Extracting Text Representations for Terms and Phrases in Technical Domains

Francesco Fusco[*]
IBM Research
ffu@zurich.ibm.com

Diego Antognini[*]
IBM Research
Diego.Antognini@ibm.com

Abstract

Extracting dense representations for terms and phrases is a task of great importance for knowledge discovery platforms targeting highly-technical fields. Dense representations are used as features for downstream components and have multiple applications ranging from ranking results in search to summarization. Common approaches to create dense representations include training domain-specific embeddings with self-supervised setups or using sentence encoder models trained over similarity tasks. In contrast to static embeddings, sentence encoders do not suffer from the out-of-vocabulary (OOV) problem, but impose significant computational costs. In this paper, we propose a fully *unsupervised approach* to text encoding that consists of training small character-based models with the objective of *reconstructing* large pre-trained embedding matrices. Models trained with this approach can not only match the quality of sentence encoders in technical domains, but are *5 times* smaller and up to *10 times* faster, even on high-end GPUs.

1 Introduction

Large pre-trained language models are extensively used in modern NLP systems. While the most typical application of language models is fine-tuning to specific downstream tasks, language models are often used as text encoders to create dense features consumed by downstream components. Among the many use cases of dense text representations there is search, question answering, and classification (Yang et al., 2020).

Static embeddings, trained with algorithms such as Word2Vec (Mikolov et al., 2013), can exploit existing information extraction pipelines to create representations for entities, phrases, and terms present in text corpora. Static embedding matrices are trained with self-supervised approaches at regular intervals, either when additional data is available or to leverage improvements in information extraction models. Pre-trained embedding matrices can be considered as static feature stores, providing dense representations for entries belonging to a fixed vocabulary. Representations for entries outside of the vocabulary are not available, leading to the out-of-vocabulary (OOV) problem.

In contrast, contextualized word embeddings leverage sentence encoders (Cer et al., 2018; Reimers and Gurevych, 2019) to dynamically create dense representations for any input text by performing a forward pass over a large language model. Specifically, a word embedding is computed at inference time based on its context, unlike static word embeddings that have a fixed (context-independent) representation. In practice, sentence encoders solve the out-of-vocabulary (OOV) problem which affects static embeddings at the cost of high computational requirements and stronger dependencies on supervised datasets for similarity.

Despite the popularity of sentence encoders, large pre-trained embedding matrices are still widely adopted in the industry to encode not only individual tokens but also multi-token entities extracted with in-house NLP pipelines. Once those embedding matrices are trained, the text representation for single- and multi-token entries encountered at *training time* can be looked up in constant time.

In this paper, we describe an effective approach taken to provide high-quality textual representations for terms and phrases in a commercially available platform targeting highly-technical domains. Our contribution is a novel *unsupervised approach* to train text encoders that bridges the gap between large pre-trained embedding matrices and computationally expensive sentence encoders. In a nutshell, we exploit the vast knowledge encoded in large embedding matrices to train small character-based models with the objective of *reconstructing* them, i.e., we use large embedding matrices trained with self-supervision as large *training datasets* mapping

[*] Equal contribution.

text to the embedding vectors.

Our approach is extremely attractive for industrial setups as it leverages continuous improvements, testing, and inference costs of existing information extraction pipelines to create large datasets to train text encoders. This way, the return on investment for annotations, development, and infrastructure costs are maximized.

In our evaluation, we highlight that by combining unsupervised term extraction annotators and static embeddings we can train lightweight character-based models that *match* the quality of supervised sentence encoders and provide *substantially better* representations than sentence encoders trained without supervision. Our models not only provide competitive representations, but are up to *5 times* smaller and *10 times* faster than sentence encoders based on large language models.

2 Existing Approaches

The mission of our industrial knowledge discovery platform is to extract knowledge from large corpora containing highly-technical documents, such as patents and papers, from diverse fields ranging from chemistry, to physics, to computer science. Information extraction is extremely challenging given the large variety of language nuances and the large cost of human annotations in such specialized domains. Therefore, it is of extreme importance to minimize the dependencies on annotated data and to use unsupervised approaches whenever possible.

A recurring requirement from many internal components of our platform is the ability to extract high-quality dense representations for technical terms, entities, or phrases which can be encountered in many distinct technical fields. High-quality representations are extremely valuable to implement semantic search, to influence the ranking, or to be used directly as model features.

