Cross-Linguistic Neural Phoneme Embeddings for Computational Historical and Typological Linguistics

Marlon Betz

July 11, 2016

Contents

1 Introduction **2**

2 Theoretical and Practical Motivation **3**

3 Related Research **3**

4 Embedding Models **3**
4.1 Neural Embedding Models 3
4.1.1 Word2Vec 5

5 Evaluation **12**
5.1 Data 12
5.2 Evaluation Methods 12
5.3 Results 12

6 Use Cases **12**
6.1 Phonemic String Comparison 12
6.2 Modeling Sound Change 12
6.3 Phoneme Inventory Clustering 12

7 Resume **12**

1 Introduction

Embeddings nowadays build the backbone of every Deep Learning NLP architecture. Due to their capacity to encode a vast amount of latent semantic and syntactic information without the need of previous manual construction, they gave rise to the contemporary Deep Learning boom, i.e. deep neural networks that can capture hidden features in data sets and by that have allowed for huge performance gains in numerous NLP fields.
While embeddings are currently being used for several linguistic units such as characters [Kim et al.2015, dos Santos and Zadrozny2014, Zhang et al.2015], words [Mikolov et al.2013a,Mikolov et al.2013b,Pennington et al.2014] or entire sentences [Kiros et al.2015], proper phonemes have been largely been excluded from this trend, although there are some data that would indeed allow for it and research areas where they could be of great use, especially in the fields of Computational Typological and Historical Linguistics, which the current deep learning boom has hardly touched yet. There, even though it is clear that some phonemes share more common features and such form natural classes, they are often treated as pure symbols that share a common distance between each other. Even phoneme representation models that do incorporate phonological features are often handcrafted [Kondrak2000,Rama2016] or include task-specific information that inherently suffer

from restricted generalization abilities when used for other tasks [Jäger2014]. Moreover, those methods usually reduce the number of possible phonemes to a minimum, getting rid of important information such as secondary or co-articulations.
In this paper, I will first a discuss a theoretical motivation for using data-driven phoneme embeddings instead of plain symbolic representations or hand-crafted feature encodings. I will then shortly take a look on related research. A big part of this paper will then review several embedding models. I will then discuss intrinsic evaluation methods for the embeddings and use those to compare the performances of the previously described models. This is then followed by a discussion of several use cases that could be interesting for those interested in data-driven approaches to Typological and Historical Linguistics. Finally, I will recapitulate the benefits and drawbacks of phoneme embeddings in a final resume.

2 Theoretical and Practical Motivation

3 Related Research

Word embeddings have proven to be a reliable tool for several NLP tasks. However, finer grained embedding models that take care of the internal structure of a word have been shown to give similar or even improved performance over traditional word embeddings as they are able to encode morphological [dos Santos and Zadrozny2014] or semantic [Chen et al.2015b] information even for out-of-vocabulary items [Ling et al.2015]. This follows the simple idea that since a word can be split up into its respective morphemes that encode morphosyntactic and semantic information, encoding subparts of a word instead of a the whole word should allow for better generalization for unseen words without the need to compute and store vector representations for every word type.

4 Embedding Models

4.1 Neural Embedding Models

Models that make use of cooccurrence count matrix factorizations usually suffer from the need of storing huge amounts of data into working memory. Neural networks with hidden layers, on the other hand, have been proven to be able to approximate any given hidden function while they still allow for online or mini-batch training. This is a huge improvement over models that only work with whole data batches, as much less working memory has to be used here. [Bengio et al.2003a] pioneered as the first to introduce Neural Language Models in general. Here, a network with two hidden layers is proposed, where a first non-linear hidden layer with locally shared weights gives embeddings of single words in the given context window, while the next layer is a concatenation of the single word embeddings and another non-linear activation function and gives a context embedding. The last layer than

