A Transformer-based Parser for Syriac Morphology

Martijn Naaijer♡[1] Constantijn Sikkel♠ Mathias Coeckelbergs♣
Jisk Attema♢ Willem Th. Van Peursen♠
♡University of Copenhagen, Denmark ♠Vrije Universiteit Amsterdam, The Netherlands
♣Katholieke Universiteit Leuven, Belgium ♢Netherlands eScience Center, The Netherlands

Abstract

In this project we train a Transformer-based model from scratch, with the goal of parsing the morphology of Ancient Syriac texts as accurately as possible. Syriac is a low-resource language, only a relatively small training set was available. Therefore, the training set was expanded by adding Biblical Hebrew data to it. Five different experiments were done: the model was trained on Syriac data only, it was trained with mixed Syriac and (un)vocalized Hebrew data, and it was trained first on (un)vocalized Hebrew data and then trained further on Syriac data. The models trained on Hebrew and Syriac data consistently outperform the models trained on Syriac data only. This shows that the differences between Syriac and Hebrew are small enough that it is worth adding Hebrew data to train the model for parsing Syriac morphology. Training models with data from multiple languages is an important trend in NLP, we show that this works well for relatively small datasets of Syriac and Hebrew.

1 Introduction

In this paper we develop a morphological parser for the Syriac language. The trained model is able to segment graphical units into distinct words, it segments the morphemes within a word, and disambiguates morphemes and lexemes, all at the same time.

Syriac is a Semitic language with a rich morphology. Therefore, to add linguistic annotations to a text, it is better to encode the smaller parts of a word (morphemes) rather than the complete words. A complication is that the Syriac language is written without vowels, which leads to the problem that a word can be parsed in different ways. Furthermore, we only have a small Syriac training set. Therefore, we try to improve the model's prediction accuracy by adding Biblical Hebrew data to the training process. Biblical Hebrew is a Semitic language that is closely related to Syriac, and the training set that we have for this language is substantially bigger.

Since the late 1970s, the Eep Talstra Center for Bible and Computer (ETCBC) of the Vrije Universiteit Amsterdam has developed and maintained a richly annotated dataset of the Masoretic Text of the Hebrew Bible. This dataset contains a wealth of linguistic features on the levels of words, phrases, clauses and larger text units. More recently, ancient texts in Syriac have been prepared in a similar way. However, a vast corpus of Syriac texts is available, and we hope to develop a faster approach to annotate these texts, because annotating them manually is a labor-intensive task.

We have trained the Transformer model in five different ways, to see which approach gives the highest accuracy on the Syriac test set: a model trained on Syriac data only, a model trained on a mix of (vocalized or unvocalized) Hebrew and Syriac data, and a model which is trained on (vocalized or unvocalized) Hebrew data first and trained further on Syriac data.

A trained model can make predictions on "new" Syriac texts, resulting in morphologically segmented texts. These results need to be corrected manually, and these corrected results can be processed further in a rule-driven way to produce the linguistic annotations. Therefore, training the models is the first step in a longer pipeline.

2 State of the art

Between 2000 and 2020 a number of studies were published in which Natural Language Processing (NLP) tasks for Semitic languages are described, often dealing with part of speech tagging (e.g.,

[1] Corresponding author: mna@teol.ku.dk.

Proceedings of the Ancient Language Processing Workshop associated with RANLP-2023, pages 23–29,
held in Varna Bulgaria, Sept 8, 2023.
https://doi.org/10.26615/978-954-452-087-8_003

Modern Hebrew: Bar Haim et al. 2008; Amharic: Tachbelle et al. 2011; Arabic: Kübler, and Mohamed 2012; Mishnaic Hebrew: Giovanetti et al. 2018). Other studies deal with morphological analysis (Daya et al. 2004, Lembersky et al. 2014) and segmentation (Zeldes 2018).