In modern industrial systems, it is often the case that static and context-dependent embedding technologies coexist on the same platform to extract representations. While static embeddings are trained in a self-supervised fashion, sentence encoders are often built by fine-tuning pre-trained models on similarity tasks using annotated datasets. Having two separate approaches for text encoding is suboptimal as those systems are *completely independent* and embed terms into *distinct embedding spaces*.

To reconcile those two worlds, we propose an approach where static embeddings and text encoders are mapping text into the *same embedding space*. Our intuition is that static embedding matrices storing embeddings for single tokens, but also multi-token terms, entities, or phrases, represent an *invaluable source of information to train text encoders*. While those matrices are built with self-supervision, they can leverage existing annotators, supervised or not, which are commonly available in large text processing platforms.

Our novel approach consists of using pre-trained embedding matrices as a training set, and training character-based models, called CharEmb, to predict the embedding vector for a text. This means that the character-based models will enable to project *any* sequence of characters in the same embedding space as the pre-trained embedding matrices.

2.1 Static Embeddings

Algorithms to train static word embeddings, such as Word2Vec (Mikolov et al., 2013), have been introduced to efficiently compute the representations for words or entire phrases extracted from large text corpora. To create the representation of entire phrases, the original Word2Vec paper suggested to simply preprocess the text to make sure that phrases are treated as distinct words in the vocabulary. In a nutshell, the preprocessing step involves merging the tokens of which a phrase is composed into a unit which is not split by the tokenizer, e.g., [*"machine"*, *"learning"*] becomes [*"machine_learning"*].

In a typical industrial setup, this approach can be naturally generalized to leverage the large set of annotators that are commonly available in large-scale natural language processing platforms. This way one can create domain-specific embeddings not just for single tokens, but for entities, terms, phrases which are extracted in a natural language processing platform. Combining self-supervised word embedding algorithms together with existing sequence tagging models is extremely attractive. First, one can fully leverage the constant enhancements of in-house models to improve the quality of the embeddings for all the entities of interests. Second, since the sequence tagging models are built in-house and independently evaluated, using them to build embedding matrices means reducing the time spent in quality assurance (QA). Third, since the model inference is anyway computed over large amount of textual data while the natural language processing platform is operating, one can amortize

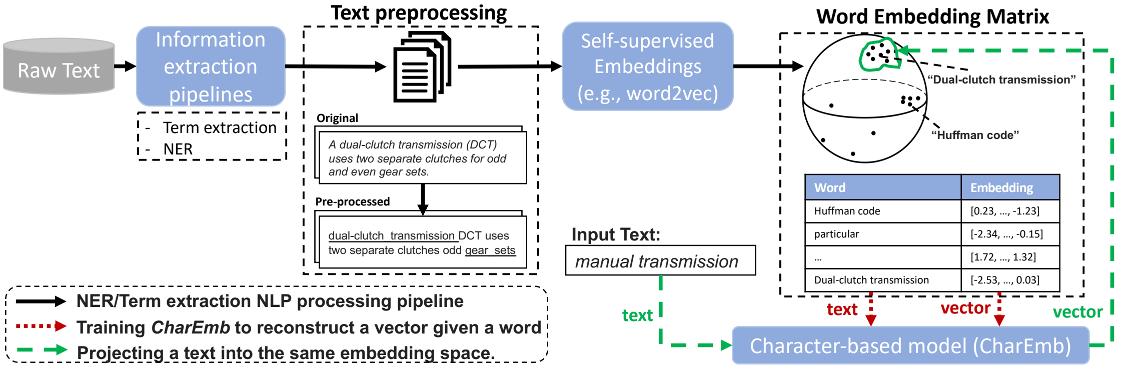
Text preprocessing
Word Embedding Matrix
Raw Text
Information extraction pipelines
- Term extraction
- NER
Original
A dual-clutch transmission (DCT) uses two separate clutches for odd and even gear sets.
Pre-processed
dual-clutch_transmission DCT uses two separate clutches odd gear_sets
Self-supervised Embeddings (e.g., word2vec)
"Dual-clutch transmission"
"Huffman code"
Word | Embedding
Huffman code | [0.23, ..., -1.23]
particular | [-2.34, ..., -0.15]
... | [1.72, ..., 1.32]
Dual-clutch transmission | [-2.53, ..., 0.03]
Input Text:
manual transmission
text
text
vector
vector
Character-based model (CharEmb)
NER/Term extraction NLP processing pipeline
Training CharEmb to reconstruct a vector given a word
Projecting a text into the same embedding space.

