Decoding Sentence Representations for Text Generation

Weihao Mao[*], Xuchen Yang[*], Liyan Wang and Yves Lepage
早稲田大学大学院情報生産システム研究科
mao_weihao@asagi.waseda.jp, yang_xuchen@asagi.waseda.jp,
wangliyan0905@toki.waseda.jp, yves.lepage@waseda.jp

Abstract

Large-scale pre-trained language models have shown their potential in generating nearly human text. However, they require a large amount of data to be trained from. We explore the task of decoding sentence representations to reconstruct texts. We design two variants of a baseline vector-to-sequence model and explore different vector composition methods: unweighted and weighted sum, and concatenation. Experimental results show that a Transformer trained from scratch on sequences of word vectors achieves the highest BLEU score (94.2 BookCorpus sentences and 96.9 on Tatoeba sentences).

1 Introduction

Text generation is a core task of natural language processing. It is part of machine translation, text summarization, and dialogue systems. It consists in outputting the corresponding human-understandable text which correspond to a given input [1]. Various neural network models such as Variational Autoencoder (VAE) and Generative Adversarial Networks (GAN) have been designed for text generation [2, 3, 4]. Pre-trained language models based on Transformers [5] have achieved the state of the art [6]. However, these models require large amounts of data for training and high power consumption because of the use of GPUs.

An alternative solution to large pre-trained language models is to use word embedding spaces to create sentence representations and decode these sentence representations into sentences [7, 8, 9]. This reduces the size of the resources needed for text generation. The experiments reported in [8] with such a model are however limited to short sentences from the Tatoeba corpus[1], where the average length of sentences is seven words. In this paper, we explore this model further and extend to more general cases.

We examine different composition methods of word vectors into sentence representations and conduct experiments with different architectures for decoding these sentence representations into a sentence.

[*]These two authors contributed equally.
[1]https://tatoeba.org/en/

2 Methods

Suppose that there is a pre-trained embedding model which projects words to d-dimensional vectors in a semantic space. Given a sentence, we can obtain a sequence of word vectors by representing words of the sentence in isolation. In Section 2.1, we examine four different composition models to derive effective representations for sentences. Then, in Section 2.2, we introduce a decoding mechanism for text generation, i.e., transferring compositional representations into sentences.

2.1 Composition methods

The simplest method to represent a sentence from word vectors is to sum up the word vectors of the words appearing in the sentence. This will be the first sentence representation that we will use, as a baseline. We call it *unweighted sum*.

Although the above method is simple and effective in some tasks [10], it has the drawback of giving the same importance to functional words as to meaningful words. As we care about the meaning of the sentence, we choose to give more weights to meaningful words while lowering the importance of functional words. For that, we use the index document frequency (idf) as weights, i.e., we use the idf-weighted sum of word vectors as a sentence representation (see Equation (1)). This produces semantically more relevant sentence representations [11]. We call this *weighted sum*.

$$v_s = \sum_{w \in s} \mathrm{idf}_w \, v_w \qquad (1)$$

Here, v_w is the word vector of word w in some pre-trained word embedding space. idf_w is the inverse document frequency (idf) of word w, computed by considering each different sentence as a different document.

To capture long-distance dependencies between words, we propose to use the encoder part of an autoencoder model to generate a sentence representation from a sequence of word vector. We call this *encoder of autoencoder*.

The three above sentence representations lose the information about word order in the sentence, and mix-up information coming from different words on each vector dimension. So as to use more fine-grained information and retain the word order information, we propose to concatenate word vectors into a $d \times l$ matrix, where l is the number of words. This fourth sentence representation will be called *concatenation.*

2.2 Text generation

To perform text generation, i.e., to decode a sentence representation into a sentence, we explore the use of three different architectures of neural networks.

The problem of solving sentence analogy puzzles is addressed in [8]. For that, a vector-to-sequence model is proposed to learn how to transfer a sentence vector, computed from the sentence vectors given in the analogy puzzle, into a sequence of words. This is done by pre-training a separate sentence decoder. Although the decoding method achieved reasonable results on the semantico-formal analogy dataset used [12], the accuracy drops significantly on other datasets containing longer sentences or different sentence styles.

