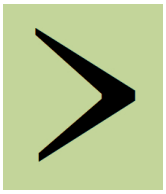

Papers in Historical Phonology
http://journals.ed.ac.uk/pihph
ISSN 2399-6714
Volume 3, 137–157 (2018)
DOI: 10.2218/pihph.3.2018.2878

Machine learning in diachronic corpus phonology: mining verse data to infer trajectories in English phonotactics

ANDREAS BAUMANN
University of Vienna

Abstract

Machine learning is a powerful method when working with large data sets such as diachronic corpora. However, as opposed to standard techniques from inferential statistics like regression modeling, machine learning is less commonly used among phonological corpus linguists. This paper discusses three different machine learning techniques (K nearest neighbors classifiers; Naïve Bayes classifiers; artificial neural networks) and how they can be applied to diachronic corpus data to address specific phonological questions. To illustrate the methodology, I investigate Middle English schwa deletion and when and how it potentially triggered reduction of final /mb/ clusters in English.

1 Introduction

In this methodological paper, I demonstrate how machine learning techniques can be used to generate more nuanced data for research in diachronic corpus phonology. This is motivated by the following problem. In the diachronic study of English, the phenomenon of final schwa deletion is complicated: it is gradual (as most linguistic changes are); spelling does not provide reliable cues for phonological analyses (and there is no audio data available for most periods to begin with); and it depends on many factors (e.g. phonological context, word length, morphosyntax, not to mention socio-geography; Minkova 1991).

However, for an English historical phonologist it is important to know if final schwa is present in a given period: (i) in metrical theory it is relevant for investigating stress clashes or numbers of syllables (Burzio 2007, Dresher & Lahiri 2005); (ii) in cognitive phonology one may be interested in diphones which function as cues for word segmentation (Dressler, Dziubalska-Kołaczyk & Pestal 2010, Daland & Pierrehumbert 2011); (iii) in phonotactics we want to be certain about

syllable structure (Hogg & McCully 1987, Dziubalska-Kołaczyk 2005), for instance if there is a coda cluster like /mb/ in Middle English items like *lambe* or if /b/ is in fact the onset of a final syllable /bə/.

To address questions like these in a statistically robust way, we need lots of data. Unfortunately, most of the data available are prose data, which are rarely phonologically annotated (but see Kopaczyk et al. 2018 for a major step towards phonological parsing in historical corpora). Verse data are arguably more suitable for studying phenomena like schwa deletion (because we can use rhythm as a diagnostic tool), but especially if we want to do long term studies involving many centuries, poetry data are sparse.

This discrepancy leads to the following idea: it would be great if we could exploit ('mine') poetry data, which are relatively reliable with respect to the phonological interpretation of final schwas, and use the information gained in this way to analyze large amounts of prose data.

Machine learning (ML) can be used to do exactly this. ML algorithms can be trained on some well-analyzed data; in a second step the trained models can be used to make predictions about new data that is not yet fully analyzed. The predicted data, finally, can be used for some further analysis of the phenomenon one is actually interested in. In this way, ML techniques provide the researcher with more nuanced data.

Of course, this procedure is well known in corpus linguistics. ML is used to do lemmatization, morphosyntactic analyses (e.g. PoS tagging), phonological and phonetic annotation, or translation (Manning 2015, Pustejovsky & Stubbs 2013, Baron & Rayson 2009; see also Ellison 1994). However, these methods are usually applied by expert computational linguists and more user-friendly tools like open-source taggers (e.g. AntCLAWSGUI, Anthony 2013) or spelling standardizers (e.g. VARD, Baron & Rayson 2008) are typically restricted to specific applications and/or historical periods.

The primary goal of this paper is to show that customized ML models can be computed by any researcher in diachronic linguistics who is familiar with basic techniques in inferential statistics. To do so, I discuss three well-known ML techniques: K nearest neighbors classifiers (KNN), Naïve Bayes classifiers (NB), and artificial neural networks (ANN) (section 2). I train them on previously analyzed verse data. The trained models are then applied to prose data which are not phonologically analyzed. To illustrate the use of the procedure, I finally infer diachronic trajectories of Middle English /mb/ clusters and discuss what can be learned about the onset of final /b/ deletion in words like *lamb* or *thumb* (section 3).

In terms of computational software, I rely on a couple of easy-to-use packages and functions in R (Team & R Development Core Team 2017). All code and data used in this paper are made available as associated material.

2 Learning schwa loss: model training and optimization

The general idea behind supervised machine learning techniques is this: in a first step, a ML model is trained on a dataset. This training dataset is complete in the sense that every single data point represents a set of input values together with an output value. Depending on the algorithm that underlies the ML technique, this dataset is used to learn which output values belong to which constellation of input values. The result of this learning procedure is a model (very much like a regression model fitted to a set of data points). After evaluating and perhaps refining the model, it can be used to predict output values for any (potentially new) constellation of input values. In what follows, I will discuss the training data used for this study and, after that, three different ML techniques: K nearest neighbors classifiers (KNN), Naïve Bayes classifiers (NB) and artificial neural networks (ANN).

2.1 Data

The goal of this study is to predict if a graphemically represented word-final schwa found in some historical (English) corpus was phonologically realized or in fact empty. Since written prose data do not give reliable cues as to how a word final <e> should be phonologically interpreted, I use corpus data which are phonologically more reliable in this regard, namely verse data. The dataset which I use as training data was originally compiled by Christina Prömer (Baumann & Prömer 2017) and is discussed in more detail in Baumann, Prömer & Ritt (in review).[1] In a nutshell, it consists of 626 lexical word tokens ending in <e> retrieved from poems distributed over the Middle and Early Modern English Period (roughly 1150 to 1700). Based on the rhythmic context, each word token was manually labeled as to whether it contains a phonologically present final schwa (schwa present 'Y'; schwa absent 'N'). The latter represents the output variable in our training data. The dataset features several potentially relevant input variables: (i) time (centuries numbered from 1 to 7, where 1 means 12th century and 7 means 18th century); (ii) length (approximated by counting the number of graphemes[2]); (iii) subsequent phonological context (vowel '1'; consonant '0'); and (iv) morphology (word ends in a suffix, e.g. plural <s(e)>: yes '1' or no '0'). Note that all input variables are numeric or transformed into numeric dummy variables (which can be easily

[1] The data used in this study can be found in the associated materials.
[2] This approximation is necessary since diachronic corpora are rarely phonologically annotated.

done, also for many-valued nominal variables like 'PoS'), while the output variable is categorical.

