Restoring the Sister: Reconstructing a Lexicon from Sister Languages using Neural Machine Translation

Remo Nitschke
The University of Arizona
nitschke@email.arizona.edu

Abstract

The historical comparative method has a long history in historical linguists. It describes a process by which historical linguists aim to reverse-engineer the historical developments of language families in order to reconstruct proto-forms and familial relations between languages. In recent years, there have been multiple attempts to replicate this process through machine learning, especially in the realm of cognate detection (List et al., 2016; Ciobanu and Dinu, 2014; Rama et al., 2018). So far, most of these experiments aimed at actual reconstruction have attempted the prediction of a proto-form from the forms of the daughter languages (Ciobanu and Dinu, 2018; Meloni et al., 2019). Here, we propose a reimplementation that uses modern related languages, or sisters, instead, to reconstruct the vocabulary of a target language. In particular, we show that we can reconstruct vocabulary of a target language by using a fairly small data set of parallel cognates from different sister languages, using a neural machine translation (NMT) architecture with a standard encoder-decoder setup. This effort is directly in furtherance of the goal to use machine learning tools to help under-served language communities in their efforts at reclaiming, preserving, or reconstructing their own languages.

1 Introduction

Historical linguistics has long employed the historical comparative method to establish familial connections between languages and to reconstruct proto-forms (cf. Klein et al., 2017b; Meillet, 1967). More recently, the comparative method has been employed by revitalization projects for lexical reconstruction of lost lexical items (cf. Delgado et al., 2019). In the particular case of Delgado et al. (2019), lost lexical items of the target language are reconstructed by using equivalent cognates of still-spoken modern sister languages, i.e., languages in the same language family that share some established common ancestor language and a significant amount of cognates with the target language. By reverse-engineering the historical phonological processes that happened between the target language and the sister-languages, one can predict what the lexical item in the target language should be. This is essentially a twist on the comparative method, using the same principles, but to reconstruct a modern sister, as opposed to a proto-antecedent.

While neural net systems have been used to emulate the historical comparative method[1] to reconstruct proto-forms (Meloni et al., 2019; Ciobanu and Dinu, 2018) and for cognate detection (List et al., 2016; Ciobanu and Dinu, 2014; Rama et al., 2018), there have not, to the best of our knowledge, been any attempts to use neural nets to predict/reconstruct lexical items of a sister language for revitalization/reconstruction purposes.

Meloni et al. (2019) report success for a similar task (reconstructing Latin proto-forms) by using cognate pattern lists as a training input. Instead of reconstructing Latin proto-forms from only Italian roots, they use Italian, Spanish, Portuguese, Romanian and French cognates of Latin, i.e., mapping from many languages to one. As our intended use-case (see section 1.1) is one that suffers from data sparsity, we explicitly explore the degree to which expanding the list of sister-languages in the many-to-one mapping can compensate for fewer available data-points. Since the long-term goal of this project is to aid language revitalization efforts, the question of available data is of utmost importance. Machine learning often requires vast amounts of data, and languages which are undergoing revitalization usually have very sparse amounts of data available. Hence, the goal for a machine learning approach

[1] Due to the nature of neural nets we do not know whether these systems actually emulate the historical comparative method or not. What is meant here is that they were used for the same tasks.

Proceedings of the First Workshop on Natural Language Processing for Indigenous Languages of the Americas, pages 122–130

here is not necessarily the highest possible accuracy, but rather the ability to operate with as little data as possible, while still retaining a reasonable amount of accuracy.

Our particular contributions are:

1. We demonstrate an approach for reframing the historical comparative method to reconstruct a target language from its sisters using a neural machine translation framework. We show that this can be done with easily accessible open source frameworks such as OpenNMT (Klein et al., 2017a).

2. We provide a detailed analysis of the degree to which inputs from additional sister languages can overcome issues of data sparsity. We find that adding more related languages allows for higher accuracy with fewer data points. However, we also find that blindly adding languages to the input stream does not always yield said higher accuracy. The results suggest that there needs to be a significant amount of cognates with the added input language and the target language.

