How to represent a word and predict it, too: Improving tied architectures for language modelling

Kristina Gulordava **Laura Aina** **Gemma Boleda**
Universitat Pompeu Fabra
Barcelona, Spain
{firstname.lastname}@upf.edu

Abstract

Recent state-of-the-art neural language models share the representations of words given by the input and output mappings. We propose a simple modification to these architectures that decouples the hidden state from the word embedding prediction. Our architecture leads to comparable or better results compared to previous tied models and models without tying, with a much smaller number of parameters. We also extend our proposal to word2vec models, showing that tying is appropriate for general word prediction tasks.

1 Introduction

In neural models, reusing representations of the same type of data (e.g., sentences or words) in different parts of the architecture can be a powerful way to aid learning: it reduces parameters, enabling more compact models and faster learning. Recurrent neural network (RNN) language models (Mikolov et al., 2010; Sundermeyer et al., 2012; Zaremba et al., 2014) have two word mappings: From the input word onto its embedding representation, and from the internal representation of the network (the hidden layer) to the weights for the prediction of the next word. In standard models, these representations are different. Recently, Inan et al. (2017) and Press and Wolf (2017) proposed to instead use a single word representation, *tying* the input and output mappings. Intuitively, both are representations of the same type of data (words), and information learnt when observing a word as input can be reused when predicting this word as output. Tied language models obtained better perplexity and better word similarity scores of embedding matrices while reducing the number of parameters. The models that achieve the latest state-of-the art results incorporate this technique (see, e.g., Merity et al., 2018).

However, note that, by tying the output mapping to the input mapping, the hidden layer of the network is optimised to match the representation of the predicted word. We suggest that this introduces a constraint that conflicts with the function of the hidden layer in language models to represent the previous context and transmit information to the next timestep. In this paper, we propose a minimal modification to tied LM architectures to address this issue: we add a linear transformation between the hidden layer and the word embedding prediction, partially decoupling the two. This has an important advantage. Standard tied architectures require the hidden layer to have the same dimensionality as the word embeddings. We lift this constraint in our architecture: by separating the hidden layer and the word mapping, we can choose a large hidden layer dimensionality while keeping the embedding dimensionality and, consequently, the size of the embedding matrix small.

In a set of experiments on LM, we show that our tied models achieve results similar to or better than models with standard or no tying, with much smaller embedding sizes and a reduction of 30-60% in the overall number of parameters. Notably, the word embeddings obtained with the modified model have a higher quality than double-sized embeddings obtained with standard tied models, as measured on word similarity.

We further extend this idea to word representation learning models (in particular, word2vec), which have a similar architecture and objective function to language models. For instance, the standard skipgram model (Mikolov et al., 2013) has two mappings, one for the context words and one for the target word. While tying these two matrices directly constraints learning too strongly, an additional linear mapping adds the sufficient capacity to learn embeddings of the same quality as the standard model using only half the parameters.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2936–2941
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

2 Tied LM architectures

2.1 Previous work

Equations (1) through (4) define a standard RNN language model (Mikolov et al., 2010):

$$\hat{\mathbf{x}}_\mathbf{t} = \mathbf{x}_\mathbf{t} E \tag{1}$$

$$\mathbf{h}_\mathbf{t} = LSTM(\hat{\mathbf{x}}_\mathbf{t}, \mathbf{h}_{\mathbf{t-1}}) \tag{2}$$

$$\mathbf{o} = \mathbf{h}_\mathbf{t} W^T \tag{3}$$

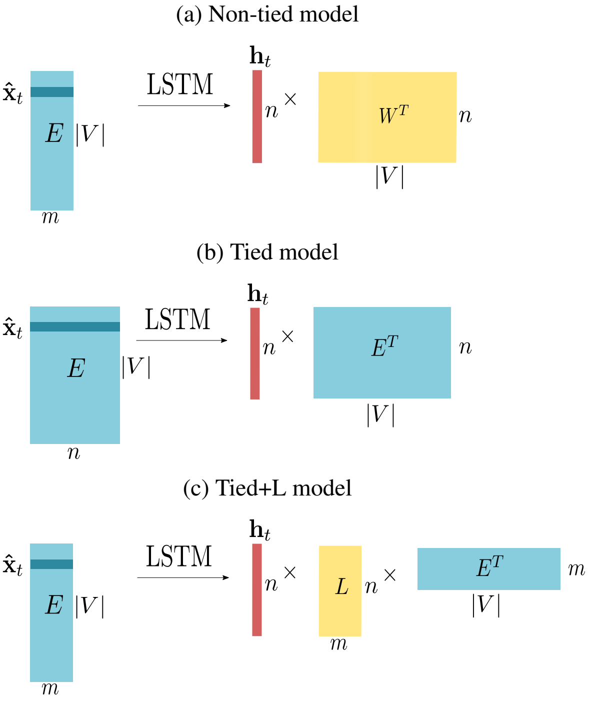
$$\mathbf{p} = softmax(\mathbf{o}), \tag{4}$$

