Unsupervised Cognate Identification with Variational Autoencoders

Marlon Betz

September 7, 2016

Contents

1 Introduction

Historical Linguistics investigates language from a diachronic perspective, i.e. it seeks to uncover the history of languages and the structure of the hidden forces that drive language change. Computational Historical Linguistics accordingly deals with computational methods to explore the history of languages and topics closely related to it, such as phylogenetic inferences of language families Bouckaert et al. (2012), migration of language speakers Gray et al. (2009), inferring lexical flows between languages Dellert (2015) or modeling sound change Bouchard-Côté et al. (2013).

On the other hand, deep neural networks have been proven to uncover latent features of data and use them for a variety of tasks, such as computer vision or various NLP tasks, e.g. Document classification, Sentiment Analysis or Speech synthesis.

Bayesian Hierarchical models have been proven to allow for a detailed analysis of data and testing hypotheses in the fields of Psycholinguistics. In fact, Bayesian models build the backbone for most phylogenetic surveys.

Over the past few years, deep neural networks have been interconnected with Bayesian methods in the form of Deep Generative Models. They combine the strength of deep networks to uncover latent features with the power of bayesian inference. Today, deep generative models are the main object of current AI research.

However, Computational Historical Linguistics has hardly been touched yet by the current Deep Learning boom (a notable exception is Rama (2016)). In fact, most computational models here are either adaptations or direct copies of those used for the description of biological data.

In this thesis, I will propose a view on cognate identification as latent representation learning. Coming from the idea that certain computer vision models are shown to generalize well how to rotate a three-dimensional object in an two-dimensional projection and to find latent linear dependencies between high-level features, this thesis proposes a model that treats cognates as the same latent objects just seen from the perspective of another language. Sound changes, as realization of such changed perspectives, can be described as linear dependencies over words embedded in a latent space. Here, differences in the actual phonological realization of a word only blur the underlying latent structure of it, which can be shared with other words that are etymologically related. That latent structure is sampled from a simple bayesian hierarchical model, which allows for bayesian inference modeling, such as setting a prior belief on what such a latent structure should look like.

The model described here is comparedly simple, as it is a direct adaptation of Variational Autoencoders, a quite recent deep generative architecture that combines non-linear dimensionality reduction with the strength of variational bayesian inference. It can be used as a basis for more detailed research on how language change can be model with deep learning tools.

I will first start by giving an overview of the problem of cognate identification and related fields of research. I will give a background on why cognate identification is important to discuss the historical connections between languages and further provide an overview on several established methods to detect cognates.

Then I will proceed to explain the motivations why variational autoencoders are helpful tools for cognate identification and modeling sound change in general.

I will then discuss the actual architecture of the inference model. This first covers a general overview on the components included in the model. I will then in detail look over all components in particular. This will first cover a discussion of different methods of phoneme vectorizations. This is followed by a general overview on autoencoders as non-linear dimensionality reduction architectures first and then a description of variational autoencoders in particular, which build the basis for the

model described here. I then come to the discussion of possible ways to cluster the words, i.e. to assign the actual inferred cognacy labels.

Then I will document how well those methods can be used to infer cognacy between words. I will compare the inferred labels with expert judgements first and then see how the inferred labels can infer language phylogenies, using both neighbor joining and established bayesian models of cognate evolution.

Finally, I will give a resume on the model described here.

2 Motivation

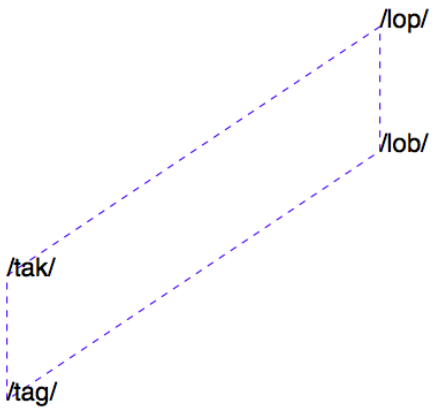

Figure 1: Visualization of the concept of sound change as a walk in latent space. Here, sound changes are vectors from one word to another, where both word forms are given as points in latent space. The vector from /lob/ to /lop/ would be the same as from /tag/ to /tak/, as both vectors would describe the loss of voice in the word-final phoneme. This linear dependence means that if we fit a regression model from /tag/ to /tak/, we could generalize well to predict /lop/ from /lob/. The different lengths of the vectors are than proportional to probabilities of such as sound change to appear. Here, final devoicing should be more probable than a change from /lop/ to /tak/.

In linguistic literature, sound change is usually described as a change of distinctive phonological features over phonological symbols. For instance, a sound change such as final devoicing as in

$$MHG/hund/ \rightarrow NHG/hunt/ \tag{1}$$

would be captured by an string edit rule like

$$/d/ \rightarrow /t//\_\# \tag{2}$$

that says that /d/ becomes /t/ before the end of a word. Such sound changes are shown not to appear randomly, but to co-occur according to some latent features. E.g. a sound change like above should leave the language without final /d/, while final /b/ and /g/ would be allowed. Such situations are evolutionary highly unstable, so we would expect the other voiced plosives to change to /p/ and /k/ accordingly. This is due to latent phonological features such as [VOICE], [LABIAL] or [ATR], which are usually described to appear either in a binary or privative feature space. Accordingly, our rule from above could be generalized to

$$[+\text{VOICE}] \rightarrow [\text{-VOICE}]/\_\# \tag{3}$$

which describes the loss of voice at the end of a word.

Computational models of sound change usually treat words as string of symbols and sound change as substitution of substrings.Bouchard-Côté et al. (2007, 2013) use a generative model that learns to substitute substrings in some latent order and achieve good performances on the reconstruction of ancient forms of words as well as on cognate identification. However, such a method has two major drawbacks: On the one hand, it needs the topology of the languages phylogeny to be known beforehand. This problem does not exist for certain language families with established tree topologies, but can be problematic if language relationships are not totally clear. On the other hand, such purely symbolic models need lot of data to generalize well. Bouchard-Côté et al. (2007) for example underline that their model cannot model chain shifts as such.

This thesis proposes a model that tries to solve the task of cognate identification (and sound change as such) under the following assumptions:

1. The functions connecting our data with some latent space should be smooth. This allows the model to generalize from training examples to nearby points in input space, even if they were not trained on.

2. The relationships between the data can be explained by some linear function, i.e. the data is linearly dependent. This should allow for good predictions even when the test data is far away from the training data.

3. The model should treat factors that change some latent representation as causes for variation of the actual data, not vice versa. That is, the model generates the latent representation.

4. The underlying linear factors that produce the data are to complex to model directly, but only exist in some highly-abstract latent space that is not directly visible.

To address the smoothness assumption, we have to make sure that such a model treats phonemes and words not as symbols, but as points or distributions in some latent space. This allows for some tokens to be inherently closer to each other. Section 4.1 will discuss several such embedding methods for phonemes and section 4.2 will explore variational autoencoders as the preferred embedding algorithms for words.

The linearity assumption allows for the generalization over the compositional character of latent features over phonemes and words. For instance, a sound change *sc* such as the final devoicing in Eq. 3 that derives a recent form of word w_{recent} from an ancient form $w_{ancient}$ should then correspond to a vector v_{sc} in such a way that

$$v_{w_{ancient}} + v_{sc} = v_{w_{recent}} \tag{4}$$

From this follows that

$$v_{sc} = v_{w_{recent}} - v_{w_{ancient}} \tag{5}$$

That is, we can formulate sound changes such as 3 without neither the actual sound change nor the conditioning specifying where the sound change should apply, but only the respective words involved. If we further assume that that sound change affects another word w', we have

$$v_{w_{recent}} - v_{w_{ancient}} = v_{w'_{recent}} - v_{w'_{ancient}} \tag{6}$$

which is equivalent to

$$v_{w_{recent}} = v_{w_{ancient}} + (v_{w'_{recent}} - v_{w'_{ancient}}) \tag{7}$$

In fact, such analogy tasks can be used as an evaluation method to test whether the learned embedding space encodes the structure expected to be inherently contained in the data (Mikolov, Sutskever, et al., 2013). For a visualization of this concept, see Fig. 1.

3 Related Research

Cognate identification systems can be divided into two major groups. Supervised systems learn from a set of positive and negative examples to differentiate between cognates and unrelated words. This is usually done with several forms of string comparisons, such as normalized levenshtein distance, Dice coefficients or alignment scores based on the Needleman-Wunsch algorithm (Bergsma & Kondrak, 2007; Inkpen et al., 2005), weighted alignments with predefined weights (Kondrak, 2000) or PMI-weighted alignments (Jäger, 2014). Rama (2016) further uses siamese convolutional networks to differentiate between cognates and unrelated words.

Unsupervised system also employ such string comparisons, but assume that cognates should share similar shapes and hence the distance between them should be low. Hence, after predefining some threshold, words with mutual distances beyond that threshold are labeled as unrelated, while words closer to each other are labeled as cognates. LexStat, originally proposed in List (2012a), can be seen as the gold-standard for unsupervised systems. LexStat first converts all phonemes into a list of features, then creates scoring schemes for all language pairs, computes pairwise distances between all words based on those scoring schemes and finally clusters the words.

4 Architecture

Inspired by the architecture of LexStat, the models described in this thesis will have a modular architecture of three components:

1. The phonemes of every word are embedded in a latent space, where similar phonemes should cluster in similar subspaces of the latent space. This should allow for better generalization of sound correspondences between phonemes of different languages, following the smoothness and linearity assumption from 2. Section 4.1 will cover this topic.

2. The words as sequences of such phoneme embeddings should themselves be embedded in another latent space, where words with similar shape should cluster among each other. Again, the main motivation here is smoothness and linearity, which should help to uncover latent

features. Moreover, this component contains the actual variation autoencoder, which samples embeddings from a latent variable and transforms them into actual words. This follows the other two motivations - the unidirectionality of causal factors and the overcomplexity over the true underlying factors that produce the data. Section 4.2 will cover this topic.

3. The word embeddings are then clustered in such a way that words that appear together in a cluster are assigned a common label, which is then the predicted cognate class. This follows again the smoothness assumption, as we expect words with similar latent features to cluster among each other. Here, it is important that the number and respective size of cognate classes per semantic concept can vary, so we have to find a way to cluster the words without any preference over the number of clusters or cluster size. Section 4.3 will cover this topic.

