Unsupervised Cognate Identification with Variational Autoencoders

Marlon Betz

August 27, 2016

Contents

1 Introduction **3**

2 Sound Change as a Walk in Latent Space **4**
2.1 Motivation . . . 4
2.2 $P(z|X)$ as Prior for Evolutionary Stable Phoneme sequences . . . 6
2.3 Sound Change as Posterior $P(w_{recent}|w_{ancient}, z)$ . . . 7
2.4 Cognate Identification as Hypothesis Comparison . . . 8

3 Related Research **8**

4 Architecture **8**
4.1 Phoneme Vectorization . . . 9
4.1.1 Hand-crafted Vectorization Models . . . 9
4.1.2 Data-driven Embeddings . . . 9
4.2 Word Embeddings . . . 12
4.2.1 Autoencoders . . . 12
4.2.2 Variational Autoencoders . . . 12
4.3 Clustering . . . 15
4.3.1 Affinity Propagation . . . 15

5 Evaluation **15**
5.1 Data . . . 15
5.2 Evaluation of $P(X)$ . . . 15
5.3 Results . . . 15

6 Resume **15**

7 Acknowledgements **15**

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 [Dellert2015] 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 [Rama2016]). In fact, most computational models here are either adaptations or direct copies of those used for the description of biological data. The aim of this thesis is hence

1. to combine methods from both Computational Historical Linguistics, Bayesian Models and the emerging field of Deep Learning

2. to propose a model of modeling sound change as a walk in latent space, which is suitable for neural networksand builds the basis for this thesis

3. to use variational autoencoders as a means to build such a latent space, and to investigate it

4. to show how this uncovered structure can be used to identify cognates in an unsupervised way

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 introduce the concept of sound change as a walk in latent space, which serves as a background for the actual inference model. Here I will talk about the main motivations for this approach as well as its major drawbacks.

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. That 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 Sound Change as a Walk in Latent Space

2.1 Motivation

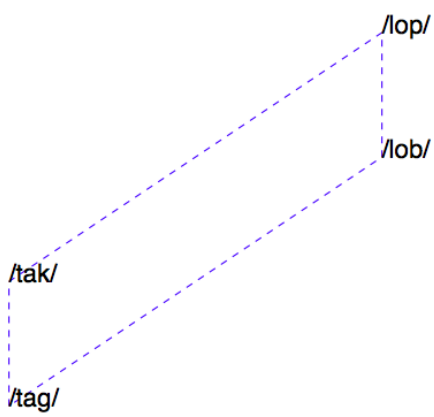

Figure 1: Visualization of the concept of sound change as a walk in latent space. Here, sound changes are vectors from one word to another, where both word forms are given as points in latent space. The vector from /lob/ to /lop/ would be the same as from /tag/ to /tak/, as both vectors would describe 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, Bouchard-Côté et al.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 model that t

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 underlying factors that produce the data are to complex to model directly.

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

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. 4.1 will discuss several such embedding methods for phonemes and 4.2 will explore several embedding algorithms for words.

The linearity assumption allows for the generalization over the compositional character of latent features over phonemes and words. For instance, in such a latent feature space $\mathcal{Z}$ over $\mathbb{R}^n$, a sound change sc 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

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

If we want to evaluate that latent feature space, we investigate in how far that compositional structure is preserved in our latent space. 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 (cf. [Mikolov et al.2013b]).

2.2 $P(z|X)$ as Prior for Evolutionary Stable Phoneme sequences

The question is then how to construct such a latent space $\mathcal{Z}$. If we follow the assumption from above that the model should treat factors in latent space as causes for variation of the actual data and not vice versa and if we further know that all words share a common structure, such as syllables and their respective internal structure, we can assume that it should be possible to model that structure by a latent variable z embedded in our latent space. As z walks through $\mathcal{Z}$, we assume that the resulting word x 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.

As we expect that it is not possible to model the underlying factors that create the data directly, a decoder function such as $f_{decode} : \mathcal{Z} \rightarrow \mathcal{X}$ cannot be linear. However, we assume f_{decode} can be approximated by some universal function approximator. This non-linearity than 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.

That is, we are interested in the posterior distribution of z - $P(z|X)$, i.e. the probability of z given our data X. We expect the posterior to be highly multimodal, as we expect e.g. that certain phoneme sequences that appear more often attract more probability density than less frequent ones. That is, we want to cluster words under the assumption that certain words are *inherently evolutionary stable sequences of phonemes* and hence appear more often, independently of their respective history. It is well known, for example, that syllables without codas are much more frequent than syllables without onsets.

To model the posterior $P(z|X)$, we can use Bayes' theorem:

$$P(z|X) = \frac{P(z)P(X|z)}{\int P(z)P(X|z)dz} \tag{8}$$

Here, $P(z)$ is the prior probability of z and $P(X|z)$ the likelihood of our data given our latent variable. The term in the denominator is the marginal probability $P(X)$ of our Data under the

given model. If we want to model z with the help of X, our objective is to maximize that marginal probability of our word list $P(X)$:

$$P(X) = \int P(z)P(X|z)dz \tag{9}$$

We further assume that $P(X|z)$ can be modeled by some output distribution and its parameters θ. This output distribution can be any distribution with continuous parameters, as they should be differentiable to allow for the backpropagation of error gradients through the model.