We propose two decoders with different architectures: RNN-based autoencoder and Transformer-based decoder.

We then implement the RNN-based decoder that is the same as the vector-to-sequence model to decode sentence vectors.

For the RNN-based autoencoder model, we keep the decoder part consistent with the decoder proposed by [8] and only add a single-layer BiLSTM as the encoder. BiLSTM learns how to convert the sequence of vectors into a sentence vector in a fixed-size and feeds it to the decoder.

We also train a model from scratch that only has the decoder part with the Transformer structure. In the Transformer decoder, we use padding and cutting to make all vectors a fixed-size. As for decoder input, we take a fixed-size sequence of vectors added with positional encoding. The hidden vector encoded by the network is mapped to the dimension of the vocabulary, and then the output is the word with the highest probability in each position. Furthermore, we introduce the attention mechanism into the vector-to-sequence model to solve the problem that the decoder does not perform well in decoding long sentences.

For Transformer-based decoder, we establish the model with a single Transformer layer.

3 Experiments

3.1 Datasets

We experiment with data from two corpora in the English language. The first one is the Tatoeba corpus in which sentences are short and are basically composed of only one main clause. The second one is BookCorpus [13]. The sentences come from novels by many different authors.

We randomly select around 60,000 (exactly 63,336) English sentences from each corpus, which we divide into training, validation, and test sets with the respective proportions of 80%, 10%, and 10%. The average length of sentences from the Tatoeba corpus is 6.7 in words and 28.5 in characters. For BookCorpus, it is 13.2 in words and 62.7 in characters.

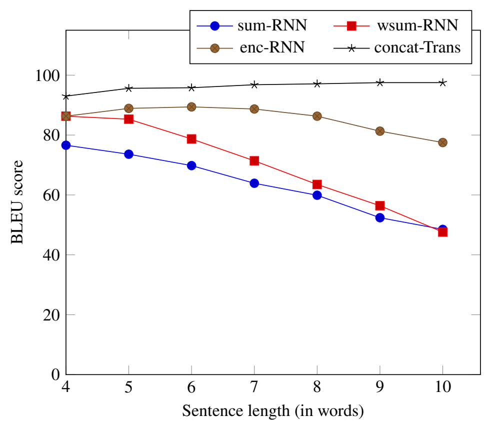
3.2 Setups

We use the pre-trained fastText model [14] to encode words into vectors.

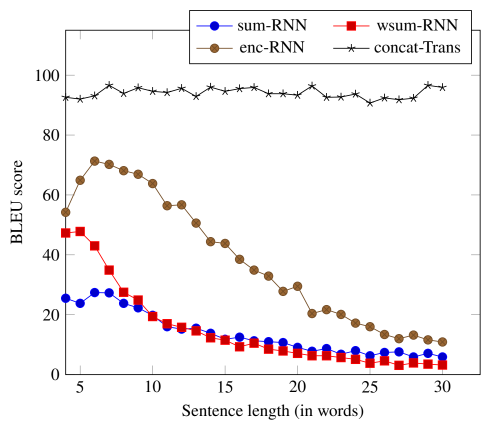
To explore the decoding performance in terms of the quality of sentence representations, we use four different vector composition methods: the unweighted sum of the word vectors (sum) and the weighted sum of the word vectors using scalar factors of the idf weighting (wsum), fixed-size vector generated by BiLSTM (enc) and the concatenation of word vectors (concat). We conduct four experiments on decoder models [2] in the following settings:

- sum-RNN: this is the model proposed in [8]. The

[2]The model size is affected by the size of the vocabulary. Considering that we want to use a small model for this task, the vocabulary of each model is just initialized with the training data.



(a) Tatoeba



(b) BookCorpus

Figure 1 Performance, in BLEU, of the different models, by lengths of sentences, for each of the corpora.

sentence vector is the unweighted sum of the word vectors of the words in the sentence. This model is our baseline model.

