Coding in the Hebrew Bible

Dirk Roorda
Data Archiving and Networked Services, https://www.dans.knaw.nl
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.
We have compiled a dataset, called BHSA, consisting of the textual source of the Hebrew Bible according to the Biblia Hebraica Stuttgartensia, and annotations by the Eep Talstra Centre for Bible and Computer. This dataset powers several websites, and is being used in education and research.
We have developed a Python package, Text-Fabric to process ancient texts together with annotations.
We show 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

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 900 AD. This final text exists in several witnesses, one of which is the Codex Leningradensis, on which the Biblia Hebraica Stuttgartensia 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.
It is this data set that lies stored in the ETCBC GitHub repository BHSA (Amstelodamensis); even in its incarnation as a data set, the Hebrew Bible has quite a bit of history.
We refer to (Roorda, 2015) for an historical overview of the digital history and references to prior work, including to the text of the Biblia Hebraica.

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 100 features. Objects are related by several linguistic relationships.

With this mass of data, you can do many kinds of research such as:
1. studying exegetical puzzles. 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.
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).

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.

Methods

The appropriate data model for the BHSA data is a *graph*, to be explained below, in section "Data model". Many people in Digital Humanities are used to working with TEI[1] 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[2]. 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[3]. 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, 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?

Explorative programming

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[4] 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[5], (Erwich & Kingham, 2017)

Data

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.
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.