1.1 Intended Use-Case and Considerations

This experiment was designed with a specific use-case in mind: Lexical reconstruction for language revitalization projects. Specifically, the situation where this type of model may be most applicable would be a *language reclamation* project in the definition of Leonhard (2007) or a *language revival* process in the definition of McCarty and Nicholas (2014). In essence, a language where there is some need to *recover or reconstruct* a lexicon. An example of such a case might be the Wampanoag language reclamation project (`https://www.wlrp.org/`), or comparable projects using the methods outlined in Delgado et al. (2019).

As this is a proof-of-concept, we use the Romance language family, specifically the non-endangered languages of French, Spanish, Italian, Portuguese and Romanian, and operate under assumption that these results can inform how one can use this approach with other languages of interest. However, we are aware that the Romance language morphology may be radically different from some of the languages that may be in the scope of this use case, such as agglutinative and polysynthetic languages, and that we cannot fully predict the performance of this type of system for such languages from the Romance example. Regardless of this, some insights gained here will still be applicable in those cases, such as the question of compensating lack of data by using multiple languages.

Languages that are the focus of language revitalization projects are typically not targets for deep learning projects. One of the reasons for this is the fact that these languages usually do not have large amounts of data available for training state of the art neural approaches. These systems need large amounts of data, and Neural Machine Translation systems, as the one used in this project, are no exception. For example, Cho et al. (2014) use data sets varying between 5.5million and 348million words. However, the task of proto-form reconstruction, which is really a task of cognate prediction, can be achieved with fairly small datasets, if parallel language input is used. This was shown by Meloni et al. (2019), whose system predicted 84% within an edit distance of 1, meaning that 84% of the predictions were so accurate that only one or 0 edits were necessary to achieve the true target. For example, if the target output is “grazie", the machine might predict “grazia" (one edit) or "grazie" (0 edits). Within a language revitalization context, this level of accuracy would actually be a very good outcome. In this scenario, a linguist or speaker familiar with the language would vet the output regardless, so small edit distances should not pose a big problem. Further, all members of a language revitalization project or language community would ultimately vet the output, as they would make a decision on whether to accept or reject the output as a lexical item of the language.

This begs the question of why a language revitalization project would want to go through the trouble of using such an algorithm in the first place, if they have someone available to vet the output, then that person may as well do the reconstructive work themselves, as proposed in Delgado et al. (2019). This all depends on two factors: First, how high is the volume of lexical items that need to be reconstructed or predicted? The effort may not be worth it for 10 or even a 100 lexical items, but beyond this an neural machine translation model can potentially outperform the manual labor. Once trained, the model can make thousands of predictions in minutes, as long as input data is available.

Second, and potentially more important, it will depend on how well the historical phonological relationships between the languages are understood.

	Spanish	French	Portuguese	Romanian	Italian (target)	status
1	-	-esque	-e:scere	-	-	*removed, no target*
2	mosto	moût	mosto	must	mosto	
3	-	-	-	-	lugano	*removed, no input*
4	párrafo	-	-	-	paragrafo	
5	-edad	-	-idade	-itate	-ità	

Table 1: Examples of data patterns, including types of data removed during cleanup (e.g., rows 1 and 3).

For a family like Romance, we have a very good understanding of the historical genesis of the languages and the different phonological processes they underwent, see for example Maiden et al. (2013). However, there are many language families in the world where these relationships and histories are less than clear. In such situations, a machine learning approach would be beneficial, because the algorithm learns[2] the relationships for us and gives predictions that just need to be vetted.

