DECODING DECODERS: FINDING OPTIMAL REPRESENTATION SPACES FOR UNSUPERVISED SIMILARITY TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Experimental evidence indicates that shallow bag-of-words models outperform complex deep networks on many unsupervised similarity tasks. Introducing the concept of an optimal representation space, we provide a simple theoretical resolution to this apparent paradox. In addition, we present a straightforward procedure that, without any retraining or architectural modifications, allows deep recurrent models to perform equally well (and sometimes better) when compared to shallow models. To validate our analysis, we conduct a set of consistent empirical evaluations and introduce several new sentence embedding models in the process. While the current work is presented within the context of natural language processing, the insights are applicable to the entire field of representation learning.

1 INTRODUCTION

Distributed representations have played a pivotal role in the current success of machine learning. In contrast with the symbolic representations of classical AI, distributed representation spaces can encode rich notions of semantic similarity in their distance measures, which in turn can allow generalisation to novel inputs, a critical component of any learning system. Methods to learn these representations have recently gained significant traction, in particular for modelling words (Mikolov et al., 2013a). They have since been successfully applied to many other domains, including images (Girod et al., 2011; Razavian et al., 2014) and graphs (Kipf & Welling, 2016; Grover & Leskovec, 2016; Narayanan et al., 2017).

Learning representations from unlabelled data that are useful in other tasks is at the forefront of modern machine learning research. The Natural Language Processing (NLP) community in particular, has invested significant efforts in the construction (Mikolov et al., 2013a; Pennington et al., 2014; Bojanowski et al., 2016; Joulin et al., 2017), evaluation (Baroni et al., 2014) and theoretical analysis (Levy & Goldberg, 2014) of distributed representations for words.

Recently, attention has shifted towards learning representations of larger pieces of text, such as phrases (Yin & Schütze, 2015; Zhang et al., 2017), sentences (Kalchbrenner et al., 2014; Kiros et al., 2015; Tai et al., 2015; Hill et al., 2016; Arora et al., 2017), and entire paragraphs (Le & Mikolov, 2014). Some of this work has relied on a sentence-level version of the distributional hypothesis (Harris, 1954), an assumption that sentences which occur in close proximity have a similar meaning. Models trained in an unsupervised manner on large corpora of text are usually applied to supervised transfer tasks, where the representation for a sentence forms the input to a supervised classification problem, or to unsupervised similarity tasks, where the (cosine) similarity of two inputs informs some downstream process, such as information retrieval.

When evaluating different architectures, Hill et al. (2016) observed that, while deep complex models are preferable in supervised settings, shallow log-linear bag-of-words (BOW) models outperform deep learning approaches on unsupervised similarity tasks. However, these unsupervised tasks are more interesting from a general AI point of view, as they test whether the machine truly understands the human notion of similarity, without being explicitly told what is similar. Current experimental evidence suggests that shallow models are somehow better at this, contradicting the established view of superiority of deep models in abundance of data.
In this work we attempt to address these observations. Our main contributions are as follows:

- We introduce the formalism of an *optimal representation space*, in which the similarity measure is optimal with respect to the objective function.
- We show that models with log-linear decoders are usually evaluated in the optimal space, while recurrent models are not. This effectively explains the performance gap on unsupervised similarity tasks.
- We show that, when evaluated in their optimal space, recurrent models close that gap.
- We validate our findings with a series of consistent empirical evaluations based on a single publicly available codebase. In the process, we introduce new hybrid models with promising performance characteristics for transfer tasks.

## 2 Unsupervised Sentence Representations

In contrast to the supervised setting, where the input and output signals are usually well-defined, learning sentence representations in an unsupervised manner is not so straightforward. It is not entirely clear what the goals of a learning system should be in this setting, or whether there are any guarantees that the resulting embeddings will be useful in any downstream tasks.

To overcome these difficulties, it is sometimes possible to reformulate the original problem as a supervised learning task by introducing some domain-dependent assumptions. For instance, word embedding models such as Word2Vec rely on the distributional hypothesis to learn to predict a word given its context, or alternatively, predict a context given a word. Such representations are well-known to capture some degree of similarity as perceived by humans (Mikolov et al., 2013b).

Based on the success of word-embeddings, many distributed sentence representations are derived from straightforward functions of constituent word vectors, such as the sum or average (Mitchell & Lapata, 2010; Milajevs et al., 2014; Wieting et al., 2015; Arora et al., 2017). Such models can yield surprisingly high-quality representations (Arora et al., 2017), but naturally cannot leverage any contextual information for building sentence representations, leading some researchers to move towards more powerful embedding models.

Inspired by the word vector learning procedure by Mikolov et al. (2013b) and the encoder-decoder framework of Cho et al. (2014), Kiros et al. (2015) showed that it is possible to learn high-quality generic sentence representations from unlabelled data by leveraging a sentence-level distributional hypothesis (Polajnar et al., 2015). Their model, SkipThought, uses a Recurrent Neural Network (RNN) encoder and two RNN decoders to predict for any given sentence its two adjacent sentences.

SkipThought has enjoyed impressive results on many supervised transfer benchmarks, particularly when implemented with layer normalisation (Ba et al., 2016). In these benchmarks, the encoder is fixed after training on a separate unannotated dataset and then used as a feature extractor for a simpler model (such as logistic regression) for the task in question. In this setting, SkipThought with layer normalisation is currently considered to be one of the best general purpose sentence-level encoders (Conneau et al., 2017).

