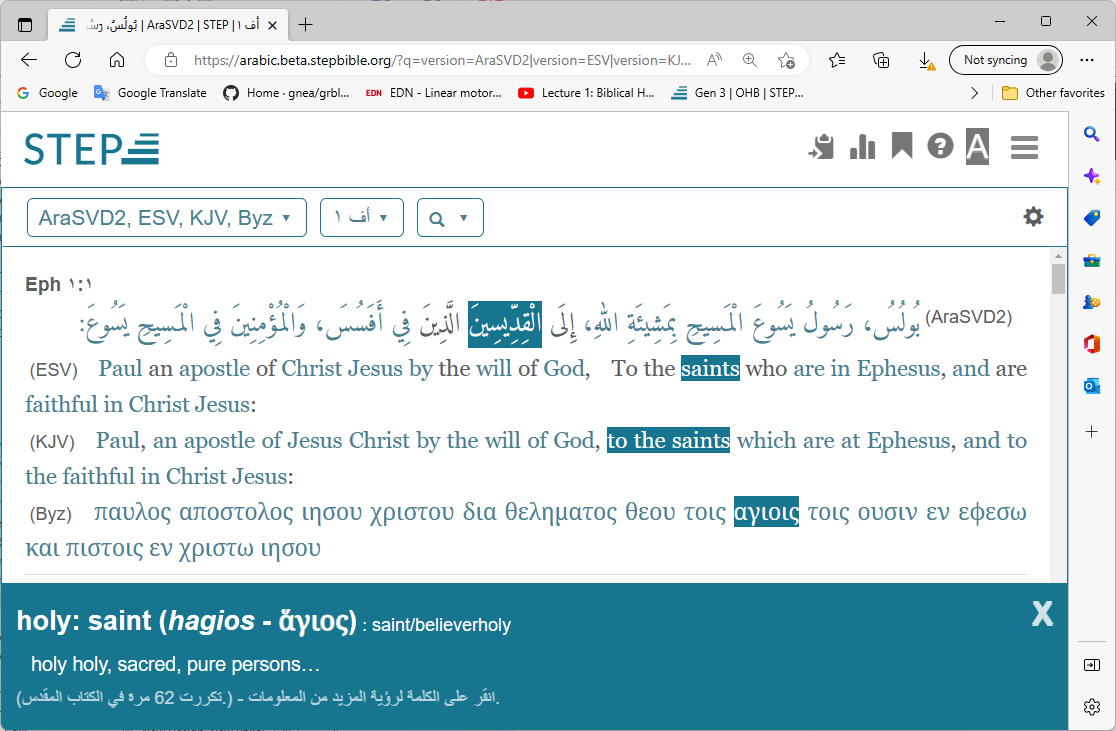
**Tagging Bible Text**

# Overview

This project was started after I used [STEPBible](https://www.stepbible.org/) and recognized the great benefit of being able to hover over or click on Bible words, for specific versions, to get a wealth of information about them. It is a great study tool. The secret behind this capability, besides the clever programming driving the web site, is that the words of those versions are tagged (associated with) coded numbers to tie them to certain lexicons and other resources. The simplest tagging is using [Strong’s numbers](https://en.wikipedia.org/wiki/Strong%27s_Concordance#:~:text=in%20the%20Bible.-,Strong%27s%20numbers,as%20the%20%22Strong%27s%20numbers%22.).

The version of Bible that I use, the Arabic Smith & Van Dyck version, did not have such tags and I wondered, what does it take to tag the Arabic Bible? I wrote a small program to help me tag the Arabic words based on King James Version tagging. If I continued in this direction, it would’ve taken me a few years to complete the task. This is when Rev Dr. David Instone-Brewer told me about his process of Automatically tagging a Bible of virtually any language. I then used his process which successfully tagged close to 80% of the Arabic Bible. The result is shown in the screenshot below:



In the above screen shot, the Arabic text of Eph1:1 is displayed along with the same verse from two English translations and an original language (Greek) version. By Hovering the mouse pointer over the word “الْقِدِّيسِينَ”, it automatically highlights the equivalent word in both English translations, and in the Greek. It also shows the word meaning at the bottom. Clicking on the same word, brings up even more details, but not shown here because it is not the purpose of the screen shot.