As we expect that the words in our word list follow the central limit theorem and in the long run should be captured by a normal distribution, our latent variable z is sampled from a isotropic multivariate normal prior:

$$z \sim \mathcal{N}(0, I) \tag{10}$$

where I is the identity matrix. To arrive at such linear dependencies as in Eq. 4, we then just have to expect them in our data X in some way - e.g in form of predefined phoneme features as in [Kondrak2000, List2012, Rama2016], pre-trained phoneme embeddings or as simple phoneme unigrams, in which case we would have the system learn latent phoneme features on its own. As X in our model would be conditioned on z, we expect the same linear dependencies over words in z as we find them in X.

2.3 Sound Change as Posterior $P(w_{recent}|w_{ancient}, z)$

As we usually do not know the ancient version of a word but only recent ones, we expect the cognates to have spread through the latent space from the original ancient word form to some directions. Since we assume that bigger jumps from one point in $\mathcal{Z}$ are less likely than small ones, we can expect that recent versions of an ancient word form should still be in the vicinity of the ancient word. Under the simplifying assumption that words in our word list did not jump from one semantic concept to another, this allows us embed all words of a given semantic concept into $\mathcal{Z}$ and to cluster them. This is then what this thesis focusses on - we embed words in $\mathcal{Z}$ and cluster the embeddings, as we expect cognates to appear close to each other.

However, there are some problems with this approach. The actual distribution of recent word forms in $\mathcal{Z}$ given the ancient forms would be given by the posterior

$$P(w_{recent}|w_{ancient}, z, X) = \frac{P(w_{recent}|z, X)P(w_{ancient}|w_{recent}, z, X)}{\int P(w_{recent}|z, X)P(w_{ancient}|w_{recent}, z, X)dw_{ancient}} \tag{11}$$

Unfortunately, we only know $P(w_{recent}|z, X)$, while we do not know $P(w_{ancient}|w_{recent}, z, X)$ directly, i.e. we cannot say anything specific about the likelihood of an ancient word given the recent words, that is the distribution of probabilities over points in $\mathcal{Z}$ where an ancient word form could be, given the distribution of its own child words. However, if we only consider those two terms and ignore the normalizing constant in the denominator, we have

$$P(w_{recent}|w_{ancient}, z, X) \propto P(w_{recent}|z, X)P(w_{ancient}|w_{recent}, z, X) \tag{12}$$

This means that if our prior $P(w_{recent}|z, X)$, which is known to us, has low probability density for the area around w_{recent}, we can be quite certain that $P(w_{recent}|w_{ancient}, z, X)$ should also be in such a low probability density area. As we except that the likelihood $P(w_{ancient}|w_{recent}, z)$ itself prefers points in $\mathcal{Z}$ that are close to w_{recent} over points that are far away from it, we can say

that w_{recent} should be close to $w_{ancient}$. However, if our prior $P(w_{recent}|z, X)$ is in an area with high probability density, the likelihood does not have such a regulating effect, as the high prior probability would allow for many points in Z to a possible point for $w_{ancient}$. That has two major implications:

- If we unpack our known prior $P(w_{recent}|z, X)$, we have

$$\begin{aligned}
P(w_{recent}|z, X) &= \frac{P(w_{recent})P(z, X|w_{recent})}{P(z, X)} \\
P(w_{recent}|z, X) &\propto P(w_{recent})P(z, X|w_{recent}) \\
P(w_{recent}|z, X) &\propto P(w_{recent})\frac{P(z, X)P(w_{recent}|X, z)}{P(w_{recent})} \\
P(w_{recent}|z, X) &\propto P(z, X)P(w_{recent}|z, X) \\
P(w_{recent}|z, X) &\propto P(X)P(z|X)P(w_{recent}|z, X)
\end{aligned} \tag{13}$$

  which involves the marginal probability of our data $P(X)$. If our data is small, we assume that most words do not look similar. For example, if we image that we have a word such as /o/, we can hardly make any claims about how some ancient form would look like - it could come from /aʊ̯t/ (Latin to Italian), /ok/ (Old Norse to Bokmål), /ob/ (Proto-Slavic to Slovak) or /akʷa̯/ (Latin to French). This is because /o/ as such receives high prior probability $P(w_{recent}|z)$ and hence gives high posterior probabilities over many possible ancient forms. However, if we have a word such as Polish /swuxat͡ɕ/ "listen", the remarkable structure of that word allows us to make much more detailed assumptions about the underlying ancient form, as we except more complex structures to be less frequent in our data and hence attract less probability density.

2.4 Cognate Identification as Hypothesis Comparison

If we want to compare the two hypotheses directly and compute the Bayes Factor, we have

$$\frac{P(w_{ancient} = w'_{ancient}|w_{recent}, w'_{recent})}{P(w_{ancient} \neq w'_{ancient}|w_{recent}, w'_{recent})} \tag{14}$$

$$= \frac{P(w_{ancient} = w'_{ancient})P(w_{recent}, w'_{recent}|w_{ancient} = w'_{ancient})}{P(w_{ancient} \neq w'_{ancient})P(w_{recent}, w'_{recent}|w_{ancient} \neq w'_{ancient})} \tag{15}$$

That is, we would have to define priors for both hypotheses. We cannot simply assume that both hypotheses are equally likely and share a flat prior, as we expect $P(w_{recent}, w'_{recent}|w_{ancient} = w'_{ancient})$ to be different to $P(w_{recent}, w'_{recent}|w_{ancient} \neq w'_{ancient})$.

3 Related Research

4 Architecture

Hence, the model should have three major components:

1. The phonemes should be embedded in a latent space, where similar phonemes should cluster in similar subspaces of the latent space.
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.
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 predicted cognate class.

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 (cf. [Bouchard-Côté et al.2007, Bouchard-Côté et al.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 (cf.[Bengio and Courville2016, 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]* (cf. [Chomsky and Halle1968]). 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

[Kondrak2000] 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.

