Unsupervised Cognate Identification with Variational Autoencoders

Marlon Betz

September 12, 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-related fields of research, e.g. Statistical Machine Translation (Zhang et al., 2014; Lauly et al., 2014), Paraphrase Detection (Socher et al., 2011), Sentiment Analysis (Socher et al., 2013) or Speech synthesis and recognition (G. Hinton et al., 2012; Zen & Senior, 2014). Moreover, Bayesian inference models have been proven to allow for a detailed analysis of data and testing hypotheses in various fields of Computational Linguistics (Crocker, 2010; Chater & Manning, 2006), while for instance in the form of Latent Dirichlet Allocations they provide a strong tool for document classification systems (Blei et al., 2003). In fact, Bayesian models build the backbone for most current phylogenetic surveys (Gray et al., 2009; Bouckaert et al., 2012; Bouchard-Côté et al., 2013). 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 (I. G. Y. Bengio & Courville, 2016, p. 654). 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, we will propose a novel view on cognate identification as latent representation learning (Y. Bengio et al., 2013; I. G. Y. Bengio & Courville, 2016). 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 (Radford et al., 2015; Dosovitskiy et al., 2015), 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 relatively simple, as it is a direct adaptation of Variational Autoencoders (D. P. Kingma & Welling, 2013), a quite recent deep generative architecture that combines non-linear dimensionality reduction with the strength of Variational Bayesian inference. However, it can be used as a basis for more detailed research on how language change can be model with deep learning architectures.

We will start by giving an overview on why cognate identification is important in order to discuss the historical connections between languages. We will also state the motivations behind the approach introduced by this thesis. Then, we will further provide an overview on several established methods to detect cognates. We will then proceed with the discussion of the actual architecture of the inference model. This first covers a general overview on the components included in the model. We 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. We then come to the discussion how to cluster the words, i.e. to assign the actual inferred cognacy labels. Then we will document how well those methods can be used to infer cognates between several Indo-European languages by comparing the inferred labels with expert judgements. Finally, we will give a resume on the models described here.

2 Motivation

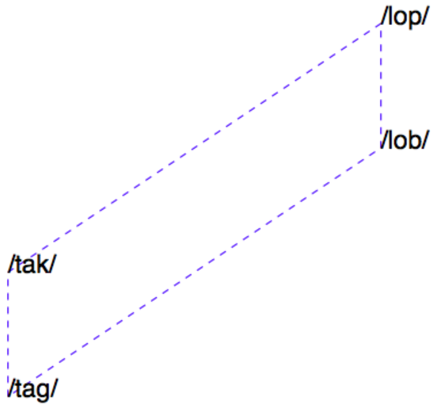

Figure 1: Visualization of the concept of sound change as a walk in latent space. Here, sound changes are vectors from one word to another, where both word forms are given as points in latent space. The vector from /lob/ to /lop/ would be the same as from /tag/ to /tak/, as both vectors would describe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

Words across different languages that share some common historic origin are called cognates (Trask, 1996, p. 193). Finding such cognates is one of the most important tasks of Historical Linguistics. If languages share vocabulary, their speakers should also share some common historic origin. This allows for the inference of migration waves (Gray et al., 2009; Bouckaert et al., 2012) or the influence of one language on another (Dellert, 2015). If we know that certain words are cognates, we can moreover infer certain assumptions on how sound change is structured and how it can be modeled.

The task of manual cognate identification is traditionally connected to the *comparative method* (Trask, 1996, p. 191ff). Here, one starts by looking for words that share roughly the same meaning and phonological shape. Then, the actual phonemes are compared to each other to find sound correspondences between languages. Taking phonological universals into account, one can then start to derive some phonemes from others, or estimate some common predecessor for those phonemes compared to each other. In that way, one can reconstruct proto-forms of words. Then, one has to check if the phonological system of such reconstructed vocabularies obeys phonological universals. If that should be the case, one repeats all the steps before with some finer grained analysis, to look for more subtle phonological changes for certain words until several words can be derived from one inferred proto-form, and hence can be called cognates .

In linguistic literature, such estimated sound changes from one ancient form to a recent one are usually described as a change of distinctive phonological features over phoneme sequences. For instance, a sound change such as final devoicing as in Middle High German /hʊnd/ → New High German /hʊnt/ ”dog” would be captured by an string edit rule like

$$d \to t/\_\# \tag{1}$$

that says that /d/ becomes /t/ before the end of a word. Usually, 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 (Chomsky & Halle, 1968). Accordingly, our rule from above could be generalized to

$$[+\textsc{voice}] \to [\text{-}\textsc{voice}]/\_\# \tag{2}$$

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

However, this method is painstaking if the number of languages and words to compare increases. Hence, computational models that perform cognate identification are needed to explore the vastness of languages that are still uncategorized or where family topologies are disputable.

Computational models of sound change usually treat words as string of symbols and sound change as substitution of substrings. This follows the idea that sound changes can be modeled by context-sensitive rules like in Eq. 1. 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 specifically 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.

To counter those problems, this thesis proposes a novel model of of sound change under the following assumptions:

1. Phonemes and words cannot only be compared according to some distance metric, but moreover are situated in some latent space that is the underlying causal factor for that distance metric. Certain subspaces of that latent space encodes some latent information of the words

embedded in that subspace, but moreover encodes information for similar words that *could* be embedded there, but are not found in the data as such. As a word walks through this latent space along some time axis, its phonological shape evolves from some ancient proto-form to its later shape in a given daughter language. This interpretation of sound change as a walk in latent space also means that functions connecting our data with that latent space should be smooth. This smoothness assumption builds the basis for several machine learning algorithms that could use such a model as a basis for further research (I. G. Y. Bengio & Courville, 2016, p. 557).

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. Again, this assumption is important for several machine learning algorithms, such as Generalized Linear Models (I. G. Y. Bengio & Courville, 2016, p. 557).

3. Those underlying linear factors that produce the data are to abstract to see directly, but only exist in some highly-abstract latent space that is not directly visible. This means that encoder and decoder functions that link latent representations to the actual data have to be non-linear.

4. The model should be probabilistic, which allows for Bayesian inference. That is, the model generates the latent representation by sampling from some prior or posterior distribution, instead of just estimating some point that maximizes some given objective. From this also follows that the model should treat factors that change some latent representation as causes for variation of the actual data and not vice versa, which should allow good generalization abilities in semi-supervised learning scenarios (I. G. Y. Bengio & Courville, 2016, p. 557).

For the task of cognate identification, the first point is the most important. As it states that cognates, as they are usually expected to have similar shapes, should be clustered in such subspaces, we can use clustering algorithms to assign labels to such clusters. Word embeddings that share such a label can than be interpreted as inferred cognates. While this thesis investigates in how far such a simple prior assumption holds and can be used to infer cognates, we are further aware that cognates can look different to each other (e.g. Icelandic *tveir* vs Armenian *erku* < PIE *dwoh$_1$ ”two”), while certain words might look similar, but are no cognates (e.g. Spanish *haber* ”have” < PIE *g^{h}eh$_1$b^h- and German *haben* ”have” < PIE *keh$_1$p-). This is the major point of interest when exploring cognate identification models in general. In our case, as described below, we assume that a neural network might learn to recognize latent features shared by certain words, which can allow of cognate inference. For instance, while we do not generally expect the model to correctly predict the three examples from above, we assume that the model can recognize that Armenian *du*, Polish *ty* and Irish Gaelic *tú* ”you (sing.)” are cognate, since they all show a dental obstruent initially, which is followed by a vowel.

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 (D. P. Kingma & Welling, 2013) as the preferred embedding algorithms for words, as they are deep probabilistic architectures that allow for both non-linearity and Bayesian inference.

