ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

Cross-Encoded Meta Embedding towards Transfer Learning

Rickard Brännvall[12] Johan Öhman[3] György Kovács[1]
Marcus Liwicki[1]

[1] *EISLAB Machine Learning* - Luleå University of Technology, Sweden
[2] *RISE ICE* - Research Institutes of Sweden, Sweden
[3] *Experimental Mechanics* - Luleå University of Technology, Sweden

Abstract.
In this paper we generate word meta-embeddings from already existing embeddings using cross-encoding. Previous approaches can only work with words that exist in each source embedding, while the architecture presented here drops this requirement. We demonstrate the method using two pre-trained embeddings, namely GloVE and FastText. Furthermore, we propose additional improvements to the training process of the meta-embedding. Results on six standard tests for word similarity show that the meta-embedding trained outperforms the original embeddings. Moreover, this performance can be further increased with the proposed improvements, resulting in a competitive performance with those reported earlier.

1 Introduction

Word embeddings (mappings of words from a natural language to a real vector space – e.g. FastText [1], GloVE [2]) have become crucial in Natural Language Processing (NLP) for a variety of tasks, including machine translation, sentiment analysis, and response generation [3]. Meta-embeddings (created by combining preexisting embeddings), have shown increased performance on some tasks [4, 5, 6, 7]. The motivation for creating meta-embeddings is that different embeddings encode words differently and thus in combination can enrich each other. Factors accounting for these differences may include the data, architecture, or loss function used in training embeddings. The performance of word embeddings is commonly compared to human ratings on NLP tasks. In this paper (following O'Neill and Bollegala [5]), we consider word similarities, by comparing the correlation between embeddings of a word pair to human produced similarity scores, using well established benchmarks.

For learning meta-embeddings we propose a cross-encoder that can operate in auto-encoder mode, encoding from an original embedding to the meta-embedding, then decoding back into an estimate of the original embedding (e.g. from GloVE to the meta-embedding, and back to GloVE). It can also encode from an original embedding and decode into an estimate of a different embedding (e.g. from FastText to the meta-embedding, and to GloVE). Some additional features are also implemented, namely a trainable weighting in the combiner step of the cross-encoder, adaptive size limiting the bandwidth for infrequent words, and compensation for the word frequency (se e.g. [8]).

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

Related Work Different approaches have been explored to generate meta-embeddings. Coates and Bollegala investigated concatenating/averaging original embeddings, and found that both leads to similar improvements over original embeddings [6]. Yin and Schütze trained a meta-embedding by finding the optimal projections from several original embeddings, and then combining these projections [7]. Bao and Bollegala created meta-embeddings by auto-encoders, investigating different architectures and their impact on performance [4]. As a continuation, O'Neill and Bollegala investigated how different loss metrics affect auto-encoder performance [5]. The above methods can only work with words that are present in all original embeddings, while our proposed cross-encoder architecture can discover meta-embeddings for words that exist only in one of the original embeddings. This facilitates the combination of embeddings trained on materially different corpora, opening up transfer learning opportunities.

2 Experimental Setting

Given two source embeddings $\mathrm{X}^{(0)}$ and $\mathrm{X}^{(1)}$, each a mapping between a set of words a real-valued vector space, the cross-encoder architecture is used to find the combined meta-embedding Z for the union both word sets. The architecture can generally be applied to arbitrarily many source embeddings, but for this paper we exemplify it for the case of two. For a word w_i that exists in the original embedding k, the source embedding $X_i^{(k)}$ corresponding to that word is passed through encoder $E^{(k)}$:

$$Z_i^{(k)} = E^{(k)}(X_i^{(k)}) \tag{1}$$

where $E^{(k)}$ is an Artificial Neural Network (ANN). A meta-embedding Z_i is then produced for the word based on $Z_i^{(k)}$. The precise details of this depend on whether the word exists in one or more of the source embeddings (as detailed below). The decoder step then predicts the initial source embedding(s):

$$Y_i^{(k)} = D^{(k)}(Z_i) \tag{2}$$

Other encoder-decoder branches follow the same design (using different weights).
Auto-encoder mode: for a word that only exists in one original embedding k, the source embedding $X_i^{(k)}$ is passed through the auto-encoder $E^{(k)}$ according to equation (1). The meta-embedding is simply set to $Z_i = Z_i^{(k)}$, and is passed directly to the decoder module (2), which predicts the initial source embedding.
Cross-encoder mode: for words that exist in both source embeddings, the corresponding embeddings $X_i^{(0)}$ and $X_i^{(1)}$ are fed in parallel to the cross-encoder, each taking a separate branch. After the encoding stage we have preliminary branch-encodings for each branch $Z_i^{(0)}$ and $Z_i^{(1)}$. The cross-encoder meta-embedding is produced by combing the two preliminary branch-encodings such as $Z_i = C(Z_i^{(0)}, Z_i^{(1)}, f)$ by a combiner module defined as

$$C(z^{(0)}, z^{(1)}, f) = (1 - f) \odot z^{(0)} + f \odot z^{(1)} \tag{3}$$

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

where $\odot$ indicates Hadamard product, and f is a vector (the same length as the meta-embeddings) that in the base configuration includes only 0.5 entries. Z_i is passed to both decoder branches to predict both original source embeddings.

