# Obtaining Better Static Word Embeddings Using Contextual Embedding Models

**Prakhar Gupta**
EPFL, Switzerland
prakhar.gupta@epfl.ch

**Martin Jaggi**
EPFL, Switzerland
martin.jaggi@epfl.ch

## Abstract

The advent of contextual word embeddings—representations of words which incorporate semantic and syntactic information from their context—has led to tremendous improvements on a wide variety of NLP tasks. However, recent contextual models have prohibitively high computational cost in many use-cases and are often hard to interpret. In this work, we demonstrate that our proposed distillation method, which is a simple extension of CBOW-based training, allows to significantly improve computational efficiency of NLP applications, while outperforming the quality of existing static embeddings trained from scratch as well as those distilled from previously proposed methods. As a side-effect, our approach also allows a fair comparison of both contextual and static embeddings via standard lexical evaluation tasks.

## 1 Introduction

Word embeddings—representations of words which reflect semantic and syntactic information carried by them are ubiquitous in Natural Language Processing. Static word representation models such as GLOVE (Pennington et al., 2014), CBOW, SKIPGRAM (Mikolov et al., 2013) and SENT2VEC (Pagliardini et al., 2018) obtain stand-alone representations which do not depend on their surrounding words or sentences (context). Contextual embedding models (Devlin et al., 2019; Peters et al., 2018; Liu et al., 2019; Radford et al., 2019; Schwenk and Douze, 2017) on the other hand, embed the contextual information as well into the word representations making them more expressive than static word representations in most use-cases.

While recent progress on contextual embeddings has been tremendously impactful, static embeddings still remain fundamentally important in many scenarios as well:

- Even when ignoring the training phase, the computational cost of using static word embeddings is typically tens of millions times lower than using standard contextual embedding models[1], which is particularly important for latency-critical applications and on low-resource devices, and in view of environmental costs of NLP models (Strubell et al., 2019).
- Many NLP tasks inherently rely on static word embeddings (Shoemark et al., 2019), for example for interpretability, or e.g. in research in bias detection and removal (Kaneko and Bollegala, 2019; Gonen and Goldberg, 2019; Manzini et al., 2019) and analyzing word vector spaces (Vulic et al., 2020) or other metrics which are non-contextual by choice.
- Static word embeddings can complement contextual word embeddings, for separating static from contextual semantics (Barsalou, 1982; Rubio-Fernández, 2008), or for improving joint embedding performance on downstream tasks (Alghanmi et al., 2020).

We also refer the reader to this article[2] illustrating several down-sides of using BERT-like models over static embedding models for non-specialist users. Indeed, we can see continued prevalence of static word embeddings in industry and research areas including but not limited to medicine (Zhang et al., 2019; Karadeniz and Özgür, 2019; Magna et al., 2020) and social sciences (Rheault and Cochrane, 2020; Gordon et al., 2020; Farrell et al., 2020; Lucy et al., 2020).

From a cognitive science point of view, Human language has been hypothesized to have both con-

[1] BERT base (Devlin et al., 2019) produces 768 dimensional word embeddings using 109M parameters, requiring 29B FLOPs per inference call (Clark et al., 2020).

[2] Do humanists need BERT? (https://tedunderwood.com/2019/07/15/)

*Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 5241–5253
August 1–6, 2021.

textual as well as context-independent properties (Barsalou, 1982; Rubio-Fernández, 2008) underlining the need for continued research in studying the expressiveness context-independent embeddings on the level of words.

Most existing word embedding models, whether static or contextual, follow Firth (1957)'s famous hypothesis - "You shall know a word by the company it keeps" , i.e., the meaning of a word arises from its context. During training existing static word embedding models, representations of contexts are generally approximated using averaging or sum of the constituent word embeddings, which disregards the relative word ordering as well as the interplay of information beyond simple pairs of words, thus losing most contextual information. Ad-hoc remedies attempt to capture longer contextual information per word using higher order n-grams like bigrams or trigrams, and have been shown to improve the performance of static word embedding models (Gupta et al., 2019; Zhao et al., 2017). However, these methods are not scalable to cover longer contexts.

In this work, we obtain improved static word embeddings by leveraging recent contextual embedding advances, namely by distilling existing contextual embeddings into static ones. Our proposed distillation procedure is inspired by existing CBOW-based static word embedding algorithms, but during training plugs in any existing contextual representation to serve as the context element of each word.

Our resulting embeddings outperform the current static embedding methods, as well as the current state-of-the-art static embedding distillation method on both unsupervised lexical similarity tasks as well as on downstream supervised tasks, by a significant margin. The resulting static embeddings remain compatible with the underlying contextual model used, and thus allow us to gauge the extent of lexical information carried by static vs contextual word embeddings. We release our code and trained embeddings publicly on GitHub[3].

## 2 Related Work

A few methods for distilling static embeddings have already been proposed. Ethayarajh (2019) propose using contextual embeddings of the same word in a large number of different contexts. They take the first principal component of the matrix formed by using these embeddings as rows and use it as a static embedding. However, this method is not scalable in terms of memory (the embedding matrix scaling with the number of contexts) and computational cost (PCA).

Bommasani et al. (2020) propose two different approaches to obtain static embeddings from contextual models.

1. **Decontextualized Static Embeddings** - The word $w$ alone without any context, after tokenization into constituents $w_1, \ldots, w_n$ is fed to the contextual embedding model denoted by $M$ and the resulting static embedding is given by $g(M(w_1), \ldots, M(w_n))$ where $g$ is a pooling operation. It is observed that these embeddings perform dismally on the standard static word embedding evaluation tasks.

2. **Aggregated Static Embeddings** - Since contextual embedding models are not trained on a single word (without any context) as input, an alternative approach is to obtain the contextual embedding of the word $w$ in different contexts and then pool(max, min or average) the embeddings obtained from these different contexts. They observe that average pooling leads to the best performance. We refer to this method (with average pooling) as ASE throughout the rest of the paper. As we see in our experiments, the performance of ASE embeddings saturates quickly with increasing size of the raw text corpus and is therefore not scalable.