Before we can train our models, we must do some more work. Since two of the ML algorithms discussed here work better if all input variables operate on the same scale (KNN and ANN)[3] all variables were normalized to assume values between 0 and 1.[4] For example, time, which goes from 1 to 7 and denotes a century a given text comes from, has been scaled down to values between 0 and 1, where 0 means 12th century and 1 means 18th century. After that, the dataset was divided randomly into two parts, one for training the models and one for evaluating them (i.e. checking if the predictions of the model are correct). This is necessary as it does not make much sense to test the model against the data it was trained on beforehand. There are different suggestions concerning the proportions of data points that should be used for training and testing the model. For small data sets, attributing 2/3 of the data to training and 1/3 to testing the model is a widely adopted procedure (cf. Kohavi 2005).[5] Large training sets lead to more precise model parameter estimates while large testing sets entail more reliable model evaluation measures. Clearly, both are relevant.

2.2 ML algorithms: training and classification

In this section I discuss three prominent ML techniques. In each subsection, I first describe the rationale which underlies the respective learning algorithm and the way in which it classifies unlabeled data points. After addressing the corresponding R functions, I briefly elaborate on the advantages and drawbacks of the learning algorithms, focusing on applications in diachronic corpus linguistics.

2.2.1 K nearest neighbors classifier (KNN)

Let us begin with one of the most basic ML techniques, the K nearest neighbors classifier (KNN; cf. Zaki & Meira 2014), first proposed by Fix

[3] This is because KNN measures (usually Euclidian) distances between data points, so that large discrepancies between scales would result in variables operating on small scales (e.g. the interval from 0 to 1) being overshadowed by variables measured on large scales (e.g. dates from 1100 to 1700). ANN, on the other hand sometimes use activation functions which are sensitive to differential scaling. See section 2.2.

[4] This was done by employing the normalization $x_i \rightarrow \frac{x_i - \min \boldsymbol{x}}{\max \boldsymbol{x} - \min \boldsymbol{x}}$.

[5] In general, larger datasets allow for smaller proportions reserved for testing the model. For instance, if 100,000 data points are available it is sufficient to attribute about 1 to 5% to testing the model.

and Hodges (1952). The idea behind the algorithm is very simple. It classifies data points based on properties of their neighbors. Every data point can be conceptualized as a point in a d-dimensional space, where d is the number of input variables. In the present case, for instance, $d = 4$. The coordinates of each point in this space are given by its set of variable values. Figure 1a illustrates this for two dimensions (length and time). If the output variable is binary (as in our example Y/N) there are two categories of data points (you can think of them as having different colors like light and dark magenta in Figure 1a).

How do we identify the category (i.e. output value) of a new and yet unclassified data point, like the gray square in Figure 1a? To arrive at a label, KNN simply looks at those K data points which are closest to the new point, i.e. the K-neighborhood (hence, 'K nearest neighbors'), represented by the circle in Figure 1a. Here, K can be any previously defined number, e.g. $K = 3$.[6] Then, the K-neighborhood consists of those three points closest to the new point.[7] Afterwards, KNN computes the distribution of output values in the K-neighborhood. This is done, for instance, by simply counting the number of light and dark magenta points in the circle describing the K-neighborhood. The one category with the most representatives in the K-neighborhood wins and assigns its value to the new data point (for that reason, it is good to select an odd value for K to avoid ties in the case of binary output variables).

[6] The optimal value for K depends on the given data set and it can be determined by assessing model accuracy for different K values (see section 2.3 for details on accuracy).

[7] In the case of ties for the Kth nearest data point, all equidistant points are considered. Another viable option would be to randomly draw a single Kth nearest point among all equidistant candidates.