While we do not make detailed assumptions here on how cognates behave compared to unrelated words except they share some surface features and have a similar meaning, such a model can indeed incorporate such more detailed assumptions in form of Bayesian priors. The exploration of such priors can be an interesting object of research on its own and is not covered here in detail.

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. 2 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

$$v_{w_{ancient}} + v_{sc} = v_{w_{recent}} \tag{3}$$

From this follows that

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

That is, we can formulate sound changes such as 2 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

$$v_{w_{recent}} - v_{w_{ancient}} = v_{w'_{recent}} - v_{w'_{ancient}} \tag{5}$$

which is equivalent to

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

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. While we will not make direct use of such linear dependencies with the means described in this thesis, we will still encounter them quite often in the latent space, as described below. To investigate how such linear dependencies can be used in particular to model sound changes that are more complicated and involve context-sensitive edit rules (e.g. haplologies, partial reduplications etc.) can constitute another object of research on its own and will not be discussed here in detail.

3 Related Research

Cognate identification systems can be divided into two three major groups. Supervised systems learn from a set of positive and negative examples to differentiate between cognates and unrelated words. The features used to discriminate between those examples are usually some sort of string comparison metric, 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). Another approach has been pioneered by Rama (2016), where siamese convolutional networks are used to differentiate between cognates and unrelated words. Unsupervised systems, on the other hand, 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. Mackay & Kondrak (2005);

Wahle (2013) use Pairwise Hidden Markov Models to score distances between words and show good performance. Wahle (2013) also reports improved performance over PMI-weighted Needleman-Wunsch alignment scores, but only tests the model on Germanic data and uses labels provided by LexStat instead of expert judgements as gold standard. Another family of models focus on modeling sound change directly, but can be used for cognate identification. Bouchard-Côté et al. (2013) learns a probabilistic state-transducer to estimate Proto-Austronesian word forms given words from contemporary languages and the already established tree topology of the language family. If the proto-form of two words is the same, they are judged as cognates. Although they report good performance, their approach has not been carried out so far for other families than Austronesian.

4 Architecture

Embedding words as sequences of phonemes is similar to embedding sentences as sequences of words. In fact, there are several deep architectures that use sentence or phrase embeddings to encode latent information stored in the respective sequences, where each token in a sequence itself is represented by latent representation of that token rather than by the token itself (Kiros et al., 2015; Zhang et al., 2014). Inspired by such models, the model described here also shows a modular architecture:

1. First, 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 Section 2. Section 4.1 will cover this topic.

2. Then, 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 variational 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. Finally, 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.

The separation of learning the phoneme embeddings on the one hand and word embeddings on the other has practical reasons. On the one hand, phonemes that are decomposed into latent features should allow for better generalization even if the pool of words to cluster is small. While encoding sequences of word unigrams has been proven to work well with recurrent variational autoencoders (Bowman et al., 2015), this approach will, however, not be explored in this thesis since such unigram-based models are know to need a vast amount of training data to generalize well. On the other hand, there are several ways of vectorizing phonemes, as described in Section 4.1. If we learn them independently of the word embeddings, we can explore the information directly contained in the phonemes as such and discuss benefits and drawbacks for the respective model.

However, sentence encoding models where words embeddings and sentence embeddings are learned jointly are proven to work well Kiros et al. (2015), thus models that learn phoneme and word embeddings jointly can nevertheless be the object of future research.

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 $P(X|z, concept)$, i.e that learns the manifold creating words given the respective semantic concept, and only those words. 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.

The main point of interest for VAE-PT is its ability to discover the latent high-level connections between words. As the training corpus here would be quite big, there are much more words that are not cognate to each other, hence the probability that cognates cluster among each other here is quite low. As we expect that as the training set increases it becomes more representative for words as such, we rather expect that more common syllable structures that are cluster among each other if they are similar, but not necessarily cognates. Here, words are rather expected to be connected through linear dependencies as described in section 2. Hence, VAE-PT can rather be seen as a "showcase" that exemplifies the strength of deep generative models employed as a basis for modeling sound change. However, we do not assume it works well for clustering cognates as such, at least as long as we use distance or similarity metrics as input for clustering algorithms. VAE-C, on the other hand, concentrates on distinguishing words that can actually be cognates. Hence, we expect it to work much better for clustering embeddings, as the probability that clusters coincide with true cognate classes is much higher. Following this, linear dependencies play a less important role here, but nevertheless allow insight into the compositionality of the words to cluster.

Moreover, it is important to note that the general architecture described here allows for much more models than these two. For instance, jointly encoding the phonological shape of a word with some vector representation of its semantic content should allow for the incorporation of different phoneme or cognate substitution rates among semantic concepts. Encoding both time and geographical information of the language of a given word should further provide an interesting starting point to condition sound change on the time and place of its language, so that words from geographically close languages could evolve similarly trough time. While we restrict our research in the evaluation of VAE-PT and VAE-C, we think that such more complex models should provide interesting starting points for further research.

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 have certain problems to generalize well to process data they have not seen during training. Such an ability is usually connected to the *smoothness assumption* (cf. section 2, (I. G. Y. Bengio & Courville, 2016, p.

555)). This assumption allows a model to generalize from training data to close points 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. We will first cover hand-crafted phoneme vectorization models that are commonly found in the field of computational historical linguistics, and then compare those with phoneme embeddings learned on some big corpus.

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)). For instance, /k/ and /t͡ʃ/ are both represented as a symbol K, since /k/ often evolves into /t͡ʃ/, for instance in German /k^{h}eːzə/ and English /t͡ʃiːz/ ”cheese”. 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). SCA, as 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. While such pooling methods do not necessarily obey the smoothness assumption, they still allow for some phonemes to be closer (i.e. equal) to each other than to other phonemes.

Kondrak (2000) further 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 (Kessler, 2007), 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. 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 their respective embeddings should be close to the embedding of /t͡ʃ/.

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. 7 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:

$$p(right|n, c) = \sigma(h^\top v'_n) \tag{7}$$

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

$$J_\theta = -\sum_{w_i \in V} [\log \sigma(h^\top v'_w) + \sum_{j=1}^{k} \log \sigma(-h^\top v'_{\tilde{w}_{ij}})] \tag{8}$$

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

4.2 Word Embeddings

4.2.1 Autoencoders

Autoencoders are a family of feed-forward 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 $g : \mathcal{Z} \to \mathcal{X}$ that both relate data points to points in some latent embedding space $\mathcal{Z}$, they are trained on encoding and reconstructing the data, yielding a loss function

$$J(x, g(f(x))) \tag{9}$$

that penalizes the reconstruction error. Although it is possible to yield meaningful representations in higher-dimensional embedding space via some sparsity constraint (Ng, 2011), 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 approximate PCA. If they are non-linear, they allow for the recognition of latent features (G. E. Hinton & Salakhutdinov, 2006).This is important for us, as we expect the network to detect shared features between words, for instance a similar syllable structure or the presence or absence of certain phonemes. Another crucial point is that the embeddings created with autoencoders usually show the underlying linear factors that constitute the data. As the networks can learn to detect latent features, they are also able to apply them to some given data via tensor products. Word2vec (Mikolov, Chen, et al., 2013; Mikolov, Sutskever, et al., 2013; Goldberg & Levy, 2014), as the probably most popular embedding algorithm that in fact is a shallow autoencoder, became popular by exploiting such linear dependencies between words. However, autoencoders are shown to generalize for the linear dependency for various other types of data (Radford et al., 2015; Dosovitskiy et al., 2015). Hence, we expect autoencoders to be the preferred tool to arrive at linear dependencies over sound changes as described in section 2. 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).

4.2.2 Variational Autoencoders

Variational Autoencoders (VAEs) were first described in D. P. 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 $\mathcal{Z}$.

