A Transformer Architecture for the Prediction of Cognate Reflexes

Giuseppe G. A. Celano
Leipzig University
Faculty of Mathematics and Computer Science
Institute of Computer Science
celano@informatik.uni-leipzig.de

Abstract

This paper presents the transformer model built to participate in the SIGTYP 2022 Shared Task on the Prediction of Cognate Reflexes. It consists of an encoder-decoder architecture with multi-head attention mechanism. Its output is concatenated with the one hot encoding of the language label of an input character sequence to predict a target character sequence. The results show that the transformer outperforms the baseline rule-based system only partially.

1 Introduction

The SIGTYP 2022 Shared Task on the Prediction of Cognate Reflexes investigates[1] the research question of to what extend machine learning can be employed to predict cognate reflexes, i.e., word forms assumed to derive from a common attested or reconstructed word form.

Such words prototypically present phonological features that are in part common to different languages and in part specific to each given language: consistent comparison between different words allows detection of structural similarities and recurrent change patterns, which can be explained by positing the existence of sets of cognate reflexes, i.e., sets of words belonging to different languages but deriving from a common form that has over time undergone consistent—and therefore to a large extent predictable—phonological/structural change according to the rules of each given language.

The study of cognate reflexes, which is at the heart of modern historical-comparative linguistics, was first applied successfully to the Indo-European languages, which have been argued, with ample and compelling evidence, to derive from a common ancestor, the Proto-Indo-European (e.g. see Lehmann, 1952). In this respect, the comparative method has even allowed reconstruction of parallel grammars, with description of detailed phonological and morphosyntactic correspondences between languages (e.g., see Sihler, 1995 for Ancient Greek and Latin).

[1]The code is made available at https://github.com/sigtyp/ST2022

The cognate-related computational research has been characterized by various tasks and approaches. Some are rule-based: for example, Dinu and Ciobanu (2014) automatically identify cognates by linking word etymologies; List (2019a) proposes an algorithm to align cognate sound segments. Others employ machine learning methods: for example, Meloni et al. (2021) build an encoder-decoder-with-attention model to predict the Latin root word common to romance language cognates. Similarly, Dekker and Zuidema (2020) investigate the use of neural networks for prediction of cognates from Slavic and Germanic subfamilies.

In the SIGTYP 2022 Shared Task, participants are requested to predict a cognate reflex form from other cognate reflex forms. The present article reports on the model I have build for the challenge. In Section 2, the dataset is described, while, in Section 3, the method employed to tackle the task is detailed. Section 4 contains the results, and Section 5 their discussion. Section 6 provides some concluding remarks.

2 Dataset

The initial dataset provided for training consisted of cognate reflexes from 10 language families (see List et al. 2022 for a description of the language database). The data for each language family in turn consists of 5 `tsv` training files containing cognate reflexes for a variable number of languages: the 5 files contain training data with different percentages of missing data, ranging from 10% to 50%.

This data structure has been designed to test model reliability on progressively sparser data, with the file containing 50% missing data being the most challenging. The 5 training files are matched

BelejeGonfoye	Fadashi	Maiyu	Undulu
g o r a	k o r a	?	g o r a f a
n d u	n d u	n d u	?
?	b o ʃ a	b ɔ ʃ a	b o: ʃ a

Table 1: Example of entries from a test file of `bremerberta`.

by 5 test files presenting the exact same structure as the training files plus question marks in those table cells whose values are requested to be predicted. Solutions and baseline results for each of the 5 test files are also provided. Table 1 shows a few example rows from a test file of the language family labeled as `bremerberta` (Bremer, 2016), where question marks indicate the cognate reflexes to predict.

It is important to note that the initial training data are only provided in order for the participant to familiarize with data structure: indeed, the test data, which only are required to be submitted for the challenge, contain different languages (also called 'surprise languages'), whose model parameters need to be calculated singularly and independently. The test data also consists of 10 language families. In what follows, I present my work concerning the surprise languages[2] only.

3 Method

3.1 Modeling Strategy and Data Vectorization

The surprise data comes from 10 language families. The data for each language family is organized in separate folders containing 5 training files with different percentage of missing data (from 10% to 50%). Remarkably, the training files with different percentages have overlapping data, and therefore have to be kept separate in the training phase.

Each file corresponds to a table, where columns represent languages and rows examples of cognate reflexes. The data is highly sparse, in that single rows can show more than one empty cell. Moreover, the test files, as shown in Table 1, require prediction of cognate reflexes not only for one but all languages, with some rows having a given language as their target variable and other rows other languages.