With the larger availability of digital (annotated) Semitic texts and the advent of large, Transformer-based language models, there is an acceleration in the development of models and tools for NLP tasks for Semitic languages. A Large Language Model which focuses on Modern Hebrew, is AlephBERT (Seker et al. 2021), which can be used for a number of tasks, including segmentation, part of speech tagging, full morphological tagging, named-entity recognition and sentiment analysis. A similar model for Arabic, AraBERT, was developed by Antoun, Bali and Hajj (2021).

Relatively close to our research is a paper on adding diacritics to consonantal Hebrew texts (Shmidman et al. 2020). It uses a combination of a machine learning ("several bi-LSTM based modules") and a rule-driven approach ("comprehensive inflection tables and lexicons"). Koppel and Shmidman (2020) give an overview of developments in Machine Learning in relation to the Hebrew language and its texts.

A list of NLP resources for Hebrew can be found here: https://github.com/NNLP-IL/Resources.

An important trend in NLP is the development of multilingual models. These are models that can be used for a number of NLP tasks in various languages. Some of these models are trained on one language, like English, and they can be trained further on other languages, but there are also models that are trained from scratch on a number of languages (Ruder 2020).

3 Data

Our dataset consists of five files[2], which are based on the ETCBC database. The Hebrew files that can serve as the input data for the model, contain vocalized or unvocalized text of the Masoretic Text (MT) of the Hebrew Bible. The Hebrew output file contains the morphologically parsed MT. The text of these datasets is based on the fifth edition of the Biblica Hebraica Stuttgartensia[3]. The Syriac input file contains some books from the Peshitta, a translation of the Hebrew Bible in Syriac[4] (Ter Haar Romeny and Van Peursen, 1966–) and some non-biblical texts[5]. The Syriac input texts are unvocalized, but they contain some diacritics, which can be found in the Syriac manuscripts.

Each line in a data file contains one verse, and the text is represented in the ETCBC transcription. The first line of the vocalized Hebrew dataset, which is the first sentence of the Hebrew Bible, looks as follows:

```
Gen 1 1 B.:R;>CIJT B.@R@> >:ELOHIJM >;T HAC.@MAJIM W:>;T H@>@REY
```

This line contains four tab-separated fields, with the following data: book, chapter, verse, and text.

In Hebrew script, the text, which means "In the beginning God created the heaven and the earth", looks as follows:

בְּרֵאשִׁית בָּרָא אֱלֹהִים אֵת הַשָּׁמַיִם וְאֵת הָאָרֶץ

All consonants, vowel signs and diacritics have a value in the transcription, e.g., ב is transcribed with B, א with >, qametz is transcribed with @, shewa with ":", and dagesh with ".". The transcription is read from left to right, unlike the text in Hebrew script.

The same line, but taken from the unvocalized dataset looks as follows:

```
Gen 1 1 BR>CJT BR> >LHJM >T HCMJM W>T H>RY
```

This text contains the same consonants as the vocalized text, but it misses the vowel signs.
Finally, the corresponding verse in the morphologically parsed output file looks as follows:

[2] The files can be found in the data folder of our GitHub repository: https://github.com/etcbc/ssi_morphology. The raw input files are s2-in (Syriac), t-in_voc (vocalized Hebrew), t-in_con (unvocalized Hebrew), the corresponding parsed output files are s2-out (Syriac) and t-out (Hebrew). In this repository one can also find the code.

[3] For an electronic edition of the MT with all the annotations, see: https://github.com/ETCBC/bhsa.

[4] A digitized version of the whole Peshitta can be found here: https://github.com/ETCBC/peshitta.

[5] For the texts, see also: https://github.com/ETCBC/linksyr/tree/master/data.

Gen 1 1 `B-R>CJT/ BR>[ >LH(J(M/JM >T H-CMJ(M/(JM W->T H->RY/:a`

The output dataset contains the same consonantal text as the input data, with a number of extra signs which indicate the morphological structure of the words:
The dash (-) separates different words within a graphical unit.
A word can have different morphemes, which are marked with special signs:
After "[" follow verbal endings, and after "/" follow nominal endings.
"+" initializes a pronominal suffix.
Between exclamation marks, one finds the verbal preformative, e.g., !J! in a 3rd person masculine singular yiqtol, !T! in a 2nd person masculine singular yiqtol or !! in a qal infinitive or imperative.
Between closing square brackets one finds the prefix that is characteristic for a verbal stem, e.g.]HT] for hitpael,]N] for niphal, etc.
"~" initializes a univalent final, for example, a ~H is a locative he.