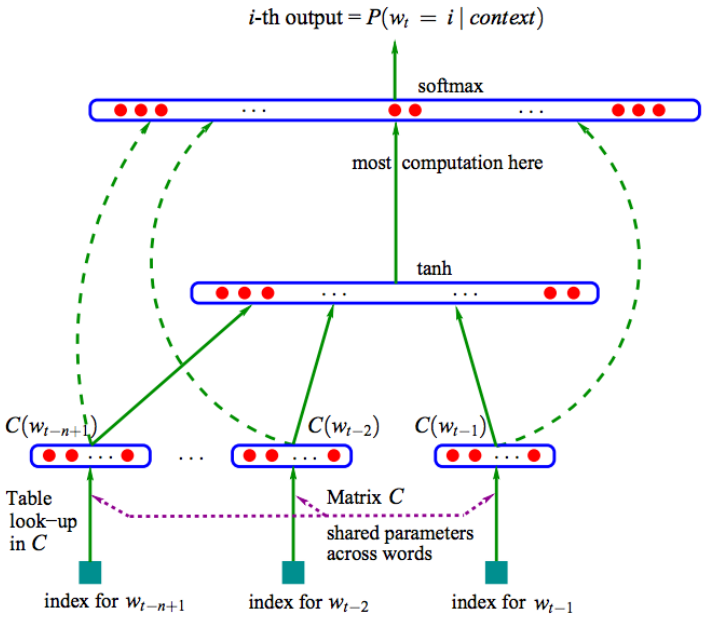

Figure 1: The neural language model as proposed in [Bengio et al.2003a].

is a softmax classifier that predicts the next word (cf. Figure 1). It later became clear that this model had two major drawbacks:

- The softmax classifier needs to compute the probability of all word types in the vocabulary. This is a major flaw also of following models (see below).
- The non-linear hidden layer severely worsened the convergence rate of the network.

[Collobert and Weston2008] then improved the model tremendously by getting rid of the expensive softmax classifier by introducing a score based loss function:

$$J_\theta = \sum_{x \in X} \sum_{w \in V} \max(0, 1 - f_\theta(x) + f_\theta(x^{(w)})) \tag{1}$$

Here, the the correct windows x containing n words are sampled from the set of all possible windows X in the corpus, while for every window x, a corrupted version $x^{(w)}$ is produced by replacing the center word of x by the another word of the vocabulary V. The objective then becomes maximizing the distance between the scores output by the model fore correct and the incorrect window with a margin of 1. The actual model then was still the same as in [Bengio et al.2003a]. This model could already embed semantically similar words close to each other, but still suffered from the non-linear hidden layer, which slowed down learning

enormously ([Bengio et al.2003a] report a training time of 7 weeks for a vocabulary of 130000 word types)

4.1.1 Word2Vec

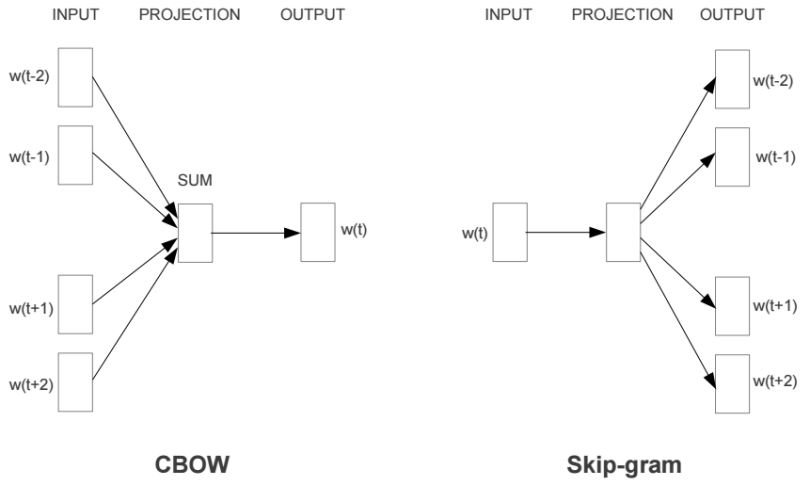

Figure 2: The two common architectures of word2vec. The Continuous Bag-of-Words (CBOW) model predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word. From [Mikolov et al.2013a].

Among general-purpose embedding models in general and among neural embedding models in particular, the word2vec algorithm [Mikolov et al.2013a, Mikolov et al.2013b] stands out as major breakthrough, as it offers good performance while the parameters to train are less then with the previous models. Different to the previous models in [Bengio et al.2003a, Collobert and Weston2008], which employ non-linear hidden layers, here the model only consists of a linear encoder layer that computes the actual embeddings. It can be seen as a full departure from traditional embedding models as a subgroup of language models towards a new kind of model family that in the first place tries to encode semantic and morphosyntactic information of a word rather than to predict the next word itself. Instead, the context window usually encompasses all surrounding words of a given target word instead of just the previous words. This makes it rather unhandy for true language modeling, but allows for much better incorporation of latent information. Moreover, the model can be trained in two ways: It either tries to predict the target word given its context (Continuous Bag-of-Words; CBOW) or tries to predict the context of a word given the target word itself (Skip-Gram, cf. Figure 2). The corresponding loss function for the