To tackle these issues, I have build as many models as the number of languages contained in each training file. The number of models created for each language family can therefore be calculated thus:

$$languages \cdot train\_files = models$$

For example, 45 models are trained for the language family labeled as `hillburmish` because it consists of 9 languages, and 5 training files with different percentages of missing data are available.

The original tabular data of each training file has been reorganized as to create new tabular structures: each language is considered as a target variable in turn, with the other languages' data being used as predictors. To address the issue of sparsity, each new data point is modeled as to only represent one single (predictor) cognate reflex plus its language label.

predictor	language	target (Undulu)
m u l h i	Maiyu	m u l h i
m u l h i	Fadashi	m u l h i
b o ŋ o ʃ	Fadashi	b o ŋ k o ʃ
m b ə m a	BelejeGonfoye	m b u m a

Table 2: Examples of remodelling of training data of `bremerberta`

Table 2 shows how data are remodeled to predict, for example, cognate reflexes of the Undulu language. The rows contain cognate reflexes that may be on one single row in the original data (this holds true for the first two rows). Each cognate reflex is considered—with regard to the target variable—separately. The language column refers to the languages of the cognate reflexes used as predictors. It is to be noted that this modeling strategy allows ignoring the missing data in the original files: if a cognate reflex for a given language is missing, it is simply ignored.

Cognate reflexes have been vectorized as character embeddings, while one hot encoding has been employed for language labels. The term 'character' is here used to refer to all space-separated values provided in the original data (most of which, but not all, correspond to one Unicode character).

3.2 A Transformer Architecture

Transformers have increasingly become popular to solve a variety of NLP tasks, especially through fine-tuning of a pretrained model (e.g, see Wolf et al., 2020).

Transformers are characterized by an encoder-decoder architecture with attention mechanism

[2]This data coincides with that in the `data-surprise` folder at `https://github.com/sigtyp/ST2022`.

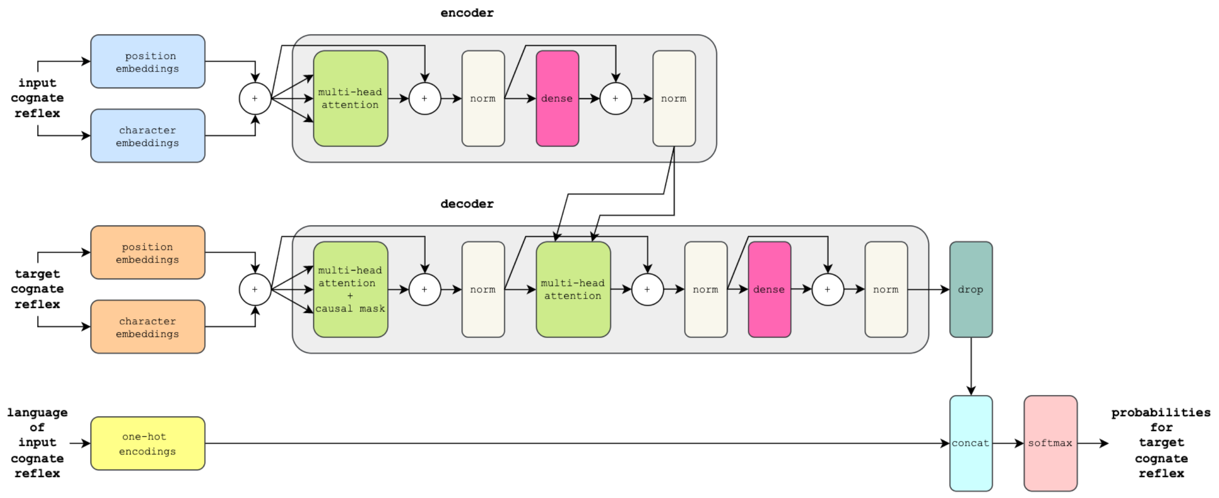

Figure 1: The transformer architecture

(Vaswani et al., 2017). In the present task, the encoder is meant to transform an input sequence of characters into a new context-aware sequence via a multi-head attention layer. The decoder can then predict each character of a target sequence relying on the entire context-aware input sequence and all the target sequence steps *preceding* the (target) sequence step to predict: this is possible because the target sequence used as an input is made context-aware via a multi-head attention layer with masking, which prevents the use of sequence steps from the future.[3]

In the implementation proposed (see Figure 1)[4], the input to the encoder is represented by character embeddings added to character position embeddings: these latter are meant to vectorize the position of each character within the character sequence (indeed, no recurrent neural network will be used to keep track of character order).