The ETCBC approach of encoding morphology distinguishes between a paradigmatic form and a realized form of the morphemes. E.g., the paradigmatic form of the masculine plural marker is JM (ים- in Hebrew script). In several places in the MT, it is spelled as M (ם). Here the J (י), which is part of the paradigmatic form, is not written. This is indicated in the encoding with an opening parenthesis. E.g., in Genesis 17:20, one finds נשיאם ("princes"), which has the morphological encoding NFJ>/(JM, indicating that the J occurs in the paradigmatic plural form, but it is not realized. The opposite can also occur. If a character occurs in the text, but not in the paradigmatic form, it is preceded by "&".

In the morphological encoding, there are some Latin letters preceded by a colon:
:a marks that a word is in absolute state.
:c marks that a word is in construct state.
:n marks the narrative vocalization of the waw.
:d marks the D-stem.
:u marks the u-a pattern of the passive.

The "=" sign is used to disambiguate consonantal homographs, e.g., one distinguishes between KBD/ (כָּבֵד, "heavy"), KBD=/ (כָּבֵד, "liver"), and KBD==/ (כֹּבֶד, "heaviness").

The alphabets of Syriac and Hebrew are identical, also in the ETCBC transcription, except that the *sin* (ש) is lacking in Syriac. The Syriac dataset contains three different Syriac diacritics: dots below and above the text, and seyame.

A limitation of the present dataset is that for every word in the input, there is only one correct parsing in the output. In some cases, the text is ambiguous, and a word could be parsed correctly in different ways. A possible improvement of the dataset is to include alternative parsing options.

4 Data preparation

We start with texts that do not have any parsing, which means that a text has not been segmented in phrases, clauses, or sentences. All verses of a book in the dataset are concatenated and split separately in shorter sequences of n graphical units. n is one of the required hyperparameters for training a model. These shorter sequences are partly overlapping and form a moving window. E.g., if the text is:

```
BR>CJT BR> >LHJM >T HCMJM W>T H>RY
```

and n is 5, the text will be split in the following three training inputs:

```
BR>CJT BR> >LHJM >T HCMJM
BR> >LHJM >T HCMJM W>T
>LHJM >T HCMJM W>T H>RY
```

When all the texts are split in partly overlapping sequences and a subset is selected randomly as Syriac test set, a problem is that part of the sequences in the test set can also be found in the training set, which means that training and test set are not independent of each other. A possible solution is to select a few complete books as test set, but that leads to the problem that the language of these books may not be representative of Syriac in general. Therefore, we have used a different solution. If n is 5, the texts of 5 consecutive verses are grouped, and from all these groups of 5 verses, the validation and test set are selected. With this approach, it is guaranteed that the texts are long enough to extract at least one sequence of 5 graphical units, and they are short enough to split a book in many sequences, with the result that parts of the book can be found in the training, validation and test set, without overlap between these

datasets. After this split, each sequence of 5 verses is split further in the partly overlapping shorter sequences of 5 graphical units. All short sequences that contain a case of ketiv/qere in the Hebrew datasets are removed, because the consonantal text that is written (the ketiv) and the morphological analysis generally do not match. These words are indicated with a "*" in the data files.