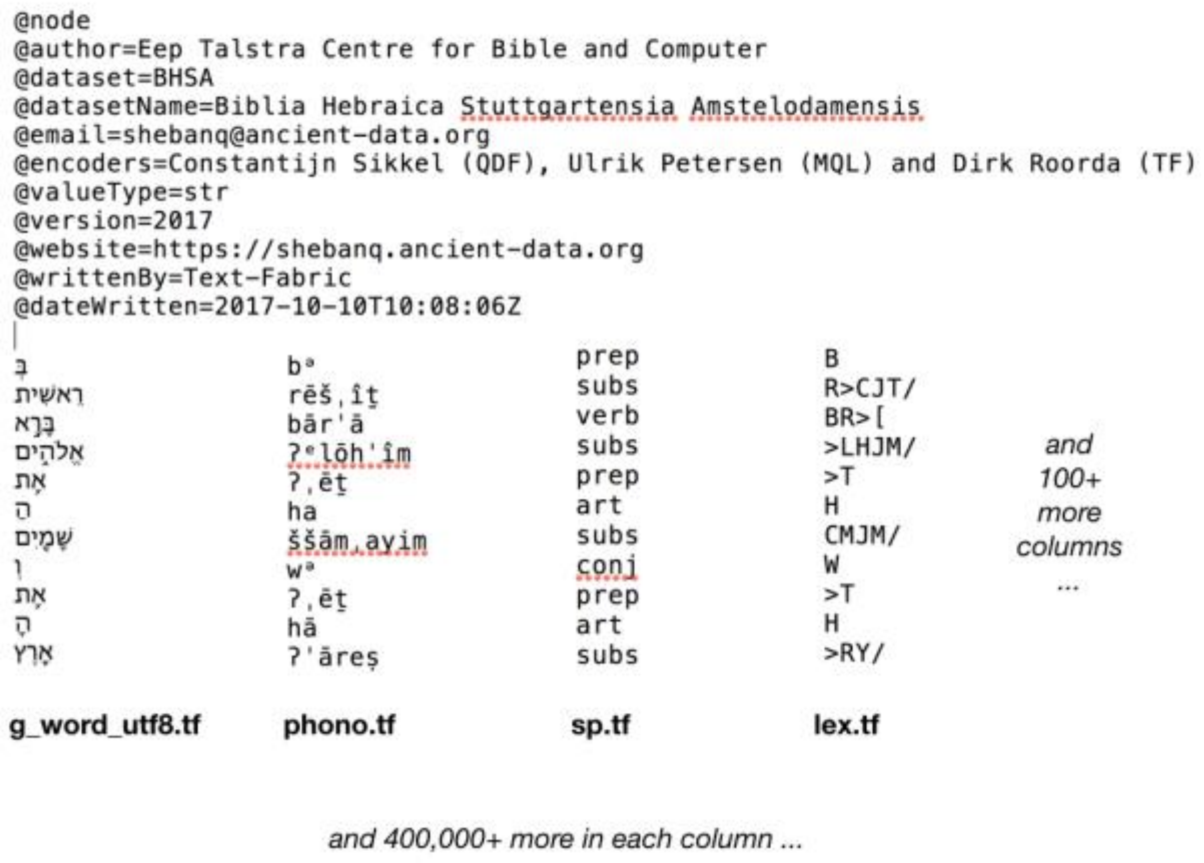
TF data sets

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`. Every dataset as 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.

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[6] 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
|
בְּ              bᵊ              prep      B
רֵאשִׁית          rēš,îṯ          subs      R>CJT/
בָּרָא            bār'ā           verb      BR>[
אֱלֹהִים          ?ᵉlōh'îm        subs      >LHJM/        and
אֵת             ?,ēṯ            prep      >T            100+
הַ              ha              art       H             more
שָּׁמַיִם          ššām,ayim       subs      CMJM/         columns
וְ              wᵊ              conj      W             ...
אֵת             ?,ēṯ            prep      >T
הָ              hā              art       H
אָרֶץ            ?'āreṣ          subs      >RY/

g_word_utf8.tf  phono.tf        sp.tf     lex.tf

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

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.

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[7].

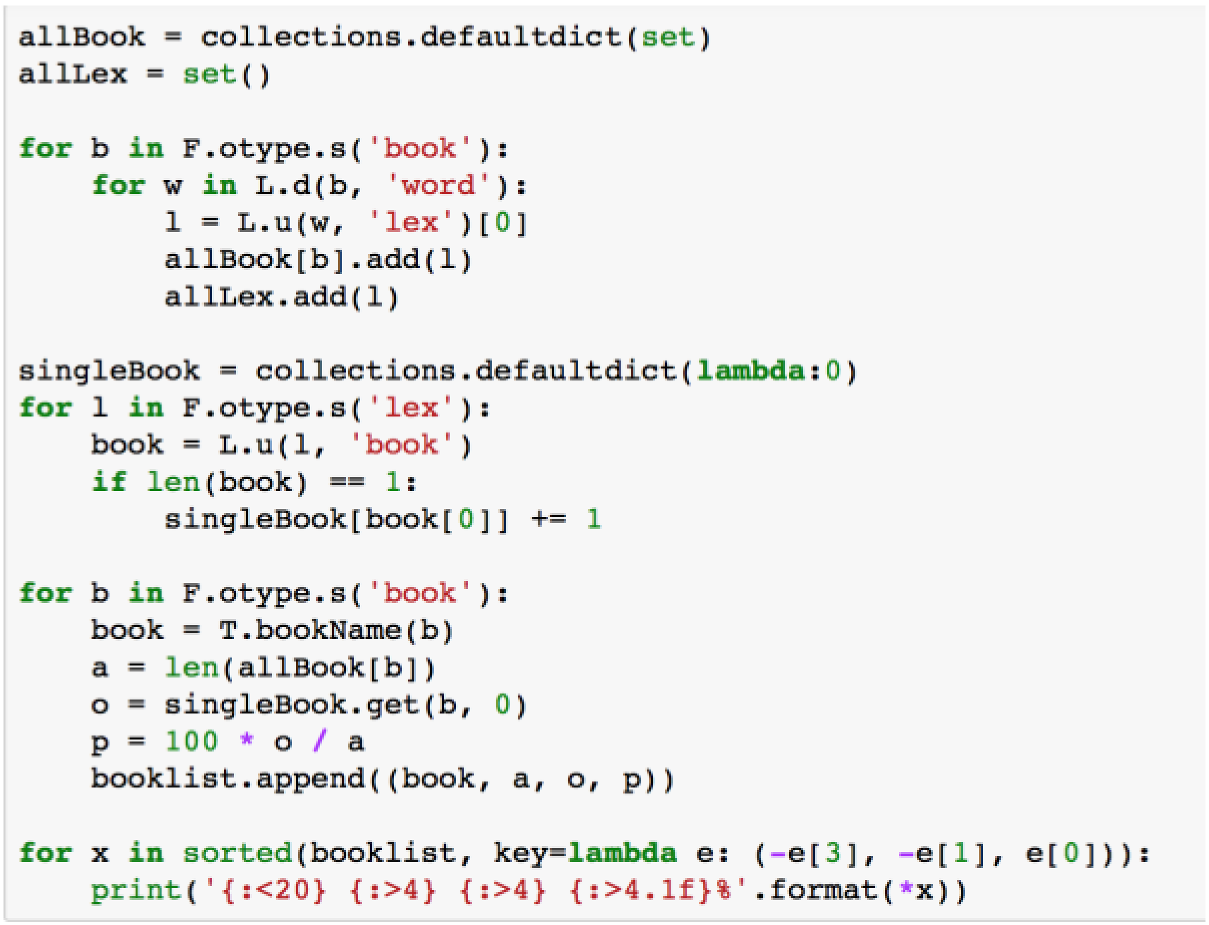
The concrete matter of a text is added as the features, such as `g_word_utf8` and friends. Features are mappings from nodes to values.