Moreover, we focus on two separate models:

1. A model that tries to maximize $P(X|z)$, i.e. that learns the manifold creating the words as such. Such as model is pre-trained on a big corpus of phonologically annotated words. Here , the assumption is that given a big corpus that is representative of the data as such, the model will recognize the latent features of *words as such* and can embed them in a meaningful even if it was not trained on them directly. This model will be referred to as VAE-PT.

2. A model that tries to maximize $P(X|z, C)$, i.e that learns the manifold creating words given the respective concept. Such a model is useful as we are only interested to cluster words given a concept class. Also, this means that we only look for latent features in a given semantic concept, which should allow for finer-grained latent features. This model will be referred to as VAE-C.

4.1 Phoneme Vectorization

Various computational models of sound change treat phonemes as purely symbolic, which means they are treated as equidistant to each other. While such models can achieve good performance on the task of modeling sound change (Bouchard-Côté et al., 2007, 2013), they cannot generalize to process data they have not seen during training. Such an ability only comes from the *smoothness assumption* that $f(x + \epsilon d) \approx f(x)$ for unit d and some small error ϵ (Bengio & Courville, 2016, p. 555). This assumption allows a model to generalize from training data to close poinst in input space. However, as long as phonemes are treated purely as equidistant symbols, any given trainable model should not be able to generalize to process unseen data. Moreover, it is established knowledge that certain phonemes share binary features such as *[velar]*, *[voice]* or*[ATR]* (Chomsky & Halle, 1968). Such latent features regulate how certain phonemes are distributed among each other - e.g. unvoiced consonants occur in word-final positions more often than voiced consonants. Hence they also influence the diachronic evolution of the phonological shape of a word - if a sound appears in a context that is not typical for its actual distribution, it either makes the context change or it changes itself to adapt to the context. For instance, palatal consonants often palatalize other surrounding consonants, while voiced consonants often devoice word-finally. Following that observation, there are a few models that use such features to differentiate between phonemes.

4.1.1 Hand-crafted Vectorization Models

Most hand-crafted vectorization models are used for weighted string alignments. Here, the distance between two phonemes is used yield an alignment score between the phonemes. Dolgopolsky (1986)

proposes a merger of all phonemes into a set of 10 phoneme groups, where transitions between members of a group are more likely than transitions between members of a single group. The groupings are based on an analysis of sound-correspondences between 400 languages of northern Eurasia (cf. (List, 2012a, p. 119)). This system has been used by several studies dealing with stochastical aspects of genetic relationships between languages (Baxter & Ramer, 2000; Mortarino, 2009; Turchin et al., 2010). An enhanced version of this system that is extended to cover all IPA characters and its most common diacritics as well as vowels builds the basis for LexStat (List, 2012b,a). To turn those sound classes into vectors that can be used as input for the variational autoencoder, each sound class is represented as a binary one-hot vector. Kondrak (2000) uses 12 real-valued features bounded between zero and one to allow for a weighted alignment of phoneme sequences. Here it is argued that binary features cannot capture the actual continuum e.g. between velar and uvular sounds. Each feature can moreover be weighted to allow for modeling the relative importance of a given feature.

Rama (2016), on the other hand, uses binary features that are adapted to the ASJP phonemic alphabet (cf. Wichmann et al. (2010)). ASJP does not always distinguish certain features for all phonemes. For instance, voice is distinguished for labial fricatives, but not for dental ones. ASJP further merges all clicks. Vowels are differentiated, but only for 5 prototypes. The whole ASJP phoneme set can be found in the appendix. Rama (2016) further merges all coarticulations and vowels, as they tend to be diachronically unstable, and yields phonemes with 16 binary features each. Rama (2016) reports superior performance using those features as a basis for word vectorizations to find cognates with siamese convolutional networks.

4.1.2 Phoneme embeddings

If we assume that phonemes do not change unconditionally nor randomly, but instead only change distinctive features given the context, it should be able to embed phonemes in a latent space where local subspaces contain clusters of phonemes that appear in similar environments. For instance, if we have the two Italian-Spanish word pairs /t͡ʃɛntɔ/ - /θi̯ento/ "hundred" and /t͡ʃɛlɔ/ - /θi̯elo/ "sky", we see that /t͡ʃ/ appears in a similar context as /θi̯/. If we should find more such cases in our data, it should be possible to embed them close in some latent space, as we assume that points that are close to each other in the latent space should also show similar behavior in the data space. In fact, /t͡ʃ/ and /θi̯/ are renderings of a common Vulgar Latin /ki̯/ - hence we can assume that phonemes with similar embeddings share a tendency to develop from one into the other. If we interpret each dimension of an embedding as a value of a latent feature, we expect to yield similar features as those hand-crafted vectorization methods proposed above.

There are several families of algorithms that perform such an embedding. Earlier models are based on factorized co-occurrence matrices, such as Latent Semantic Analysis Landauer et al. (2013). This approach is inherently intuitive, as the factorized context of a given phoneme would then be taken as point in latent space, so similar contexts than inherently lead to proximity in latent space. However, over the past few years, more recent neural embedding models such as word2vec Mikolov, Chen, et al. (2013); Mikolov, Sutskever, et al. (2013); Goldberg & Levy (2014) have been shown to outperform those count-based models, although GloVe Pennington et al. (2014), as more recent count-based model, seems to achieve similar performance.

Word2vec is a shallow network that consists of a linear encoder and softmax classification layer. There are two architectures: Continuous Bag Of Words (CBOW), that learns to predict a token given its context, and Skip-Gram (SG) that is trained to predict the context given the token (cf.

Fig. 2). They both have in common that they assume the token and its context to occupy the same location in latent space. This allows for a compositionality of the learned latent features. For instance, to re-use the example from above, the embeddings for /θ / and /i̯/ are not expected to be close in the latent space, as the overall contexts they can appear in should be quite different. However, given that they appear together as /θi̯/, the addition or mean of the respective embeddings should be close to each other.

As traditional softmax classification is computaltionally expensive expensive given a larger number of token types, several alternative classifiers have emerged during the past few years. One is Hierarchical Softmax (HS, Morin & Bengio (2005)). Here, the classifier layer is a binary tree with the actual token types as leaves. At every node, the network learns to either follow the left or the right branch with a certain probability as in Eq. **??** that is equal to the sum of the child elements of the respective branch, but is actually computed as the product of $h^\top$ and the output vector of the word at node n pulled through a logistic sigmoid:

$$p(right|n, c) = \sigma(h^\top v'_n) \tag{8}$$

This means that in order to calculate the softmax probability of word, we only have to follow the path down to the word leaf instead of summing over all possible token types. For binary trees, this means we only have to pass at most $\log_2(\mid V \mid)$ nodes to calculate the probability of a word, which is a huge performance boost over the traditional softmax classifier.

Another family of alternative classifiers use sampling-based approaches. Among them, Negative Sampling (NEG; cf. Goldberg & Levy (2014)) is the most popular. NEG is originally derived from Noise Contrastive Estimation (cf. Gutmann & Hyvärinen (2010); Mnih & Teh (2012)). Instead of a classifier as such, NEG samples random tokens from the available token types and learns to distinguish them from the actual tokens that appear in a context. The loss function would then be

$$J_\theta = -\sum_{w_i \in V} [\log \sigma(h^\top v'_w) + \sum_{j=1}^{k} \log \sigma(-h^\top v'_{\tilde{w}_{ij}})] \tag{9}$$

where k is the number of sampled tokens and $v'_{\tilde{w}_{ij}}$ the vector representation of such a sampled word.

A first visual inspection shows that the phoneme embeddings indeed capture some latent information (Fig **??** and 3). Broader natural classes such as vowels cluster among each other, clearly differentiated by their type of articulation, such as nasalization or glottalization. In general, sounds that share a common coarticulation are embedded close to each other. Velar and uvular sounds are also closer to nasalized vowels than plain vowels, as both involve an attracted tongue root during production. Less frequent phonemes, however, cannot really be embedded in a meaningful way.

To evaluate further if such data driven phoneme embeddings are able to capture the natural distinction between phoneme classes, a test set over 108 analogy tests was used to conduct a grid search over several word2vec architectures and their parameters. Each such analogy test was of the form

$$v(phoneme_1) + v(phoneme_2) - v(phoneme_3) \approx v(phoneme_4) \tag{10}$$

where $v(phoneme_1)$ is the vector corresponding to $phoneme_1$ and $v(phoneme_2) - v(phoneme_3)$ can be seen as the latent phonological information available in $phoneme_2$ but not in $phoneme_3$, while $v(phoneme_1)+v(phoneme_2)-v(phoneme_3)$ is then this latent phonological information added to $phoneme_1$, which should be close to the vector corresponding to $phoneme_4$. For example, in

$$v(/\mathbf{i}/) + v(/\mathbf{u}*/) - v(/\mathbf{u}/) \approx v(/\mathbf{i}*/) \tag{11}$$

$v(/\mathbf{u}*/) - v(/\mathbf{u}/)$ describes a latent feature that should correspond to [+NASALIZED], which is then added to the phoneme **/i/** to yield a nasalized **/i*/**. The grid search was conducted over the following parameters:

- Model architecture: CBOW, SG
- Classifier: Softmax, negative sampling
- Embedding dimensions: $[2, 5, 10, 20, 50, 100, 150, 200, 400]$
- Context window size: $[1, 2, 3, 4, 5, 8]$
- Number of negative samples: $[1, 2, ..., 20]$

Each model iterates five times over the corpus, which consists of all words from the ASJP corpus given the language they belong to is neither artificial nor reconstructed. Every word is bracketed by word boundary tokens.

Fig. 5 shows that the quality varies significantly with regards to the articulation type of the phonemes. For pulmonic consonants the models achieve reasonable performance, while for other coarticulation types the performance is quite bad. When following Jäger (2014); Rama (2016) and pooling coarticulation types, the models perform much better, for analogies over plosives, while the performance for fricatives has only slightly improved.