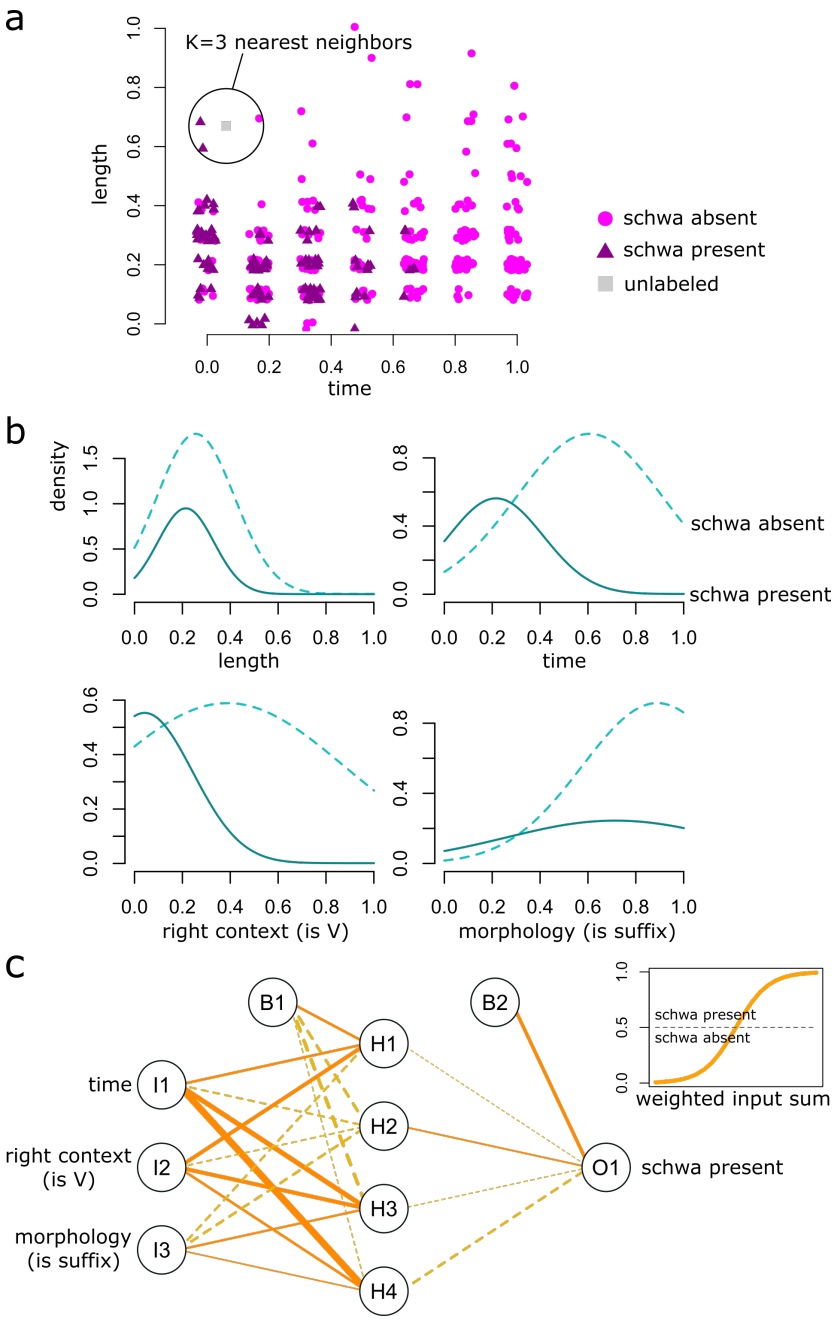

Figure 1: Model visualization: (a) KNN procedure of classifying an as yet unlabeled point (gray square) in the time×length plane (magenta circles: schwa absent; dark magenta triangles: schwa present; points jittered for better visibility). Morphology and right context are not represented. (b) NB probabilities of input variables given present (solid, dark cyan) and absent schwa (dashed, light cyan), respectively. (c) ANN: input nodes (I1 to I3), hidden nodes (H1 to H4), and an output node (schwa present, O1). B1 and B2 define additive bias terms. Solid dark orange lines represent positive weights while dashed light orange lines represent negative weights between nodes. Thickness corresponds to weight size. Upper right corner: decision function of output node.

In Figure 1a, the gray point gets the dark magenta label 'Y', i.e. schwa is present. The procedure is repeated for all unclassified data points.

In R, KNN predictions can be computed with, for example, the `knn()` function from the `class` package (Venables & Ripley 2002). It takes a data set of input values and a corresponding vector of output values, as well as a data set of unlabeled input values as arguments, and returns a vector of predicted output categories. The code can be found in the associated script (see the 'associated material' section at the end of this paper).

In general, KNN is a very simple and non-parametric algorithm (i.e. it does not have any distributional requirements) with moderate accuracy (cf. Kotsiantis 2007, Table 4). Its drawbacks are low speed of classification (which is certainly not an issue if working on small data sets, but potentially problematic in the case of large corpora) and low tolerance of missing values (Guo et al. 2003, Kotsiantis 2007). Since the definition of the *K*-neighborhood depends on a distance measure (usually Euclidian distance) it is sensitive to differences between scales of the input variables (Ripley 1996, 192) so that scale normalization is recommended. This is particularly important in diachronic linguistics where time is usually measured in years, decades or centuries (see section 2.1).

2.2.2 Naïve Bayes classifier (NB)

The Naïve Bayes classifier (NB) adopts an approach which is, as it were, methodologically orthogonal to that of the KNN algorithm. Rather than inspecting the close neighborhood of a data point and thereby capturing all properties (input values) of nearby data points at once, NB estimates global distributions of each input variable separately and combines them to arrive at a prediction for a given set of input values (Zaki & Meira 2014, Hand & Yu 2001).

Most fundamentally, the algorithm makes use of Bayes' theorem to determine the (so-called posterior) probability of a data point x (given by a set of d input values) having class c_i, in short $P(c_i|x)$. If this probability is known, one can simply choose the one class with the highest probability. Bayes' theorem states that this probability depends on the probability of a class c_i having a certain value x in the following way: $P(c_i|x) = P(c_i)P(x|c_i)/P(x)$. The first factor can be easily determined, given a set of training output values, and we can ignore the denominator if we are only interested in finding the category c_i yielding the maximal posterior probability. The conditional probability $P(x|c_i)$, however, is more complicated. The trick behind the NB algorithm is that

it naively pretends that all input variables are independent from each other. As a mathematical consequence of this, the latter probability can be written as the product $P(x|c_i) = P(x_1|c_i) \cdot P(x_2|c_i) \cdots P(x_d|c_i)$. From a computational point of view, it is not difficult to estimate the distributional properties of each of these factors separately given the training data.

Estimated (Gaussian) density distributions for the present training data are shown in Figure 1b. In each plot, x_j (the input variable, e.g. time) is measured on the horizontal axis and density is measured on the vertical axis (that is, each curve can be read like a very fine-grained histogram with a total area of 1). For each input dimension (i.e. input variable), there are two curves, one for the output value 'schwa present' and one for the value 'schwa absent'. For example, the distribution of present schwas with respect to time is concentrated on the left (most present schwas occur early), while absent schwas are concentrated on the right (in late periods schwa is lost). Given a new data point, say $x = (x_1 = 1, x_2 = 0.56, x_3 = 0.44, x_4 = 0)$, these curves are then used with the above formulas to determine the overall posterior probability for both categories (schwa absent/present). The category which scores higher wins and is assigned to *x*.

NB models can be conveniently computed in R with the `NaiveBayes()` function from the `klaR` package. It draws on the same syntax that is known from more basic statistical techniques, such as linear regression models (e.g. `lm()`). That is, the function defines a model object which can be plugged as an argument together with some as yet unclassified input data into the `predict()` function to yield predicted output values.