- wsum-RNN: same as the previous model, except that the sentence vector is the idf-weighted sum of the word vectors.
- enc-RNN: the sentence representation is the fixed-size vector generated by the encoder of autoencoder . The decoder part of the autoencoder is used to output a sequence of words.
- concat-Trans: the sentence representation is the concatenation of the sequence of word vectors. A Transformer-based decoder is used to output a sequence of words. This Transformer model is trained from scratch.

4 Results and discussion

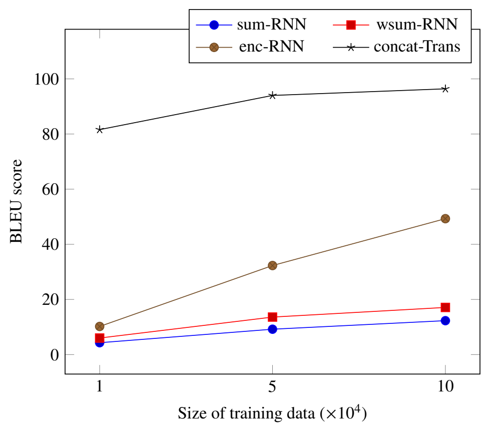


Figure 2 Performance of models learned from training data of different sizes.

4.1 Results

We use three evaluation metrics to quantify the decoding performance: accuracy, BLEU, and Levenshtein distance. Table 1 shows the results of the four experiments introduced in the previous section.

In terms of accuracy, i.e., outputting exactly the input sentence, the idf-weighted sum sentence representation provides an improvement of 16.0% and 3.1% on the two corpora, Tatoeba and BookCorpus, in comparison with the unweighted sum sentence representation (baseline model). On Tatoeba, the two proposed models (RNN model with the autoencoder structure and Transformer decoder) outperform the baseline by 34.7% and 51.4% , respectively. On BookCorpus, their accuracy is lower than on Tatoeba, especially for the RNN model. Although the accuracy of the Transformer decoder has also declined, it is still the best among all models, by a large margin.

It is worth mentioning that the Levenshtein distance using Transformer decoder is only 0.1. This means that even when decoding a not exactly same sentence, only less than one word in the sentence is incorrect on average.

Moreover, the Transformer decoder obtain 94.2 BLEU score, indicating that the decoded wrong sentences are still highly similar to the reference sentences.

We explore the effect of length of sentences and size of dataset on decoding result. As shown in Figure 1, on Tatoeba, the performance of baseline model using two sentence representations drop heavily as the length of sentence increasing. For the RNN-based autoencoder, the BLEU score only decreases slightly. Transformer decoder still

Input	Model		BLEU	Accuracy	Levenshtein distance	
Vector composition method	Architecture	Size (M)		(%)	in words	in chars
Tatoeba						
unweighted sum	RNN	3.8	62.5	40.5	1.4	6.7
weighted sum	RNN	3.8	68.9	56.5	1.3	5.6
encoder of autoencoder	RNN	4.4	86.2	75.2	0.4	2.2
concatenation	Transformer	4.6	**96.9**	**91.9**	**0.1**	**0.6**
BookCorpus						
unweighted sum	RNN	10.0	15.0	2.6	8.9	36.3
weighted sum	RNN	10.0	14.8	5.7	9.5	37.0
encoder of autoencoder	RNN	10.6	39.6	19.6	5.1	23.8
concatenation	Transformer	10.8	**94.2**	**76.0**	**0.3**	**1.9**

Table 1 Performance of the different models on the two datasets Tatoeba and BookCorpus. Recall that the sentences are three times longer in average in BookCorpus than in Tatoeba. The model sizes for each corpus is different because we filter out any word that does not belong to the training set from the word embedding space. The vocabulary size of Tatoeba is much lower that that of BookCorpus.

Reference	Results generated by			
	sum-RNN	wsum-RNN	enc-RNN	concat-Trans
this obviously meant something to him .	*obviously meant* ***to*** *something him.*	*obviously meant something* ***meant*** *to him .*	*this obviously meant something to him .*	*this obviously meant something to him .*
i 'm willing to let him do that , but i 'm not sure he 'd bring me to the garage for it .	*i 'm willing to let* ***me*** *, but i 'm sure* ***that*** *he 'd* ***do to let him that it 'd not .***	*i 'm* ***sure i 'm sure to bring me to bring me*** *, but i 'm sure* ***for me to do 'm*** *.*	*i 'm willing to let him do* ***there*** *, but i 'm not sure* ***i would put me toward the hospital*** *for it .*	*i 'm willing to let him do that , but i 'm not sure he 'd* ***come*** *me to the garage for it .*

Table 2 Various text generation results on BookCorpus with the different models.

maintain the great performance even when the sentences become longer.

