Injecting Wiktionary to improve token-level contextual representations using contrastive learning

Anna Mosolova[1,2], **Marie Candito**[1] , **Carlos Ramisch**[2]
[1]Université Paris Cité, CNRS, LLF, Paris, France
[2]Aix Marseille Univ, CNRS, LIS, Marseille, France
first.last@u-paris.fr, first.last@lis-lab.fr

Abstract

While static word embeddings are blind to context, for lexical semantics tasks context is rather too present in contextual word embeddings, vectors of same-meaning occurrences being too different (Ethayarajh, 2019). Fine-tuning pre-trained language models (PLMs) using contrastive learning was proposed, leveraging automatically self-augmented examples (Liu et al., 2021b). In this paper, we investigate how to inject a lexicon as an alternative source of supervision, using the English Wiktionary. We also test how dimensionality reduction impacts the resulting contextual word embeddings. We evaluate our approach on the Word-In-Context (WiC) task, in the unsupervised setting (not using the training set). We achieve new SoTA result on the original WiC test set. We also propose two new WiC test sets for which we show that our fine-tuning method achieves substantial improvements. We also observe improvements, although modest, for the semantic frame induction task. Even if we experimented on English to allow comparison with related work, our method is adaptable to the many languages for which large Wiktionaries exist.

1 Introduction

Pretrained language models (PLMs) have brought great advances in most NLP tasks. As far as word embeddings are concerned, though, we have moved from one extreme to the other, namely from static word embeddings providing a single representation for a given form, no matter how ambiguous it is, to contextual token embeddings providing one representation per occurrence. For lexical level tasks, while it is desirable that token-level vectors of the same word sense are close in the semantic space, this is not the case for the majority of PLMs (Ethayarajh, 2019).

In this paper, we address the tuning of token-level contextual representations to better target the lexical sense instantiated by a given token. We use the contrastive learning (CL), which proved efficient for getting sentence embeddings that better capture sentence-level similarity (Reimers and Gurevych, 2019; Gao et al., 2021; Chuang et al., 2022; Fang et al., 2020) and for getting better token-level embeddings (Liu et al., 2021b; Su et al., 2022). These approaches use self-supervised CL, with positive examples created by pairing an original sentence and an automatically modified version of it.

In this paper, we rather investigate how to leverage hand-crafted lexicons. Although these are not always perfectly tailored to NLP tasks, due to coverage and granularity mismatches with the task or domain at hand, they do contain an enormous amount of lexical information that is a pity not to make use of. To do so, we use CL on the example sentences of the English Wiktionary, a crowd-sourced lexicon. We will show the approach is beneficial for both the Word-in-Context (WiC) task (intrinsic evaluation), and for the frame induction task (extrinsic evaluation). Crucially, although we experiment on English to allow comparison with related work, our method is adaptable to a large number of languages for which large Wiktionaries exist.

We also investigate whether reducing dimensions can provide better-suited token-level contextual embeddings.

In the following, we describe related work (§ 2), and how we adapted the CL loss to Wiktionary examples (§ 3). We present our language model fine-tuning experiments, along with an evaluation on the Word-in-Context task (§ 4). We test whether our fine-tuned token embeddings can help cluster verbal occurrences into semantic frames (§ 5).

2 Related Work

Within the deep metric learning paradigm, contrastive learning (CL) became increasingly popular in computer vision and in NLP (Kaya and Bilge, 2019). It consists in modifying the representation

Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics
Volume 2: Short Papers, pages 34–41

space so that similar objects (positive examples) are brought closer while dissimilar objects are pushed away from each other. Hadsell et al. (2006) proposed one of the first contrastive loss functions, for binary positive examples. CL methods are either supervised or self-supervised. While the former rely on labeled data, the latter employ automatic modifications of objects to produce binary positive pairs (self-augmentation). Since there can be more than two examples of the same class, Khosla et al. (2020) adapt the contrastive loss to handle "multiple-positive" examples for computer vision.