In contrast, the SkipThought model has proved notably less successful on unsupervised similarity tasks (Hill et al., 2016; Conneau et al., 2017). These benchmarks measure how the cosine distance between sentence pairs correlates with corresponding human judgements of semantic similarity. Since no additional model is being trained in this scenario, the encoder is evaluated on its own merits. While this shortcoming of SkipThought and RNN-based models in general has been pointed out, to the best of our knowledge, it has never been systematically addressed in the literature before.

Motivated by the success and, at the same time, dissatisfied with the slow training time of SkipThought, Hill et al. (2016) proposed a conceptually similar approach, FastSent. This log-linear model replaces computationally expensive RNN encoder and decoders with much simpler BOW versions, admittedly sacrificing word order information as a consequence. Interestingly, this change made FastSent among state-of-the-art on unsupervised similarity tasks, but dramatically reduced the performance on supervised benchmarks (Hill et al., 2016; Conneau et al., 2017).

Others in the literature have recognised the need to analyse the geometric characteristics of representation spaces (Almahairi et al., 2015; Schnabel et al., 2015). Most relevantly, Hill et al. (2016)
concluded that “Deeper, more complex models are preferable for representations to be used in supervised systems, but shallow log-linear models work best for building representation spaces that can be decoded with simple spatial distance metrics.” Our work provides the beginnings of this analysis via an extension and refinement of above statement.

3 Optimal Representation Space

3.1 Notation

Before we describe each model separately, we formally introduce the setting. Let \( S = (s_1, s_2, \ldots, s_N) \) be an ordered corpus of contiguous sentences where each sentence \( s_i = w^i_1 w^i_2 \ldots w^i_{\tau_i} \) consists of words from a pre-defined vocabulary \( V \) of size \( |V| \). Additionally, \( x_w \) denotes a one-hot encoding of \( w \) and \( v_w \) is the corresponding (input) word embedding. We transform the corpus into a set of pairs \( D = \{(s_i, c_i)\}_{i=1}^{N} \), where \( s_i \in S \) and \( c_i \) is a context of \( s_i \). Throughout this paper we assume that contexts \( c_i \) are given by \( c_i = s_{i-1} \cup s_{i+1} \); our analysis readily generalises to different definitions of the context.

3.2 FastSent and Log-linear Decoders

FastSent consists of an additive BOW encoder and two log-linear BOW decoders. Due to the model’s simplicity, it is particularly fast to train and evaluate, yet has shown state-of-the-art performance in unsupervised similarity tasks \cite{Hill2015}. Here we describe and analyse the model in three parts: the encoder, decoders, and objective.

Encoder. A simple BOW encoder represents a sentence \( s_i \) as a sum of the input word embeddings:

\[
\mathbf{h}_i = \sum_{w \in s_i} \mathbf{v}_w. \tag{1}
\]

Decoders. The two decoders share weights and can be considered as a single decoder that outputs a probability distribution over the vocabulary conditional on a sentence \( s_i \):

\[
(\hat{y})_w = p_{\text{model}}(w|s_i) = \frac{\exp (\mathbf{u}_w \cdot \mathbf{h}_i)}{\sum_{w' \in V} \exp (\mathbf{u}_{w'} \cdot \mathbf{h}_i)}, \tag{2}
\]

where \( \mathbf{u}_w \in \mathbb{R}^d \) is the output word embedding for a word \( w \). (Biases are omitted for brevity.)

Objective. The objective is to maximise the model probability of contexts \( c_i \) given sentences \( s_i \) across the corpus \( D \) which corresponds to finding the Maximum Likelihood Estimator (MLE) for the trainable parameters \( \theta \):

\[
\theta_{\text{MLE}} = \arg \max_{\theta} \prod_{(s_i, c_i) \in D} p_{\text{model}}(c_i|s_i; \theta). \tag{3}
\]

In the case of the log-linear BOW decoder above, the context \( c_i \) contains words from both \( s_{i-1} \) and \( s_{i+1} \) and the probabilities of words are independent, yielding

\[
p_{\text{model}}(c_i|s_i; \theta) = \prod_{w \in c_i} p_{\text{model}}(w|s_i; \theta) = \prod_{w \in c_i} \frac{\exp (\mathbf{u}_w \cdot \mathbf{h}_i)}{\sum_{w' \in V} \exp (\mathbf{u}_{w'} \cdot \mathbf{h}_i)} = \prod_{w \in c_i} \frac{\exp (\mathbf{u}_w \cdot \mathbf{h}_i)}{|c_i| \sum_{w' \in V} \exp (\mathbf{u}_{w'} \cdot \mathbf{h}_i)}. \tag{4}
\]

Switching to the negative log-likelihood and inserting the above expression, we arrive at the following optimisation problem:

\[
\theta_{\text{MLE}} = \arg \min_{\theta} \left[ - \sum_{(s_i, c_i) \in D} \left( \sum_{w \in c_i} \mathbf{u}_w \cdot \mathbf{h}_i + |c_i| \log \sum_{w' \in V} \exp (\mathbf{u}_{w'} \cdot \mathbf{h}_i) \right) \right]. \tag{5}
\]

\(^1\)In practice, we minimise the Kullback-Leibler Divergence (KLD) between the data distributions \( y \) and model distributions \( \hat{y} \), which is known to be an equivalent problem.
Noticing that
\[ \sum_{w \in c_i} u_w \cdot h_i = \left( \sum_{w \in c_i} u_w \right) \cdot h_i = c_i \cdot h_i, \quad (6) \]
we see that the objective in Equation (5) forces the sentence representation \( h_i \) to be similar under dot product to its context representation \( c_i \), which is simply the sum of the output embeddings of the context words. Simultaneously, output embeddings of words that do not appear in the context of a sentence are forced to be dissimilar to its representation. We explicitly demonstrate this in Appendix A.