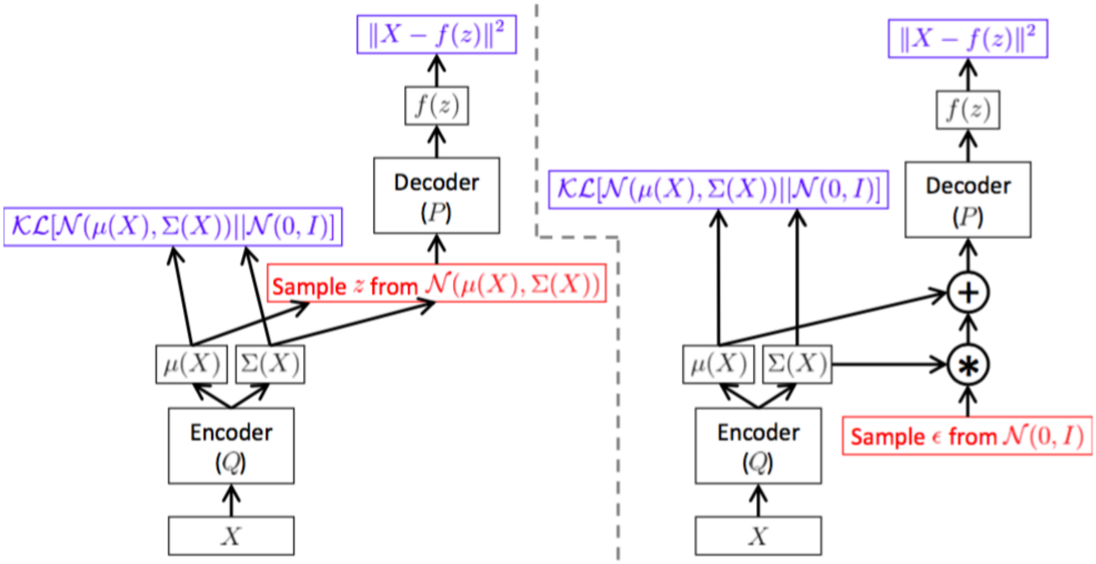
$\|X - f(z)\|^2$
$f(z)$
Decoder (P)
$\mathcal{KL}[\mathcal{N}(\mu(X),\Sigma(X))||\mathcal{N}(0,I)]$
Sample z from $\mathcal{N}(\mu(X),\Sigma(X))$
$\mu(X)$
$\Sigma(X)$
Encoder (Q)
X
$\|X - f(z)\|^2$
$f(z)$
Decoder (P)
$\mathcal{KL}[\mathcal{N}(\mu(X),\Sigma(X))||\mathcal{N}(0,I)]$
$+$
$*$
$\mu(X)$
$\Sigma(X)$
Sample ϵ from $\mathcal{N}(0,I)$
Encoder (Q)
X

Figure 2: Visualization of the Variational autoencoder architecture. (left) The model with original objective as in Eq. 12. The stochastic unit is inside the network and would not allow for the backpropagation of error gradients through the network. (right) The model after the reparameterization with the objective as in Eq. 20. Here, the sampling is interpreted as an input variable, so error gradients can be backpropagated through the whole model. From Doersch (2016).

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:

$$D_{KL}(Q(z|X)||P(z|X)) = E_{z\sim Q}[\log Q(z|X) - \log P(z|X)] \tag{10}$$

Here, we can use Bayes' rule to yield

$$D_{KL}(Q(z|X)||P(z|X)) = E_{z\sim Q}[\log Q(z|X) - \log P(X|z) - \log P(z)] + \log P(X) \tag{11}$$

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

$$\log P(X) - D_{KL}(Q(z|X)||P(z|X)) = E_{z\sim Q}[\log P(X|z) - D_{KL}(Q(z|X)||P(z))] \tag{12}$$

Here, the left hand side makes it clear why Eq. 12 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 expectation that the encoder distribution $Q(z|X)$ approximates the true prior $P(z)$ and then z is decoded back to it initial value. 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 multivariate Gaussians, we can calculate $-D_{KL}(Q(Z)||P(Z))$ directly as

$$\begin{aligned} -D_{KL}(Q(Z)||P(Z)) &= \int Q(z)(\log P(z) - \log Q(Z))dz \\ &= \frac{1}{2}\sum_{j=1}^{J}(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2) \end{aligned} \tag{13}$$

This follows (D. P. Kingma & Welling, 2013, p. 10-11). To estimate these parameters μ and σ of Q, D. P. 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 and symbolic one-hot vectors, we train our decoder distribution as a multivariate Bernoulli, in which case the loss is defined as the cross-entropy between the actual data and the reconstruction:

$$\log P(x|z) = \sum_{i=1}^{D} x_i \log y_i + (1 - x_i)\log(1 - y_i) \tag{14}$$

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

$$y = f_\sigma(W_2 f(W_1 z + b_1) + b_2) \tag{15}$$

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 (D. P. Kingma & Welling, 2013, p. 11).

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. 12 as a regulization term:

$$J_\theta = \sum_{i=1}^{D} x_i \log y_i + (1 - x_i)\log(1 - y_i) + \frac{1}{2}\sum_{j=1}^{D_{\mathcal{Z}}}(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2) \tag{16}$$

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. 12, 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. D. P. 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 $p(\epsilon)$:

$$\epsilon \sim p(\epsilon) \tag{17}$$

D. P. 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:

$$\epsilon \sim \mathcal{N}(0, 1) \tag{18}$$

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

$$z = \mu + \sigma \odot \epsilon \tag{19}$$

where $\odot$ defines element-wise multiplication. The right hand side of Eq. 12 then becomes

$$E_{\epsilon \sim \mathcal{N}(0,1)}[\log P(X|z = \mu + \sigma \odot \epsilon) - D_{KL}(Q(z|X)||P(z))] \tag{20}$$

The model architecture is visualized in Fig. 2.

4.2.3 Interpretation of $\mathcal{Z}$

The question is then how to interpret such an emerging 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 /aǂan/ 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. 3, 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 actually 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 surface features, and it focusses instead to encode the more abstract latent 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

As we expect that cognates show similar phonological shape, we expect them to cluster close to each other in latent space. Following this idea, we can use clustering algorithms to find such accumulations of embeddings and label them under a shared inferred cognate class. However, we have to make sure that the number of estimated 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 hierarchical UPGMA clustering algorithm (Sokal, 1958). Here, some threshold is needed in order to derive actual labels from the inferred hierarchical cluster tree. While List (2012a) mentions a threshold of 0.6 as optimal, this should not necessarily be true for our embeddings. As this threshold would thus be the object of an expensive grid search too, this thesis proposes Affinity Propagation (Frey & Dueck, 2007) to be used instead. 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. Affinity Propagation has been used for variety of tasks where it shows good performance, as it does not make any prior assumptions on what clusters should look like. The algorithm can take any pairwise similarities as input. In our case, where we compare the embeddings with euclidean distances 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. One parameter to set beforehand is the preference of every datapoint to be chosen as an exemplar. Another parameter is the damping factor that prevents numerical oscillations that can appear when updating the messages (Frey & Dueck, 2007, p. 972).

5 Evaluation

5.1 Data

To pre-train VAE-PT and the phoneme embeddings, 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 IPA-annotated words for 250 concepts for 163 Indo-European 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 Phoneme Embeddings

5.2.1 Visual Inspection

A first visual inspection after 5 iterations over ASJP shows that the phoneme embeddings indeed capture some latent information (Fig. 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.