In NLP, CL is primarily used to improve sentence representations, better capturing sentence similarity, mainly in the self-supervised paradigm. Self-augmentation techniques include back translation (Fang et al., 2020), text corruption (Liu et al., 2021a), or PLM's dropout to produce slightly different embeddings per encoding run (Gao et al., 2021; Chuang et al., 2022). Zhuo et al. (2023) combine whitening and CL to fine-tune sentence representations by PLMs. Supervised CL is much less common. We can only cite Gunel et al. (2021) who use it for fine-tuning a PLM while learning a downstream sentiment-analysis classifier.

In contrast to sentence embeddings, fewer works focus on token-level PLM representations. Liu et al. (2021a,b) fine-tune contextual embeddings using self-supervised CL, creating positive pairs with dropout and random masking of context tokens. Su et al. (2022) use CL to favor more isotropic token-level representations. They train a student BERT model on the masked language modeling task with a help of a frozen teacher BERT model: CL aims at increasing the similarity of student and teacher token representations.

Apart from CL, there was also work in enhancing BERT with senses during pre-training. For example, Levine et al. (2020) add supersense prediction for every masked word as pre-training objective.

Finally, since we heavily rely on similarities of contextual embeddings, we mention studies reporting the particularities of such spaces. Timkey and van Schijndel (2021) show that very few dimensions dominate the cosine similarity and propose postprocessing methods to smooth this effect. Zhou et al. (2022) identified and Wannasuphoprasit et al. (2023) tried to solve the problem of underestimated cosine similarity for high-frequency words.

Our goal is to obtain token-level contextual representations more aware of lexical semantics, by injecting lexicon-based information using CL. We show that this injection is beneficial for the closely related WiC task, and, to some extent, for the more downstream task of frame induction.

3 CL for lexical sense examples

Our method fine-tunes the token-level contextual representations of a PLM using supervised CL, taking the examples of a lexicon as supervision. More precisely, each example sentence in the lexicon is associated with a word sense and contains a target word occurrence used in this particular sense.

We adapt the multiple-positive contrastive loss of Khosla et al. (2020) to the use of a lexicon as labeled data.[1] Let $E(l)$ be the set of example sentences for lemma l. For an example $j \in E(l)$, let $S(j)$ be the subset of $E(l)$ of examples concerning the same word sense as j, except for j itself. For every lemma l, we create a single batch, and we define a loss summing over the set $E(l)$ of all examples of l:

$$\mathcal{L}(l) = \sum_{j \in E(l)} \frac{-1}{|S(j)|} \sum_{j' \in S(j)} \log \frac{e^{s(j,j')/\tau}}{\sum_{k \in E(l) \setminus j} e^{s(j,k)/\tau}}$$

with $E(l) \setminus j$ being $E(l)$ except j. We write $s(m, n)$ for the similarity between the embeddings of the target tokens in examples m and n (s can be any vector similarity function), and τ is a scalar temperature hyperparameter.

In order to cope with known flaws of cosine similarity for high-dimensional spaces, we also experiment with a simple PCA reduction of the PLM embeddings, with or without whitening.

4 PLM fine-tuning experiments

Training dataset More precisely, our training data includes the examples for all verbs having from 1 to 10 senses, except verbs having a single sense with a single example, and multiword verbs. In total, we obtained a dataset of 13,118 verbs having in total 26,398 senses, with a total of 68,271 examples. Mean number of examples per sense is 2.59 (std. dev. is 5.41). Mean number of senses per verb is 2.01 (std. dev. is 1.54). Mean number of examples per verb is 5.21 (std. dev. is 12.68). Each example concerns a target verb occurrence. For

[1] Khosla et al. (2020) test two formulations, varying in the precedence of log and summation over the same-class examples. They empirically show the superiority of applying log first. Gunel et al. (2021) also adopt this formulation.

hyperparameter tuning and evaluation, we split the dataset into 95/5/5% for training, development and test sets, ensuring that verb lemmas do not overlap between the three sets.

Training details We report experiments using the bert-base-uncased model (Devlin et al., 2019).[2] For the similarity metric (the s function), we settled for cosine after a few experiments with various similarity metrics (euclidean distance, dot product).