At this point, I started using the program that I originally developed, to review and complete the tagging. This document describes the process used, and associated programs.

# Acknowledgment

This project wouldn’t have been possible without the help, support and guidance of Rev Dr. David Instone-Brewer. He provided me with the necessary material and patiently answered all my questions.

# Process Summary

**The Aligner**:

The aligner is a piece of software that is meant for use by language translation programs. The idea is to feed the aligner with text A in a specific language, and its accurate translation in a different language as text B. It then generates a mapping associating each word in text A with the corresponding word in the translation B.  The process is associated with some probabilities, so the mapping is not 100% accurate. The longer the text, the higher the accuracy.

For example if we feed the aligner with the text of the book of Genesis in English, and the book of Genesis in Farsi, it will produce a mapping associating each English word in Genesis, with the corresponding word in Farsi. Some times the program is not able to determine the association, so, it provides no mapping.  Words with higher frequency will be mapped more accurately.

The accuracy will increase if we feed the aligner with the whole Old Testament. Or even better, the whole Bible.

**Tagging**:

We use this ability of the Aligner to tag the Bible text, of virtually any language,  with Strong’s numbers, with varying degrees of success. The basic idea is that text A in this case will be one of the Testaments of the Bible in the target language. Text B will be the same Bible Testament expressed entirely in Strong’s numbers. For the aligner, Strong’s numbers will appear to be as just words of a translation language. To illustrate, here is the first few verses of Genesis in Strong’s numbers.

|  |
| --- |
| Gen 1:1 7225 1254 0430 8064 0776  Gen 1:2 0776 1961 8414 0922 2822 5921 6440 8415 7307 0430 7363 5921 6440 4325  Gen 1:3 0559 0430 1961 0216 1961 0216  Gen 1:4 7200 0430 0216 3588 2896 0914 0430 0996 0216 0996 2822  Gen 1:5 7121 0430 0216 3117 2822 7121 3915 1961 6153 1961 1242 3117 0259  Gen 1:6 0559 0430 1961 7549 8432 4325 1961 0914 0996 4325 4325  Gen 1:7 6213 0430 7549 0914 0996 4325 0834 8478 7549 0996 4325 0834 5921 7549 1961 3651  Gen 1:8 7121 0430 7549 8064 1961 6153 1961 1242 3117 8145  Gen 1:9 0559 0430 6960 4325 8478 8064 0413 4725 0259 7200 3004 1961 3651 |

The aligner then, generate a mapping associating each word in text of the target language with the corresponding Strong’s number. The mapping is then used to generate the tagged text.

**Variances**:

The Bible written in Strong’s numbers doesn’t have a standard version. For example, let us use a tagged KJV version and extract from it Strong’s numbers, then write them in the format above. If we do the same, using ESV, the result will be somewhat different from that of KJV. Therefore, the result of this automated tagging process depends to a great degree on the source of Strong’s numbers.

For my work, I use the Tyndale OT Hebrew Tagged text, and the Translators Amalgamated Greek NT to generate the Strong’s versions of the Old and New Testament. They are found [here](https://github.com/STEPBible/step) under [Creative Commons License](https://creativecommons.org/licenses/by-nc/3.0/deed.en_GB)

# Process Flow Diagram



# Process

The process described below uses the Arabic Bible as an example. The process has been used also to tag Zokam (Burmese) Bible.

1. Diacritics are grammatical tool to ensure the meaning is conveyed correctly. For example using different diacritics can change the word from being a subject of a verb to being an object without changing the position of the word in the sentence. They cause a lot of noise in the text which can hinder the efficient use of Partext Interlineariser and the Berkley aligner. So, the first step is to remove all diacritics and punctuations from the text. This is done using a small c# program: CleanArabicText.