The NB algorithm is fast, simple to apply (there are no model parameters that have to be adjusted manually) and, by design, not severely sensitive to missing values in the data set (Kotsiantis 2007). This can come in handy if, for example, only partially annotated corpus data are available. Moreover, differential scaling is not an issue with the NB algorithm, which is practical for diachronic applications. It is, however, in general less accurate than other machine learning algorithms. Consequently, it requires larger training data sets, which may be problematic when working with historical language data. On the other hand, if many variables are involved it is potentially superior to other algorithms even if data sets are small (Hand & Yu 2001).

2.2.3 Artificial neural network (ANN)

The final machine learning approach discussed in this paper is inspired by the neurological sciences (McCulloch & Pitts 1943, Ripley 1996, Venables & Ripley 2002). Artificial neural networks (ANN) have been

developed to model interactions among linked neurons. ANNs consist of a number of different layers: an input layer, an output layer and (potentially multiple) hidden layers in between. Each layer consists of nodes, and nodes from different layers are linked to each other. In machine-learning terms the nodes in the input layer represent the input variables and the nodes in the output layer represent the output variables. In this paper we will only discuss unidirectional (so-called feed forward) networks with a single hidden layer and a single output node (schwa present). Figure 1c shows a representation of the ANN which accounts for the data introduced in 2.1.[8] ANNs with a single hidden layer are also referred to as 'shallow'. Shallow ANNs have a much simpler architecture than 'deep' ANNs which may feature many hidden layers and hundreds of thousands of nodes (LeCun, Bengio & Hinton 2015, Schmidhuber 2015). The training and usage of deep ANNs ('deep learning') goes beyond the scope of this paper.

The question now is: under which condition is the output node activated, *viz*. when is it true or false? This depends on two factors: first, on the values of the input nodes, and second on the strengths of the links among nodes (as well as some constant additive bias terms, very much like intercepts in regression models, labeled B1 and B2 in Figure 1c). These strengths (or weights) can be positive if activation of a node is promoted, or negative if it is inhibited by its predecessor (shown as dark orange solid and light orange dashed lines in Figure 1c), respectively. The actual activation of a node depends on the sum of all weights going into that node and is modeled by some activation (or decision) function. In simple networks, a common choice for the activation function is the logistic function $f(x) = 1/(1 + e^{-x})$. It has a sigmoid shape and maps real values (which may be positive or negative) to values within the unit interval. The actual classification takes place in the output layer: if $f(x)$ scores above a certain activation level (most often 0.5), the node is activated; here, schwa present, and absent otherwise (see plot in the upper right corner of Figure 1c).

Given a set of weights (with biases) and input values, one can thus predict the output value. The determination of the weights is exactly what happens during learning, i.e. if the ANN is trained on a set of given input and output values, just like coefficients which are estimated in regression procedures (Cheng & Titterington 1994). To arrive at a set of weights that account for the training data in an accurate way, a so-called loss function must be computed which measures the overall deviation of the predicted output values from the actual output values.

[8] Note that it only features three input variables. This will be discussed in section 2.3.

This loss function clearly depends on the model weights. Finally, the weights are chosen in such a way that the loss function is minimized. Again, this is equivalent to minimizing the sum of squared residuals in regression models. Since ANNs typically involve many weights, this can be computationally intensive. There are various minimization procedures, some of which are faster than others. For a review and comparison of ANNs and (logistic) regression models see Dreiseitl and Ohno-Machado (2002).[9] A brief introduction to the optimization of deep ANNs is provided by LeCun et al. (2015)

There are many packages in R which allow one to compute ANNs, one of which is the `nnet` package (Venables & Ripley 2002) with the identically named built in function. The `nnet()` function computes ANNs with a single hidden layer and has in-built options for different activation functions (logistic as default). Syntactically, it works like most inferential methods in R. Its output is a model object which can be used to predict output values for a given set of input values.

ANNs are characterized as an accurate ML technique with fast classification, albeit relatively slow learning and low tolerance of missing data values (Kotsiantis 2007). Their biggest disadvantage is that arriving at an optimal network architecture is not trivial. Two questions arise: (i) how many hidden layers should be included and (ii) how many nodes per hidden layer are optimal? Regarding the first question, it has been argued that a single hidden layer is sufficient for addressing relatively simple problems (Ripley 1996, Dreiseitl & Ohno-Machado 2002). As to the second question, there are rough guidelines of how many hidden nodes should be included, such as $N_H \leq N_I + 1$ (which was employed in this paper; see Kiranyaz et al. 2009 for a concise summary). Optimizing network structure is a difficult task and usually done experimentally through many cycles of training and evaluating different ANNs.

The major advantage of ANNs is that they can learn extremely complex relationships between input and output data provided that there is (a) enough data available and (b) an appropriate network architecture (Schmidhuber 2015).

It is worth pointing out that ANNs with logistic activation functions have a clear advantage over other ML algorithms with regard to diachronic linguistics: it has been pointed out repeatedly that phenomena of linguistic change adopt a sigmoid, i.e. logistic, development (Kroch 1989, Denison 2003, Blythe & Croft 2012, Wang & Minett 2005). Since time is, by definition, a relevant variable in

[9] As a matter of fact, ANNs can be seen as generalization of logistic (or multinomial) regression. See also Ripley (1996).

diachronic corpora, it is evident that logistic link functions, which effectively mimic logistic models, provide a useful tool for modeling dynamic language phenomena. Thus, beyond having computational advantages, ANNs also account for language change in a linguistically meaningful way.

2.3 Evaluation and model optimization

After running the ML algorithms described in the previous subsection, the models derived from the training data need to be evaluated. This is done by comparing the predictions for the output values based on the input test data (in our case 1/3 of the complete data set) against the actual output values in the test data.