Figure 1: Illustration of the data flow for our approach. We introduce CharEmb, a character-based model that projects a given text into the same embedding space of a large pre-trained embedding model. CharEmb uses the pre-trained embedding only at training to learn the projection function between a text and the its embedding vector.

that compute costs to accelerate another task, i.e., providing high-quality text representation.

2.2 Sentence Embeddings

Static embedding matrices built with the previous approach can provide representations for several hundred millions entries when trained over large text corpora pre-processed with several name entity recognition (NER) models. Despite the large size, one can still experience the problem of out-of-vocabulary (OOV), which means, downstream components might require text representations for text entries which are not present in the vocabulary.

Text encoders have been introduced to solve the OOV problem. They provide ways to create embeddings that are not static but contextualized, which means that the embedding vector must be created on the fly via a model inference. Contextualized embeddings can be created using pre-trained models trained with self-supervised setups such as BERT (Devlin et al., 2019) or with text encoders which are still based on large pre-trained models, but fine-tuned with task similarity datasets. Sentence encoders trained with supervised setups using for example the NLI datasets (Bowman et al., 2015; Williams et al., 2018), such as S-BERT (Reimers and Gurevych, 2019) are well known to perform well in practice to create representations for entire sentences or features for finer grained text snippets. The limitation of supervised approaches for sentence encoding is that *creating large annotated datasets for similarity is extremely expensive*, especially in technical fields. Therefore, improving the sentence encoder themself requires substantial investments in annotations. Unsupervised sentence encoder approaches (Gao et al., 2021; Wang et al., 2021), on the other hand are well known to offer poorer performance than supervised counterparts.

3 Our Model: CharEmb

Instead of having two completely disjoint systems to create text representations, we use character-based models trained over large static embedding matrices, that project a sequence of text into the same embedding space as the embedding matrices used as training data. In practice, *we approach text encoding as a compression problem*, where character-based models are trained to reconstruct the pre-trained static embedding matrices, as shown in Figure 1. This training approach can rely on supervised sequence tagging models or can be implemented using fully unsupervised methods, such as the term extraction technologies described in the literature (Fusco et al., 2022). As we highlight in the evaluation section, a text encoder trained without any supervision can match the performance of supervised sentence encoders in creating representations for words or phrases in technical domains.

To train our models, we consider a static pre-trained embedding matrix as gold dataset. An individual training sample associates a single- or multi-token text to an embedding vector. To leverage the dataset we train a text encoder model to minimize the cosine similarity between the produced vector and the original vector stored in the static embedding matrix. The models rely on character-level tokenization to generalize better on unseen inputs in technical domains. Figure 2 highlights a simple yet effective LSTM-based model architecture. The pre-trained static embedding matrix (on the right) contains $|V|$ embedding vectors of size k, where V is the vocabulary of the static embedding matrix. The model receives as input the text t, tokenize it

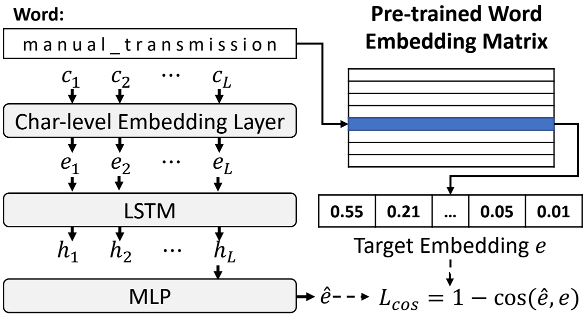

Figure 2: CharEmb is trained to predict an embedding that has a close cosine similarity with the target one.

in characters, for which an embedding matrix is trained. Character-level embeddings are used as input for a Long Short-Term Memory sequence model. The last state of the LSTM layer is used to produce, via a Multi-Layer Perceptron, a vector of dimension k that represents the embedding for the text t. The network is trained to minimize the cosine distance between the predicted embedding and the original one stored in the embedding matrix. The number of distinct training samples is $|V|$, which means that embeddings with large vocabularies correspond to bigger training datasets.