[Rama2016] uses binary features that are adapted to the ASJP phonemic alphabet. 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. [Rama2016] further merges all coarticulations and vowels, as they tend to be diachronically unstable, and yields phonemes with 16 binary features each. [Rama2016] reports superior performance using those features as a basis for word vectorizations to find cognates in a supervised way.

4.1.2 Data-driven 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/ - /θi͡ento/ "hundred" and /t͡ʃɛlɔ/ - /θi͡elo/ "sky", we see that /t͡ʃ/ appears in a similar context as /θi͡/. 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 /θi͡/ are renderings of a common Vulgar Latin /ki͡/ - 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 et al.2013a, Mikolov et al.2013b, Goldberg and Levy2014] 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 (c.f. 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 /i͡/ 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 /θi͡/, 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, cf. [Morin and Bengio2005]). 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:

$$p(right|n, c) = \sigma(h^\top v'_n) \tag{16}$$

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 and Levy2014]) is the most popular. NEG is originally derived from Noise Contrastive Estimation (cf. [Gutmann and Hyvärinen2010, Mnih and Teh2012]). 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

$$J_\theta = -\sum_{w_i \in V} [\log \sigma(h^\top v'_w) + \sum_{j=1}^{k} \log \sigma(-h^\top v'_{\tilde{w}_{ij}})] \tag{17}$$

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 3 and 4). 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

$$v(phoneme_1) + v(phoneme_2) - v(phoneme_3) \approx v(phoneme_4) \tag{18}$$

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

$$v(/\mathbf{i}/) + v(/\mathbf{u*}/) - v(/\mathbf{u}/) \approx v(/\mathbf{i*}/) \tag{19}$$

$v(/\mathbf{u*}/) - v(/\mathbf{u}/)$ 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. 6 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 [Jäger2014, Rama2016] and pooling coarticulation types, the models perform much better, for analogies over plosives, while the performance for fricatives has only slightly improved.

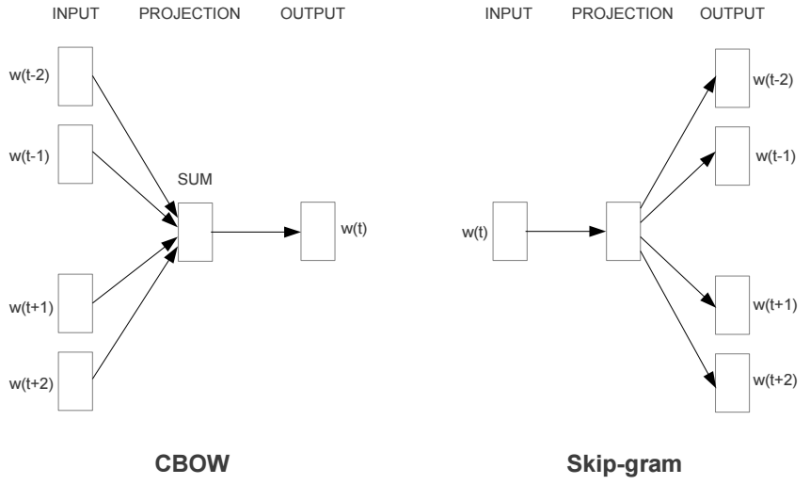

Figure 2: The two common architectures of word2vec. The Continuous Bag-of-Words (CBOW) model predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word. From [Mikolov et al.2013a].

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

$$L(x, g(f(x))) \tag{20}$$

where L 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 dimensionality of the data. If f and g are linear functions, autoencoders are proven to be equal to PCA. If they are non-linear, they should allow for the detection of latent features.

4.2.2 Variational Autoencoders

Variational Autoencoders (VAEs) were first described in [Kingma and Welling2013]. 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. Markov Chain Monte Carlo methods such as Gibbs sampling or Hamiltonian Monte

