Skip-Prop: Representing Sentences with One Vector Per Proposition

Rachel Rudinger, Kevin Duh, Benjamin Van Durme
Johns Hopkins University
{rudinger,kevinduh,vandurme}@cs.jhu.edu

Abstract

We introduce the notion of a multi-vector sentence representation based on a "one vector per proposition" philosophy, which we term *skip-prop* vectors. By representing each predicate-argument structure in a complex sentence as an individual vector, skip-prop is (1) a response to empirical evidence that single-vector sentence representations degrade with sentence length, and (2) a representation that maintains a semantically useful level of granularity. We demonstrate the feasibility of training skip-prop vectors, introducing a method adapted from *skip-thought* vectors, and compare skip-prop with "one vector per sentence" and "one vector per token" approaches.

1 Introduction

The length and complexity of written natural language sentences is highly variable. Sentences from New York Times (NYT) stories (August 1997), for example, contain on average 23 tokens, with a standard deviation of 12. By information-theoretic measures, too, natural language sentences convey differing amounts of information (Hale, 2003; Genzel and Charniak, 2002). It is natural to suppose, then, that methods in computational linguistics that aim to learn fixed-size semantic representations of sentences, i.e., with vectors of fixed dimension, may be limited in their expressiveness or efficiency. Indeed, on many NLP tasks for which neural sentence embedding methods have been adapted, degraded performance on longer input sentences is commonly observed: in machine translation (Cho et al., 2014), question-answering (Kumar et al., 2016), and semantic role labeling (Zhou and Xu, 2015), for example.

Motivated by these observations, we introduce *skip-prop vectors*, a method for learning multi-vector sentence representations following a "one vector per proposition" strategy. Our approach is based on the *skip-thought* method of Kiros et al. (2015), which combines neural sequence-to-sequence models (Sutskever et al., 2014) with a skip-gram-like training objective (Mikolov et al., 2013) to obtain general-purpose sentence representations as a fixed-size vector. Skip-prop capitalizes on the idea that a complex sentence may be represented in terms of the simpler sentences, or *propositions*, that constitute it.

2 Motivation

There are many potential motivations for taking a "one vector per proposition" approach to representing the meaning of a sentence. As discussed in §1, it has been observed that NLP approaches that embed an entire sentence into a single, fixed-size vector may degrade in performance on longer sentences. One answer to this problem is to use finer-grained, multi-vector sentence representations that can grow with sentence complexity. Indeed, most neural (or otherwise continuous-space) models of sentences provide some finer-grained vector representations, most notably at the token level (i.e., standard RNN implementations), sub-token level (Sennrich et al., 2016), character-level (Kim et al., 2016), and syntactic constituent level (Dyer et al., 2016), and are often used in task-specific attention mechanisms. For many tasks involving search or attention, however, such as open question-answering or document-level analysis, preserving each such intermediate representation may be prohibitively expensive.

In comparison to other fine-grained, multi-vector representations, skip-prop offers two advantages: (1) the number of propositions (and hence vectors) per sentence is relatively few, and (2) the proposition is its own interpretable unit of meaning. Figs. 1 and 2 illustrate the granularity-expense tradeoff between one-vector-per sentence, proposition, and token representations. One sentence on average corresponds to 3.5 propositions and 22.8 tokens.[1] Which point in this tradeoff is optimal is likely a task-specific matter; by training *skip-prop* vectors, however, we introduce a new point in this tradeoff scale.

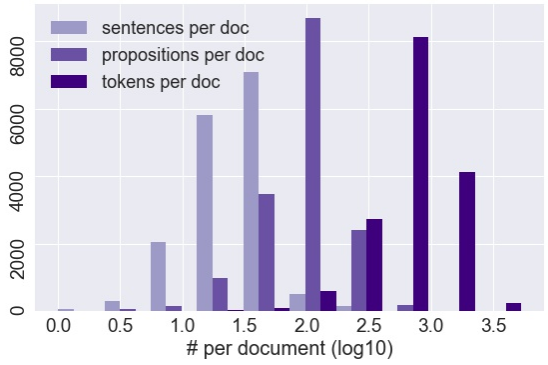

Figure 1: Histogram shows typical NYT documents contain more propositions than sentences, and many more tokens than propositions. (Log scale x-axis.)