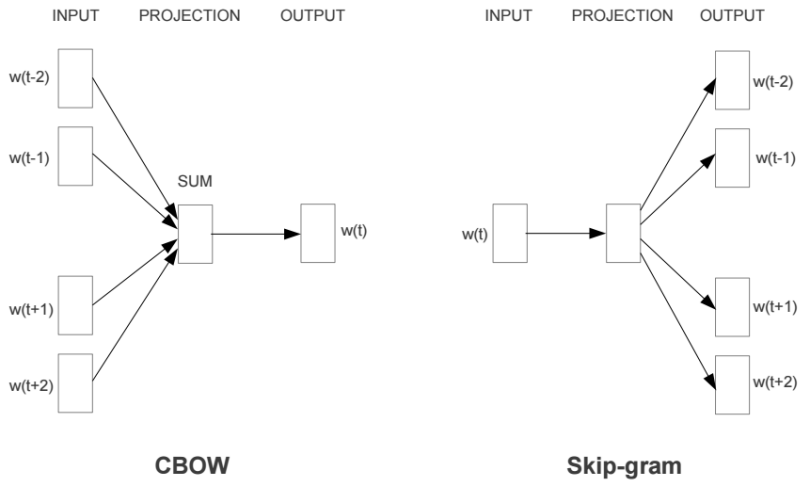

Figure 2: The two common architectures of word2vec. The Continuous Bag-of-Words (CBOW) model predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word. From Mikolov, Chen, et al. (2013).

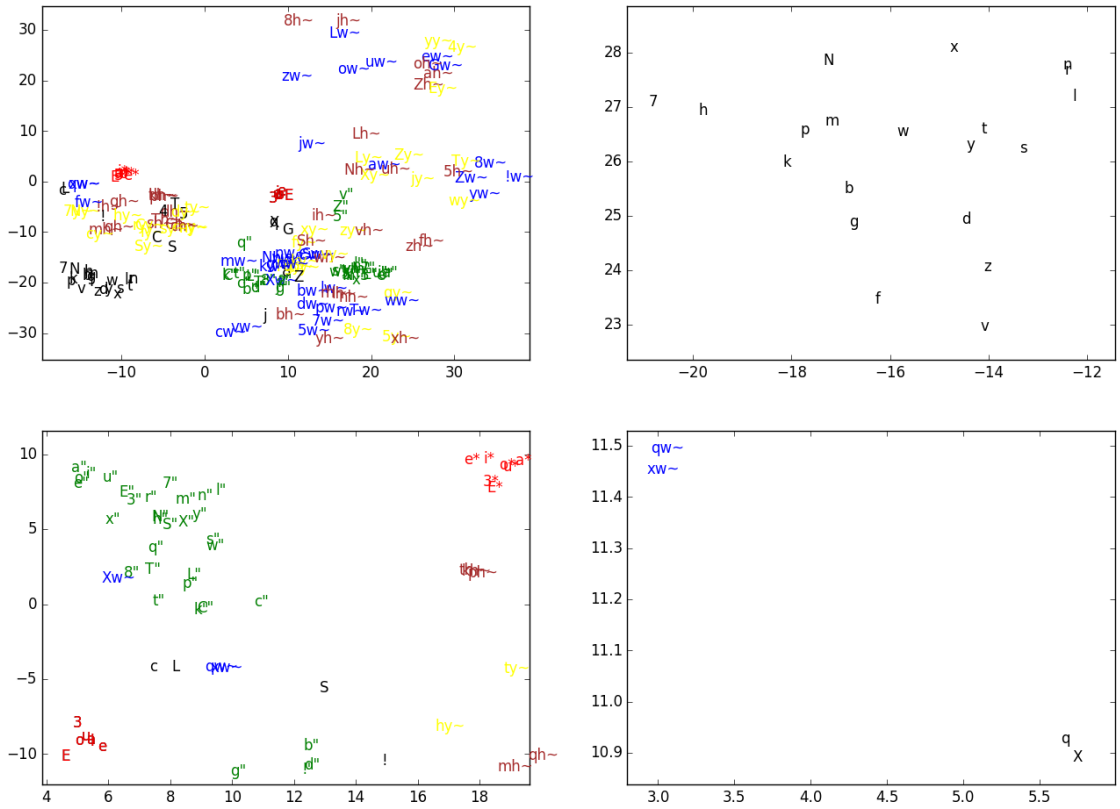

Figure 3: Some t-SNE visualizations of the embeddings created by word2vec. (top left) The model learns to clearly separate natural classes such as vowels, plain pulmonic or glottalized consonants, while other articulations seem to spread over the feature space. The colors indicate membership of a natural phonological class. (top right) A more detailed view on plain pulmonic consonants. Note the linear dependencies between voiced and unvoiced plosives and their respective nasal variant. (bottom left) Another detailed view. Note how the labialized uvular sounds cluster among glottalized consonants. (top right) The model seems to capture different manners of articulations across articulation type boundaries, as the linear dependency shows here.

4.2 Word Embeddings

4.2.1 Autoencoders

Autoencoders are a family of neural network architectures that are trained to construct a code of some given data. Given an encoder function $f : \mathcal{X} \to \mathcal{Z}$ and a decoder (i.e. generator) function $g : \mathcal{Z} \to \mathcal{X}$, they are trained on encoding and reconstructing the data, yielding a loss function

$$J(x, g(f(x))) \tag{12}$$

where J describes some loss function that penalizes the reconstruction error. The dimensionality of $\mathcal{Z}$ is usually assumed to be lower-dimensional, so that autoencoders can be used to reduce the

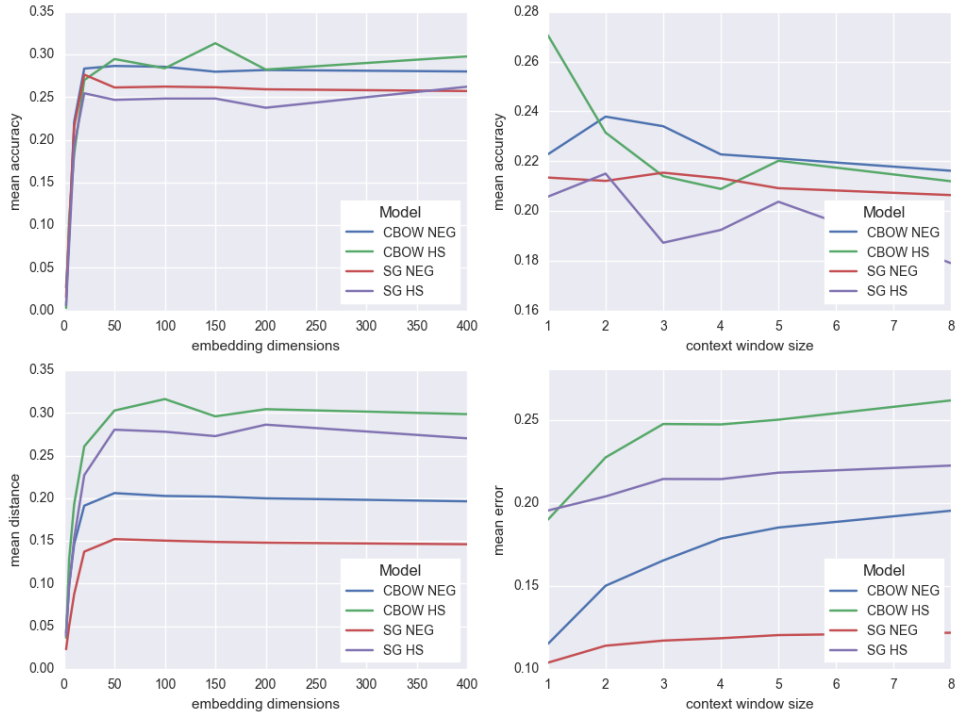

Figure 4: Comparison of the four word2vec models evaluated. (top left) Both CBOW Models perform better than the skip-gram models over all numbers of embedding dimension. (bottom left) Mean distance between the predicted vector and the target with regard to embedding dimensions. Here, negative sampling yields less error than hierarchical softmax. (top right) Mean accuracy for the models with regard to the context window size. The models perform worse the bigger the context is. (bottom right) Mean distance between the predicted vector and the target with regard to context window size. Again, bigger contexts lead to worse predictions.

dimensionality of the data, although sparse autoencoders use higher-dimensional hidden layers with some sparsity constraint. If f and g are linear functions, autoencoders are proven to be equal to PCA. If they are non-linear, they allow for the recognition of latent features (Hinton & Salakhutdinov, 2006). Autoencoders are used in various NLP fields, such as Machine Translation, (Lauly et al., 2014; Zhang et al., 2014), Paraphrase Detection (Socher et al., 2011) or Information Retrieval (Silberer & Lapata, 2014; Le & Mikolov, 2014). Also, word2vec (Mikolov, Chen, et al., 2013; Mikolov, Sutskever, et al., 2013; Goldberg & Levy, 2014) as the probably most popular embedding algorithm is in fact a shallow autoencoder.

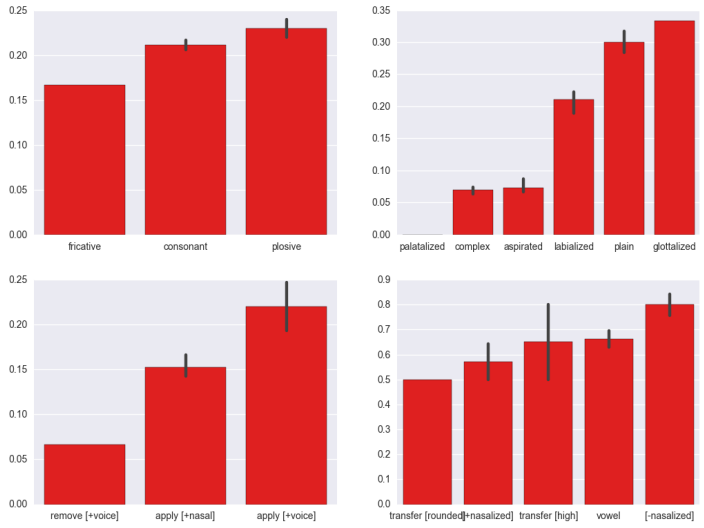

Figure 5: Mean accuracies for analogies. The model does not seem to be able to capture the compositionality of the latent features too well. (top left) Accuracy is best for plosives across plain pulmonic as well as more complex articulations, while fricatives perform worse. (top right) Glottalized and Plain pulmonic consonant phonemes yield best performance. Among complex articulations, which all perform bad, labialized phonemes yield best results, while aspirated and palatalized phonemes perform even worse. (bottom left) Adding voice works better than removing it or adding nasality. (bottom right) Vowel analogies work far better than consonant analogies. This should be due to the small number of possible vowel phonemes.