5.2.2 Analogy Tests

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 is the form

$$v(\mathit{phoneme}_1) + v(\mathit{phoneme}_2) - v(\mathit{phoneme}_3) \approx v(\mathit{phoneme}_4) \tag{21}$$

where $v(\mathit{phoneme}_1)$ is the vector corresponding to $\mathit{phoneme}_1$ and $v(\mathit{phoneme}_2) - v(\mathit{phoneme}_3)$ can be seen as the latent phonological information available in $\mathit{phoneme}_2$ but not in $\mathit{phoneme}_3$, while $v(\mathit{phoneme}_1)+v(\mathit{phoneme}_2)-v(\mathit{phoneme}_3)$ is then this latent phonological information added to $\mathit{phoneme}_1$, which should be close to the vector corresponding to $\mathit{phoneme}_4$. For example, in

$$v(/\mathbf{i}/) + v(/\mathbf{u*}/) - v(/\mathbf{u}/) \approx v(/\mathbf{i*}/) \tag{22}$$

$v(/\mathbf{u*}/) - v(/\mathbf{u}/)$ 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:

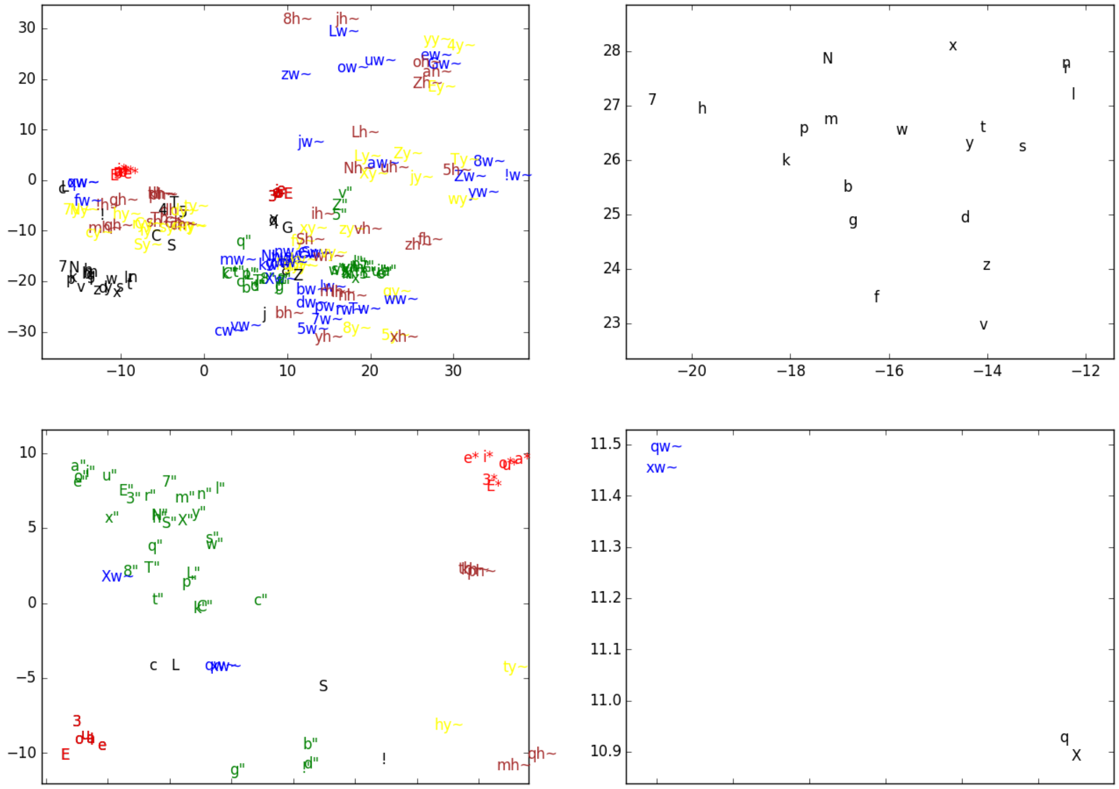

Figure 3: Some t-SNE visualizations of the embeddings created by word2vec. (top left) The model learns to clearly separate natural classes such as vowels, plain pulmonic or glottalized consonants, while other articulations seem to spread over the feature space. The colors indicate membership of a natural 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.

- 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.

Figure 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. In fact, when evaluating the correlation between frequency and performance of a given natural phoneme class, both Spearman and Pearson correlation tests indicate correlation indices of 0.9. 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 (see Figure 6).

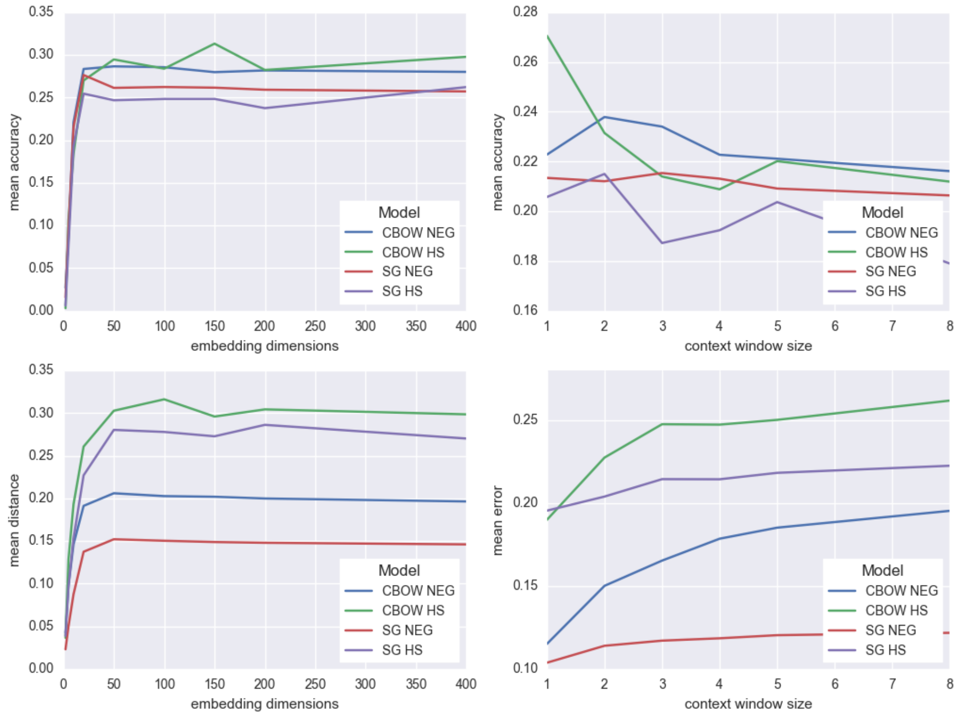

Figure 4: Comparison of the four word2vec models evaluated. (top left) Both CBOW Models perform better than the skip-gram models over all numbers of embedding dimension. (bottom left) Mean distance between the predicted vector and the target with regard to embedding dimensions. Here, negative sampling 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.

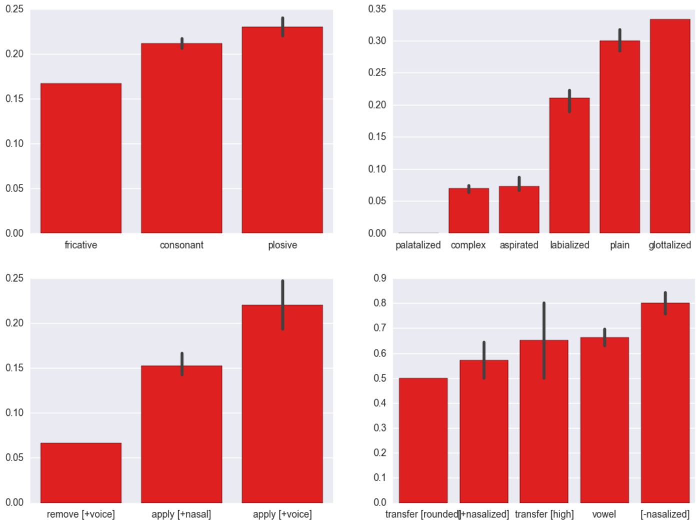

Figure 5: Mean accuracies for analogies. The model does not seem to be able to capture the compositionality of the latent features too well. (top left) Accuracy is best for plosives across plain pulmonic as well as more complex articulations, while fricatives perform worse. (top right) Glottalized and 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.