Loss function: is simply summing over the normed prediction error, which itself is defined as the difference between the original source embedding and the predicted source embedding for each word. We considered both the L_2-norm and the squared cosine loss (SCL), but based on our preliminary experiments, decided on the former:

$$l_{L2} = \sum_{i \in \mathcal{I}_{\text{all}}} \|x_i - y_i\|_2^2 \, \omega_i \tag{4}$$

Note that each individual loss is weighted by ω_i which is specific to word w_i, as defined below

Word frequency weighting: words do not occur with the same frequency in a language, instead some words are orders of magnitude more common than others. Zipf's law describes this based on an observed power law in empirical data (see e.g. [8] for a review). When training an embedding from a real corpus the relative frequencies will be approximately respected as the methods are based on sampling from the texts. However, a meta-embedding is trained on data from pre-trained embeddings where each word occurs at most once in every source set. To better reflect the natural word frequencies also when training meta-embeddings, we introduce a power-law function based on rank (i.e. the order the words occurred in the original source corpus) that up-weights common words relative to uncommon words

$$\omega_i \propto r_i^{-\xi} \tag{5}$$

where r_i is the rank of the word w_i in the combined embedding set and ξ is a non-negative power law coefficient.

Trainable combiner weights: an extension of the model lets the encoder weighting f in Eq. (3) be learnt when training the model, for example implemented as a sigmoid element-wise transformation of a constant vector. More complex models could have this be a function of the intermediate embeddings on each path, $Z_i^{(k)}$, or for the case of more than three source embeddings, the signature of which of the these that has the word w_i among its supported words.

Adaptive embedding size: overfitting is a well known and frequently occurring problem when data is sparse and the high capacity of a model is exploited by the optimizer to achieve good scores on training set at the cost of poor out-of-sample performance. The embeddings of infrequent words in the original corpus could then be troubled by overfitting as very few contexts were available to determine their precise vector representation. The word-frequency weighting discussed above goes some length to prevent noise from rare words to swamp the signal for frequent words, however as a further mitigating step we want to explore adaptive embedding length introduced by Baevski and Auli [9] where the capacity is assigned according to word-frequency.

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

The encoder step is now split into the following sub-steps, where each of the sub-encodings $Z_i^{(k,1)}$ and $Z_i^{(k,2)}$ are of length $d/2$ for $Z_i^{(k)}$ of length d.

$$Z_i^{(k,1)} = E^{(k,1)}(X_i^{(k)}) \tag{6}$$

$$m_i^{(k)} = \mathbb{1}(r_i^{(k)} > r^*) \tag{7}$$

$$Z_i^{(k,2)} = Z_i^{(k,1)} + m_i^{(k)} E^{(k,2)}(X_i^{(k)}) \tag{8}$$

The sub-encodings are then concatenated to produce the branch encoding $Z_i^{(k)} = (Z_i^{(k,1)}, Z_i^{(k,2)})$. For an infrequent word the rank will be below the threshold r^* and the result will just be the concatenation of two identical vectors, however more frequent words can explore the second sub-embedding layer to add complexity to the representation on top of the residual connection.

This approach could of course be extended to having three or more sub-embeddings (with residual connections) that be concatenated in a similar manner as just described above for the case of only two such layers. Note that the complexity (in terms of trainable parameters) can decrease substantially with the number of splits for sub-embedding modules based on standard fully connected neural-network cells more than one level deep.

Datasets: two pretrained word embeddings are used to generate the meta-embedding. The first is the Glove embedding pretrained on Wikipedia 2014 and the GigaWord 5 datasets [2]. The second pretrained embedding is FastText pretrained on the Wikipedia 2017 and the statmt.org news datasets [10]. Both original embeddings are vectors in R^{300}. The top 100000 words from each dataset are used for training of the cross-encoder.

Training Procedure: the model is trained using the popular Adam optimiser for between 50 and 500 epochs. From all the embedding data, a random sample of 20% is set aside as validation set. Loss score at each epoch during training is recorded for both training and validation sets. Hyperparameter selection is based on the validation score using gridsearch on a predefined grid. Based on this the optimal architecture choice has only a single hidden layer, uses batch-normalization for regularization, doesn't apply early stopping and has a self-adaptive size of 1000 (where applicable).

3 Results

We adapted the word similarities test described by O'Neill and Bollegala [5]. The test is performed on six different word similarity datasets (Simlex [11], WordSim-353 [12], RG [13], MechanicalTurk-771 [14], RareWord [15] and MEN [16]) using the GenSim [17] software package. These data sets consist of word pairs and associated correlation scores. The test then correlates the score with the score obtained from the embeddings. The spearman correlation-coefficient for both the original embeddings and the cross-encoder is shown in Table 1. The best performing auto-encoder from [5] is also presented for reference.