where $\mathbf{x}_\mathbf{t}$ is the one-hot encoding of the input word at time t, $\hat{\mathbf{x}}_\mathbf{t}$ is its embedded representation and E is the embedding matrix (input mapping). We follow the majority of previous work in adopting an LSTM (Hochreiter and Schmidhuber, 1997) as the recurrent unit, as it was shown to outperform other recurrent architectures for LM (Jozefowicz et al., 2015). The hidden vector $\mathbf{h}_\mathbf{t}$ is used as input to the LSTM for the next time step and also as input to a linear transformation W (output mapping) which produces the output weights. These weights are normalised into probability scores using the softmax function.

The tied models proposed by Inan et al. (2017) and Press and Wolf (2017) set W to be equal to E (Figure 1b). Note that since E is of size $|V| \times m$, where m is the embedding size, and W is of size $|V| \times n$, where n is the hidden state size, the hidden and embedding dimensions must be equal, that is, $m = n$.

Press and Wolf (2017) observe that the output matrix W represents an embedding matrix since two similar words with indices i and j are learnt to receive similar probabilities given a context and hence the rows W_i and W_j should also be similar. Indeed, they show that on some word similarity evaluation tasks the output matrix W outperforms significantly the input matrix E. Tying E and W makes the model share the representations for the input and output vocabularies.

Inan et al. (2017) suggest a theoretical motivation for the tying technique and derive it as an instance of a more general approach of augmenting the cross-entropy loss. They show that a loss that takes into account not only the target word (i.e., $\log p_i$ for cross-entropy) but the scores for all words in the vocabulary according to their similarity to the target (computed as dot-product of embeddings in E) improves performance on LM.


(a) Non-tied model

(b) Tied model

(c) Tied+L model

Figure 1: Effect of the three architectures on mapping sizes. Note that the actual difference between vocabulary size $|V|$ and n, m is of 2 to 3 orders of magnitude.

One important practical advantage of tying input and output matrices is the reduction in number of parameters with respect to a standard model with the same hidden and embedding dimensions, since instead of two matrices E and W of size $|V| \times m$ we have only one matrix of size $|V| \times m$.

2.2 Proposed modification

One potential problem with the tied model, where $E = W$, is that the hidden state $\mathbf{h}_\mathbf{t}$ is optimised to be close to the embedding of the target word. To see this, consider that $o_j = \mathbf{e_j} \cdot \mathbf{h_t}, \forall j$. The cross-entropy objective is to maximise $\log p_i$ for the target word with index i and consequently to minimise $\log p_j \forall j \neq i$. Hence, $o_i = \mathbf{e_i} \cdot \mathbf{h_t}$ will be increased by the gradient descent update and $\mathbf{h_t}$ will be aligned closer with $\mathbf{e_i}$ (for each dimension k of the two vectors, $h_{tk} \cdot e_{ik}$ will be increased). This association between $\mathbf{h_t}$ and word embedding space could prevent efficient retention of LSTM history in $\mathbf{h_t}$, which is used as input for the following time step.

To address this issue, we propose a simple modification to the standard tied model, replacing the output transformation (3) as follows:

$$\begin{aligned}\hat{\mathbf{h}}_\mathbf{t} &= \mathbf{h}_\mathbf{t} L \\ \mathbf{o} &= \hat{\mathbf{h}}_\mathbf{t} E^T.\end{aligned} \tag{5}$$

L is an additional linear transformation that decouples the hidden state $\mathbf{h}_\mathbf{t}$, which is passed to the next time step and represents the previous, possibly long-term, linguistic context, from $\hat{\mathbf{h}}_\mathbf{t}$, which is instead optimised to match the embedding of the output word.

As Figure 1 illustrates, an important advantage of the additional transformation L is that a model can have different dimensions for the hidden vector (n) and the embedding vectors (m). The embedding matrix is the largest part of the model when the vocabulary is large ($|V| \gg n$). Reducing the size of the embedding m leads to a significant reduction in the number of parameters, proportional to $|V|$, and the acceleration of softmax computation. On the other hand, the size of the additional matrix L is only $n \times m$ and contributes very little to the overall size of the model. We test empirically how reducing the embedding size affects the performance of language models by varying hidden and embedding sizes in our experiments and evaluating embedding matrices on word similarity tasks.

Standardly used LM models often have two layers of LSTM cells. Thus, the issue we identifed might be mitigated in practice, since the hidden state of the first layer is not directly affected by tying the output and input transformation matrices. Moreover, an LSTM cell carries over information both through the hidden state and a memory state; the latter is affected by tying only indirectly (see Hochreiter and Schmidhuber (1997) for details on LSTM architectures). However, our experiments show that, in practice, two-layer LSTM LMs are still affected by tying despite these caveats.