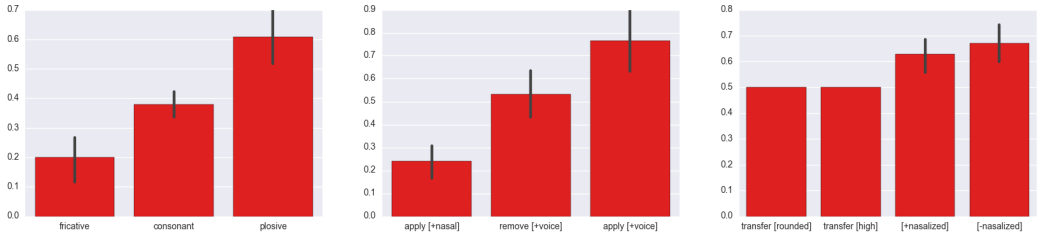

Figure 6: Mean accuracies for analogies over the pooled consonant phonemes. Here, the model is expected to yield better results, as the number of latent features should have been reduced. (left) Accuracy is doubled compared with the unpooled phonemes. (middle) Adding voice works quite well, while removing it still yields acceptable performance. Applying nasality again works quite bad. (right) Vowel analogy tasks seem to work reasonably well, but worse than with unpooled consonants.

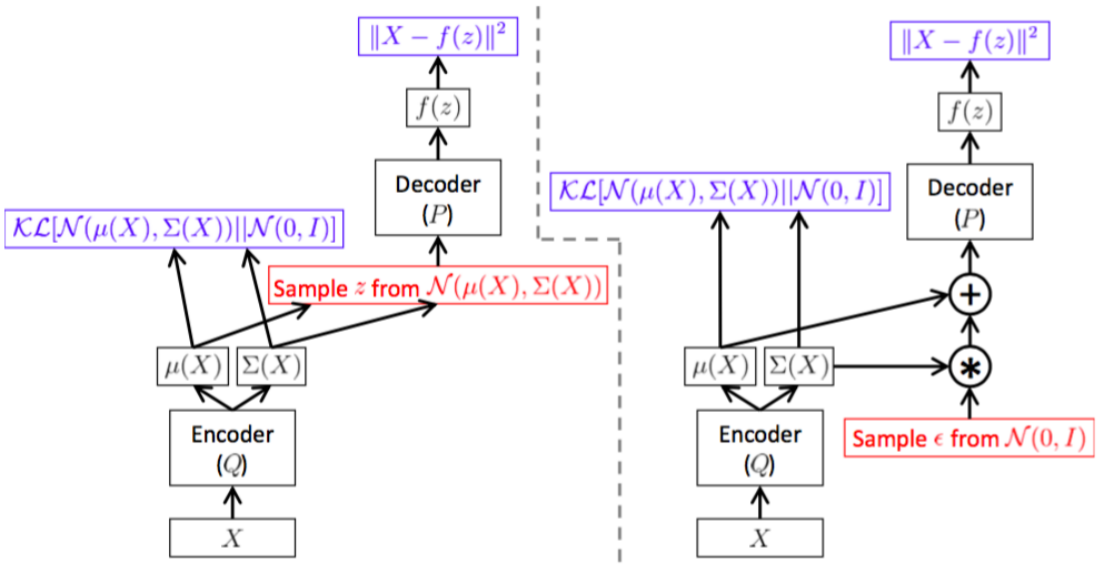

Figure 7: Visualization of the Variational autoencoder architecture. (left) The model with original objective as in Eq. 15. The stochastic unit is inside the network and would not allow for the back-propagation of error gradients through the network. (right) The model after the reparameterization with the objective as in Eq. 23. Here, the sampling is interpreted as an input variable, so error gradients can be backpropagated through the whole model. From Doersch (2016).

4.2.2 Variational Autoencoders

Variational Autoencoders (VAEs) were first described in Kingma & Welling (2013). Instead of just using a lower-dimensional layer as a bottle neck for non-linear dimensionality reduction, VAEs assume that the code can be modeled by a latent variable z with some prior $p(z)$. This allows for the incorporation of bayesian inference tasks where the marginalization of variables is important, including maximum a posteriori estimates of θ and an approximation of the posterior $P(z|X)$. Instead of encoder and decoder functions, distributions are used instead to model the relationship between $\mathcal{X}$ and $\mathcal{Z}$.

As variational models, VAEs do not attempt to sample from the true posterior distribution $P(z|X)$ directly. Popular Markov Chain Monte Carlo methods, such as Gibbs sampling (Geman & Geman, 1984) or Hamiltonian Monte Carlo (Duane et al., 1987), can approximate the true posterior, but rely on whole batches during training, which is not suitable for bigger data sets. Instead, variational method assume that the true posterior can be approximated by another parametric distribution $Q(z|X)$ whose parameters can be estimated analytically.

In order to approximate the true posterior $P(z|X)$ with the approximate posterior $Q(z|X)$, we want to minimize the Kullback-Leibler divergence from the true posterior to its approximation:

$$D_{KL}(Q(z|X)||P(z|X)) = E_{z\sim Q}[\log Q(z|X) - \log P(z|X)] \tag{13}$$

Here, we can use Bayes' rule to yield

$$D_{KL}(Q(z|X)||P(z|X)) = E_{z\sim Q}[\log Q(z|X) - \log P(X|z) - \log P(z)] + \log P(X) \tag{14}$$

where $\log P(X)$ is outside the expectation since it is not dependent on z. We can rearrange that to

$$\log P(X) - D_{KL}(Q(z|X)||P(z|X)) = E_{z\sim Q}[\log P(X|z) - D_{KL}(Q(z|X)||P(z))] \tag{15}$$

Here, the left hand side makes it clear why Eq. 15 is to be maximized, as it increases the probability of X and minimizes the divergence from the true to the approximated posterior. We further have $Q(z|X)$ as an *encoder* that maps data points into the latent space $\mathcal{Z}$ and $P(X|z)$ as a *decoder* that reconstructs a data point, given a sample z. As the universal approximation theorem states that networks with at least one hidden layer can approximate any continuous functions, we can reduce $D_{KL}(Q(z|X)||P(z|X))$ to zero and optimize $P(x)$ directly. The right hand side is the *variational lower bound* that defines the probability that the encoder distribution $Q(z|X)$ approximates the true prior $P(z)$ and then z is decoded back to it initial value in $\mathcal{X}$. Also, since we reduce both $D_{KL}(Q(z|X)||P(z|X))$ and $D_{KL}(Q(z|X)||P(z))$ during training, our encoder distribution $Q(z|X)$ becomes equal to the prior if X is equal to the training data.

As we model both prior and posterior with gaussians, we can calculate $-D_{KL}(Q(Z)||P(Z))$ directly as

$$\begin{aligned} -D_{KL}(Q(Z)||P(Z)) &= \int Q(z)(\log P(z) - \log Q(Z))dz \\ &= \frac{1}{2}\sum_{j=1}^{J}(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2) \end{aligned} \tag{16}$$

To estimate these parameters μ and σ of Q, Kingma & Welling (2013) propose Multilayer Perceptrons (MLPs) as universal function approximators. In fact, various other encoders and decoders have been proposed, such as convolutional or recurrent networks (Kulkarni et al., 2015; Bowman et al., 2015; Fabius & van Amersfoort, 2014). In the case of binary phoneme features, our decoder distribution is defined as a multivariate Bernoulli, where each probability is calculated by a MLP:

$$\log P(x|z) = \sum_{i=1}^{D} x_i \log y_i + (1 - x_i)\log(1 - y_i) \tag{17}$$

where x is a given datapoint in X and D is the number of dimensions of x. y is the output of of the MLP, defined as

$$y = f_\sigma(W_2 f(W_1 z + b_1) + b_2) \tag{18}$$

where f_σ is the elementwise sigmoid activation fuction, f some other non-linear activation function, W_1,W_2 weights and b_1,b_2 biases of the network.

The phoneme embeddings can be scaled between $[0, 1]$, so that every dimension becomes a binary latent feature. The rescaled location in that dimension then becomes the possibility of that phoneme having that feature.

In order to arrive at the final objective function J_θ, we take the cross entropy between the true phoneme features and the predictions as reconstruction error and the KL-Divergence from Eq. 15 as a regulization term:

$$J_\theta = \sum_{i=1}^{D} x_i \log y_i + (1 - x_i) \log(1 - y_i) + \frac{1}{2} \sum_{j=1}^{D_\mathcal{Z}} (1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2) \tag{19}$$

where D is the dimensionality of the original data and $D_\mathcal{Z}$ the dimensionality of the latent space.

However, given that we use neural networks, we have to make sure that all gradients can be backpropagated through the network. In Eq. 15, moving the gradient symbol into the expectation on the right hand side would yield gradients that would be independent of the parameters of Q. This is due to the location of the sampler as a unit inside the network - since sampling from $Q(z|X)$ is a non-continuous operation, it cannot have gradients and hence would block the backpropagation of errors to the encoder network. Kingma & Welling (2013) instead propose another objective that treats the sampling as an input to the network: Here, an auxiliary noise variable ϵ is sampled from some distribution $p(\epsilon)$:

$$\epsilon \sim p(\epsilon) \tag{20}$$

Kingma & Welling (2013) mention that $p(\epsilon)$ can be any continuous distribution or compositions of such distributions, but propose to use a standard normal distribution if z is model by a normal distribution:

$$\epsilon \sim \mathcal{N}(0, 1) \tag{21}$$

With ϵ at hand, we can reformulate z as a deterministic variable:

$$z = \mu + \sigma \odot \epsilon \tag{22}$$

where $\odot$ defines element-wise multiplication. The right hand side of Eq. 15 then becomes

$$E_{\epsilon \sim \mathcal{N}(0,1)}[\log P(X|z = \mu + \sigma \odot \epsilon) - D_{KL}(Q(z|X)||P(z))] \tag{23}$$

The model architecture is visualized in Fig. 7.