Under this perspective, the best model might not necessarily be the one that produces the most accurate output, but perhaps the one that produces the fewest incorrigible mistakes. An incorrigible mistake here would be the algorithm predicting an item that is completely unrelated to the target root e.g., predicting "cinque" for a target of "grazie"). Further, ease of usability and accessibility will be another factor for this kind of use-case, as not every project of this type will have a computational linguist to call on. Hence, another aim should be a low-threshold for reproducability and the utilization of easy to use open-source frameworks. In the spirit of the latter, all data and code necessary to reproduce the results are open-source and freely available. This paper is intended for computational linguists and linguists and/or community members who are involved with projects surrounding languages which might benefit from this approach. As such, it is written with both audiences in mind, with Section 6 ("Warning Labels for Interested Linguists") specifically aimed at linguists and community members interested in a potential application of this method.

2 The Dataset

The data set used for this experiment was provided by Shauli Rafvogel of Meloni et al. (2019). The initial set consisted of 5420 lines of cognate sextuples of the Romance language family, specifically: Romanian, French, Spanish, Portuguese, Italian and Latin. As the aim for this experiment was to reconstruct from sister languages to a sister language, the Latin items were removed from the set and instead Italian was chosen to be the target language for the experiment, since it had the most complete pattern with respect to the other languages in the set. Table 1 illustrates the types of lines present in the initial dataset.

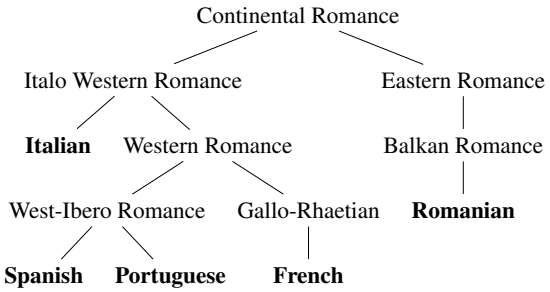

Figure 1: An abridged family tree of the relevant Romance languages. Adapted from glottolog (Hammarström et al., 2020).

Lines with no target and lines with no input were removed. Lines where there was a target but no input (row 3) were also removed, as well as lines where there was input but no target (line 1). After the removal of all lines which lead to empty patterns in the Italian set, and all lines which were empty patterns in the input, 3527 remained. From these, 2466 lines were taken as training data, 345 were taken for validation, and 717 were set aside for testing.

Meloni et al. (2019) use both an orthographic and an IPA data set, and show that the orthographic set yielded more accurate results. Here, we use only orthographic representations, which we prefer not for accuracy, but because orthographic datasets are more easily acquired for most languages, particularly those of interest in language reclamation projects. If both an IPA set and an orthographic set are available, one may attempt using both to boost the accuracy of the results. Chen (2018) showed

[2]Or, rather, it interprets.

that this is possible with glossing data in the case of sentence level neural machine translation. We will discuss this implementation in Section 5.2.

See Figure 1 for a very *simplified* phylogenetic tree representation of the familial relations of the Romance languages used in this dataset. This tree was constructed using data from glottolog (Hammarström et al., 2020), and is included just for illustrative purposes and not as a statement about the phylogeny of Romance languages.[3]

3 Experimental Setup

This experiment was run using the OpenNMT-pytorch neural machine translation (Klein et al., 2017a) framework, using the default settings (a 2-layer LSTM with 500 hidden units on both the encoder and decoder). The opennmt-py default setup was chosen intentionally; the envisioned use-case requires an easily reproducable approach for interested users or communities who might profit from using this method for their own purposes, but who don't necessarily have deep expertise in machine learning or tuning neural models. A publicly available toolkit, like opennmt, and a no-configuration setup helps lower the bar to entry for these parties.

Neural machine translation (NMT) frameworks are designed to translate sentences from one language to another, but they can be used for a number of sequential data tasks (Neubig, 2017). One such task is the prediction of a cognate from a set of input words, as used here. These frameworks are typically an encoder-decoder setup, where both the encoder and decoder are often implemented as LSTM (Long Short-Term Memory) networks (Hochreiter and Schmidhuber, 1997), which have the advantage of effectively capturing long-distance dependencies (Neubig, 2017). In an encoder-decoder setup, the encoder reads in the character based input representation and transforms it into a vector representation. The decoder takes this vector representation and transforms it into a character based output representation (Cho et al., 2014).