The training procedure iterates for E epochs, each epoch looping over shuffled training batches (one batch per lemma). We limited the batches' size by randomly selecting at most 64 examples per lemma ($\max(|E(l)|) = 64$). For a given batch, each example sentence j is encoded using the current version of the PLM. The similarities $s(m, n)$ are computed by extracting the embedding, at the last layer, of the target tokens in m and in n.[3]

Intrinsic evaluation: Word-in-Context (WiC) is a binary classification task taking as input a pair of sentences containing the same target lexical unit, and predicting whether this target unit is used with the same meaning or not (Pilehvar and Camacho-Collados, 2019). We use this task both to tune our CL method and to evaluate its benefits. We stress that since our objective is to evaluate contextual embeddings, we only consider the unsupervised scenario of the WiC task. Hence, we do not use the training WiC data at all.

For our hyperparameter tuning and evaluation, we use three kinds of WiC data **(i) WiktWiC** is the data closest to our training data, namely the dev and test Wiktionary example set mentioned in § 4, **(ii) OrigWiC** are the original dev and test sets of the WiC task dataset[4] and **(iii) FramenetWiC**, containing FrameNet 1.7 example pairs for the same verb, annotated with the same or different frames. Statistics for these datasets are provided in Appendix A.1, Table 3. Each dataset is balanced for positive and negative pairs, hence the default metric is macro-averaged accuracy.

We perform the WiC task by applying a threshold on the cosine similarity between the target token embeddings (at the last layer) for the input sentences. Thus, we evaluate the impact of fine-tuning on the embeddings, without the influence of any additional classifier. The threshold is tuned with step size 0.02 on the development sets.

FT	PCA	Wikt WiC	Frame WiC	Orig WiC
-	-	55.9	67.3	65.4
-	+	59.6	72.4	68.4
+	-	70.0(±0.9)	69.6(±0.4)[5]	69.6(±0.6)
+	+	**70.5**(±0.8)	**73.1**(±0.4)	**71.4**(±0.2)
MirrorWiC		-	-	69.6

Table 1: Results on WiC test sets. **FT**: with or without fine-tuning. **PCA**: with or without PCA dimensionality reduction (100 components, with whitening). FT=+ rows are averages of 5 runs (std. dev. in parentheses).

Hyperparameter tuning To tune the hyperparameters, we used as a criterion the WiC accuracy, macro-averaged on the three development sets (Table 3). The tested values and their results are provided in Appendix A.2, Table 4. We chose the hyperparameter combination leading to the highest accuracy on average for the five runs, namely: learning rate = 5e-6, 2 epochs, temperature=0.5, PCA with whitening and 100 components.

Unsupervised WiC results As a baseline, we use the bert-base-uncased model, without applying PCA (first row of Table 1). The results are statistically significant[5] in comparison with the baseline according to McNemar's test with $\alpha = 0.05$. We observe that our fine-tuning improves results for the three test sets. The best improvement is for the test set of the closest kind (WiktWiC), but improvements are also substantial for the two other test sets, which shows the method generalizes to other kinds of sense definitions, of varying granularity. We further observe that PCA is beneficial when applied to plain BERT embeddings, and the improvements add up when applying both fine-tuning and PCA.

We also compare our results on the OrigWiC dataset to MirrorWiC (Liu et al., 2021b), which leverages self-supervised CL to improve the last 4 layers of the token-level PLM embeddings. Our approach outperforms MirrorWiC, which shows that supervision even from a crowd-sourced lexicon surpasses the use of self-augmented examples.

[2] Pilehvar and Camacho-Collados (2019) report BERT as the best-performing model in the unsupervised setting for the WiC task (§ 4). We used the -base instead of -large model to reduce the computational cost.

[3] Sub-word token embeddings are averaged per word.

[4] The original WiC dataset contains examples from VerbNet, WordNet and Wiktionary (Pilehvar and Camacho-Collados, 2019). We deleted from all our Wiktionary dataset (train, dev, and test) *all* examples in OrigWiC.

[5] Except for the result of the fine-tuned model without PCA on the Frame WiC dataset, where the improvement was statistically significant on 3 runs out of 5.