4 Evaluation

In this section, we compare our text representations using the *Patent Phrase Similarity Dataset* built by Google (Aslanyan and Wetherbee, 2022). Given two multiword technical concepts (called *anchor* and *target*), the task consists of predicting a similarity score between the two terms. Unlike general-purpose sentence datasets, such as STS-B (Cer et al., 2017) or SICK (Marelli et al., 2014), we focus on technical concepts in the patent and scientific domains. The dataset contains $38,771$ unique concepts and $36,473$, $2,843$, and $9,232$ concept pairs with humanly-annotated similarity scores for the train, validation, and test sets, respectively.

In our case, we are only interested in the *zero-shot* scenario, and thus, we *only* consider the test set and ignore the train and validation sets. We evaluate the quality of the text representations using the same approach described in Aslanyan and Wetherbee (2022): we compute the Pearson and Spearman correlation between the cosine similarity of the two embeddings and the human-annotated scores.[1]

4.1 Static Pre-trained Word Embeddings

First, we compare the performance of publicly-available pre-trained embedding matrices with embeddings trained with in-domain data. Following the approach described in Aslanyan and Wetherbee (2022), we compute the representation for concepts consisting of multiple tokens as the average of the embedding vectors of each unigram.

To highlight the importance of in domain-data, we train static embedding matrices using a relatively small corpus consisting of 120 million sentences sampled from the the ArXiv (Clement et al., 2019) and the HUPD (Suzgun et al., 2022) datasets. To train our embeddings, we pre-process the text after running term-extraction using the unsupervised method described in Fusco et al. (2022). This way, our method can be considered fully unsupervised, and its evaluation does not depend on proprietary annotations and model architectures.

The size of the text after pre-processing is 18 Gigabytes accounting for 1.9 billion tokens with term-extraction enabled and 2.2 billion tokens without. We train the static embeddings using CBOW (Mikolov et al., 2013). Training for one epoch takes 25 minutes on a 16-core *AMD EPYC 7742*, which corresponds to less than *10 dollars* of compute costs with current cloud offerings. We do not consider the term extraction as part of the overall training costs since in practice, the large amount of annotations that a large-scale NLP platform produces during its execution can be entirely reused.

Finally, we train three variants. The first contains unigrams and multiword expressions extracted with our term extractor represented with one token (i.e., "machine learning" → "machine_learning"). The second considers only unigrams (i.e., with the term-extraction component disabled). For the third we use FastText (Bojanowski et al., 2017) instead.

We compare our embedding matrices with the official pre-trained models for GloVe (Pennington et al., 2014), Word2Vec (Bojanowski et al., 2017), and FastText (Bojanowski et al., 2017). Those are trained on corpora that are substantially larger than our ArXiv-HUPD dataset (up to *300 times*).

Table 1 reports the Pearson and Spearman correlations when using the representations of the static word embedding matrices. Not surprisingly the embeddings trained over the ArXiv-HUPD corpus, which contains text of the same domain, provide substantially better results than embeddings pre-trained over corpora that are out-of-domain, such

[1] For reproducibility purposes, we include all experimental details and the hyperparameters in Appendix A.

				Correlation	
Pre-trained embedding	|Voc.|	Size (MB)	Dim.	Pear.	Spear.
GloVe (6B)	0.4M	458	300	42.37	43.95
GloVe (42B)	1.9M	2,194	300	40.30	45.83
GloVe (840B)	2.2M	2,513	300	44.83	49.71
FastText wiki-news (16B)	1.0M	1,144	300	39.01	46.03
FastText crawl (600B)	2.0M	2,289	300	47.36	49.32
Word2Vec news (100B)	3.0M	2,861	250	44.04	44.72
ArXiv-HUPD (Ours)					
- uni (FastText) (2.2B)	5.3M	6,006	300	51.25	49.92
- uni (Word2Vec) (2.2B)	1.8M	1,403	200	50.82	52.97
- uni + terms (1.8B)	5.2M	3,984	200	**51.62**	**53.91**

Table 1: **Static** context-independent word embeddings. Brackets denote the number of token of the corpus. Training static embeddings on ArXiv-HUPD improves significantly the correlation with the human annotations.