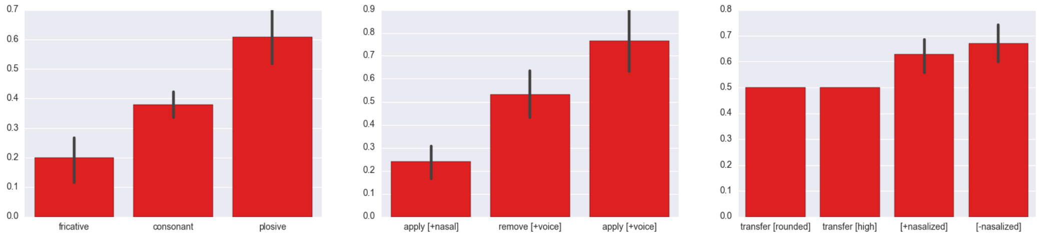

Figure 6: Mean accuracies for analogies over the pooled consonant phonemes. Here, the model is expected to yield better results, as the number of latent features should have been reduced. (left) Accuracy is doubled compared with the unpooled phonemes. (middle) Adding voice works quite well, while removing 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.

5.3 Word Embeddings

5.3.1 Visual Inspection

To gain a first visual understanding how VAE-PT structures the latent space, we ran it for 100 iterations on ASJP. Each word was encoded with the binary features from Rama (2016) and zero-padded to a maximum length of 15 phonemes. The encoding and decoding MLPs had hidden layers of 500 units each. Indeed, VAE-PT learns to cluster similar strings close to each other (Fig. 7). 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. 8), 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. 9). 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 population. 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. 10). 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. 11). Hence, if the given semantic concept only consists of members of a single cognate class, the whole posterior shows strong linear dependencies (Fig. 11 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. However, this is actually what we expected (cf. section 4.2.3). Hence, VAE-PT is excluded from the following investigations how to improve the actual clustering.

VAE-C, on the other hand, yields a far better basis for cognate identification as such (see Figure 12). 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.

5.3.2 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:

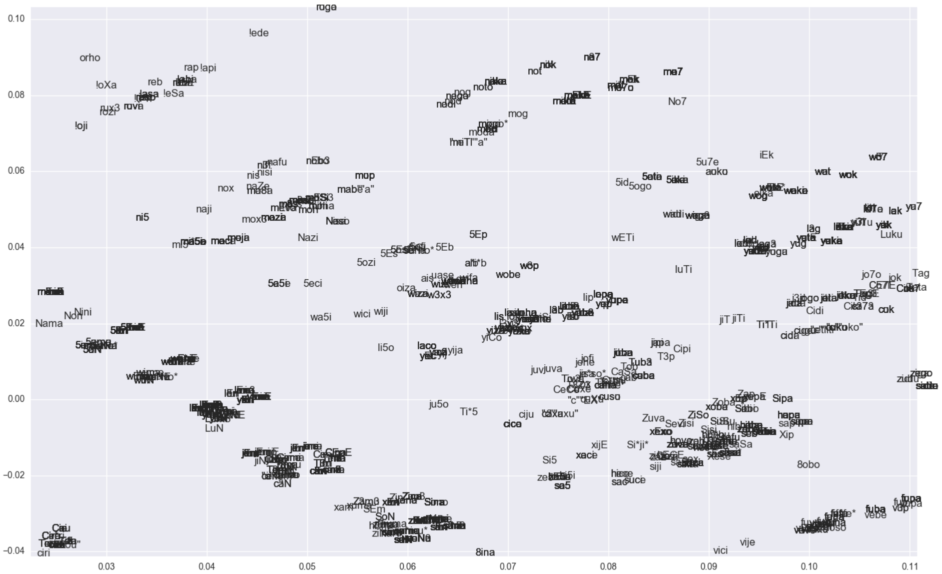

Figure 7: A detailed view on a subsample of ASJP embedded into $\mathcal{Z}$ after trained on the whole ASJP data set for 100 iterations. As can be seen, words accumulate in local subspaces with higher probability mass. Here, all words are contained in an area with shorter words that show simple CV syllable 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.

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

- 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 validation set of 10 concepts, which are randomly sampled from all IELex concepts beforehand. The models are trained for 4000 iterations. The encoding and decoding MLPs both have hidden layers of 1000 units. All words were zero-padded to a maximum length of 10 phonemes, following Rama (2016). As an optimization algorithm, we used Adam (D. Kingma & Ba, 2014). For the actual clustering, the preferences were set the median of the input similarities and the damping factor to 0.5, following (Frey & Dueck, 2007, p. 972). The performance is measured in adjusted rand indices (Rand, 1971; Hubert & Arabie, 1985).

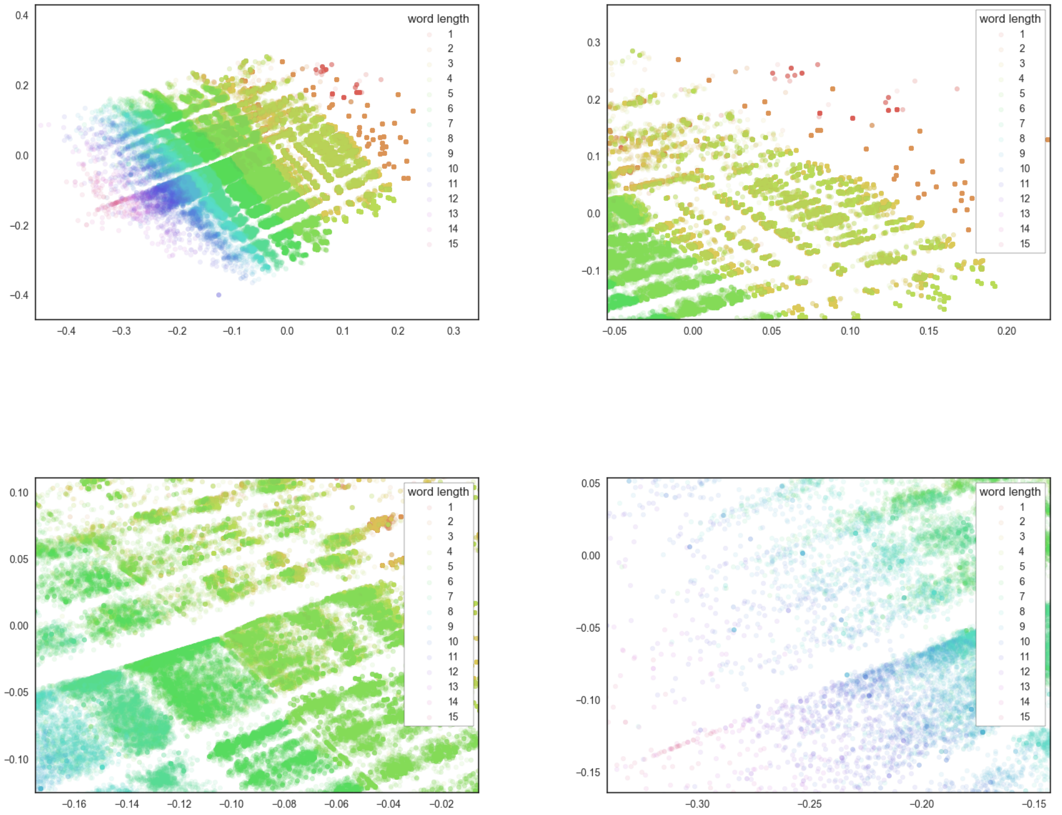

Figure 8: The ASJP data set embedded into $\mathcal{Z}$ after trained on it for 100 iterations, colored by word length. The model learns to cluster words according to their respective length, with more frequent word lengths located in subspaces with higher probability mass.

