BRILL

RESEARCH DATA JOURNAL FOR THE HUMANITIES AND SOCIAL SCIENCES (2018) 1-15

BRILL OPEN

brill.com/rdj

Coding the Hebrew Bible

Linguistics and Literature

Dirk Roorda

Data Archiving and Networked Services (DANS), Royal Netherlands Academy of Arts and Sciences (KNAW), The Hague, Netherlands

dirk.roorda@dans.knaw.nl

Abstract

The text of the Hebrew Bible is a subject of ongoing study in disciplines ranging from theology to linguistics to history to computing science. In order to study the text digitally, one has to represent it in bits and bytes, together with related materials. The author has compiled a dataset, called BHSA (Biblia Hebraica Stuttgartensia (Amstelodamensis)), consisting of the textual source of the Hebrew Bible according to the Biblia Hebraica Stuttgartensia (BHS), and annotations by the Eep Talstra Centre for Bible and Computer. This dataset powers the website SHEBANQ and others, and is being used in education and research. The author has developed a Python package, Text-Fabric, to process ancient texts together with annotations. He shows how Text-Fabric can be used to process the BHSA. This includes creating new research data alongside it, and sharing it. Text-Fabric also supports versioning: as versions of the BHSA change over time, and people invest a lot in applications based on the data, measures are needed to prevent the loss of earlier results.

Keywords

Hebrew Bible – corpus linguistics – theology – exegesis – text processing – information retrieval – data science – open science

- Related data set "BHSA" with URL http://doi.org/10.5281/zenodo.1302798 in repository "Zenodo".
- See the showcase of the data in the Exhibit of Datasets: http://dansdata-journal.nl/rdp/dsdoc.html?id=roorda2018

1. Introduction

The Hebrew Bible is an old text, with a period of origin ranging roughly from 1000 BC to 1000 AD, if we also count the long sequence of copying and editing activities leading to the Masoretic Text in the 10th century AD. This final text exists in several witnesses, one of which is the Codex Leningradensis, on which the Biblia Hebraica Stuttgartensia (BHS) text is based.

The Eep Talstra Centre for Bible and Computer (formerly known as Werkgroep Informatica) started pioneering with texts and computers in the 1970s, and essentially never stopped doing so. They harbour a comprehensive database of the text and linguistics of the BHS, which is still growing in sophistication. There have been more efforts to represent the Hebrew Bible in databases, and Bible Software is a commodity by now, but the distinguishing trait of the work of the ETCBC is that it tries hard to be data-driven. Most morphological decisions are based on distributional evidence, not on intuition. A consequence of this is that the terminology in the ETCBC database has idiosyncratic tendencies. It looks linguistic at times, but it does not always conform to the mainstream linguistic conventions.

It is this data set that lies stored in the ETCBC GitHub repository BHSA (BHS Amstelodamensis); even in its incarnation as a data set, the Hebrew Bible has quite a bit of history.

We refer to (Roorda, 2017a) for an historical overview of the digital history and references to prior work, including to the text of the Biblia Hebraica.

2. Problem

The plain text of the BHSA is just under 5MB and it contains 425,000 words. But that is just the start. The text is organized in 1,000,000 linguistic objects, annotated by 33,000,000 values in ca. 100 features. Objects are related by several linguistic relationships.

Looking at the plain text, this is a small corpus, but looking at the size of the annotations, this is no small data set. Earlier versions of the data have been stored in LAF (Linguistic Annotation Framework, an XML representation, which resulted in several GB of XML) (Peursen, 2015). Whereas corpora of living language data tend to be bigger in the size of language utterings, the BHSA, being corpus based on an intensely studied historical text, has a small utterance size but an uncommonly large body of annotations.

In 2013–2014 the ETCBC conducted the CLARIN[1]-funded project SHEBANQ in order to put this all this data online. While the website SHEBANQ[2] offers researchers facilities to interact with the BHSA, the construction process of SHEBANQ required a much deeper level of interaction with the data. In order to reach that level, I wrote a Python library for off-line data processing, LAF-Fabric.[3] Researchers are unlikely to be content with the functions of any particular website, so I thought they deserve the same level of control over their data as I had when developing SHEBANQ. However, LAF-Fabric was needlessly complicated in several respects. Text-Fabric (Roorda and Kingham, 2017) is a minimalistic reinvention of LAF-Fabric with new for template-based search and pretty display.