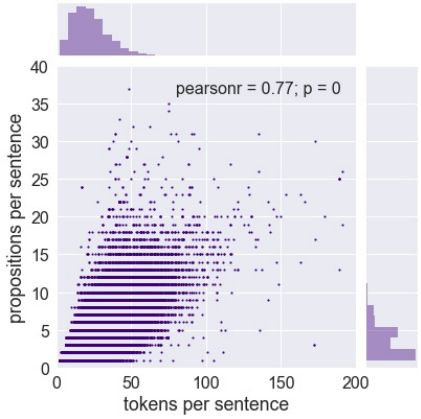

Figure 2: Scatter plot shows longer sentences contain more propositions. Most sentences contain fewer than 50 tokens, and fewer than 8 propositions.

3 Background

Sequence-to-Sequence Models Sequence-to-sequence (seq-to-seq) models are a class of neural networks that compute the conditional probability of an output sequence given an input sequence, i.e., $P(y_1...y_n|x_1...x_m)$. They have been applied to many tasks in NLP (Bahdanau et al., 2014; Vinyals et al., 2015; McClosky et al., 2006), though here we train them to encode multi-vector sentence representations.

Typical seq-to-seq models consist of two recurrent neural networks (RNNs): an *encoder* and *decoder*, which iterate over the input and output sequences, respectively. The final hidden state of the encoder RNN, h_m, is passed as the initial state to the decoder RNN. Thus, the vector h_m is a representation of the entire input, and the decoder computes the conditional distribution: $P(y_1...y_n|x_1...x_m) = P(y_1...y_n|h_m) = \prod_{i=1}^{n} P(y_i|y_{<i}, h_m)$. We train skip-prop with a multi-encoder, multi-decoder variant of seq-to-seq (§4), borrowing aspects of the skip-thought vector model (Kiros et al., 2015).

Sentences to Propositions Our method of learning a "one vector per proposition" representation relies on the use of PredPatt[2], a publicly-available tool for predicate-argument analysis of sentences, run atop Universal Dependency parses.[3] PredPatt extracts predicate-argument structures, or *propositions*, from sentences, including those arising from embedded clauses within the sentence. (See example in Fig. 3.) Though formal accounts of what constitutes a *proposition* may vary (McGrath, 2014), here we refer to a single extracted pattern as a *proposition*, comprising one fully-specified predicate-argument structure.

```
?a extracts ?b from ?c          ?a extracts ?b from ?c
     ?a:  PredPatt                   ?a:  PredPatt
     ?b:  predicates                 ?b:  arguments
     ?c:  text                       ?c:  text
```

Figure 3: An analysis of the sentence "PredPatt extracts predicates and arguments from text." Two propositions are extracted (when option to resolve conjunctions is enabled).

[1] Note that a binary parse of a sentence with N tokens has N-1 non-leaf nodes.

[2] `https://github.com/hltcoe/PredPatt` (commit eb42a8e, run with all flags enabled)

[3] Our method for training *skip-prop* vectors is in principle extensible to any language with UD parsers.

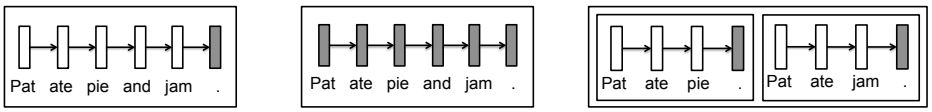

Figure 4: Left to right, encoders for ST, STA, and SP models. Each vertical rectangle represents the LSTM hidden state, h. Only shaded states are visible to the decoders. SP has one encoder per proposition.

Linearization We use a simple method to linearize each extracted proposition into a sequence of tokens so that each may be encoded by a linear-chain RNN. Specifically, each argument variable (`?a`, `?b`...) in the predicate pattern is replaced with the argument it stands for. Thus, linearizing the extraction in Fig. 3 yields "PredPatt extracts predicates from text" and "PredPatt extracts arguments from text."

4 Models

We compare **skip-prop vectors** (SP) with two other representations: **skip-thought vectors** (ST) as "one vector per sentence," and **skip-thought vectors with attention** (STA) as "one vector per token." (Fig 4.)

These models' architectures have many overlapping components, which we present in a unified fashion, drawing distinctions across models as necessary. Each model is a variant of seq-to-seq (encoder-decoder), trained on sentence tuples, (s_l, s_c, s_r), where sentences s_l and s_r are the left (previous) and right (next) context sentences of sentence s_c in a text document, following the approach of Kiros et al. (2015). The *encoder* (one or more RNNs) computes a representation of s_c and passes it to a left RNN decoder and a right RNN decoder, which compute $P(s_l|s_c)$ and $P(s_r|s_c)$, respectively. Together, the two decoders determine the total loss of the network: $-log(P(s_l|s_c)) + -log(P(s_r|s_c))$. We refer to this as the "skip-thought objective" (in contrast with the "autoencoder objective," described below). The gradient of each network parameter with respect to this loss is computed using backpropagation, and all parameters are updated according to the Adam optimization algorithm for stochastic gradient descent.