Model	Layer	α_2	#pLU	#C	Pu/iPu/PiF$_1$	BcP/BcR/BcF$_1$	Pu/iPu/PiF	BcP/BcR/BcF
B	11/2	0.6	1059	313	95.3/**99.6**/**96.8**	94.4/**99.5**/**96.0**	65.0/**75.5**/69.8	56.3/**67.1**/61.3
B+P	10/2	0.5	1083	307	95.5/99.2/96.7	94.7/98.9/95.9	65.3/72.2/68.6	54.7/62.4/58.3
B+FT	11/2	0.1	1228	394	**97.4**/96.3/96.3	**96.7**/95.3/95.2	68.4/72.2/70.2	59.8/62.9/61.3
B+FT+P	11/2	0.2	1157	381	96.6/97.8/96.7	95.8/97.2/95.7	**69.9**/73.6/**71.7**	**60.5**/63.9/**62.1**

Table 2: Results on the frame induction test set of Y21. **B**: bert-base-uncased, **P**: with PCA (100 components, with whitening), **FT**: with our fine-tuning. **Layer x/y**: layer x used for 1st step, and y for 2nd step clustering. α_2: weight of the masked embedding for the 2nd step. **#pLU**: number of pseudo-lexical units after the 1st step, **#C**: number of clusters after the 2nd step. Clustering algorithms are X-means (1st step) and group-average (2nd step). Gold number of LUs is 1,188, actual number of frames is 393. FT=+ rows report averages of 5 runs. **Pu/iPu/PiF**$_1$: purity, inverse purity, and Fscore for the first step. **BcP/BcR/BcF**$_1$: B-cubed precision/recall/Fscore for the first step. **Pu/iPu/PiF** and **BcP/BcR/BcF**: same but for the 2nd step.

To the best of our knowledge, 71.4% is the new state-of-the-art on the OrigWiC test set in the unsupervised setting, and it even surpasses some supervised settings that use the OrigWiC training set (see Loureiro et al. (2022)).

5 Extrinsic evaluation : frame induction

We now turn to evaluating our fine-tuning approach on semantic frame induction. Compared to word sense induction, frame induction seeks to identify semantic classes (or frames) that may group senses of different lemmas. It is thus a challenging task for token embeddings. We reuse the dataset of Yamada et al. (2021) (hereafter **Y21**), extracted from the lexicographic part of Framenet 1.7.

We reproduce the approach of Y21 with minor modifications. It takes as input a set of words, each in the context of a sentence. Occurrences of the same lemma are clustered first, and the resulting clusters (called pseudo-lexical units) are then averaged and further clustered to form frames. To represent the target words to cluster, Y21 use a weighted average of two token embeddings obtained after applying a PLM on the original sentence, with and without masking the target word. We describe our minor modifications and hyperparameter tuning on Y21's dev set in Appendix A.3.

We select the best hyperparameter combination (using the F-B-Cubed metric of the second clustering step) for each of the four types of embeddings: with and without CL fine-tuning, and with and without PCA. Results on the test set are provided in Table 2, for the four systems[6] (results on the dev set are in Table 5, Appendix A.4). We did not perform the statistical significance test for this task, as it would require using bootstrapping which is extremely costly given that a new clustering must be created for each resampled pseudo-test set. For the first step, fine-tuning improves Purity and B-Cubed Precision, which means that clusters identified with the fine-tuned model contain less noise. However, items from the same frame tend to be divided into several clusters. With the two-step algorithm, such errors are recoverable, as the additional clusters can be merged during the second step, whereas over-merging cannot be undone by the second step.

For frame induction (second step), while for the dev set our CL fine-tuning is clearly beneficial (+5.1 points for BcF), the increment on the test set is more modest and is only obtained with PCA (62.1 compared to 61.3). The utility of CL fine-tuning for this task is thus limited, but with PCA it provides shorter embeddings, reducing computational cost for downstream tasks.

We also notice that the best layers are high layers for the first step, but low layers for the second step. Moreover, after fine-tuning, the tuned α_2 is close to 0, suggesting that flaws of the original unmasked token representations that were fixed when combining with the masked embeddings, were smoothed away during the fine-tuning step.

6 Conclusions

We presented a new approach for fine-tuning token-level representations of PLMs, using contrastive learning with examples from the English Wiktionary, a crowd-sourced lexicon. We show its effectiveness on the Word-in-Context task: we establish the new SoTA on the WiC test set, in the unsupervised setting (not using the WiC training set), and we also obtain substantial gains on two new

[6]For plain BERT, we were unable to reproduce Y21's results (PiF=73.0%, BcF=64.4%), despite extensive tests. This might be due to hyperparameters left implicit in their description. We could not obtain answers from the authors.

