Waste Not: Meta-Embedding of Word and Context Vectors

Selin Değirmenci, Aydın Gerek, and Murat Can Ganiz

Marmara University, İstanbul 34730, TR
selindegirmenci@marun.edu.tr, {aydin.gerek,murat.ganiz}@marmara.edu.tr

Abstract. The word2vec and fastText models train two vectors per word: a word and a context vector. Typically the context vectors are discarded after training, even though they may contain useful information for different NLP tasks. Therefore we combine word and context vectors in the framework of meta-embeddings. Our experiments show performance increases at several NLP tasks such as text classification, semantic similarity, and analogy. In conclusion, this approach can be used to increase performance at downstream tasks while requiring minimal additional computational resources.

Keywords: Meta-Embedding · Word Embeddings · Word2vec · FastText · Text Classification · Semantic Similarity · Analogy.

1 Introduction and Motivation

The choice of word embedding model is an important hyperparameter for many NLP tasks, since it has been observed that different embedding models tend to provide stronger representations for different types of downstream tasks [4]. It is also known that ensembles of machine learning models tend to perform better than their individual constituents. It makes sense, then, to combine different embedding models in order to improve the downstream NLP tasks' performance.

While using ensembles of downstream models seeded with different types of word embeddings had been tried before [1], the idea of combining word embeddings directly to form meta-embeddings starts with the work of [23]. In that work the authors form meta-embeddings by concatenation, by factorization of the concatenated vectors (SVD), and a method called 1toN that learns a meta-embedding from which (also learned) projections exist to the source embeddings, with said projections minimizing the mean square error between the projected meta-embedding and source embedding of the same word for all words. A simpler but overlooked idea of averaging source embeddings is explored in [5]. In [3] autoencoders are employed to dimension reduce the concatenated (CAEME) and averaged (AAEME) meta-embeddings as well as dimension reducing source embeddings and concatenating them (DAEME). Finally in [9] source embeddings are dynamically combined using a self-attention mechanism [11].

One of the best known word embedding models is word2vec [15, 16], which during its learning procedure not only learns a word vector for each word in the training corpus, but also a context vector for it. However context vectors are typically discarded after training. In [18] it is briefly mentioned that adding word and context vectors may result in a small performance boost. However it is not thoroughly investigated. In this study we investigate it in detail by forming meta-embeddings of word and context vectors in several different ways and conducting detailed experiments. We observe that combining the word and context embeddings to form a meta-embedding in several different settings yields a higher performance at the text classification, semantic similarity and analogy tasks.

In section 2, we describe our novel approach and meta-embedding types. In section 3, we describe our experimental setup, our implementation and NLP tasks that we perform. In section 4, we observe results on text classification, semantic similarity and word analogy and compare our meta-embeddings with others. In section 5, we describe our conclusions and possible future works according to the results.

2 Approach

Our novel approach focuses on exploiting otherwise ignored information encoded in context vectors. We formulate and experiment with seven different types of meta-embeddings. A total of nine results are given in our tables for comparison where the first two are traditional word and context embeddings which constitute the baselines.

In order to see if including context vectors help improving performance in several NLP tasks, first, we create a meta-embedding by concatenating word and its context embeddings which is simply named as "concat". This will result in doubling the dimension size. Our second approach is to average word and context embeddings which is named as "average". Third approach is to apply a max pooling filter to word and context embeddings to create a meta-embedding. This is donated as "maxpool(word,context)". Fourth one is a more complicated meta-embedding which is obtained by concatenation, averaging and maxpooling of word and context embeddings. This is indicated as "cam" in our result tables. Following this, we have three additional auto-encoder based meta-embeddings [3] of word and context embeddings, namely Averaged Autoencoded Meta-Embedding (AAEME), Concatenated Autoencoded Meta-Embedding (CAEME), and Decoupled Autoencoded Meta-Embedding (DAEME). Please note that different meta-embedding approaches we have taken result in different dimensional vectors. This can be seen in Table 1.

In order to obtain word and context embeddings we use two of the most popular word embedding models; the word2vec and fastText [2, 8]. For both models we use the skip-gram negative sampling architecture as it is more popular. In the case of fastText models while character n-grams are trained alongside word and context embeddings, we've choose not to include those in our meta-embeddings for comparability reasons.