For notation, we say sentences s_l, s_c, and s_r consist of L, C, and R tokens, each. The tokens of s_c are $w_c^1, ..., w_c^C$ with corresponding embeddings $x_c^1, ..., x_c^C$. For skip-prop, s_c is preprocessed with PredPatt (§3), generating propositions $\pi_1, ..., \pi_P$. A proposition π_p is a sequence of C_p tokens $w_p^1, ..., w_p^{C_p}$.

Encoder Both skip-thought models (ST and STA) use the same encoder: one RNN with a long short-term memory (LSTM) cell (Hochreiter and Schmidhuber, 1997). At time step t, the LSTM cell applies its recurrent update equations to its previous state, (c^{t-1}, h^{t-1}), and an input, x^t, to yield its new state, (c^t, h^t). The final hidden state of the encoder, h^C, represents the entire input sequence s_c.

Skip-prop uses an identical LSTM architecture in its encoder; however, because skip-prop has P input sequences (i.e., one per proposition π_p, instead of just one for the full sentence s_c), it uses P *identical copies* of this LSTM (with shared parameters) to encode each proposition. Thus, the P encoders of skip-prop yield P final-state representations, $h_1^{C_1}, ..., h_P^{C_P}$, to be passed to the decoders. An attention mechanism is used in the skip-prop decoders to accommodate this variable number of vectors passed from the encoder (see below). The dimensionality of h and x is 256 for all models.

Decoder The ST, STA, and SP models all use two LSTM-based decoders, with the same basic architecture as the LSTM encoder (see above). In each model, the left and right decoders are identical (though with separate parameters), so we sometimes drop the l and r subscripts. For clarity, d_t denotes the *decoder* hidden state, akin to h_t in the encoder. For models ST and STA, the decoder's hidden state is initialized as the final hidden encoder state, i.e. $d_0 = h_C$; in SP, d_0 is a trainable parameter. The dimensionality of the hidden decoder states d is the same as the encoder state for all models (256). Each decoder computes the probability of an output sequence, i.e. s_l or s_r, conditioned on the encoder's representation of s_c (§3). For the right-hand decoder, specifically,

$$P(y_i|y_{<i}, h_m) = P(w_r^t|w_r^{<t}, h^C) = P(w_r^t|d_t) = \text{softmax}(d_t^\intercal W_d) \quad (1)$$

Model	Skip-Thought Obj.			Autoencoder Obj.		
	train	dev	test	train	dev	test
Skip-Thought (ST)	171.32	223.72	216.00	74.22	87.63	88.60
Skip-Thought w/ Attn. (STA)	152.89	204.51	199.16	2.04	2.20	2.22
Skip-Prop (SP)	169.29	213.01	206.29	29.69	41.42	43.21

Table 1: Average per-token perplexity, both with skip-thought and autoencoder objectives.

where W_d is an output vocabulary embedding matrix, also a trainable parameter.

Attention Mechanism The decoder described in the previous section is modified in the case of models STA and SP with an attention mechanism. Our attention mechanism is adapted from Vinyals et al. (2015). At each time step in the decoder, a weighted average over a set of vectors passed from the encoder is dynamically computed. In STA, this set is all encoder hidden states $h_1 ... h_C$; in SP, it is the final hidden state of each encoder (one per proposition), i.e. $h_{C_1} \dots h_{C_P}$. The weighted average at decoder time t is computed according to Vinyals et al. (2015):

$$a_i^t = \text{softmax}_i(v^\mathsf{T} \tanh(W_1 h_i + W_2 d_t)) \qquad (2) \qquad\qquad d'_t = h_1 a_1^t + h_2 a_2^t + ... + h_C a_C^t \qquad (3)$$

where W_1, W_2, and v are learnable parameters. The resulting vector d'_t is concatenated with d_t to create a new decoder output, though the hidden state as passed to the next time step remains unchanged. A result of this concatenation step, the output embedding matrix (W_d) in STA and SP is doubled in dimension.

Autoencoder Variant We train a second version of each model with an autoencoder-like objective in place of the skip-thoughts objective. That is, rather than use two decoders to predict the left and right context sentences, use a single decoder to predict the original sentence that was fed to the encoder (s_c). The autoencoder variant of each model is designated with the suffix -AUTO.