Example:

|  |
| --- |
| Original text:  Gen 1:1 فِي الْبَدْءِ خَلَقَ اللهُ السَّمَاوَاتِ وَالأَرْضَ.  After running the program  Gen 1:1 في البدء خلق الله السماوات والارض |

1. Next, we need to generate the morphology file from Partext Interlineariser. This will help us identify the stems of the Arabic words. Paratext requires the files to be in USFM (Unified Standard Format Markers) format, therefore we use a small c# program: Convert2USFM. to convert the cleaned Arabic text into USFM format Bible books. The first section of Genesis USFM file looks like so:

|  |
| --- |
| \id GEN Smith & Van Dyck Arabic translation  \ide UTF-8  \h تكوين  \p  \c 1  \p  \v 1 في البدء خلق الله السماوات والارض |

1. In Paratext, we create a new project (one project per Testament) and add to it all the Bible books from step 2. Then, start the Inerlineariser. This will automatically generate a morphology xml file.
2. Now, we can use the morphology file from step 3, and the clean text from step 1, and use a small c# program: StemmingArabicText to reduce the Arabic words to their stems.

|  |
| --- |
| Noe the above verse becomes  Gen 1:1 في البدء خلق الله سماوات الارض |

1. We need to Generate separate file containing Strong’s numbers for replacing each corresponding Hebrew word. We call this the Tags file, since it will ultimately be the tags in the text. Use a small c# program: GenerateHebrewAndTags & GenerateGreekAndTags.

|  |
| --- |
| The corresponding Tag verse to Gen 1:1  Gen 1:1 7225 1254 0430 8064 0776 |

1. Use the program: GenerateAlignerFiles to mainly put a version of the stemmed Arabic and the tags file in the Berkley Aligner train folder, after removing the references from the beginning of each verse.

|  |
| --- |
| So, the above becomes:  في البدء خلق الله سماوات الأرض  and  7225 1254 0430 8064 0776 |

1. Run the Aligner to generate the h-a-align file and g-a-align.

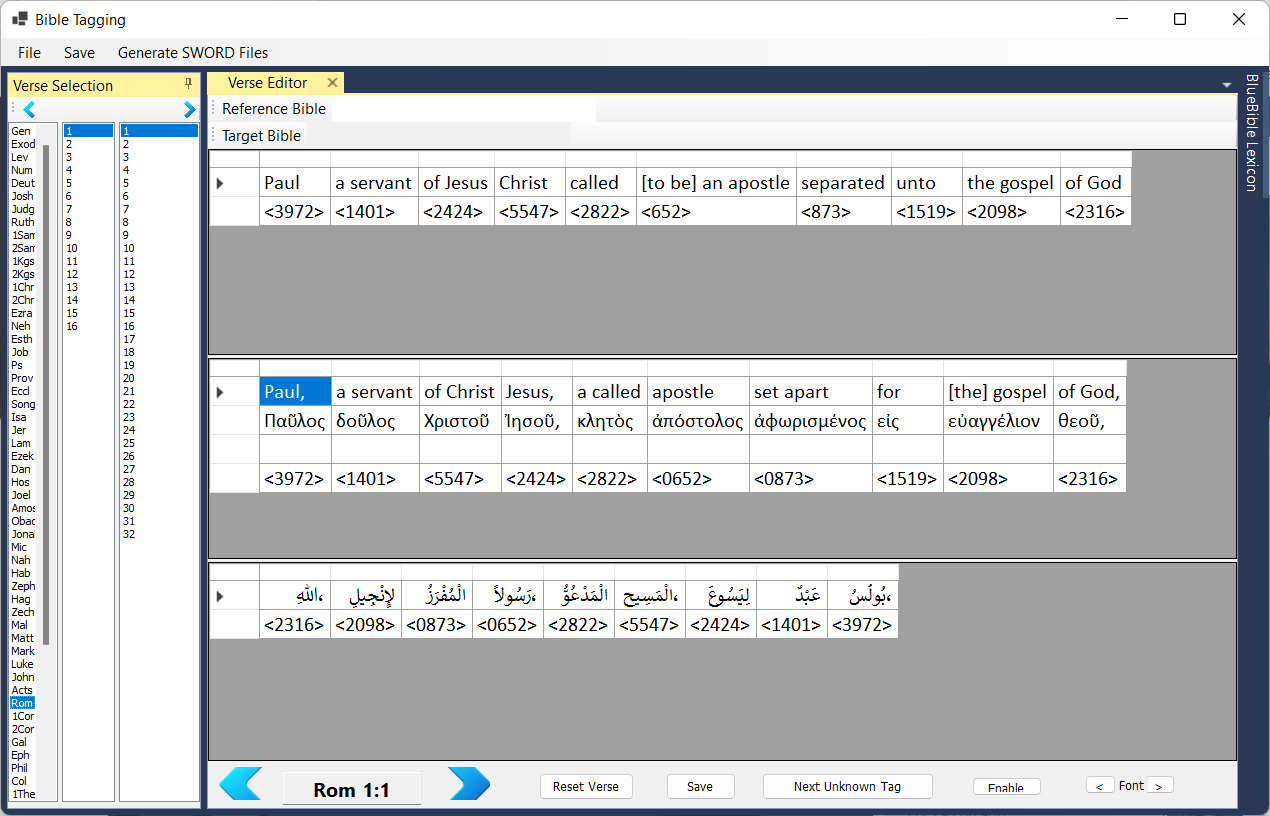
java -server -mx1000m -cp berkeleyaligner.jar edu.berkeley.nlp.wordAlignment.Main ++myConfs/arabicot.conf