Other related work includes distillation of contextual word embeddings to obtain sentence embeddings (Reimers and Gurevych, 2019). We also refer the reader to Mickus et al. (2020) for a discussion on the semantic properties of contextual models (primarily BERT) as well as Rogers et al. (2020), a survey on different works exploring the inner workings of BERT including its semantic properties.

## 3 Proposed Method

To distill existing contextual word representation models into static word embeddings, we augment a CBOW-inspired static word-embedding method as our anchor method to accommodate additional contextual information of the (contextual) teacher model. SENT2VEC (Pagliardini et al., 2018) is a

[3] https://github.com/epfml/X2Static

modification of the CBOW static word-embedding method which instead of a fixed-size context window uses the entire sentence to predict the masked word. It also has the ability to learn n-gram representations along with unigram representations, allowing to better disentangle local contextual information from the static unigram embeddings. SENT2VEC, originally meant to obtain sentence embeddings and later repurposed to obtain word representations (Gupta et al., 2019) was shown to outperform competing methods including GLOVE (Pennington et al., 2014), CBOW, SKIPGRAM (Mikolov et al., 2013) and FASTTEXT (Bojanowski et al., 2016) on word similarity evaluations. For a raw text corpus $\mathcal{C}$ (collection of sentences), the training objective is given by

$$\min_{\boldsymbol{U},\boldsymbol{V}} \sum_{S\in\mathcal{C}} \sum_{w_t\in S} f(\boldsymbol{u}_{w_t}, E_{\textsf{ctx}}(S, w_t)) \qquad (1)$$

where $f(\boldsymbol{u}, \boldsymbol{v}) := \ell(\boldsymbol{u}^\top \boldsymbol{v}) + \sum_{w'\in N} \ell(-\boldsymbol{u}_{w'}^\top \boldsymbol{v})$. Here, $w_t$ is the masked target word, $\boldsymbol{U}$ and $\boldsymbol{V}$ are the target word embedding and the source n-gram matrices respectively, $N$ is the set of negative target samples and, $\ell : x \mapsto \log(1 + e^{-x})$ is the logistic loss function.

For SENT2VEC, the context encoder $E_{\textsf{ctx}}$ used in optimizing (1) is simply given by the (static, non-contextual) sum of all vectors in the sentence without the target word,

$$E_{\textsf{ctx}}(S, w_t) := \frac{1}{|R(S\backslash\{w_t\})|} \sum_{w\in R(S\backslash\{w_t\})} \boldsymbol{v}_w , \qquad (2)$$

where $R(S)$ denotes the optional expansion of the sentence $S$ from words to short n-grams, i.e., the context sentence embedding is obtained by averaging the embeddings of word n-grams in the sentence $S$.

We will now generalize the objective (1) by allowing the use of arbitrary modern contextual representations $E_{\textsf{ctx}}$ instead of the static context representation as in (2). This key element will allow us to translate quality gains from improved contextual representations also to better static word embedding in the resulting matrix $\boldsymbol{U}$. We propose two different approaches of doing so, which differ in the granularity of context used for obtaining the contextual embeddings.

### 3.1 Approach 1 - Sentences as context

Using contextual representations of all words in the sentence $S$ (or the sentence $S \setminus \{w_t\}$ without the target word) allows for a more refined representation of the context, and to take in account the word order as well as the interplay of information among the words of the context.

More formally, let $M(S, w)$ denote the output of a contextual embedding-encoder, e.g. BERT, corresponding to the word $w$ when a piece of text $S$ containing $w$ is fed to it as input. We let $E_{\textsf{ctx}}(S, w)$ to be the average of all contextual embeddings of words $w$ returned by the encoder,

$$E_{\textsf{ctx}}(S, w_t) := \frac{1}{|S|} \sum_{w\in S} M(S, w) \qquad (3)$$

This allows for a more refined representation of the context as the previous representation did not take in account neither the word order nor the interplay of information among the words of the context. Certainly, using $S_{m_{w_t}}$ ($S$ with $w_t$ masked) and $w$ would make for an even better word-context pair but that would amount to one contextual embedding-encoder inference per word instead of one inference per sentence as is the case in (3) leading to a drastic drop in computational efficiency.

### 3.2 Approach 2 - Paragraphs as context

Since contextual models are trained on large pieces of texts (generally $\geq 512$ tokens), we instead use paragraphs instead of sentences to obtain the contextual representations. However, in order to predict target words, we use the contextual embeddings within the sentence only. Consequently, for this approach, we have

$$E_{\textsf{ctx}}(S, w_t) := \frac{1}{|S|} \sum_{w\in S} M(P_S, w), \qquad (4)$$

where $P_S$ is the paragraph containing sentence $S$.

In the transfer phase, this approach is more computationally efficient than the previous approach, as we have to invoke the contextual embedding model $M$ only once for each paragraph as opposed to once for every constituent sentence. Moreover, it encapsulates the related semantic information in paragraphs in the contextual word embeddings.

We call our models X2STATIC$_{sent}$ in the sentence case (3), and X2STATIC$_{para}$ in the paragraph case (4) respectively where X denotes the parent model.

## 4 Experiments and Discussion

### 4.1 Corpus Preprocessing and Training

We use the same English Wikipedia Dump as Pagliardini et al. (2018); Gupta et al. (2019) to

| Epoch(s) trained | Max Vocab. Size | Number of Negatives Sampled | Target Word Subsampling hyperparameter | Minimum Word Count | Initial Learning Rate | Batch Size |
|---|---|---|---|---|---|---|
| 1 | 750000 | 10 | 5e-6 | 10 | 0.001 | 128 |

Table 1: **Training hyperparameters used for training X2Static models**

| Model | Epoch(s) trained | Max Vocab. Size | Number of Negatives Sampled | Target Word Subsampling hyperparameter | Min. Word Count | Initial Learning Rate | Word N-grams | Character N-grams | Window Size |
|---|---|---|---|---|---|---|---|---|---|
| Sent2vec | {5,**10**,15} | 750000 | {5,8,**10**} | {1e-4, 5e-6, 1e-5, **5e-6**} | 10 | 0.2 | {1,2,**3**} | N.A. | N.A. |
| Skipgram | {5,**10**,15} | N.A. | {**5**,8,10} | {1e-4, 5e-6, **1e-5**, 5e-6} | 10 | 0.05 | N.A. | {N.A.,**3-6**} | {2,5,**10**} |
| CBOW | {5,10,**15**} | N.A. | {**5**,8,10} | {1e-4, 5e-6, **1e-5**, 5e-6} | 10 | 0.05 | N.A. | {N.A.,**3-6**} | {2,5,**10**} |