NMT frameworks also employ a "vocabulary" set, which contains vocabulary of the language that is being translated from and vocabulary of the language that is being translated to. The size of this vocabulary is often an issue for the effectiveness of NMT models (Hirschmann et al., 2016). In our case, the source vocabulary simply contains all of the characters that occur in all the input language examples and the target vocabulary contains the characters that occur in the target language example. To illustrate: if this task was about predicting English words, then the target vocabulary would contain all the letters of the English alphabet.

3.1 Input Concatenation

Since the input in our case is a list of cognates from different languages, we need to consider how we feed this input to the machine. There are two obvious options for this task. We can either feed the cognates one by one, or we can merge the cognates first, before feeding them to the machine. In this experiment, we merge the words character by character to construct the input lines. This means that for every line in the input, the first character of each word was concatenated, then the second character of each word was concatenated, and so on. For an illustration:

(1) patterns in the input: aille, alha, al, aie

(2) target patterns: aglia

(3) $input$: aaaaillilhelae

(4) $target$: aglia

This merging delivered marginally better results than simple concatenation in early testing, which is why it was selected. It is unclear as to why this is the case. We suspect that the merged input makes it easier for the model to recognize if the same characters appear in the same position of the input, as is the case with "a" in the initial position in the above example. However, we are cautious to recommend this input representation in general, because different morphologies may be better represented in a concatenation.

3.2 Different Training Setups

To determine the performance gains from simply having more data versus having data from more languages, we create several training scenarios. In each, we use the same aforementioned 2-layer LSTM. To understand the benefit of additional language, we first train with the entire training set with all four languages, then successively remove languages from the input set until only one remains. Next, to compare this to the impact of simply having fewer data points, but from all languages, we generate several impoverished versions of the data set. For these impoverished versions, lines were

[3] We also acknowledge that tree representations are not necessarily the most accurate way to represent these relationships (Kaylan and François, 2019).

removed randomly[4] from the set reducing the data by 70%, 50%, 30% and 10% respectively.

4 Evaluation Measures

Machine translation is usually evaluated using the BLEU (Papineni et al., 2002) score, but BLEU is designed with sentence level translations in mind. We instead evaluate the output according to edit distance in the style of Meloni et al. (2019) by calculating the percentage of the output which is within a given edit distance. In addition to this metric, we also use a custom evaluation metric designed to emphasize the *usability* of the output for the intended use-case, i.e., as predictions to be vetted by an expert to save time over doing the entire analysis manually. In order to calculate this score, we calculate the Damerau-Levenshtein edit distance to the target for each word and assign weights to them by their edit distance. That is:

$$score = (a + b * .9 + c * .8 + d * .7 + e * .6)/t$$

where a is number of predictions with distance 0, b is the number with distance 1, c is the number with distance 2, d is the number with distance 3, e the number with distance 4, and t is the total number of predictions. As an example, consider a scenario where there are three predicted cognates. If system 1 produces 3 output patterns within an edit distance of 2, it would receive a score of 0.8. If system 2 produces two output patterns with edit distance 0 and one within a distance of 5, this would result in a score of 0.67.

The logic behind this metric is that any prediction with an edit distance larger than 4 edits is essentially useless for the proposed task. Since such a large edit distance essentially constitutes an incorrigible mistake as mentioned in (Section 1.1). The edit distance of 4 constitutes an arbitrary cut-off to a degree, but it allows us a simple and informative evaluation metric for our use case. This metric will rank a model that has a large number of items in a and a large number of items beyond 4 edits lower than a model with items mostly in the b-d range. Presumably, the latter is more useful to the task, as small errors can be adjusted by linguists or language users.

Using this metric, we can rank different input combinations according to their assumed usefulness to the task of lexical reconstruction for revitalization purposes.

5 Results

Table 2 shows the edit distance percentages and scores of different runs at 10,000 steps of training.[5] We can compare the difference in outcome between using fewer languages in the input versus using less input lines overall. This addresses the question of whether adding multiple languages to the input helps compensate for fewer data points (cognate pairs). The runs with successively reduced numbers of languages (top half of the table), are all trained with all available input lines (2466) but excluding specific columns/languages. The "reduced input" runs (bottom half of the table), on the other hand, are done with all four languages but with fewer cognates, by excluding rows. These runs had the following amount of training input lines: 10%: 2220 lines of input, 30%: 1793 lines of input, 50%: 1345 lines of input, 70%: 896 lines of input (recall that the total number of input lines available for training was 2466). All runs were tested on the same testing data target.

In Table 2 (see following page), we can observe that, unsurprisingly, the training sample with the most languages and data (Span-Fre-Port-Ro) performs best. 44.6% within edit distance 0 means that almost half the predictions the machine makes are correct. In terms of accuracy, this is not incredible, Meloni et al. (2019) report 64.1% within edit distance 0. However, considering that we are using a data set approximately a third the size of theirs for training (2466 cognates compared with 7038), the performance is surprisingly good. The more important measure for the intended use-case is the fact that over 80% of items are within an edit distance of 3, meaning that of the output produced, 80% need only three edits or fewer to meet the target.

We can also observe that the performance successively drops as we remove languages, with the Spanish only[6] performing worst. However, the way in which this performance drops is not entirely transparent. It appears that in terms of scoring, the Spanish-French (Spa-Fre) sample actu-

[4] This was done by simply removing every n^{th} line depending on how much reduction was needed.

[5] One step of training means that the algorithm has gone through one batch of input lines. The default batch-size for opennmt is 64.

[6] Spanish only was only trained for 5000 steps, as the model plateaus around 1000 steps. The performance of the Spanish only model was measured every 500 steps for Figure 2.

Edit Distance	0	≤1	≤ 2	≤3	≤4	score
Span-Fre-Port-Ro	44.63%	57.74%	69.6%	80.33%	88.42%	0.82
Span-Fre-Port	42.68%	53.27%	68.34%	77.68%	84.94%	0.78
Span-Port-Ro	42.54%	53.28%	66.39%	74.76%	81.59%	0.75
Span-Fre	39.9%	50.9%	63.88%	74.62%	83.4%	0.76
Spanish only	35.6%	47.98%	60.25%	69.03%	74.76%	0.68
10% Reduced Input	40.17%	54.25%	69.6%	81.31%	87.59%	0.8
30% Reduced Input	39.75%	50.91%	66.11%	73.36%	83.12%	0.77
50% Reduced Input	33.19%	45.61%	60.95%	71.27%	82.4%	0.75
70% Reduced Input	17.02%	26.08%	41%	50.77%	65.97%	0.59

Table 2: Edit distance percentiles at 10,000 training steps. Shown are the results from using all data points with different combinations of languages (top), as well as using all languages but with random downsampling of the data from each (bottom). All scores are calculated from the testing data.

ally performs better than the Spanish-Portuguese-Romanian sample. Further, while Span-Port-Ro has significantly better values in the 0-2 edit range, it is outperformed by Span-Fre in terms of score because Span-Fre has more items in the ≤ 4 edit range.

The noticeable difference between Span-Fre-Port and Span-Port-Ro is surprising and warrants some examination. The likely explanation is twofold. First, The Romanian set is the one with the most empty patterns. The Romanian training data only includes 930 filled patterns, in comparison, Portuguese includes 1905 patterns, French includes 1790, and Spanish has 2125. It may be the case that the Romanian data is too small in comparison with the others to have a significant impact on the outcome. The other factor may be that Romanian is phylogenetically the most distant from the target language (Italian) (Figure 1).

This becomes even more apparent in Figure 2, which. shows the performance of different models over time.[7] Here we can observe that there is hardly any difference between the performance of Span-Fre-Port-Ro and Span-Fre-Port over time, and it is only at 10,000 steps that they start to diverge. This divergence at the 10,000 step mark is likely random, the graph suggest that their overall performance is almost identical in regards to scoring. Another point in this direction are the seemingly convergent graphs of Span-Fre and Span-Port-Ro, suggesting that there is no difference between using 2 or 3 languages as input if the third language

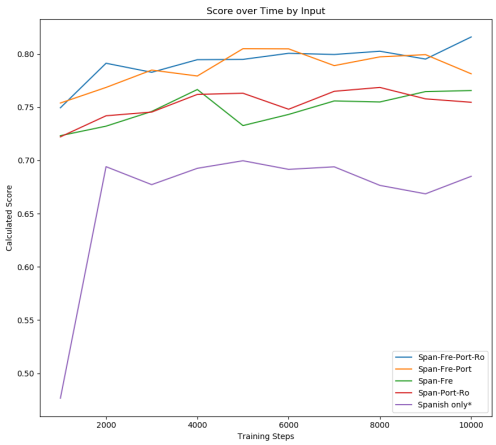

Figure 2: Performance at different training steps for models with different combinations of input languages, plotted by custom score. All scores are calculated from the testing data.

is Romanian.

Discounting the performance of the exclusion/inclusion of Romanian, we can observe that performance overall tends to increase with each parallel language added. This is especially evident with the obvious drop-off in performance of the Spanish only input. If we assume that Romanian has no impact, then we can see that 3 languages (blue and orange) perform similarly and two languages (red and green) perform similarly, and there is an obvious drop-off between those two patterns. This suggests that using parallel language input can compensate smaller datasets.

Due to the small dataset, the scores plateau fairly early, around the 3000 epoch mark for most. This

[7]This can give a better representation of the performance, because a neural net constantly adjusts its weights, so looking at just one point in time can be deceiving.

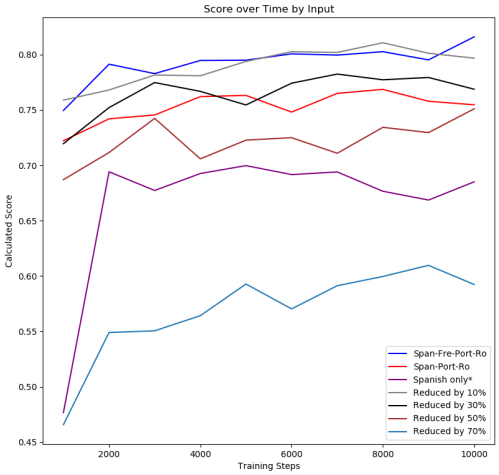

Figure 3: Performance of models trained on all four languages, but with varying levels of downsampled data. Included for comparison are models trained with all data on different language combinations. Plotted is the custom score over steps. Scores are calculated every 1000 training steps. All models were run on OpenNMT-py default parameters.

suggests that it would be sufficient to run these models at 3000 epochs, which would save some time on low-end hardware. However, with these small datasets, training time should rarely exceed 5 hours on consumer grade PCs.[8]

5.1 Parallel Languages vs Input Reduction

Let us now consider the second question of this paper: Can parallel language input compensate for small dataset size? We know that performance reduces if we reduce the number of languages in the input mix. Now we compare this drop-off to the reduction in performance caused by reducing the overall amount of input data. This can be seen in Figure 3, which shows the performance at different training steps for models trained on decreasing amounts of data. Included for comparison are models trained on all data using all four (Span-Fra-Port-Ro), three (Span-Port-Ro), and one (Span) input language.[9]

First, we observe that a 10% reduction in training data (grey) does not seem to have a strong impact, as this performs mostly equal to Span-Fre-Port-Ro. Further, we can see is that the 30% reduced case performs marginally better than Span-Port-Ro. This is a good result, as it suggests that we can compensate for a fair amount of data by using additional languages. Essentially, in this case we can observe that removing a language from the input can be equivalent to removing 30% of the input or more. Even the 50% reduced case (brown) still performs better than using just one language (Spanish only).

The extreme fall-off between the 50% reduction and the 70% reduction suggests that there is some point beyond which even multiple languages cannot compensate for lack of data points. Where this fall-off point is exactly, will likely fluctuate depending on the data set.

5.2 Potential Improvements

Chen (2018) shows that neural machine translation tasks can be greatly improved by adding glossing data to the input mix (We will gloss over the technical details of the implementation here). While there is no direct equivalent to the gloss-sentence relationship, there might be a close analog for words: phonetic transcriptions. Orthography may be conservative and often misleading, but phonetic representations are not.

Meloni et al. (2019) use a phonetic dataset in their experiment, but they map from phonetic representations to phonetic representations, so their input and their target items are represented in IPA. This performs worse than the orthographic task. An interesting further experiment would be to blend orthographic representations and phonetic representations in the input, in the style of Chen (2018), mapping that to an orthographic output. This would be a close analog to the sentence-gloss to sentence mapping that Chen (2018) reports success with.

One thing to consider, is that this may be not ideal for the use-case. Phonetic datasets are not easy to produce and the orthography is often more readily available. While this might improve performance, needing a phonetic as well as an orthographic dataset would likely increase the threshold of reproducability for interested parties.

6 Warning Labels for Interested Linguists

There are some important aspects of this kind of approach that linguists, or community members who are interested in utilizing it for their purposes,

[8] These were trained on an i5-5200 CPU with 2.2GHz, and training took anywhere between 4-7 hours for 10,000 steps.

[9] Since in Figure 2 we observe that Span-Port-Ro and Span-Fre perform quite similarly, and Span-Fre-Port performs similarly to Span-Fre-Port-Fro, to make the graph easier to read, we remove Span-Fre and Span-Fre-Port from this graph.

should be aware of.

There are certain things that this type of approach can and cannot do for a community or project. The model does not so much reconstruct a word for the community, but rather proposes what the word *could be*, according to the data it has been fed. The model will propose these recommendations on the basis of an abstract notion of what the historic phonological and morphological differences are between languages ABC and language D. This does not necessarily mean that the model *learns* or *understands* the historical phonological and morphological processes that separate the input sister languages from the target languages. It has simply learned a way to generalize from the input to the output with some degree of accuracy. What is learned need not necessarily overlap with what linguists believe to have happened.

Therefore, this type of model will *only ever generate cognates of the input*. It cannot generate novel items. This is an important factor to consider for any community or linguist planning on using this approach.

Consider the following case: Imagine we are trying to use this approach to reconstruct English from other Germanic languages. A large part of the English lexicon is not of Germanic ancestry. However, any lexicon we would try to reconstruct using this trained algorithm would give us *approximations of a Germanic derived lexeme* for the word we are trying to reconstruct. This is a potentially undesirable effect of the way the model was trained. Linguists and interested community members need to be aware of this and implement their own quality control.

However, this approach can potentially be useful for any language project where a community and or linguists are working with an incomplete lexicon for a language. The prerequisite for this being a useful tool in such a scenario is the assumption that the *sister languages* to the target language are somewhat well documented and have at least dictionaries available from which data can be extracted. A final prerequisite is the presence of minimally a small dictionary of the target language.

The model would then be trained using the sister languages as input, and the target language list as a target output. After training confirms a reasonable accuracy, the model can then be fed with other known words in the sister language to get a prediction of those words in the target language.

After producing said output, the linguist, or language community, needs to subject the output to a quality control and decide on a series of questions: Do the output patterns match what we know of the target language? Can we assume that these words are cognates in the target language, or is there some evidence that other forms were present? Finally, if this is used by a community to fill in empty patterns in their language, the community needs to decide whether the output is something that the community wants in their language. The algorithm is not infallible, and only proposes. Ultimately, a language community using this tool must make a decision whether to accept or reject the algorithm's recommendations.

7 Conclusions

In this paper, we have shown that NMT frameworks can be used to predict cognates of a target language from cognates of its sister languages. We have further shown that adding or removing input languages has interesting effects on the accuracy of the model. This indicates that we can use additional sister languages to compensate the lack of data in a given situation, though, as demonstrated in the case of Romanian, we cannot blindly add sister languages, nor assume that all additions are equally useful. This might be a promising method for situations where not a lot of data is present, but there are multiple well-documented related languages of the target language.

The next step for this line of research is to move from a proof of concept to an implementation in an actual language revitalization scenario. This is something we are currently working on. A further question that need to be addressed as well, is how well this approach performs with languages that exhibit a different morphology from the Romance languages, such as agglutinative and polysynthetic languages.

All code and data used for this project are open-source and can be found here, in order to reproduce these results.

Something we would like to address in this final paragraphs is that machine learning is a potential *tool*. Like every tool, it has its uses and cases where it is not useful. The decision of using such a tool to expand the lexicon of a language is a decision of that language community, and not of a linguist.

Acknowledgements

The author would like to thank the anonymous reviewers for their comments and give special thanks to Becky Sharp for helping with last minute edits.

References

Yuan Lu Chen. 2018. *Improving Neural Net Machine Translation Systems with Linguistic Information*. Phd thesis, University of Arizona.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Alina Maria Ciobanu and Liviu P. Dinu. 2014. Automatic detection of cognates using orthographic alignment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 99–105, Baltimore, Maryland. Association for Computational Linguistics.

Alina Maria Ciobanu and Liviu P. Dinu. 2018. Ab initio: Automatic Latin proto-word reconstruction. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1604–1614, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Leighton Delgado, Irene Navas, Conor Quinn, Tina Tarrant, Wunetu Tarrant, and Harry Wallace. 2019. Digital documentation training for long island algonquian community language researchers: a new paradigm for community linguistics. Presented at: 51st Algonquian Conference, McGill University, Montréal, QC, 24-27 October.

Harald Hammarström, Robert Forkel, Martin Haspelmath, and Sebastian Bank. 2020. *Glottolog 4.3*. Jena.

Fabian Hirschmann, Jinseok Nam, and Johannes Fürnkranz. 2016. What makes word-level neural machine translation hard: A case study on English-German translation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3199–3208, Osaka, Japan. The COLING 2016 Organizing Committee.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9.

Siva Kaylan and Alexandre François. 2019. Freeing the comparative method from the tree model: A framework for historical glottometry. In *Let's talk about trees: Genetic relationships of languages and their phylogenetic representation*. Cambridge University Press, online.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017a. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Jared Klein, Brian Joseph, and Matthias Fritz. 2017b. *Handbook of Comparative and Historical Indo-European Linguistics : An International Handbook*. De Gruyter, Berlin.

Wesley Y. Leonhard. 2007. *Miami Language Reclamation in the Home: A Case Study*. Phd thesis, University of California, Berkeley.

Johann-Mattis List, Philippe Lopez, and Eric Bapteste. 2016. Using sequence similarity networks to identify partial cognates in multilingual wordlists. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 599–605.

M. Maiden, J. Smith, and A. Ledgeway, editors. 2013. *The Cambridge History of the Romance Languages*, volume 2. Cambridge University Press, Cambridge.

Teresa L. McCarty and Sheilah E. Nicholas. 2014. Reclaiming indigenous languages: A reconsideration of the roles and responsibilities of schools. *Review of Research in Education*, 31.

Antoine Meillet. 1967. *The comparative method in historical linguistics*. Librairie Honoré Champion, Paris.

Carlo Meloni, Shauli Ravfogel, and Yoav Goldberg. 2019. Ab antiquo: Proto-language reconstruction with rnns.

Graham Neubig. 2017. Neural machine translation and sequence-to-sequence models: A tutorial. *CoRR*, abs/1703.01619.

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.

Taraka Rama, Johann-Mattis List, Johannes Wahle, and Gerhard Jäger. 2018. Are automatic methods for cognate detection good enough for phylogenetic reconstruction in historical linguistics? *CoRR*, abs/1804.05416.