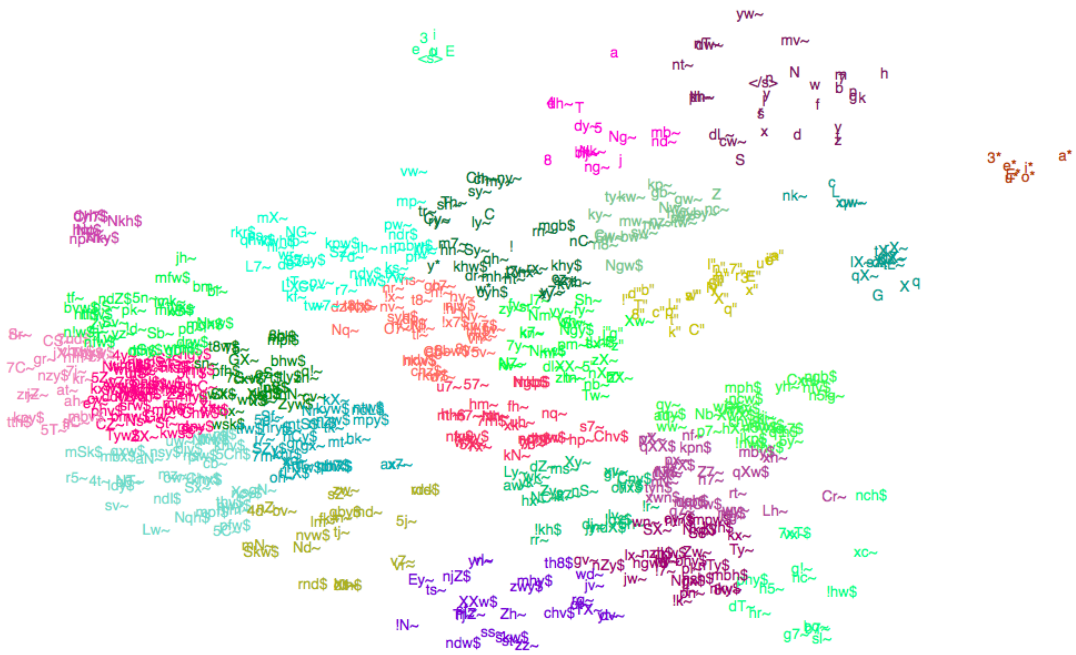

Figure 3: t-SNE visualization of the embeddings of all 720 phonemes contained in ASJP, trained with word2vec. The colors represent clusters inferred by Affinity Propagation. The model can clearly separate vowels and various forms of special articulation types. Pulmonic consonants can also be separated clearly from other coarticulation variants, as can uvular or pre-nasalized sounds. Less frequent phonemes, however, cannot be clearly differentiated.

Carlo 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:

$$D_{KL}(Q(z|X)|P(z|X)) = E_{z\sim Q}[\log Q(z|X) - \log P(z|X)] \tag{21}$$

Here, we can use Bayes' rule to yield

$$D_{KL}(Q(z)|P(z|X)) = E_{z\sim Q}[\log Q(z|X) - \log P(X|z) - \log P(z)] + \log P(X) \tag{22}$$

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

$$\log P(X) - D_{KL}(Q(z|X)|P(z|X)) = E_{z\sim Q}[\log P(X|z) - D_{KL}(Q(z|X)||P(z))] \tag{23}$$

Here, the left hand side makes it clear why Eq. 23 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 $\mathcal{X}$.

To estimate the parameters of Q, [Kingma and Welling2013] propose Multilayer Perceptrons (MLPs) as universal function approximator.

However, given that we use neural networks, we have to make sure that all gradients can be backpropagated through the network. In Eq. 23, 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 and Welling2013] 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 $p(\epsilon)$:

$$\epsilon \sim p(\epsilon) \tag{24}$$

[Kingma and Welling2013] 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:

$$\epsilon \sim \mathcal{N}(0,1) \tag{25}$$

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

$$z = \mu + \sigma \odot \epsilon \tag{26}$$

The right hand side of Eq. 23 then becomes

$$E_{\epsilon \sim \mathcal{N}(0,1)}[\log P(X|z = \mu + \sigma \odot \epsilon) - D_{KL}(Q(z|X)||P(z))] \tag{27}$$

The model architecture is visualized in Fig. 8.

Following this, 4 different models are investigated here:

1. A model that tries to maximize $p(W = w|z; W)$, i.e. that learns the manifold creating the words as such

2. A model that tries to maximize $p(W = w, C = c|z; W, C)$, i.e. that learns the manifold creating the words and the respective concepts

3. A model that tries to maximize $p(W = w, C = c, G = g|z; W, C, G)$, i.e. that learns the manifold creating the words, the respective concepts and the geographical location

4. A model that tries to maximize $p(W = w|z, C = c; W)$, i.e that learns the manifold creating words given the respective concept.

4.3 Clustering

4.3.1 Affinity Propagation

5 Evaluation

5.1 Data

5.2 Evaluation of $P(X)$

A first visual inspection shows that the model learns to cluster similar strings close to each other (Fig. 9). 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. 10), 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. 11). 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. 12). 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 $\mathcal{Z}$.

5.3 Results

6 Resume

7 Acknowledgements

For training the phoneme embeddings, I used the word2vec implementations provided by the gensim package [Řehůřek and Sojka2010]. The Autoencoder was implemented with Keras [Chollet2015] and Tensorflow [Abadi et al.2015]. 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]

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

References

[Abadi et al.2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Bengio and Courville2016] Bengio, I. G. Y. and Courville, A. (2016). Deep learning. Book in preparation for MIT Press.

[Bouchard-Côté et al.2013] Bouchard-Côté, A., Hall, D., Griffiths, T. L., and 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 et al.2007] Bouchard-Côté, A., Liang, P., Griffiths, T. L., and Klein, D. (2007). A probabilistic approach to diachronic phonology. In *EMNLP-CoNLL*, pages 887–896. Citeseer.

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

[Chollet2015] Chollet, F. (2015). Keras. https://github.com/fchollet/keras.

[Chomsky and Halle1968] Chomsky, N. and Halle, M. (1968). The sound pattern of english.

[Dellert2015] Dellert, J. (2015). Uralic and its neighbors as a test case for a lexical flow model of language contact.

[Doersch2016] Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

[Goldberg and Levy2014] Goldberg, Y. and Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

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

[Gutmann and Hyvärinen2010] Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, volume 1, page 6.

[Jäger2014] Jäger, G. (2014). Phylogenetic inference from word lists using weighted alignment with empirically determined weights. In *Quantifying Language Dynamics*, pages 155–204. Brill.

[Kingma and Welling2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[Kondrak2000] 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*, pages 288–295. Association for Computational Linguistics.

[Landauer et al.2013] Landauer, T. K., McNamara, D. S., Dennis, S., and Kintsch, W. (2013). *Handbook of latent semantic analysis*. Psychology Press.

[List2012] List, J.-M. (2012). Sca: phonetic alignment based on sound classes. In *New Directions in Logic, Language and Computation*, pages 32–51. Springer.

[Mikolov et al.2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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

[Mnih and Teh2012] Mnih, A. and Teh, Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

[Morin and Bengio2005] Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer.