On the BookCorpus corpus, in which the sentence length varies greatly, the performance of the three RNN models decreases rapidly as the sentence length increases. When the sentence length is close to 30, the BLEU scores of the three RNN models are close to 0. However, the BLEU score of Transformer decoder is stable across all sentence lengths. The model with attention mechanism exhibits better performance than the RNN models when decoding longer sentences. Some decoding examples with different lengths are presented in Table 2.

For the size of the training data set, the three models show interesting and different trends. As shown in Figure 2, for the Transformer decoder, increasing the training data improves the BLEU Score, but the improvement is not significant. With the increase of training data, BLEU Score has a more noticeable improvement for the other two models of RNN structure, especially the RNN of the autoencoder structure.

4.2 Discussion and Future Work

The use of sequence of vectors as sentence representation achieved better results in the decoding experiments. The performance of the Transformer decoder is much better than that of other models, especially on BookCorpus according to four different indicators. The attention mechanism plays a critical role in it. In the future, we will also apply attention on the RNN structure to let model focus on important words.

Although the experiments on sequence of vectors obtain better results than sentence vector, sentence vector is a lighter representation than sequence of vectors. The sentence representation in vector form is more efficient in manipulation and computation than the form of the sequence of vectors. Research on improving sentence vector decoding is indispensable.

5 Conclusion

We proposed two different models for decoding sentence representations into actual sentences. Our results showed that both the Transformer decoder and the RNN-decoder have a specific improvement compared with the latest research results. The Transformer decoder can also perform well in more general cases. Improving the decoder effect is of great significance for exploring the potential information of the vector space and future work from sentence representation mapping to text.

References

[1] Albert Gatt and Emiel Krahmer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Int. Res.*, Vol. 61, No. 1, p. 65–170, jan 2018.

[2] Wenlin Wang, Zhe Gan, Hongteng Xu, Ruiyi Zhang, Guoyin Wang, Dinghan Shen, Changyou Chen, and Lawrence Carin. Topic-guided variational auto-encoder for text generation. 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)*, pp. 166–177, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[3] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 10–21, Berlin, Germany, August 2016. Association for Computational Linguistics.

[4] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2157–2169, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, Vol. 30, , 2017.

[6] Genta Indra Winata, Andrea Madotto, Zhaojiang Lin, Rosanne Liu, Jason Yosinski, and Pascale Fung. Language models are few-shot multilingual learners. In *Proceedings of the 1st Workshop on Multilingual Representation Learning*, pp. 1–15, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[7] Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, Vol. 6, pp. 437–450, 2018.

[8] Liyan Wang and Yves Lepage. Vector-to-sequence models for sentence analogies. In *2020 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pp. 441–446, 2020.

[9] Pengjie Wang, Liyan Wang, and Yves Lepage. Generating the middle sentence of two sentences using pre-trained models: a first step for text morphing. In *Proceedings of the 27th Annual Conference of the Association for Natural Language Processing*, pp. 1481–1485, Kokura, Japan, March 2021. Association for Natural Language Processing.

[10] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *International conference on learning representations*, 2017.

[11] Marek Rei and Ronan Cummins. Sentence similarity measures for fine-grained estimation of topical relevance in learner essays. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 283–288, San Diego, CA, June 2016. Association for Computational Linguistics.

[12] Yves Lepage. Analogies between short sentences: A semantico-formal approach. In *Human Language Technology. Challenges for Computer Science and Linguistics: 9th Language and Technology Conference, LTC 2019, Poznan, Poland, May 17–19, 2019, Revised Selected Papers*, p. 163–179, Berlin, Heidelberg, 2019. Springer-Verlag.

[13] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

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