Table 1: Embedding and Dimension

Embedding	Dimension
word	200
context	200
concat	400
average	200
maxpool(word,context)	200
cam	400
AAEME	300
CAEME	400
DAEME	400

3 Experiments

3.1 Datasets

We trained our embeddings on Text8 [13], which is a corpus based on the the first 10^9 bytes of the Wikipedia dump of March 3, 2006. As a next step of experiment, we trained our embeddings on a large Wikipedia dump. This corpus contains 19,251,790 articles and occupies approximately 16GB of disk space.

For the text classification task we use the following datasets: AG's News Corpus [24] consisting of $120,000$ documents in 4 classes, WEBKB which is a highly imbalanced dataset of 8,282 documents in 7 classes [14, 19], Yelp Reviews Polarity [22, 24] consisting of 560,000 documents in 2 classes, and DBPedia [10, 24] also consisting of 560,000 documents but in 14 classes.

For the semantic similarity test we use the following datasets: WS [6] (353 word pairs), RG [21] (65 word pairs), RW [12] (2034 word pairs), SL [7] (999 word pairs).

For the analogy test we use the GL [16] dataset (19557 analogy questions).

3.2 Experimental Setup

We use the gensim library [20] implementations of word2vec and fastText. For both we train vectors dimension of 200, and use default hyperparameters otherwise. We also use the word similarity and analogy tests implemented in the gensim library. We report the Spearman Correlation between the cosine similarity of word vectors and human assigned similarity scores.

For the text classification experiments we use Support Vector Machines (SVM) algorithm, more specifically Linear Support Vector Classifier (LinearSVC) which is commonly used in this domain. We use the one implemented in the scikit-learn library [17] with the default hyper parameters. Documents to be classified are represented as averages of their words' vectors.

The text classification experiments were run with 10-fold cross validation. We report the average accuracy and the standard deviations for the classification experiments.

In text classification experiments, in order to see if the performance improvement of meta-embeddings such as "concat" is due to the increased (actually doubled) number of dimensions or not, we conduct two sets of experiments.

First, we compare meta-embeddings of size 200 (100 word + 100 context) with a baseline word embedding vectors of 200. In the second set of similar experiments we double the vector sizes.

4 Results and Discussion

4.1 Text Classification

According to our results, as seen in tables 2 and 3, we see that for text classification the concatenation approach has a distinct advantage over all other approaches. The auto-encoder meta-embeddings appear to perform better than the average and the baseline meta-embeddings. However for the WEBKB dataset, which is a highly class imbalanced one, we observe a different pattern. In this dataset autoencoder based meta-embeddings perform poorly compared to others.

The concatenation meta-embeddings of both word2vec and fastText models exceed the classification performance of the other meta-embeddings in all datasets.

Table 2: Performance of word2vec meta-embeddings trained on text8 for text classification task

	AG News	WEBKB	Yelp Polarity	DBpedia
word	85.41 +/- 1.11	67.61 +/- 2.07	79.51 +/- 0.78	94.78 +/- 1.49
context	86.26 +/- 1.12	67.11 +/-2.45	81.29 +/- 0.79	94.75 +/- 1.52
concat	**87.42** +/- 1.07	69.56 +/- 2.27	**83.31** +/- 0.80	96.02 +/- 1.21
average	85.68 +/- 1.14	67.40 +/- 2.14	79.79 +/- 0.74	94.87 +/- 1.47
maxpool(word,context)	85.51 +/- 1.07	66.97 +/- 2.17	79.33 +/- 1.00	94.63 +/- 1.49
cam	86.79 +/- 1.06	**69.79** +/- 1.88	82.44 +/- 0.89	**96.05** +/- 1.20
AAEME	86.71 +/- 1.21	58.53 +/- 2.72	82.21 +/- 0.73	94.75 +/- 1.58
CAEME	86.90 +/- 1.15	59.14 +/- 2.72	82.96 +/- 0.75	94.99 +/- 1.54
DAEME	86.33 +/- 1.12	56.79 +/- 2.92	82.19 +/- 0.77	94.56 +/- 1.59

Table 3: Performance of fastText meta-embeddings trained on text8 for text classification task