The heart of the encoder coincides with a multi-head attention layer outputting a context-aware representation of the input embeddings that accounts for how strongly each character is associated with the others within the character sequence. Remarkably, attention mechanism is character order independent. Since `query`, `key`, and `value` of the encoder's multi-head attention layer are all input character embeddings, the layer instantiates self-attention.

The decoder component has a more complex architecture, which consists of two main layers:

- A multi-head attention layer whose input is a target sequence masked as to avoid that prediction of a target sequence step takes advantage from future steps
- A multi-head attention layer aimed to merge the encoder output, used as `key` and `value`, with the output of the masked multi-head attention layer used as `query`.

The output of the decoder is passed to a dropout layer, and the output of it is then concatenated with the one-hot encoding computed for the language of the input character sequence. Finally, a `softmax` function is meant to output the probabilities for each target character.

The new remodeling of data described in Section 3.1 also requires a strategy to deal with multiple cognate reflexes 'at once' at inference time: since test files contain more than one cognate reflex for a given target cognate reflex (i.e., there are many cognate reflexes as predictors on a single row), the probabilities returned for each predictor cognate reflex are summed and then averaged (by the number of predictor cognate reflexes) to produce one single target tensor of probabilities (see Figure 2). At inference time, the string for the target cognate reflex used as an input first consists only of a dollar sign, which conventionally represents the beginning of a target character sequence: recursively, after a character is predicted, it is added to the target character

[3]At training time, the target character sequence and the target character sequence used as input differ in that the former is offset by one step.

[4]The architecture is the one implemented at `https://github.com/keras-team/keras-io/blob/master/examples/nlp/neural_machine_translation_with_transformer.py`, adapted for the present SIGTYP 2022 Shared Task to account for the presence of language labels as predictors. The code is available at `https://github.com/sigtyp/ST2022/tree/main/systems/PRECOR_transformer`.

sequence used as an input, so that it can also be used for prediction of the following character, until a hash sign, which conventionally signals the end of a cognate reflex, is reached.

4 Results

Results are shown in Table 3. The scores given for each language family are averages over all the scores for each language within a given language family. They have been calculated using the function `compare_words` made available by the SIGTYP 2022 Shared Task organizers. Three main metrics have been employed: the Levenshtein distance (Levenshtein, 1965), the B-Cubed F-scores (List, 2019b), and the BLEU scores (Papineni et al., 2002). In Table 3, each score of the transformer is provided together with the corresponding one of the baseline rule-based system (this latter being in parentheses). Highlighted are the best scores for each language family. More details on the metrics of the SIGTYP 2022 Shared Task and the baseline scores are given in List et al. (2022).

Proportion 0.1			
Language family	**ED**	**B-Cubed FS**	**BLEU**
bantubvd	1.37 (1.13)	0.67 (0.78)	0.52 (0.61)
beidazihui	1.04 (1.10)	0.67 (0.73)	0.59 (0.58)
birchallchapacuran	2.42 (1.63)	0.51 (0.65)	0.36 (0.53)
bodtkhobwa	0.56 (0.49)	0.73 (0.76)	0.69 (0.72)
bremerberta	1.39 (1.72)	0.71 (0.72)	0.63 (0.50)
deepadungpalaung	1.26 (1.07)	0.74 (0.76)	0.39 (0.44)
hillburmish	1.66 (1.21)	0.53 (0.65)	0.42 (0.56)
kesslersignificance	2.49 (2.77)	0.45 (0.49)	0.15 (0.16)
luangthongkumkaren	0.87 (0.38)	0.76 (0.91)	0.65 (0.84)
wangbai	0.81 (0.62)	0.73 (0.80)	0.65 (0.73)

Proportion 0.2			
Language family	**ED**	**B-Cubed FS**	**BLEU**
bantubvd	1.70 (1.38)	0.58 (0.69)	0.45 (0.53)
beidazihui	1.03 (1.14)	0.64 (0.68)	0.59 (0.57)
birchallchapacuran	2.90 (2.02)	0.44 (0.58)	0.29 (0.45)
bodtkhobwa	0.62 (0.45)	0.67 (0.75)	0.66 (0.75)
bremerberta	1.68 (1.68)	0.61 (0.67)	0.53 (0.51)
deepadungpalaung	1.51 (1.30)	0.58 (0.67)	0.33 (0.39)
hillburmish	1.76 (1.23)	0.49 (0.63)	0.39 (0.55)
kesslersignificance	2.56 (2.93)	0.38 (0.40)	0.14 (0.14)
luangthongkumkaren	1.01 (0.47)	0.68 (0.87)	0.59 (0.80)
wangbai	1.00 (0.76)	0.64 (0.75)	0.60 (0.68)