With the BHSA as data and Text-Fabric as tool, you can do many kinds of research such as:

(1) studying exegetical riddles. A difficult piece of text often exhibits a peculiar syntactic structure. This database has enough information in it to search for such structures. Quite often there are more instances of a phenomenon than come to mind, even after centuries of Bible research. Considering all occurrences of the same syntactic pattern may help to decide whether riddle can be solved linguistically, or whether something else must be the case.
(2) studying the linguistics of Biblical Hebrew. The Hebrew verb exhibits behaviour that is still not completely understood, especially in poetry. In order to get clarity, one needs to collect all relevant verb and clause sequences, so that they can be categorized. This rigorous, explorative data mining for verb behaviour is still in its infancy (Kalkman, 2015).
(3) studying the textual history of the Bible. The Bible is not one book, but consists of many fragments that have been pieced together over the centuries. Language has evolved over that time. Charting the linguistic variation is a key activity for which you need good data processing (Rezetko & Naaijer, 2016).

1 Common Language Resources and Technology Infrastructure, https://www.clarin.eu; the Dutch section has funded SHEBANQ: https://portal.clarin.nl/node/4210.

2 SHEBANQ: System for HEBrew Text: ANnotations for Queries and Markup, https://shebanq.ancient-data.org/hebrew/text.

3 LAF-Fabric, https://github.com/Dans-labs/laf-fabric.

In order for researchers to tackle these questions, they need practical software to process this data. The BHSA is essentially a table with 1.4 million rows and 100 columns. Excel is not the optimal tool for this.

The BHSA is a research database, and it is continually changing in detail, and sometimes even in structure. Even after 40 years, it is still work in progress. If a researcher uses this database for academic work, the evidence should be reproducible for many years, but the data on which the conclusions have been based, might no longer be available in that form.

One of the reasons that Text-Fabric exists, is that commercial Bible Software, despite its qualities, does not sufficiently address the issues that researchers care about: open source, open data, transparency, reproducibility, and a high level of computational control over the data.

3. Methods

The appropriate data model for the BHSA data is a *graph*, to be explained below, in section 3.3. Many people in Digital Humanities are used to working with TEI[4] data, where data is modelled in XML, but this does not play nice with graph models. There is a related conscious effort to get linguistic graph data represented in XML: LAF.[5] In an earlier stage, I have used LAF extensively for this data set, and it worked quite well, but the activities of sharing, storing, adding and combining data turned out to become needlessly complicated.

To cut through all the cruft, I developed Text-Fabric (Roorda & Kingham, 2017) out its precursor LAF-Fabric. It is a data model, a file format, and a tool on the one hand, and a strategy, even an ethos on the other hand.

It is a programmer's interface to the data. This is a significant step, because earlier interfaces to the data were more like query languages,[6] which enable users to retrieve data in certain ways, but detract from the real power of programmatically processing the data.

So how does Text-Fabric hand over power to researchers without requiring them to become software engineers?

4 Text Encoding Initiative, http://www.tei-c.org/index.xml.

5 Linguistic Annotation Framework, https://www.iso.org/standard/37326.html.

6 Nowadays, Text-Fabric does have a query language, but there is no barrier between issuing queries and running your own code.

3.1. *Explorative Programming*

In the software world, Python is one of the friendliest languages for starters in programming. It is not so much a language for writing software as a product as well for writing dedicated computational stories about data sets.

Text-fabric is a Python package, easily installed with pip. In the Python ecosphere, there are many packages for data processing and visualization, and there is the Jupyter[7] notebook for interactive programming. When you work with Text-Fabric, all this is accessible to you. Text-Fabric is just an API, not something that takes control.

The recommended way to start with Text-Fabric is to use it in a Jupyter Notebook, because this gives you many opportunities to explore the data while you program along. This ethos of inspecting the data programmatically can be picked up from the tutorial[8] (Erwich & Kingham, 2017).

4. Data

- **BHSA deposited at Zenodo** – DOI:10.5281/zenodo.1007624
  - ETCBC/BHSA with compact data files as assets, version 1.3 – DOI:10.5281/zenodo.1302798
- **Temporal coverage:** 2011-2017, continuing

The BHSA is a big data set. It contains the Hebrew Bible with linguistic annotations in 5 versions: `3` (2011), `4` (2014), `4b` (2015), `2016`, and `2017`. There is also a continuous version, that is kept up-to-date on a regular basis. This version will change, whereas the others will be kept fixed.