One straightforward measure of the performance of a model is accuracy, i.e. the fraction of correctly identified labels. Since we have a binary output variable, the performance of the model can be arranged as a 2-by-2 contingency table. For the KNN algorithm applied to the present test data we have for instance:

	N (predicted)	Y (predicted)
N (actual)	131	19
Y (actual)	9	39

Table 1: KNN performance (schwa present 'Y'; schwa absent 'N')

Thus, 170 out of 198 items in the test data were labeled correctly. This amounts to a reasonably high accuracy of 86%.

As in regression modeling, having too many input variables bears the risk of overfitting, so that model estimates are unreliable. We can use accuracy to conduct a basic model optimization procedure. For this, we simply run an ML algorithm (e.g. KNN) for every possible subset of input variables. For each run, we compute accuracy and subsequently rank all input variable sets by their performance. Figure 2 shows accuracy scores for all possible combinations of input variables in the three ML techniques discussed in this paper.

A couple of remarks are in order. First, KNN and ANN show relatively equal accuracy while NB scores lower overall. Second, the choice of variables does not affect KNN and ANN that much while NB is severely affected by the selection of input variables. Third, almost all constellations featuring time belong to the best input variable sets in all algorithms. This indicates that time is particularly relevant for explaining schwa loss (evidently, not a surprising result). Finally, model

accuracy does not necessarily increase with the number of input variables. In particular, we can see that while the optimal input configuration of KNN and NB features all four variables, length is dropped in the top-most ANN model. For that reason, only three variables were used for the ANN in this paper (cf. Figure 1c).

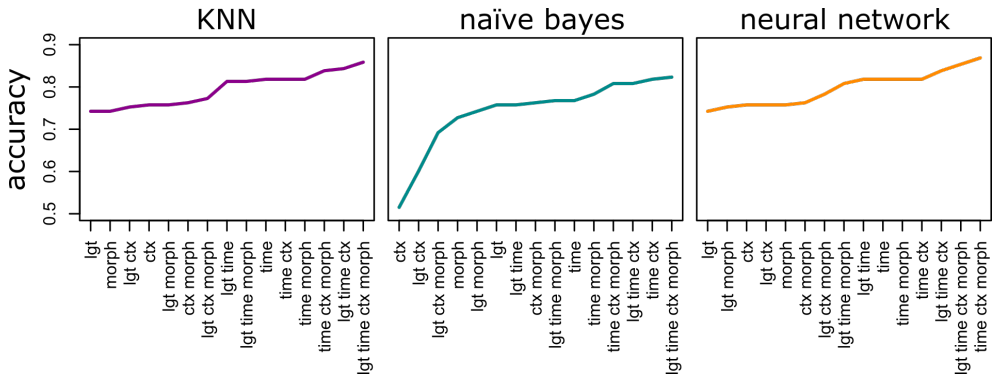

Figure 2: Model optimization through comparison of input variables. Vertical axis: model accuracy (fraction of correct predictions in training set). Horizontal axis: subsets of input variables among length (lgt), morphology (morph), right context (ctx), and time, ranked by model accuracy. KNN and NB are optimal if all available input variables are considered; ANN prefers a smaller set of input variables. Overall, KNN and ANN show a similar slightly increasing behavior while NB accuracy crucially depends on the selection of input variables.

This procedure of comparing models is very basic. Ideally, models should be validated at multiple random (mutually disjoint) subsets, which is often not possible if data sets are small. For a more comprehensive discussion of accuracy and more robust model selection procedures see Kohavi (1995).

3 Application: inferring the diachrony of final /mb(ə)/

In order to demonstrate the use of ML techniques in diachronic phonology, let us apply the ML models trained on verse data described in the previous section to some corpus data. One interesting phenomenon in the history of English phonotactics is that of /b/ loss in final /mb/ clusters, like in *lamb* or *thumb*, in most English varieties.[10] The change has been suggested to have occurred at some point during

[10] Final /mb/ is a convenient test case for yet another reason: it almost exclusively occurs in nouns so that word class can be excluded as potentially relevant factor. I would like to thank Klaus Hofmann for pointing this out.

the Middle English period (Ritt 1994, Dziubalska-Kołaczyk 2005, Starčević 2006).

It is very likely that this change happened as a reaction to schwa loss. If final schwa is dropped, /b/ moves into the coda of the final syllable. Since both segments are voiced and articulated at the same place, this may lead to insufficient articulatory and perceptual contrast (contra the principle of sonority sequencing and the obligatory contour principle; see e.g. Clements 1990, Guy & Boberg 1997, Boll-Avetisyan & Kager 2016). One resolution of this conflict would be to devoice final /b/; another one is cluster reduction, which actually took place.

A question relevant to the present phenomenon is: when did it start to propagate? It is reasonable to assume that /mb(ə)/ reduction required a reasonable number of final /mb/ sequences (we have pointed out elsewhere that high frequency destabilizes rather than stabilizes phonotactic items; see Baumann & Ritt 2018). However, for lack of phonologically annotated corpora it is difficult to measure token frequencies, because we cannot be sure if graphemically represented schwas have a phonological counterpart. Furthermore, there is probably not enough verse data available to yield statistically robust measures of phonologically interpretable final <mbe> items.

At this point, we can make use of our machine learning models trained on verse data. The idea is to extract all final <mbe> sequences from a large corpus (the Penn Helsinki corpora; Kroch & Taylor 2000, Kroch, Santorini & Delfs 2004) and let the trained ML models predict whether or not graphemically represented schwas are phonologically empty for each particular token. Such a list of potentially final /mb/ items can be conveniently retrieved from the ECCE database (Ritt, Prömer & Baumann 2017), together with information about the subsequent phonological context, morphological information (i.e. whether there is suffixation involved), and different measures of time. It is straightforward to compute the number of graphemes for each word token as well. The resulting list of <mbe> instances can be found in the associated materials (394 items in total).[11]

Counting the number of items with lost schwa ('N') in the predicted output data set per century leads to the historical trajectories shown in Figure 3. The gray line in this plot corresponds to frequencies predicted by a logistic model. These frequencies can be extracted from the ECCE database, which contains probabilistic weights computed for each token. Predicted frequencies in ECCE then are simply the sums of all

[11] To keep the present illustration as simple as possible, I only look at final <mbe> and do not take final <mb> into account (of which there are 90 tokens spread over all periods).

weights in each century (note that these weights were computed based on a logistic model which only depends on the right context and time; see Baumann et al. forthcoming, and for a related approach Baumann, Ritt & Prömer 2016).