java -server -mx1000m -cp berkeleyaligner.jar edu.berkeley.nlp.wordAlignment.Main ++myConfs/arabicnt.conf

1. Use the AlignerMapping program to generate the Arabic tagged text.

|  |
| --- |
| The aligner mapping for G1.1 will look like this  1-0 5-4 4-3 3-2 2-1  In each pair of numbers, the first represents the Arabic word number and the second represents corresponding tag word number. The first word in the line is word 0. Notice that Arabic is displayed right to left, so, the rightmost word is word 0.  Arabic line Hebrew line  0 في Maps to nothing  1 البدء Maps to 0 7225  2 خلق Maps to 1 1254  3 الله Maps to 2 0430  4 سماوات Maps to 3 8064  5 الأرض Maps to 4 0776 |

1. The Bible Tagging Utility can be used to review each verse and make correction as necessary.

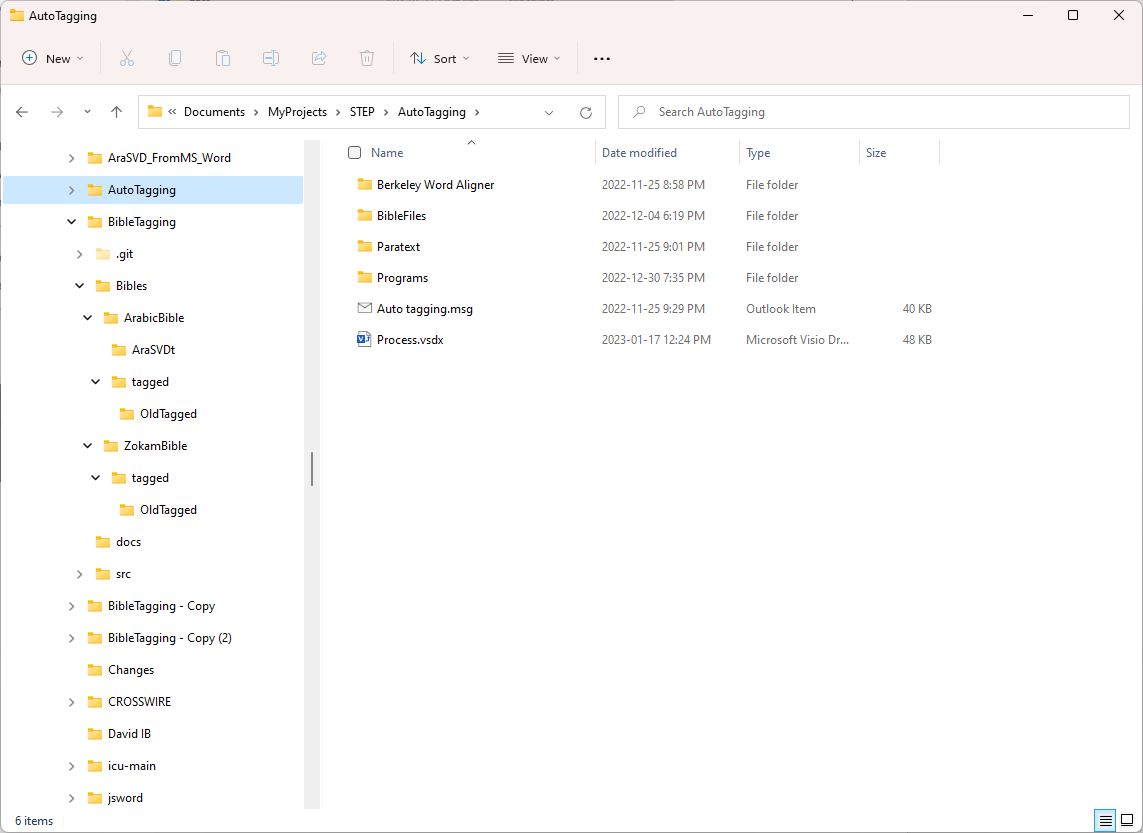


The Bible Tagging Utility has a “Generate SWORD Files” menu item to achieve points 10 and 11 described below

1. In order to use Sword module generator, we need to covert the Arabic text to OSIS format using the GenerateOSIS program.
2. Finally use osis2mod to generate the Arabic module.

# Directory Structure

Bible Tagging is publicly available under BSD 3-Clause "New" or "Revised" License at this [repository](https://github.com/sabdelmalik/BibleTagging).



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| .git |  |  |  | Special folder used by github. Not stored in the repository. |
| Bibles |  |  |  | Contains the tagged Bibles. In this example it contains an Arabic and a Zokam Bibles.  For creating SWORD modules, this folder needs to also contain the following files:  osis2mod.exe, icudt51.dll, icuin51.dll, icuuc51.dll, libsword.dll  Which can be obtained from [Osis2mod - CrossWire Bible Society](https://wiki.crosswire.org/Osis2mod) |
|  | ArabicBible |  |  | Main folder for Arabic Bible tagging. It contains:   1. Arabic Bible Configuration File. 2. Reference tagging Bibles (KJV, THOT and THNT. (not on github) 3. tagged folder as described below. |