CBOW model is given as

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log p(w_t \mid w_{t-n}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+n}) \tag{2}$$

where $p(w_t \mid w_{t-1}, \ldots, w_{t-n+1})$ is given as a softmax

$$p(w_t \mid w_{t-1}, \ldots, w_{t-n+1}) = \frac{\exp(h^\top v'_{w_t}))}{\sum_{w_i \in V} \exp(h^\top v'_{w_i})} \tag{3}$$

while the loss for the Skip-Gram model is given as

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \sum_{-n \leq j \leq n, \neq 0} \log p(w_{t+j} \mid w_t) \tag{4}$$

where $p(w_{t+j} \mid w_t)$ is again defined as a softmax

$$\log p(w_{t+j} \mid w_t) = \frac{\exp(v_{w_t}^\top v'_{w_{t+j}}))}{\sum_{w_i \in V} \exp(h^\top v'_{w_i})} \tag{5}$$

where v_{wi} is an input vector and v'_{wi} an output vector representation of a word w_i and $h^\top$ the transpose of the sum of the output vectors of the words in the context. Again, the calculation of the softmax here includes summing over all contexts in the corpus, which computationally is expensive. Further research largely concentrated on finding a way to define p as a memory- and time-friendly approximation of the true softmax probability. A first solution was given by using a hierarchical softmax classifier [Morin and Bengio2005] nstead. Here, the classifier layer is a binary tree with the actual words as leaves. At every node, the network learns to either follow the left or the right branch with a certain probability as in Eq. 5 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:

$$p(right|n, c) = \sigma(h^\top v'_n) \tag{6}$$

This means 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 vocabulary entries. 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 way of approximation the softmax is using Differentiated Softmax (cf. [Chen et al.2015a]), where the idea is that rare words should help to adjust less parameters than more frequent words. The fully connected softmax layer becomes a sparse matrix here, where certain blocks inside of that matrix are used to predict words with a certain frequency.

[Chen et al.2015a] mention their model is fast and accurate, but has less predictive power with rare words, as they are given less parameters to be predicted.

Another family of models approximate the true softmax via sampling. Here, it seems to be common to reformulate the softmax defintions from above. For instance, if we use negative log probabilities instead of raw probabilities and if we further use a more general score function $\mathcal{E}(w)$ instead of the more specific dot products of the context and word vectors and start with Eq. 3, the loss function J_θ becomes

$$J_\theta = \mathcal{E}(w) + \log \sum_{w_i \in V} \exp(-\mathcal{E}(w_i)) \tag{7}$$

For back-propagating the error, we calculate the gradients of this term:

$$\nabla_\theta J_\theta = \nabla_\theta \mathcal{E}(w) + \nabla_\theta \log \sum_{w_i \in V} \exp(-\mathcal{E}(w_i)) \tag{8}$$

$$\nabla_\theta J_\theta = \nabla_\theta \mathcal{E}(w) + \frac{1}{\sum_{w_i \in V} \exp(-\mathcal{E}(w_i))} \nabla_\theta \sum_{w_i \in V} \exp(-\mathcal{E}(w_i)) \tag{9}$$

$$\nabla_\theta J_\theta = \nabla_\theta \mathcal{E}(w) + \frac{1}{\sum_{w_i \in V} \exp(-\mathcal{E}(w_i))} \sum_{w_i \in V} \nabla_\theta \exp(-\mathcal{E}(w_i)) \tag{10}$$

$$\nabla_\theta J_\theta = \nabla_\theta \mathcal{E}(w) + \frac{1}{\sum_{w_i \in V} \exp(-\mathcal{E}(w_i))} \sum_{w_i \in V} \exp(-\mathcal{E}(w_i)) \nabla_\theta (-\mathcal{E}(w_i)) \tag{11}$$

which can be rewritten as

$$\nabla_\theta J_\theta = \nabla_\theta \mathcal{E}(w) + \sum_{w_i \in V} \frac{\exp(-\mathcal{E}(w_i))}{\sum_{w_i \in V} \exp(-\mathcal{E}(w_i))} \nabla_\theta (-\mathcal{E}(w_i)) \tag{12}$$

Here, we can replace the softmax in the second term by $P(w_i)$ and reposition the negative coefficient:

$$\nabla_\theta J_\theta = \nabla_\theta \mathcal{E}(w) - \sum_{w_i \in V} P(w_i) \nabla_\theta (\mathcal{E}(w_i)) \tag{13}$$

