Skip-thought Vectors

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>As Published</td>
<td><a href="https://papers.nips.cc/paper/5950-skip-thought-vectors">https://papers.nips.cc/paper/5950-skip-thought-vectors</a></td>
</tr>
<tr>
<td>Publisher</td>
<td>Neural Information Processing Systems Foundation</td>
</tr>
<tr>
<td>Version</td>
<td>Final published version</td>
</tr>
<tr>
<td>Accessed</td>
<td>Sun Mar 31 03:09:34 EDT 2019</td>
</tr>
<tr>
<td>Citable Link</td>
<td><a href="http://hdl.handle.net/1721.1/113418">http://hdl.handle.net/1721.1/113418</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>Article is made available in accordance with the publisher's policy and may be subject to US copyright law. Please refer to the publisher's site for terms of use.</td>
</tr>
<tr>
<td>Detailed Terms</td>
<td></td>
</tr>
</tbody>
</table>
Skip-Thought Vectors

Ryan Kiros 1, Yukun Zhu 1, Ruslan Salakhutdinov 1,2, Richard S. Zemel 1,2
Antonio Torralba 3, Raquel Urtasun 1, Sanja Fidler 1
University of Toronto 1
Canadian Institute for Advanced Research 2
Massachusetts Institute of Technology 3

Abstract

We describe an approach for unsupervised learning of a generic, distributed sentence encoder. Using the continuity of text from books, we train an encoder-decoder model that tries to reconstruct the surrounding sentences of an encoded passage. Sentences that share semantic and syntactic properties are thus mapped to similar vector representations. We next introduce a simple vocabulary expansion method to encode words that were not seen as part of training, allowing us to expand our vocabulary to a million words. After training our model, we extract and evaluate our vectors with linear models on 8 tasks: semantic relatedness, paraphrase detection, image-sentence ranking, question-type classification and 4 benchmark sentiment and subjectivity datasets. The end result is an off-the-shelf encoder that can produce highly generic sentence representations that are robust and perform well in practice.

1 Introduction

Developing learning algorithms for distributed compositional semantics of words has been a long-standing open problem at the intersection of language understanding and machine learning. In recent years, several approaches have been developed for learning composition operators that map word vectors to sentence vectors including recursive networks [1], recurrent networks [2], convolutional networks [3, 4] and recursive-convolutional methods [5, 6] among others. All of these methods produce sentence representations that are passed to a supervised task and depend on a class label in order to backpropagate through the composition weights. Consequently, these methods learn high-quality sentence representations but are tuned only for their respective task. The paragraph vector of [7] is an alternative to the above models in that it can learn unsupervised sentence representations by introducing a distributed sentence indicator as part of a neural language model. The downside is that test time, inference needs to be performed to compute a new vector.

In this paper we abstract away from the composition methods themselves and consider an alternative loss function that can be applied with any composition operator. We consider the following question: is there a task and a corresponding loss that will allow us to learn highly generic sentence representations? We give evidence for this by proposing a model for learning high-quality sentence vectors without a particular supervised task in mind. Using word vector learning as inspiration, we propose an objective function that abstracts the skip-gram model of [8] to the sentence level. That is, instead of using a word to predict its surrounding context, we instead encode a sentence to predict the sentences around it. Thus, any composition operator can be substituted as a sentence encoder and only the objective function becomes modified. Figure 1 illustrates the model. We call our model skip-thoughts and vectors induced by our model are called skip-thought vectors.

Our model depends on having a training corpus of contiguous text. We chose to use a large collection of novels, namely the BookCorpus dataset [9] for training our models. These are free books written by yet unpublished authors. The dataset has books in 16 different genres, e.g., Romance (2,865 books), Fantasy (1,479), Science fiction (786), Teen (430), etc. Table 1 highlights the summary statistics of the book corpus. Along with narratives, books contain dialogue, emotion and a wide range of interaction between characters. Furthermore, with a large enough collection the training set is not biased towards any particular domain or application. Table 2 shows nearest neighbours
Given a tuple \((s_{i-1}, s_i, s_{i+1})\) of contiguous sentences, with \(s_i\) the \(i\)-th sentence of a book, the sentence \(s_i\) is encoded and tries to reconstruct the previous sentence \(s_{i-1}\) and next sentence \(s_{i+1}\). In this example, the input is the sentence triplet I got back home, I could see the cat on the steps, This was strange. Unattached arrows are connected to the encoder output. Colors indicate which components share parameters. \(\langle\text{eos}\rangle\) is the end of sentence token.