Table 2: **Hyperparameter search space description for the training of Sent2vec, Skipgram and CBOW models**: Best hyperparameters for the chosen model in our experiments are shown in bold. N.A. indicates not applicable.

generate distilled X2Static representations. as our corpus for training static word embedding baselines as well as for distilling static word embeddings from pre-trained contextual embedding models. We remove all paragraphs with less than 3 sentences or 140 characters, lowercase the characters and tokenize the corpus using the Stanford NLP library (Manning et al., 2014) resulting in a corpus of approximately 54 Million sentences and 1.28 Billion words. We then use the Transformers library[4] (Wolf et al., 2020) to generate representations from existing transformer models. Our X2Static representations are distilled from the last representation layers of these models.

We use the same hyperparameter set for training all X2Static models, i.e., no hyperparameter tuning is done at all. We use 12-layer as well as 24-layer pre-trained models using BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and GPT2 (Radford et al., 2019) architectures as the teacher model to obtain X2Static word embeddings. All the X2Static models use the same set of training parameters except the parent model. Training hyperparameters are provided in Table 1. The distillation/training process employs the lazy version of the Adam optimizer (Kingma and Ba, 2015a), suitable for sparse tensors. We use a subsampling parameter similar to FastText (Bojanowski et al., 2016) in order to subsample frequent target words during training. Each X2Static model was trained using a single V100 32 GB GPU. Obtaining X2Static embeddings from 12-layer contextual embedding models took 15-18 hours while it took 35-38 hours to obtain them from their 24-layer counterparts.

To ensure a fair comparison, we also evaluate Sent2vec, CBOW and Skipgram models that were trained on the same corpus. We do an extensive hyperparameter tuning for these models and choose the one which shows best average performance on the 5 word similarity datasets used in Subsection 4.2. These hyperparameter sets can be accessed in Table 2 where the chosen hyperparameters are shown in bold. We set the number of dimensions to be 768 to ensure parity between them and the X2Static models compared. We used the Sent2vec library[5] for training Sent2vec and the FastText library[6] for training CBOW and Skipgram models. We also evaluate some pre-trained 300 dimensional GloVe (Pennington et al., 2014) and FastText (Bojanowski et al., 2016) models in Table 3. The GloVe model was trained on Common-Crawl corpus of 840 Billion tokens (approximately 650 times larger than our corpus) while the FastText vectors were trained on a corpus of 16 Billion tokens (approximately 12 times larger than our corpus)). We also extract ASE embeddings from each layer using the same Wikipedia corpus.

We perform two different sets of evaluations. The first set corresponds to unsupervised word similarity evaluations to gauge the quality of the obtained word embeddings. However, we recognize that there are concerns regarding word-similarity

[4] https://huggingface.co/transformers/

[5] https://github.com/epfml/sent2vec

[6] https://github.com/facebookresearch/fastText/

evaluation tasks (Faruqui et al., 2016) as they are shown to exhibit significant difference in performance when subjected to hyperparameter tuning (Levy et al., 2015). To address these limitations in the evaluation, we also evaluate the X2STATIC embeddings on a standard set of downstream supervised evaluation tasks used in Pagliardini et al. (2018).

## 4.2 Unsupervised word similarity evaluation

To assess the quality of the lexical information contained in the obtained word representations, we use the 4 word-similarity datasets used by (Bommasani et al., 2020), namely *WordSim353* (353 word-pairs) (Agirre et al., 2009) dataset; *SimLex-999* (999 word-pairs) (Hill et al., 2014) dataset; *RG-65* (65 pairs) (Joubarne and Inkpen, 2011); and *SimVerb-3500* (3500 pairs) (Gerz et al., 2016) dataset as well as the *Rare Words RW-2034* (2034 pairs) (Luong et al., 2013) dataset. To calculate the similarity between two words, we use the cosine similarity between their word embeddings. These similarity scores are compared to the human ratings using Spearman's $\rho$ (Spearman, 1904) correlation scores. We use the tool[7] provided by Bommasani et al. (2020) to report these results on ASE embeddings. It takes around 3 days to obtain ASE representations of the 2005 words in these word-similarity datasets for 12-layer models and around 5 days to obtain them for their 24-layer counterparts on the same machine used for learning X2STATIC representations. All other embeddings are evaluated using the MUSE repository evaluation tool[8] (Lample et al., 2018).

We perform two sets of experiments concerning the unsupervised evaluation tasks. The first set is the comparison of our X2STATIC models with competing models. For ASE, we report two sets of results, one which per task reports the best result amongst all the layers and other, which reports the results obtained on the best performing layer on average.

We report our observations in Table 3. We provide additional results for larger models in Appendix B. We observe that X2STATIC embeddings outperform competing models on most of the tasks. Moreover, the extent of improvement on SimLex-999 and SimVerb-3500 tasks compared to the previous models strongly highlights the advantage of using improved context representations for training static word representations.

Second, we study the performance of the best ASE embedding layer with respect to the size of corpus used. Bommasani et al. (2020) report their results on a corpus size of only up to $N = 100,000$ sentences. In order to measure the full potential of the ASE method, we obtain different sets of ASE embeddings as well as X2STATIC$_{para}$ embeddings from small chunks of the corpus to the full wikipedia corpus itself and compare their performance on SimLex-999 and RW-2034 datasets. We choose SimLex-999 as it captures true similarity instead of relatedness or association (Hill et al., 2014) and RW-2034 to gauge the robustness of the embedding model on rare words. We report our observations in Figure 1. We observe that the performance of the ASE embeddings tends to saturate with the increase in the corpus size while X2STATIC$_{para}$ embeddings are either significantly outperforming the ASE embeddings or still show a significantly greater positive growth rate in performance w.r.t. the corpus size. Thus, the experimental evidence suggests that on larger texts, X2STATIC embeddings will have an even better performance and hence, X2STATIC is a better alternative than ASE embeddings from any of the layers of the contextual embedding model, and obtains improved static word embeddings from contextual embedding models.