2.3 Extending the tied technique to word2vec

The tied technique, as formulated above, can be in principle applied to any model which has the same general objective of LM: predicting a target word given context words. The CBOW word2vec model for word representation learning (Mikolov et al., 2013) is the primary candidate for testing the applicability of the tied technique beyond LM, since we can see it as substituting the LSTM function (2) with a simple sum of context word embeddings $\mathbf{h} = \sum_i \hat{\mathbf{x}}_\mathbf{i}$ (where x_i are words in the context window, e.g., of size 5). Similarly then, the equations in (5) describe the tied version of CBOW model. The linear step here provides the capacity to learn a transformation from the sum of embeddings to the predicted embedding $\hat{\mathbf{h}}$. Without such transformation, the tying model would assume that the sum function is always a good approximation of the output embedding.

The skipgram word2vec model (Mikolov et al., 2013) employs a variant of the LM objective: It is trained to predict context words given a word, instead of the opposite. As in CBOW and neural language models, words are both inputs and targets, making the use of tying an option also for this architecture. Press and Wolf (2017) apply direct tying to this architecture and report that the quality of the obtained embeddings is below the quality of non-tied skipgram embeddings. Unlike CBOW or LSTM, the input-to-hidden state function of the skipgram model is identity, reducing the tying model objective to $\hat{\mathbf{x}}_\mathbf{i} = \hat{\mathbf{x}}_\mathbf{j}$ for every pair of input-output words i, j. It is thus not surprising that enforcing the tying constraint leads to poor empirical results. We test whether adding an additional linear transformation improves performance of the tied technique also for a skipgram model.

3 Experiments and results

3.1 Evaluation data

We use two corpora for the evaluation of language models. First, we employ a medium-sized corpus of approximately 100M tokens with a relatively large vocabulary, 50K words, created from a Wikipedia dump (henceforth, Wiki).[1] To allow comparison with previous work, we also evaluate on the Penn Treebank (PTB), which is small but has been used as a benchmark for LM since Mikolov et al. (2011). The PTB has approximately 1M tokens and is preprocessed to have 10K vocabulary words; we use the standard train-validation-test split.

Furthermore, to evaluate the quality of the embeddings induced by the language models, as well as for the word representation experiments in Section 3.4, we use three standard word similarity datasets: SimLex-999 (Hill et al., 2015; SimLex), MEN (Bruni et al., 2014), and RareWords (Luong et al., 2013; RW). The performance on these datasets is evaluated in terms of Spearman corre-

[1] https://dumps.wikimedia.org/enwiki/20180301/

Hid	Emb	Model	Valid	Test	Δ	Size
200	200	non-tied	95.0	91.1		4.7M
		tied	90.8	86.6	-4.5	2.7M
		tied+L	89.8	85.8	-5.3	2.7M
400	200	non-tied	89.4	85.3		8.3M
		tied+L	83.4	80.3	-5.0	4.3M
	400	non-tied	87.2	83.5		10.6M
		tied	82.0	78.2	-5.3	6.6M
		tied+L	81.9	78.0	-5.5	6.7M
600	400	non-tied	85.8	82.4		15.3M
		tied+L	79.0	76.0	-6.4	9.5M
	600	non-tied	84.3	81.3		17.8M
		tied	79.7	76.1	-5.2	11.8M
		tied+L	78.7	75.5	-5.8	12.1M
Inan2017 VD tied 650			77.1	73.9		-
Zaremba2014 1500			82.2	78.4		66M
P&W2016 tied 1500			77.7	74.3		51M

Table 1: LM perplexity results on PTB. Δ: difference in test perplexity of the tied models with respect to the non-tied model with the same number of hidden units.

lation between the cosine similarity of word pairs and human judgments.[2]

3.2 Training setup

As our base language models, we adopt the ones proposed in Zaremba et al. (2014). We use 2-layer LSTMs with dropout applied to the input embedding, to the output of the first LSTM layer and to the output of the second layer. We used the PyTorch implementation[3] and modified it to include the additional linear layer for our tied models. We report the best model after the hyperparameter search for dropout and learning rate (see the details in Appendix A).

3.3 Language modelling results

We present the LM results for the standard non-tied model, the tied model as in Inan et al. (2017) and Press and Wolf (2017), and our tied model with an additional linear transformation (*tied+L*) in Tables 1 (PTB) and 2 (Wiki).