5 Experiments

Data All training, development, and test data consist of articles from the NYT portion of the Concretely Annotated Gigaword corpus labeled "story" (Ferraro et al., 2014). Train is 100K random sentence triples from Aug. 1997 NYT stories; development is 5K random sentence triples from Sept. 1997; and test is 5K random sentence triples from Oct. 1997. The vocabulary is approximately 39K tokens from Sept. 1997 NYT with minimum frequency of 15. Each model is trained for one epoch on the entire train set using mini-batches of size 1. As described in §4, a sentence triple (s_l, s_c, s_r) consists of a contiguous set of three sentences from a news story: a "left," "center," and "right" sentence. For the qualitative nearest-neighbor experiments, two datasets are used: (1) a 100K superset of the NYT development set (Sept. 1997), and (2) all sentences from the SICK corpus (Marelli et al., 2014).

Results As a preliminary evaluation of *skip-prop* vectors, we present both quantitative and qualitative results. These results show that (1) it is feasible to train *skip-prop* vectors with our proposed method, and (2) some notion of semantic similarity over propositions is preserved in this representation.

Table 1 shows the perplexity attained by each model. Here, perplexity is computed either from the two decoders' predictions of the left and right context sentences (skip-thought objective), or one decoder's prediction of the original sentence (autoencoder objective). In all cases, the *skip-prop* models score in between skip-thought and skip-thought with attention models. This is not surprising: the skip-prop decoder has, on average, access to more vectors than the skip-thought decoder, but fewer than the skip-thought with attention decoder.[4] (See Figs. 4 and 1.) This kind of result supports the plausibility

[4]This result is particularly magnified under the autoencoder objective, where the skip-thought with attention model attains very low perplexity by learning to attend to the token it needs to decode at each step.

of *skip-prop* vectors as a sentence representation that successfully trades off between the size and cost of one-vector-per-sentence strategies (ST) and one-vector-per-token strategies (STA).

Table 2 shows the qualitative results of a nearest neighbor search for both *skip-thought* and *skip-prop* vectors. We use both in-domain and out-of-domain data: 100K sentences from our NYT development set, and about 40K sentences from the SICK corpus (Marelli et al., 2014). Both query sentences are a random sentence from NYT or SICK with a correct predicate-argument analysis. The corresponding query propositions are each an extracted proposition from the query sentence.

The results in Table 2 suggest that skip-prop vectors provide a useful level of granularity for representing sentence meaning. For example, the NYT query sentence contains multiple salient propositions (`?a owns ?b`, `?a will jettison ?b`, etc.). While the skip-thought representation must pack all of this information into a single vector, skip-prop vectors allow us to represent each proposition *individually*. Accordingly, in the corresponding NYT query proposition, we see that it is possible to isolate a particular proposition of interest (`?a owns ?b`) and find nearest-neighbors of that proposition, without regard to rest of the sentence's content. This allows a more targeted search using skip-prop vectors.

NYT Query Sentence: H&R Block Inc. , which owns 80 percent of CompuServe , will jettison a business it 's been trying to unload for more than a year .
(ST) New Mexico would notify New York when they release convicted murderers into the Empire State ; New York would notify New Mexico .
(ST-AUTO) IAI Balanced Fund is for investors who want a little bit of everything , Hoelting said .
(SP) The Air Line Pilots Association and US Airways management have been at odds over the labor contract since for more than a year .
(SP-AUTO) H&R Block Inc. , which owns 80 percent of CompuServe , rose 11/16 to 40 7/8 .
NYT Query Proposition: *H&R Block Inc.* **owns** *80 percent of CompuServe*
(SP) H&R Block Inc. , the Kansas City , Missouri , *tax preparation company* that **owns** *80 percent of CompuServe* , will save on taxes by minimizing its gain on the sale .
(SP-AUTO) *Charterhouse* **owns** *80 percent of HRC* , while Accor owns the rest .
SICK Query Sentence: The woman with a black hat is wearing sunglasses
(ST) The blonde girl with the pink top is smiling and wearing funny glasses with a large nose attached
(ST-AUTO) the black and white dog is running outdoors
(SP) *The man with brown hair* **is wearing** *sunglasses* and is sitting listlessly at a table with cans of soda and other drinks
(SP-AUTO) *The woman* is sitting on a bench and **is wearing** a gray jacket and *black pants*
SICK Query Proposition: *a hat* **is/are black**
(SP) A dog and a black man are running through brown leaves [*a man* **is/are black**]
(SP-AUTO) A bride with a black veil is looking down [*a veil* **is/are black**]

Table 2: 1-best nearest-neighbor search over sentences and propositions, using vectors from trained skip-thought (ST) and skip-prop (SP) encoders. Both training objectives, "skip-thought" and "autoencoder" (-AUTO) are compared. Resulting nearest-neighbor propositions are shown within the full sentence they were extracted from; however, only the proposition's **predicate** and *arguments* are represented in its vector. Scoring by cosine similarity.