	Pearson Correlation			
Models	Original	Reconstr.	Δ	Compression
CharEmb Small $_{13MB}$	51.62	49.70	-3.7%	306x
CharEmb Base $_{38MB}$		54.33	$+5.3\%$	236x
CharEmb Large $_{86MB}$		55.97	$+8.4\%$	46x

Table 2: **Reconstructed static** word embeddings. We report the correlation of the original ArXiv-HUPD embeddings (uni + terms) and the reconstructed ones inferred by our models. CharEmb Base achieves a compression by a factor of 236 and an improvement of $+5.3\%$.

			Correlation	
Models	Size (MB)	Dim.	Pear.	Spear.
BERT (Unsup.)	1,344	1,024	43.07	41.40
Patent-BERT (Unsup.)	1,344	1,024	54.00	54.47
SimCSE (Unsup.)	438	768	53.35	51.91
Patent-SimCSE (Unsup.)	438	768	50.51	48.33
Sentence-BERT (Sup.)	438	768	59.82	57.66
SimCSE (Sup.)	438	768	56.63	56.81
ArXiv-HUPD - unigrams (Word2Vec) (Ours)				
CharEmb Small (Unsup.)	13	200	41.68	39.55
CharEmb Base (Unsup.)	38		47.57	46.16
CharEmb Large (Unsup.)	86		47.58	45.60
ArXiv-HUPD - unigrams + terms (Ours)				
CharEmb Small (Unsup.)	13	200	55.53	56.73
CharEmb Base (Unsup.)	38		58.53	59.66
CharEmb Large (Unsup.)	86		**59.84**	**60.52**

Table 3: **Sentence**-based word embeddings via predictions. CharEmb is unsupervised, at least 5x smaller, and outperforms large supervised and unsupervised baselines. Training CharEmb on ArXiv-HUPD that contains unigrams clearly underperforms, showing how important multiword expressions are during training.

as news and crawled websites. Our embeddings trained on only 2 billion tokens outperform embeddings trained over corpora that are up to *2 order of magnitude larger*. Further, we see that our static-embedding matrices including terms are providing only a marginal improvement, as the terms do not necessarily cover concepts present in the dataset.

After focusing on the raw embedding matrices, we evaluate the quality of our CharEmb models as compressors. In practice, we repeat the same experiments when the best ArXiv-HUPD word embedding matrix is ***fully reconstructed*** by projecting each vocabulary entry using a character-based model trained with our approach. We report correlations for models based on the Long Short-Term Memory (Hochreiter and Schmidhuber, 1997), because in our experiments, it offered significantly better results than Gated Recurrent Unit (Chung et al., 2014) and Transformer (Vaswani et al., 2017). We report the performance for three model sizes: *Small* (13MB), *Base* (38MB), and *Large* (86MB).

Table 2 shows that our base (38MB) and large (86MB) models compress the embedding matrix they are trained on *and improve its quality* according to the Pearson correlation (similar trend with Spearman). This means that a model of solely 38 MB not only can fully reconstruct the *3.98 GB* matrix it has been trained on, given only its vocabulary (236x space reduction), but also that the *reconstructed* matrix provides a correlation gain of $+5.3\%$ compared to the original one.

4.2 Sentence (Contextualized) Embeddings

In Table 2 we use our models to reconstruct an original embedding matrix. The reconstructed matrix is used as a static pre-trained embedding matrix: given a phrase in the test of the patent dataset, we compute the representation as the average of the unigrams. Instead, in this section we use our models as *text encoders*, which means performing the inference with our CharEmb models to extract representations of the phrases present in the test set.