All these versions are data sets that are self-contained, having the full text and all annotations in them.

Every version in the repository comes with the source files, from which the `.tf` files have been derived. The programs responsible for the conversion are also included.

4.1. *TF Data Sets*

Under the `tf` subdirectory, you find the data sets in text-fabric format, for each version a dataset. A text-fabric dataset is a plain directory with a number of feature files, all with extension `.tf`. The name of the file is the name of the feature.

7 Jupyter: doing interactive data science, https://jupyter.org.

8 Tutorial, https://nbviewer.jupyter.org/github/etcbc/bhsa/blob/master/tutorial/start.ipynb.

Every dataset has two or three standard features, with the fixed names `otype`, `oslots`, and `otext`, which contain the essential, abstract information on the slots, nodes, and edges. The other features correspond to the 100 columns mentioned above, when the BHSA was compared to a table with 1.4 million rows and 100 columns.

In the sequel, we use the word feature, not the word column.

4.2. *File Format*

A `.tf` feature file is a plain Unicode text file. You can open them with an ordinary text editor to see the contents. You then see a few lines of metadata at the top, then an empty line, followed by (many) lines of data. Each data line contains the value of the feature for a single node. Normally, the value on line `n` is the value of that feature for node `n`, but text-fabric has a few conventions to optimize[9] long sequences of empty lines away, as well as long sequences with equal values.

Here are the first 11 lines (Genesis 1:1) of a few features.

```
@node
@author=Eep Talstra Centre for Bible and Computer
@dataset=BHSA
@datasetName=Biblia Hebraica Stuttgartensia Amstelodamensis
@email=shebanq@ancient-data.org
@encoders=Constantijn Sikkel (QDF), Ulrik Petersen (MQL) and Dirk Roorda (TF)
@valueType=str
@version=2017
@website=https://shebanq.ancient-data.org
@writtenBy=Text-Fabric
@dateWritten=2017-10-10T10:08:06Z

```

g_word_utf8.tf	phono.tf	sp.tf	lex.tf	
בְּ	bᵊ	prep	B	
רֵאשִׁית	rēš,îṯ	subs	R>CJT/	
בָּרָא	bār'ā	verb	BR>[	
אֱלֹהִים	ʔᵉlōh'îm	subs	>LHJM/	*and*
אֵת	ʔ,ēṯ	prep	>T	*100+*
הַ	ha	art	H	*more*
שָּׁמַיִם	ššām,ayim	subs	CMJM/	*columns*
וְ	wᵊ	conj	W	...
אֵת	ʔ,ēṯ	prep	>T	
הָ	hā	art	H	
אָרֶץ	ʔ'āreṣ	subs	>RY/	

and 400,000+ more in each column ...

FIGURE 1 *Text-Fabric files from the inside.*

9 See the Wiki documentation of text-fabric, https://dans-labs.github.io/text-fabric/Model/Optimizations/.

Note that `phono.tf` is not in the BHSA data set, but is part of another data *module*, which illustrates how naturally data combination comes to text-fabric.

4.3. *Data Model*

We adhere to a minimalistic representation of all data. In Text-Fabric a text is a sequence of slots. Slots are nodes, identified by a number, starting with 1, and each slot corresponds with the position of a word in the text. Slots are not the words themselves, only their positions.

There are more nodes, also numbered, starting after the last slot, and these correspond with sets of slots, such as phrases, clauses, sentences. These nodes are linked to sets of slots, the ones that contain the words that *belong* to phrases, clauses, sentences. This *belonging* is a relationship between nodes. Such relationships we call *edges*. Every relationship between nodes that is relevant to a text, can be coded as a set of edges.

So, the nodes and edges contain the abstract structure of the text: a graph.[10]

It would carry us too far to discuss the reasons for adopting this kind of model for annotated text. The short account is that a graph can express all concepts that matter to text, such as sequence and embedding. This can be done flexibly, without the need to work around limitations of the model, and it also achieves *separation of concerns*, i.e. it is easy to leave out aspects of the data and the model that are not relevant for the analysis at hand.

The model has been introduced by (Doedens, 1994), a Ph.D. thesis that has led to a practical implementation of it (Petersen, 2002), which powers an important part of the data creation workflow at the ETCBC and SHEBANQ.

The concrete matter of a text is added as values of features, such as `g_word_utf8`.[11] Features are mappings from nodes to values.