Finally, using \( \cos \sim \) to denote close under cosine similarity, we find that if two sentences \( s_i \) and \( s_j \) have similar contexts, then \( c_i \cos \sim c_j \). Additionally, the objective function in Equation (5) ensures that \( h_i \cos \sim c_i \) and \( h_j \cos \sim c_j \). Therefore, it follows that \( h_i \sim h_j \).

Putting it differently, sentences that occur in related contexts are assigned representations that are similar under cosine similarity \( \cos (\cdot, \cdot) \) and thus \( \cos (\cdot, \cdot) \) is the appropriate similarity measure in the case of log-linear decoders.

Interestingly, the objective in Equation (5) only depends on the sum encoder from Equation (1) through the encoder output \( h_i \). Thus we could exchange the encoder with other functions, such as a deep or even recurrent neural network, and would still arrive at the same conclusion.

These observations lead us to a simple but important statement: in any model where the decoder is log-linear with respect to the encoder, the space induced by the encoder and equipped with \( \cos (\cdot, \cdot) \) as the similarity measure is what we call an optimal distributed representation space: “a space in which semantically close concepts (or inputs) are close in distance” (Goodfellow et al., 2016) and that distance is optimal with respect to the model’s objective.

As a practical corollary, FastSent is among the best on unsupervised similarity tasks because these tasks use \( \cos (\cdot, \cdot) \) for similarity and hence evaluate the models in their optimal representation space. Admittedly, evaluating a model in its optimal space does not by itself guarantee any good performance downstream as the tasks might deviate from the model’s assumptions. For example, if sentences “my cat likes my dog” and “my dog likes my cat” are labelled as dissimilar, FastSent will stand no chance of succeeding. However, as we show later, evaluating the model in a suboptimal space may very well hurt its performance.

### 3.3 SkipThought and Recurrent Sequence Decoders

SkipThought consists of a recurrent RNN encoder along with two RNN decoders that effectively predict, word for word, the context of a sentence. While computationally complex, it is currently the state-of-the-art model for supervised transfer tasks (Hill et al., 2016). The same cannot be said about its performance on unsupervised similarity tasks, where SkipThought lags behind much simpler models. In this section we explain why and show how to close the performance gap. Its encoder, decoders, and objective are defined as follows:

**Encoder.** The encoder is a Gated Recurrent Unit (GRU) (Cho et al., 2014):

\[
\begin{align*}
r^t &= \sigma (W_r v^t + U_r h^{t-1}), \\
z^t &= \sigma (W_z v^t + U_z h^{t-1}), \\
\tilde{h}^t &= \tanh \left[ Wv^t + U (r^t \odot h^{t-1}) \right], \\
h^t &= (1 - z^t) \odot h^{t-1} + z^t \odot \tilde{h}^t.
\end{align*}
\]

where \( \odot \) denotes the element-wise (Hadamard) product.

---

2Evidently, the correct measure is actually the dot product. However, the unboundedness of the dot product makes it impossible to tell what is similar and antisimilar in absolute terms, so we use cosine similarity instead.

3We follow the notation of Kiros et al. (2015) where \( z \leftrightarrow 1 - z \) compared to standard conventions.
Decoders. The previous and next sentence decoders are also GRUs. The initial state for both is given by the final state of the encoder

\[ h_{i-1}^0 = h_i^0 = h_i^{t_i}. \]  

and the update equations are the same as in Equations (7) to (10). Note that the original implementation in Kiros et al. (2015) additionally biases the gates by the encoder output, while our version makes no such provision. We discuss this change and its performance impact in Appendix B.

The time unrolled states of the previous sentence decoder are converted to probability distributions over the vocabulary conditional on the sentence \( s_i \) and all the previously occurring words

\[ \hat{y}_{i-1}^t_w = p_{\text{model}}(w_{i-1}^t | w_{i-1}^{t-1}, \ldots, w_{i-1}^1, s_i; \theta) = \frac{\exp(\mathbf{u}_w \cdot h_{i-1}^t)}{\sum_{w' \in V} \exp(\mathbf{u}_{w'} \cdot h_{i-1}^t)}. \]  

The outputs \( \hat{y}_{i+1}^t \) of the next sentence decoder are computed analogously.

Objective. We define the probability of a context \( c_i \) given a sentence \( s_i \) as

\[ p_{\text{model}}(c_i | s_i; \theta) = p_{\text{model}}(s_{i-1} | s_i; \theta) \times p_{\text{model}}(s_{i+1} | s_i; \theta). \]  

where

\[ p_{\text{model}}(s_{i-1} | s_i; \theta) = \prod_{t=1}^{\tau_{i-1}} p(w_i^t | s_i; \theta) = \prod_{t=1}^{\tau_{i-1}} \exp(\mathbf{u}_{w_i^t} \cdot h_{i-1}^t) \]  

and similarly for \( p_{\text{model}}(s_{i+1} | s_i; \theta) \).