Following Aslanyan and Wetherbee (2022), we compare our CharEmb variants with the following pre-trained models used as text encoders: BERT (Devlin et al., 2019), Patent-BERT, and the sentence encoder Sentence-BERT (Reimers and Gurevych, 2019) trained on the natural language inference datasets (Bowman et al., 2015; Williams et al., 2018). We augment the proposed baselines with a popular sentence-encoding method SimCSE (Gao et al., 2021), which can be trained with supervision (similarly to S-BERT) or in an unsupervised manner. For the latter, we include the publicly-available variant trained on Wikipedia (Unsupervised SimCSE) and train our own model over the small ArXiv-HUPD dataset (Patent-SimCSE).[2]

In Table 3, we report the Pearson and Spearman correlation when using text encoders to produce text representations via inferencing to a model. Thanks to our approach, lightweight LSTM-based models outperform larger BERT-based models in a *zero-shot* setup. Our large model is *5x smaller* than Sentence-BERT and SimCSE and yet provides *better representations*. Training CharEmb does not require any manual annotation, since embeddings are trained with self-supervision and the term extraction is fully unsupervised. *Our smallest model outperforms all unsupervised approaches.* Furthermore, we note that term extraction is a *fundamental component* for the creation of high-quality CharEmb models. When training over unigram-only embeddings, our models performance drops significantly to the levels of BERT.

Finally, in Figure 3 we show that our models not only provide the best representations, but also offers substantially lower inference latencies on both high-end GPUs and a single-core CPU. Moreover, training CharEmb Large on an embedding matrix with a vocabulary of five million entries takes only *three hours* on a single NVIDIA Tesla A100, which is a negligible time compared to the 10 days required to train SimCSE on the same dataset.

5 Related Work

Static embeddings trained with self-supervised setups became popular with word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). While those algorithms have been originally introduced to embed individual tokens, the approach can be generalized to entire phrases or multiple token entities by preprocessing training corpora such that multiple tokens are merged into one. FastText (Bojanowski et al., 2017) can be seen as an extension to Word2Vec which relies on n-grams to extract representations of unseen text.

Contextualized embeddings (e.g., Elmo (Peters et al., 2018)) are created by taking into account the contex of each token. Sentence encoders (Schuster et al., 2019; Cer et al., 2018) are a generalization of contextual embeddings. They can be trained on sentence-level tasks using supervised datasets, such as NLI, or with unsupervised methods (Gao et al., 2021; Wang et al., 2021). Our method to train text encoders is fully unsupervised and provides higher-quality representations than supervised encoders.

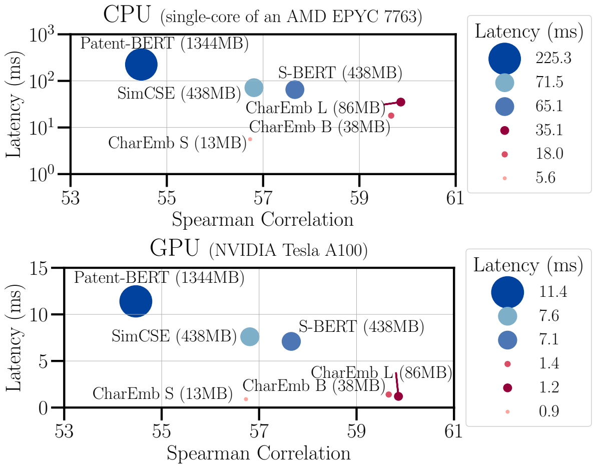

Figure 3: Inference latencies with batch size 1 on a Tesla A100 GPU and a single-core CPU. Our models provide high-quality embeddings on a low-compute budget.

Embedding compression is a topic of great interest not only for natural language processing (Pansare et al., 2022; Liao et al., 2020), but also in recommender systems (Zhang et al., 2020; Kim et al., 2020; Shi et al., 2020). The primary goal of our work is not to reduce the size of a static embedding matrix, but rather to generalize the embeddings to entries not seen at training time.

Work has been done to align embedding spaces coming from different models (Joulin et al., 2018; Schuster et al., 2019; Grave et al., 2019). Instead of aligning spaces coming from static embeddings and sentence encoders, we introduce text encoders trained to project text in the same space of a static embedding matrix used as a training dataset.

6 Conclusion

Creating embeddings for terms and phrases is of paramount importance for complex natural language processing platforms targeting highly-technical domains. While out-of-the-box pretrained sentence encoders are often considered as baselines, representations of *similar quality* can be obtained with substantially lighter and simpler character-based models which are *5 times smaller* in size and *10 times faster* at inference time, even on high-end GPUs. The key to obtaining such results is to realize that large static embedding matrices storing representations for tokens and terms constitute a very rich supervised dataset to train text encoders working at the character level. Since both term extraction and embedding training can be per-

[2] Additional results when training CharEmb on GloVe, FastText, and Word2Vec embeddings are shown in Appendix B.

formed without any labeled data, we have proposed a method to train text encoders which does not require any label. Those models are trained with the objective of reconstructing the original embedding matrix and can not only be used as lighter alternatives to sentence encoders, but also as lossless compressors for large embedding matrices.