[Pedregosa et al.2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Pennington et al.2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

[Rama2016] Rama, T. (2016). Siamese convolutional networks based on phonetic features for cognate identification. *arXiv preprint arXiv:1605.05172*.

[Řehůřek and Sojka2010] Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

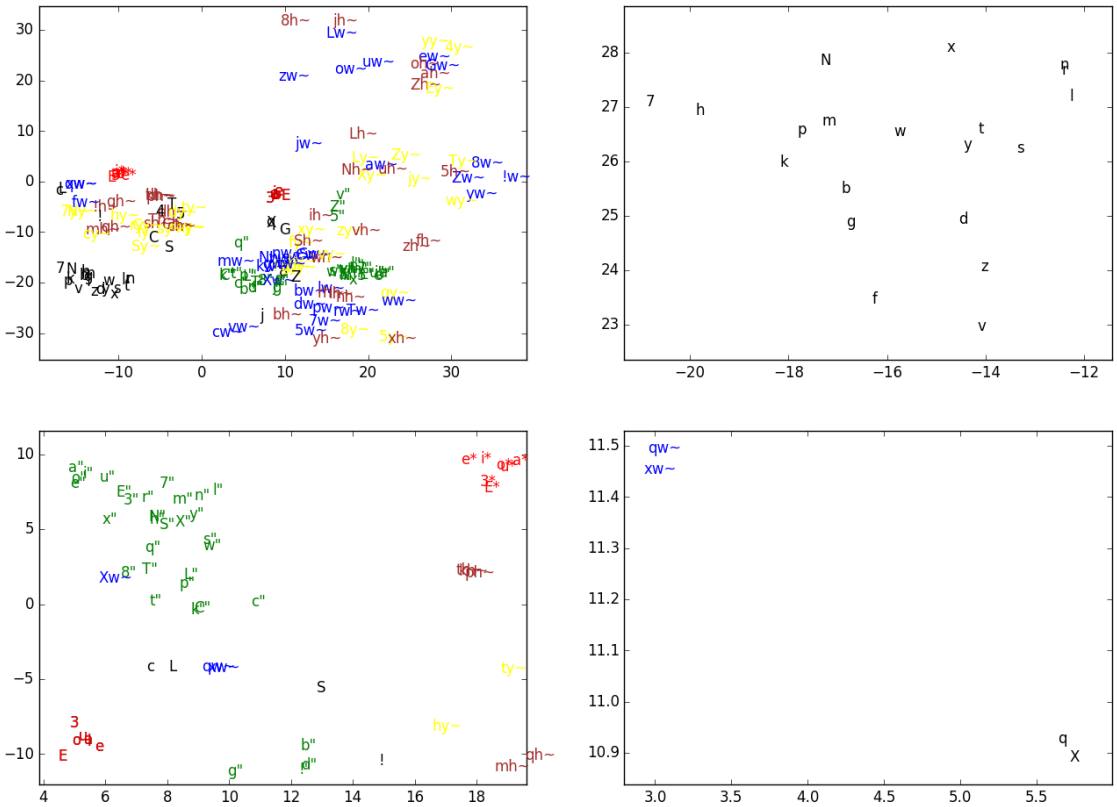

Figure 4: Other t-SNE visualizations of the embeddings created by word2vec. (top left) The model learns to clearly separate natural classes such as vowels, plain pulmonic or glottalized consonants, while other articulations seem to spread over the feature space. The colors indicate membership of a natural 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.

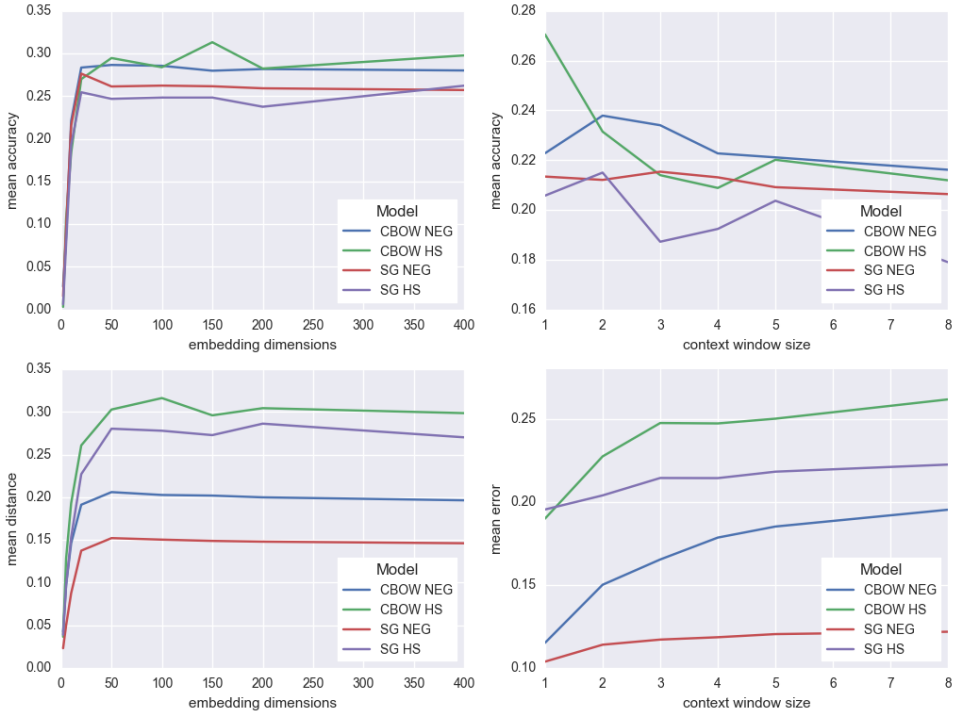

Figure 5: Comparison of the four word2vec models evaluated. (top left) Both CBOW Models perform better than the skip-gram models over all numbers of embedding dimension. (bottom left) Mean distance between the predicted vector and the target with regard to embedding dimensions. Here, negative sampling 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.