	AG News	WEBKB	Yelp Polarity	DBpedia
word	85.32 +/- 1.16	68.08 +/- 2.37	79.78 +/- 0.74	94.29 +/- 1.75
context	86.61 +/- 1.23	67.51 +/-2.32	81.26 +/- 0.84	94.79 +/- 1.54
concat	**87.49** +/- 1.17	70.13 +/- 2.30	**84.05** +/- 0.88	**95.98** +/- 1.25
average	85.77 +/- 1.16	68.23 +/- 2.23	80.16 +/- 0.79	94.51 +/- 1.66
maxpool(word,context)	85.21 +/-1.28	67.21 +/- 2.69	79.43 +/- 0.67	94.79 +/- 1.86
cam	86.86 +/- 1.08	**70.85** +/-2.60	83.05+/- 0.80	95.72 +/- 1.31
AAEME	86.75 +/- 1.17	59.72 +/- 2.70	83.05 +/- 0.88	94.75 +/- 1.72
CAEME	87.07 +/- 1.20	60.31 +/- 2.74	83.86 +/- 0.83	95.15 +/- 1.59
DAEME	86.61 +/- 1.26	58.25 +/- 2.49	83.08 +/- 0.86	94.63 +/- 1.73

The improvement is most obvious in the Yelp Reviews Polarity dataset with an increase of 3.8 percentage points over word embeddings for word2vec, and 4.27 percentage points for fastText.

As the second step of experiments we run the same text classification tasks using our meta-embedding models trained using Wikipedia. According to results, as seen in table 4 and 5, again concatenation meta-embeddings of both word2vec and fastText models exceed the classification performance of the other meta-embeddings in all datasets.

Table 4: Performance of word2vec meta-embeddings trained on Wikipedia for text classification task

	AG News	WEBKB	Yelp Polarity	DBpedia
word	88.82 +/- 1.11	68.31 +/- 3.02	84.06 +/- 0.73	96.53 +/- 1.20
context	88.98 +/- 1.21	67.96 +/- 3.30	84.71 +/-0.70	96.56 +/- 1.20
concat	**89.42** +/- 1.10	**69.89** +/- 3.36	86.07 +/- 0.67	96.90 +/- 1.11
average	88.98 +/- 1.16	68.13 +/- 3.19	84.50 +/- 0.69	96.53 +/-1.18
maxpool(word,context)	88.87 +/- 1.21	68.23 +/- 3.10	84.09 +/- 0.73	96.40 +/- 1.22
cam	89.38 +/- 1.14	69.04 +/-3.64	**86.23** +/- 0.68	**96.97** +/- 1.11

Table 5: Performance of fastText meta-embeddings trained on wikipedia for text classification task

	AG News	WEBKB	Yelp Polarity	DBpedia
word	88.04 +/-1.35	67.30 +/- 1.97	82.12 +/- 1.47	96.32 +/-1.18
context	88.56 +/-1.13	66.48+/ -2.18	84.77 +/ -0.74	96.35 +/- 1.26
concat	**88.76** +/-1.10	67.97 +/- 2.11	**85.34** +/- 1.01	96.83 +/- 1.07
average	88.23 +/- 1.29	68.17 +/- 1.84	83.37+/- 0.78	96.38 +/- 1.19
maxpool(word,context)	87.61 +/- 1.54	67.05 +/-4.01	81.79 +/-1.11	95.98 +/- 1.39
cam	88.33 +/-1.21	**69.47** +/- 2.24	83.17 +/- 2.79	**96.69** +/- 1.20

For text classification, as seen in tables 6, 7, 8 and 9, we also compare the performance of same size meta-embedding and baseline embedding vectors. We observe that concatenation of word and context vector still shows higher accuracy than word embedding, even though the dimension of embeddings are same.

Table 6: Performance comparison of word2vec meta-embedding "concat" with same vector size baseline for text classification task

	AG News	WEBKB	Yelp Polarity	DBpedia
concat(200d word,200d context)	**87.42** +/- 1.07	**69.56** +/- 2.27	**83.31** +/- 0.80	**96.02** +/- 1.21
400d word	86.26 +/- 1.03	68.46 +/- 1.82	81.32 +/- 0.73	95.67 +/-1.24

Table 7: Performance comparison of fastText meta-embedding "concat" with same vector size baseline for text classification task

	AG News	WEBKB	Yelp Polarity	DBpedia
concat(200d word,200d context)	**87.49** +/- 1.17	**70.13** +/- 2.30	**84.05** +/- 0.88	**95.98** +/- 1.25
400d word	86.30 +/- 1.07	69.59 +/-2.66	82.42 +/- 0.87	95.46 +/- 1.39