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

Model	Wordsim	SimLex	MTurk	MEN	RW	RG	avg score
FastText	0.6782	0.4330	0.6847	0.6842	0.5315	0.7575	0.6282
GLoVe	0.6752	0.3692	0.6501	0.7485	0.4275	0.7693	0.6066
CrossCoder	0.6812	0.3895	0.6740	0.7545	0.4761	0.8057	0.6302
+ ZipfLaw	0.7118	0.4314	0.7141	0.8038	0.5532	0.8187	0.6721
+ TrainComb	0.6621	0.3753	0.6495	0.7400	0.4755	0.7770	0.6132
+ AdaptSize	0.6786	0.3747	0.6534	0.7715	0.4729	0.8242	0.6292
+ All	0.7214	0.4562	0.7165	0.7928	0.5423	0.8588	0.6813
Reference	0.7244	0.4485	0.7063	0.8194	0.5074	0.8541	0.6767

Table 1: Comparison of results on six standard word similarity tests.

Additionally it would be of interest to see if the local structure from the original embeddings are preserved. It is desirable that nearby words in the original embedding also are nearby in the meta embedding. To investigate this, a test is performed where, for each word in the meta-embedding it checks if the closest word is in the top-1, top-3 and top-5 of nearby words in the original embeddings. For all words in the meta-embedding the most similar word is also the most similar word in the Glove embedding. For the FastText embedding the top-1 scores are ranging between $94\% - 98\%$ for the different implementations of the meta-embedding. The top-5 score is above 99% for all cases.

4 Analysis

Comparing the results in Table 1 we find that the basic cross-encoder outperforms both original embeddings on total score, and at least one of the original embeddings on each separate test. Regarding further modifications, we see the largest performance boost when adding Zipf's law for weighting the training data. This is expected since the meta-embeddings trained on the basic cross-encoder lose the information about word frequency and give too high weight to rare words. The two other modifications, trainable combiner weights and adaptable size, had little or adverse affects on the performance when applied individually.

The cross-encoder with all modifications applied outperforms or performs on par with the reference model on the individual tests. One should be aware that we did not have access to the exact implementation of the tests for the reference meta-embedding. Therefore the numbers should not be compared strictly and overall it is hard to say if one performance better than the other.

5 Conclusions and Future Work

With the proposed meta-embedding it was possible to increase the performance compared to that of the original embeddings on word similarity tests. Furthermore the meta-embedding proposed here outperformed previously reported results for meta-embeddings on some tests (albeit with only modest improvements). The importance of weighting the training data according to Zipf's law

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

is demonstrated and should be applied when creating meta-embeddings from already existing embeddings. The introduction of trainable combiner weights and adaptable size modifications also improved the performance when used in combination with Zipf's weighting.

The original embeddings used for training the meta-embedding are trained on similar kind of data sets. Both are, for instance, trained on some version of the Wikipedia corpus. For future work it would therefore be of interest to see if the performance of meta-embeddings is improved when the original embeddings are trained on corpora from different contexts. Alternatives to the word similarities test could also be investigated as it might not be best equipped to measure the transfer learning capacity of the cross-encoder.

References

[1] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[2] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proc. EMNLP*, pages 1532–1543, 2014.

[3] Yoav Goldberg. A primer on neural network models for natural language processing. *CoRR*, abs/1510.00726, 2015.

[4] Danushka Bollegala and Cong Bao. Learning word meta-embeddings by autoencoding. In *Proc. COLING*, pages 1650–1661, 2018.

[5] James O'Neill and Danushka Bollegala. Angular-based word meta-embedding learning. *CoRR*, abs/1808.04334, 2018.

[6] Joshua Coates and Danushka Bollegala. Frustratingly easy meta-embedding - computing meta-embeddings by averaging source word embeddings. *CoRR*, abs/1804.05262, 2018.

[7] Wenpeng Yin and Hinrich Schütze. Learning word meta-embeddings by using ensembles of embedding sets. *CoRR*, abs/1508.04257, 2015.

[8] David M. W. Powers. Applications and explanations of zipf's law. In *Proc. NeM-LaP3/CoNLL*, pages 151–160, 1998.

[9] Alexei Baevski and Michael Auli. Adaptove input representation for neural language modeling. *arXiv preprint arXiv:1809.10853v3*, 2019.

[10] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proc. LREC*, 2018.

[11] Felix Hill, Kyunghyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. Embedding word similarity with neural machine translation. *arXiv preprint arXiv:1412.6448*, 2014.

[12] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131, January 2002.

[13] Herbert Rubenstein and John B. Goodenough. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633, October 1965.

[14] Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. Large-scale learning of word relatedness with constraints. In *Proc. ACM SIGKDD*, pages 1406–1414, 2012.

[15] Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. In *Proc. ACL-IJCNLP*, pages 11–19, 2015.

[16] Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. Distributional semantics in technicolor. In *Proc. ACL*, pages 136–145, 2012.

[17] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proc. LREC*, pages 45–50, 2010.