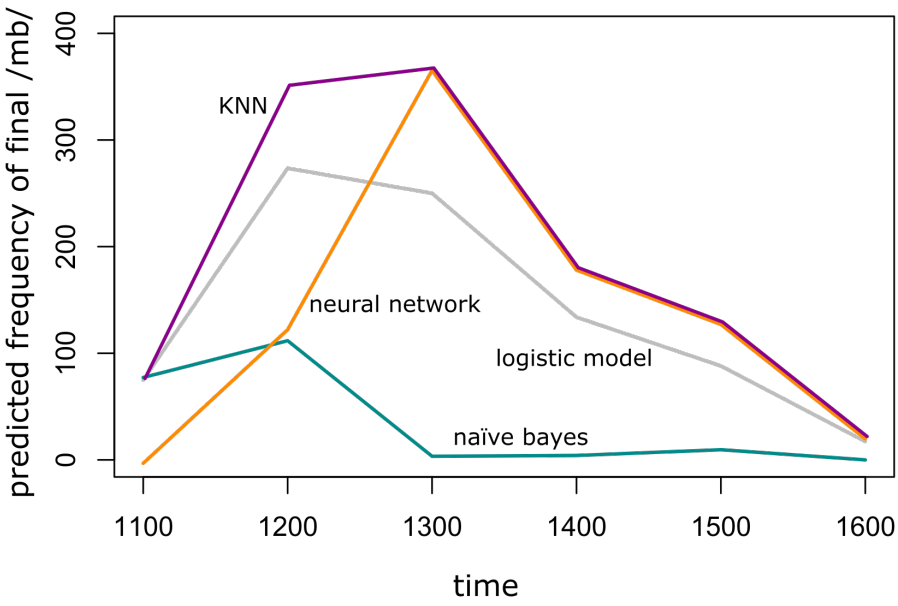

Figure 3: Predicted (per million normalized) token frequencies of final <mbe> sequences realized as /mb/ from 1100 to 1600. Color code: KNN magenta; NB cyan; ANN orange. The gray curve represents frequencies predicted from the logistic model in Baumann & Prömer (2017). It can be seen that /mb/ peaks around 1300. Note that in the present study, KNN predicts absent schwa for all items.

There are remarkable differences between the predictions by the three ML techniques discussed in this paper. KNN maximizes schwa loss in the sense that all <mbe> items are classified as /mb/.[12] NB goes in the opposite direction and predicts that final schwas have been present throughout the entire observation period. The ANN trajectory is somewhere in between: at the beginning, schwas are treated as present while they are lost from 1300 onwards. The trajectory predicted by the logistic model is slightly below the one of KNN (and later ANN). Note, however, that the trajectories predicted by the ML

[12] This can be likely attributed to the fact that all variables in our data are effectively measured on discrete scales (centuries; number of graphemes; binary distinctions) so that points are clustered together. See Figure 1a which uses jittered coordinates for easier visibility. Overall, there are more N than Y labels. Since KNN includes all equidistant *K*th neighbors for majority votes this entails it is much more likely to choose label N (schwa lost) than to choose label Y (schwa present).

models depend on a single training sample. Using a different sample may change the predicted trajectories (as the reader can explore by running the associated script). A thorough analysis would require comparisons of multiple trajectories based on a larger number of training samples (which, ideally, should be independent from each other).

What can we learn about /mb/ reduction given these frequency developments? Judging from the respective peaks of the trajectories, it can be inferred that the pressure for dropping final /b/ was highest in the late 12th or 13th century. Consequently, we can expect reduction of /mb/ to have started already in the middle of the Middle English period. This is in line with Lass (1992), who suggests pre-cluster lengthening (as in *climb*) to have occurred early in ME. Given our analysis we must remain agnostic with respect to the offset of that change, though. Note that the present analysis can give us no answer to the question of why final /mb/ decreased in frequency in the EME period either. It only tells us predicted frequencies of final /mb/ from which we infer the period showing the highest pressure for reducing /mb/ to /b/. We conclude that /mb/ reduction was a relatively early development.

4 Conclusion and outlook

In this paper, I have reviewed three standard machine learning techniques and demonstrated how they can be employed to make use of written (prose) corpus data for diachronic research in phonology. I trained three classifiers — K-nearest neighbors, naïve Bayes, and an artificial neural network — on verse data which are phonologically interpretable with respect to a specific phonological property (presence/absence of final schwa). Finally, the predictions from the ML models were used to infer token frequencies of final /mb/ clusters in the history of English. The key feature of the procedure is that I used models trained on phonologically interpretable data to classify a relatively large amount of prose data for which a reliable phonological assessment is difficult without the help of statistical techniques.

Which of the three ML algorithms is most promising for applications in diachronic corpus phonology? Clearly, there is no straightforward answer to this question, as the choice of methodology always depends on the data at hand. For instance, if datasets are huge, more parsimonious algorithms like NB might be preferred to KNN, which show comparably slow classification. Likewise, the training of ANNs might take too long if the network architecture is too complex. NB potentially outperforms KNN and ANN if datasets are incomplete, i.e. if

some entries are missing. The latter is a non-negligible problem if morphosyntactic annotations (e.g. PoS tags) or lemmas function as input variables, because historical language data are often difficult to classify (e.g. due to spelling variation or ongoing grammatical change).

When it comes to reliability, however, neural networks seem to be the method of choice, at least in the present analysis: based on the given training sample, they score highest on accuracy and the predictions for final schwa loss in /mb/ were neither too conservative (like NB) nor too generous (like KNN). Finally, their formal setup makes ANNs particularly useful for diachronic applications. Since ANNs are effectively generalizations of logistic models (if, as commonly done, sigmoid activation functions are used) they account for phenomena of linguistic change in a theoretically plausible and empirically supported way (Kroch 1989, Ripley 1996, Denison 2003).