Proportion 0.3			
Language family	**ED**	**B-Cubed FS**	**BLEU**
bantubvd	2.18 (1.55)	0.47 (0.66)	0.34 (0.51)
beidazihui	1.09 (1.12)	0.60 (0.67)	0.56 (0.57)
birchallchapacuran	3.19 (2.36)	0.39 (0.54)	0.26 (0.41)
bodtkhobwa	0.67 (0.48)	0.64 (0.73)	0.63 (0.72)
bremerberta	1.97 (1.84)	0.55 (0.63)	0.46 (0.49)
deepadungpalaung	1.63 (1.35)	0.50 (0.60)	0.30 (0.38)
hillburmish	2.00 (1.40)	0.44 (0.58)	0.33 (0.49)
kesslersignificance	2.78 (3.10)	0.33 (0.35)	0.13 (0.11)
luangthongkumkaren	1.13 (0.45)	0.66 (0.87)	0.56 (0.81)
wangbai	1.10 (0.81)	0.60 (0.72)	0.56 (0.66)

Proportion 0.4			
Language family	**ED**	**B-Cubed FS**	**BLEU**
bantubvd	2.16 (1.64)	0.45 (0.63)	0.34 (0.49)
beidazihui	1.12 (1.11)	0.59 (0.67)	0.55 (0.57)
birchallchapacuran	3.51 (2.82)	0.37 (0.50)	0.24 (0.36)
bodtkhobwa	0.71 (0.59)	0.62 (0.69)	0.60 (0.66)
bremerberta	2.32 (2.32)	0.48 (0.57)	0.40 (0.39)
deepadungpalaung	1.84 (1.51)	0.43 (0.54)	0.24 (0.32)
hillburmish	2.17 (1.58)	0.41 (0.54)	0.30 (0.45)
kesslersignificance	2.89 (3.91)	0.28 (0.30)	0.13 (0.05)
luangthongkumkaren	1.23 (0.53)	0.61 (0.85)	0.53 (0.78)
wangbai	1.29 (0.90)	0.55 (0.70)	0.49 (0.62)

Proportion 0.5			
Language family	**ED**	**B-Cubed FS**	**BLEU**
bantubvd	2.36 (2.00)	0.41 (0.57)	0.29 (0.39)
beidazihui	1.18 (1.15)	0.56 (0.66)	0.53 (0.56)
birchallchapacuran	3.72 (3.17)	0.34 (0.47)	0.21 (0.31)
bodtkhobwa	0.81 (0.62)	0.58 (0.66)	0.55 (0.65)
bremerberta	2.50 (2.53)	0.44 (0.53)	0.36 (0.34)
deepadungpalaung	2.04 (1.62)	0.36 (0.49)	0.19 (0.30)
hillburmish	2.42 (2.13)	0.37 (0.46)	0.25 (0.33)
kesslersignificance	3.00 (4.06)	0.27 (0.28)	0.11 (0.05)
luangthongkumkaren	1.47 (0.66)	0.55 (0.81)	0.46 (0.73)
wangbai	1.54 (1.02)	0.49 (0.66)	0.42 (0.58)

Table 3: Results for the transformer and the baseline rule-based system (in parentheses).

5 Discussion

The presence of 5 different training sets for each language family and the need of building a model for each language within a language family (see Section 3) resulted in the training of as high a number of models as 495. Since training of different algorithms and, especially, hyperparameter optimization for each model would have required a high computation load, I focused on a transformer architecture with fixed hyperparameters for all languages.[5]

The Levenshtein distance shows that the transformer always performs better than the rule-based system relative to the language family `kesslersignificance`. The transformer's performances for `beidazihui` and `bremerberta` are better relative to the datasets with proportions 0.1, 0.2, and 0.3 and the datasets with proportions 0.1 and 0.5, respectively.

The major challenge posed by the SIGTYP 2022 Shared Task seems to be data scarsity and sparsity, which affects data representativeness. The transformer's model tended to overfit the training data. A dropout layer and early stopping were employed, but more regularization and hyperparameter tuning would probably lead to better results. Due to the high variance, I consider the results of the transformer and the baseline model very similar.

[5]Details can be found at `https://github.com/sigtyp/ST2022/tree/main/systems/PRECOR_transformer`.