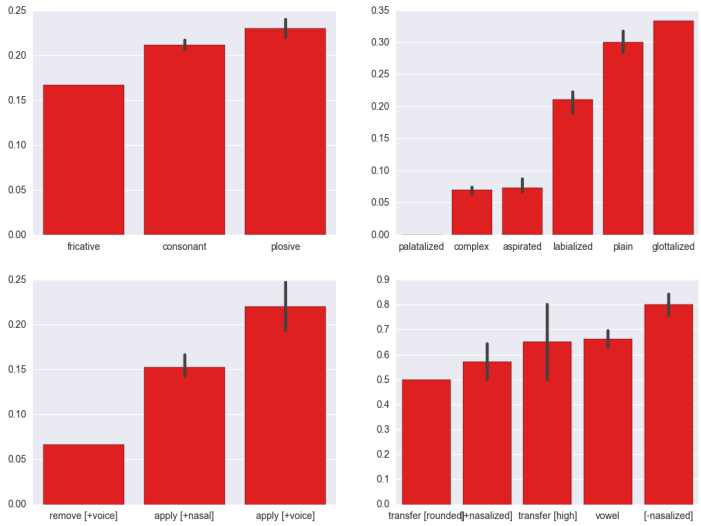

Figure 6: Mean accuracies for analogies. The model does not seem to be able to capture the compositionality of the latent features too well. (top left) Accuracy is best for plosives across plain pulmonic as well as more complex articulations, while fricatives perform worse. (top right) Glottalized and 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.

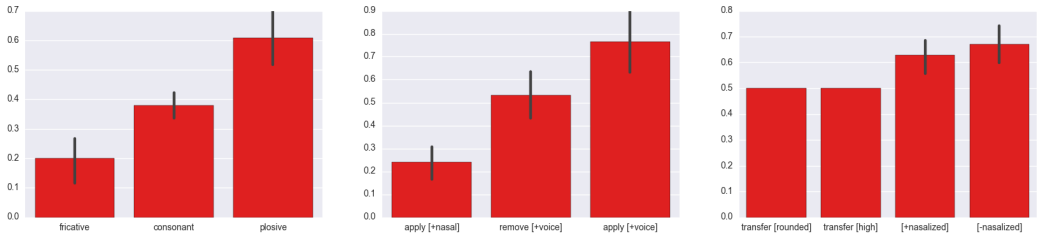

Figure 7: Mean accuracies for analogies over the pooled consonant phonemes. Here, the model is expected to yield better results, as the number of latent features should have been reduced. (left) Accuracy is doubled compared with the unpooled phonemes. (middle) Adding voice works quite well, while removing 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.

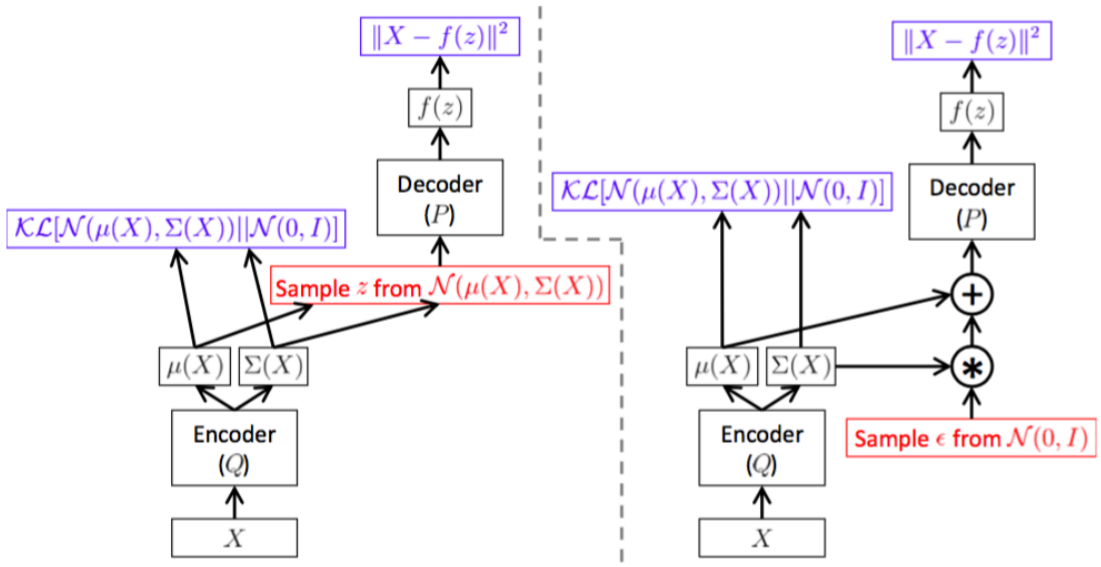

Figure 8: Visualization of the Variational autoencoder architecture. (left) The model with original objective as in Eq. 23. The stochastic unit is inside the network and would not allow for the back-propagation of error gradients through the network. (right) The model after the reparameterization with the objective as in Eq. 27. Here, the sampling is interpreted as an input variable, so error gradients can be backpropagated through the whole model. From [Doersch2016].