Besides providing a review of a couple of ML techniques, one goal of this paper was to demonstrate that applying ML to diachronic corpus data is not particularly difficult. In fact, software packages and functions dedicated to machine learning are numerous (e.g. the R packages used in this paper) and running ML algorithms is in general no more difficult than computing a straightforward regression model (certain technical subtleties are present in both, nevertheless). While the primary use of regression techniques in linguistic research lies in the identification of significant interactions among certain factors (e.g. linguistic structure, frequency and time), supervised ML classification techniques tend to focus on a different aspect: they can yield more nuanced data. Particularly in historical corpus linguistics, I think, having more nuanced data is never a bad thing.

Comments invited

PiHPh relies on post-publication review of the papers that it publishes. If you have any comments on this piece, please add them to its comments site. You are encouraged to consult this site after reading the paper, as there may be comments from other readers there, and replies from the author. This paper's site is here:

https://doi.org/10.2218/pihph.3.2018.2878

Acknowledgements

I would like to thank Klaus Hofmann, Kamil Kaźmierski, Theresa Matzinger, Niki Ritt, Marlene Schwarz, as well as Patrick Honeybone

and Julian Bradfield for careful proof reading and many valuable comments on the manuscript and the associated script.

Associated material

The R script used to compute the ML models, predictions and visualizations in this paper can be found here (click 'Download/Downloaden' or 'View in browser/Objekt anzeigen'):

http://phaidra.univie.ac.at/o:754466

By running this script all necessary data are automatically imported from the Phaidra repository (schwa loss estimates from verse data in Baumann & Prömer 2017: phaidra.univie.ac.at/o:740449; final /mb(e)/ sequences retrieved from ECCE database, Ritt et al. 2017: phaidra.univie.ac.at/o:740448; text sizes for frequency normalization drawn from Kroch & Taylor 2000 and Kroch et al. 2004: phaidra.univie.ac.at/o:740372). Note that certain R packages must be installed beforehand (see script for further details).

Author contact details

Andreas Baumann
University of Vienna
Spitalgasse 2-4, 8.3
Vienna, A-1180
Austria

andreas.baumann@univie.ac.at

References

Anthony, Laurence. 2013. AntCLAWSGUI. Tokyo: Waseda University. http://www.laurenceanthony.net/software.

Baron, Alistair & Paul Rayson. 2008. VARD2: a tool for dealing with spelling variation in historical corpora. *Proceedings of the Postgraduate Conference in Corpus Linguistics*.

Baron, Alistair & Paul Rayson. 2009. Automatic standardisation of texts containing spelling variation: How much training data do you need? *Proceedings of Corpus Linguistics 2009*.

Baumann, Andreas & Christina Prömer. 2017. Interpolating diachronic phonotactic data: On the logistic spread of Middle English schwa loss. *23rd ICHL, San Antonio, TX*.

Baumann, Andreas, Christina Prömer & Nikolaus Ritt. Forthcoming. Reconstructing the spread of Middle English schwa deletion. *Italian Journal of Linguistics* (Special Issue, edited by Anderson,

Cormac & Natalia Kuznetsova).
Baumann, Andreas & Nikolaus Ritt. 2018. The basic reproductive ratio as a link between acquisition and change in phonotactics. *Cognition* 176. 174–183. doi:10.1016/j.cognition.2018.03.005.
Baumann, Andreas, Nikolaus Ritt & Christina Prömer. 2016. Diachronic dynamics of Middle English phonotactics provide evidence for analogy effects among lexical and morphonotactic consonant clusters. *Papers in Historical Phonology* 1. 50–75. doi.org/10.2218/pihph.1.2016.1693
Blythe, Richard A & William Croft. 2012. S-curves and the mechanism of propagation in language change. *Language* 88(2). 269–304.
Boll-Avetisyan, Natalie & René Kager. 2016. Is speech processing influenced by abstract or detailed phonotactic representations? The case of the Obligatory Contour Principle. *Lingua* 171. 74–91. doi:10.1016/j.lingua.2015.11.008.
Burzio, Luigi. 2007. Phonology and phonetics of English stress and vowel reduction. *Language Sciences* 29(2–3). 154–176.
Cheng, B & Dm Titterington. 1994. Neural networks: A review from a statistical perspective. *Statistical science*. 9 (1). 38–42. doi:10.1214/ss/1177010642.
Clements, G N. 1990. The role of the sonority cycle in core syllabification. In J Kingston & M Beckman (eds.), *Papers in Laboratory Phonology I: Between the grammar and the physics of speech*, 282–333. Cambridge: Cambridge Univ. Press.
Daland, Rober & Janet Pierrehumbert. 2011. Learning diphone-based segmentation. *Cognitive Science* 35. 119–155.
Denison, David. 2003. Log(ist)ic and simplistic S-curves. In Raymond Hickey (ed.), *Motives for Language Change*, 54–70. Cambridge: Cambridge University Press.
Dreiseitl, Stephan & Lucila Ohno-Machado. 2002. Logistic regression and artificial neural network classification models: A methodology review. *Journal of Biomedical Informatics*. 35(5-6). 352–359 doi:10.1016/S1532-0464(03)00034-0.
Dresher, B Elan & Aditi Lahiri. 2005. Main stress left in Early Middle English. *Historical Linguistics 2003: Selected Papers from the 16th International Conference on Historical Linguistics, Copenhagen, 11-15 August 2003.* 75–85.
Dressler, Wolfgang Ulrich, Katarzyna Dziubalska-Kołaczyk & Lina Pestal. 2010. Change and variation in morphonotactics. *Folia Linguistica Historica* 31. 51–68. doi:10.1515/flih.2010.003.
Dziubalska-Kołaczyk, Katarzyna. 2005. Phonotactics of consonant clusters in the history of English. In Antonio Bertacca (ed.), *Historical Linguistic Studies of Spoken English*, 3–21. Pisa:

University of Pisa: PLUS.
Ellison, T Mark. 1994. The machine learning of phonological structure. University of Western Australia.
Fix, Evelyn ; Hodges, Jr, J L. 1952. Discriminatory analysis – Nonparametric discrimination: Small sample performance. *Project No. 21-49-004, Report No. 11, Contract No. AF 41(129)-31, USAF School of Aviation, Randolph Field, Texas.*, 1–48.
Fogel, D B. 1991. An information criterion for optimal neural network selection. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 2(5). 490–7. doi:10.1109/72.134286.
Guo, Gongde, Hui Wang, David Bell, Yaxin Bi & Kieran Greer. 2003. kNN Model-Based Approach in Classification. *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE* 2888. 986–996. doi:10.1007/b94348.
Guy, Gregory R & Charles Boberg. 1997. Inherent variability and the obligatory contour principle. *Language Variation and Change* 9(2). 149–164. doi:10.1017/S095439450000185X.
Hand, David J. & Keming Yu. 2001. Idiot's Bayes – Not so stupid after all? *International Statistical Review* 69(3). 385–398. doi:10.1111/j.1751-5823.2001.tb00465.x.
Hogg, Richard M & Christopher B McCully. 1987. *Metrical Phonology: A Course Book*. Cambridge: Cambridge University Press.
Kiranyaz, Serkan, Turker Ince, Alper Yildirim & Moncef Gabbouj. 2009. Evolutionary artificial neural networks by multi-dimensional particle swarm optimization. *Neural Networks* 22(10). 1448–1462. doi:10.1016/j.neunet.2009.05.013.
Kohavi, Ron. 1995. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *Appears in the International Joint Conference on Articial Intelligence (IJCAI)*, 1–7. doi:10.1067/mod.2000.109031.
Kopaczyk, Joanna, Benjamin Molineaux, Vasilios Karaiskos, Rhona Alcorn, Bettelou Los & Warren Maguire. 2018. Towards a grapho-phonologically parsed corpus of medieval Scots: Database design and technical solutions. *Corpora* 13(2). 255–269.
Kotsiantis, S B. 2007. Supervised Machine Learning: A Review of Classification Techniques. *Informatica* 31. 249–268. doi:10.1115/1.1559160.
Kroch, Anthony. 1989. Reflexes of Grammar in Patterns of Language Change. *Language Variation and Change* 1. 199–244.
Kroch, Anthony, Beatrice Santorini & Lauren Delfs. 2004. Penn-Helsinki Parsed Corpus of Early Modern English. http://www.ling.upenn.edu/hist-corpora/.

Kroch, Anthony & Ann Taylor. 2000. Penn-Helsinki Parsed Corpus of Middle English. http://www.ling.upenn.edu/hist-corpora/.

Lass, Roger. 1992. Phonology and morphology. In Norman Blake (ed.), *The Cambridge History of the English Language*, 23–155. (The Cambridge History of the English Language). New York: Cambridge University Press.

LeCun, Yann, Yoshua Bengio & Geoffrey Hinton. 2015. Deep learning. *Nature* 521. 436–444. doi:10.1038/nature14539.

Manning, Christopher D. 2015. Computational Linguistics and Deep Learning. *Computational Linguistics* 41(4). 701–707. doi:10.1162/COLI_a_00239.

McCulloch, Warren S. & Walter H. Pitts. 1943. A logical calculus of ideas imminent in nervous activity. *Bulletin of Mathematics Biophysics* 5. 115–133. doi:10.1007/BF02478259.

Minkova, Donka. 1991. *The history of final vowels in English: The sound of muting*. Berlin: Walter de Gruyter.

Murata, Noboru, Shuji Yoshizawa & Shun Ichi Amari. 1994. Network Information Criterion—Determining the Number of Hidden Units for an Artificial Neural Network Model. *IEEE Transactions on Neural Networks* 5(6). 865–872. doi:10.1109/72.329683.

Pustejovsky, J & a Stubbs. 2013. *Natural language annotation for machine learning*. *Vasa*. doi:1332788036. http://it-ebooks.info/book/681/%5Cnpapers3://publication/uuid/906A922E-DE39-4CAB-8067-F222D065ACEF.

Ripley, Brian D. 1996. Pattern Recognition and Neural Networks. *Analysis*. Cambridge: Cambridge University Press. http://www.stats.ox.ac.uk/pub/PRNN/.

Ritt, Nikolaus. 1994. *Quantity adjustment: Vowel lengthening and shortening in Early Middle English*. (Cambridge Studies in Linguistics). Cambridge: Cambridge University Press.

Ritt, Nikolaus, Christina Prömer & Andreas Baumann. 2017. Evolution of Consonant Clusters in English (ECCE): a diachronic phonotactic database. Vienna: Department of English and American Studies, University of Vienna. ecce.acdh.oeaw.ac.at (30 October, 2017).

Schmidhuber, Jürgen. 2015. Deep Learning in neural networks: An overview. *Neural Networks* 61. 85-117. doi:10.1016/j.neunet.2014.09.003.

Starčević, Attila. 2006. Middle English Quantity Changes — Further Squibs. *The Even Yearbook* 7. 1–37.

Team, R Development Core & R R Development Core Team. 2017. R: A Language and Environment for Statistical Computing. *R Foundation for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. doi:10.1007/978-3-540-74686-7.

http://www.r-project.org.
Venables, W N & B D Ripley. 2002. *Modern Applied Statistics with S Fourth edition by*. *World*. Vol. 53. doi:10.2307/2685660. www.stats.ox.ac.uk/pub/MASS4/VR4stat.pdf.
Wang, William & James Minett. 2005. The invasion of language: Emergence, change and death. *Trends in Ecology and Evolution* 20. 263–269.
Zaki, Mohammed J & Wagner Meira. 2014. *Data mining and analysis: Fundamental concepts and algorithms*. New York: Cambridge University Press.