Similarly to Equation (5), MLE for the model parameters \( \theta \) can be found as

\[ \theta_{\text{MLE}} = \arg \min_\theta \left[ -\sum_{s_i \in D} \sum_{j \in \{i-1, i+1\}} \sum_{t=1}^{\tau_j} \left( \mathbf{u}_{w_j^t} \cdot h_j^t + \log \sum_{w' \in V} \exp(\mathbf{u}_{w'} \cdot h_j^t) \right) \right]. \]  

Using \( \oplus \) to denote vector concatenation, we note that

\[ \sum_{j \in \{i-1, i+1\}} \sum_{t=1}^{\tau_j} \mathbf{u}_{w_j^t} \cdot h_j^t = \left( \bigoplus_{j \in \{i-1, i+1\}} \tau_j \mathbf{u}_{w_j^t} \right) \cdot \left( \bigoplus_{j \in \{i-1, i+1\}} \tau_j h_j^t \right) = c_i \cdot h_i^D, \]  

where the sentence representation \( h_i^D \) is now an ordered concatenation of the hidden states of both decoders, and the context representation \( c_i \) is an ordered concatenation of the output embeddings of the context words. Hence we can come to the same conclusion as in the log-linear case, except we have order-sensitive representations as opposed to unordered ones. As before, \( h_i^D \) is forced to be similar under dot product to the context \( c_i \), and is made dissimilar to sequences of \( \mathbf{u}_{w'} \) that do not appear in the context.

The “transitivity” argument from Section 3.2 remains intact, except the decoder hidden state sequences might differ in length from sentence to sentence. To avoid this problem, we can formally treat them as infinite-dimensional vectors in \( \ell^2 \) with only a finite number of initial components occupied by the sequence and the rest set to zero. Alternatively, we can agree on the maximum sequence length, which in practice can be determined from the training corpus.

Regardless, the above space of unrolled concatenated decoder states, equipped with cosine similarity, is the optimal representation space for models with recurrent decoders. Consequently, this space should be a much better candidate for unsupervised similarity tasks, a fact we experimentally confirm in Section 5.

We refer to the method of accessing the decoder states at every time step as unrolling the decoder. Note that accessing the decoder output does not require re-architecting or retraining the model, giving a potential performance boost on unsupervised similarity tasks almost “for free”. We will demonstrate the effectiveness of this technique empirically in Section 5.

In practice, models such as SkipThought are evaluated in the space induced by the encoder, where cosine similarity is not an optimal measure with respect to the objective. However, by using \( D \)
Table 1: Performance of different architectures and sentence representations on unsupervised similarity tasks.

<table>
<thead>
<tr>
<th>Encoder</th>
<th>Decoder</th>
<th>STS12</th>
<th>STS13</th>
<th>STS14</th>
<th>STS15</th>
<th>STS16</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>BOW</td>
<td>0.466/0.496</td>
<td>0.376/0.414</td>
<td>0.478/0.482</td>
<td>0.424/0.454</td>
<td>0.552/0.586</td>
</tr>
<tr>
<td></td>
<td>RNN</td>
<td>0.323/0.357</td>
<td>0.320/0.319</td>
<td>0.345/0.345</td>
<td>0.402/0.409</td>
<td>0.373/0.408</td>
</tr>
<tr>
<td></td>
<td>RNN-mean</td>
<td>0.430/0.458</td>
<td>0.457/0.446</td>
<td>0.490/0.481</td>
<td>0.511/0.516</td>
<td>0.526/0.542</td>
</tr>
<tr>
<td></td>
<td>RNN-concat</td>
<td>0.419/0.445</td>
<td>0.426/0.414</td>
<td>0.466/0.452</td>
<td>0.497/0.503</td>
<td>0.511/0.529</td>
</tr>
<tr>
<td>BOW</td>
<td>RNN</td>
<td>0.497/0.517</td>
<td>0.526/0.520</td>
<td>0.576/0.561</td>
<td>0.604/0.605</td>
<td>0.592/0.592</td>
</tr>
<tr>
<td></td>
<td>RNN</td>
<td>0.508/0.526</td>
<td>0.483/0.489</td>
<td>0.575/0.562</td>
<td>0.644/0.641</td>
<td>0.585/0.585</td>
</tr>
<tr>
<td></td>
<td>RNN-mean</td>
<td>0.533/0.551</td>
<td>0.509/0.517</td>
<td>0.578/0.565</td>
<td>0.637/0.635</td>
<td>0.605/0.601</td>
</tr>
<tr>
<td></td>
<td>RNN-concat</td>
<td>0.521/0.540</td>
<td>0.491/0.498</td>
<td>0.561/0.554</td>
<td>0.627/0.625</td>
<td>0.584/0.581</td>
</tr>
</tbody>
</table>

to denote the decoder part of the model, the encoder space equipped with the induced similarity measure \( \langle \cdot, \cdot \rangle \) is again an optimal space. While in some ways this is a simple change of notation, it shows that a model may have many optimal spaces and they can be constructed using the layers of the network itself.

As a downside, concatenating hidden states of the decoder leads to very high dimensional vectors, which might be undesirable in applications. We find that averaging hidden states also works and actually improves the results slightly. Intuitively, this corresponds to destroying the word order information the decoder has learned. The performance gain might be due to the nature of the downstream tasks. Additionally, because of the way we unroll decoders during inference time, we observe the “softmax drifting effect,” which causes a drop in performance for longer sequences. We elaborate on this effect in Section 5.