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)`.

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

As an example of what the coding looks like, we show a piece of code from the tutorial[8] that lists for each book in the Bible the percentage of lexemes that occur exclusively in that book.



```python
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))
```

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%
```

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.

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.

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).

Versioning

The evolutionary nature of the data, combined with the fact that other systems[9] have been built on it, requires a versioning strategy. 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[10] 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, it 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[11].

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[12].

Concluding remarks

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.

A copy of the authors version of this paper is stored in the BHSA repository.

References

Dirk Roorda, & 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. http://doi:10.5281/zenodo.1007625

Dirk Roorda. (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

Dirk Roorda, & 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

Dirk Roorda, & 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

Dirk Roorda, & Cody Kingham. (2017, March 7). ETCBC/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://github.com/Dans-labs/text-fabric/wiki . Archived at Zenodo. http://doi.org/10.5281/zenodo.375595

Dirk Roorda. (2015, January 8). *The Hebrew Bible as Data: Laboratory - Sharing - Experiences*. Preprint on Arxiv: https://arxiv.org/abs/1501.01866 . To be published as Roorda, D. 2017. 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

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

Kalkman, Gino J. (2015), *Verbal Forms in Biblical Hebrew Poetry: Poetical Freedom or Linguistic System?*, PhD thesis, VU University, Amsterdam. https://shebanq.ancient-data.org/tools?goto=verbsystem .

Robert Rezetko, & 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.2016.v16.a1

Christiaan Erwich, & 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

[1] Text Encoding Initiative, http://www.tei-c.org/index.xml
[2] Linguistic Annotation Framework, https://www.iso.org/standard/37326.html
[3] LAF-Fabric, https://github.com/Dans-labs/laf-fabric
[4] Jupyter: doing interactive data science, http://jupyter.org
[5] Tutorial, https://github.com/ETCBC/bhsa/blob/master/tutorial/start.ipynb
[6] See the Wiki documentation of text-fabric, https://github.com/Dans-labs/text-fabric/wiki/Optimizations
[7] 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 .
[8] Tutorial, https://github.com/ETCBC/bhsa/blob/master/tutorial/start.ipynb
[9] For an overview of other tools based on the BHSA, see the home page of SHEBANQ, https://shebanq.ancient-data.org .
[10] Mapping the nodes of between versions, https://github.com/ETCBC/bhsa/blob/master/programs/versionMappings.ipynb
[11] "Phrases of ages", (https://github.com/ETCBC/bhsa/blob/master/programs/versionPhrases.ipynb)

[12] Feature documentation, https://etcbc.github.io/bhsa/