Table 1: Summary statistics of the BookCorpus dataset \cite{9}. We use this corpus to train our model.

We evaluate our vectors in a newly proposed setting: after learning skip-thoughts, freeze the model and use the encoder as a generic feature extractor for arbitrary tasks. In our experiments we consider 8 tasks: semantic-relatedness, paraphrase detection, image-sentence ranking and 5 standard classification benchmarks. In these experiments, we extract skip-thought vectors and train linear models to evaluate the representations directly, without any additional fine-tuning. As it turns out, skip-thoughts yield generic representations that perform robustly across all tasks considered.

One difficulty that arises with such an experimental setup is being able to construct a large enough word vocabulary to encode arbitrary sentences. For example, a sentence from a Wikipedia article might contain nouns that are highly unlikely to appear in our book vocabulary. We solve this problem by learning a mapping that transfers word representations from one model to another. Using pre-trained word2vec representations learned with a continuous bag-of-words model \cite{8}, we learn a linear mapping from a word in word2vec space to a word in the encoder’s vocabulary space. The mapping is learned using all words that are shared between vocabularies. After training, any word that appears in word2vec can then get a vector in the encoder word embedding space.

2 Approach

2.1 Inducing skip-thought vectors

We treat skip-thoughts in the framework of encoder-decoder models \cite{1}. That is, an encoder maps words to a sentence vector and a decoder is used to generate the surrounding sentences. Encoder-decoder models have gained a lot of traction for neural machine translation. In this setting, an encoder is used to map e.g. an English sentence into a vector. The decoder then conditions on this vector to generate a translation for the source English sentence. Several choices of encoder-decoder pairs have been explored, including ConvNet-RNN \cite{10}, RNN-RNN \cite{11} and LSTM-LSTM \cite{12}. The source sentence representation can also dynamically change through the use of an attention mechanism \cite{13} to take into account only the relevant words for translation at any given time. In our model, we use an RNN encoder with GRU \cite{14} activations and an RNN decoder with a conditional GRU. This model combination is nearly identical to the RNN encoder-decoder of \cite{11} used in neural machine translation. GRU has been shown to perform as well as LSTM \cite{14} on sequence modelling tasks \cite{14} while being conceptually simpler. GRU units have only 2 gates and do not require the use of a cell. While we use RNNs for our model, any encoder and decoder can be used so long as we can backpropagate through it.

Assume we are given a sentence tuple \((s_{i-1}, s_i, s_{i+1})\). Let \(w_t^i\) denote the \(t\)-th word for sentence \(s_i\) and let \(x_t^i\) denote its word embedding. We describe the model in three parts: the encoder, decoder and objective function.

**Encoder.** Let \(w_1^i, \ldots, w_N^i\) be the words in sentence \(s_i\) where \(N\) is the number of words in the sentence. At each time step, the encoder produces a hidden state \(h_t^i\) which can be interpreted as the representation of the sequence \(w_1^i, \ldots, w_t^i\). The hidden state \(h_N^i\) thus represents the full sentence.

1 A preliminary version of our model was developed in the context of a computer vision application \cite{9}.
he ran his hand inside his coat, double-checking that the unopened letter was still there.
he slipped his hand between his coat and his shirt, where the folded copies lay in a brown envelope.

although she could tell he had n’t been too invested in any of their other chitchat, he seemed genuinely curious about this.
although he had n’t been following her career with a microscope, he’d definitely taken notice of her appearances.

an annoying buzz started to ring in my ears, becoming louder and louder as my vision began to swim.
a weighty pressure landed on my lungs and my vision blurred at the edges, threatening my consciousness altogether.

if he had a weapon, he could maybe take out their last imp, and then beat up errol and vanessa.
if he could ram them from behind, send them sailing over the far side of the levee, he had a chance of stopping them.
then, with a stroke of luck, they saw the pair head together towards the portaloos.

then, from out back of the house, they heard a horse scream probably in answer to a pair of sharp spurs digging deep into its flanks.

Table 2: In each example, the first sentence is a query while the second sentence is its nearest neighbour. Nearest neighbours were scored by cosine similarity from a random sample of 500,000 sentences from our corpus.