4 EXPERIMENTAL SETUP

To support the theory in Section 3, we train several models with the same overall architecture but different combinations of encoders and decoders. We use the SentEval tool (Conneau et al. [2017]) to benchmark sentence embeddings on both supervised and supervised transfer tasks.

Models and training. Each model has an encoder for the current sentence, and decoders for the previous and next sentences. Using the notation ENC-DEC, we train RNN-RNN, RNN-BOW, BOW-BOW, and BOW-RNN. Note that RNN-RNN corresponds to SkipThought, and BOW-BOW to FastSent. In addition, for models that have RNN decoders, we unroll between 1 and 10 decoder hidden states and report on the best-performing one (with results for all given in Appendix C). We refer to these as *-RNN-concat for the concatenated states and *-RNN-mean for the averaged states. All models are trained on the Toronto Books Corpus (Zhu et al. [2015]), a dataset of 70 million ordered sentences from over 7,000 books. The sentences are pre-processed such that tokens are lower case and splittable on space.

Evaluation tasks. The supervised tasks in SentEval include paraphrase identification (MSRP) (Dolan et al. [2004]), movie review sentiment (MR) (Pang & Lee [2005]), product review sentiment (CR), (Hu & Liu [2004]), subjectivity (SUBJ) (Pang & Lee [2004]), opinion polarity (MPQA) (Wiebe et al. [2005]), and question type (TREC) (Voorhees [2002]). In addition, there are two supervised tasks on the SICK dataset, entailment and relatedness (denoted SICK-E and SICK-R) (Marelli et al. [2014]). For the supervised tasks, SentEval trains a logistic regression model with 10-fold cross-validation using the model’s embeddings as features.

The unsupervised similarity tasks are STS12-16 (Cer et al. [2017]; Agirre et al. [2012], [2013], [2014], [2015]; Agirre et al. [2016], which are scored in the same way as SICK-R but without train-
Table 2: Performance of different architectures and sentence representations on supervised transfer tasks. On each task, the highest performing setup for each decoder type is highlighted in bold and the highest performing setup overall is underlined. All reported values indicate test accuracy on the task, except for SICK-R where we report the Pearson correlation with human-provided scores. SICK-R and SICK-E scores for RNN-concat are omitted due to memory constraints. Note that the analysis in Section 5 is not readily applicable here, as instead of using a similarity measure in the representation space directly, the supervised transfer tasks train an entirely new model on top of the chosen representation.

<table>
<thead>
<tr>
<th>Encoder Decoder</th>
<th>MR</th>
<th>CR</th>
<th>MPQA</th>
<th>SUBJ</th>
<th>SST</th>
<th>TREC</th>
<th>MRPC</th>
<th>SICK-R</th>
<th>SICK-E</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOW</td>
<td>75.78</td>
<td>79.34</td>
<td>86.25</td>
<td>90.77</td>
<td>81.99</td>
<td>84.60</td>
<td>70.35</td>
<td>0.80</td>
<td>78.81</td>
</tr>
<tr>
<td>RNN</td>
<td>77.06</td>
<td>81.77</td>
<td><strong>88.59</strong></td>
<td><strong>92.56</strong></td>
<td><strong>82.65</strong></td>
<td>86.60</td>
<td>71.94</td>
<td>0.83</td>
<td><strong>81.10</strong></td>
</tr>
<tr>
<td>RNN-mean</td>
<td>76.55</td>
<td>81.03</td>
<td>87.35</td>
<td>92.29</td>
<td>81.11</td>
<td>84.80</td>
<td>73.61</td>
<td>0.84</td>
<td>78.22</td>
</tr>
<tr>
<td>RNN-concat</td>
<td>76.20</td>
<td>82.07</td>
<td>85.96</td>
<td>91.80</td>
<td>80.83</td>
<td><strong>87.20</strong></td>
<td>71.59</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>BOW</td>
<td>76.16</td>
<td>81.14</td>
<td>87.03</td>
<td>92.77</td>
<td>81.66</td>
<td>84.20</td>
<td>71.07</td>
<td>0.84</td>
<td>80.58</td>
</tr>
<tr>
<td>RNN</td>
<td>76.05</td>
<td><strong>82.07</strong></td>
<td>85.80</td>
<td>92.13</td>
<td>80.83</td>
<td>87.20</td>
<td>72.99</td>
<td>0.82</td>
<td>78.87</td>
</tr>
<tr>
<td>RNN-mean</td>
<td>75.85</td>
<td>81.30</td>
<td>85.54</td>
<td>90.80</td>
<td>80.12</td>
<td>84.00</td>
<td>71.13</td>
<td>0.81</td>
<td>77.76</td>
</tr>
<tr>
<td>RNN-concat</td>
<td><strong>77.27</strong></td>
<td>82.04</td>
<td><strong>88.74</strong></td>
<td>92.88</td>
<td><strong>81.82</strong></td>
<td><strong>89.60</strong></td>
<td>73.68</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Implementation and hyperparameters. Our goal is to study how different decoder types affect the performance of sentence embeddings on various tasks. To this end, we use identical hyperparameters and architecture for each model (except encoder and decoder types), allowing for a fair head-to-head comparison. Specifically, for RNN encoders and decoders we use a single layer GRU with layer normalisation ([Ba et al., 2016]). All the weights (including word embeddings) are initialised uniformly over $[-0.1, 0.1]$ and trained with Adam without weight decay or dropout ([Kingma & Ba, 2014]). Sentence length is clipped or zero-padded to 30 tokens and the end-of-sentence tokens are used throughout training and evaluation. Following [Kiros et al., 2015], we use a vocabulary-size of 20k, 620-dimensional word embeddings, and 2400 hidden units in RNN encoders / decoders.

