְ" [k̲ᵊ] "<as>"
5      U
6       n {HNcmsa} "נֵ֫פֶל" [nˈēfel] "<miscarriage>"
```

```
5       U
6         vb {HVqsmsa} "טָמוּן" [ˈṭāmûn] "<hide>"
4     NegP
5       ng {HTn} "לֹא" [lˈō] "<not>"
4     VP
5       vb {HVqi1cs} "אֶהְיֶה" [ʔehyˈeh] "<be>"
3   Ccoor
4     PP
5       pp {HR} "כְּ" [ˈkᵊ] "<as>"
5       n {HNcmpa} "עֹלְלִים" [ʕōlᵊlˈîm] "<child>"
5       Cattr
6         NegP
7           ng {HTn} "לוֹא" [lō-] "<not>"
6         VP
7           vb {HVqp3cp} "רָאוּ" [rˌāʔû] "<see>"
6         NP
7           n {HNcbsa} "אוֹר" [ʔˈôr] "<light>"
```

4.1 Text and structure as optional concerns

In Text-Fabric we go so far that text and structure themselves are concerns that you can abstract from. Structure sits in the edge features. If you do not load edge features, you do not encounter structure. The textual data is just a column in a big table, and sometimes that is exactly what you want. But the moment you need structure, you can call it in, by loading edge features. Text-Fabric has an API that is sensitive to structure. Going from objects to structurally related objects is one of the supported operations.

4.2 Weaving a weave

Every Text-Fabric data set has two obligatory features: *otype* is a node feature, and provides the node type for each node; *oslots* is an edge feature that contains the complete embedding structure of nodes. Together they are the *warp* of a textual fabric, the fixed threads. They are slots, but not yet filled. They are objects with types and interconnections, but otherwise void. Everything else is comes from the *wefts*, the colorful threads that are woven in. Even the words of the text are a weft, not a warp. A Text-Fabric resource is a *weave* of which the wefts can be distributed separately, to be woven into other weaves. This makes every Text-Fabric resource an extremely versatile player in complicated data combination games.

5. Example B: Comparing BHSA and OSM

We have used Text-Fabric to compare the BHSA with the OSM. The BHSA is already given in Text-Fabric format, the OSM is given in XML. If we look at the OSM, we face two challenges: we have to extract the morphology from amidst all the other information (easy), and we have to connect the morphology bits to the corresponding words in the BHSA (non-trivial). The OSM uses the WLC, the BHSA uses the BHS. They are largely the same, but we have to work around a few alignment problems. The OSM uses textual entities separated by white space, and then divides them into morphemes. The BHSA divides the text in linguistic words, which sometimes attach themselves to the next word, like articles and prepositions. But the BHSA takes pronominal suffixes as part of the word. Hence it is quite challenging to map the OSM morph tags of the morphemes to the correct slots of the BHSA.

But it can be done, it has been done, and it is not too hard to repeat it when the OSM nears its completion. See the BHSAbridgeOSM notebook[1]. The result is two new features, *osm*, linked to the words in the BHSA and *osm_sf*, linked to the suffixes in the BHSA. These features contain the original morphology tags of the OSM. This is top of the *osm* feature. You see the morphology tags for Genesis 1:1 (11 words).

```
@node
@conversion=notebook openscriptures in BHSA repo
@conversion_author=Dirk Roorda
@coreData=BHSA
@coreVersion=2017
@description=primary morphology string according to OpenScriptures
@source=Open Scriptures
@source_url=https://github.com/openscriptures/morphhb
@valueType=str
@writtenBy=Text-Fabric
@dateWritten=2018-01-12T13:21:01Z