4.2 Semantic Similarity and Word Analogy

Word2vec meta-embedding semantic similarity and analogy results that can be seen in table 10 for three of the five datasets the average meta-embedding

Table 8: Performance comparison of word2vec meta-embedding "concat" with same vector size baseline for text classification task

	AG News	WEBKB	Yelp Polarity	DBpedia
concat(100d word,100d context)	**86.07** +/- 1.24	67.50 +/- 2.55	**81.17** +/- 0.71	**94.79** +/- 1.51
200d word	85.41 +/- 1.11	**67.61** +/- 2.07	79.51 +/- 0.78	94.78 +/- 1.49

Table 9: Performance comparison of fastText meta-embedding "concat" with same vector size baseline for text classification task

	AG News	WEBKB	Yelp Polarity	DBpedia
concat(100d word,100d context)	**86.54** +/- 1.22	**68.25** +/- 2.29	**80.64** +/- 0.73	**94.49** +/- 1.72
200d word	85.32 +/- 1.16	68.08 +/- 2.37	79.78 +/- 0.74	94.29 +/- 1.75

performs better. For the RW dataset auto-encoder based meta-embedding DAEME slightly outperforms the average meta-embedding. Interestingly other auto-encoder based meta-embeddings under perform the average meta-embedding in the same dataset. One outlier in the semantic similarity task is the SL dataset. In this one concatenation outperforms all other methods by a large margin.

In the fastText meta-embedding semantic similarity and analogy results which can be seen in table 11. We observe a pattern differing from its word2vec counterparts. We see a much better picture for auto-encoder based meta-embedding methods. For four of the five datasets the auto-encoder base meta-embeddings outperform all others visibly. The only dataset where they do not is the RG dataset which is the smallest dataset used in the semantic similarity task. In this dataset the average meta-embeddings perform better.

Of note is the fact that for every dataset except the SL dataset, the average meta-embeddings outperform the concatenation meta-embeddings at the semantic similarity task, and at the analogy task as well.

Table 10: Performance of word2vec meta-embeddings trained on text8 for semantic similarity and analogy tasks

	WS	RG	RW	SL	GL
word	0.622	0.504	0.328	0.261	24.4
context	0.445	0.320	0.336	0.258	18.0
concat	0.625	0.487	0.347	**0.336**	24.4
average	**0.642**	**0.525**	0.367	0.269	**27.1**
maxpool(word,context)	0.608	0.441	0.373	0.262	20.5
cam	0.629	0.480	0.378	0.264	24.2
AAEME	0.593	0.441	0.348	0.285	25.4
CAEME	0.576	0.409	0.353	0.279	25.8
DAEME	0.598	0.444	**0.375**	0.270	24.9

We also see a difference in the performance of fastText word and context embeddings. For instance in the analogy task the context embeddings only solve 13.8 percent of the analogy questions, whereas the word vectors manage 40.6 percent. In the SL dataset for the semantic similarity task the context vectors significantly outperform word vectors (Spearman correlation of 0.3 versus 0.242). This should be due to the difference in the training of fastText and word2vec vectors. Namely in the word2vec model the similarity score is calculated as a function of the dot product between word and context vectors, whereas in

Table 11: Performance of fastText meta-embeddings trained on text8 for semantic similarity and analogy tasks

	WS	RG	RW	SL	GL
word	0.435	0.377	0.305	0.242	40.6
context	0.393	0.352	0.304	0.300	13.8
concat	0.437	0.365	0.309	0.251	41.1
average	0.473	**0.414**	0.328	0.254	42.1
maxpool(word,context)	0.425	0.418	0.320	0.220	36.1
cam	0.451	0.430	0.328	0.239	39.7
AAEME	**0.475**	0.398	0.345	0.316	**44.5**
CAEME	0.471	0.389	0.345	**0.320**	44.1
DAEME	0.427	0.397	**0.349**	0.317	39.2

the fastText model word and character n-gram embeddings are summed before computing a dot product with the context vectors.

Thus while word and context vectors are symmetric in the word2vec model,they are not in the fastText model. We suspect the differences in performance are due to this fundamental asymmetry.

5 Conclusions and Future Work

By combining word and context vectors of word2vec and fastText models using several different meta-embedding approaches we evaluate how much improvement context vectors can provide to word vectors' performances in downstream NLP tasks such as text classification, semantic similarity, and analogy. Furthermore we investigate which meta-embedding approaches are better at these tasks.