Table 1 confirms that tying generally brings gains with respect to not tying. This is also true for the cases when the hidden and embedding sizes are different (e.g. 400/200 and 600/400), where our tied+L model outperforms the non-tied model by 5 to 6.4 points having around 40% less parameters. Furthermore, our decoupled model slightly but consistently improves results with respect to standard tying, confirming our intuition that the coupling of the hidden state to the embedding representation is a limiting constraint. Smaller tied+L models perform well compared to larger tied models. In particular, the tied+L model with 600/400 units has perplexity of 76.0, compared to 76.1 of the tied 600/600 model, with 55% the number of parametres. Note that our results are comparable to previously reported perplexity values on PTB for similar models. Our best results of 75.5 test perplexity is only 1.2 points behind the large tied model with 1500 units reported in Press and Wolf (2017) and is only 1.6 points behind the medium tied model with 650 units and variational dropout (Gal and Ghahramani, 2016) reported in Inan et al. (2017).

On the Wiki corpus with larger vocabulary (Table 2), we find that tied models achieve slightly lower perplexity than non-tied models with half the number of parameters, and our proposed tied+L model achieves lower perplexity than the tied model. The most relevant result of the present experiment, however, is that the tied+L model with 300 embedding units is actually better than the tied model with 600 units (38.5 vs 39.7 points; the tied+L model has 20M parameters compared to 36M of the tied model) – that is, a smaller model outperforms a larger model. Thus, our decoupling mechanism not only allows models to have better perplexity, but also more compact word embeddings, which are of a higher quality also as measured on word similarity: .42/.61/.68 for the tied+L embeddings of size 300, compared to .39/.55/.64 for the tied embeddings of size 600.

3.4 Experiments on word2vec models

Table 3 presents the evaluation of word2vec models on the three word similarity datasets. We ran the experiments only on the Wiki corpus due to its higher coverage (50K vocabulary), and used embeddings of size 300.

Our results on CBOW show that the tied+L architecture obtains comparable results to the non-

[2] We computed correlation on the word pairs covered by the Wiki corpus, namely 98%, 88% and 31% (973, 2648 and 623 datapoints) for SimLex, MEN, and RW, respectively.

[3] `https://github.com/pytorch/examples/tree/master/word_language_model`

				Perplexity		Spearman's ρ		
Hidden	Emb	Model	Size	Valid	Test	SimLex	RW	MEN
600	300	non-tied	50M	40.0	39.2	.33/.34	.52/.50	.55/.60
		tied+L	20M	39.3	38.5	.42	.61	.68
	600	non-tied	66M	40.8	40.0	.33/.34	.52/.50	.55/.60
		tied	36M	40.5	39.7	.39	.55	.64
		tied+L	36M	38.6	37.9	.42	.59	.70

Table 2: Results on Wiki on LM (perplexity; lower is better) and word similarity (Spearman's ρ; for non-tied models, results for input and output matrix are reported).

Model	Type	SimLex	RW	MEN
CBOW	non-tied	.38	.51	.63
	tied+L	.38	.50	.65
skipgram	non-tied	.39	.52	.74
	tied	.18	.25	.50
	tied+L	.35	.51	.72

Table 3: Results (ρ) for word2vec models.

tied architecture with almost half the parameters (15.1M vs 30M). This confirms that tying with an additional linear transformation is appropriate not only for language models but for word learning models more generally.

The skipgram algorithm shows a small degradation of performance for the tied+L architecture with respect to the non-tied one; note that, as explained in Section 2.3, tying makes the most sense for CBOW. However, the fact that standard tying obtains much worse results (similarly to the results of Press and Wolf, 2017) shows that the linear mapping substantially relaxes the tying constraint.

4 Conclusions

Overall, our simple modification to tied language modelling architectures generalises previous work by allowing tying without imposing constraints on the number of hidden and embedding dimensions. This leads to flexible architectures with a more efficient use of both hidden states and embeddings. For word representation learning models, having an additional linear transformation reduces the number of parameters while maintaining learning capacity. In general, reducing model size without harming performance is a desirable feature in practice, for example in the case of language models running on mobile devices, and it is also desirable on theoretical grounds, since it is a better use of the learning capacity of neural networks.

Acknowledgements

We thank Germán Kruszewski and the AMORE team for the helpful discussions. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 715154), and from the Ramón y Cajal programme (grant RYC-2015-18907) and the Catalan government (SGR 2017 1575). We gratefully acknowledge the support of NVIDIA Corporation with the donation of GPUs used for this research. This paper reflects the authors' view only, and the EU is not responsible for any use that may be made of the information it contains.

References

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating Semantic Models with Genuine Similarity Estimation. *Comput. Linguist.*, 41(4):665–695.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9 8:1735–80.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In ICLR'17. arXiv preprint arXiv:1611.01462.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2342–2350. JMLR.org.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113. Association for Computational Linguistics.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and Optimizing LSTM Language Models. In ICLR'18. arXiv preprint arXiv:1708.02182.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. pages 1–12. ArXiv preprint arXiv:1301.3781.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163. Association for Computational Linguistics.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM Neural Networks for Language Modeling. In *INTERSPEECH*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization.