Word prediction in computational historical linguistics

Peter Dekker[1] *and Willem Zuidema*[2]
[1] Vrije Universiteit Brussel
[2] University of Amsterdam

ABSTRACT

In this paper, we investigate how the prediction paradigm from machine learning and Natural Language Processing (NLP) can be put to use in computational historical linguistics. We propose *word prediction* as an intermediate task, where the forms of unseen words in some target language are predicted from the forms of the corresponding words in a source language. Word prediction allows us to develop algorithms for phylogenetic tree reconstruction, sound correspondence identification and cognate detection, in ways close to attested methods for linguistic reconstruction. We will discuss different factors, such as data representation and the choice of machine learning model, that have to be taken into account when applying prediction methods in historical linguistics. We present our own implementations and evaluate them on different tasks in historical linguistics.

Keywords: computational historical linguistics, machine learning, deep learning

INTRODUCTION 1

How are the languages of the world related and how have they evolved? This is the central question in one of the oldest linguistic disciplines: *historical linguistics*. In this paper, we aim to contribute to

answering these questions using some of the newest methods: computational modelling, machine learning and big data.[1]

Our work can be seen as part of what has been called the *quantitative turn in historical linguistics*: computational methods have been applied to automate parts of the workflow of historical linguistics (Jäger and List 2016), which, in part, has become possible due to the increased availability of digital datasets (the new Cross Linguistic Data Formats initiative proposes new standards for a unified representation of cross-linguistic data, Forkel *et al.* (2018), enabling further expansion and connection of datasets in the coming years).

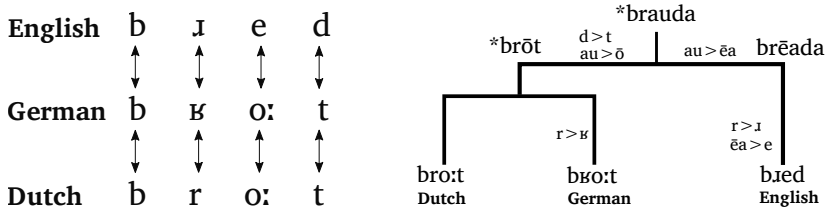
Such large datasets provide new possibilities, but are at the same time too large to be processed by human experts. Research in computational historical linguistics has therefore attempted to automate several tasks in historical linguistics. Different approaches have been applied to *cognate detection* – the task to detect ancestrally related words (cognates) in different languages – (Inkpen *et al.* 2005; List 2012; Rama 2016; Jäger *et al.* 2017; Dellert 2018), inference of sound correspondences (Hruschka *et al.* 2015), protoform reconstruction (Bouchard-Côté *et al.* 2013) and phylogenetic tree reconstruction (Jäger 2015; Chang *et al.* 2015).

These computational methods have thus opened up many new research directions, and, arguably, provide better replicability than manual methods because of the inherent necessity to specify formal guidelines (Jäger 2019). In recent years, studies in computational historical linguistics have drawn much attention, but also sparked much controversy. Examples are Gray and Atkinson (2003), which charted the age of Indo-European languages, and Bouckaert *et al.* (2012), which proposed to map the Indo-European homeland to Anatolia.

1.1 *The comparative method*

The relation between computational methods and more traditional methods is, however, not always straightforward, and differs in more dimensions than just mathematical formalization. Some computational methods stay conceptually closer to the standard methodology

[1] This paper is based on the first author's unpublished MSc thesis (Dekker 2018).



(a) Phenotypic: comparisons are made between the phonemes in aligned word forms

(b) Genotypic: word forms are compared based on a model of the complete ancestral history of the languages, revolving around sound correspondences

Figure 1: Schematic visualization of the difference between phenotypic and genotypic methods, when comparing the English, German and Dutch word forms for the concept *bread*

in historical linguistics: the 'comparative method' (Clackson 2007). Linguists applying the comparative method typically make use of a fixed, basic vocabulary, look at the relevant phonetic forms, and focus on *cognates* (words that are ancestrally related). Based on these cognates, they can then identify sound correspondences and, using regular sound correspondences as criterion for descent, reconstruct the ancestral tree of a particular language family, distinguishing between genetic relationships and borrowing.

Although computational methods usually also employ some form of the comparative method, they mostly focus on what can be called *phenotypic* similarity rather than *genotypic* similarity (Lass 1997; List 2012). *Genotypic* methods compare languages based on the language-specific *regular sound correspondences* that can be established between the languages. *Phenotypic* methods compare languages based on the surface forms of words. When comparing words based on surface similarity, it is more difficult to detect the ancestral relatedness of words which underwent much phonetic change and it is more challenging to detect borrowings. Figure 1 shows the difference between phenotypic and genotypic methods schematically.

Genotypic methods are thus preferable to reliably determine ancestral relationship. However, many genotypic methods, like the successful Bayesian MCMC methods for phylogenetic reconstruction, require cognate judgments as input. These cognate judgments have to be performed by human experts, re-introducing human labour, and therefore limiting the amount of data that can be processed. Alterna-

tively, cognate judgments can be automatically inferred via cognate detection methods that are currently developed, but this adds another step to the reconstruction process, with its own inaccuracies.

1.2 *Word prediction*

An ideal computational method in historical linguistics would automate as much of the process as possible, while still staying close to the comparative method. In this paper, we investigate the usefulness of *word prediction* as an intermediate task that may allow us to arrive at computational methods in historical linguistics. The use of word prediction in historical linguistics was first proposed in the first author's master's thesis (Dekker 2018) and independently by Ciobanu and Dinu (2018), followed by recent approaches (List 2019a; Meloni *et al.* 2019; Cathcart and Wandl 2020; Cathcart and Rama 2020; Fourrier and Sagot 2020a). Word prediction is a methodology that enables the use of surface word forms as data (like phenotypic methods), while still capturing the genetic signal through sound correspondences (like genotypic methods), thus allowing for reliable reconstructions of language relationship based on large amounts of data.

Word prediction allows us to rephrase the reconstruction of language ancestry as a machine learning problem. A machine learning model is trained on pairs of phonetic word forms $(w_{c,A}, w_{c,B})$ denoting the same concept c in two languages A and B. By learning the sound correspondences between the two languages, the model can then predict, for a concept d, the unseen word form $w_{d,B}$, given a word form $w_{d,A}$. Based on the training data, the model learns sound correspondences between the two languages. Therefore, the error between the predicted word and the target word in the test set, for a given source word, provides a distance between source word and target word which is informed by sound correspondences. This contrasts with directly comparing the source and target word – as phenotypic methods do – which yields distances not informed by sound correspondences.

In the rest of this paper, we investigate what is required to successfully apply word prediction in the area of historical linguistics. Our main research questions are the following:

1. What are *suitable data* from historical linguistics and what is a good *representation of input data*, in order to be processed by a machine learning algorithm for the word prediction task?
2. *Which machine learning model* can be used to perform word prediction? A model should be able to learn the relationship between consecutive phonemes (sounds) in a word, and the sound correspondences between source and target word.

In the next sections, we will try to answer these questions, where possible by evaluating multiple alternative solutions. We would like to provide general lessons on factors enabling the use of word prediction in historical linguistics. After these sections, we will develop our own models, based on the answers to the questions we find, and report on the results on different applications in historical linguistics. We end the paper with a discussion of related work and some reflections on the potential and the limitations of word prediction.

DATA 2

Our first question is: which data, and in which representation, are suitable for word prediction? We will first describe which type of data is suitable for this task, and then review different encodings to represent the data.

Datasets 2.1

Data from many linguistic levels can be used to study language change, including lexical, phonetic, morphological and syntactic data. Using word forms (in orthographic or phonetic representation) seems suitable for the prediction task. There are many training examples (words) available per language and the prediction algorithm can generalize over the relations between phonemes. Word forms also have a lower probability of being borrowed or being similar by chance than syntactic data (Greenhill *et al.* 2017). The benefit of phonetic word forms

over orthographic forms is that phonetic forms stay closer to the actual use of language by speakers. Word forms in orthographic representation depend on conventions: the same sound can be described by different letters in different languages.

Ciobanu and Dinu (2018) evaluate their model on three different datasets of orthographic forms for multiple Romance languages (Spanish, Italian, Portuguese, French, Romanian, Latin). In all datasets, the word forms have been grouped into cognate sets: this is needed because the model takes pairs of cognate words as input. The datasets are taken from Bouchard-Côté *et al.* (2007) (585 cognate sets), Reinheimer Ripeanu (2001) (1102 cognate sets) and Ciobanu and Dinu (2014) (3218 cognate sets). Meloni *et al.* (2019) use the Romance cognate dataset from Ciobanu and Dinu (2014) as a basis and augment it with word forms from Wiktionary, arriving at a total of 8799 cognate sets. The authors perform experiments on both orthographic and phonetic word forms. The phonetic word forms are acquired by running a computational transcription library on the orthographic word forms. Cathcart and Wandl (2020) base their dataset on an etymological dictionary by Derksen (2007), and extract Slavic proto-words and their accompanying (cognate) contemporary words in 13 Slavic languages, yielding a dataset of 11400 forms. Fourrier and Sagot (2020a) use phonetic cognate data from Latin, Spanish and Italian, originating from an etymological database (Fourrier and Sagot 2020b). Fourrier (2020) applies the same workflow, but uses data from Polish, Czech, Lithuanian and Italian.

In our own experiments, we use the NorthEuraLex dataset (Dellert *et al.* 2019),[2] which consists of phonetic word forms for 1016 concepts in 107 languages in Northern Eurasia. The languages in the dataset belong to many language families (among others Uralic, Indo-European, Turkic and Mongolic), so the number of cognate sets has to be calculated per language family. The size of the dataset is therefore not directly comparable to the size of datasets in Ciobanu and Dinu (2018) and Meloni *et al.* (2019). We use a larger dataset than generally used in historical linguistics. Usually, only basic vocabulary (e.g. kinship terms, body parts) is taken into account because this vocabulary is

[2] Available for download from `http://northeuralex.org/`. We used the 0.9 release. As of the 0.9.2 release, the dataset contains 30 more languages.

least prone to borrowing (Campbell 2013, p. 352). However, machine learning algorithms need a large number of examples to train on and a meaningful number of examples to evaluate the algorithm. We hope that increasing the performance of the algorithm by using enough training examples compensates for the possible performance decrease caused by borrowing. We use a version of the dataset which is formatted in the *ASJPcode* alphabet (Brown *et al.* 2008). ASJPcode consists of 41 sound classes, considerably less than the number of IPA phonemes, reducing the complexity of the prediction problem. For clarity, in this paper, we converted all ASJP forms to IPA using the `pyclts` library (Anderson *et al.* 2018).[3] As ASJP characters represent broader classes of phonetic features than IPA phonemes, the shown IPA phonemes may differ from the original phonemes used in the words.

There can be multiple word forms for a concept in one language. Per language pair, we create word pairs by taking the Cartesian product of all alternative word forms for one concept in both languages. No word pairs are created across concepts. For example, if there are 2 alternative word forms for a concept in language A, and 3 alternative forms for that concept in language B, this yields a total of 6 word pairs. We then split the dataset into a training set (80%), development set (10%) and test set (10%). The training and test set should be separated, so the model predicts on different data than it learned from. The development set is used to tune model parameters (see Section 4).

2.2 *Data representation*

To enable a machine learning algorithm to process the phonetic data, every phoneme has to be encoded as a numerical vector. We will consider three types of encoding: *one-hot*, *phonetic* and *embedding* encoding.

Ciobanu and Dinu (2018) do not describe their encoding of the data. They do however perform a number of pre-processing steps. First, the word forms of a word pair are aligned using Needleman-Wunsch alignment (Needleman and Wunsch 1970). Subsequently, characters in the output word which remain the same as in the input words, are represented by a special character.

[3] `https://github.com/cldf-clts/pyclts`

2.2.1 One-hot

In *one-hot encoding*, every phoneme is represented by a vector of length $n_{characters}$, with a 1 at the position which corresponds to the current character, and 0 at all other positions. No qualitative information about the phoneme is stored. Cathcart and Wandl (2020) encode every phoneme of an input word using one-hot encoding, but extends this with a learned embedding per language. Table 1 gives an example of a one-hot feature matrix.

Table 1: Example of feature matrix for one-hot encoding, for an alphabet consisting of four phonemes. Every phoneme is represented by one feature that is turned on, that feature is unique for that phoneme

IPA				
p	1	0	0	0
b	0	1	0	0
f	0	0	1	0
v	0	0	0	1

2.2.2 Phonetic

In *phonetic* encoding, a phoneme is encoded as a vector of its phonetic features (e.g. back, bilabial, voiced), enabling the model to generalize observed sound changes across different phonemes. Rama (2016), using a neural network approach to cognate detection, shows that a phonetic representation yields better performance than one-hot encoding for some datasets. In our model, we use the phonetic feature matrix for ASJP tokens from Brown *et al.* (2008), formatted as a binary feature matrix by Rama (2016). As mentioned, in this paper, we use IPA to denote the ASJP tokens. Table 2 shows an example of a phonetic feature matrix.

Table 2: Example of feature matrix for phonetic encoding: every phoneme can have multiple features turned on

IPA	Voiced	Labial	Denta	Alveolar	⋯
p	0	1	0	0	⋯
b	1	1	0	0	⋯
f	0	1	1	0	⋯
v	1	1	1	0	⋯
m	1	1	0	0	⋯
θ	1	0	1	0	⋯

Embedding 2.2.3

A third type of encoding that we will consider is the *embedding* encoding, where a linguistic item is encoded using the distribution of items appearing in its context. Most well known are *word embeddings*, which have successfully been applied in many NLP tasks (e.g., Mikolov *et al.* 2013; Pennington *et al.* 2014). The assumption is that "you shall know a word by the company it keeps" (Firth 1957). If two words have a similar embedding vector, they usually appear in the same context and can thus relatively easily be interchanged. But the idea of using embeddings that reflect the context of a linguistic item can also be applied analyzing smaller units than the word level. For instance, it has also succesfully been applied in NLP by introducing character-based language models (Kim *et al.* 2016).

In computational historical linguistics, Rama and List (2019) used skip-grams, which capture the context of a phoneme, to perform fast cognate detection. Meloni *et al.* (2019) use an embedding layer in their neural network, which learns an embedding vector of size 100 for every phoneme, from the data. The embedding consists of a language-specific and a language-dependent part. Cathcart and Wandl (2020) use one-hot encoding for phonemes of the input word, and concatenate this with a trained embedding per language.

Similarly, we propose to encode a phoneme as a vector of the phonemes occurring in its context. The same interchangeability of word embeddings is assumed: if two phoneme vectors are similar, they appear in a similar context. This corresponds to language-specific rules in *phonotactics* (the study of the combination of phonemes), which specify that a certain class of phonemes (e.g. approximant) can follow a certain other class (e.g. voiceless fricative). It can be expected that embeddings of phonemes inside a certain class are more likely to be similar to each other than to phonemes in other classes. In some respects, the embedding encoding learns the same feature matrix as the *phonetic* encoding, but inferred from the data, and with more emphasis on language-specific phonotactics.

In our experiments, we also use embedding coding, but do not apply high-dimensional learned embeddings as do Meloni *et al.* (2019). Instead, we want to put more emphasis on the direct neighbours of a phoneme, as most phonotactic rules describe these relations.

Table 3: Example of feature matrix for embedding encoding: every phoneme is represented by an array of floating point values, which correspond to the probabilities that other phonemes occur before or after this phoneme. The values in a row sum to 1

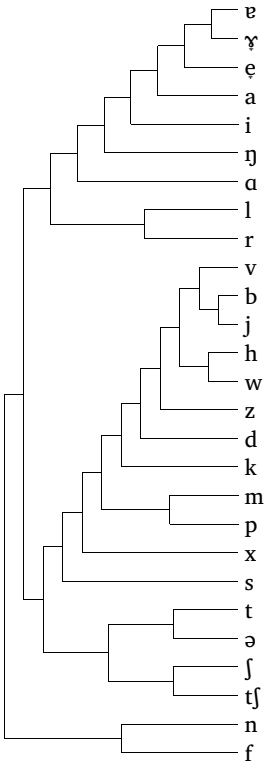
IPA	START	*i* LEFT	*S* LEFT	*p* RIGHT	⋯
ə	0.004	0.003	0.001	0.002	⋯
a	0.024	0.000	0.000	0.003	⋯
ɐ	0.050	0.002	0.000	0.012	⋯
b	0.388	0.000	0.000	0.004	⋯
p	0.152	0.039	0.000	0.000	⋯

We create language-specific embedding encodings from the whole NorthEuraLex corpus. For every phoneme, the preceding and following phonemes, for all occurrences of the phoneme in the corpus, are counted. Position is taken into account, i.e., an /a/ appearing before a certain phoneme is counted separately from an /a/ appearing after a certain phoneme. Start and end tokens, for phonemes at the start and end of a word, are also counted. After collecting the counts, the values are normalized per row, so all the features for a phoneme sum to 1. Table 3 shows an example of an embedding feature matrix.

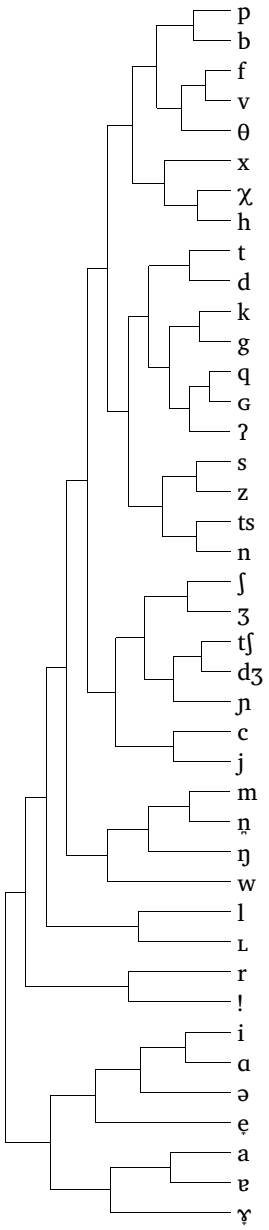
2.2.4 Visualization of embedding

Following Meloni *et al.* (2019), to analyze what representation of the phonetic space the embedding encoding learns, we performed hierarchical clustering on embeddings. We computed pairwise euclidean distances between phonemes for the embedding matrix learned from the Dutch portion of the NorthEuraLex dataset and for the phonetic feature matrix from Brown *et al.* (2008). Hierarchical clustering was performed on the distance matrices using neighbour joining (Saitou and Nei 1987). Figure 2 shows the results.

The figure shows that the embedding method groups most vowels together, but also adds some consonants to this group, and places the vowel /ə/ in another group. When looking at the groupings in the embedding encoding, it seems the embedding encoding mainly represents phonemes by their place of articulation rather than by their manner of articulation. /p/ and /m/ are grouped together, both bilabial, but one is a stop or fricative, the other a nasal. /l/ and /r/ are grouped, which are both (apico-)alveolar, but one is an approximant, the other a trill. Groupings on manner, like the stops /t/ and /d/ in the phonetic encoding, are less visible in the embedding encoding.



(a) Clustering of Dutch learned embeddings from NorthEuraLex



(b) Clustering of phonetic feature matrix from Brown *et al.* (2008)

Figure 2: Hierarchical clusterings, using euclidean distance and neighbour joining, of NorthEuralex learned phoneme embedding for Dutch and the phonetic feature matrix from Brown *et al.* (2008)

Looking at the data, it becomes visible why close phonemes like /d/ and /t/ have more remote embeddings. /d/ occurs in 206 word forms, /t/ in 402 word forms. The position in the words of the two phonemes is quite different. The most frequent position for /d/ (30%) is the first position, in words like: /dɣ̞ktər/ “doctor”, /dɐk/ “roof” and /dẹ̞kə/ “blanket”. For /t/, the most frequent position is the 4th position (34%), in words like: /ɐfstɐnt/ “distance”, /lẹ̞ftait/ “age” and /hɣ̞xtə/ “height”. This different position of the phonemes in word forms in the corpus can lead to representations in embedding encoding which are not close to each other.

2.2.5 Evaluation of input encodings

With the simple one-hot encoding, the time-tested phonetic feature encoding and the novel embedding encoding, we now have three different ways to represent linguistic items. How well suited are each of these encoding styles for the task of word prediction? We evaluate the three input encodings in combination with two machine learning models (that will be introduced in Section 3): the encoder-decoder and the structured perceptron. Table 4 shows the average word prediction distance over two language families, for the different parameter settings, on the test set. Although the differences in word prediction distance are small, the embedding encoding tends to work best in most test cases. For the Germanic language family, one-hot encoding works slightly better than embedding encoding, but the difference is minimal.

Table 4: Evaluation of different data encodings, on two models, by word prediction distance (edit distance between prediction and target) for two language families: Slavic and Germanic. The distance is the mean of the distance of all language pairs in the family. Lower distance means better prediction

Method		Language family	
Model	Input encoding	Slavic	Germanic
Enc-dec	One-hot	0.5582	0.5721
Enc-dec	Phonetic	0.5767	0.5853
Enc-dec	Embedding	**0.5579**	**0.5710**
Struct perc	One-hot	0.3436	**0.4374**
Struct perc	Phonetic	0.3465	0.4497
Struct perc	Embedding	**0.3423**	0.4375

MACHINE LEARNING MODEL 3

Our second question is: which machine learning models are suitable to perform the task of word prediction? We need a model that can convert sequential input (the source word) to sequential output (the target word). We would like to apply an algorithm that can model the sequential dependencies between consecutive phonemes in a word. We evaluate two of these sequence models, which are both prototypical for a larger class of models: a simple *structured perceptron* (probabilistic sequence model) and a more complex *RNN encoder-decoder* (deep neural network). Furthermore, we will introduce two baseline models, to compare performance.

Structured perceptron 3.1

We will look at the *structured perceptron* (Collins 2002; Daume and Marcu 2006), an example of a probabilistic sequence model, which among others has been applied to part-of-speech tagging. The structured perceptron is an extension of a *perceptron* (one-layer neural network) (Rosenblatt 1958) for performing sequential tasks. In this paper, we will evaluate this as one of the models to predict words between languages.

The structured perceptron algorithm is run for I iterations. At every iteration, all N data points are processed. For every input sequence (word, in this case) x, a sequence $\hat{y}$ is predicted, based on the current model parameters $\boldsymbol{w}$:

(1) $$\hat{y} = argmax_{u \in Y} \boldsymbol{w}^T \boldsymbol{\phi}(x, u)$$

By the *argmax*, the feature function $\boldsymbol{\phi}$ has to be evaluated for all possible output sequences $u \in Y$; the value which gives the highest output is used as prediction $\hat{y}$. This *argmax* is computationally expensive, but the Viterbi algorithm (Viterbi 1967) can be run to efficiently estimate the best value $\hat{y}$.

If the predicted sequence $\hat{y}$ is different from the target sequence y', the weights are updated using the difference between the feature func-

tion applied to the target and the feature function applied to the predicted value:

(2) $$\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{\phi}(x, y') - \boldsymbol{\phi}(x, \hat{y})$$

After I iterations, the weights $\boldsymbol{w}$ of the last iteration are returned. In practice, the *averaged structured perceptron* is used, which outputs an average of the weights over all updates.

We use the implementation from the `seqlearn` library.[4] In the experiments, the structured perceptron algorithm is run for 100 iterations of parameter training.

Ciobanu and Dinu (2018) use a Conditional Random Field (Lafferty *et al.* 2001), a structured prediction technique related to the structured perceptron. With this model, predictions between different Romance languages and Latin are made. These pairwise predictions are then ensembled, to arrive at a protoform for Latin.

3.2 *RNN encoder-decoder*

Deep neural networks have shown recent success in multiple tasks in NLP, like machine translation, using different model architectures, such as the encoder-decoder (Sutskever *et al.* 2014; Cho *et al.* 2014), attention-based models (Bahdanau *et al.* 2014) and transformers (Vaswani *et al.* 2017; Devlin *et al.* 2019). Meloni *et al.* (2019) use an encoder-decoder structure, but add an attention layer, which allows for focusing on segments of the input word that are useful for predicting the target word. The authors would like to predict a Latin word form from a number of contemporary Romance languages. In order to do this, the encoder accepts multiple inputs, one for every language. Fourrier and Sagot (2020a) use a multiway encoder-decoder with attention. In this architecture, there is one model for all language pairs, with a separate encoder per source language and a separate decoder per target language. Cathcart and Wandl (2020) apply an encoder-decoder with a specific type of attention, 0th order hard monotonic attention (Wu and Cotterell 2019). The task is to predict contemporary word forms from proto-Slavic word forms. This is done using one

[4] `https://github.com/larsmans/seqlearn`

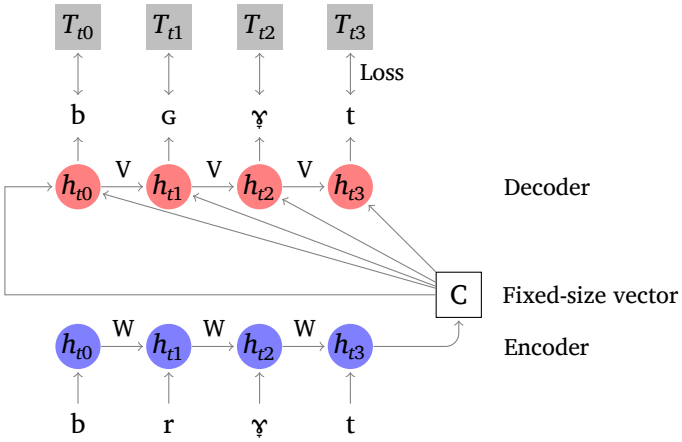


Figure 3:
Structure of our RNN encoder-decoder model

encoder-decoder model for all languages, where every input word is paired with a language-specific embedding. The language-specific embedding is a straight-through embedding (Bengio *et al.* 2013; Courbariaux *et al.* 2016), which has a discrete representaton at prediction time, but also a continuous representation to calculate loss. This enables the interpretation of the neural network as a latent variable model.

In this paper, we consider a relatively simple model, a recurrent neural network (RNN) in encoder-decoder structure, as representant for the class of deep neural networks. A RNN takes a sequence as input and produces a sequence. RNNs are good at handling sequential information because the output of a recurrent node depends on both the input at the current time step (the phoneme at the current position) and on the values of the previous recurrent node, carrying a representation of previous phonemes in the word. An encoder-decoder model consists of two RNNs, see Figure 3. This architecture enables the use of different source and target lengths and outputs a phoneme based on the whole input string.

In our approach, we use a vanilla RNN encoder-decoder, without attention. We evaluated different architectures and parameter settings in preliminary experiments, and picked the best performing (Section 4.1). As recurrent network nodes, we use Gated Recurrent Units (GRU) (Cho *et al.* 2014), which are capable of capturing long-distance dependencies. The GRU is an adaptation of the Long Short

Term Memory (LSTM) unit (Hochreiter and Schmidhuber 1997). Both the encoder and decoder consist of 400 hidden units. The output from the last time step of the encoder is used, a fixed-size (independent of input length) vector. We apply a bidirectional encoder architecture, running a forward-direction and a backward-direction encoder on the input, and combining the output using a dense layer, to reduce it to the dimensionality of the output of a single encoder. The fixed-size vector, which contains information of the forward and backward pass, is then fed to the decoder at every time step.

Because we use one-hot output encoding, predicting a phoneme corresponds to single-label classification: only one element of the vector can be 1. Therefore, the output layer of the network is a *softmax* layer, which outputs a probability distribution over the possible one-hot positions, corresponding to phonemes. The network outputs are compared to the target values using a *categorical cross-entropy* loss function, which is known to work together well with *softmax* output. We add an *L2 regularization term* to the loss function, which penalizes large weight values, to prevent overfitting on the training data.

To give an impression of the degrees of freedom the network has when learning correspondences, we give an estimation of the number of weights in the network. The weights consist of the weights in the encoders, the decoder, the dense encoder concatenation layer and the dense output layer. The number of weights is dependent on the number of features in the encoding chosen for the input and target language, and on the maximum length of words in the languages. For the language pair Dutch-German, with embedding input encoding, there is a total of 2.4 million weights.

The weights of the network are initialized using *Xavier initialization* (Glorot and Bengio 2010). With the right initialization, the network can be trained faster because the incoming data fits better to the activation functions of the layers. We apply dropout, the random disabling of network nodes to prevent overfitting to training data; the dropout factor is 0.1. Data is supplied in batches, the default batch size is 10. The applied optimization algorithm is *Adagrad* (Duchi *et al.* 2011): this is an algorithm to update the weights with the gradient of the loss, using an adaptive learning rate. The initial learning rate is 0.01. The threshold for gradient clipping is set to 100. In the experiments, the default number of training epochs, the number of times

the training set is run through, is 15. The network was implemented using the `Lasagne` neural network library (Dieleman *et al.* 2015).

Baseline models 3.3

The prediction results are compared to two baselines, for which the distance between target and baseline is calculated. The first baseline is the trivial *source prediction* baseline, predicting exactly the source word.

The second baseline is based on *Pointwise Mutual Information (PMI)* (Church and Hanks 1990; Wieling *et al.* 2009; Jäger 2014; Jäger *et al.* 2017). PMI similarity between words gives high similarity to words with phonemes that are often aligned to each other in the languages. In general, PMI gives higher similarity to words which are cognate, which are predictable through regular sound correspondences (Jäger and Sofroniev 2016). Following the approach in Jäger *et al.* (2017), PMI scores between phonemes are calculated by aligning all words for a language pair in the training set with each other, using Needleman-Wunsch alignment (NW) (Needleman and Wunsch 1970). We perform 50 iterations of NW alignment. At every NW iteration, the weights that determine the match between phonemes, are determined by the PMI scores of the previous iteration. At prediction time, the alignment of the last training iteration is used. For every source phoneme, the target phoneme with the highest probability of being aligned to the source phoneme is predicted. The internal table of PMI scores is essentially a table of sound correspondences the model learns.

Evaluation of machine learning models 3.4

Table 5 shows the results for the two machine learning models, and two baseline models, evaluated on the test set. It can be observed that the structured perceptron, in spite of its simpler structure, performs better than the encoder-decoder. For the Germanic language family, the structured perceptron also performs better than the PMI-based baseline model. For the Slavic language model, the PMI-based baseline works better. This difference may be explained by the fact

Table 5: Evaluation of machine learning models and baseline models, with one-hot input data, evaluated on word prediction distance (edit distance between prediction and target) for different test conditions, for two language families: Slavic and Germanic. The distance is the mean of the distance of all language pairs in the family. Lower distance means better prediction

Method	Language family	
Model	Slavic	Germanic
Encoder-decoder	0.5582	0.5721
Structured perceptron	0.3436	**0.4374**
Source prediction baseline	0.3714	0.4933
PMI-based baseline	**0.3249**	0.4520

that the Slavic languages in the dataset are more closely related and therefore easier for the baseline model to predict correctly, as smaller changes have to be made to the source words.

4 IMPLEMENTATION DETAILS

Having discussed the different choices for dataset, data representation and machine learning model for word prediction, we will discuss how useful word prediction is in algorithms for computational historical linguistics. We will first, however, briefly describe some of the implementation details needed to get the word prediction models to produce optimal results.

4.1 *Parameter optimization*

In preliminary experiments, we tested the different models with a range of different parameters and evaluated on a development set. The parameter settings with the highest performance on the development set, were then used for the experiments reported in this paper, in the previous two sections. For the experiments on data encoding and machine learning models in the two previous sections, we used the test set because we regard these as model evaluation, not as parameter tuning. For the experiments in the next sections, on applications, we also use the test set.

Training 4.2

Training is performed on the full training set per language pair, which consists of both cognate and non-cognate words. It would be easier for the model to learn sound correspondences if it would only receive cognate training examples. However, we want to develop a model that can be applied to problems where no cognate judgments are available.

Data preparation 4.3

For the structured perceptron, the input and output word must have the same length, but the lengths of words in a language pair may differ. For this model, the input and output word are matched in terms of length, by padding the shortest word at the end with dummy symbols (.). For the encoder-decoder, all words in a language must have the same fixed length, to fit the fixed shape of the layers, but the input language may have a different fixed length than the output language. To prepare the data for this model, maximum lengths per language are calculated, and words are padded with dummy symbols (.) at the end to match the maximum length.

For the target data for the encoder-decoder model, one-hot encoding is used regardless of the input encoding. This means that target words are encoded in one-hot encoding and the algorithm will output predictions in one-hot encoding. One-hot output encoding facilitates convenient decoding of the predictions. Other output encodings did not show good results in preliminary experiments. As a target for the structured perceptron model, unencoded data is supplied, since this is a format suitable for the used implementation of the algorithm.

The training data is standardized in order to fit it better to the activation functions of the neural network nodes. For the training data, the mean and standard deviation per feature are calculated over the whole training set for this language pair. The mean is subtracted from the data and the resulting value is divided by the standard deviation. After standardization, per feature, the standardized training data has a mean of 0 and a standard deviation of 1. The test data is standardized using the mean and standard deviation of the training data. This transfer of knowledge can be regarded as being part of the training procedure.

4.4 Quantitative evaluation of word prediction

We evaluate the models by comparing the predictions and targets on the test section of the dataset. We only evaluate on cognate pairs of words. If words are not genetically related, the algorithm will not be able to predict this word via regular sound correspondences. Cognate judgments from the IELex dataset are used.[5] For words in NorthEuraLex for which no IELex cognate judgments are available, LexStat (List 2012) automatic cognate judgments are generated (threshold 0.6).

Languages which are not closely related do not share many cognates. Because we only evaluate on cognate words, the test set for those language pairs will become too small. To alleviate this problem, we evaluate only on groups of more closely related languages. In these groups, every language in the group shares at least *n* cognates with all other languages. We determine these groups, by generating a graph of all languages where two languages are connected if and only if the number of shared cognates exceeds the threshold *n*. Then, we determine the *maximal cliques* in this graph: groups of nodes where all nodes are connected to each other and it is not possible to add another node that is connected to all existing nodes. These *maximal cliques* correspond to our definition of language groups which share *n* cognates. The largest cliques were the Slavic (Czech, Bulgarian, Russian, Belarusian, Ukrainian, Polish, Slovak, Slovenian, Croatian) and Germanic (Swedish, Icelandic, English, Dutch, German, Danish, Norwegian) subfamilies, which we use for our experiments. The distance metric used between target and prediction is *normalized edit distance*: *Levenshtein distance* (Levenshtein 1966) (or: edit distance) divided by the length of the longest sequence. An average of this distance metric, over all words in the test set, is used as the distance between two languages.

We use the prediction distances for two purposes: to determine the distance of languages to each other and to determine the general accuracy of a model. If a certain model has a lower prediction distance

[5] An intersection, which applied the IELex cognate judgments to the NorthEuraLex dataset, was supplied by Gerhard Jäger.

over all language pairs than another model, we consider it to be more accurate.

Qualitative evaluation of word prediction 4.5

Using the best parameters determined in the preceding sections, it is worth looking at some qualitative examples of the predictions the algorithms make. Table 6 shows the output of the word prediction algo-

Input	Target	Prediction	Distance
starktə	ʃtarkə	ʃtarkən	0.14
krais	kɢɣits	kɢɐiʃ	0.40
brɑr	bɢɑdɐ	bɢɑr	0.40
mɣrxə	mɣrgən	mɣrgə	0.17
sɐmə	tsɑzɐmən	ʃɐmə	0.57
varxɐn	fargɸən	fargən	0.14
klimə	klatɐn	klimə	0.67
bindəl	bindəl	bəndəl	0.17
linkər	liŋkɐ	liŋkən	0.33
wɣnə	vɣnən	vɑn	0.60
sxəlt	ʃɑlt	ʃɢəlt	0.40
brɐndə	bɢanən	bɢɐndə	0.50
ski	ski	ʃən	1.00
zikzain	kɢɐŋkzɐin	ziʃzɐin	0.56
drɐin	dɢɸən	dɢɐin	0.40
rɸxə	ɢɸgən	ɢɸgə	0.20
sidərə	tsitɐn	ʃidəɢə	0.83
halft	halftə	halft	0.17
ɣvarwinə	zigən	ɑfarviən	0.75
fɸrtəx	firtsiʃ	fartsən	0.50
tɸkə	tsɐiʃən	tsɸən	0.50
nɐkt	nɐkt	nɐit	0.25
hɣŋɸrix	hɑŋɢiʃ	hɑŋɸɢiʃ	0.14
wraivə	ɢɐibən	vɢɐibə	0.33
zɣndə	zində	zɣndə	0.20
zixvarzɐmələ	ziʃfarzɐməln	ziʃfarzɐmələ	0.08
hɣŋər	hɑŋɐ	hɑŋən	0.40
zɣmər	zɣmɐ	zɣmən	0.40
hɐrt	harts	hɐrt	0.50
bədrixə	bətɢigən	bədɢigə	0.25

Table 6:
Word prediction output for a structured perceptron (embedding encoding) on language pair Dutch-German. *Prediction* is the German word predicted by the model when *Input* is given as Dutch input. The edit distance between the prediction and the *target* German word, which is not seen by the model, is calculated. Lower distance is better performance

rithm for a structured perceptron model on the language pair Dutch-German. For every word, a prediction distance is calculated. This distance per word will be used for cognate detection in Subsection 5.3. From these word distances, a mean distance per language pair is calculated. This will be used for the application of phylogenetic tree reconstruction in Subsection 5.1. When again taking the mean of the scores of all language pairs in a family, one could get a score which represents the performance of a model on a language family. These distances were used in the previous sections to evaluate different parameter settings.

5 APPLICATIONS

After we looked at appropriate data and a machine learning model for word prediction, it is now time to discuss a number of applications of word prediction in historical linguistics: *phylogenetic tree reconstruction*, *sound correspondence identification* and *cognate detection*. The applications use the outcomes of word prediction as a basis. After describing each application, we will evaluate the results.

5.1 *Phylogenetic tree reconstruction*

We regard the prediction score between language pairs as a measure of ancestral relatedness and use these scores to reconstruct a phylogenetic tree (see Section 6.3 for a further discussion). We perform hierarchical clustering on the matrix of edit distances for all language pairs, using the *UPGMA* (Sokal and Michener 1958) and *neighbour joining* (Saitou and Nei 1987) algorithms, implemented in the `LingPy` library (List *et al.* 2019). The generated trees are then compared to reference trees from Glottolog (Hammarström *et al.* 2020), based on current insights in historical linguistics. Evaluation is performed using Generalized Quartet Distance (Pompei *et al.* 2011), a generalization of Quartet Distance (Bryant *et al.* 2000) to non-binary trees. We apply the algorithm as implemented in the `QDist` software package (Mailund and

Method	Clustering	
Model	UPGMA	Neighbour joining
Struct perc (embedding enc)	**0.047619**	**0.047619**
Source prediction baseline	0.269841	**0.047619**
PMI-based baseline	**0.047619**	**0.047619**

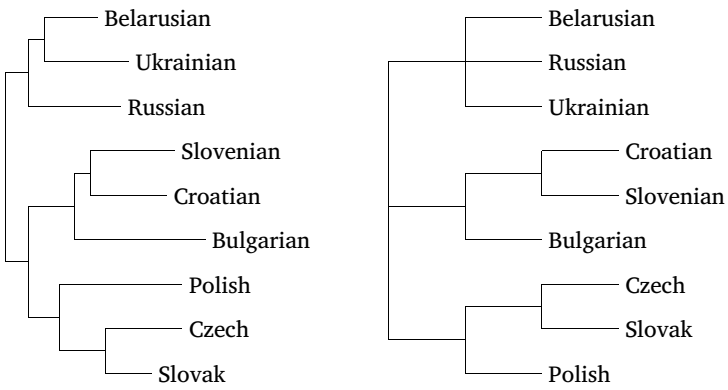
Table 7:
Generalized Quartet distance between trees of the Slavic language family, inferred from word prediction results using the structured perceptron model, and the Glottolog reference tree. Lower is better: a generated tree equal to the reference tree will have a theoretical distance of 0. In this case, the lower bound is 0.047619 because the generated binary trees will never precisely match the multiple-branching reference tree

Pedersen 2004). Trees were visualized using the `ete3` library (Huerta-Cepas *et al.* 2016).

This is the first approach to infer phylogenetic trees from prediction distance. Cathcart and Wandl (2020) infer phylogenetic trees, based on the language-specific embedding vectors, which are paired with the input and trained with the model. Distances between embedding vectors are calculated using cosine distance, and the resulting distance matrix is clustered using neighbour joining.

Table 7 shows the generalized Quartet distance between the generated trees, for different conditions, and a Glottolog reference tree. In the table, one could see that the structured perceptron model consistently creates valid trees. The baseline models, especially the PMI model, also create valid trees. The performance differences between models are smaller than the differences for the word prediction task. This is not very surprising, given that phylogenetic tree reconstruction is an easier task than word prediction: there are fewer possible branchings in a tree, than possible combinations of phonemes in a word. Even a model with lower performance on word prediction can generate a relatively good tree.

Figure 4 graphically shows the trees inferred from word prediction using the structured perceptron. The Glottolog reference tree is added for comparison. The perceptron tree receives the lowest possible distance to the reference tree of 0.047619: the generated binary trees will never precisely match the multiple-branching reference tree. This is also the reason why no generated tree reaches the Quartet distance of 0 in Table 7.



(a) Structured perceptron, embedding encoding, NJ clustering

(b) Glottolog

Figure 4: Phylogenetic trees for the Slavic language family, using structured perceptron prediction, and the Glottolog reference tree. Quartet distance between trees: 0.047619

5.2 *Sound correspondence identification*

To be able to make predictions, the word prediction model has to learn the probabilities of phonemes changing into other phonemes, given a certain context. We would like to extract these correspondences from the model. It is challenging to identify specific neural network nodes that fire when a certain sound correspondence is applied. Instead, we estimate the *internal* sound correspondences that the network learned, by looking at the *output*: the substitutions made between the source word and the prediction. Pairs of source and predictions words are aligned using the Needleman-Wunsch algorithm. Then, the pairs of substituted phonemes between these source-prediction alignments can be counted. These can be compared to the counts of substituted phonemes between source and target. In other prediction approaches, sound correspondences are not directly determined. Instead of listing sound correspondences, Meloni *et al.* (2019) and Cathcart and Wandl (2020) make an analysis of the errors that the algorithm makes per phonological phenomenon, evaluated on both real and synthetic data.

We identified sound correspondences between Dutch and German, two closely related Germanic languages. Source-prediction substitutions were extracted using a structured perceptron model, run for

Substitution		Source-prediction frequency	Source-target frequency
r	ɢ	37	32
s	ʃ	27	18
x	g	23	17
v	f	16	11
a	ɐ	16	14
w	v	11	10
–	n	10	55
t	ts	9	7
ɣ̟	ɑ	8	5
r	n	6	1
p	f	6	4
ɐ̯	a	5	4
x	ʃ	5	8
ə	–	4	5
a	ə	4	1
a	i	3	0
i	ə	3	1
n	–	3	0
t	–	3	3
v	b	3	2

Table 8:
Substitutions between aligned source-prediction pairs and substitutions between aligned source-target pairs for Dutch-German word prediction, using a structured perceptron model and embedding encoding. The list is ordered on frequency of source-prediction substitutions, the 20 most frequent entries are shown

the default 100 iterations (Section 3.1). Table 8 shows the most frequent sound substitutions for source-prediction and source-target. It can be observed that the most frequent substitutions between source and prediction are also frequent between source and target. This implies that the model learned meaningful sound correspondences.

Cognate detection 5.3

Cognate detection is the detection of word forms in different languages (usually per concept), which derive from the same ancestral word. In order to perform cognate detection based on word prediction, we

cluster words for the same concept in different languages based on the prediction distances per word.

First, word prediction is performed for all language pairs which we want to evaluate. In the normal word prediction workflow (Section 4.4), predictions are made only on word pairs which are deemed cognates by existing judgments. When performing cognate detection, the whole point is to make these judgments, so we perform word prediction on the full test set: cognates and non-cognates. We take into account concepts for which word forms occur in all languages; this vastly reduces the number of concepts. For every concept, we create a distance matrix between the word forms in all languages, based on the prediction distance *per word*. Next, we run a flat clustering algorithm on this distance matrix. Applied clustering algorithms are *flat UPGMA* (Sokal and Michener 1958), *link clustering* (Ahn *et al.* 2010) and *MCL* (van Dongen 2000), implemented in the `LingPy` library. Preliminary experiments on the development set show that a threshold of $\theta = 0.7$ gives best results for MCL and link clustering, and $\theta = 0.8$ gives best results for flat UPGMA.

Conceptually, the performed cognate detection operation is the same as the phylogenetic tree reconstruction operation, but now we cluster per word, instead of per language, and we perform a flat clustering instead of an hierarchical clustering.

For evaluation, we use cognate judgments from IElex (Dunn 2012). Evaluation is performed using the B-Cubed F measure (Bagga and Baldwin 1998; Amigó *et al.* 2009), implemented in the `bcubed` library.[6] We perform cognate detection for the Slavic and Germanic language families, by clustering words based on word prediction distances. We evaluate performance for the structured perceptron model, compared to the source prediction baseline. During cognate detection, contrary to the default setting, prediction is performed on both cognates and non-cognates. We apply three clustering algorithms: *MCL* ($\theta = 0.7$), *Link clustering* ($\theta = 0.7$) and *Flat UPGMA* ($\theta = 0.8$).

Table 9 shows B-Cubed F scores for cognate detection on the Slavic and Germanic language families. For the Germanic language family, the structured perceptron, using MCL clustering, performs best. For the Slavic language family, the source prediction baseline

[6] `https://github.com/hhromic/python-bcubed`

Model	Cluster algorithm	Language family	
		Slavic	Germanic
Structured perceptron (one-hot)	MCL	0.877458	**0.932077**
Structured perceptron (one-hot)	LC	0.915680	0.905044
Structured perceptron (one-hot)	fUPGMA	0.919739	0.889788
Source prediction	MCL	0.920806	0.851781
Source prediction	LC	0.926061	0.875475
Source prediction	fUPGMA	**0.929840**	0.878705

Table 9:
B-Cubed F scores for cognate detection on the Slavic (27 concepts) and Germanic (29 concepts) language families. MCL = MCL clustering ($\theta = 0.7$), LC = Link Clustering ($\theta = 0.7$), fUPGMA = Flat UPGMA ($\theta = 0.8$). Higher F score means better correspondence between computed and real clustering

model slightly outperforms the structured perceptron. It must be noted that the sample of shared concepts in a language family is small: this makes results less stable.

DISCUSSION 6

Contribution 6.1

In this paper, we evaluated under which conditions the machine learning paradigm, successful in many computing tasks, can be useful in historical linguistics. We proposed the task of *word prediction*: by training a machine learning model on pairs of words in two languages, it learns the sound correspondences between the two languages and should be able to predict unseen words. We regard this as a method that stays close to the central aspects of the traditional comparative method in historical linguistics and is therefore a good candidate for reliable reconstruction of language ancestry. Multiple factors which could lead to an effective use of prediction methods in historical linguistics were evaluated: the choice of machine learning model and encoding of the input data. We evaluated existing models of word prediction (Ciobanu and Dinu 2018; Meloni *et al.* 2019; Cathcart and Wandl 2020; Fourrier

and Sagot 2020a) and came up with our own model, which enables applications on several tasks in historical linguistics. In this paper, we have proposed new approaches for phylogenetic tree reconstruction and cognate detection, based on word prediction error. We evaluated two sequential neural network models, a RNN encoder-decoder and a structured perceptron. To find an appropriate data representation, we evaluated embedding encoding, inspired by word embeddings in natural language processing, and compared its performance to existing one-hot and phonetic encodings. Our results suggest that a simple structured perceptron performs better than a RNN encoder-decoder, and embedding encoding performs slightly better than existing encodings on the prediction task. It should also be noted that one of our baselines, the PMI-based prediction model, performs relatively well, probably because this simple method does have an internal representation of sound correspondences. More research is needed to find the exact model architectures, parameter settings and data encodings to obtain optimal performance in the word prediction tasks. Note that the goal of our current paper is merely exploratory: we explore the conditions under which the prediction paradigm can be used in historical linguistics, and what possible applications of prediction could be in historical linguistics.

6.2 *Related work*

We will now discuss two types of related work. Firstly, we will look at other approaches in computational historical linguistics which try to capture a genotypic relationship. Secondly, we will look at other approaches which use the concept of *word prediction*, in multiple contexts.

6.2.1 Genotypic methods in computational historical linguistics

Many approaches in computational historical linguistics try to capture a genetic signal, by staying close to one or more steps of the comparative method, where sound correspondences are a central notion. Hruschka *et al.* (2015) create a phylogeny of languages using Bayesian MCMC, while at the same time giving a probabilistic description of

regular sound correspondences. By explicitly modelling the sound correspondences, this approach stays close to one of the main principles of the comparative method. Bouchard-Côté *et al.* (2013) directly compare phonetic strings of words, in order to reconstruct the protoforms of words and perform cognate detection. Probabilistic string transducers model the sound changes, taking into account context. A tree is postulated, and in an iterative process, candidate protoforms are generated. Parameters are estimated using Expectation Maximization. Sound correspondences and protoforms, central notions in the comparative method, are explicitly modelled.

In cognate detection, some approaches depart from comparing phonetic forms, but then follow a genotypic path, by extracting sound correspondences. List (2012) places phonetic strings of words into sound classes. Then, a matrix of language-pair dependent scores for sound correspondences is extracted. Based on this matrix, distances are assigned to cognate candidates. Finally, they are clustered into cognate classes. The matrix that is internally kept, is essentially a matrix of sound correspondences.

Different approaches are applied to follow the comparative method in phylogenetic tree reconstruction. Jäger (2015) applies a distance-based clustering algorithm for tree reconstruction. String similarities between alignments of words are directly used as distances between the languages. Although direct string comparison looks like a phenotypic step in this method, common ancestry is captured by explicitly removing words which show chance resemblances and borrowings, using a statistical test (Cronbach's alpha). Jäger (2018) introduces *soundclass-concept* characters to encode word forms as input for character-based phylogenetic models. In this approach, a word is represented by the presence or absence of (classes of) phonemes, leading to a representation of sound changes.

Word prediction 6.2.2

Approaches similar to word prediction have been applied before in the natural language processing community, but with differences in implementation and goal to our approach. An early example is Mulloni (2007), who used Support Vector Machine classifiers to predict words, with the goal of improving bilingual terminology lists and machine

translation algorithms. Beinborn *et al.* (2013) and Ciobanu (2016) perform word prediction, using methods from statistical machine translation, to assess the learnability of words for second language learners.

Some recent prediction approaches from NLP have the goal, like our approach, to reconstruct language ancestry. Ciobanu and Dinu (2018) predict from several Romance languages to Latin using Conditional Random Fields, and perform preliminary experiments using RNNs. Results for the different language pairs are then combined using an ensemble system to arrive at Latin protoforms. Ciobanu *et al.* (2020) and Ciobanu and Dinu (2020) report on variations of the same method, for prediction to modern languages. Meloni *et al.* (2019) use encoder-decoder neural networks with attention to predict Latin forms, from word lists from multiple Romance languages simultaneously as input. An analysis is performed of the structures the network learned, by evaluating the model on synthetic input words. Cathcart and Wandl (2020) predict words, in phonetic form, from proto-Slavic protoforms to contemporary Slavic languages, using an encoder-decoder with attention. The authors perform an elaborate error analysis, and use the trained embeddings of their model to reconstruct a phylogenetic tree of Slavic languages. Fourrier and Sagot (2020a) and Fourrier (2020) predict words back and forth between contemporary and proto-languages, and between contemporary languages, using artificial and realistic data, applying a model from statistical machine translation and a neural multiway encoder-decoder.

Although the applied methods differ, what the preceding approaches have in common is that their input consists solely of cognates. In our approach, the algorithm can be trained on data which is not labelled for cognacy, avoiding the need for manual cognate judgments. Moreover, in our approach, prediction results serve as a starting point for performing a number of diverse tasks in historical linguistics, such as phylogenetic tree reconstruction and cognate detection.

Recently, the prediction paradigm has gained ground in the computational historical linguistics community as well. List (2019a) uses a network analysis algorithm to predict word forms. This method is evaluated by predicting forms for unexplored languages, which are then attested by performing linguistic fieldwork (Bodt and List 2019, 2020).

Using prediction error as distance 6.3

Is the prediction error (normalized edit distance) between the predicted word and the target word a good measure for the distance between languages? The intuition behind using the prediction error as language distance is that two languages which are well-predictable through regular sound correspondences, must be closely related. There are however a few issues involved.

Firstly, assigning a distance of 0 for word forms which are fully predictable using regular sound correspondences, is somewhat problematic. One could argue that languages which differ only through regular sound correspondences should also receive a non-zero distance because they are not identical. For the specific case of reconstructing proto-languages, List (2019b) proposes that two reconstruction systems for a proto-language, that produce protoforms only differing by regular correspondences, can be seen as *structurally identical*. The rationale behind the concept of structual identity is that reconstructions of proto-languages are to some extent abstractions, in which arbitrary symbols could be used in protoforms. However, contemporary languages only differing through regular sound correspondences could not be called structurally identical, as these languages are not abstractions. Ideally, a model of language ancestry would give multiple distances: one distance based on the number of mutations made using regular sound correspondences, and one distance based on the number of irregular mutations made (including non-cognate words). Hruschka *et al.* (2015) created a model which explicitly models the distinction between regular and irregular sound changes, when creating a phylogenetic family tree. This model does not, however, rely on prediction nor prediction distances.

Another issue is that prediction error is not a very informative distance for languages which have non-cognate word pairs for many concepts. For non-cognate word pairs, the model cannot apply any learned sound correspondences to predict the word. The word will in many cases be completely incorrectly predicted, with a prediction error (normalized edit distance) towards 1. This does not inform us about the linguistic distance between these words. As most non-cognate word pairs will receive an error towards 1, when averaging over all con-

cepts for a language, this will give a measure of the proportion of non-cognates between two languages.

The final issue that needs to be discussed is that, when using the prediction error as language distance, the prediction error also signifies the model performance. When using a better-performing model, the distances between some or all languages may suddenly be smaller. Ideally, one would want a metric of language distance which is independent of the performance of the underlying model. A distance metric that discounts for model complexity could possibly draw upon ideas from the Minimum Description Length principle (MDL) (Rissanen 1978; Grunwald 2004). In the MDL framework, the best model to describe a dataset is the simplest model that is accurately able to compress the data by finding regularities. This corresponds to the model with the lowest description length. The description length is the sum of the length it takes to describe the model (model complexity) and the length it takes to describe the data with the help of the model (prediction error). In our case, when using description length as a distance metric, the low prediction error of a complex model will be discounted by adding the model complexity to the distance. MDL has been used before as a distance metric between languages to perform phylogenetic reconstruction (Wettig *et al.* 2011; Fischer *et al.* 2018).

All in all, prediction error is not a perfect measure for language distance. However, it is a reasonable approximation for our purposes, which is straightforward to obtain from a prediction model.

6.4 *Historical linguistics as latent variable model*

Taking a step back, the problem of inferring the phylogeny of languages from present-day language data can be viewed as a *latent variable model*. A latent variable model is a model where latent variables $\boldsymbol{\eta}$, whose value cannot be observed, are connected to observed variables $\boldsymbol{y}$. In these models, one could infer the value of a hidden variable from a certain observed variable. As a genotypic method, word prediction is one instantiation of the latent variable problem of inferring phylogeny, although it has not been explicitly modelled as such a problem in this paper. Cathcart and Wandl (2020) (cf. earlier approaches

Doyle *et al.* (2014); Murawaki (2017)), explicitly model latent variables to describe language history, by using a neural network with discrete straight-through embeddings. A prediction model, where latent variables can be identified, such as phonological processes of change, appears to be a viable direction for the future.

CONCLUSION 7

With this paper, we hope to contribute to future insights about the ancestry of languages. By applying computational methods in historical linguistics, advances have been made in recent years. In this paper, we built further upon this development and proposed a central role for the prediction paradigm from machine learning in historical linguistics. We showed that a simple probabilistic sequence model and embedding encoding of input data can be good implementation choices. We came up with approaches to apply the prediction paradigm to multiple tasks in historical linguistics: phylogenetic tree reconstruction, sound correspondence identification and cognate detection. After validating these techniques on well-studied language families, they can be especially valuable for language families for which ample data is available, but the exact language history remains unclear. We are looking forward to future research on prediction methods in historical linguistics that can further explore good computational models to come to new linguistic insights.

CODE 8

A user-friendly, interactive version of the code, in a Jupyter notebook, can be downloaded from `https://github.com/peterdekker/prediction-histling/`. This code is meant for educational purposes, results may differ slightly from those presented in this paper. For the original code used to generate the results in this paper, see `https://bitbucket.org/pdekker/wordprediction/`.

ACKNOWLEDGEMENTS

We would like to thank Gerhard Jäger, Johann-Mattis List, Guus Kroonen and Bart de Boer for their valuable comments. PD was supported by a PhD Fellowship fundamental research (11A2821N) of the Research Foundation – Flanders (FWO) and by funding from the Flemish Government under the *Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen* programme. WZ was funded by the Netherlands Organization for Scientific Research (NWO), through a Gravitation Grant 024.001.006 to the Language in Interaction Consortium.

REFERENCES

Yong-Yeol Ahn, James P. Bagrow, and Sune Lehmann (2010), Link Communities Reveal Multiscale Complexity in Networks, *Nature*, 466(7307):761–764.

Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo (2009), A Comparison of Extrinsic Clustering Evaluation Metrics Based on Formal Constraints, *Information Retrieval*, 12(4):461–486.

Cormac Anderson, Tiago Tresoldi, Thiago Chacon, Anne-Maria Fehn, Mary Walworth, Robert Forkel, and Johann-Mattis List (2018), A Cross-Linguistic Database of Phonetic Transcription Systems, *Yearbook of the Poznan Linguistic Meeting*, 4(1):21–53, ISSN 2449-7525, doi:10.2478/yplm-2018-0002.

Amit Bagga and Breck Baldwin (1998), Entity-Based Cross-Document Coreferencing Using the Vector Space Model, in *Proceedings of the 17th International Conference on Computational Linguistics*, volume 1, pp. 79–85.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio (2014), Neural Machine Translation by Jointly Learning to Align and Translate, *arXiv preprint arXiv:1409.0473*.

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych (2013), Cognate Production Using Character-Based Machine Translation, in Ruslan Mitkov and Jong C. Park, editors, *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pp. 883–891, Nagoya, Japan.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville (2013), Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation, *arXiv:1308.3432 [cs]*.

Timotheus A. Bodt and Johann-Mattis List (2019), Testing the Predictive Strength of the Comparative Method: An Ongoing Experiment on Unattested Words in Western Kho-Bwa Languages, *Papers in Historical Phonology*, 4:22–44, ISSN 2399-6714, doi:10.2218/pihph.4.2019.3037.

Timotheus A. Bodt and Johann-Mattis List (2020), The Multiple Benefits of Making Predictions in Linguistics, *Babel: The Language Magazine*, 31(2):8–12, doi:http://dx.doi.org/10.17613/m688-4b90.

Alexandre Bouchard-Côté, David Hall, Thomas L. Griffiths, and Dan Klein (2013), Automated Reconstruction of Ancient Languages Using Probabilistic Models of Sound Change, *Proceedings of the National Academy of Sciences*, 110(11):4224–4229.

Alexandre Bouchard-Côté, Percy Liang, Thomas L. Griffiths, and Dan Klein (2007), A Probabilistic Approach to Diachronic Phonology, in Jason Eisner, editor, *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 887–896.

Remco Bouckaert, Philippe Lemey, Michael Dunn, Simon J. Greenhill, Alexander V. Alekseyenko, Alexei J. Drummond, Russell D. Gray, Marc A. Suchard, and Quentin D. Atkinson (2012), Mapping the Origins and Expansion of the Indo-European Language Family, *Science*, 337(6097):957–960.

Cecil H. Brown, Eric W. Holman, Søren Wichmann, and Viveka Velupillai (2008), Automated Classification of the World's Languages: A Description of the Method and Preliminary Results, *STUF – Language Typology and Universals*, 61(4):285–308.

David Bryant, John Tsang, Paul E. Kearney, and Ming Li (2000), Computing the Quartet Distance between Evolutionary Trees, in *Symposium on Discrete Algorithms: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, volume 9, pp. 285–286.

Lyle Campbell (2013), *Historical Linguistics: An Introduction*, MIT Press, second edition.

Chundra Cathcart and Taraka Rama (2020), Disentangling Dialects: A Neural Approach to Indo-Aryan Historical Phonology and Subgrouping, in Raquel Fernández and Tal Linzen, editors, *Proceedings of the 24th Conference on Computational Natural Language Learning*, pp. 620–630, Association for Computational Linguistics, Online, doi:10.18653/v1/2020.conll-1.50.

Chundra Cathcart and Florian Wandl (2020), In Search of Isoglosses: Continuous and Discrete Language Embeddings in Slavic Historical Phonology, in Garrett Nicolai, Kyle Gorman, and Ryan Cotterell, editors, *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pp. 233–244, Association for Computational Linguistics, Online, doi:10.18653/v1/2020.sigmorphon-1.28.

Will Chang, Chundra Cathcart, David Hall, and Andrew Garrett (2015), Ancestry-Constrained Phylogenetic Analysis Supports the Indo-European Steppe Hypothesis, *Language*, 91(1):194–244.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014), Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation, in Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Association for Computational Linguistics, Doha, Qatar, doi:10.3115/v1/D14-1179.

Kenneth Ward Church and Patrick Hanks (1990), Word Association Norms, Mutual Information, and Lexicography, *Computational Linguistics*, 16(1):22–29.

Alina Maria Ciobanu (2016), Sequence Labeling for Cognate Production, *Procedia Computer Science*, 96:1391–1399, ISSN 18770509, doi:10.1016/j.procs.2016.08.184.

Alina Maria Ciobanu and Liviu P. Dinu (2014), Building a Dataset of Multilingual Cognates for the Romanian Lexicon, in Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pp. 1038–1043.

Alina Maria Ciobanu and Liviu P. Dinu (2018), Ab Initio: Automatic Latin Proto-Word Reconstruction, in Emily M. Bender, Leon Derczynski, and Pierre Isabelle, editors, *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1604–1614.

Alina Maria Ciobanu and Liviu P. Dinu (2020), Automatic Identification and Production of Related Words for Historical Linguistics, *Computational Linguistics*, 45(4):667–704, ISSN 0891-2017, 1530-9312, doi:10.1162/coli_a_00361.

Alina Maria Ciobanu, Liviu P. Dinu, and Laurentiu Zoicas (2020), Automatic Reconstruction of Missing Romanian Cognates and Unattested Latin Words, in *Proceedings of the 12th Language Resources and Evaluation Conference*, pp. 3226–3231.

James Clackson (2007), *Indo-European Linguistics: An Introduction*, Cambridge University Press.

Michael Collins (2002), Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms, in *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing-Volume 10*, pp. 1–8.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio (2016), Binarized Neural Networks: Training Deep Neural

Networks with Weights and Activations Constrained to +1 or −1, *arXiv:1602.02830 [cs]*.

Harold Charles DAUME and Daniel MARCU (2006), *Practical Structured Learning Techniques for Natural Language Processing*, University of Southern California.

Peter DEKKER (2018), *Reconstructing Language Ancestry by Performing Word Prediction with Neural Networks*, MSc thesis, University of Amsterdam.

Johannes DELLERT (2018), Combining Information-Weighted Sequence Alignment and Sound Correspondence Models for Improved Cognate Detection, in *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 3123–3133.

Johannes DELLERT, Thora DANEYKO, Alla MÜNCH, Alina LADYGINA, Armin BUCH, Natalie CLARIUS, Ilja GRIGORJEW, Mohamed BALABEL, Hizniye Isabella BOGA, Zalina BAYSAROVA, Roland MÜHLENBERND, Johannes WAHLE, and Gerhard JÄGER (2019), NorthEuraLex: A Wide-Coverage Lexical Database of Northern Eurasia, *Language Resources and Evaluation*, ISSN 1574-020X, 1574-0218, doi:10.1007/s10579-019-09480-6.

Rick DERKSEN (2007), *Etymological Dictionary of the Slavic Inherited Lexicon*, Brill.

Jacob DEVLIN, Ming-Wei CHANG, Kenton LEE, and Kristina TOUTANOVA (2019), BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding, in Jill BURSTEIN, Christy DORAN, and Thamar SOLORIO, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Association for Computational Linguistics, Minneapolis, Minnesota, doi:10.18653/v1/N19-1423.

Sander DIELEMAN, Jan SCHLÜTER, Colin RAFFEL, Eben OLSON, Søren Kaae SØNDERBY, Daniel NOURI, Daniel MATURANA, Martin THOMA, Eric BATTENBERG, Jack KELLY, Jeffrey De FAUW, Michael HEILMAN, Diogo Moitinho DE ALMEIDA, Brian MCFEE, Hendrik WEIDEMAN, Gábor TAKÁCS, Peter DE RIVAZ, Jon CRALL, Gregory SANDERS, Kashif RASUL, Cong LIU, Geoffrey FRENCH, and Jonas DEGRAVE (2015), Lasagne: First Release., doi:10.5281/zenodo.27878.

Gabriel DOYLE, Klinton BICKNELL, and Roger LEVY (2014), Nonparametric Learning of Phonological Constraints in Optimality Theory, in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1094–1103, Association for Computational Linguistics, Baltimore, Maryland, doi:10.3115/v1/P14-1103.

John DUCHI, Elad HAZAN, and Yoram SINGER (2011), Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Michael Dunn (2012), Indo-European Lexical Cognacy Database (IELex).

John R. Firth (1957), A Synopsis of Linguistic Theory, 1930-1955, *Studies in linguistic analysis*.

Andrea K. Fischer, Jilles Vreeken, and Dietrich Klakow (2018), Beyond Pairwise Similarity: Quantifying and Characterizing Linguistic Similarity between Groups of Languages by MDL, *Computación y Sistemas*, 21(4), ISSN 2007-9737, 1405-5546, doi:10.13053/cys-21-4-2865.

Robert Forkel, Johann-Mattis List, Simon J. Greenhill, Christoph Rzymski, Sebastian Bank, Michael Cysouw, Harald Hammarström, Martin Haspelmath, Gereon A. Kaiping, and Russell D. Gray (2018), Cross-Linguistic Data Formats, Advancing Data Sharing and Re-Use in Comparative Linguistics, *Scientific Data*, 5:180205, ISSN 2052-4463, doi:10.1038/sdata.2018.205.

Clémentine Fourrier (2020), Évolution phonétique des langues et réseaux de neurones: travaux préliminaires, in Christophe Benzitoun, Chloé Braud, Laurine Huber, David Langlois, Slim Ouni, and Sylvain Pogodalla, editors, *Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition)*, volume 3: Rencontre des Étudiants Chercheurs en Informatique pour le TAL.

Clémentine Fourrier and Benoît Sagot (2020a), Comparing Statistical and Neural Models for Learning Sound Correspondences, in *LT4HALA 2020: First Workshop on Language Technologies for Historical and Ancient Languages*, Marseille, France.

Clémentine Fourrier and Benoît Sagot (2020b), Methodological Aspects of Developing and Managing an Etymological Lexical Resource: Introducing EtymDB-2.0, in Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 12th Language Resources and Evaluation Conference*, pp. 3207–3216, European Language Resources Association, Marseille, France, ISBN 979-10-95546-34-4.

Xavier Glorot and Yoshua Bengio (2010), Understanding the Difficulty of Training Deep Feedforward Neural Networks, in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256.

Russell D. Gray and Quentin D. Atkinson (2003), Language-Tree Divergence Times Support the Anatolian Theory of Indo-European Origin, *Nature*, 426(6965):435–439.

Simon J. Greenhill, Chieh-Hsi Wu, Xia Hua, Michael Dunn, Stephen C. Levinson, and Russell D. Gray (2017), Evolutionary Dynamics of Language Systems, *Proceedings of the National Academy of Sciences*, 114(42):E8822–E8829.

Peter GRUNWALD (2004), A Tutorial Introduction to the Minimum Description Length Principle, *arXiv:math/0406077*.

Harald HAMMARSTRÖM, Robert FORKEL, Martin HASPELMATH, and Sebastian BANK (2020), Glottolog 4.2.1.

Sepp HOCHREITER and Jürgen SCHMIDHUBER (1997), Long Short-Term Memory, *Neural Computation*, 9(8):1735–1780.

Daniel J. HRUSCHKA, Simon BRANFORD, Eric D. SMITH, Jon WILKINS, Andrew MEADE, Mark PAGEL, and Tanmoy BHATTACHARYA (2015), Detecting Regular Sound Changes in Linguistics as Events of Concerted Evolution, *Current Biology*, 25(1):1–9.

Jaime HUERTA-CEPAS, François SERRA, and Peer BORK (2016), ETE 3: Reconstruction, Analysis, and Visualization of Phylogenomic Data, *Molecular Biology and Evolution*, 33(6):1635–1638.

Diana INKPEN, Oana FRUNZA, and Grzegorz KONDRAK (2005), Automatic Identification of Cognates and False Friends in French and English, in *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pp. 251–257.

Gerhard JÄGER (2014), Phylogenetic Inference from Word Lists Using Weighted Alignment with Empirically Determined Weights, in *Quantifying Language Dynamics*, pp. 155–204, Brill.

Gerhard JÄGER (2015), Support for Linguistic Macrofamilies from Weighted Sequence Alignment, *Proceedings of the National Academy of Sciences*, 112(41):12752–12757.

Gerhard JÄGER (2018), Global-Scale Phylogenetic Linguistic Inference from Lexical Resources, *Scientific Data*, 5(1):180189, ISSN 2052-4463, doi:10.1038/sdata.2018.189.

Gerhard JÄGER (2019), Computational Historical Linguistics, *Theoretical Linguistics*, 45(3–4):151–182, ISSN 0301-4428, 1613-4060, doi:10.1515/tl-2019-0011.

Gerhard JÄGER and Johann-Mattis LIST (2016), Statistical and Computational Elaborations of the Classical Comparative Method.

Gerhard JÄGER, Johann-Mattis LIST, and Pavel SOFRONIEV (2017), Using Support Vector Machines and State-of-the-Art Algorithms for Phonetic Alignment to Identify Cognates in Multi-Lingual Wordlists, in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 1205–1216, Association for Computational Linguistics, Valencia, Spain, doi:10.18653/v1/E17-1113.

Gerhard JÄGER and Pavel SOFRONIEV (2016), Automatic Cognate Classification with a Support Vector Machine, in *Proceedings of the 13th Conference on Natural Language Processing*, volume 16.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush (2016), Character-Aware Neural Language Models, in *Thirtieth AAAI Conference on Artificial Intelligence*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira (2001), Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, in *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pp. 282–289, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ISBN 1-55860-778-1.

Roger Lass (1997), *Historical Linguistics and Language Change*, Cambridge University Press, Cambridge.

Vladimir I. Levenshtein (1966), Binary Codes Capable of Correcting Deletions, Insertions, and Reversals, in *Soviet Physics Doklady*, volume 10, pp. 707–710.

Johann-Mattis List (2012), LexStat: Automatic Detection of Cognates in Multilingual Wordlists, in Miriam Butt, Sheelagh Carpendale, Gerald Penn, Jelena Prokić, and Michael Cysouw, editors, *Proceedings of the EACL 2012 Joint Workshop of LINGVIS & UNCLH*, pp. 117–125, Association for Computational Linguistics, Avignon, France.

Johann-Mattis List (2019a), Automatic Inference of Sound Correspondence Patterns across Multiple Languages, *Computational Linguistics*, 45(1):137–161, ISSN 0891-2017, 1530-9312, doi:10.1162/coli_a_00344.

Johann-Mattis List (2019b), Beyond Edit Distances: Comparing Linguistic Reconstruction Systems, *Theoretical Linguistics*, 45(3-4):247–258, ISSN 0301-4428, 1613-4060, doi:10.1515/tl-2019-0016.

Johann-Mattis List, Simon Greenhill, Tiago Tresoldi, and Robert Forkel (2019), LingPy. A Python Library for Historical Linguistics, *Jena: Max Planck Institute for the Science of Human History*, doi:https://zenodo.org/badge/latestdoi/5137/lingpy/lingpy.

Thomas Mailund and Christian NS Pedersen (2004), QDist – Quartet Distance between Evolutionary Trees, *Bioinformatics*, 20(10):1636–1637.

Carlo Meloni, Shauli Ravfogel, and Yoav Goldberg (2019), Ab Antiquo: Proto-Language Reconstruction with RNNs, *arXiv:1908.02477 [cs]*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean (2013), Distributed Representations of Words and Phrases and Their Compositionality, in *Advances in Neural Information Processing Systems*, pp. 3111–3119.

Andrea Mulloni (2007), Automatic Prediction of Cognate Orthography Using Support Vector Machines, in Chris Biemann, Violeta Seretan, and Ellen Riloff, editors, *Proceedings of the ACL 2007 Student Research Workshop*, pp. 25–30, Association for Computational Linguistics, Prague, Czech Republic.

Yugo MURAWAKI (2017), Diachrony-Aware Induction of Binary Latent Representations from Typological Features, in *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, volume 1: Long papers, pp. 451–461.

Saul B. NEEDLEMAN and Christan D. WUNSCH (1970), A Gene Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins, *Journal of Molecular Biology*, 48:443–453.

Jeffrey PENNINGTON, Richard SOCHER, and Christopher MANNING (2014), Glove: Global Vectors for Word Representation, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543.

Simone POMPEI, Vittorio LORETO, and Francesca TRIA (2011), On the Accuracy of Language Trees, *PLoS One*, 6(6):e20109.

Taraka RAMA (2016), Siamese Convolutional Networks for Cognate Identification, in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*.

Taraka RAMA and Johann-Mattis LIST (2019), An Automated Framework for Fast Cognate Detection and Bayesian Phylogenetic Inference in Computational Historical Linguistics, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6225–6235, Association for Computational Linguistics, Florence, Italy, doi:10.18653/v1/P19-1627.

Sanda REINHEIMER RIPEANU (2001), Lingvistica Romanica: Lexic, Morfologie, Fonetica.

Jorma RISSANEN (1978), Modeling by Shortest Data Description, *Automatica*, 14(5):465–471, ISSN 00051098, doi:10.1016/0005-1098(78)90005-5.

Frank ROSENBLATT (1958), The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, *Psychological review*, 65(6):386–408, doi:10.1037/h0042519.

Naruya SAITOU and Masatoshi NEI (1987), The Neighbor-Joining Method: A New Method for Reconstructing Phylogenetic Trees, *Molecular Biology and Evolution*, 4(4):406–425.

Robert. R. SOKAL and Charles. D. MICHENER (1958), A Statistical Method for Evaluating Systematic Relationships, *University of Kansas Scientific Bulletin*, 28:1409–1438.

Ilya SUTSKEVER, Oriol VINYALS, and Quoc V. LE (2014), Sequence to Sequence Learning with Neural Networks, in *Advances in Neural Information Processing Systems*, pp. 3104–3112.

Stijn Marinus VAN DONGEN (2000), *Graph Clustering by Flow Simulation*, Ph.D. thesis, University of Utrecht.

Ashish VASWANI, Noam SHAZEER, Niki PARMAR, Jakob USZKOREIT, Llion JONES, Aidan N. GOMEZ, Łukasz KAISER, and Illia POLOSUKHIN (2017), Attention Is All You Need, in I. GUYON, U. V. LUXBURG, S. BENGIO, H. WALLACH, R. FERGUS, S. VISHWANATHAN, and R. GARNETT, editors, *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, Curran Associates, Inc.

Andrew J. VITERBI (1967), Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm, *IEEE Transactions on Information Theory*, 13(2):260–269, ISSN 0018-9448, doi:10.1109/TIT.1967.1054010.

Hannes WETTIG, Suvi HILTUNEN, and Roman YANGARBER (2011), MDL-Based Models for Alignment of Etymological Data, in *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pp. 111–117.

Martijn WIELING, Jelena PROKIĆ, and John NERBONNE (2009), Evaluating the Pairwise String Alignment of Pronunciations, in *Proceedings of the EACL 2009 Workshop on Language Technology and Resources for Cultural Heritage, Social Sciences, Humanities, and Education*, pp. 26–34, Association for Computational Linguistics.

Shijie WU and Ryan COTTERELL (2019), Exact Hard Monotonic Attention for Character-Level Transduction, in Preslav NAKOV and Alexis PALMER, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1530–1537, Association for Computational Linguistics, Florence, Italy, doi:10.18653/v1/P19-1148.

Peter Dekker
0000-0003-4734-2668
peter.dekker@ai.vub.ac.be

AI Lab
Vrije Universiteit Brussel
Pleinlaan 2
1050 Brussels
Belgium

Willem Zuidema
0000-0002-2362-5447
W.H.Zuidema@uva.nl

Institute for Logic, Language
and Computation (ILLC)
University of Amsterdam
P.O. Box 94242
1090 GE Amsterdam
The Netherlands

Peter Dekker and Willem Zuidema (2020), *Word prediction in computational historical linguistics*, Journal of Language Modelling, 8(2):295–336
https://dx.doi.org/10.15398/jlm.v8i2.268