order similarity transformation of the input word vector space $\boldsymbol{X}$ (for which $n = 1$) can be obtained as $\boldsymbol{M}_n(\boldsymbol{X}) = \boldsymbol{M}_1(\boldsymbol{X}\boldsymbol{W}_{(n-1)/2})$, with $\boldsymbol{W}_\alpha = \boldsymbol{Q}\boldsymbol{\Gamma}^\alpha$, where $\boldsymbol{Q}$ and $\boldsymbol{\Gamma}$ are the matrices obtained via eigendecomposition of $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{Q}\boldsymbol{\Gamma}\boldsymbol{Q}^T$. $\boldsymbol{\Gamma}$ is a diagonal matrix containing eigenvalues of $\boldsymbol{X}^T\boldsymbol{X}$; $\boldsymbol{Q}$ is an orthogonal matrix with eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$ as columns. While the motivation for the UNCOVEC methods does originate from adjusting discrete similarity orders, note that α is in fact a continuous real-valued hyper-parameter which can be carefully tuned. For more technical details we refer the reader to the original work of Artetxe et al. (2018).

As mentioned, all post-processing methods can be seen as unsupervised retrofitting methods that, given an arbitrary input vector space $\boldsymbol{X}$, produce a perturbed/transformed output vector space $\boldsymbol{X}'$, but unlike common retrofitting methods (Faruqui et al. 2015; Mrkšić et al. 2017), the perturbation is completely unsupervised (i.e., self-contained) and does not inject any external (semantic similarity oriented) knowledge into the vector space. Note that different perturbations can also be stacked: e.g., we can apply UNCOVEC and then use ABTT on top the output UNCOVEC vectors. When using UNCOVEC and ABTT we always length-normalize and mean-center the data first (i.e., we apply the simple MC normalization). Finally, we tune the two hyper-parameters d_A (for ABTT) and α (UNCOVEC) on the English Multi-SimLex and use the same values on the datasets of all other languages; we report results with $dd_A = 3$ or $dd_A = 10$, and $\alpha = -0.3$.

Contextualized Word Embeddings. We also evaluate the capacity of unsupervised pretraining architectures based on language modeling objectives to reason over lexical semantic similarity. To the best of our knowledge, our article is the first study performing such analyses. State-of-the-art models such as BERT (Devlin et al. 2019), XLM (Conneau and Lample 2019), or ROBERTA (Liu et al. 2019b) are typically very deep neural networks based on the Transformer architecture (Vaswani et al. 2017). They receive subword-level tokens as inputs (such as WordPieces (Schuster and Nakajima 2012)) to tackle data sparsity. In output, they return contextualized embeddings, dynamic representations for words in context.

To represent words or multi-word expressions through a pretrained model, we follow prior work (Liu et al. 2019a) and compute an input item's representation by 1) feeding it to a pretrained model *in isolation*; then 2) averaging the H last hidden representations for each of the item's constituent subwords; and then finally 3) averaging the resulting subword representations to produce the final d-dimensional representation, where d is the embedding and hidden-layer dimensionality (e.g., $d = 768$ with BERT). We opt for this approach due to its proven viability and simplicity (Liu et al. 2019a), as it does not require any additional corpora to condition the induction of contextualized embeddings.[9] Other ways to extract the representations from pretrained models (Aldarmaki and Diab 2019; Wu et al. 2019; Cao, Kitaev, and Klein 2020) are beyond the scope of this work, and we will experiment with them in the future.

In other words, we treat each pretrained encoder ENC as a black-box function to encode a single word or a multi-word expression x in each language into a d-dimensional contextualized representation $\mathbf{x}_{\text{ENC}} \in \mathbb{R}^d = \text{ENC}(x)$ (e.g., $d = 768$ with BERT). As multilingual pretrained encoders, we experiment with the multilingual BERT model (M-BERT) (Devlin et al. 2019) and XLM (Conneau and Lample 2019). M-BERT is pretrained

9 We also tested another encoding method where we fed pairs instead of single words/concepts into the pretrained encoder. The rationale is that the other concept in the pair can be used as disambiguation signal. However, this method consistently led to sub-par performance across all experimental runs.