WiC test sets, with different sense inventories. We also report improvements, though more modest, on the downstream task of semantic frame induction. Although we experimented on English, our method is adaptable to the many languages for which large Wiktionaries exist and provides a simple way to obtain token-level embeddings more adapted for lexical semantic tasks. A promising continuation of this work is to create positive examples using Wiktionary example sentences for distinct lemmas.

7 Limitations

This paper proposes a new approach for fine-tuning token-level representations of PLMs. Our study is based on fine-tuning a single bert-base-uncased model. We believe that fine-tuning of its large version or other PLMs should also be studied to prove the generalisability of the method. Additionally, we conduct our experiments only using datasets in the English language. Our assumption of its applicability to other languages must also be tested in future work. As for the training dataset, we use only verbal lemmas for its constriction. However, it should be verified whether using lemmas of all parts of speech improves or worsens the fine-tuning results.

We show the limited utility of CL fine-tuning for the frame induction task compared to the improvements achieved on the WiC datasets. We used only a single extrinsic task due to space limitations. Other lexical level tasks, such as word sense induction, can also be easily applied to investigate further abilities of the new representations (e.g. Task 14 of SemEval-2010 (Manandhar et al., 2010)).

Acknowledgements

We thank the reviewers for their valuable feedback on our work.

This work has been funded by the French Agence Nationale pour la Recherche, through the SELEXINI project (ANR-21-CE23-0033-01).

References

Yung-Sung Chuang, Rumen Dangovski, Hongyin Luo, Yang Zhang, Shiyu Chang, Marin Soljacic, Shang-Wen Li, Scott Yih, Yoon Kim, and James Glass. 2022. DiffCSE: Difference-based contrastive learning for sentence embeddings. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4207–4218, Seattle, United States. Association for Computational Linguistics.

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.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. 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 55–65, Hong Kong, China. Association for Computational Linguistics.

Hongchao Fang, Sicheng Wang, Meng Zhou, Jiayuan Ding, and Pengtao Xie. 2020. CERT: Contrastive Self-supervised Learning for Language Understanding. ArXiv:2005.12766 [cs, stat].

Charles J Fillmore and Collin F Baker. 2010. A frame semantic approach to linguistic analysis. In *Oxford Handbook of Linguistic Analysis*. Oxford University Press.

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.

Beliz Gunel, Jingfei Du, Alexis Conneau, and Ves Stoyanov. 2021. Supervised Contrastive Learning for Pre-trained Language Model Fine-tuning. ArXiv:2011.01403 [cs].

R. Hadsell, S. Chopra, and Y. LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742.

Mahmut Kaya and Hasan Şakir Bilge. 2019. Deep metric learning: A survey. *Symmetry*, 11(9):1066.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673.

Yoav Levine, Barak Lenz, Or Dagan, Ori Ram, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2020. SenseBERT: Driving some sense into BERT. In *Proceedings of the*

58th Annual Meeting of the Association for Computational Linguistics, pages 4656–4667, Online. Association for Computational Linguistics.

Fangyu Liu, Ivan Vulić, Anna Korhonen, and Nigel Collier. 2021a. Fast, effective, and self-supervised: Transforming masked language models into universal lexical and sentence encoders. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1442–1459, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Qianchu Liu, Fangyu Liu, Nigel Collier, Anna Korhonen, and Ivan Vulić. 2021b. MirrorWiC: On eliciting word-in-context representations from pretrained language models. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 562–574, Online. Association for Computational Linguistics.

Daniel Loureiro, Alípio Mário Jorge, and Jose Camacho-Collados. 2022. Lmms reloaded: Transformer-based sense embeddings for disambiguation and beyond. *Artificial Intelligence*, 305:103661.

Suresh Manandhar, Ioannis Klapaftis, Dmitriy Dligach, and Sameer Pradhan. 2010. SemEval-2010 task 14: Word sense induction &disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 63–68, Uppsala, Sweden. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. 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 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

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*. Association for Computational Linguistics.