We show that even we use much larger training corpus for embedding models, resulting meta-embeddings show similar behavior, the concatenation of word and context embeddings usually leads to higher accuracy in text classification task.

It is interesting to note that just as the performances of word embedding models differ according to task, so do those of meta-embeddings of word and context vectors. In particular concatenation meta-embeddings perform better at text classification tasks, and average meta-embeddings tend to perform better at semantic similarity and analogy tasks.

We plan to combine word and context embeddings using a greater variety of meta-embedding methods. Namely, we think that the averaging method will perform better if the word and context embeddings are aligned via an orthogonal transformation first. We would also like to evaluate the 1toN [23] in this context. Another interesting approach will be inclusion of character n-gram embeddings of fastText in the various combinations.

In the future we would like to shed some light onto performance differences of auto-encoder based meta-embeddings by throughout analysis.

References

1. Bansal, M., Gimpel, K., Livescu, K.: Tailoring continuous word representations for dependency parsing. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). vol. 2, pp. 809–815 (2014)

2. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017), http://aclweb.org/anthology/Q17-1010
3. Bollegala, D., Bao, C.: Learning word meta-embeddings by autoencoding. In: Proceedings of the 27th International Conference on Computational Linguistics. pp. 1650–1661 (2018)
4. Chen, Y., Perozzi, B., Al-Rfou, R., Skiena, S.: The expressive power of word embeddings. In: ICML 2013 Workshop on Deep Learning for Audio, Speech, and Language Processing. Atlanta, GA, USA (July 2013), https://sites.google.com/site/deeplearningicml2013/TheExpressive-PowerOfWordEmbeddings.pdf
5. Coates, J., Bollegala, D.: Frustratingly easy meta-embedding–computing meta-embeddings by averaging source word embeddings. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). vol. 2, pp. 194–198 (2018)
6. Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin, E.: Placing search in context: The concept revisited. ACM Transactions on information systems **20**(1), 116–131 (2002)
7. Hill, F., Reichart, R., Korhonen, A.: Simlex-999: Evaluating semantic models with (genuine) similarity estimation. Computational Linguistics **41**(4), 665–695 (2015)
8. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers. pp. 427–431. Association for Computational Linguistics (2017), http://aclweb.org/anthology/E17-2068
9. Kiela, D., Wang, C., Cho, K.: Dynamic meta-embeddings for improved sentence representations. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. pp. 1466–1477 (2018)
10. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al.: Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web **6**(2), 167–195 (2015)
11. Lin, Z., Feng, M., dos Santos, C.N., Yu, M., Xiang, B., Zhou, B., Bengio, Y.: A structured self-attentive sentence embedding (2017)
12. Luong, T., Socher, R., Manning, C.: Better word representations with recursive neural networks for morphology. In: Proceedings of the Seventeenth Conference on Computational Natural Language Learning. pp. 104–113 (2013)
13. Mahoney, M.: About the test data (2011), http://mattmahoney.net/dc/textdata
14. McCallum, A., Nigam, K., et al.: A comparison of event models for naive bayes text classification. In: AAAI-98 workshop on learning for text categorization. vol. 752, pp. 41–48. Citeseer (1998)
15. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR **abs/1301.3781** (2013), http://arxiv.org/abs/1301.3781
16. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. pp. 3111–3119. NIPS'13, Curran Associates Inc., USA (2013), http://dl.acm.org/citation.cfm?id=2999792.2999959

17. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. Journal of machine learning research **12**(Oct), 2825–2830 (2011)
18. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
19. Poyraz, M., Kilimci, Z.H., Ganiz, M.C.: Higher-order smoothing: a novel semantic smoothing method for text classification. Journal of Computer Science and Technology **29**(3), 376–391 (2014)
20. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010), http://is.muni.cz/publication/884893/en
21. Rubenstein, H., Goodenough, J.B.: Contextual correlates of synonymy. Communications of the ACM **8**(10), 627–633 (1965)
22. Yelp: Yelp reviews dataset challenge (2015), https://www.yelp.com/dataset/challenge
23. Yin, W., Schütze, H.: Learning word meta-embeddings. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 1351–1360 (2016)
24. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Advances in neural information processing systems. pp. 649–657 (2015)