References

Grigor Aslanyan and Ian Wetherbee. 2022. Patents phrase to phrase semantic matching dataset. *arXiv preprint arXiv:2208.01171*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Colin B. Clement, Matthew Bierbaum, Kevin P. O'Keeffe, and Alexander A. Alemi. 2019. On the use of arxiv as a dataset.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Francesco Fusco, Peter Staar, and Diego Antognini. 2022. Unsupervised term extraction for highly technical domains. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP), (Industry track)*. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Edouard Grave, Armand Joulin, and Quentin Berthet. 2019. Unsupervised alignment of embeddings with wasserstein procrustes. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1880–1890. PMLR.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2984, Brussels, Belgium. Association for Computational Linguistics.

Yeachan Kim, Kang-Min Kim, and SangKeun Lee. 2020. Adaptive compression of word embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3950–3959, Online. Association for Computational Linguistics.

Siyu Liao, Jie Chen, Yanzhi Wang, Qinru Qiu, and Bo Yuan. 2020. Embedding compression with isotropic iterative quantization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8336–8343.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA).

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in*

Neural Information Processing Systems 26, pages 3111–3119.

Niketan Pansare, Jay Katukuri, Aditya Arora, Frank Cipollone, Riyaaz Shaik, Noyan Tokgozoglu, and Chandru Venkataraman. 2022. Learning compressed embeddings for on-device inference. In *Proceedings of Machine Learning and Systems 2022, MLSys 2022, Santa Clara, CA, USA, August 29 - September 1, 2022*. mlsys.org.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E. 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.

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

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1599–1613, Minneapolis, Minnesota. Association for Computational Linguistics.

Hao-Jun Michael Shi, Dheevatsa Mudigere, Maxim Naumov, and Jiyan Yang. 2020. Compositional embeddings using complementary partitions for memory-efficient recommendation systems. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 165–175, New York, NY, USA. Association for Computing Machinery.

Mirac Suzgun, Luke Melas-Kyriazi, Suproteem K. Sarkar, Scott Duke Kominers, and Stuart M. Shieber. 2022. The harvard uspto patent dataset: A large-scale, well-structured, and multi-purpose corpus of patent applications.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Kexin Wang, Nils Reimers, and Iryna Gurevych. 2021. TSDAE: Using transformer-based sequential denoising auto-encoderfor unsupervised sentence embedding learning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 671–688, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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, New Orleans, Louisiana. Association for Computational Linguistics.

Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-hsuan Sung, Brian Strope, and Ray Kurzweil. 2020. Multilingual universal sentence encoder for semantic retrieval. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 87–94, Online. Association for Computational Linguistics.

Caojin Zhang, Yicun Liu, Yuanpu Xie, Sofia Ira Ktena, Alykhan Tejani, Akshay Gupta, Pranay Kumar Myana, Deepak Dilipkumar, Suvadip Paul, Ikuhiro Ihara, Prasang Upadhyaya, Ferenc Huszar, and Wenzhe Shi. 2020. Model size reduction using frequency based double hashing for recommender systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, page 521–526, New York, NY, USA. Association for Computing Machinery.

A Training Details

To perform the experiments described in Table 3 we use pre-trained models publicly available in HuggingFace:

- **BERT**: bert-large-uncased.
- **Patent-BERT**: anferico/bert-for-patents.
- **Sentence-BERT**: sentence-transformers/all-mpnet-base-v2.
- **Supervised-SimCSE**: princeton-nlp/sup-simcse-bert-base-uncased.
- **Unsupervised-SimCSE**: princeton-nlp/unsup-simcse-bert-base-uncased.