5 The model

The morphological analysis is approached here as a sequence to sequence (seq2seq) problem, for which we use a Transformer model [6]. The Transformer is the state-of-the-art model for numerous NLP tasks (Vaswani et al. 2017) and is also the basis of Large Language Models like ChatGPT and GPT4. The Transformer seq2seq model has an encoder/decoder architecture. The encoder consists of a stack of encoder layers, in which the output of one layer serves as the input of the next one. Each layer consists of two components: multi-head attention and a feedforward network. Fundamental for the transformer model is the concept of self-attention, with which a word is related to all other words in a text sequence. In the self-attention mechanism, the embedding matrix of a sentence is multiplied with three randomly initialized matrices W^Q, W^K, and W^V, thus forming three new matrices Q (Query), K (Key) and V (Value). From these matrices, the attention matrix Z_1 is calculated as follows:

$$Z_1 = softmax\left(\frac{QK^T}{\sqrt{(d^k)}}\right)V_1$$

Z_1 has the index 1, because this is the first attention head. There can be an arbitrary number of heads that are concatenated:

$$Multihead\ attention = Concatenate(Z_1, Z_2, Z_3, \dots)W_0$$

in which W_0 is a new weight matrix.

After that, information of the word order in a sentence is added using positional encoding. The resulting matrix is fed to a feedforward network consisting of two dense layers with ReLU activation.

Just like the encoder, the decoder consists of a number of layers, one layer giving its output to the next one.

The decoder of the transformer model starts with a start symbol and the representation of the sentence produced by the encoder, and from that the first word of the output after the start symbol is generated. Then, the representation, the start symbol and the first word together are fed to the encoder, after which the second word is generated. This is done until a stop symbol is generated.

In the present implementation, various hyperparameters can be tweaked, which can be found in the README of the GitHub repo. The only thing that we vary in the experiments described here are the number of epochs and the training datasets.

The model is trained from scratch, which makes it possible to get a good impression of what the difference is between a model trained on Syriac data alone, and a model that is trained on Hebrew and Syriac data.

In all our experiments, the number of heads in the encoder is 8, and the number of encoder layers and decoder layers is 3. The feedforward hidden dimension is 512. During decoding we used beam search, with a beam size of 3. The length of the partly overlapping text sequences is 7 graphical units.

6 Results

The model was trained with five different training strategies:

1. The model was trained on Syriac data.
2. The model was trained on a mix of unvocalized Hebrew and Syriac data.
3. The model was trained on a mix of vocalized Hebrew and Syriac data.
4. The model was trained first on unvocalized Hebrew data (10 epochs), and after that trained further on Syriac data.
5. The model was trained first on vocalized Hebrew data (10 epochs), and trained further on Syriac data.

The approach of two experiments is called transfer learning. In transfer learning, a model is

[6] The code for the model can be found in the file model_transformer.py in the scr folder in the GitHub repository.

trained first on a large dataset, after which the model is trained further on a smaller dataset for a specialized task. This is generally beneficial if there is only a small training dataset available for the specialized task, like in our case.

In all the experiments, we varied the number of epochs in the main training loop (20, 25, 30, 35, and 40 epochs).

We checked the accuracy of the predictions on the Syriac test set, which is identical for each experiment. The accuracy is defined as the percentage of graphical units that is predicted fully correctly at a specific index of the test sequences. These test sequences are partly overlapping, just like the training sequences. Therefore, for most words in the test set multiple predictions are made.

The results can be found in figure 1, which shows the results of the index with the highest accuracy.

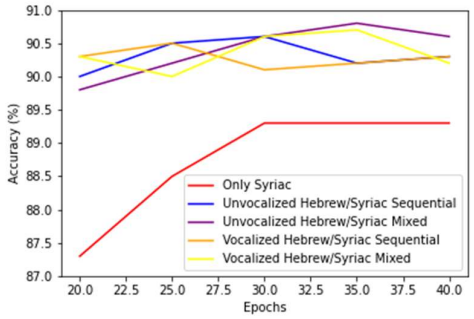

Figure 1. The accuracy of predictions on the Syriac test set with five different training strategies.

The accuracy of the model trained on Syriac data increases with the number of epochs from 87.3% for 20 epochs to 89.3% for 30 or more epochs. The accuracy of the predictions of the models trained on Hebrew and Syriac data vary somewhat between 89.8% (20 epochs) and 90.8% (35 epochs), both achieved by the model trained simultaneously on unvocalized Hebrew and Syriac data.

The models trained on Hebrew and Syriac data perform consistently better than the models trained on Syriac data only. Even though the accuracy of the latter models is only 1-2% higher, this is quite substantial, and it is hard to achieve this result by tuning hyperparameters.