5 Results

Performance across unsupervised similarity tasks is presented in Table 1 and performance across supervised transfer tasks is presented in Table 2. When the encoder is an RNN, the supervised transfer results validate our claims in Section 3.5. The results are less conclusive when the encoder is a BOW, see the caption of Table 1 for more details.

When we look at the performance on supervised transfer in Table 2 combined with the similarity results in Table 1, we see that the notion models cannot be good at both supervised and transfer tasks needs refining. Our results show that, for example, the raw encoder output for SkipThought (RNN-RNN) achieves strong performance on supervised transfer, whilst its mean decoder output (RNN-mean) achieves strong performance on supervised transfer. Instead, our results demonstrate that a single model can perform well on both types of transfer task if the representation spaces chosen for each task are allowed to be different.

Curiously, the unusual combination of a BOW encoder and concatenation of the RNN decoders leads to the best performance on most benchmarks, even slightly exceeding that of InferSent ([Conneau et al., 2017]).

Finally, the performance of the unrolled models is presented in Figure 1 with the numeric values of all tasks and unrolls up to 10 presented in Appendix C. We observe that the performance to peak at around 2-3 hidden states and fall off afterwards. In principle, one might expect the peak to be around the average sentence length of the corpus. A possible explanation of this behaviour is the “softmax drifting effect”. As there is no target sentence during inference time, we generate the word embedding for the next time step using the softmax output $\tilde{p}_{t-1}$ from the previous time step, i.e. $\tilde{v}_t = V^T \tilde{p}_{t-1}$, where $V$ is the input word embedding matrix. Given the ambiguity about what the surrounding sentences might be, a potentially softmax output might “drift” the sequence of $\tilde{v}_t$ away from the word embeddings expected by the decoder. It is likely that beam search would stabilise this behaviour, but further work is needed to understand this and other possible causes in detail.
Figure 1: Performance on the STS14 task depending on a number of unrolled hidden states of the decoders. In case of RNN encoder, RNN-RNN-mean at its peak matches the performance of RNN-BOW and both unrolling strategies strictly outperform RNN-RNN. In case of BOW encoder, only BOW-RNN-mean outperforms competing models (possibly because the BOW encoder is unable to preserve word order information).

6 Conclusion

In this work we have presented a simple explanation for the observed performance gap between FastSent (BOW-BOW) and SkipThought (RNN-RNN) architectures when using encoder output as a sentence embedder on unsupervised similarity tasks. Specifically, we note that the encoder-decoder training objective induces a similarity measure between embeddings on an optimal representation space. When the task uses the same similarity measure, we observe improved performance.

Assuming the use of cosine similarity, we show that the optimal representation space for FastSent is precisely its encoder output, whereas in the SkipThought case it is not, but is instead constructed by concatenating the decoder output states. The observed performance gap can then be explained by noting that previous uses of log-linear architectures for unsupervised similarity tasks correctly have leveraged their optimal representation space, but RNN architectures like SkipThought have not.

We then validate our claims by comparing the empirical performance of different architectures across transfer tasks. In general, we observe that unrolling the RNN decoder for different numbers of hidden states yields a performance that interpolates between the lower performance of the raw encoder output with RNN decoder and higher performance of the BOW decoder across all similarity tasks. This demonstrates how our insights into optimal similarity spaces can motivate novel ways of extracting vector embeddings from neural networks.

Ultimately, a good representation is one that makes a subsequent learning task easier. For unsupervised similarity tasks, either within or outside of the context of NLP, this essentially reduces to how well the model separates objects in the chosen representation space, and how appropriately the similarity measure compares objects in that space. Our findings lead us to the following practical advice: i) Use a simple model architecture where the optimal representation space is clear by construction, or ii) use an arbitrarily complex model architecture and analyse the objective function to reveal, for a given vector representation of choice, an appropriate similarity metric.

We hope that future work will utilise a careful understanding of what similarity means and how it is linked to the objective function, and that our analysis can be applied to help boost the performance of other complex models.
REFERENCES


A Optimising Objective

The task is to maximise the quantity $Q$ found in eq. [5]

$$Q = \sum_{(s_i, c_i) \in D} \sum_{w \in c_i} \left[ u_w \cdot h_i - \log \sum_{w' \in V_W} \exp (u_{w'} \cdot h_i) \right] = \sum_{(s_i, c_i) \in D} \sum_{w \in V_W} q_{iw},$$  \hfill (17)

where

$$q_{iw} = \log (x) - \log (x + y),$$  \hfill (18)

where we drop the sentence and word subscript on $x$ and $y$ for brevity (but in the following equations it is understood we are referring to a specific given word $w$ given specific sentence $s$), and

$$x = \exp (u_w \cdot h_i), \quad y = \sum_{w' \in V_W \setminus \{w\}} \exp (u_{w'} \cdot h_i).$$  \hfill (19)

We find the derivatives

$$\frac{\partial q_{iw}}{\partial x} = \frac{y}{x(x+y)} , \quad \frac{\partial q_{iw}}{\partial y} = -\frac{1}{x(x+y)},$$  \hfill (20)

and conclude that since both $x$ and $y$ are therefore positive, that for a given word $w$ and sentence $s_i$, the quantity $q_{iw}$ is made larger by

- Increasing $x$, leading to an increase in the dot product of the word present in the context with the context vector, and
- Reducing $y$, leading to a decrease in the dot products of all other words.