Fig. 13 summarizes the results. The binary phoneme features outperform the word2vec phoneme embeddings, which perform nevertheless much better than the sound class one hot vectors. The fact that the one-hot representations perform worse is not surprising. As they merge certain phonemes together, a lot of information is lost compared to representations that directly encode phonological features. While certain cognates might end up with the same Dolgopolsky- or SCA-encoding, similar but unrelated words might too, without the possibility for the network to tell those words apart. Hence, the network can hardly detect latent features between the words and embed them in a meaningful way, which severely decreases the clustering performance.

Also, the number of latent dimensions does not seem to influence the performance too much. While 10 latent dimensions slightly seem to outperform 20 and 50 in this grid search, higher-dimensional latent spaces are generally known to require more time to have a network converge, which could also be the case here - the 4000 iterations simply might not be enough for such higher-dimensional spaces. While we think that it is quite possible to improve the performance with an increased dimensionality of the latent space, as it should then be able to encode much more information in general, we do not think it is for practical value in our case. One the one hand, iterating a model over every semantic concept in the data set takes time. In our setup, 1000 iterations over one concept roughly took 20 seconds, varying by the number of latent dimensions and hidden dimensions of the encoding and decoding MLPs. Summed up for every single concept in the data and 4000 iterations for every such concept, it already took around 8 hours to perform that grid search from above. On the other hand, we do not expect the clustering to work efficiently in higher-dimensional space anymore anyway, as due to the the curse of dimensionality all points should then become more and more equidistant to each other, and thus clusters should be harder to identify.

The standard deviation of the auxiliary noise σ_ϵ also does not seem to influence the performance too much either. However, a rather high standard deviation seems to outperform the other configurations slightly. This might be due to the reasons discussed in section 4.2.3 - an increased auxiliary noise forces the network to learn more abstract features.

5.3.3 Performance on IELex

VAE-C was then run on the remaining 197 semantic concepts on IELex with the parameters found optimal in the grid search above, i.e. binary phoneme features, 10 latent dimensions, 1000 intermediate dimensions and $\sigma_\epsilon = 0.1$. The models iterated 8000 times over each concept, to make sure the results do not suffer from a possible non-convergence of the networks. Fig. 14 and table 1 summarizes the results, measured in adjusted rand indices, adjusted mutual information (Vinh et al., 2010), cluster homogeneity, cluster completeness, v-measure (Rosenberg & Hirschberg, 2007) as well as pairwise precision, recall and F1-score. The model indeed finds cognates and is far better than random labeling, but the overall performance is only mediocre, especially compared to LexStat. Cluster homogeneity and pairwise precision is better than completeness and recall respectively. This is a good indication, since our null hypothesis is that all words are unrelated to each other and thus we favor false negatives to false positives. However, this is common to all models compared. Also, pairwise recall is nearly identical for all models except random labeling, although their architectures differ quite a lot. This might be an indicator that cognate identification needs more data than just word lists. LexStat only shows better performance due to a very high precision. Interestingly, the performance of VAE-C is also nearly identical to the results obtained by simply using Normalized Levenshtein Distances between the IPA-annotated words to cluster them. As both models in the end use bare distances between words as input for the clustering, this might indicate that for cognate identification in particular, one has to make use of prior assumptions on how cognates should behave compared to unrelated words. The incorporation of such priors is fully possible with variational autoencoders, as described in section 2, but not explored in this thesis. Also note that random labeling still manages to gain relatively high v-measures and pairwise F1-scores, as both are not normalized with regards to the number of samples, predicted clusters and true cognate classes. In fact, most literature on cognate identification (Mackay & Kondrak, 2005; List, 2012a; Wahle, 2013; Rama, 2016) only use pairwise precision, recall and F1-score to measure model performances, but do not discuss in how far sample size, predicted and true cluster size actually influence their respective results compared to random labeling.

	ARI	AMI	Homogeneity	Completeness	V-Measure	Precision	Recall	F1
VAE-C	0.32	0.37	0.68	0.56	0.58	0.68	0.44	0.49
LexStat	**0.58**	**0.71**	**0.99**	**0.91**	**0.95**	**0.94**	**0.46**	**0.62**
NLD-AP	0.32	0.38	0.74	0.57	0.60	0.71	0.42	0.48
Random	0.00	0.01	0.42	0.30	0.32	0.42	0.18	0.22

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

6 Resume

In this thesis, we have shown how recent tools from the emerging deep learning framework can be used for unsupervised cognate identification. Here, words cannot only be compared with distances, but are themselves embedded in a latent space that encodes abstract features of words. Also phonemes, which form the underlying base for words, are embedded in a latent space and compared to other established models of phoneme vectorization. While we found that the resulting phoneme embeddings indeed cover latent features that correlate with established distinctive phonological features, they still suffer from the lack of very big training data. Words, encoded as sequences of such phoneme vectorizations, were than embedded in a latent space, using variational autoencoders with fully-connected encoder and decoder networks.

The models described here 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. However, for the task of cognate identification in particular, we did not achieve performance superior to well-established methods of unsupervised cognate identification. Instead, we found that the model rather finds distances between words that correlate to established string distances, such as Normalized Levenshtein Distance. This might be an indicator that further prior information on how cognates behave differently to other unrelated words should be incorporated into the model. While we did not make great use of Bayesian inference here in particular, we can still be sure that those models described here can be further improved by exploring different prior assumptions on the behavior of cognates and different kinds of sound change as such. Another point to optimize is the choice of the actual encoding and decoding networks, where convolutional or recurrent networks might result in better performance. We are sure that all this might be an interesting object of further research.

7 Acknowledgements

For training the phoneme embeddings, we 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 as well as running LexStat on IELex, we 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)

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798–1828.

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

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

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*.

Chater, N., & Manning, C. D. (2006). Probabilistic models of language processing and acquisition. *Trends in cognitive sciences*, *10*(7), 335–344.

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

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

Crocker, M. W. (2010). Computational psycholinguistics. *Computational Linguistics and Natural Language Processing Handbook. Wiley Blackwell, London, UK*.

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

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

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

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.

Dosovitskiy, A., Tobias Springenberg, J., & Brox, T. (2015). Learning to generate chairs with convolutional neural networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1538–1546).

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., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., ... others (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97.

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.

Kessler, B. (2007). Word similarity metrics and multilateral comparison. In *Proceedings of ninth meeting of the acl special interest group in computational morphology and phonology* (pp. 6–14).

Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Skip-thought vectors. In *Advances in neural information processing systems* (pp. 3294–3302).

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

Mackay, W., & Kondrak, G. (2005). Computing word similarity and identifying cognates with pair hidden markov models. In *Proceedings of the ninth conference on computational natural language learning* (pp. 40–47).

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.

Ng, A. (2011). Sparse autoencoder. *CS294A Lecture notes*, *72*, 1–19.

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).

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

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).

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (emnlp)* (Vol. 1631, p. 1642).

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`

Trask, R. L. (1996). *Historical linguistics.* Oxford University Press.

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.

Wahle, J. (2013). *Alignment and word comparison with pair hidden markov models* (Unpublished doctoral dissertation). Eberhard Karls Universität Tübingen.

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*.

Zen, H., & Senior, A. (2014). Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis. In *2014 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 3844–3848).

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).