4.2.3 Interpretation of $\mathcal{Z}$

The question is then how to interpret such a latent space $\mathcal{Z}$. In the case of VAE-PT, where we train the model on a big corpus, we assume it encodes the inherent structure of words as such. As the latent variable z walks through $\mathcal{Z}$, we assume that the decoded word changes in a *meaningful* way - for instance it should change certain phonological features of some phonemes in x given that such changes are plausible, given the context of the other phonemes in the words. It should also be possible to add whole new affixes, but it should not inherit implausible phoneme sequences or change phoneme features in some random way. The multi-modality of our encoder distribution $Q(z|X)$ leads to the situation that similar points in $\mathcal{Z}$ should not necessarily coincide with similar points in our data space $\mathcal{X}$. For instance, a fictional word such as /aban/ should be closer to /ovã/ than to /a‡an/ in $\mathcal{Z}$ but not necessarily in $\mathcal{X}$, as it should be more probable that intervocalic voiced plosives lenite to fricatives and syllable-final nasals drop but leave some compensatory nasalization

- here even at the same time - but there should be hardly any evidence of a direct sound change from /b/ to a palatal click. To arrive at such linear dependencies over sound changes as in Eq. 4, we then just have to expect them in our data X in some latent way. As X in the decoder distribution $P(X|z)$ would be conditioned on z, we expect the same linear dependencies over words in z as we find them in X.

In the case of VAE-C, where we train the model only on a handful of words, we do not expect strong generalization abilities. Instead, we expect the model to focus on latent features important to distinguish the words trained on, and only those words. This means that the model severely overfits to the data, but since we are only interested in the embeddings of the training data, this concern is only secondary. As we expect that certain features are more salient than others , we can tell the model to focus on those by increasing the standard deviation of the auxiliary noise σ_ϵ. As σ_ϵ increases, it becomes harder for the model to find less frequent latent features, and it focusses instead to encode the more salient ones. This means that we can use σ_ϵ as parameter of the model that corresponds to the expected salience of the cognate classes to be found: If there is strong evidence for several words to be cognates to each other as they all share similar shapes, while the rest of the words trained on are quite different to each other, we expect the model to cluster those similar words at one subspace and all the others at some other subspace, as a bigger auxiliary noise hinders the model to generalize the latent features of the less frequent words. On the other hand, if σ_ϵ is small, we focus on the discovery on all latent features in the data, allowing for finer-grained clustering.

4.3 Clustering

To cluster the embeddings, we have to make sure that the number of clusters can vary, as can the number of members for each cluster. For instance, if our word list consists of languages where the majority of languages are closely related and should show similar words, but the remaining languages are not related at all, we assume that the resulting clusters of words are not of the same size. If we do not know at all if the respective languages are related, we cannot really say what size the clusters should have either. Hence, both are variables which we do not know beforehand. LexStat, as the best current model for unsupervised cognate identification, makes use of the UPGMA clustering algorithm citepsokal1958statistical.

This thesis proposes Affinity propagation (Frey & Dueck, 2007) to solve that task. Here the algorithm exchanges real-valued messages between the data points until a high-quality set of exemplars and corresponding clusters gradually emerges. This allows for the number of clusters and individual cluster size to be left undefined. The algorithm can take any pair-wise similarities as input. In the case of euclidean distance as the preferred measure, the authors suggest taking the the negative squared euclidean distance, such that higher distance corresponds to lower similarity. There are two types of messages: the responsibility $r(i, k)$ sent from a data point i to a candidate exemplar point k encodes the evidence that k can serve as an exemplar for i. The availability $a(i, k)$, which is sent from k back to i, encodes the evidence that k should be the exemplar for i, taking into account all other points that choose k as an exemplar.

5 Evaluation

5.1 Data

To pre-train VAE-PT, the data provided by the Automatic Similarity Judgement Program (ASJP, Wichmann et al. (2010)) will serve as a training corpus. ASJP provides wordlists of roughly 7200 languages, where each word is given as a string of phonemes. This vast amount of different phoneme inventories will allow for the unbiased comparison of phonemes over language boundaries. However, ASJP does not make use of IPA, but a simplified phonemic alphabet that is based on ASCII-characters. This makes it possible to incorporate data where no detailed IPA description is given, but might also reduce the quality of the resulting embedding space, as latent features that might be encoded in more specific IPA descriptions are merged. For all other data, respective conversions into the ASJP system are used. An overview of the ASJP phonemes and their IPA counterparts are given in the appendix.

VAE-PT is trained and tested on IELex (Dunn, 2012), while VAE-PT is tested on it. IELex provides 250 concepts for 163 doculects. After the removal of ancient language forms and legacy doculects, this leaves around 60 languages, as the number of languages varies per concept.

5.2 Visual Inspection

A first visual inspection shows that VAE-PT, trained for 100 iterations, learns to cluster similar strings close to each other (Fig. 8). Words with similar syllable structures are located in specific subspaces of $\mathcal{Z}$. It further connects strings beyond local clusters through linear dependencies, which can interchange whole syllables of a word. On a more global perspective, words are embedded into a high-level feature continuum that encodes word lengths (Fig. 9), with smooth transitions between the respective subspaces. This is remarkable, given that there are no word boundary tokens involved during training. The model itself learns to recognize the transition from the binary code of the actual phonemes of the word to the zero-padding following it. Moreover, the linear dependencies are not simple renderings of the binary input code. The model learns high-level latent features, such as if a word has a phoneme with a certain distinctive feature or not (Fig. 10). If so, linear dependencies connect such words, even over very long distances. In fact, some of those linear dependencies form bigger clusters that can span the entire posterior distribution. Even more remarkable is that the model learns to cluster words with similar coarticulation features *even though they were excluded during training*, but only existent on some latent level (Fig. 11). Here, the model learns to distinguish plain /w j h/ from true labialization, palatalization and aspiration, although they were merged in the input code. If the model treated them as equal, we would expect a uniform distribution over the respective words. However, we see that the words accumulate in different subspaces. Since the model learns to cluster words according to their phonological structure, it recognizes that the context plain /w j h/ occur in differs from the context of coarticulated phonemes and hence locates the respective words in different subspaces of $\mathcal{Z}$.

The posterior distributions for several IELex concepts show linear dependencies for members of the same cognate class (Fig. 12). Hence, if the given semantic concept only consists of members of a single cognate class, the whole posterior shows strong linear dependencies (Fig. 12 top left). However, this focus on linear dependencies as a means to encode latent information makes it hard to cluster cognates, which rely on distances or similarities between the embeddings. In fact, the posteriors for several cognate classes are often intertwined.

VAE-C, on the other hand, yields a far better basis for cognate identification as such. Similar words form clusters that are pushed away from the population mean, while words without similar counterparts remain close the center of the latent space. Although there are linear dependencies between similar words, it does not play such an important role as it does for VAE-PT.

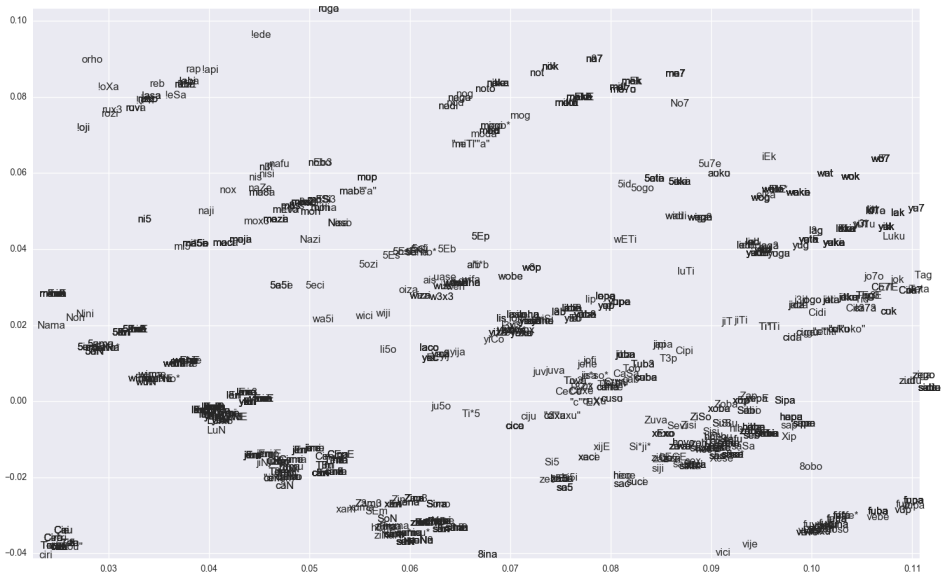

Figure 8: A detailed view on a subsample of ASJP embedded into $\mathcal{Z}$ after trained on the whole ASJP data set for 100 iterations. As can be seen, words accumulate in local subspaces with higher probability mass. Here, all words are contained in an area with shorter words that show simple CV syllable structures. Words that only differ in one distinctive feature are very close to each other, while linear dependencies over longer distances signify relationships over whole syllables that are exchanged.

5.3 Grid Search over Model Parameters

To explore the performances of VAE-C under different parameter settings, a grid search over the following parameters was performed:

- Phoneme vectorization:
  - Dolgopolsky one hots (cf. Dolgopolsky (1986); List (2012b))
  - SCA one hots (cf. List (2012b))
  - Binary ASJP phoneme features (cf. Rama (2016))
  - Phoneme embeddings, as explored in 4.1.2, i.e. 20 embedding dimensions, CBOW, HS, trained on ASJP with all coarticulations removed.