6 Conclusion

In this paper, we have proposed *skip-prop* vectors, a multi-vector sentence representation that extends the approach of *skip-thought* vectors (Kiros et al., 2015), to represent sentences according to a *one-vector-per-proposition* strategy. We have discussed how skip-prop vectors offer a potential solution to the observed issue of RNN performance degradation on longer sequences, allowing the sentence representation to grow roughly with its length or information content. We have also demonstrated how skip-prop vectors offer a new trade-off point between one-vector-per-sentence and one-vector-per-token strategies, balancing representation size (number of vectors) with performance (test perplexity), at a meaningful level of granularity (the proposition). Test perplexity results indicate that the skip-prop representation is feasible to train, while qualitative results suggest that skip-prop vectors capture some notion of meaning at the proposition level. We believe that this set of attributes makes skip-prop vectors a potentially suitable representation for tasks like question-answering, summarization, or machine reading, and hope to pursue such applications in future work.

Acknowledgments

This work is supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1232825, DARPA LORELEI, the Johns Hopkins Human Language Technology Center of Excellence (HLTCOE), and the Johns Hopkins Center for Language and Speech Processing (CLSP). We would also like to thank Sheng Zhang, Pushpendre Rastogi, and three anonymous reviewers for their feedback. Any opinions expressed in this work are those of the authors.

References

Bahdanau, D., K. Cho, and Y. Bengio (2014). Neural machine translation by jointly learning to align and translate. *CoRR abs/1409.0473*.

Cho, K., B. van Merrienboer, D. Bahdanau, and Y. Bengio (2014, October). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, Doha, Qatar, pp. 103–111. Association for Computational Linguistics.

Dyer, C., A. Kuncoro, M. Ballesteros, and N. A. Smith (2016, June). Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California, pp. 199–209. Association for Computational Linguistics.

Ferraro, F., M. Thomas, M. R. Gormley, T. Wolfe, C. Harman, and B. Van Durme (2014). Concretely annotated corpora. In *AKBC Workshop at NIPS*.

Genzel, D. and E. Charniak (2002). Entropy rate constancy in text. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Graff, D. and C. Cieri (2003). English gigaword corpus. *Linguistic Data Consortium*.

Hale, J. (2003). The information conveyed by words in sentences. *Journal of Psycholinguistic Research 32*(2), 101–123.

Hochreiter, S. and J. Schmidhuber (1997, November). Long short-term memory. *Neural Comput. 9*(8), 1735–1780.

Kim, Y., Y. Jernite, D. Sontag, and A. M. Rush (2016). Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 2741–2749. AAAI Press.

Kiros, R., Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler (2015). Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems 28*, pp. 3294–3302. Curran Associates, Inc.

Kumar, A., O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher (2016, 20–22 Jun). Ask me anything: Dynamic memory networks for natural language processing. In M. F. Balcan and K. Q. Weinberger (Eds.), *Proceedings of The 33rd International Conference on Machine Learning*, Volume 48 of *Proceedings of Machine Learning Research*, New York, New York, USA, pp. 1378–1387. PMLR.

Marelli, M., S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, and R. Zamparelli (2014). A sick cure for the evaluation of compositional distributional semantic models. In *LREC*, pp. 216–223.

McClosky, D., E. Charniak, and M. Johnson (2006, June). Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, New York City, USA, pp. 152–159. Association for Computational Linguistics.

McGrath, M. (2014). Propositions. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Spring 2014 ed.). Metaphysics Research Lab, Stanford University.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013). Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc.

Napoles, C., M. Gormley, and B. Van Durme (2012). Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pp. 95–100. Association for Computational Linguistics.

Sennrich, R., B. Haddow, and A. Birch (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725. Association for Computational Linguistics.

Sutskever, I., O. Vinyals, and Q. V. Le (2014). Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 27*, pp. 3104–3112. Curran Associates, Inc.

Vinyals, O., L. u. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton (2015). Grammar as a foreign language. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems 28*, pp. 2773–2781. Curran Associates, Inc.

Zhou, J. and W. Xu (2015, July). End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Beijing, China, pp. 1127–1137. Association for Computational Linguistics.