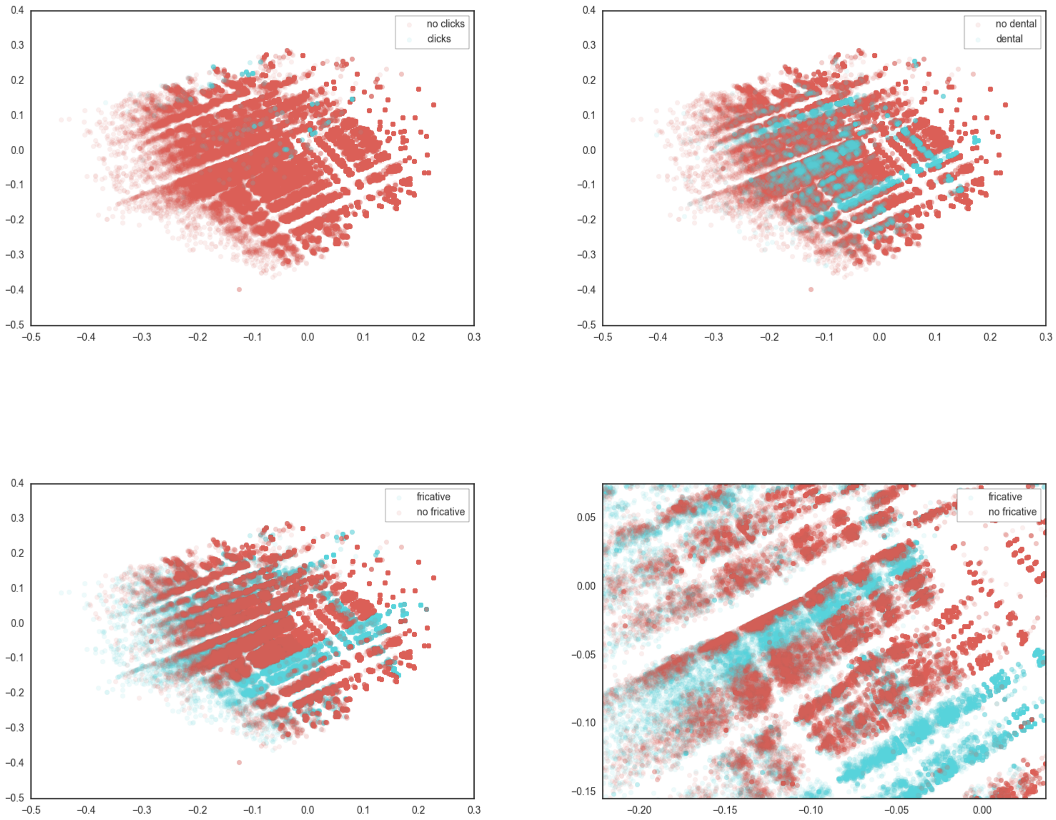

Figure 9: The ASJP data set embedded into $\mathcal{Z}$ after trained on it for 100 iterations, colored by whether a given word has a phoneme with some specific distinctive feature. (top left) The model learns that clicks are highly unlikely to emerge evolutionary and hence assigns low probability mass 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.

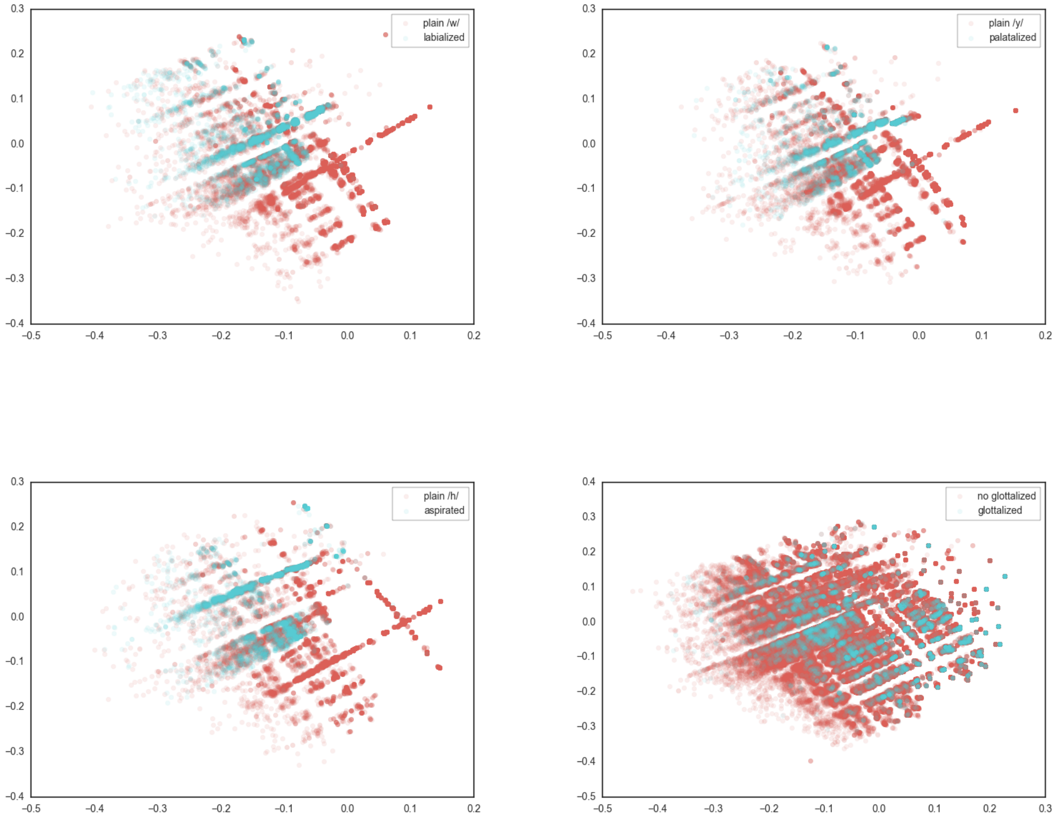

Figure 10: The ASJP data set embedded into $\mathcal{Z}$ after trained on it for 100 iterations. Since the binary features ignore coarticulations as such and encode them as independent phonemes instead, the model might learn to distinguish them, given some underlying latent features. (top left) The distribution 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.