---

10 Technically, a graph is a set of nodes who are related by edges, https://en.wikipedia.org/wiki/Graph_(discrete_mathematics). In the past, most data is either stored in a relational database, or in a document database. Both are optimized for certain types of relations: tabular and hierarchical, respectively. Graphs have unconstrained relations, and require different techniques to be handled optimally, https://en.wikipedia.org/wiki/Graph_database.

11 `g_word` and `g_word_utf8` contain is the fully-pointed Hebrew text of a word; the one with `_utf8` in the name contains it as a Unicode string, the other one as an ASCII transliteration. Likewise, `g_cons` and `g_cons_utf8` contain the un-pointed, consonantal Hebrew text of a word; `phono` contains a phonological representation of the vocalized Hebrew word.

All textual material and characteristics can be added to the data as features. Some features contain information for slot nodes, others for non-slot nodes, or both. Part-of-speech is a typical slot feature, whereas type-of-constituent typically applies to nodes that are used for clauses or phrases.

In text-fabric, these features are made easily available to the program, by the same names as appear in the dataset. In order to get the `g_word_utf8` feature of a node `n`, you say `F.g_word_utf8.v(n)`. Users might disagree with the names of the features. However, those names are not a choice dictated by Text-Fabric. They are the choice of the corpus designer, in this case the ETCBC. When I wrote Text-Fabric, my objective was that programming theologians recognize the names they are familiar with in the code. The keywords of Text-Fabric itself, such as `F`, have been chosen short and iconic, so that the focus is on the feature name. This stylistic aspect is a bit uncommon in the Python-sphere, so I have also longer synonyms for the members of the Text-Fabric API, such as `Feature for F`.[12]

When researchers get used to this model, they quickly gain mastery over the text.

See for example the "pretty" display in figure 2 of a particular clause that surfaced in a real-life query.[13]

The structure of the textual objects is visible and their nodes show up. The nodes act as barcodes: with a node in hand one can request all available information about that object. Note that phrases and clauses may be chopped up by dashed lines: those chunks are called phrase atoms and clause atoms. Phrases and clauses may be interrupted. When that happens, the interruption occurs always between atoms.

This screenshot is taken from the search tutorial for the BHSA.[14]

As an example of what the coding looks like, we show a piece of code from the tutorial[15] that lists for each book in the Bible the percentage of lexemes that occur exclusively in that book.

12 https://dans-labs.github.io/text-fabric/Api/General/#features.

13 Stephen Ku: Verbless Clauses. Query saved on SHEBANQ: https://shebanq.ancient-data.org/hebrew/query?version=4&id=1314.

14 SearchFromMQL: https://nbviewer.jupyter.org/github/etcbc/bhsa/blob/master/tutorial/searchFromMQL.ipynb#By-Stephen-Ku.

15 Tutorial, https://nbviewer.jupyter.org/github/etcbc/bhsa/blob/master/tutorial/start.ipynb.

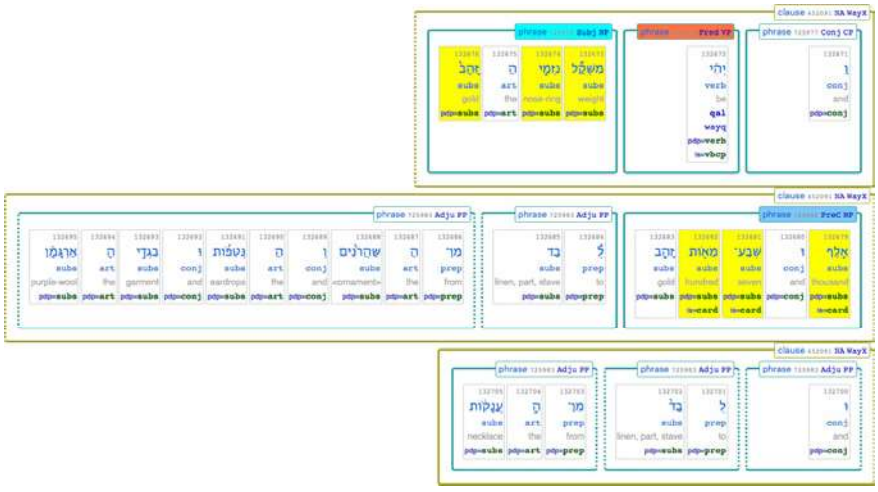

FIGURE 2 *Pretty display of Hebrew text based on BHSA data.*

```
allBook = collections.defaultdict(set)
allLex = set()

for b in F.otype.s('book'):
    for w in L.d(b, 'word'):
        l = L.u(w, 'lex')[0]
        allBook[b].add(l)
        allLex.add(l)

singleBook = collections.defaultdict(lambda:0)
for l in F.otype.s('lex'):
    book = L.u(l, 'book')
    if len(book) == 1:
        singleBook[book[0]] += 1

for b in F.otype.s('book'):
    book = T.bookName(b)
    a = len(allBook[b])
    o = singleBook.get(b, 0)
    p = 100 * o / a
    booklist.append((book, a, o, p))

for x in sorted(booklist, key=lambda e: (-e[3], -e[1], e[0])):
    print('{:<20} {:>4} {:>4} {:>4.1f}%'.format(*x))
```

FIGURE 3 *Code for finding the lexeme base of the books.*

The result can be found directly below the code:

book	#all	#own	%own
Daniel	1121	428	38.2%
1_Chronicles	2015	488	24.2%
Ezra	991	199	20.1%
Joshua	1175	206	17.5%
Esther	472	67	14.2%
Isaiah	2553	350	13.7%
Numbers	1457	197	13.5%
Ezekiel	1718	212	12.3%
Song_of_songs	503	60	11.9%
Job	1717	202	11.8%
Genesis	1817	208	11.4%
Nehemiah	1076	110	10.2%
Psalms	2251	216	9.6%
Leviticus	960	89	9.3%
Judges	1210	99	8.2%
Ecclesiastes	575	46	8.0%
Proverbs	1356	103	7.6%
Jeremiah	1949	147	7.5%
2_Samuel	1304	89	6.8%
1_Samuel	1256	85	6.8%
2_Kings	1266	85	6.7%
Exodus	1425	92	6.5%
1_Kings	1291	81	6.3%
Deuteronomy	1449	80	5.5%
Lamentations	592	31	5.2%
2_Chronicles	1411	67	4.7%
Nahum	357	16	4.5%
Hosea	742	33	4.4%
Ruth	319	14	4.4%
Habakkuk	393	17	4.3%
Amos	652	27	4.1%
Joel	398	14	3.5%
Zechariah	726	25	3.4%
Obadiah	167	5	3.0%
Micah	586	16	2.7%
Zephaniah	367	10	2.7%
Jonah	252	5	2.0%
Haggai	208	3	1.4%
Malachi	314	4	1.3%

FIGURE 4 *Books and their lexemes.*

From this exercise to the research questions above is a huge step, but there are no blocking difficulties to overcome in terms of data processing.

4.4. *Processing*

When text-fabric processes a dataset, it reads the feature files. For efficiency, after reading a feature file, a compact binary representation of the data structure is saved in a cache. The next time a text-fabric run needs this feature, it loads in a fraction of a second.

You can point text-fabric to multiple locations where it searches for `.tf` files. In this way, text-fabric can work with features in several datasets at the same time, without the need to shift that data in place on beforehand. The text-fabric API adapts itself to the files it has encountered in those locations. The phono feature above is an example of this.

The way Text-Fabric conducts data processing has something in common with weaving, although the metaphor may not be immediately obvious. Interested readers may consult the documentation.[16]

4.5. *Agility of Sharing*

There is no overhead in combining features. If you work with a core dataset, such as the BHSA, and you produce new features on your own, you can just distribute your features in a GitHub repository. The BHSA is also in a GitHub repository.

If you need features from different repositories, the only thing needed is to clone both repositories, and point text-fabric to both clones on your computer. You can then immediately run your text-fabric programs.

The fact that the `.tf` files are plain text, and not too big individually, mean that they play very nicely in a GitHub environment, which is a good thing if you really want to move your data around. Currently, the ETCBC has issued 3 data modules next to the BHSA: phono, valence, and parallels (Roorda, 2017).

4.6. *Versioning*

The evolutionary nature of the data, combined with the fact that other systems[17] have been built on it, requires a versioning strategy. For example, SHEBANQ provides permanent access to saved queries based on this data for the academic record. As versions come and go, the version against which a particular saved query has been executed, must be preserved as well.

16 Weaving a fabric: https://dans-labs.github.io/text-fabric/Model/Data-Model/.

17 For an overview of other tools based on the BHSA, see http://etcbc.nl/data/.

Versions arise when the data creation workflow at the ETCBC has run with updated parameters for its algorithms and/or new manual input. Versions may differ in few or many details, and we receive them without an exhaustive list of changes. However, we can *detect* those changes.

Text-Fabric separates the abstract structure of the text from the concrete mass of features, and this turns out to be helpful in processing different versions. We have mapped[18] the nodes from each version to those of the next one. That gives programmers the possibility to get at the data BHSA in version-independent ways, both into the future as into the past. The creation of the node mapping is not completely automatic, but surprisingly little manual work is needed to create a high-quality mapping. In this process, no knowledge is needed from the data creation workflow. When you zoom in to certain features, you can follow the amount of coding that has happened between versions. We also highlighted the changes in a particular feature, function, between 2011 and 2017. For example, function contained the value "Unkn" for more than 40,000 phrases. In 2015 all these had been resolved into better values. Note that there are 250,000 phrases in total.[19]

4.7. *Versatility*

The reader might be tempted to regard Text-Fabric as a tool for processing the Hebrew Bible. The fact is, nothing in Text-Fabric is dependent on the specifics of the BHSA corpus. A proof of this fact is the recent conversion of a corpus of Proto-Cuneiform tablets from the Uruk period (Johnson and Roorda, 2018) into Text-Fabric format. However, both the BHSA and the Uruk corpus make use of additional modules on top of Text-Fabric which do have knowledge of their target corpora.

4.8. *Documentation*

As the paragraph above makes clear, it is vitally important to know what the features mean, and have that information constantly under your fingertips. The documentation is in the repository in Markdown format, which feed the repository's GitHub pages website.[20]

18 Mapping the nodes of between versions, https://github.com/ETCBC/bhsa/blob/master/programs/versionMappings.ipynb.

19 "Phrases of ages", (https://github.com/ETCBC/bhsa/blob/master/programs/versionPhrases.ipynb).

20 Feature documentation, https://etcbc.github.io/bhsa/.

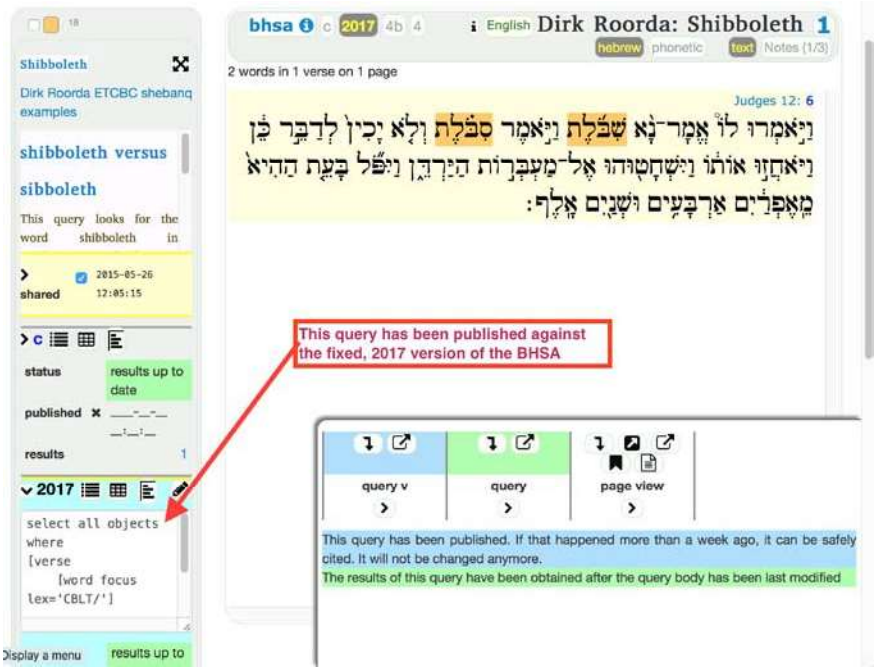

FIGURE 5 *Shibboleth query in SHEBANQ. It is published under the URL https://shebanq.ancient-data.org/hebrew/query?version=2017&id=18.*

5. Concluding Remarks

The combo of BHSA data and Text-Fabric programming provides a powerful way of exploring the text of the Hebrew Bible. Still, it is mainly a programmer's interface. The website SHEBANQ[21] is complementary in that sense. It provides a rich interface for reading the text itself and for querying its linguistic annotations. That way, it is a stepping stone towards a full computational treatment of the materials. Because SHEBANQ is based on the same BHSA, there is a high level of interoperability between queries in SHEBANQ and programming in Text-Fabric.

