Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)
Sister Conferences Best Papers Track

Specializing Word Embeddings (for Parsing) by Information Bottleneck (Extended Abstract) *

Xiang Lisa Li and **Jason Eisner**
Department of Computer Science
Johns Hopkins University
xli150@jhu.edu, jason@cs.jhu.edu

Abstract

Pre-trained word embeddings like ELMo and BERT contain rich syntactic and semantic information, resulting in state-of-the-art performance on various tasks. We propose a very fast variational information bottleneck (VIB) method to nonlinearly compress these embeddings, keeping only the information that helps a discriminative parser. We compress each word embedding to either a discrete tag or a continuous vector. In the *discrete* version, our automatically compressed tags form an alternative tag set: we show experimentally that our tags capture most of the information in traditional POS tag annotations, but our tag sequences can be parsed more accurately at the same level of tag granularity. In the *continuous* version, we show experimentally that moderately compressing the word embeddings by our method yields a more accurate parser in 8 of 9 languages, unlike simple dimensionality reduction.

1 Introduction

Word embedding systems like BERT and ELMo use spelling and context to obtain contextual embeddings of word tokens. These systems are trained on large corpora in a task-independent way. The resulting embeddings have proved to then be useful for both syntactic and semantic tasks, with different layers of ELMo or BERT being somewhat specialized to different kinds of tasks [Peters *et al.*, 2018b; Goldberg, 2019]. State-of-the-art performance on many NLP tasks can be obtained by fine-tuning, i.e., back-propagating task loss all the way back into the embedding function [Peters *et al.*, 2018a; Devlin *et al.*, 2018].

In this paper, we explore what task-specific information appears in the embeddings *before* fine-tuning takes place. We focus on the task of dependency parsing, but our method can be easily extended to other syntactic or semantic tasks. Our method compresses the embeddings by extracting just their syntactic properties—specifically, the information needed to reconstruct parse trees (because that is our task). Our nonlinear, stochastic compression function is explicitly trained

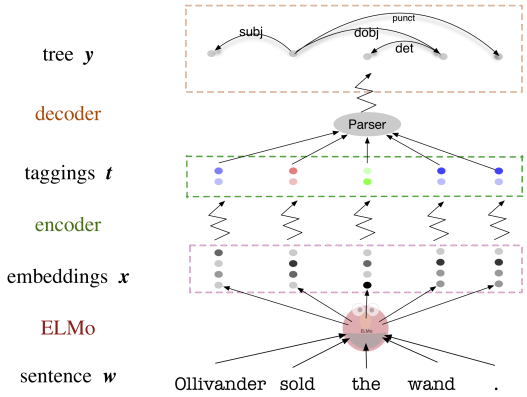


Figure 1: Our instantiation of the information bottleneck, with bottleneck variable T. A jagged arrow indicates a stochastic mapping, i.e. the jagged arrow points from the parameters of a distribution to a sample drawn from that distribution.

by variational information bottleneck (VIB) to forget task-irrelevant information. This is reminiscent of canonical correspondence analysis [Anderson, 2003], a method for reducing the dimensionality of an input vector so that it remains predictive of an output vector, although we are predicting an output tree instead. However, VIB goes beyond mere dimensionality reduction to a fixed lower dimensionality, since it also avoids unnecessary *use* of the dimensions that are available in the compressed representation, blurring unneeded capacity via randomness. The effective number of dimensions may therefore vary from token to token. For example, a parser may be content to know about an adjective token only that it is adjectival, whereas to find the dependents of a verb token, it may need to know the verb's number and transitivity.

We try compressing to both discrete and continuous task-specific representations. Discrete representations yield an interpretable clustering of words. We also extend information bottleneck to control the contextual specificity of the token embeddings, making them more like type embeddings.

This specialization method is complementary to the previous fine-tuning approach. Fine-tuning introduces *new* information into word embeddings by backpropagating the loss, whereas the VIB method learns to exploit the *existing* information found by the pretrained language model. VIB also has less danger of overfitting, since it fits fewer parameters than

*Originally published in Proceedings of the 2019 Conference on EMNLP-IJCNLP [Li and Eisner, 2019]

fine-tuning. VIB is also very fast to train on a single GPU.