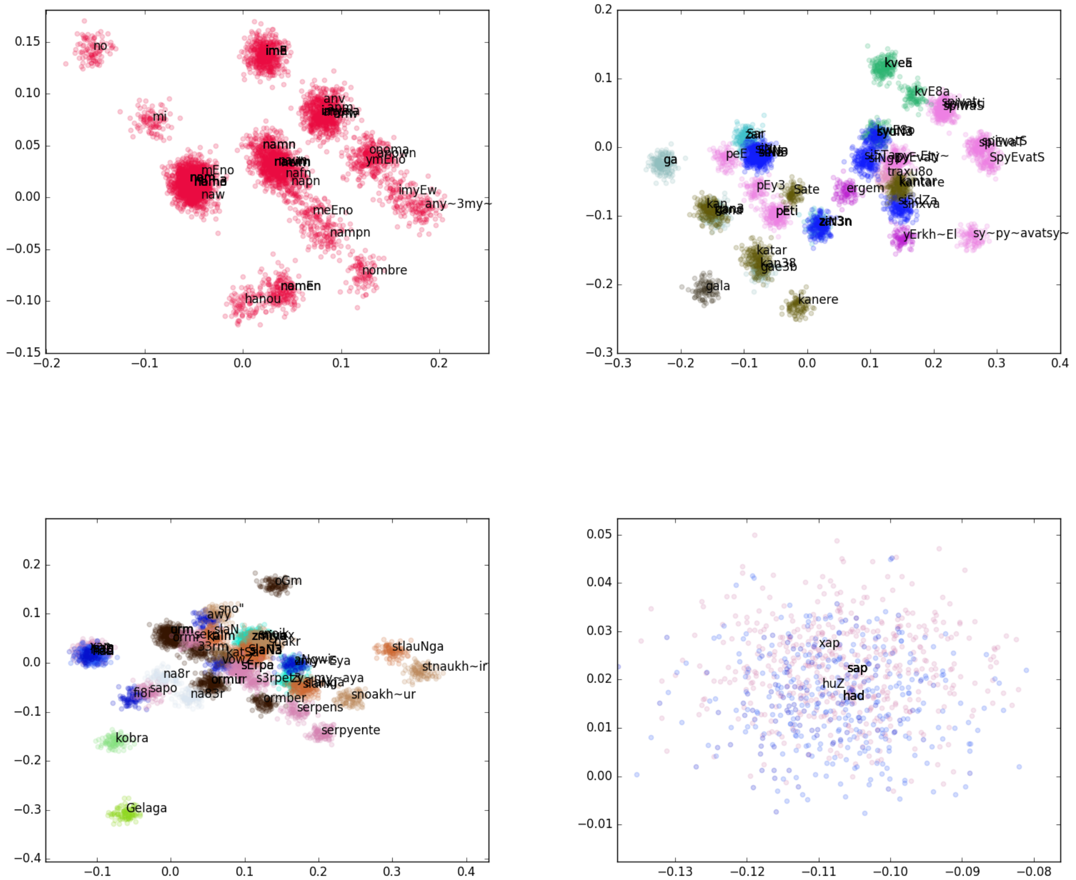

Figure 11: The encoder distribution $Q(z|X)$ with X being the IELex data with regards to the single semantic concepts. The model was trained on ASJP for 2000 iterations. Colors indicate the respective true cognate class. (top left) The posterior for the concept "name". As all words are cognates to each 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 than unrelated words. (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.

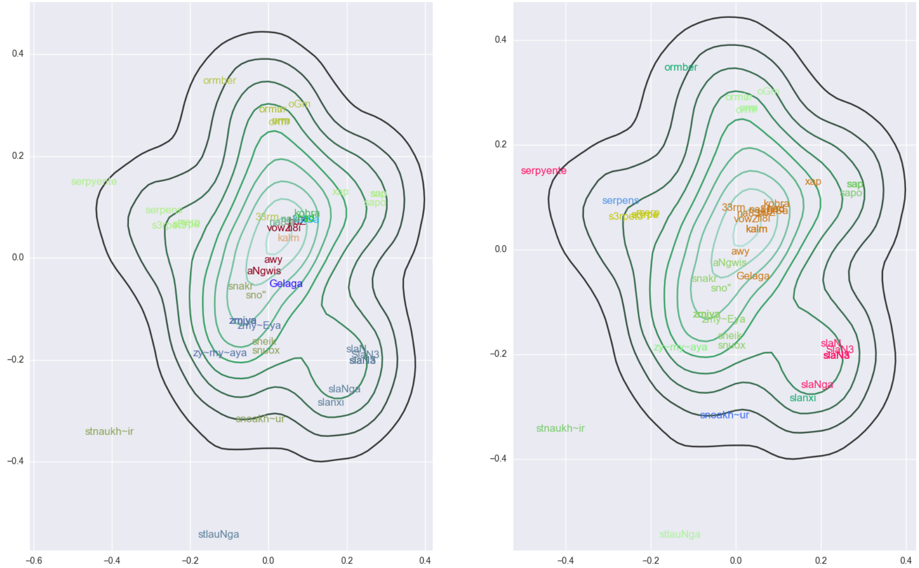

Figure 12: VAE-C trained on the IELex data for ”snake”, where every word is encoded as a sequence of binary phoneme features. The standard deviation of the auxiliary noise is set to 0.1, which results in an encoder distribution approximating a Gaussian distribution. The left plot shows true cognates 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.

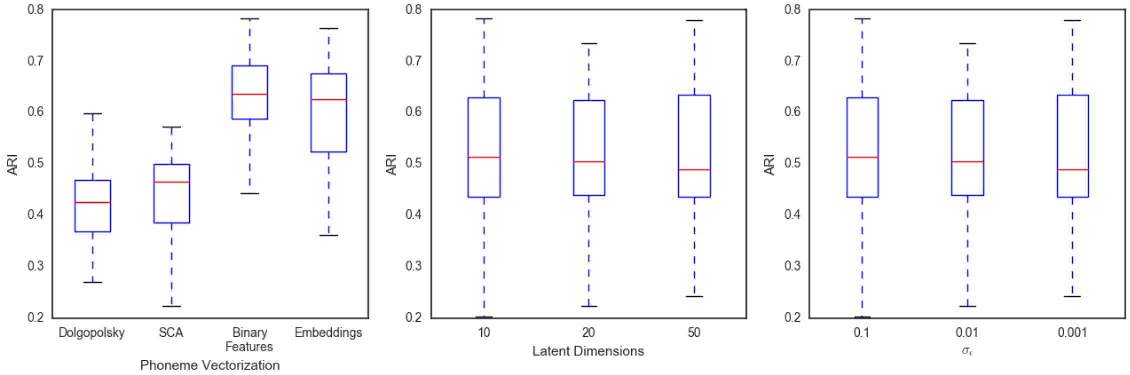

Figure 13: The result of the grid search over several parameters of VAE-C, measured in adjusted rand indices. (left) The binary phoneme features outperform the phoneme embeddings. The sound class one hot vectors only show mediocre performance. (middle) the number of latent dimensions does not seem to influence the performance too much. An increased dimensionality shows slightly worse performance, but this might due to the network not having converged yet. (right) An increased auxiliary noise coincides with better performance over all model.

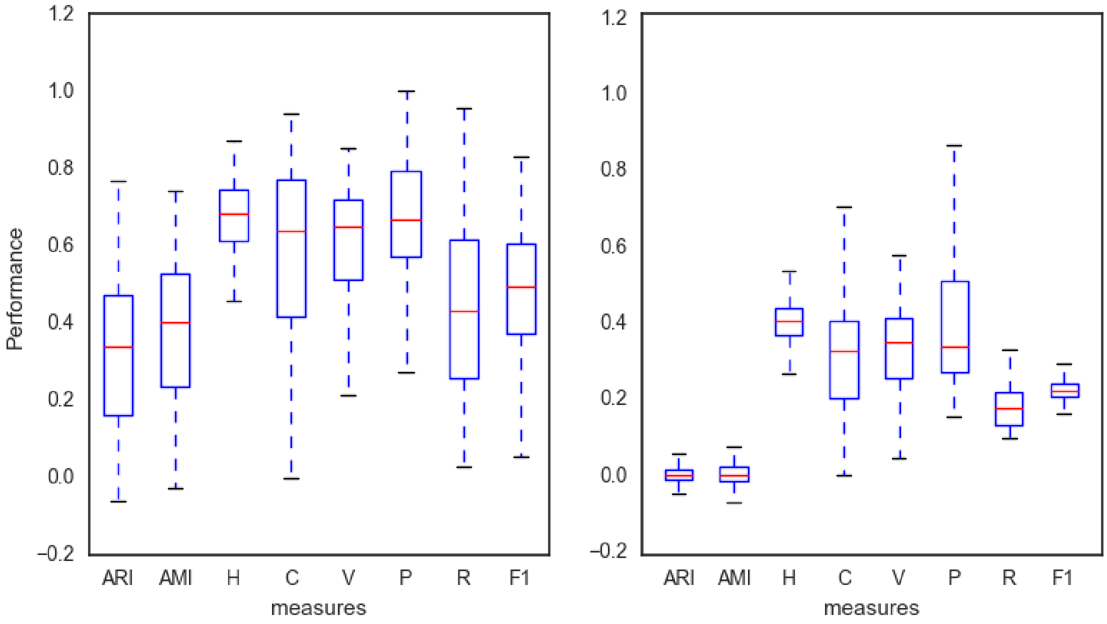

Figure 14: The performance of VAE-C on IELex, using the parameters obtained by the grid search (left), and random labeling (right). The measures are adjusted rand indices, adjusted mutual information, cluster homogeneity, cluster completeness, v-measure, pairwise precision, recall and F1-score. The model learns to cluster similar words together, although the overall performance is mediocre.