Regarding hyperparameter tuning, the baselines do not need any tuning since all experiments are in a zero-shot fashion. For EmbChar, we only tune

				Correlation					
				Original		Reconstr.		Context.	
Pre-trained embedding	|Voc.|	Size (MB)	Dim.	Pear.	Spear.	Pear.	Spear.	Pear.	Spear.
GloVe (6B)	0.4M	458	300	42.37	43.95	36.82	41.15	27.36	31.36
GloVe (42B)	1.9M	2,194	300	40.30	45.83	29.89	42.93	20.66	21.35
GloVe (840B)	2.2M	2,513	300	44.83	49.71	39.32	47.39	18.97	22.79
FastText wiki-news (16B)	1.0M	1,144	300	39.01	46.03	30.72	45.66	28.82	27.47
FastText crawl (600B)	2.0M	2,289	300	47.36	49.32	45.91	49.79	34.49	35.67
Word2Vec news (100B)	3.0M	2,861	250	44.04	44.72	40.77	45.28	45.54	46.09
ArXiv-HUPD uni (2.2B)	1.8M	1,403	200	50.82	52.97	54.28	55.21	47.58	45.6
ArXiv-HUPD uni + terms (1.8B)	5.2M	3,984	200	**51.62**	**53.91**	**55.97**	**57.27**	**59.84**	**60.52**

Table 4: Additional results when training *CharEmb Large (86MB)* on standard pre-trained word embedding matrices. Without multiword expression, the reconstruction and contextualized via prediction performance are limited.

the encoder by using LSTM, GRU, or Transformer on the validation set. More specifically, we split the word embedding matrix into train and validation sets with a ratio of 80-20. No other hyperparameters have been explored. We stop the training of EmbChar using early-stopping on the validation set when the average cosine similarity has not been improved since 10 epochs.

Our hyperparameters are shown in Table 5. We train our word embedding with Word2Vec with CBOW and a window size of 8 and 25 epochs. For FastText, we kept the default parameters.

All experiments have been run on the following hardware:

- **CPU**: AMD EPYC 7763 64-core processor.
- **RAM**: 1.96 TB.
- **GPU**: NVIDIA Tesla A100.
- **OS**: Red Hat Enterprise Linux 8.6.
- **Software**: PyTorch 1.12.1, CUDA 11.6.

We emphasise that we train the word embedding matrices on a 16-core virtual machine hosted on *AMD EPYC 7742*. An epoch takes approximately 25 minutes. Training our embedding matrix *ArXiv-HUPD uni + terms* requires *less* than 10 dollars of compute budget in the cloud. Training our model EmbChar Large thereafter takes a few hours on a single NVIDIA Tesla A100, costing approximately \$5 to \$10[3]. In contrast, training SimCSE on the same dataset takes around 10 days.

Hyperparameter	EmbChar Small	EmbChar Base	EmbChar Large
Hidden dimension	512	512	768
Number of layers	1	2	2
Bidirectional	True	True	True
Dropout	0.2	0.2	0.2
Learning rate	0.001	0.001	0.0005
Weight decay	1e-8	1e-8	1e-8
Batch size	256	256	256

Table 5: The hyperparameter for all EmbChar variants.

B Additional Results

In Table 4 we report the results for the experiments in Section 4 when training our CharEmb Large (86MB) on different word embedding matrices. For each of the embedding matrix considered, we measure the Pearson and Spearman correlation for the three setups: i) the original embedding matrix (*Original*), ii) an embedding matrix reconstructed using the same vocabulary of the original one (*Reconstr.*), and iii) when the embedding of given terms are contextual, i.e., predicted with a CharEmb model trained over the original matrix (*Context.*). The table showcases multiple important findings. First, among the pre-trained models, the vocabulary size plays a significant role to achieve high correlation, with the Word2Vec model with a vocabulary of 3 million entries outperforming embedding matrices that have been trained over larger datasets (e.g., Glove 840B or FastText crawl). Second, the domain of the corpus used to train the embeddings plays a significant role. By training with a corpus of only 2 billion in-domain tokens, an embedding matrix with a vocabulary of 1.8 million entries achieves similar correlation

[3] The hourly pricing for spot instances with one A100 GPU is in the range \$1.25-\$1.5 in public cloud offerings.

of much larger embedding matrices. Third, our CharEmb model achieves the best performance when trained with an embedding matrix containing embeddings for terms. Predicting the embeddings with our CharEmb model allows to achieve significantly higher correlation than the original matrix containing terms.