The BHSA is not just a dataset. It is an ancient, well-researched text, at the heart of a body of work meant to sustain the scholarship around it. It lies embedded in an emerging infrastructure, of a loose nature, that can be maintained by the theologians themselves.

21 SHEBANQ website, https://shebanq.ancient-data.org.

References

All hyperlinks accessed on 2018-06-06.

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

Elliger, Karl & 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/.

Erwich, Christiaan, & Cody Kingham. (2017), *Text-Fabric: what, how and why*, presentation and given at SBL Berlin, https://github.com/ETCBC/sbl_berlin17_textfabric/blob/master/Text_Fabric_Presentation_SBLBERLIN.ipynb.

Johnson, Cale, & Dirk Roorda. (2018, March 7). Nino-cunei/uruk: Conversion verified, tools maturing (Version v1.0.0). Zenodo. http://doi.org/10.5281/zenodo.1193842. Current: https://github.com/Nino-cunei/uruk.

Kalkman, Gino J. (2015), *Verbal Forms in Biblical Hebrew Poetry: Poetical Freedom or Linguistic System?*, PhD thesis, VU University, Amsterdam. https://github.com/ETCBC/Biblical_Hebrew_Analysis.

Petersen, Ulrik. (2002). EMDROS. *Text database engine for analyzed or annotated text*. Open Source software https://emdros.org.

Peursen, Prof. Dr. W.T. van (Eep Talstra Centre for Bible And Computing, VU University Amsterdam); Sikkel, M. Sc. C. (Eep Talstra Centre for Bible And Computing, VU University Amsterdam); Roorda, Dr. D. (Data Archiving and Networked Services, Royal Netherlands Academy of Arts and Sciences) (2015): *Hebrew Text Database ETCBC4b*. DANS. https://doi.org/10.17026/dans-z6y-skyh.

Rezetko, Robert, & Martijn Naaijer (2016), *An Alternative Approach to the Lexicon of Late Biblical Hebrew*, Journal of Hebrew Scriptures, Volume 16, Article 1, http://dx.doi.org/10.5508/jhs.2015.v15.a10 (the DOI mentioned in the article itself is not valid). Data set archived at DANS: https://doi.org/10.17026/dans-256-4hcy.

Roorda, Dirk. (2017, October 10). ETCBC/phono: *Phonetic Transcription of Biblical Hebrew*. Data Repository on GitHub: https://github.com/ETCBC/phono. Archived at Zenodo. http://doi.org/10.5281/zenodo.1007637.

Roorda, Dirk. (2017a). *The Hebrew Bible as Data: Laboratory – Sharing – Experiences*. In: Odijk, J and van Hessen, A. (eds.) *CLARIN in the Low Countries*, Pp. 211–224. London: Ubiquity Press. doi:http://dx.doi.org/10.5334/bbi.18. License: CC-BY 4.0 Preprint on Arxiv: https://arxiv.org/abs/1501.01866 (2015).

Roorda, Dirk, & Janet Dyk. (2017, October 10). ETCBC/valence: *Verbal Valence Patterns in the Hebrew Bible*. Data Repository on GitHub: https://github.com/ETCBC/valence. Archived at Zenodo. http://doi.org/10.5281/zenodo.1007647.

Roorda, Dirk, & Cody Kingham. (2017, March 7). Dans-labs/text-fabric: *Text representation for text with stand-off annotations and linguistic features*. Code repository on GitHub: https://github.com/Dans-labs/text-fabric. Documentation: https://dans-labs.github.io/text-fabric/. Archived at Zenodo. http://doi.org/10.5281/zenodo.375595.

Roorda, Dirk, & Cody Kingham. (2017, October 10). ETCBC/BHSA: *Biblia Hebraica Stuttgartensia* (*Amstelodamensis*). *Versions 2011–2017 in Text-Fabric format*. Data Repository on GitHub: https://github.com/ETCBC/bhsa. Archived at Zenodo. https://doi.org/10.5281/zenodo.1007624.

Roorda, Dirk, & Martijn Naaijer. (2017, October 10). ETCBC/parallels: *Parallel Passages in the Hebrew Bible*. Data Repository on GitHub: https://github.com/ETCBC/parallels. Archived at Zenodo. http://doi.org/10.5281/zenodo.1007643.