Performing this analysis across all words in a context leads to the maximisation of the dot products of the context representation $c_i$ with the sentence representation $h_i$

$$\sum_{w \in c_i} u_w \cdot h_i = c_i \cdot h_i$$

and a minimisation of the dot product of the sentence representation $h_i$ word vectors $u_{w'}$ that are not in the context $c_i$

$$\sum_{w \in c_i} \log \sum_{w' \in V_W \setminus \{w\}} \exp (u_{w'} \cdot h_i).$$

B The Original SkipThought Decoder

Kiros et al. (2015) additionally bias each decoder gate by the encoder output $h_i^\tau$.

$$r^t = \sigma \left( W_r^D v^t + U_r^D h_{t-1}^i + C_r h_i^\tau \right),$$  \hfill (21)

$$z^t = \sigma \left( W_z^D v^t + U_z^D h_{t-1}^i + C_z h_i^\tau \right),$$  \hfill (22)

$$\tilde{h}^t = \tanh \left[ W_v^D v^t + U_v^D (r^t \odot h_{t-1}^i) + Ch_i^\tau \right],$$  \hfill (23)

$$h = (1 - z^t) \odot h_{t-1}^i + z^t \odot \tilde{h}^t.$$  \hfill (24)

We suspect this change has substantial impact on similarity performance as the encoder contributes more directly to the decoder hidden states and is therefore “closer” to the optimal space.
## C Unrolled RNN Results

Table 3: Performance of different architectures and sentence representations on unsupervised similarity tasks with different length of unrolled sentence. On each task, for each encoder-decoder combination, the highest performing setup is highlighted in bold. All reported values indicate Pearson/Spearman correlation coefficients for the task.