As one can see, we basically end up with a positive and a negative reinforcement term. As the negative reinforcement term is basically the gradient of $\mathbb{E}_{w_i \sim P}$, the final reformulation of the softmax is then given as in Eq. 14.

$$\nabla_\theta J_\theta = \nabla_\theta \mathcal{E}(w) - \mathbb{E}_{w_i \sim P}[\nabla_\theta \mathcal{E}(w_i)] \tag{14}$$

The goal of sampling-based approaches is now to approximate the negative reinforcement term $\mathbb{E}_{w_i \sim P}[\nabla_\theta \mathcal{E}(w_i)]$ without going over the whole vocabulary. Earlier models used some forms of Importance Sampling [Liu2008, Bengio et al.2003b, Bengio and Senécal2008,

Cho et al.2015], where a proposal distribution Q is used to approximate P. Although Importance sampling seems to provide a speed-up over the standard softmax, the model suffers from divergence issues. Another group of sampling models does not try to estimate the probability of a word directly but uses an auxiliary loss that as a by-product produces probability estimates of correct words. Noise Contrastive Estimation (NCE, [Gutmann and Hyvärinen2010, Mnih and Teh2012]) trains a model to differentiate between a true word w_i and noise data $\tilde{w}_{ik}$, where k is the number of noisy data points. As a start point, one can use a logistic regression loss that minimized the negative log-likelihood as in Eq 15, where Q is a noise distribution:

$$J_\theta = -\sum_{w_i \in V} [\log P(y=1|w_i, c_i) + k\mathbb{E}_{\tilde{w}_{ik}\sim Q}[\log P(y=0|\tilde{w}_{ik}, c_i)]] \tag{15}$$

Since we want to avoid summing over all words in the vocabulary to estimate $\mathbb{E}_{\tilde{w}_{ik}\sim Q}$, we substitute it with the mean as a Monte Carlo estimate, which leaves us with the sum over all log probabilities of the k noisy words:

$$J_\theta = -\sum_{w_i \in V} [\log P(y=1|w_i, c_i) + \sum_{j=1}^{k} \log P(y=0|\tilde{w}_{ik}, c_i)] \tag{16}$$

As w_i is sampled from the true Distribution P while $\tilde{w}_{ik}$ comes from the noise distribution Q, this leaves us with a mixture model, where the probability of sampling a specific word with the corresponding label (that is, whether it comes from the true distribution in the training set or is noise) can be given as

$$P(y, w|c) = \frac{1}{k+1}P(w|c) + \frac{k}{k+1}Q(w) \tag{17}$$

This can then be used to estimate the probability of the word to come from the true distribution:

$$P(y=1|w, c) = \frac{\frac{1}{k+1}P(w|c)}{\frac{1}{k+1}P(w|c) + \frac{k}{k+1}Q(w)} \tag{18}$$

which can be simplified to

$$P(y=1|w, c) = \frac{P(w|c)}{kP(w|c) + Q(w)} \tag{19}$$

Here, instead of taking the traditional softmax for $P(w|c)$, NCE allows for a reparametrization of it: instead of summing over all probabilities in the lexicon to normalize the data, the denominator of the softmax is substituted by a parameter $Z(c)$ that is estimated by the model itself. In fact, and report equal performance while setting $Z(c)$ to 1, that is, the model learns to normalize the probabilities completely on its own. Coming from the

definition of this probability in the CBOW architecture from Eq. 3, this leaves us with the probability of a word given a context as

$$P(w|c) = \exp(h^\top v'_w) \tag{20}$$

Inserting this into Eq. 19, we have

$$P(y = 1|w, c) = \frac{\exp(h^\top v'_w)}{\exp(h^\top v'_w) + kQ(w)} \tag{21}$$

which can be inserted into the logistic loss from Eq. 16 to yield the final NCE loss:

$$J_\theta = -\sum_{w_i \in V} [\log \frac{\exp(h^\top v'_{w_i})}{\exp(h^\top v'_{w_i}) + kQ(w_i)} + \sum_{j=1}^{k} \log(1 - \frac{\exp(h^\top v'_{\tilde{w}_{ij}})}{\exp(h^\top v'_{\tilde{w}_{ij}}) + kQ(\tilde{w}_{ij})})] \tag{22}$$