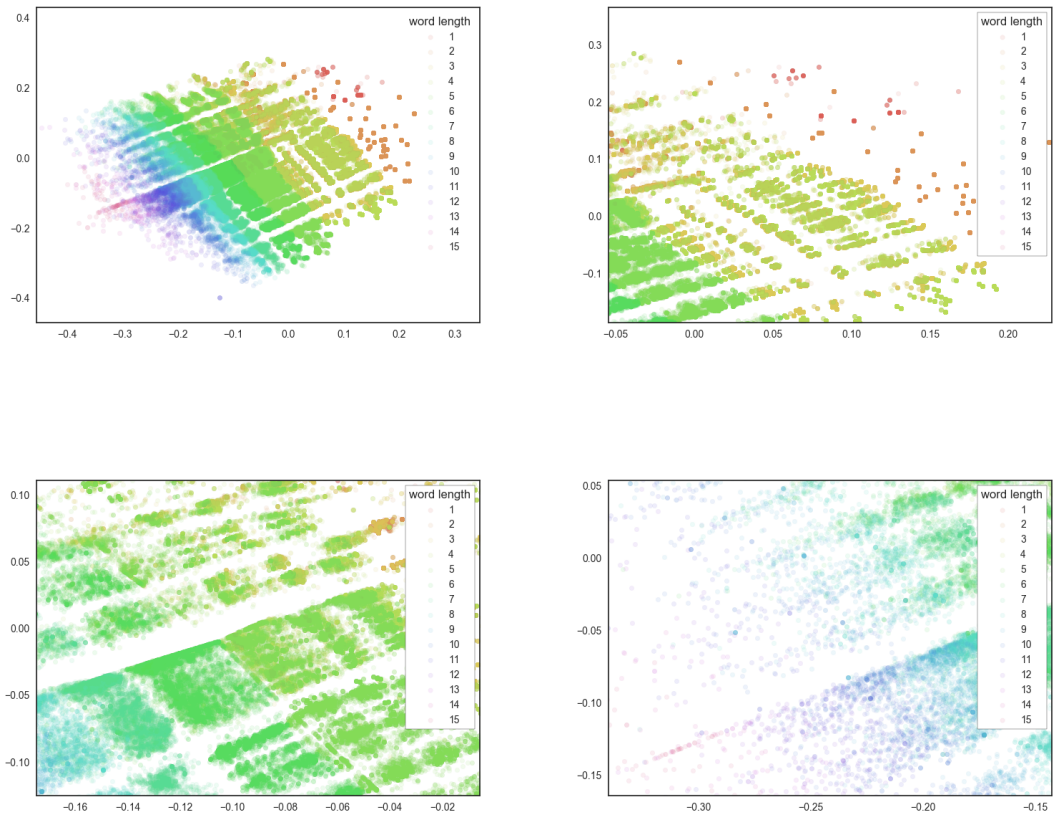

Figure 9: The ASJP data set embedded into $\mathcal{Z}$ after trained on it for 100 iterations, colored by word length. The model learns to cluster words according to their respective length, with more frequent word lengths located in subspaces with higher probability mass.

- number of latent dimensions: [10,20,50]
- the standard deviation of the auxiliary noise σ_ϵ: [0.1,0.01,0.001]

For all settings, the models were trained on a subset of 10 concepts, which are randomly sampled from all IELex concepts beforehand. The models are trained for 2000 iterations. The performance is measured in adjusted rand indices (Rand, 1971; Hubert & Arabie, 1985). Fig. 14 summarizes the results. The binary phoneme features outperform the word2vec phoneme embeddings. Moreover, more than 20 latent dimensions do not improve the performance. As the intermediate dimensions of the encoding and decoding MLPs increase, also the performance improves. The standard deviation of the auxiliary noise ϵ also improves the performance when decreased.

	ARI	AMI	Homogeneity	Completeness	V-Measure
VAE-C	0.32	0.36	0.67	0.57	0.57
LexStat	**0.59**	**0.71**	**0.99**	**0.92**	**0.95**
NLD-AP	0.38	0.44	0.77	0.61	0.64
Random	0.00	0.00	0.42	0.30	0.32

Table 1: The performance of VAE-C compared to LexStat. NLD-AP uses Normalized Levenshtein Distances as distance metric and Affinity Propagation as clustering algorithm. VAE-C performs only mediocre, but at least better than random labeling.

5.4 Performance on IELex

VAE-C was then run on the IELex data with the parameters found optimal in the grid search above, i.e. binary phoneme features, 20 latent dimensions, 500 intermediate dimensions and $\sigma_\epsilon = 0.01$. Fig. 15 summarizes the results, measured in adjusted rand indices, adjusted mutual information (Vinh et al., 2010), cluster homogeneity, cluster completeness and v-measure (Rosenberg & Hirschberg, 2007). The model indeed finds cognates and is far better than random labeling, but the overall performance is only mediocre, especially compared to LexStat (See table **??**)

5.5 Discussion

VAE-PT shows extraordinary performance in uncovering latent features. It can recognize the compositional structure of words, as it learns to form linear dependencies over distinctive phonological features, whole phonemes or even syllables. However, most information is encoded by such dependencies which is not suitable for flat clustering methods, where we expect that the location of a data point can be used to compare it with other data points. Although the smoothness assumption holds for VAE-PT, as we end up with clusters of very similar words, only a handful of actual cognates can be found in those. Following this, the performance of the model is quite bad, only slightly better than random labeling.

VAE-C however, performs much better than VAE-PT. It overfits to the training data, but by that encodes as much latent information as possible. Words with similar phonological structures cluster among each other. Although an increased auxiliary noise has its theoretical motivations, bigger values of σ_ϵ rather decrease the overall performance of the model.

6 Resume

In this thesis, I have shown how recent tools from the emerging deep learning framework can be used for unsupervised cognate identification. Although the two models described here do not perform better than well-established methods on the task of cognate identification as such, they allow for great introspection on the compositional structure of words and the linear factors that create it. Especially when trained on a big corpus, the model can uncover very abstract features and hence could be a basis for more research on how and why certain words evolve differently than others.

7 Acknowledgements

For training the phoneme embeddings, I used the word2vec implementations provided by the gensim package (Řehůřek & Sojka, 2010). The Autoencoder was implemented with Keras (Chollet, 2015), Tensorflow (Abadi et al., 2015) and theano (Theano Development Team, 2016). For converting phonemes into the Dolgopolsky and SCA format, I used the lingpy library (List & Forkel, 2016). The clustering algorithms used here were provided by scikit-learn (Pedregosa et al., 2011). All code connected to this thesis can be found on my github [1].

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `http://tensorflow.org/` (Software available from tensorflow.org)

Baxter, W. H., & Ramer, A. M. (2000). Beyond lumping and splitting: probabilistic issues in historical linguistics. *Time depth in historical linguistics*, *1*, 167–188.

Bengio, I. G. Y., & Courville, A. (2016). *Deep learning.* Retrieved from `http://www.deeplearningbook.org` (Book in preparation for MIT Press)

Bergsma, S., & Kondrak, G. (2007). Alignment-based discriminative string similarity. In *Annual meeting-association for computational linguistics* (Vol. 45, p. 656).

Bouchard-Côté, A., Hall, D., Griffiths, T. L., & Klein, D. (2013). Automated reconstruction of ancient languages using probabilistic models of sound change. *Proceedings of the National Academy of Sciences*, *110*(11), 4224–4229.

Bouchard-Côté, A., Liang, P., Griffiths, T. L., & Klein, D. (2007). A probabilistic approach to diachronic phonology. In *Emnlp-conll* (pp. 887–896).

Bouckaert, R., Lemey, P., Dunn, M., Greenhill, S. J., Alekseyenko, A. V., Drummond, A. J., ... Atkinson, Q. D. (2012). Mapping the origins and expansion of the indo-european language family. *Science*, *337*(6097), 957–960.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., & Bengio, S. (2015). Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Chollet, F. (2015). *Keras.* `https://github.com/fchollet/keras`. GitHub.

Chomsky, N., & Halle, M. (1968). The sound pattern of english.

Dellert, J. (2015). Uralic and its neighbors as a test case for a lexical flow model of language contact.

Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

[1] https://github.com/marlonbetz/BA

Dolgopolsky, A. B. (1986). A probabilistic hypothesis concerning the oldest relationships among the language families of northern eurasia. *Typology, Relationship and Time: A collection of papers on language change and relationship by Soviet linguists*, 27–50.

Duane, S., Kennedy, A. D., Pendleton, B. J., & Roweth, D. (1987). Hybrid monte carlo. *Physics letters B*, *195*(2), 216–222.

Dunn, M. (2012). *Indo-european lexical cognacy database (ielex).* URL: http://ielex. mpi. nl.

Fabius, O., & van Amersfoort, J. R. (2014). Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*.

Frey, B. J., & Dueck, D. (2007). Clustering by passing messages between data points. *science*, *315*(5814), 972–976.

Geman, S., & Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*(6), 721–741.

Goldberg, Y., & Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Gray, R. D., Drummond, A. J., & Greenhill, S. J. (2009). Language phylogenies reveal expansion pulses and pauses in pacific settlement. *science*, *323*(5913), 479–483.

Gutmann, M., & Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Aistats* (Vol. 1, p. 6).

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of classification*, *2*(1), 193–218.

Inkpen, D., Frunza, O., & Kondrak, G. (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).

Jäger, G. (2014). Phylogenetic inference from word lists using weighted alignment with empirically determined weights. In *Quantifying language dynamics* (pp. 155–204). Brill.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kondrak, G. (2000). A new algorithm for the alignment of phonetic sequences. In *Proceedings of the 1st north american chapter of the association for computational linguistics conference* (pp. 288–295).

Kulkarni, T. D., Whitney, W. F., Kohli, P., & Tenenbaum, J. (2015). Deep convolutional inverse graphics network. In *Advances in neural information processing systems* (pp. 2539–2547).

Landauer, T. K., McNamara, D. S., Dennis, S., & Kintsch, W. (2013). *Handbook of latent semantic analysis*. Psychology Press.

Lauly, S., Larochelle, H., Khapra, M., Ravindran, B., Raykar, V. C., & Saha, A. (2014). An autoencoder approach to learning bilingual word representations. In *Advances in neural information processing systems* (pp. 1853–1861).

Le, Q. V., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Icml* (Vol. 14, pp. 1188–1196).

List, J.-M. (2012a). Lexstat: Automatic detection of cognates in multilingual wordlists. In *Proceedings of the eacl 2012 joint workshop of lingvis & unclh* (pp. 117–125).

List, J.-M. (2012b). Sca: phonetic alignment based on sound classes. In *New directions in logic, language and computation* (pp. 32–51). Springer.