Yixuan Su, Fangyu Liu, Zaiqiao Meng, Tian Lan, Lei Shu, Ehsan Shareghi, and Nigel Collier. 2022. TaCL: Improving BERT pre-training with token-aware contrastive learning. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2497–2507, Seattle, United States. Association for Computational Linguistics.

William Timkey and Marten van Schijndel. 2021. All bark and no bite: Rogue dimensions in transformer language models obscure representational quality. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4527–4546, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Saeth Wannasuphoprasit, Yi Zhou, and Danushka Bollegala. 2023. Solving cosine similarity underestimation between high frequency words by ℓ_2 norm discounting. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8644–8652, Toronto, Canada. Association for Computational Linguistics.

Kosuke Yamada, Ryohei Sasano, and Koichi Takeda. 2021. Semantic frame induction using masked word embeddings and two-step clustering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 811–816.

Kaitlyn Zhou, Kawin Ethayarajh, Dallas Card, and Dan Jurafsky. 2022. Problems with cosine as a measure of embedding similarity for high frequency words. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 401–423, Dublin, Ireland. Association for Computational Linguistics.

Wenjie Zhuo, Yifan Sun, Xiaohan Wang, Linchao Zhu, and Yi Yang. 2023. WhitenedCSE: Whitening-based contrastive learning of sentence embeddings. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12135–12148, Toronto, Canada. Association for Computational Linguistics.

A Appendix

A.1 Statistics for the three Word-in-Context datasets

We provide the statistics for the three WiC datasets in table 3. We introduce 2 datasets: Wikt-WiC, which is a derivative of the Wiktionary DBnary dataset distributed under the Creative Commons Attribution-ShareAlike 3.0 license, and Framenet-Wic, which is created from the Framenet 1.7 examples (Fillmore and Baker, 2010)[7] shared under the Creative Commons Attribution-Only license. We also reuse the original WiC dataset distributed under the Creative Commons Attribution-NonCommercial 4.0 license.

[7] http://framenet.icsi.berkeley.edu/

Dataset	Dev	Test
Orig-WiC	638	1400
Wikt-WiC	1200	1200
Framenet-WiC	1800	1700

Table 3: Statistics for three WiC evaluation datasets.

A.2 Hyperparameter tuning of BERT fine-tuning by contrastive learning with Wiktionary examples, on the development sets of the WiC task

We tuned the following hyperparameters using grid search: learning rate (tested values: 5e-7, 1e-6, 5e-6, 1e-5, 3e-5, 5e-5), number of epochs (from 1 to 6), temperature[8], whether to use PCA or not (with or without whitening and number of components (tested values: from 100 to 700 with the step 100).

We made five runs for each hyperparameter combination to determine the variance of the results.

Table 4 shows the top 10 hyperparameter combinations of the bert-base-uncased CL fine-tuning. Additionally, we report results without fine-tuning as a baseline and MirrorWiC results on the development set (results from (Liu et al., 2021b)).

The average training time of the bert-base-uncased model[9] (110M parameters) for one epoch is 30 minutes on one 4Gb GPU. For the fine-tuning, we used Transformers and SentenceTransformers libraries (Reimers and Gurevych, 2019). We also use PCA implementation from the scikit-learn library (Pedregosa et al., 2011).

A.3 Hyperparameter tuning for the frame induction experiments

To represent the target words to cluster, Y21 use a weighted average of two token embeddings obtained after applying a PLM on the original sentence, with and without masking the target word. The used embedding for a target word is $\alpha \cdot v_{MASK} + (1 - \alpha) \cdot v_{WORD}$. Y21 use $\alpha_1 = 1$ for the first step, and a tuned α_2 for the second step. We also tune α_2, but we rather use $\alpha_1 = 0$, namely a plain embedding of the target word, without any masking, as we observed no impact on the results. Another difference in our implementation is that we may use different BERT layers for the first and second clustering steps, while Y21 always use the same. The hyperparameter tuning, on the development set, is the following:

- First step clustering algorithm:
  - X-means with minimum and maximum number of clusters set to 1 and 15 respectively,
  - Agglomerative clustering with group average linkage.
- Combination of BERT layers for first and second steps: out of the 144 layer combinations, we first selected the 10 best combinations using the bert-base-uncased model with $\alpha_2 = 0$ and checked only 10 best combinations with the rest of hyperparameters.
- α_2 : tested values from 0 to 1 with step 0.1.