HR
HNcfsa
HVqp3ms
HNcmpa
HTo
HTd
HNcmpa
HC
HTo
HTd
HNcbsa

(and 400,000 lines more).
```

Based on these features we can now undertake the comparison of morphology assignment between the BHSA and the OSM.

5.1 Comparison of morphology

How did we compare the BHSA and the OSM? Bit by bit. We started with a language comparison. Do both resources identify the same stretches of Aramaic material? Nearly! See the language notebook[2]. Then we did a first exploration in part-of-speech assignment. At first sight there were 30,000 discrepancies, but it turned out that they were mostly systematic. If we generate lexeme-based exception rules, we could regulate 29,000 cases, leaving only 1000 cases for inspection. See the part-of-speech notebook[3]. We ended up with a much more profound comparison of *categories*. Both the BHSA and the OSM have a system of a base category (part-of-speech) and a refined category within that. If we compare these fully refined categories, we see an awesome lot of agreement, and very few discrepancies per combination of OSM category and BHSA category. Because there are many combinations, there over 600 such cases. They deserve closer inspection.

[1] https://github.com/ETCBC/bridging/blob/master/programs/BHSAbridgeOSM.ipynb
[2] https://github.com/ETCBC/bridging/blob/master/programs/language.ipynb
[3] https://github.com/ETCBC/bridging/blob/master/programs/part-of-speech.ipynb

To give a glimpse of what is going on, here is how the OSM category *verb* relates to BHSA categories.

verb

verb::	(84% = 50691x)
verb:quotation verb:	(10% = 6137x)
verb:copulative verb:	(5% = 3246x)
noun::	(0% = 6x)
adjective::	(0% = 3x)
preposition::	(0% = 1x)
proper noun::	(0% = 1x)

This is an excellent agreement, and possibly, after inspection of only 11 cases a full agreement can be reached.

Here is how OSM *preposition* maps to BHSA labels:

preposition

preposition::	(96% = 50697x)
noun:potential preposition:	(3% = 1643x)
adverb:conjunctive adverb:	(0% = 194x)
interrogative particle::	(0% = 169x)
noun:cardinal:	(0% = 13x)
conjunction::	(0% = 5x)
noun::	(0% = 2x)
proper noun::	(0% = 2x)
article::	(0% = 1x)
verb::	(0% = 1x)

While the general agreement is still very good, there are more discrepancies: 0.76% = 387x out of 51084 cases. For an overview of all combinations and a list of all rare cases, we refer to the category notebook[4].

5.2 Follow up

What can we learn from this comparison? Inspection of the rare cases might reveal a number of glitches on the side of both BHSA and OSM. It could also point to genuine differences of insight in assigning morphological categories to words. It could be that the systems of categories of BHSA and OSM are not fully compatible. It could be that some occurrences in the text are inherently ambiguous,

[4] https://github.com/ETCBC/bridging/blob/master/programs/category.ipynb

morphologically speaking. In all cases, we can now precisely and completely document, quantify and explain the discrepancies, even in a machine actionable way.

6. Conclusions

We have conducted a lot of work with Text-Fabric, more than we addressed in this paper.

The examples we showed are testimony to the Ikea-like pattern of data logistics that Text-Fabric supports.

As data-driven research expands in Biblical research, it is helpful if data producers become aware of the needs of avid data consumers. The best thing they can do is to distribute their data not *only* mixed with the full text, but *also* has a single-concern module. Text-Fabric might help with that.

When open sharing of Biblical resources becomes efficient as well, there is hope that there will be no shortage of Bible software for researchers. (Roorda 2012).

Acknowledgments

To the ETCBC people, especially those who adopted Text-Fabric before I could finish it: Martijn Naaijer, Christiaan Erwich, and Cody Kingham.

To the Open Scriptures people, of whom I only know Jesse Griffin at the moment.

To the Biblical Humanities people, especially Jonathan Robie for sparring.

References

Doedens, Crist-Jan (1994), *Text Databases. One Database Model and Several Retrieval Languages*, number 14 in *Language and Computers*, Editions Rodopi, Amsterdam, Netherlands and Atlanta, USA. ISBN: 90-5183-729-1, http://books.google.nl/books?id=9ggOBRz1dO4C .

Elliger, Karl and Wilhelm Rudolf Rudolph, editors (1997), *Biblia Hebraica Stuttgartensia*, 5th corrected ed., Deutsche Bibelgesellschaft, Stuttgart, Germany. http://www.bibelwissenschaft.de/startseite/wissenschaftliche-bibelausgaben/biblia-hebraica/bhs/

Petersen, Ulrik (2004), Emdros - a text database engine for analyzed or annotated text, *Proceedings of COLING 2004*, p. 1190–1193. http://emdros.org/petersen-emdros-COLING-2004.pdf .

Peursen, Wido Th. and Dirk Roorda (2015), Hebrew text database ETCBC4b. Dataset available online at Data Archiving and Networked services, Den Haag, Netherlands. DOI: http://dx.doi.org/10.17026/dans-z6y-skyh

Roorda, Dirk, Jan Krans, Bert-Jan Lietaert-Peerbolte, Wido Th. van Peursen, Ulrik Sandborg- Petersen, and Eep Talstra (2012), Scientific report of the workshop biblical scholarship and humanities computing: Data types, text, language and interpretation, held at the Lorentz Centre Leiden from 6 feb 2012 through 10 feb 2012, *Technical report*, Lorentz Center, Leiden. http://www.lorentzcenter.nl/lc/web/2012/480/report.php3?wsid=480&venue=Oort

Roorda, D. 2017. The Hebrew Bible as Data: Laboratory - Sharing - Experiences. In: Odijk J. & van Hessen A, *CLARIN in the Low Countries*. London: Ubiquity Press. DOI: https://doi.org/10.5334/bbi.18

Dirk Roorda, & Cody Kingham. (2017, October 31). ETCBC/bhsa: Many fixes. (Version v1.1.0). Zenodo. http://doi.org/10.5281/zenodo.1039470