## 4.3 Downstream supervised evaluation

We evaluate the obtained word embeddings on various sentence-level supervised classification tasks. Six different downstream supervised evaluation tasks namely classification of movie review sentiment(MR) (Pang and Lee, 2005), product reviews(CR) (Hu and Liu, 2004), subjectivity classification(SUBJ) (Pang and Lee, 2004), opinion polarity (MPQA) (Wiebe et al., 2005), question type classification (TREC) (Voorhees, 2002) and fine-grained sentiment analysis (SST-5) (Socher et al., 2013) are employed to gauge the performance of the obtained word embeddings.

We use a standard CNN based architecture on the top of our embeddings to train our classifier. We use 100 convolutional filters with a kernel size of 3 followed by a ReLU activation function. A global max-pooling layer follows the convolution layer. Before feeding the max-pooled output to a

[7] https://github.com/rishibommasani/Contextual2Static

[8] https://github.com/facebookresearch/MUSE

| Model \ Distilled Model | Parent Model \ Other details | Dim. | RG-65 | WS-353 | SL-999 | SV-3500 | RW-2034 | Average |
|---|---|---|---|---|---|---|---|---|
| *Existing pre-trained models* | Size of the training corpus relative to ours | | | | | | | |
| FastText | 12x | 300 | 0.7669 | 0.596 | 0.416 | 0.3274 | 0.5226 | 0.5276 |
| GloVe | 650x | 300 | 0.6442 | 0.5791 | 0.3764 | 0.2625 | 0.4607 | 0.4646 |
| *Models trained by us* | | | | | | | | |
| Skipgram | N.A. | 768 | 0.8259 | 0.7141 | 0.4064 | 0.2722 | 0.4849 | 0.5407 |
| CBOW | N.A. | 768 | 0.8348 | 0.4999 | 0.4097 | 0.2626 | 0.4043 | 0.4823 |
| Sent2vec | N.A. | 768 | 0.7811 | 0.7407 | 0.5034 | 0.3297 | 0.4248 | 0.55594 |
| *Models distilled by us* | Parent Model | | | | | | | |
| ASE - best layer per task | BERT-12 | 768 | 0.7449(1) | 0.7012(1) | 0.5216(4) | 0.4151(5) | 0.4577(5) | 0.5429(3) |
| ASE - best overall layer | BERT-12 | 768 | 0.6948(3) | 0.6768(3) | 0.5195(3) | 0.3889(3) | 0.4343(3) | 0.5429(3) |
| BERT2Static$_{sent}$ | BERT-12 | 768 | 0.7421 | 0.7297 | **0.5461** | **0.4437** | **0.5469** | **0.6017** |
| BERT2Static$_{para}$ | BERT-12 | 768 | 0.7555 | **0.7598** | **0.5384** | **0.4317** | **0.5299** | **0.6031** |
| ASE - best layer per task | RoBERTa-12 | 768 | 0.673(0) | 0.7023(0) | 0.554(5) | 0.4602(4) | 0.5075(3) | 0.5600(0) |
| ASE - best overall layer | RoBERTa-12 | 768 | 0.673(0) | 0.7023(0) | 0.5167(0) | 0.4424(0) | 0.4657(0) | 0.5600(0) |
| RoBERTa2Static$_{sent}$ | RoBERTa-12 | 768 | 0.7999 | **0.7452** | 0.5507 | **0.4658** | **0.5496** | **0.6222** |
| RoBERTa2Static$_{para}$ | RoBERTa-12 | 768 | 0.8057 | **0.7638** | **0.5544** | **0.4717** | **0.5501** | **0.6291** |
| ASE - best layer per task | GPT2-12 | 768 | 0.7013(1) | 0.6879(0) | 0.4972(2) | 0.3905(2) | 0.4556(2) | 0.5365(2) |
| ASE - best overall layer | GPT2-12 | 768 | 0.6833(2) | 0.6560(2) | 0.4972(2) | 0.3905(2) | 0.4556(2) | 0.5365(2) |
| GPT$_2$2Static$_{sent}$ | GPT2-12 | 768 | 0.7484 | 0.7151 | **0.5397** | **0.4676** | **0.5760** | **0.6094** |
| GPT$_2$2Static$_{para}$ | GPT2-12 | 768 | 0.7881 | 0.7267 | **0.5417** | **0.4733** | **0.5668** | **0.6193** |

Table 3: **Comparison of the performance of different embedding methods on word similarity tasks.** Models are compared using Spearman correlation for word similarity tasks. All X2Static method performances which improve over all ASE methods on their parent model as well as all static models are shown in bold. Best performance in each task is underlined. For all ASE methods, the number in parentheses for each dataset indicates which layer was used for obtaining the static embeddings.

classifier, it is passed through a dropout layer with dropout probability of 0.5 to prevent overfitting. We use Adam (Kingma and Ba, 2015b) to train our classifier. To put the performance of these static models into a broader perspective, we also fine-tune linear classifiers on the top of their parent models as well as sentence-transformers (Reimers and Gurevych, 2019) obtained from RoBERTa-12 and BERT-12. For the sentence-transformer models, we use the sentence-transformer models obtained by fine-tuning their parent models on the Natural Language Inference(NLI) task using the combination of Stanford NLI (Bowman et al., 2015) and the Multi-Genre NLI (Williams et al., 2018) datasets. The models are refered to as sBERT-base-NLI and sRoBERTa-base-NLI in the rest of the paper.

The hyperparameter search space for the fine-tuning process involves the number of epochs (8-16) and the learning rates[1e-4,3e-4,1e-3]. Wherever train, validation, and test split is not given, we use 60% of the data as the training data, 20% of the data as validation data and the rest as the test data. After obtaining the best hyperparameters, we train on the train and validation data together with these hyperparameters and predict the results on the test set. For the linear classifiers on the top of parent models, we set the number of epochs and learning rate search space for parent model + linear classifier combination to be [3,4,5,6] and [2e-5,5e-5] respectively. The learning rates in the learning rate search space are lower than those for static embeddings as the contextual embeddings are also fine-tuned and follow the recommendation of Devlin et al. (2019). For the sentence-transformer models, we only train the linear classifier and set the number of epochs and learning rate search space to be [3,4,5,6] and [1e-4,3e-4,1e-3] respectively. We use cross-entropy

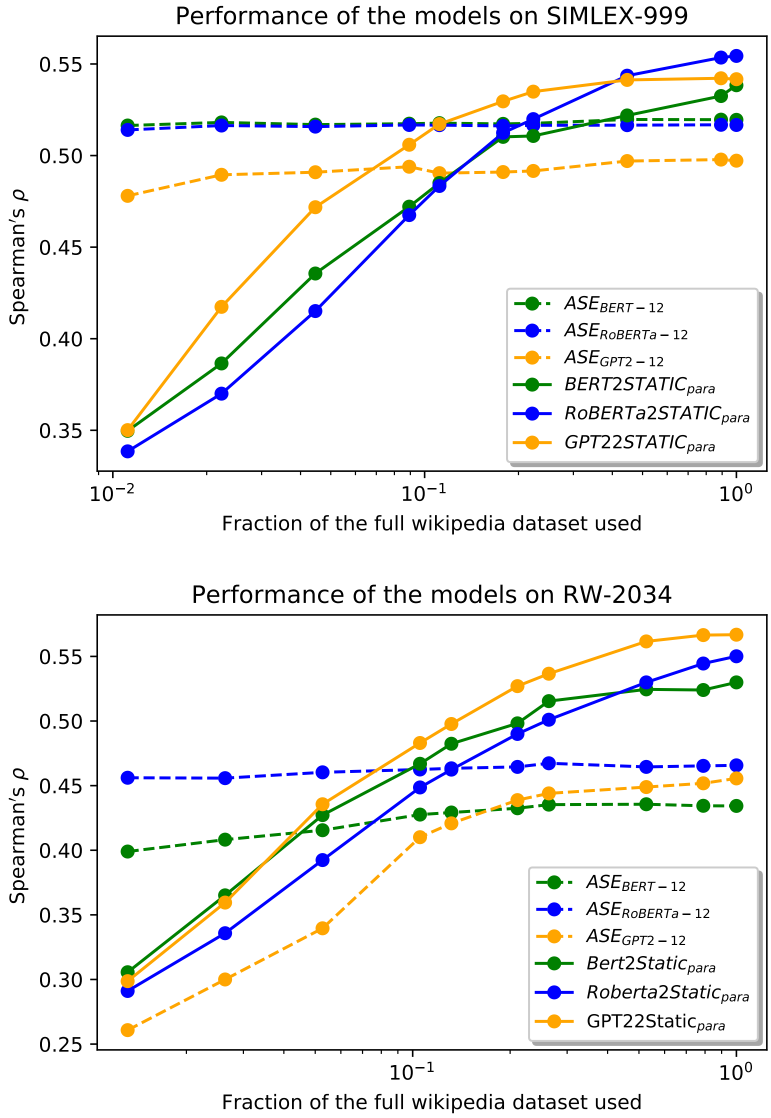

Figure 1: **Effect of corpus size** on the word-embedding quality for ASE best task independent layer and X2STATIC$_{para}$ : In the legend, parent model is indicated in subscript.

loss for training all the models. We use Macro-F1 score and Accuracy to gauge the quality of our predictions. We compare X2STATIC models with all other static models trained from scratch on the same corpus as well as the GLOVE and FASTTEXT models used in the previous section. We also use existing GLOVE embeddings trained on tweets(27 billion tokens - 20 times larger than our corpus) (Pennington et al., 2014) to make the comparison even more extensive. We report our observations in Table 4. For ASE embeddings, we take the layer with best average macro-F1 performance.

We observe that when measuring the overall performance, with the exception of ROBERTA2STATIC$_{sent}$ which has similar average F-1 score to ASE owing to its dismal performance on the CR task, all X2STATIC embeddings outperform their competitors by a significant margin. Even though the GLOVE and FASTTEXT embeddings were trained on corpora of one to two magnitudes larger and have a larger vocabulary, their performance lags behind that of the X2STATIC embeddings. To ensure statistical soundness, we measure mean and standard deviation of the performance on 6 runs of X2STATIC$_{para}$ model training followed by downstream evaluation along with 6 runs of ASE embedding downstream evaluation with different random seeds in Table 5 in the Appendix. We see that X2STATIC$_{para}$ embeddings outperform ASE

| Embeddings \Task | Dim | CR<br>F1 / Acc. | MR<br>F1 / Acc. | MPQA<br>F1 / Acc. | SUBJ<br>F1 / Acc. | TREC<br>F1 / Acc. | SST-5<br>F1 / Acc. | Average<br>F1 / Acc. |
|---|---|---|---|---|---|---|---|---|
| *Existing pre-trained models* | | | | | | | | |
| GLOVE | 300 | 81.6/83.2 | 78.2/78.2 | 85.1/87.6 | 90.9/90.9 | 45.4/86.2 | 15.5/43.2 | 66.1/78.1 |
| GLOVE (Twitter) | 200 | 79.0/80.9 | 74.1/74.2 | 82.1/85.0 | 89.6/89.7 | 49.1/87.8 | 13.1/37.5 | 64.5/75.9 |
| FASTTEXT | 300 | 80.3/81.9 | 78.3/78.4 | 86.5/88.1 | 90.9/90.9 | 45.3/85.9 | 13.9/43.9 | 66.2/78.2 |
| *Models trained by us* | | | | | | | | |
| SKIPGRAM | 768 | 78.4/80.9 | 75.2/75.2 | 83.1/85.8 | 91.5/91.5 | 50.2/88.6 | 13.9/39.0 | 65.4/76.8 |
| CBOW | 768 | 75.9/78.5 | 72.6/72.7 | 83.3/86.0 | 85.5/85.5 | 43.2/85.7 | 13.4/38.9 | 62.0/74.6 |
| SENT2VEC | 768 | 79.8/81.2 | 74.1/74.1 | 81.0/84.5 | 89.4/89.4 | 42.9/84.1 | 13.2/38.6 | 63.4/75.3 |
| *Models distilled by us* | | | | | | | | |
| ASE - BERT-12 (5) | 768 | 81.5/83.0 | 78.5/78.5 | 86.0/86.0 | 91.0/91.0 | 48.3/87.6 | 15.0/42.1 | 66.7/78.0 |
| BERT2STATIC$_{sent}$ | 768 | 80.1/82.0 | **78.9/78.9** | **87.4/89.1** | **91.8/91.8** | **50.6/88.7** | **16.1**/43.7 | **67.5/79.0** |
| BERT2STATIC$_{para}$ | 768 | 81.1/**83.6** | **80.8/80.8** | **87.3/89.3** | **91.6/91.6** | **51.8/89.2** | **16.1/44.9** | **68.1/79.9** |
| ASE - ROBERTA-12 (2) | 768 | 78.4/81.2 | 78.3/78.3 | 86.4/88.5 | 89.5/89.5 | 52.0/89.1 | 15.2/43.0 | 66.6/78.3 |
| ROBERTA2STATIC$_{sent}$ | 768 | 76.5/79.6 | **80.2/80.2** | 85.6/88.0 | **92.2/92.2** | 49.7/**89.1** | **15.7**/43.8 | **66.7/78.8** |
| ROBERTA2STATIC$_{para}$ | 768 | 80.9/82.3 | **80.0/80.1** | **87.3/89.4** | **92.4/92.4** | 49.3/**88.8** | **16.3**/43.4 | **67.7/79.4** |
| ASE - GPT2-12 (4) | 768 | 81.0/82.1 | 80.1/80.1 | 84.8/86.2 | 91.2/91.2 | 51.0/88.8 | 15.5/42.0 | 67.3/78.4 |
| GPT$_2$2STATIC$_{sent}$ | 768 | 81.5/**83.5** | **79.5/79.5** | **86.5/88.5** | **91.8/91.8** | **51.8/89.2** | **16.2/43.8** | **67.9/79.4** |
| GPT$_2$2STATIC$_{para}$ | 768 | 81.0/82.6 | **79.7/79.7** | **86.9/88.8** | **92.1/92.1** | **53.0/89.1** | **16.2/44.1** | **68.1/79.4** |
| *Parent contextual models and derivatives* | | | | | | | | |
| BERT-12 | 768 | 89.6/90.6 | 87.4/87.4 | 89.4/90.8 | 96.7/96.7 | 77.6/94.7 | 30.7/54.0 | 78.6/85.7 |
| sBERT-BASE-NLI | 768 | 87.4/88.7 | 83.3/83.3 | 86.8/88.2 | 93.6/93.6 | 41.6/72.2 | 25.3/48.2 | 69.7/79.1 |
| ROBERTA-12 | 768 | 90.0/90.8 | 90.1/90.1 | 89.1/90.6 | 96.3/96.3 | 95.1/99.2 | 34.0/57.6 | 82.4/87.4 |
| sROBERTA-BASE-NLI | 768 | 87.6/88.6 | 86.3/86.3 | 86.8/88.8 | 94.6/94.6 | 52.4/80.6 | 23.7/53.5 | 72.7/82.1 |
| GPT2-12 | 768 | 88.5/89.5 | 87.1/87.1 | 87.3/89.1 | 96.1/96.1 | 76.8/94.3 | 30.8/54.5 | 77.8/85.1 |

Table 4: **Comparison of the performance of different static embeddings on downstream tasks.** All X2STATIC method performances which improve or are at par over all other static embedding methods and the best ASE layer on their parent model are shown in bold. Best static embedding performance for each task is underlined. For each ASE method, the number in brackets indicates the layer with best average performance. We use macro-F1 scores and accuracy as the metrics to gauge the performance of models on these downstream tasks. **Note**: Contextual embeddings for BERT-12, ROBERTA-12 and GPT2-12 in the SOTA section are also fine-tuned while sBERT-BASE-NLI and sROBERTA-BASE-NLI are not.

by a significant margin.

For both word similarity evaluations and downstream supervised tasks, we observe that X2STATIC$_{para}$ embeddings perform slightly better than X2STATIC$_{sent}$ embeddings. However, since no hyperparameter tuning was performed on the distillation of X2STATIC embeddings, it is hard to discern which X2STATIC variant shows better performance. Moreover, owing to the same fact concerning hyperparameter tuning, we expect to see even larger improvements with proper hyperparameter tuning as well as training on larger data.

## 5 Conclusion and Future Work

This work proposes to augment earlier WORD2VEC-based methods by leveraging recent more expressive deep contextual embedding models to extract static word embeddings. The resulting distilled static embeddings, on an average, outperform their competitors on both unsupervised as well downstream supervised evaluations and thus can be used to replace compute-heavy contextual embedding models (or existing static embedding models) at inference time in many compute-resource-limited applications. The resulting embeddings can also be used as a task-agnostic tool to measure the lexical information conveyed by contextual embedding models and allow a fair comparison with their static analogues.

Further work can explore extending this distillation framework into cross-lingual domains (Schwenk and Douze, 2017; Lample and Conneau, 2019) as well as using better pooling methods instead of simple averaging for obtaining the context representation, or joint fine-tuning to obtain even stronger static word embeddings. Another promising avenue is the use of a similar approach to learn sense embeddings from contextual embedding models. We would also like to investigate the performance of these embeddings when distilled on a larger corpus along with more extensive hyperparameter tuning. Last but not the least, we would like to release X2Static models for different languages for further public use.

## References

Eneko Agirre, Enrique Alfonseca, Keith B. Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *HLT-NAACL*.

Israa Alghanmi, Luis Espinosa Anke, and Steven Schockaert. 2020. Combining BERT with Static Word Embeddings for Categorizing Social Media. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 28–33.

L. Barsalou. 1982. Context-independent and context-dependent information in concepts. *Memory & Cognition*, 10:82–93.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Rishi Bommasani, Kelly Davis, and Claire Cardie. 2020. Interpreting pretrained contextualized representations via reductions to static embeddings. In *ACL*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *EMNLP-IJCNLP - Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 55–65. ACL.

T. Farrell, Óscar Araque, Miriam Fernández, and H. Alani. 2020. On the use of jargon and word embeddings to explore subculture within the reddits manosphere. *12th ACM Conference on Web Science*.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. In *RepEval@ACL*.

J. R. Firth. 1957. A synopsis of linguistic theory, 1930-1955.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simverb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2173–2182.

Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. In *NAACL-HLT*.

Joshua Gordon, Marzieh Babaeianjelodar, and Jeanna Matthews. 2020. Studying political bias via word embeddings. In *WWW '20 - Companion Proceedings of the Web Conference 2020*, page 760764.

Prakhar Gupta, Matteo Pagliardini, and Martin Jaggi. 2019. Better word embeddings by disentangling contextual n-gram information. In *NAACL-HLT*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41:665–695.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Colette Joubarne and Diana Inkpen. 2011. Comparison of semantic similarity for different languages using the google n-gram corpus and second-order co-occurrence measures. In *Canadian Conference on AI*.

Masahiro Kaneko and Danushka Bollegala. 2019. Gender-preserving debiasing for pre-trained word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1641–1650, Florence, Italy. ACL.

Ilknur Karadeniz and Arzucan Özgür. 2019. Linking entities through an ontology using word embeddings and syntactic re-ranking. *BMC Bioinformatics*, 20.

Diederik P Kingma and Jimmy Ba. 2015a. Adam: A method for stochastic optimization. In *ICLR*.

Diederik P. Kingma and Jimmy Ba. 2015b. Adam: A method for stochastic optimization. In *ICLR - International Conference on Learning Representations*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. In *NeurIPS 2019 - Advances in Neural Information Processing Systems*.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations*.

Omer Levy, Y. Goldberg, and I. Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Li Lucy, Dorottya Demszky, Patricia Bromley, and Dan Jurafsky. 2020. Content analysis of textbooks via natural language processing: Findings on gender, race, and ethnicity in texas u.s. history textbooks. *AERA Open*, 6.

Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*.

Andrés Alejandro Ramos Magna, Héctor Allende-Cid, Carla Taramasco, C. Becerra, and R. Figueroa. 2020. Application of machine learning and word embeddings in the classification of cancer diagnosis using patient anamnesis. *IEEE Access*, 8:106198–106213.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL*.

Thomas Manzini, Lim Yao Chong, Alan W. Black, and Yulia Tsvetkov. 2019. Black is to criminal as caucasian is to police: Towards detecting, evaluating and removing multiclass bias in word embeddings. In *NAACL 2019*.

Timothee Mickus, Denis Paperno, Mathieu Constant, and Kees van Deemter. 2020. What do you mean, BERT? Assessing BERT as a Distributional Semantics Model. *Proceedings of the Society for Computation in Linguistics*, 3(1):350–361.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR - International Conference on Learning Representations*.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *NAACL-HLT*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. ACL.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *EMNLP-IJCNLP - Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3982–3992, Hong Kong, China. ACL.

L. Rheault and C. Cochrane. 2020. Word embeddings for the analysis of ideological placement in parliamentary corpora. *Political Analysis*, 28:112–133.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Paula Rubio-Fernández. 2008. Concept narrowing: The role of context-independent information. *J. Semant.*, 25:381–409.

Holger Schwenk and Matthijs Douze. 2017. Learning joint multilingual sentence representations with neural machine translation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 157–167, Vancouver, Canada. ACL.

Philippa Shoemark, Farhana Ferdousi Liza, Dong Nguyen, Scott Hale, and Barbara McGillivray. 2019. Room to Glo: A systematic comparison of semantic change detection approaches with word embeddings. In *EMNLP-IJCNLP - Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 66–76, Hong Kong, China. ACL.

R. Socher, Alex Perelygin, J. Wu, Jason Chuang, Christopher D. Manning, A. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Charles Spearman. 1904. The proof and measurement of association between two things. *The American journal of psychology*, 15(1):72–101.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. In *ACL*.

Ellen M Voorhees. 2002. Overview of the TREC 2001 question answering track. In *NIST special publication*, pages 42–51.

Ivan Vulic, Sebastian Ruder, and Anders Søgaard. 2020. Are all good word vector spaces isomorphic? In *EMNLP*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2):165–210.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP - Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. ACL.

Yijia Zhang, Qingyu Chen, Z. Yang, H. Lin, and Zhiyong Lu. 2019. Biowordvec, improving biomedical word embeddings with subword information and mesh. *Scientific Data*, 6.

Zhe Zhao, Tao Liu, Shen Li, Bofang Li, and Xiaoyong Du. 2017. Ngram2vec: Learning improved word representations from ngram co-occurrence statistics. In *EMNLP*.

## A Comparison of multiple downstream runs

| Embeddings \Task | Average<br>Mean F1 / Acc. |
|---|---|
| ASE - BERT-12 (5) | $67.0 \pm 0.2/78.1 \pm 0.2$ |
| BERT2STATIC$_{para}$ | $68.3 \pm 0.3/79.9 \pm 0.2$ |
| ASE - RoBERTa-12 (2) | $67.0 \pm 0.2/78.2 \pm 0.3$ |
| RoBERTa2STATIC$_{para}$ | $67.9 \pm 0.2/79.6 \pm 0.3$ |
| ASE - GPT2-12 (4) | $67.4 \pm 0.3/78.3 \pm 0.3$ |
| GPT$_2$2STATIC$_{para}$ | $68.4 \pm 0.2/80.0 \pm 0.4$ |

Table 5: **Comparison of the overall performance of X2STATIC$_{para}$ with ASE on downstream tasks.** Mean and standard deviation of performance on each task over six runs is shown.

## B Experiments on larger models

In addition to the smaller 12-layer contextual embedding models, we also obtain X2STATIC word vectors from larger 24-layer contextual embedding models, once again outperforming their ASE counterparts by a significant margin. The evaluation results can be accessed in the Table 6.

| Model \ Distilled Model | Parent Model \ Other details | Dim. | RG-65 | WS-353 | SL-999 | SV-3500 | RW-2034 | Average |
|---|---|---|---|---|---|---|---|---|
| **Existing models** | Size of the training corpus relative to ours | | | | | | | |
| FastText | 12x | 300 | 0.7669 | 0.596 | 0.416 | 0.3274 | 0.5226 | 0.5276 |
| GloVe | 650x | 300 | 0.6442 | 0.5791 | 0.3764 | 0.2625 | 0.4607 | 0.4646 |
| **Models trained by us** | | | | | | | | |
| Skipgram | N.A. | 768 | 0.8259 | 0.7141 | 0.4064 | 0.2722 | 0.4849 | 0.5407 |
| CBOW | N.A. | 768 | 0.8348 | 0.4999 | 0.4097 | 0.2626 | 0.4043 | 0.4823 |
| Sent2vec | N.A. | 768 | 0.7811 | 0.7407 | 0.5034 | 0.3297 | 0.4248 | 0.55594 |
| **Models distilled by us** | Parent Model | | | | | | | |
| ASE - best layer per task | BERT-12 | 768 | 0.7449(1) | 0.7012(1) | 0.5216(4) | 0.4151(5) | 0.4577(5) | 0.5429(3) |
| ASE - best overall layer | BERT-12 | 768 | 0.6948(3) | 0.6768(3) | 0.5195(3) | 0.3889(3) | 0.4343(3) | 0.5429(3) |
| BERT2Static$_{sent}$ | BERT-12 | 768 | 0.7421 | 0.7297 | **0.5461** | **0.4437** | **0.5469** | **0.6017** |
| BERT2Static$_{para}$ | BERT-12 | 768 | 0.7555 | **0.7598** | **0.5384** | **0.4317** | **0.5299** | **0.6031** |
| ASE - best layer per task | BERT-24 | 1024 | 0.7745(9) | 0.7267(6) | 0.5404(15) | 0.4364(10) | 0.4735(6) | 0.5782(7) |
| ASE - best task independent layer | BERT-24 | 1024 | 0.7677(7) | 0.7052(7) | 0.5209(7) | 0.4307(7) | 0.4665(7) | 0.5782(7) |
| BERT2Static$_{sent}$ | BERT-24 | 1024 | 0.8031 | 0.7239 | **0.5675** | **0.4692** | **0.5595** | **0.6247** |
| BERT2Static$_{para}$ | BERT-24 | 1024 | 0.8085 | **0.7652** | **0.5607** | **0.4543** | **0.5504** | **0.6278** |
| ASE - best layer per task | RoBERTa-12 | 768 | 0.673(0) | 0.7023(0) | 0.554(5) | 0.4602(4) | 0.5075(3) | 0.5600(0) |
| ASE - best overall layer | RoBERTa-12 | 768 | 0.673(0) | 0.7023(0) | 0.5167(0) | 0.4424(0) | 0.4657(0) | 0.5600(0) |
| RoBERTa2Static$_{sent}$ | RoBERTa-12 | 768 | 0.7999 | **0.7452** | 0.5507 | **0.4658** | **0.5496** | **0.6222** |
| RoBERTa2Static$_{para}$ | RoBERTa-12 | 768 | 0.8057 | **0.7638** | **0.5544** | **0.4717** | **0.5501** | **0.6291** |
| ASE - best layer per task | RoBERTa-24 | 1024 | 0.6782(8) | 0.6736(6) | 0.5526(18) | 0.4571(9) | 0.5385(9) | 0.5680(9) |
| ASE - best task independent layer | RoBERTa-24 | 1024 | 0.6738(6) | 0.6270(9) | 0.5437(9) | 0.4571(9) | 0.5385(9) | 0.5680(9) |
| RoBERTa2Static$_{sent}$ | RoBERTa-24 | 1024 | 0.7677 | 0.7336 | 0.5397 | 0.4576 | **0.5720** | **0.6141** |
| RoBERTa2Static$_{para}$ | RoBERTa-24 | 1024 | 0.7939 | **0.7523** | 0.5476 | **0.4663** | **0.5739** | **0.6268** |
| ASE - best layer per task | GPT2-12 | 768 | 0.7013(1) | 0.6879(0) | 0.4972(2) | 0.3905(2) | 0.4556(2) | 0.5365(2) |
| ASE - best overall layer | GPT2-12 | 768 | 0.6833(2) | 0.6560(2) | 0.4972(2) | 0.3905(2) | 0.4556(2) | 0.5365(2) |
| GPT$_2$2Static$_{sent}$ | GPT2-12 | 768 | 0.7484 | 0.7151 | **0.5397** | **0.4676** | **0.5760** | **0.6094** |
| GPT$_2$2Static$_{para}$ | GPT2-12 | 768 | 0.7881 | 0.7267 | **0.5417** | **0.4733** | **0.5668** | **0.6193** |
| ASE - best layer per task | GPT2-24 | 1024 | 0.6574(1) | 0.6957(0) | 0.4988(13) | 0.4226(12) | 0.4566(12) | 0.5155(13) |
| ASE - best task independent layer | GPT2-24 | 1024 | 0.5773(13) | 0.6242(13) | 0.4988(13) | 0.4210(13) | 0.4561(13) | 0.5155(13) |
| GPT$_2$2Static$_{sent}$ | GPT2-24 | 1024 | 0.7815 | 0.7311 | **0.5537** | **0.4774** | **0.5939** | **0.6275** |
| GPT$_2$2Static$_{para}$ | GPT2-24 | 1024 | 0.7907 | 0.7331 | **0.5488** | **0.4850** | **0.5828** | **0.6281** |

Table 6: **Comparison of the performance of different embedding methods on word similarity tasks.** Models are compared using Spearman correlation for word similarity tasks. All X2Static method performances which improve over all ASE methods on their parent model as well as all static models are shown in bold. Best performance in each task is underlined. For all ASE methods, the number in parentheses for each dataset indicates which layer was used for obtaining the static embeddings.