We do not tune the following hyperparameters:

- Number of components for PCA is always 100 with whitening application (the best combination identified in the WiC tuning).
- Algorithm for the second step: Agglomerative clustering with group average linkage (with termination criterion as defined by Y21).

A.4 Results of the frame induction task on the development set

In the table 5, we present the results on the development set of the frame induction task. We can see the improvement of all results after fine-tuning and a small degradation of the results after the PCA application. However, the clustering time is shorter by 13% when reduced embeddings are used (2 minutes vs 2.3 minutes). Also, we observe that α_2 values are close to 0 after fine-tuning suggesting removing the masked embedding completely as the overall computation time will be reduced by 2 times without its application.

B-Cubed metrics are computed using f-b-cubed python library[10], purity metrics are computed with scikit-learn (Pedregosa et al., 2011).

[8] We did some preliminary tests with all values from 0 to 1 with the step 0.1, and we finally only tested values 0.3 and 0.5 for the grid search.

[9] https://huggingface.co/bert-base-uncased

[10] https://github.com/hhromic/python-bcubed

LR	E	τ	N comp.	Whitening	Macro-Accuracy	Orig-WiC	Framenet-WiC	Wikt-WiC
bert-base-uncased			-	-	65.6	67.9	70.9	58.0
bert-base-uncased			100	True	67.5	69.6	73.9	58.9
5e-6	2	0.5	100	True	**71.4**(±0.1)	73.5(±0.5)	76.0(±0.2)	64.8(±0.5)
5e-6	3	0.5	100	True	71.4(±0.2)	73.7(±0.4)	75.8(±0.2)	64.8(±0.3)
5e-6	3	0.5	300	True	71.4(±0.4)	72.0(±0.7)	77.6(±0.4)	64.4(±0.4)
5e-6	2	0.5	300	False	71.3(±0.2)	**73.9**(±0.4)	74.6(±0.2)	65.3(±0.4)
5e-6	2	0.5	300	True	71.3(±0.4)	71.9(±0.6)	**77.8**(±0.3)	64.1(±0.6)
5e-6	3	0.5	400	True	71.2(±0.4)	72.0(±0.8)	77.5(±0.4)	64.1(±0.5)
5e-6	3	0.5	200	True	71.2(±0.2)	72.6(±0.5)	76.7(±0.2)	64.3(±0.4)
5e-6	2	0.5	200	False	71.2(±0.3)	73.5(±0.5)	74.6(±0.3)	**65.4**(±0.3)
5e-6	1	0.5	100	True	71.2(±0.1)	72.8(±0.4)	75.8(±0.2)	64.9(±0.4)
5e-6	2	0.5	400	False	71.1(±0.3)	73.6(±0.5)	74.5(±0.2)	65.2(±0.4)
MirrorWiC			-	-	-	71.9	-	-

Table 4: Results on the development set of the WiC task. **LR** is learning rate, **E** - number of epochs, τ - temperature parameter of the loss function, **N comp.** - number of components for PCA. Reported metric is accuracy, all values are an average of 5 runs (std. dev. in parentheses). First two lines are baseline results before fine-tuning.

Model	Layer	α_2	#pLU	#C	PiF_1	BcF_1	PiF	BcF
B	11/2	0.6	266	141	96.6	95.9	76.3	70.3
B+P	10/2	0.5	275	144	96.9	96.1	75.4	69.3
B+FT	11/2	0.1	300	171	**97.2**	**96.4**	**80.7**	**75.4**
B+FT+P	11/2	0.2	294	163	97.2	96.4	80.3	74.8

Table 5: Results on the frame induction development set. Model name corresponds to **B** - bert-base-uncased, **P** - application of PCA (reduction to 100 components with whitening), **FT** - the fine-tuned version of the BERT model. The layer column indicates which BERT layer was used: left value stands for the first step clustering layer, right value is the second step clustering layer. First step clustering algorithm is always X-Means, second step - Group Average. α_2 is the weight of the masked embedding for the second step. #pLU is the number of pseudo-lexical units after the first step clustering, #C is the number of clusters after the second step. Actual number of LUs is 300, actual number of frames is 169. Every FT=+ row reports an average of 5 runs.