We discover that our syntactically specialized embeddings are predictive of the gold POS tags, validating the intuition that a POS tag summarizes a word token's *syntactic* properties. However, our representations are tuned explicitly for discriminative parsing, so our discrete tags prove to be even more useful for this task than POS tags, even at the same level of granularity. Our continuous tags are also more useful than the uncompressed ELMo representations, when it comes to generalizing to test data.

2 Formal Model

The information bottleneck (IB) method originated in information theory and has been adopted by the machine learning community as a training objective [Tishby *et al.*, 2000; Tishby and Zaslavsky, 2015]

Let X represent an "input" random variable such as a sentence, and Y represent a correlated "output" random variable such as a parse. Suppose we know the joint distribution $p(X, Y)$. (In practice, we will use the empirical distribution over a sample of (x, y) pairs.) Our goal is to learn a stochastic map $p_\theta(t \mid x)$ from X to some compressed representation T, which in our setting will be something like a tag sequence. IB seeks to minimize

$$\mathcal{L}_{IB} = -\,\mathrm{I}(Y;T) + \beta \cdot \mathrm{I}(X;T) \tag{1}$$

where $\mathrm{I}(\cdot;\cdot)$ is the mutual information. A low loss means that T does not retain very much information about X (the second term), while still retaining enough information to predict Y.[1] The balance between the two MI terms is controlled by a Lagrange multiplier β. By increasing β, we increase the pressure to keep $\mathrm{I}(X;T)$ small, which "narrows the bottleneck" by favoring compression over predictive accuracy $\mathrm{I}(Y;T)$.

We extend the original IB objective (1) and add terms $\mathrm{I}(T_i; X|\hat{X}_i)$ to control the context-sensitivity of the extracted tags. Here T_i is the tag associated with the ith word, X_i is the ELMo token embedding of the ith word, and $\hat{X}_i$ is the same word's ELMo type embedding (before incorporating context).

$$\mathcal{L}_{IB} = -\,\mathrm{I}(Y;T) + \beta\,\mathrm{I}(X;T) + \gamma \sum_{i=1}^{n} \mathrm{I}(T_i; X|\hat{X}_i) \tag{2}$$

In this section, we justify the additional term and sketch how to efficiently estimate variational bounds on all terms (lower bound for $\mathrm{I}(Y;T)$ and upper bound for the rest).

We instantiate the variational IB (VIB) estimation method [Alemi *et al.*, 2016] on our dependency parsing task, as illustrated in Figure 1. We compress a sentence's word embeddings X_i into continuous vector-valued tags or discrete tags T_i ("encoding") such that the tag sequence T retains maximum ability to predict the dependency parse Y ("decoding"). Our chosen architecture compresses each X_i independently using the same stochastic, information-losing transformation.

The IB method introduces the new random variable T, the tag sequence that compresses X, by defining the conditional distribution $p_\theta(t \mid x)$. In our setting, p_θ is a stochastic tagger, for which we will adopt a parametric form (§ 2.1 below). Its parameters θ are chosen to minimize the IB objective (2). By IB's independence assumption,[1] the joint probability can be factored as $p_\theta(x, y, t) = p(x) \cdot p(y \mid x) \cdot p_\theta(t \mid x)$.

2.1 $\mathrm{I}(X;T)$ — the Token Encoder $p_\theta(t \mid x)$

Under this distribution, $\mathrm{I}(X;T) \stackrel{\text{def}}{=} \mathbb{E}_x\left[\mathbb{E}_{t\sim p_\theta(t|x)}\left[\log \frac{p_\theta(t|x)}{p_\theta(t)}\right]\right]$. Making this term small yields a representation T that, on average, retains little information about X. The outer expectation is over the true distribution of sentences x; we use an empirical estimate, averaging over the unparsed sentences in a dependency treebank. To estimate the inner expectation, we could sample, drawing taggings t from $p_\theta(t \mid x)$.