The Hebrew datasets consisting of 22946 verses are substantially bigger than the Syriac datasets (5596 verses) we used. Therefore, training a model with Hebrew data takes substantially longer, which may be a disadvantage for including this dataset, especially if one wants to optimize the model further by tuning the hyperparameters. So, as is often the case, there is a tradeoff between speed and performance.

7 Error analysis

In the predictions on the test set, the model can make different kinds of mistakes. We provide a notebook in the GitHub repository[7], with which each mistaken prediction is classified as one of six error categories, with the goal of further improving the model. The following kinds of mistakes are distinguished:

0. Parse errors in the encoding. In this case, the prediction is ungrammatical according to the parsing conventions.
1. The consonantal form of the prediction and the true surface form differ.
2. Ungrammatical morpheme type combinations. This is the case if there is, e.g., a combination of verbal and nominal morphemes that do not match.
3. Unparadigmatic morphemes. In this case the model predicts a morpheme that falls outside of the ETCBC inventory of paradigmatic Syriac morphemes.
4. Difference in number of analytical words with the true form. In this case, the number of "-" signs in the graphical unit is incorrect.
5. Difference in morphemes with the true form. In this case, the analysis of the word is grammatically correct, but not within the given context, there could for instance be an incorrect number of "=" signs at the end of the lexeme.

[7] This is the file evaluation_syriac.ipynb in the folder badness_analysis.

The results are shown in figure 2. It shows the results for the number of epochs with the highest accuracy.

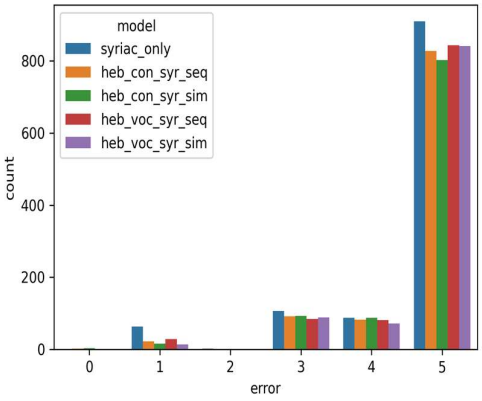

Figure 2. Frequencies of different types of errors made by the five models.

In general, the models show similar patterns. For every model, the most frequent type of error is 5, which means that the parsing is grammatically correct, but not in the given context. The error types 0 and 2 hardly occur.

In most error categories, the model which was trained on Syriac only has more errors than the other models. An important difference between this model and the other models is found in error type 1, indicating errors in the surface text, where the model trained on Syriac has 2-3 times more errors than the models trained on both Hebrew and Syriac data. The consonantal text of the input and output should be identical, and this is language independent. This is a clear sign that adding the Hebrew data helps here, simply because the volume increases. The same may be true for the error categories 3, 4, and 5. Here and there, the Hebrew may help because a morpheme is the same as in Syriac, but it is likely that it helps mostly because it adds volume to the dataset, which helps to make the model more consistent in analyzing morphemes.

8 Conclusions

In this paper we trained a Transformer model from scratch with the goal of analyzing Syriac morphology. An important part of the research was to see if adding Hebrew to the training set would improve the accuracy of the predictions on the Syriac test set. We compared results of the models that were trained on Syriac data alone, models that were trained on (un)vocalized Hebrew and then trained on Syriac, and models that were trained on (un)vocalized Hebrew and Syriac simultaneously. The highest accuracy of the model trained on Syriac data was 89.3%. The best model overall was trained on unvocalized Hebrew and Syriac simultaneously with an accuracy of 90.8%, which outperforms the best “Syriac only” model with 1.5%.

Further improvements can possibly be achieved by optimizing the hyperparameters of the models, but it is clear that adding Hebrew data to the training set helps with improving the performance on the Syriac test set. The same effect may be expected with a larger Syriac dataset, but as long as that dataset is relatively small, adding Hebrew data is a good solution. Another way to expand the dataset is to use data augmentation, which we are considering for future experiments.