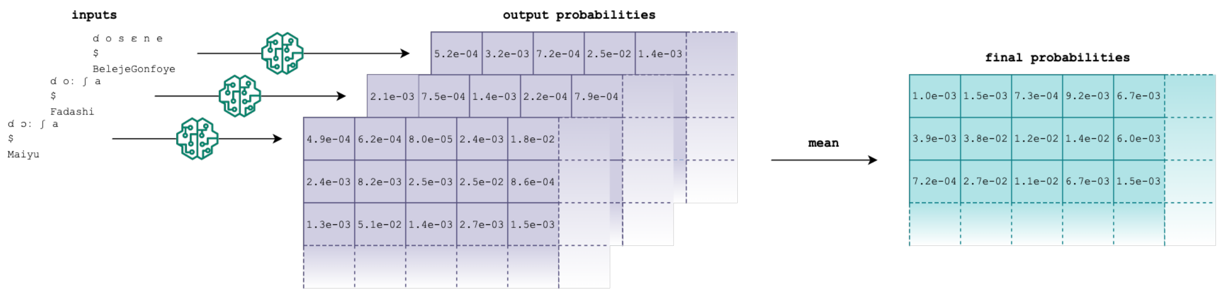
inputs
ɗ o s ɛ n e
$
BelejeGonfoye
ɗ oː ʃ a
$
Fadashi
ɗ ɔː ʃ a
$
Maiyu
output probabilities
5.2e-04 3.2e-03 7.2e-04 2.5e-02 1.4e-03
2.1e-03 7.5e-04 1.4e-03 2.2e-04 7.9e-04
4.9e-04 6.2e-04 8.0e-05 2.4e-03 1.8e-02
2.4e-03 8.2e-03 2.5e-03 2.5e-02 8.6e-04
1.3e-03 5.1e-02 1.4e-03 2.7e-03 1.5e-03
mean
final probabilities
1.0e-03 1.5e-03 7.3e-04 9.2e-03 6.7e-03
3.9e-03 3.8e-02 1.2e-02 1.4e-02 6.0e-03
7.2e-04 2.7e-02 1.1e-02 6.7e-03 1.5e-03

Figure 2: Example of calculation of final probabilities at inference time (fictitious numbers). The inputs are represented by cognate reflexes of one single row in the file `test-0.10.tsv` of `bremerberta`.

6 Conclusions

The paper presented the transformer model built for the SIGTYP 2022 Shared Task. Despite the transformer's complex architecture, which can model input characters in context and even rely on past target sequence steps, its performance was not overall superior to that of the baseline rule-based system. This may be due to data scarsity and sparsity. For this reason and in light of the considerable computational overhead that may be required at inference time, in that target sequence decoding may involve thousands of dictionary lookups—which can only be executed on CPU—one might prefer to test simpler model architectures.

Supplementary Material

The code described in this paper has been archived at `https://github.com/sigtyp/ST2022/releases/tag/v1.4` and `https://doi.org/10.5281/zenodo.6586772`.

Acknowledgements

This work has been supported by the German Research Foundation (DFG project number 408121292).

References

Nate D. Bremer. 2016. *A sociolinguistic survey of six Berta speech varieties in Ethiopia*. SIL International, Addis Ababa.

Peter Dekker and Willem Zuidema. 2020. Word prediction in computational historical linguistics. *Journal of Language Modelling*, 8:295–336.

Liviu P. Dinu and Alina Maria Ciobanu. 2014. Building a dataset of multilingual cognates for the Romanian lexicon. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1038–1043.

Winfred Philipp Lehmann. 1952. *Proto-Indo-European Phonology*. University of Texas Press.

Vladimir. I. Levenshtein. 1965. Dvoičnye kody s ispravleniem vypadenij, vstavok i zameščenij simvolov. *Doklady Akademij Nauk SSSR*, 163(4):845–848.

Johann-Mattis List. 2019a. Automatic inference of sound correspondence patterns across multiple languages. *Computational Linguistics*, 45(1):137–161.

Johann-Mattis List. 2019b. Beyond Edit Distances: Comparing linguistic reconstruction systems. *Theoretical Linguistics*, 45(3-4):1–10.

Johann-Mattis List, Ekaterina Vylomova, Robert Forkel, Nathan W. Hill, and Ryan Cotterell. 2022. The SIGTYP 2022 shared task on the prediction of cognate reflexes. In *The Fourth Workshop on Computational Typology and Multilingual NLP*, Online. Association for Computational Linguistics.

Carlo Meloni, Shauli Ravfogel, and Yoav Goldberg. 2021. Ab antiquo: Neural proto-language reconstruction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4460–4473.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Andrew L. Sihler. 1995. *New Comparative Grammar of Greek and Latin*. Oxford University Press.

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*, 30.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.