The probably most prominent sampling model, however, is the Negative Sampling Model (NEG). Building on NCE, it diverges from it insofar as it naively assumes that $kQ(w) = 1$. This yields Eq. 21 as

$$P(y = 1|w, c) = \frac{\exp(h^\top v'_w)}{\exp(h^\top v'_w) + 1} \tag{23}$$

which is equal to the NCE loss as long as $k = |V|$ and Q is uniform. Eq. 23 can then be reformulated as a logistic sigmoid:

$$P(y = 1|w, c) = \frac{1}{1 + \exp(-h^\top v'_w)} \tag{24}$$

Inserted into the logistic regression loss, this yields

$$J_\theta = -\sum_{w_i \in V} [\log \frac{1}{1 + \exp(-h^\top v'_w)} + \sum_{j=1}^{k} \log(1 - \frac{1}{1 + \exp(-h^\top v'_{\tilde{w}_{ij}})})] \tag{25}$$

By simplification and substitution of the sigmoid term with $\sigma = \frac{1}{1+\exp(-x)}$, this derives the NEG loss:

$$J_\theta = -\sum_{w_i \in V} [\log \sigma(h^\top v'_w) + \sum_{j=1}^{k} \log \sigma(-h^\top v'_{\tilde{w}_{ij}})] \tag{26}$$

For a more in-depth analysis of NEG, see [Goldberg and Levy2014].

As NEG is only equivalent to NCE when $k = |V|$ holds and Q is uniform, this has a large impact on the performance of NEG when when used as a proper language model. As it cannot approximate the true softmax in practice, it gives poor performance on typical

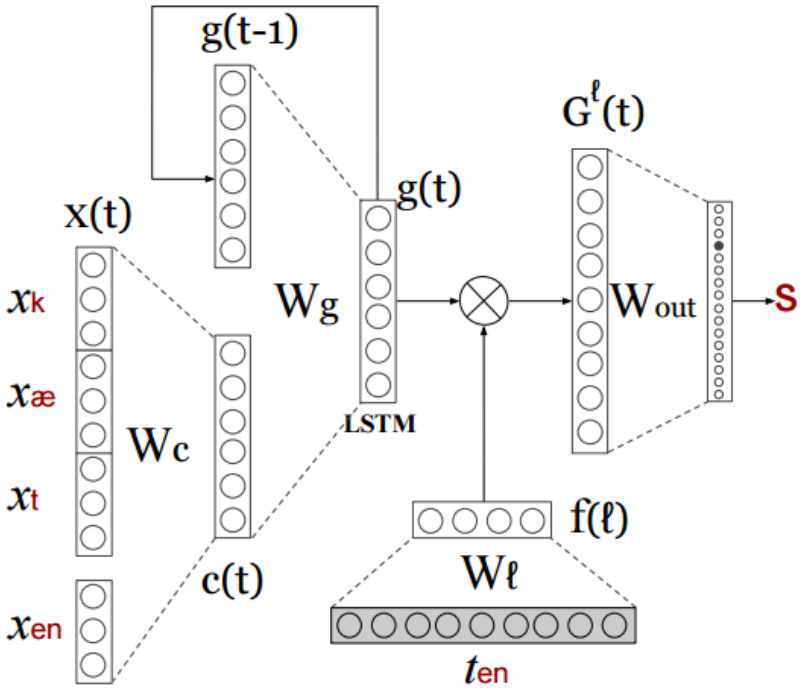

Figure 3: The Polyglot Language Model. From [Tsvetkov et al.2016]

intrinsic language model evaluation tasks such as perplexity estimation, while it shows superior performance on embedding-specific evaluations such as similarity measures.

One of the reasons why word2vec has become the most popular neural embedding model (and probably the most popular embedding model in general) is that it does not involve any hidden layers that would affect the training speed performance. However, with the rise of Long Short-term Memory Networks (LSTM, [Hochreiter and Schmidhuber1997]), there is an increase of research on embeddings trained jointly with LSTMs to predict tokens or whole sequences [Chen et al.2015b,Ling et al.2015,Kim et al.2015,dos Santos and Zadrozny2014], while some learn them jointly with convolutional models [Zhang et al.2015]. One family of models important for our case are Polyglot Neural Language Models (PLM) [Tsvetkov et al.2016]. Here, a deep model tries to predict the next phoneme given the previous phonemes, the language of the whole sequence as well as typological information from WALS [Dryer and Haspelmath2013], PHOIBLE [Moran et al.2014] and Ethnologue [Lewis et al.2015] (see Fig. 3). The model first embeds the whole context into a hidden local context representation c_t

$$c_t = W_{x_x} x_t + W_{c_{lang}} x_{lang} + b_c \tag{27}$$

where x_t is a character embedding for character c_t at position t, x_{lang} is a one-hot vector for the respective language of the sequence and b_c a bias for the character c. This then is fed into an LSTM to yield a global context layer g_t

$$g_t = LSTM(c_t, g_{t-1}) \tag{28}$$

Typological language information (such as if a single segment is represented in the phoneme inventory of a given language, or if there is a minimal contrast between the segment and another one in the phoneme inventory) is first mapped into a low-dimensional space to obtain a dense representation f_ℓ

$$f_\ell = \tan(W_\ell t_\ell + b_\ell) \tag{29}$$

and then multiplied with the global context vector using Hadamard multiplication, yielding a gloval context-language matrix G_t^ℓ

$$G_t^\ell = g_t \otimes f_\ell^\top \tag{30}$$

The resulting matrix is then vectorized into a column vector and and output sequence is computed:

$$p(\phi_t = i | \phi_i, \ldots, \phi_{t-1}, \ell) = softmax(W_{out} vec(G_t^\ell) + b_{out})_i \tag{31}$$

Accordingly, the loss function here is a categorical cross-entropy $H(\phi, \hat{\phi})$ over all phonemes:

$$H(\phi, \hat{\phi}) = -\sum_i \hat{\phi}_i \log \phi_i \tag{32}$$

The authors report superior perplexity over a baseline LSTM model, even with the typological information left out. Also downstream experiments on loan word detection and speech synthesis leave the PLM model superior over the baseline LSTM model.

5 Evaluation

5.1 Data

5.2 Evaluation Methods

5.3 Results

6 Use Cases

6.1 Phonemic String Comparison

6.2 Modeling Sound Change

6.3 Phoneme Inventory Clustering

7 Resume

References

[Bengio et al.2003a] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003a). A neural probabilistic language model. *journal of machine learning research*, 3(Feb):1137–1155.

[Bengio and Senécal2008] Bengio, Y. and Senécal, J.-S. (2008). Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks*, 19(4):713–722.

[Bengio et al.2003b] Bengio, Y., Senécal, J.-S., et al. (2003b). Quick training of probabilistic neural nets by importance sampling. In *AISTATS*.

[Chen et al.2015a] Chen, W., Grangier, D., and Auli, M. (2015a). Strategies for training large vocabulary neural language models. *arXiv preprint arXiv:1512.04906*.

[Chen et al.2015b] Chen, X., Xu, L., Liu, Z., Sun, M., and Luan, H. (2015b). Joint learning of character and word embeddings. In *Proceedings of IJCAI*, pages 1236–1242.

[Cho et al.2015] Cho, S. J. K., Memisevic, R., and Bengio, Y. (2015). On using very large target vocabulary for neural machine translation.

[Collobert and Weston2008] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

[dos Santos and Zadrozny2014] dos Santos, C. N. and Zadrozny, B. (2014). Learning character-level representations for part-of-speech tagging. In *ICML*, pages 1818–1826.

[Dryer and Haspelmath2013] Dryer, M. S. and Haspelmath, M., editors (2013). *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

[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*.

[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.

[Hochreiter and Schmidhuber1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

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

[Kim et al.2015] Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2015). Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

[Kiros et al.2015] Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

[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.

[Lewis et al.2015] Lewis, M. P., Simons, G. F., and Fennig, C. D. (2015). Ethnologue: Languages of the world. dallas, texas: Sil international. online version.

[Ling et al.2015] Ling, W., Luís, T., Marujo, L., Astudillo, R. F., Amir, S., Dyer, C., Black, A. W., and Trancoso, I. (2015). Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

[Liu2008] Liu, J. S. (2008). *Monte Carlo strategies in scientific computing*. Springer Science & Business Media.

[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*.

[Moran et al.2014] Moran, S., McCloy, D., and Wright, R., editors (2014). *PHOIBLE Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

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

[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*.

[Tsvetkov et al.2016] Tsvetkov, Y., Sitaram, S., Faruqui, M., Lample, G., Littell, P., Mortensen, D., Black, A. W., Levin, L., and Dyer, C. (2016). Polyglot neural language models: A case study in cross-lingual phonetic representation learning. *arXiv preprint arXiv:1605.03832*.

[Zhang et al.2015] Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.