|  |  | AraSVDt |  | This folder gets created only when the “Generate SWORD Module” menu is invoked in the Tagging Utility. |
|  |  | tagged |  | Initially this folder contains the text generated in step 8 above. That is the original Bible with the auto generated tags. As the tagged text is modified through the utility, this file gets backed up to the OldTagged folder, and a new file with the updates takes its place. The updated file name is the same as the original file name plus a timestamp appended to it, |
|  |  |  | OldTagged | Contains old versions of the tagged file |
|  | ZokamBible |  |  | See ArabicBible for details. |
|  |  | tagged |  | See ArabicBible for details. |
|  |  |  | OldTagged | See ArabicBible for details. |
| docs |  |  |  | Currently contains this document. |
| src |  |  |  | Contains the source code for all the programs used in the process written in c# (It does not contain Berkely Alignment or Paratext) |

Bible configuration file

For the Bible Tagging utility to work, each Bible folder must contain “BiblesConfig.txt” file. As an example, following is the config file for the Arabic Bible.

|  |
| --- |
| [Tagging]  untaggedBible=AraSVD.txt  taggedBible=AraSVD.txt  targetTextDirection=rtl  kjv=kjv+codes.txt  hebrewReferences=HebOT+GkAdds+dStrongs-2022f.txt  greekReferences=TAGNT Mat-Jhn - Translators Amalgamated Greek NT - STEPBible.org CC-BY.txt,TAGNT Act-Rev - Translators Amalgamated Greek NT - STEPBible.org CC-BY.txt  [OSIS]  osisIDWork=AraSVDt  osisRefWork=bible  language=ar  language-type=IETF  title=Smith Van Dyck Arabic Bible (Tagged)  contributor-role=ctb  contributor-name=Sami Abdel Malik  type=Bible  identifier=AraSVD-Tagged  description=This work adds Strong's references to Smith Van Dyck Arabic Bible.  rights=Arabic Bible Outreach Ministry  refSystem=Bible  ot-vpl-file=ot\_tagged\_text.txt  nt-vpl-file=nt\_tagged\_text.txt  output-file=AraSVDt.xml |