List, J.-M., & Forkel, R. (2016). *Lingpy. a python library for historical linguistics.* Jena: Max Planck Institute for the Science of Human History. Retrieved from `http://lingpy.org` doi: https://zenodo.org/badge/latestdoi/5137/lingpy/lingpy

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111–3119).

Mnih, A., & Teh, Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

Morin, F., & Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *Aistats* (Vol. 5, pp. 246–252).

Mortarino, C. (2009). An improved statistical test for historical linguistics. *Statistical Methods and Applications*, *18*(2), 193–204.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Emnlp* (Vol. 14, pp. 1532–43).

Rama, T. (2016). Siamese convolutional networks based on phonetic features for cognate identification. *arXiv preprint arXiv:1605.05172*.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, *66*(336), 846–850.

Řehůřek, R., & Sojka, P. (2010, May 22). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (pp. 45–50). Valletta, Malta: ELRA. (`http://is.muni.cz/publication/884893/en`)

Rosenberg, A., & Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In *Emnlp-conll* (Vol. 7, pp. 410–420).

Silberer, C., & Lapata, M. (2014). Learning grounded meaning representations with autoencoders. In *Acl (1)* (pp. 721–732).

Socher, R., Huang, E. H., Pennin, J., Manning, C. D., & Ng, A. Y. (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in neural information processing systems* (pp. 801–809).

Sokal, R. R. (1958). A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin*, *38*, 1409–1438.

Theano Development Team. (2016, May). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, *abs/1605.02688*. Retrieved from `http://arxiv.org/abs/1605.02688`

Turchin, P., Peiros, I., & Gell-Mann, M. (2010). Analyzing genetic connections between languages by matching consonant classes. *Journal of Language Relationship*, *3*, 117–126.

Vinh, N. X., Epps, J., & Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, *11*(Oct), 2837–2854.

Wichmann, S., Müller, A., Velupillai, V., Brown, C. H., Holman, E. W., Brown, P., ... others (2010). The asjp database (version 13). *See http://email. eva. mpg. de/wichmann/languages. htm*.

Zhang, J., Liu, S., Li, M., Zhou, M., Zong, C., et al. (2014). Bilingually-constrained phrase embeddings for machine translation. In *Acl (1)* (pp. 111–121).

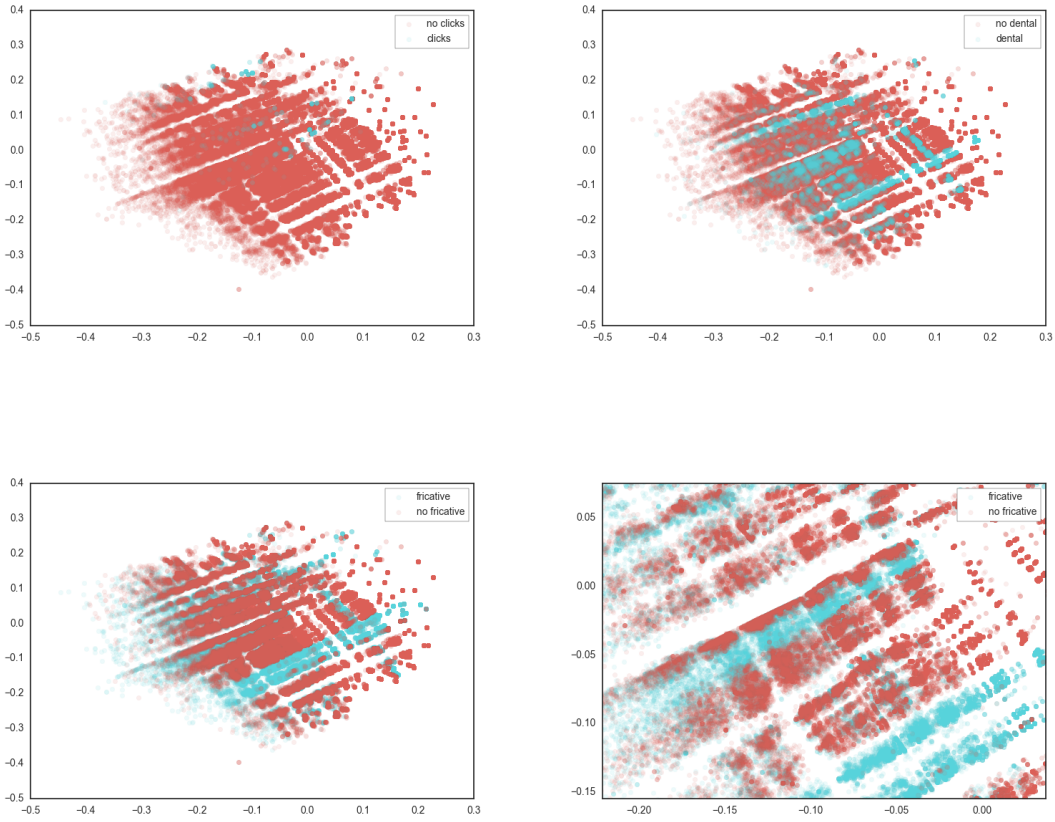

Figure 10: The ASJP data set embedded into $\mathcal{Z}$ after trained on it for 100 iterations, colored by whether a given word has a phoneme with some specific distinctive feature. (top left) The model learns that clicks are highly unlikely to emerge evolutionary and hence assigns low probability mass to their respective subspace. (top right) The distribution of words with and without dentals. The linear dependencies are clearly visible. (bottom left) The distribution of words with and without fricatives. Again, it can be seen how they are linearly dependent of each other. (bottom right) A detailed view on the words with and without fricatives.

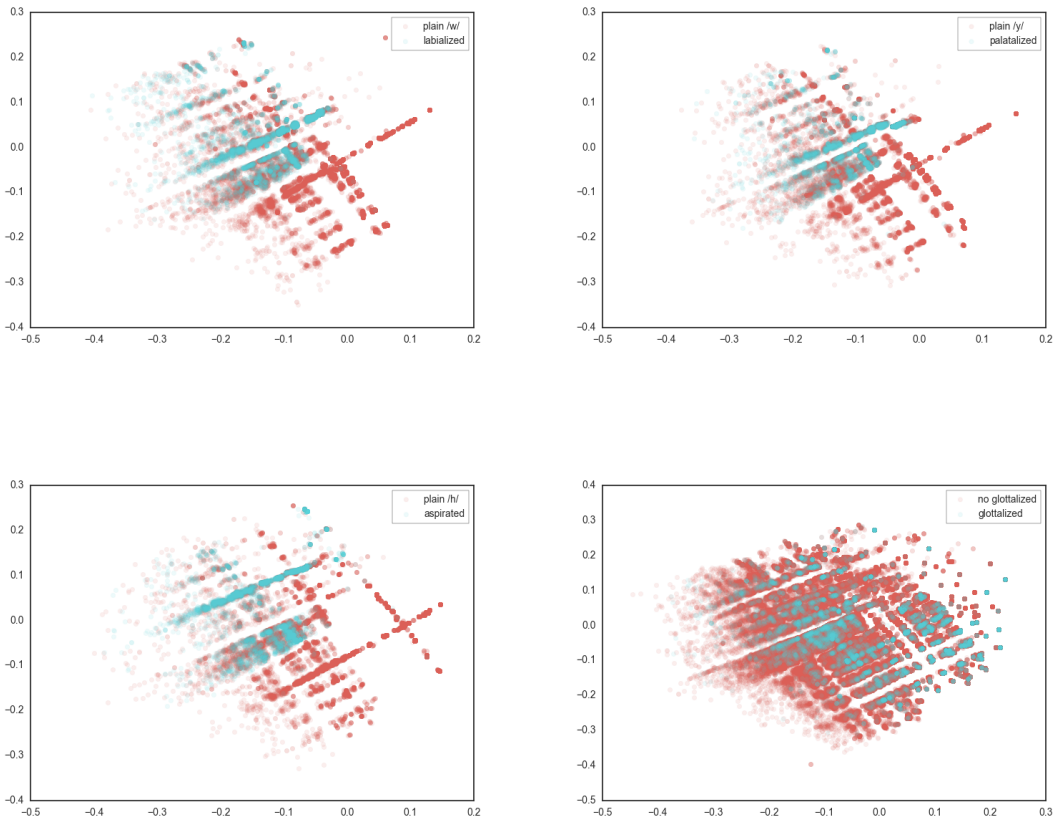

Figure 11: The ASJP data set embedded into $\mathcal{Z}$ after trained on it for 100 iterations. Since the binary features ignore coarticulations as such and encode them as independent phonemes instead, the model might learn to distinguish them, given some underlying latent features. (top left) The distribution of words that contain plain /w/ versus words with proper labialization. If the model did not learn the difference, we would expect some uniform distributions over all words that contain /w/ in the input. However, we can clearly see that words with proper labialization are located in other subspaces than words with proper /w/. Moreover, we see that words with labialized phonemes show linear dependence. (top right) The distribution of words that contain plain /y/ versus words with proper palatalization. Again, we see a clear distinction between the two. (bottom left) The distribution of words that contain plain /h/ versus words with proper aspiration. Again, we see a clear distinction between the two. (bottom right) The distribution of words that contain glottalized sounds versus that of words without. As glottalization can cover both vowels and consonants, the distributions are spread over the whole population. However, we can see that glottalized sounds cluster at the bottom right, a subspace containing mostly mostly words with velar and uvular consonants.

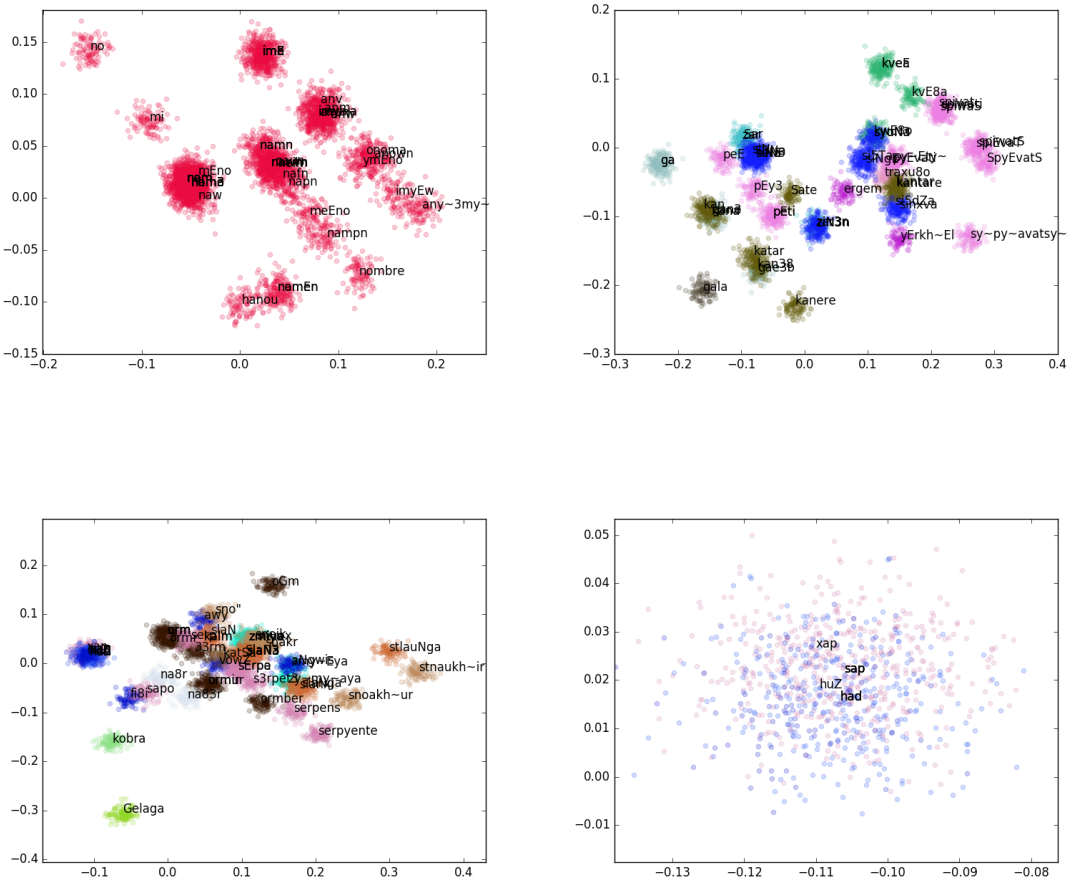

Figure 12: The posterior distribution $P(z|X)$ with X being the IELex data with regards to the single semantic concepts. The model was trained on ASJP for 2000 iterations. Colors indicate the respective true cognate class. (top left) The posterior for the concept "name". As all words are cognates to each other, we see strong linear dependencies over the distribution. Also note that the embeddings are embedded into a latent continuum of word lengths, from the the top left to the bottom right. Moreover, the members of the left hand side clusters all start with consonants, while the members of the clusters on right hand side all start with vowels or approximants. (top right) The posterior for "sing". Again, cognates show stronger linear dependencies among each other. (bottom left) The posterior for "snake". The romance words (pink samples) show linear dependencies. (bottom right) A closer look on some words for "snake". As VAE-PT concentrates on clustering words with similar syllable structures, it tends to cluster words that share such similarities close to each other. Here, it overestimates the similarity between Slovak /had/ and Upper Sorbian /huZ/ on the one side and the words /sap/ (Bihari,Magahi,Urdu,Marathi) and /xap/ (Assamese) on the other.

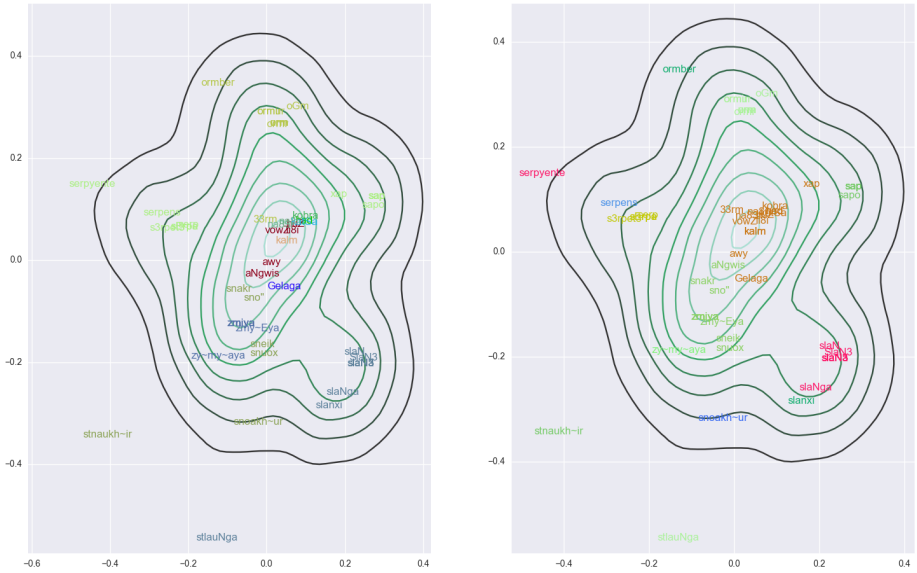

Figure 13: VAE-C trained on the IELex data for "snake". The left plot shows true cognates classes, the right plot shows inferred labels. Cognates, as long as they appear similar, are clustered at some subspace of $\mathcal{Z}$. Words with low self-information, such was words with no inferred cognates, are seen as "noise" by the model and grouped close to each other at the center of the latent space.

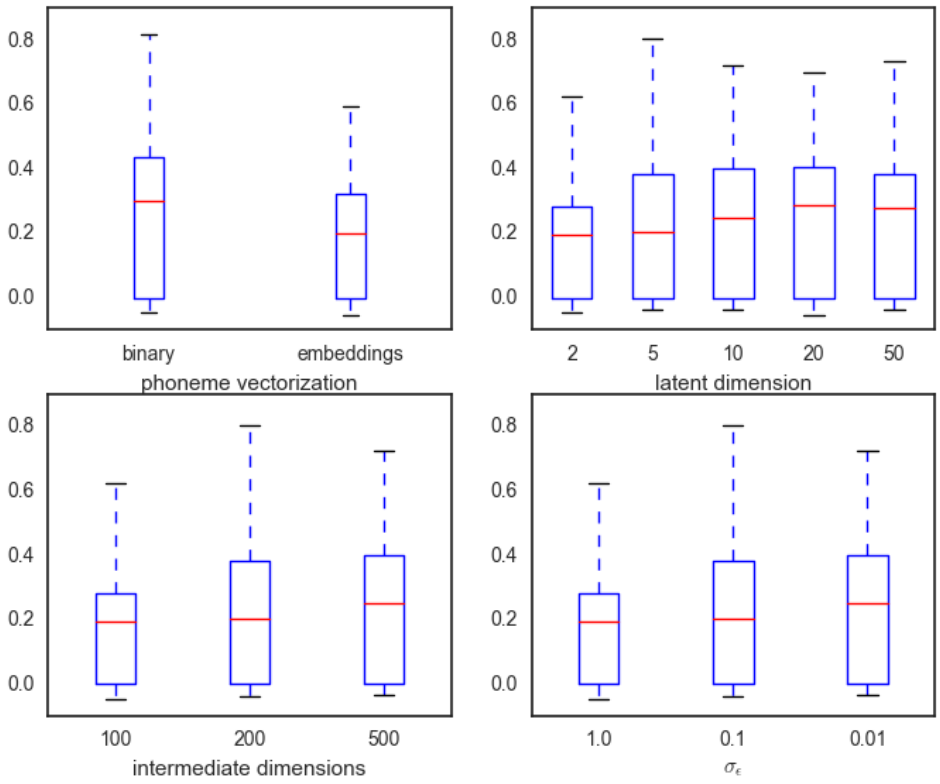

Figure 14: The result of the grid search over several parameters of VAE-C, measured in adjusted rand indices. In general, the performance is mediocre. (top left) The binary phoneme features outperform the phoneme embeddings. (top right) 20 latent dimensions seem to outperform the other parameters. (bottom left) As the number of intermediate dimensions of the encoder and decoder MLPs increases, the performance also improves slightly. (bottom left) The results suggest that a smaller standard deviation of the auxiliary noise improves the performance.

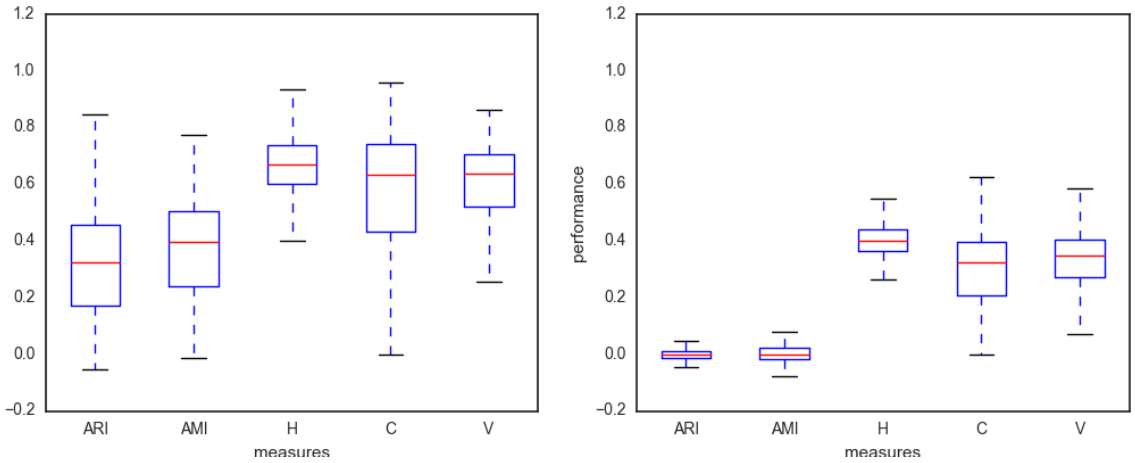

Figure 15: The performance of VAE-C on IELex, using the parameters obtained by the grid search (left), and random labeling (right). The measures are adjusted rand indices, adjusted mutual information, cluster homogeneity, cluster completeness and v-measure, as the harmonic mean of the latter two. The model learns to cluster similar words together, although the overall performance is mediocre.