<table>
<thead>
<tr>
<th>Unroll</th>
<th>STS12</th>
<th>STS13</th>
<th>STS14</th>
<th>STS15</th>
<th>STS16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.408/0.434</td>
<td>0.410/0.400</td>
<td>0.446/0.434</td>
<td>0.486/0.492</td>
<td>0.493/0.514</td>
</tr>
<tr>
<td>1</td>
<td>0.419/0.445</td>
<td>0.426/0.414</td>
<td>0.466/0.452</td>
<td>0.497/0.503</td>
<td>0.511/0.529</td>
</tr>
<tr>
<td>2</td>
<td>0.404/0.430</td>
<td>0.410/0.404</td>
<td>0.444/0.434</td>
<td>0.476/0.483</td>
<td>0.487/0.513</td>
</tr>
<tr>
<td>3</td>
<td>0.391/0.416</td>
<td>0.395/0.390</td>
<td>0.433/0.424</td>
<td>0.459/0.468</td>
<td>0.478/0.505</td>
</tr>
<tr>
<td>4</td>
<td>0.374/0.397</td>
<td>0.382/0.375</td>
<td>0.421/0.411</td>
<td>0.443/0.451</td>
<td>0.461/0.493</td>
</tr>
<tr>
<td>5</td>
<td>0.354/0.380</td>
<td>0.384/0.374</td>
<td>0.415/0.403</td>
<td>0.430/0.435</td>
<td>0.440/0.478</td>
</tr>
<tr>
<td>6</td>
<td>0.337/0.363</td>
<td>0.366/0.355</td>
<td>0.401/0.387</td>
<td>0.416/0.419</td>
<td>0.418/0.458</td>
</tr>
<tr>
<td>7</td>
<td>0.321/0.347</td>
<td>0.353/0.339</td>
<td>0.387/0.371</td>
<td>0.404/0.405</td>
<td>0.396/0.426</td>
</tr>
<tr>
<td>8</td>
<td>0.309/0.337</td>
<td>0.343/0.325</td>
<td>0.380/0.362</td>
<td>0.394/0.396</td>
<td>0.378/0.402</td>
</tr>
<tr>
<td>9</td>
<td>0.300/0.328</td>
<td>0.331/0.310</td>
<td>0.372/0.354</td>
<td>0.386/0.387</td>
<td>0.364/0.385</td>
</tr>
<tr>
<td>10</td>
<td>0.414/0.439</td>
<td>0.415/0.406</td>
<td>0.452/0.440</td>
<td>0.489/0.495</td>
<td>0.499/0.519</td>
</tr>
<tr>
<td>1</td>
<td>0.429/0.456</td>
<td>0.441/0.430</td>
<td>0.485/0.469</td>
<td>0.510/0.516</td>
<td>0.526/0.541</td>
</tr>
<tr>
<td>2</td>
<td>0.430/0.458</td>
<td>0.457/0.446</td>
<td>0.499/0.481</td>
<td>0.511/0.516</td>
<td>0.528/0.542</td>
</tr>
<tr>
<td>3</td>
<td>0.425/0.453</td>
<td>0.454/0.443</td>
<td>0.497/0.478</td>
<td>0.504/0.509</td>
<td>0.527/0.542</td>
</tr>
<tr>
<td>4</td>
<td>0.420/0.447</td>
<td>0.450/0.440</td>
<td>0.496/0.477</td>
<td>0.498/0.503</td>
<td>0.520/0.538</td>
</tr>
<tr>
<td>5</td>
<td>0.416/0.445</td>
<td>0.445/0.436</td>
<td>0.496/0.476</td>
<td>0.498/0.504</td>
<td>0.516/0.535</td>
</tr>
<tr>
<td>6</td>
<td>0.413/0.444</td>
<td>0.440/0.433</td>
<td>0.495/0.474</td>
<td>0.498/0.505</td>
<td>0.509/0.532</td>
</tr>
<tr>
<td>7</td>
<td>0.407/0.441</td>
<td>0.432/0.425</td>
<td>0.490/0.467</td>
<td>0.496/0.503</td>
<td>0.501/0.523</td>
</tr>
<tr>
<td>8</td>
<td>0.401/0.437</td>
<td>0.424/0.417</td>
<td>0.483/0.459</td>
<td>0.489/0.497</td>
<td>0.490/0.512</td>
</tr>
<tr>
<td>9</td>
<td>0.393/0.430</td>
<td>0.413/0.405</td>
<td>0.476/0.452</td>
<td>0.481/0.489</td>
<td>0.480/0.503</td>
</tr>
<tr>
<td>10</td>
<td>0.518/0.538</td>
<td>0.484/0.491</td>
<td>0.556/0.550</td>
<td>0.627/0.625</td>
<td>0.578/0.575</td>
</tr>
<tr>
<td>1</td>
<td>0.521/0.540</td>
<td>0.491/0.498</td>
<td>0.561/0.554</td>
<td>0.627/0.625</td>
<td>0.584/0.581</td>
</tr>
<tr>
<td>2</td>
<td>0.512/0.532</td>
<td>0.486/0.495</td>
<td>0.554/0.548</td>
<td>0.616/0.616</td>
<td>0.571/0.573</td>
</tr>
<tr>
<td>3</td>
<td>0.506/0.526</td>
<td>0.469/0.479</td>
<td>0.542/0.537</td>
<td>0.600/0.602</td>
<td>0.562/0.564</td>
</tr>
<tr>
<td>4</td>
<td>0.494/0.514</td>
<td>0.461/0.470</td>
<td>0.531/0.526</td>
<td>0.585/0.589</td>
<td>0.550/0.554</td>
</tr>
<tr>
<td>5</td>
<td>0.471/0.499</td>
<td>0.453/0.461</td>
<td>0.515/0.512</td>
<td>0.565/0.573</td>
<td>0.522/0.536</td>
</tr>
<tr>
<td>6</td>
<td>0.452/0.480</td>
<td>0.436/0.443</td>
<td>0.496/0.495</td>
<td>0.548/0.558</td>
<td>0.504/0.522</td>
</tr>
<tr>
<td>7</td>
<td>0.429/0.458</td>
<td>0.424/0.428</td>
<td>0.474/0.475</td>
<td>0.525/0.537</td>
<td>0.486/0.509</td>
</tr>
<tr>
<td>8</td>
<td>0.412/0.442</td>
<td>0.423/0.423</td>
<td>0.462/0.461</td>
<td>0.512/0.524</td>
<td>0.471/0.492</td>
</tr>
<tr>
<td>9</td>
<td>0.399/0.425</td>
<td>0.412/0.411</td>
<td>0.447/0.444</td>
<td>0.498/0.507</td>
<td>0.459/0.477</td>
</tr>
<tr>
<td>10</td>
<td>0.528/0.548</td>
<td>0.488/0.496</td>
<td>0.564/0.556</td>
<td>0.633/0.630</td>
<td>0.594/0.590</td>
</tr>
<tr>
<td>1</td>
<td>0.533/0.551</td>
<td>0.504/0.511</td>
<td>0.575/0.564</td>
<td>0.638/0.636</td>
<td>0.605/0.600</td>
</tr>
<tr>
<td>2</td>
<td>0.533/0.551</td>
<td>0.509/0.517</td>
<td>0.578/0.565</td>
<td>0.637/0.635</td>
<td>0.605/0.601</td>
</tr>
<tr>
<td>3</td>
<td>0.530/0.548</td>
<td>0.503/0.512</td>
<td>0.575/0.560</td>
<td>0.630/0.628</td>
<td>0.601/0.598</td>
</tr>
<tr>
<td>4</td>
<td>0.529/0.546</td>
<td>0.501/0.510</td>
<td>0.574/0.558</td>
<td>0.624/0.624</td>
<td>0.599/0.598</td>
</tr>
<tr>
<td>5</td>
<td>0.522/0.541</td>
<td>0.500/0.507</td>
<td>0.571/0.555</td>
<td>0.617/0.619</td>
<td>0.591/0.592</td>
</tr>
<tr>
<td>6</td>
<td>0.516/0.537</td>
<td>0.494/0.501</td>
<td>0.566/0.550</td>
<td>0.611/0.613</td>
<td>0.585/0.588</td>
</tr>
<tr>
<td>7</td>
<td>0.507/0.530</td>
<td>0.483/0.491</td>
<td>0.558/0.543</td>
<td>0.601/0.604</td>
<td>0.576/0.581</td>
</tr>
<tr>
<td>8</td>
<td>0.500/0.525</td>
<td>0.482/0.488</td>
<td>0.553/0.538</td>
<td>0.595/0.599</td>
<td>0.568/0.574</td>
</tr>
<tr>
<td>9</td>
<td>0.494/0.519</td>
<td>0.481/0.486</td>
<td>0.549/0.533</td>
<td>0.588/0.593</td>
<td>0.562/0.568</td>
</tr>
</tbody>
</table>