We must also compute the quantities within the inner brackets. The $p_\theta(t \mid x)$ term is defined by our parametric form. The troublesome term is $p_\theta(t) = \mathbb{E}_{x'}[p_\theta(t \mid x')]$, since even estimating it from a treebank requires an inner loop over treebank sentences x'. To avoid this, variational IB replaces $p_\theta(t)$ with a simpler distribution to obtain an upper bound on $\mathrm{I}(X;T)$. See [Li and Eisner, 2019] for the form of this distribution and how we optimize its parameters to tighten the uppper bound, jointly with optimizing θ.

2.2 Two Token Encoder Architectures

We choose to define $p_\theta(t \mid x) = \prod_{i=1}^{n} p_\theta(t_i \mid x_i)$. That is, our stochastic encoder will compress each word x_i individually (although x_i is itself a representation that depends on context): see Figure 1. We make this choice not for computational reasons—our method would remain tractable even without this—but because our goal in this paper is to find the syntactic information in each individual ELMo token embedding (a goal we will further pursue in § 2.3 below).

To obtain continuous tags, define $p_\theta(t_i \mid x_i)$ such that $t_i \in \mathbb{R}^d$ is Gaussian-distributed with mean vector and diagonal covariance matrix computed from the ELMo word vector x_i via a feedforward neural network. Alternatively, to obtain discrete tags, define $p_\theta(t_i \mid x_i)$ such that $t_i \in \{1, \ldots, k\}$ follows a softmax distribution, where the k softmax parameters are similarly computed by a feedforward network. See [Li and Eisner, 2019] for architecture details.

2.3 $\mathrm{I}(T_i; X|\hat{X}_i)$ — the Type Encoder $s_\xi(t_i|\hat{x}_i)$

While the IB objective (1) asks each tag t_i to be informative about the parse Y, we were concerned that it might not be interpretable as a tag of word i specifically. Given ELMo or any other black-box conversion of a length-n sentence to a sequence of contextual vectors $x_1, \ldots, x_n$, it is possible that x_i contains not only information about word i but also information describing word $i + 1$, say, or the syntactic constructions in the vicinity of word i. Thus, while $p_\theta(t_i \mid x_i)$ might extract some information from x_i that is very useful for parsing, there is no guarantee that this information came from word i and not its neighbors. Although we *do* want tag t_i to consider context—e.g., to distinguish between noun and verb uses of word i—we want "most" of t_i's information to come from word i itself. Specifically, it should come from ELMo's level-0 embedding of word i, denoted by $\hat{x}_i$—a word *type* embedding that does *not* depend on context.

[1] Since T is a stochastic function of X with no access to Y, it obviously cannot convey more information about Y than the uncompressed input X does. As a result, Y is independent of T given X, as in the graphical model $T \to X \to Y$.

To penalize T_i for capturing "too much" contextual information, our modified objective (2) adds a penalty term $\gamma \cdot \mathrm{I}(T_i; X|\hat{X}_i)$, which measures the amount of information about T_i given by the sentence X as a whole, beyond what is given by $\hat{X}_i$: $\mathrm{I}(T_i; X \mid \hat{X}_i) \stackrel{\text{def}}{=} \mathbb{E}_x\left[\mathbb{E}_{t_i \sim p_\theta(t_i|x)}\left[\log \frac{p_\theta(t_i|x)}{p_\theta(t_i|\hat{x}_i)}\right]\right]$. Setting $\gamma > 0$ will reduce this contextual information.

We can derive an upper bound on $\mathrm{I}(T_i; X|\hat{X}_i)$ by approximating the conditional distribution $p_\theta(t_i \mid \hat{x}_i)$ with a variational distribution $s_\xi(t_i \mid \hat{x}_i)$. See details in [Li and Eisner, 2019].

Type Encoder Architectures Notice that $s_\xi(t_i \mid \hat{x}_i)$ may be regarded as a type encoder, with the same neural architecture as $p_\theta(t_i \mid x_i)$ except that p_θ takes a token vector as input whereas s_ξ takes a context-independent type vector. s_ξ is not used at test time, but only as part of our training objective.

2.4 $\mathrm{I}(Y;T)$ — the Decoder $q_\phi(y \mid t)$

Finally, $\mathrm{I}(Y;T) \stackrel{\text{def}}{=} \mathbb{E}_{y,t\sim p_\theta}\left[\log \frac{p_\theta(y|t)}{p(y)}\right]$. The $p(y)$ can be omitted during optimization as it does not depend on θ. Thus, making $\mathrm{I}(Y;T)$ large tries to obtain a high log-probability $p_\theta(y \mid t)$ for the true parse y when reconstructing it from t alone.

But how do we compute $p_\theta(y \mid t)$? This quantity effectively marginalizes over possible sentences x that *could have* explained t. To avoid this expensive computation, we obtain a lower bound of $\mathrm{I}(Y;T)$ by replacing $p_\theta(y \mid t)$ with a variational approximation $q_\phi(y \mid t)$. Here q_ϕ is a tractable conditional distribution, and may be regarded as a stochastic parser that runs on a compressed tag sequence t.

In short, when t is a stochastic compression of a treebank sentence x, we would like our variational parser *on average* to assign high log-probability $q_\phi(y \mid t)$ to its treebank parse y.

3 Training and Inference

With the variational approximations in §2, our final minimization objective is an upper bound on (2). Training involves computing two tractable KL divergence terms and a stochastic gradient step. See [Li and Eisner, 2019] for algorithmic and mathematical details.

To evaluate our trained model's ability to parse a sentence x from compressed tags, we obtain a parse as $\mathrm{argmax}_y\, q_\phi(y \mid t)$, where $t \sim p_\theta(\cdot \mid x)$ is a single sample. A better parser would instead estimate $\mathrm{argmax}_y\, \mathbb{E}_t\left[q_\phi(y \mid t)\right]$ where $\mathbb{E}_t$ averages over many samples t, but this is computationally hard.

4 Experimental Setup

Data Throughout §§5–6, we will examine our compressed tags on a subset of Universal Dependencies [Nivre et al., 2018], or UD, a collection of dependency treebanks. We experiment on 9 languages: Arabic, Hindi, English, French, Spanish, Portuguese, Russian, Italian, and Chinese — languages with different syntactic properties like word order. For each sentence, x is obtained by running the standard pre-trained ELMo [Gardner *et al.*, 2017; Che *et al.*, 2018] on the UD token sequence, and y is the labeled UD dependency parse *without* any part-of-speech (POS) tags.

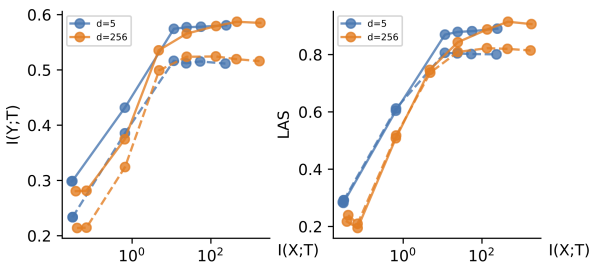


Figure 2: Compression-prediction tradeoff curves of VIB in our dependency parsing setting as we lower β from 10^1 to 10^{-6} on a log-scale. The dashed lines are for test data, and the solid lines for training data. The "dim" in the legends means the dimensionality of the continuous tag vector.

Optimization We optimize with Adam [Kingma and Ba, 2014], a variant of stochastic gradient descent. We train for 50 epochs with minibatches of size 20 and L_2 regularization. For each training sentence, we average over 5 i.i.d. samples of T to reduce the variance of the stochastic gradient. We experiment with different dimensionalities $d \in \{5, 32, 256, 512\}$ for the continuous tags, and different cardinalities $k \in \{32, 64, 128\}$ for the discrete tag set. We also tried different values $\beta, \gamma \in \{10^{-6}, 10^{-5}, \cdots, 10^{1}\}$ of the compression tradeoff parameter.

5 Scientific Evaluation

In this section, we study what information about words is retained by our automatically constructed tagging schemes. First, we show the relationship between $I(Y;T)$ and $I(X;T)$ on English as we reduce β to capture more information in our tags.[2] Second, across 9 languages, we study how our automatic tags correlate with gold part-of-speech tags (and in English, with other syntactic properties). See [Li and Eisner, 2019] for full tables and experimental details.

5.1 Tradeoff Curves

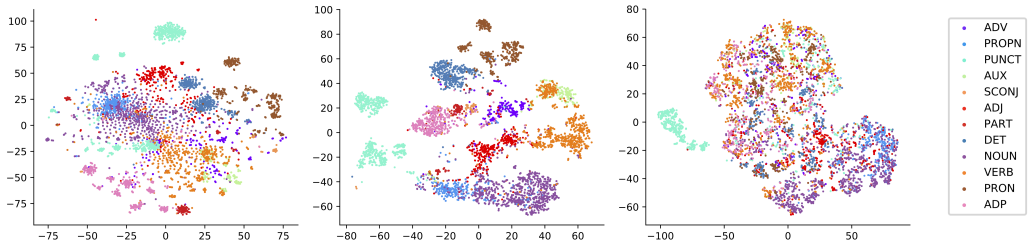
As we lower β to retain more information about X, both $\mathrm{I}(X;T)$ and $\mathrm{I}(Y;T)$ rise (Figure 2). There are diminishing returns: after some point, the additional information retained in T does not contribute much to predicting Y. Also noteworthy is that at each level of $\mathrm{I}(X,T)$, very low-dimensional tags ($d = 5$) perform on par with high-dimensional ones ($d = 256$). (Note that the high-dimensional stochastic tags will be noisier to keep the same $\mathrm{I}(X,T)$.) The low-dimensional tags allow far faster CPU parsing. This indicates that VIB can achieve strong practical task-specific compression.

5.2 Learned Tags vs. Gold POS Tags

We investigate how our automatic tag T_i correlates with the gold POS tag A_i provided by UD.

Continuous Version In Figure 3, the first figure shows the original uncompressed level-1 ELMo embeddings of the tokens in test data. In the two-dimensional visualization, the POS tags are vaguely clustered but the boundaries merge together and some tags are diffuse. The second figure is when

[2] We always set $\gamma = \beta$ to simplify the experimental design.



(a) ELMo, $I(X;T) = H(X) \approx 400.6$ (b) $I(X;T) \approx 24.3$ (c) $I(X;T) \approx 0.069$

Figure 3: t-SNE visualization of VIB model ($d = 256$) on the projected space of the continuous tags. Each marker in the figure represents a word token, colored by its gold POS tag. This series of figures (from left to right) shows a progression from no compression to moderate compression and to too-much compression.

Models	Arabic	Hindi	English	French	Spanish	Portuguese	Russian	Chinese	Italian
Iden	0.751	**0.870**	0.824	0.784	0.808	0.813	0.783	0.709	**0.863**
MLP	0.759	**0.871**	0.839	0.816	**0.835**	0.821	0.800	0.734	**0.867**
VIBc	**0.779**	**0.866**	**0.851**	**0.828**	**0.837**	**0.836**	**0.814**	**0.754**	**0.867**
POS	0.652	0.713	0.712	0.718	**0.739**	**0.743**	**0.662**	0.510	0.779
VIBd	**0.672**	**0.736**	**0.742**	**0.723**	**0.725**	0.710	**0.651**	**0.591**	**0.781**

Table 1: Parsing accuracy of 9 languages (LAS). Black rows use continuous tags; gray rows use discrete tags (which does worse). In each column, the best score for each color is boldfaced, along with all results of that color that are not significantly worse.

$\beta = 10^{-3}$ (moderate compression): our compressed embeddings show clear clusters that correspond well to gold POS tags. The third figure is when $\beta = 1$ (too much compression), when POS information is largely lost. An interesting observation is that the purple NOUN and blue PROPN distributions overlap in the middle distribution, meaning that it was unnecessary to distinguish common nouns from proper nouns for purposes of our parsing task[3].

Beyond Gold POS Tags, we study other syntactic distinctions: tense, number, and transitivity. Even with moderate compression, t_i achieves 0.87 classification accuracy in distinguishing between transitive and intransitive English verbs.

Discrete Version We also quantify how well our specialized discrete tags capture the traditional POS categories, by investigating $I(A;T)$. In effect, we are doing transfer learning, fixing our trained IB encoder (p_θ) and now using it to predict A instead of Y, but otherwise following § 2.4. We experiment on all 9 languages, taking T_i at the moderate compression level $\beta = 0.001$, $k = 64$. Experimental details and results are in [Li and Eisner, 2019]. Averaging over the 9 languages, the reconstruction retains 71% of POS information. We can conclude that the information encoded in the specialized tags correlates with the gold POS tags, but does not perfectly predict the POS.

6 Engineering Evaluation

As we noted in §1, learning how to compress ELMo's tags for a given task is a fast alternative to fine-tuning all the ELMo parameters. We find that indeed, training a compression method to keep only the relevant information does improve our generalization performance on the dependency parsing task.

Baselines. The continuous baselines include: **Iden**, which leaves the ELMo embeddings uncompressed; **MLP**, which uses a multi-layer perceptron ([Dozat and Manning, 2016]) to reduce the dimensionality in a task-specific and nonlinear way. The discrete baseline: **POS** uses the $k \leq 17$ gold POS tags from the UD dataset.

Runtime. Our VIB approach is quite fast: it trains on 10,000 sentences in 100 seconds, per epoch, on a single GPU.

Analysis. In the continuous case, the VIB representation outperforms all three baselines in 8 of 9 languages, and is not significantly worse in the 9th language (Hindi). In short, our VIB joint training generalizes better to test data. This is because the training objective (2) includes terms that focus on the parsing task and also regularize the representations. In the discrete case, the VIB representation outperforms gold POS tags (at the same level of granularity) in 6 of 9 languages, and of the other 3, it is not significantly worse in 2. This suggests that our learned discrete tag set could be an improved alternative to gold POS tags [Klein and Manning, 2003] when a discrete tag set is needed for speed.

7 Conclusion

In this paper, we have proposed two ways to syntactically compress ELMo word token embeddings, using variational information bottleneck. We automatically induce stochastic discrete tags that correlate with gold POS tags but are as good or better for parsing. We also induce stochastic continuous token embeddings (each is a Gaussian distribution over $\mathbb{R}^d$) that forget non-syntactic information captured by ELMo. These stochastic vectors yield improved parsing results, in a way that simpler dimensionality reduction methods do not. They also transfer to the problem of predicting gold POS tags, which were not used in training.

Acknowledgments

This work was supported by the National Science Foundation under Grant No. 1718846.

[3] Both can serve as arguments of verbs and prepositions. Both can be modified by determiners and adjectives.

References

[Alemi *et al.*, 2016] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. *Proceedings of the International Conference on Learning Representations (ICLR)*, abs/1612.00410, 2016.

[Anderson, 2003] T.W. Anderson. *An Introduction to Multivariate Statistical Analysis*. Wiley Series in Probability and Statistics. Wiley, 2003.

[Che *et al.*, 2018] Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[Dozat and Manning, 2016] Timothy Dozat and Christopher D. Manning. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734, 2016.

[Gardner *et al.*, 2017] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. AllenNLP: A Deep Semantic Natural Language Processing Platform. 2017.

[Goldberg, 2019] Yoav Goldberg. Assessing BERT's syntactic abilities. *CoRR*, abs/1901.05287, 2019.

[Kingma and Ba, 2014] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[Klein and Manning, 2003] D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, 2003.

[Li and Eisner, 2019] Xiang Lisa Li and Jason Eisner. Specializing word embeddings (for parsing) by information bottleneck. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*, pages 2744–2754, Hong Kong, November 2019. Best Paper Award.

[Nivre et al., 2018] Joakim Nivre et al. Universal dependencies 2.3, 2018. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

[Peters *et al.*, 2018a] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018.

[Peters *et al.*, 2018b] Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1509, 2018.

[Tishby and Zaslavsky, 2015] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5, 2015.

[Tishby *et al.*, 2000] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.