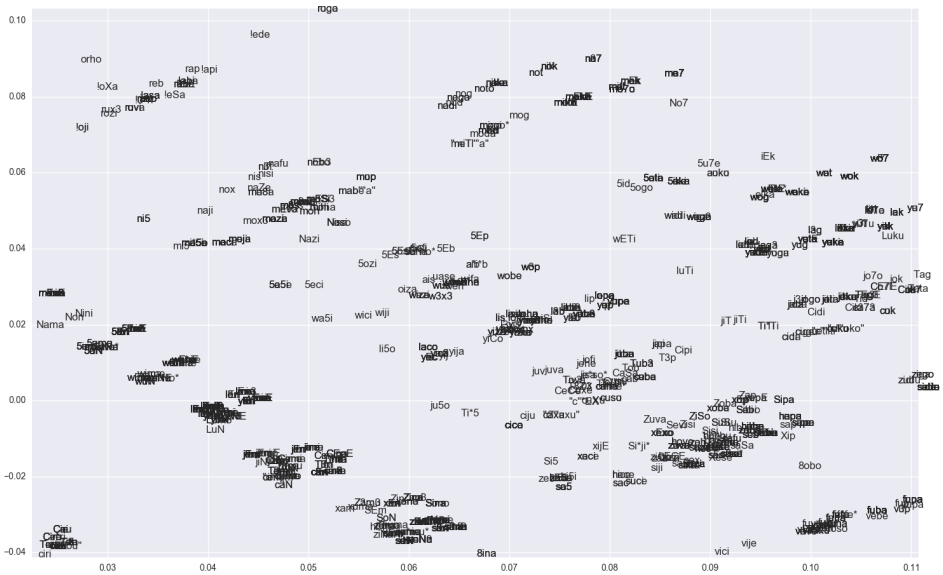

Figure 9: A detailed view on a subsample of ASJP embedded into $\mathcal{Z}$ after trained on the whole ASJP data set for 100 iterations. As can be seen, words accumulate in local subspaces with higher probability mass. Here, all words are contained in an area with shorter words that show simple CV syllable 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.

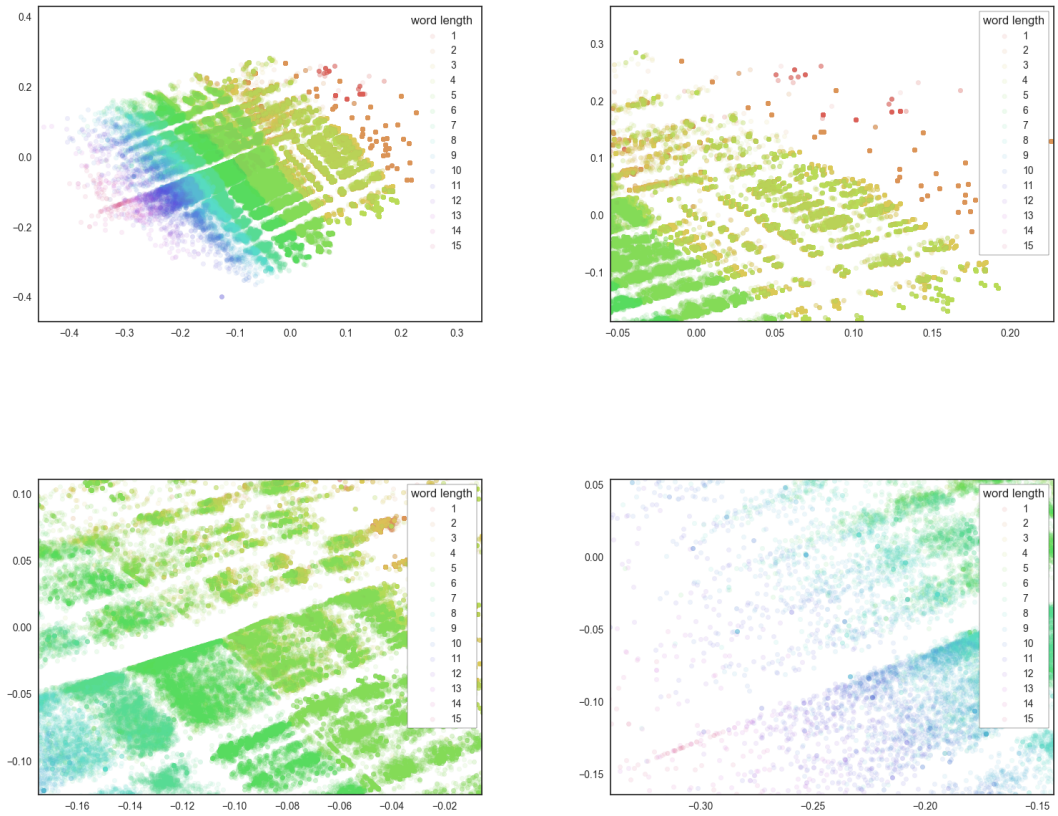

Figure 10: The ASJP data set embedded into $\mathcal{Z}$ after trained on it for 100 iterations, colored by word length. The model learns to cluster words according to their respective length, with more frequent word lengths located in subspaces with higher probability mass.