It has been shown in other tasks that a model trained on a variety of data can be very useful to be trained further for specialized tasks. In our project we see the same phenomenon. The experiment could be broadened in various ways. One could for instance use one of our models and train it further on data from other languages than Hebrew and Syriac, such as Akkadian or Arabic, or train models to parse Syriac texts syntactically.

Acknowledgments

We thank Martin Ehrensvärd for proofreading the manuscript and the Netherlands eScience Center for their support in this project.

References

Shihadeh Alqrainy and Muhammed Alawairdhi. 2020. Towards Developing a Comprehensive Tag Set for the Arabic Language. *Journal of Intelligent Systems* 30:287–296. https://doi.org/10.1515/jisys-2019-0256.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2021. AraBERT: Transformer-based Model for Arabic Language Understanding. *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, 9–15. https://aclanthology.org/2020.osact-1.2.

Roy Bar-Haim, Khalil Sima'an, and Yoad Winter. 2008. Part-of-speech tagging of Modern Hebrew text. *Natural Language Engineering* 14(2):223–251. https://doi.org/10.1017/S135132490700455X.

Ezra Daya, Dan Roth, and Shuly Wintner. 2004. Learning Hebrew Roots: Machine Learnng with Linguistic Constraints. *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 357–364.

Emiliano Giovannetti, Davide Albanesi, Andrea Bellandi, Simone Marchi, and Alessandra Pecchioli. 2018. Constructing an Annotated Resource for Part-Of-Speech Tagging of Mishnaic Hebrew. *Proceedings of the Fifth Italian Conference on Computational Linguistics CLiC-it 2018: 10–12 December 2018, Torino*. Torino: Accademia University. https://doi.org/10.4000/books.aaccademia.3394.

Moshe Koppel and Avi Shmidman. 2020. Torah Study and the Digital Revolution, a Glimpse of the Future. https://thelehrhaus.com/commentary/torah-study-and-the-digital-revolution-a-glimpse-of-the-future.

Sandra Kübler and Emad Mohamed. 2012. Part of speech tagging for Arabic. *Natural Language Engineering* 18(4):521–548. https://doi.org/10.1017/S1351324911000325.

Gennadi Lembersky, Danny Shacham, and Shuly Wintner. 2012. Morphological disambiguation of Hebrew: A case study in classifier combination, *Natural Language Engineering* 20(1):69–97. https://doi.org/10.1017/S1351324912000216.

Amit Seker, Elron Bandel, Dan Bareket, Idan Brusilovsky, Refael Shaked Greenfeld, and Reut Tsarfaty. 2021. AlephBERT: A Hebrew Large Pre-Trained Language Model to Start-off your Hebrew NLP Application. https://arxiv.org/abs/2104.04052.

Avi Shmidman, Shaltiel Shmidman, Moshe Koppel, and Yoav Goldberg. 2020. Nakdan: Professional Hebrew Diacritizer. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Association for Computational Linguistics 197–203. https://doi.org/10.18653/v1/2020.acl-demos.23.

Martha Y. Tachbelie, Solomon T. Abate, and Laurent Besacier. 2011. Part-of-Speech Tagging for Under-Resourced and Morphologically Rich Languages–The Case of Amharic. *Conference on Human Language Technology for Development, Alexandria, Egypt, 2–5 May 2011*. https://www.cle.org.pk/hltd/pdf/HLTD201109.pdf.

Bas Ter Haar Romeny and Willem Th. Van Peursen (eds.). 1966–. *The Old Testament in Syriac according to the Peshiṭta Version*. Leiden: Brill.

Sebastian Ruder. 2022. The State of Multilingual AI. https://www.ruder.io/state-of-multilingual-ai.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 6000–6010. https://arxiv.org/pdf/1706.03762.pdf.

Amir Zeldes. 2018. A Characterwise Windowed Approach to Hebrew Morphological Segmentation. *Proceedings of the 15th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and* Morphology, 101–110. http://dx.doi.org/10.18653/v1/W18-5811.