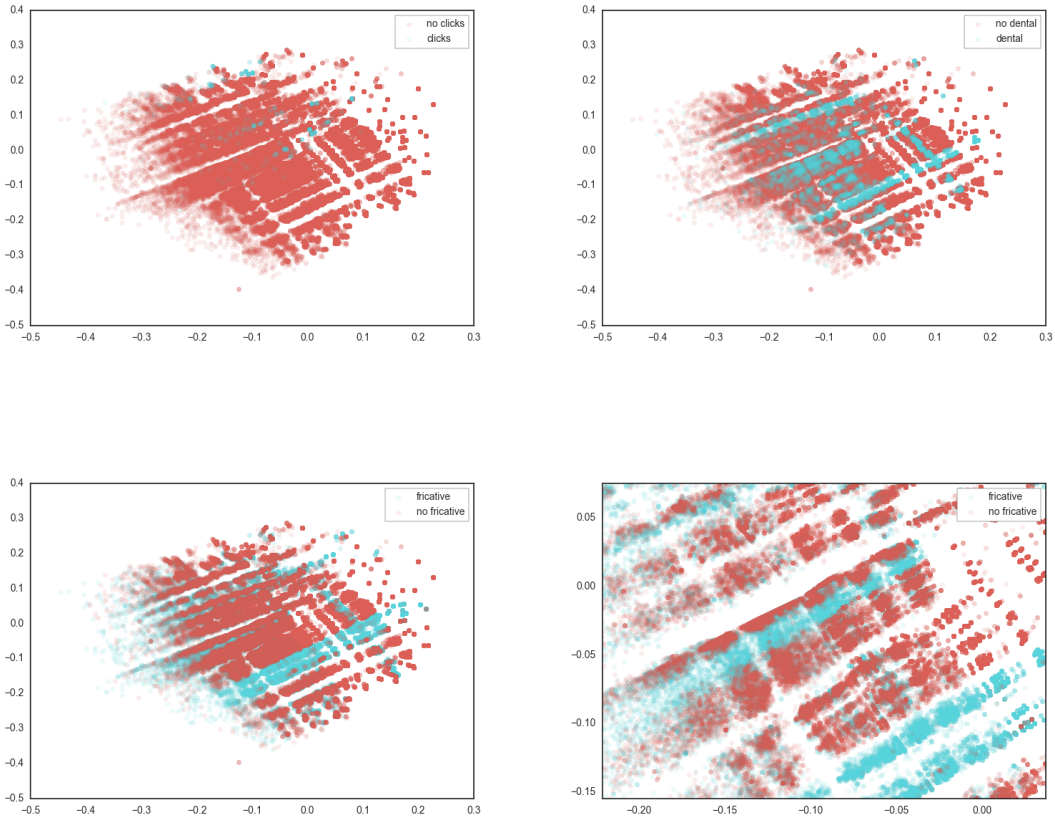

Figure 11: The ASJP data set embedded into $\mathcal{Z}$ after trained on it for 100 iterations, colored by whether a given word has a phoneme with some specific distinctive feature. (top left) The model learns that clicks are highly unlikely to emerge evolutionary and hence assigns low probability mass 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.

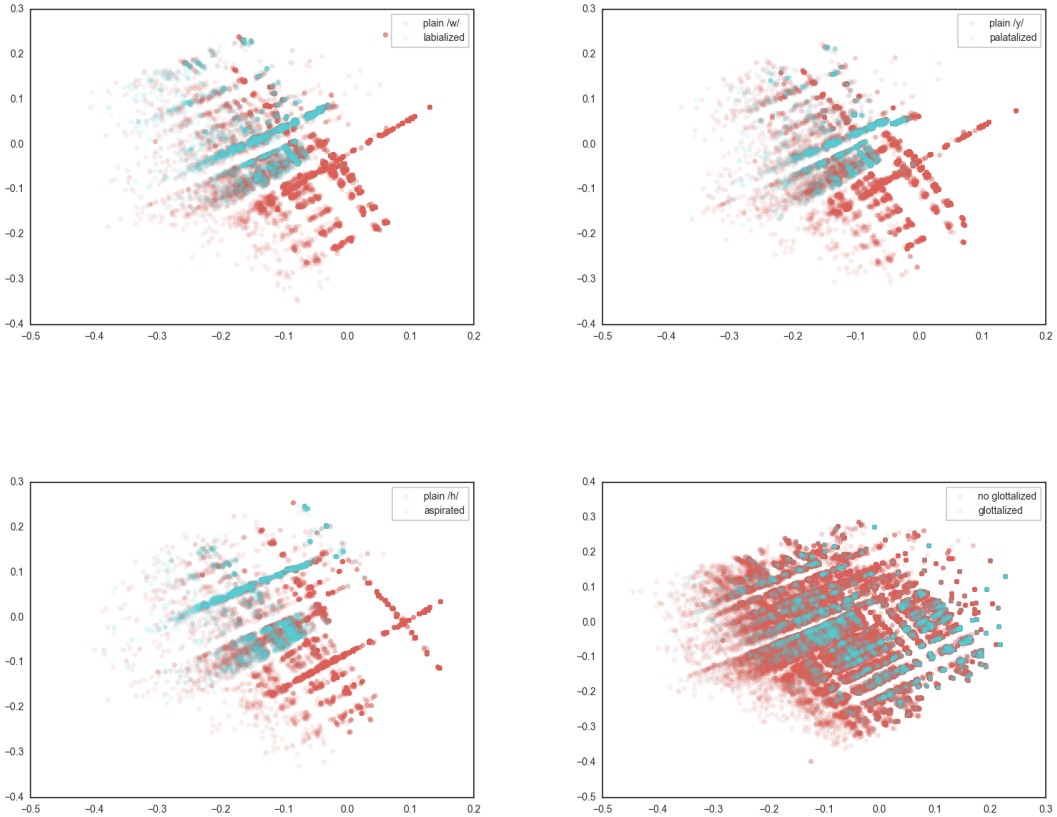

Figure 12: The ASJP data set embedded into $\mathcal{Z}$ after trained on it for 100 iterations. Since the binary features ignore coarticulations as such and encode them as independent phonemes instead, the model might learn to distinguish them, given some underlying latent features. (top left) The distribution 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.