To encode a sentence, we iterate the following sequence of equations (dropping the subscript $i$):

$r^t = \sigma(W^r x^t + U^r h^{t-1})$ \hspace{1cm} (1)

$z^t = \sigma(W^z x^t + U^z h^{t-1})$ \hspace{1cm} (2)

$h^t = \tanh(W x^t + U (r^t \odot h^{t-1}))$ \hspace{1cm} (3)

$h^t = (1-z^t) \odot h^{t-1} + z^t \odot \bar{h}^t$ \hspace{1cm} (4)

where $\bar{h}^t$ is the proposed state update at time $t$, $z^t$ is the update gate, $r_t$ is the reset gate ($\odot$) denotes a component-wise product. Both update gates takes values between zero and one.

**Decoder.** The decoder is a neural language model which conditions on the encoder output $h_i$. The computation is similar to that of the encoder except we introduce matrices $C_z$, $C_r$ and $C$ that are used to bias the update gate, reset gate and hidden state computation by the sentence vector. One decoder is used for the next sentence $s_{i+1}$, while a second decoder is used for the previous sentence $s_{i-1}$. Separate parameters are used for each decoder with the exception of the vocabulary matrix $V$, which is the weight matrix connecting the decoder’s hidden state for computing a distribution over words. In what follows we describe the decoder for the next sentence $s_{i+1}$ although an analogous computation is used for the previous sentence $s_{i-1}$. Let $h_{i+1}^t$ denote the hidden state of the decoder at time $t$. Decoding involves iterating through the following sequence of equations (dropping the subscript $i+1$):

$r^t = \sigma(W^d x^t + U^d h^{t-1} + C_i h_i)$ \hspace{1cm} (5)

$z^t = \sigma(W^z x^t + U^z h^{t-1} + C_i h_i)$ \hspace{1cm} (6)

$h^t = \tanh(W^d x^t + U^d (r^t \odot h^{t-1}) + Ch_i)$ \hspace{1cm} (7)

$h_{i+1}^t = (1-z^t) \odot h^{t-1} + z^t \odot \bar{h}^t$ \hspace{1cm} (8)

Given $h_{i+1}^t$, the probability of word $w_{i+1}^t$ given the previous $t-1$ words and the encoder vector is

$P(w_{i+1}^t|w_{i+1}^{t-1}, h_i) \propto \exp(v_{w_{i+1}^t} h_{i+1}^t)$ \hspace{1cm} (9)

where $v_{w_{i+1}^t}$ denotes the row of $V$ corresponding to the word of $w_{i+1}^t$. An analogous computation is performed for the previous sentence $s_{i-1}$.

**Objective.** Given a tuple $(s_{i-1}, s_i, s_{i+1})$, the objective optimized is the sum of the log-probabilities for the forward and backward sentences conditioned on the encoder representation:

$\sum_{t} \log P(w_{i+1}^t|w_{i+1}^{t-1}, h_i) + \sum_{t} \log P(w_{i-1}^t|w_{i-1}^{t-1}, h_i)$ \hspace{1cm} (10)

The total objective is the above summed over all such training tuples.
2.2 Vocabulary expansion

We now describe how to expand our encoder’s vocabulary to words it has not seen during training. Suppose we have a model that was trained to induce word representations, such as word2vec. Let $V_{w2v}$ denote the word embedding space of these word representations and let $V_{rnn}$ denote the RNN word embedding space. We assume the vocabulary of $V_{w2v}$ is much larger than that of $V_{rnn}$. Our goal is to construct a mapping $f : V_{w2v} \rightarrow V_{rnn}$ parameterized by a matrix $W$ such that $v' = Wv$ for $v \in V_{w2v}$ and $v' \in V_{rnn}$. Inspired by [15], which learned linear mappings between translation word spaces, we solve an un-regularized L2 linear regression loss for the matrix $W$. Thus, any word from $V_{w2v}$ can now be mapped into $V_{rnn}$ for encoding sentences.

3 Experiments

In our experiments, we evaluate the capability of our encoder as a generic feature extractor after training on the BookCorpus dataset. Our experimentation setup on each task is as follows:

- Using the learned encoder as a feature extractor, extract skip-thought vectors for all sentences.
- If the task involves computing scores between pairs of sentences, compute component-wise features between pairs. This is described in more detail specifically for each experiment.
- Train a linear classifier on top of the extracted features, with no additional fine-tuning or backpropagation through the skip-thoughts model.

We restrict ourselves to linear classifiers for two reasons. The first is to directly evaluate the representation quality of the computed vectors. It is possible that additional performance gains can be made throughout our experiments with non-linear models but this falls out of scope of our goal. Furthermore, it allows us to better analyze the strengths and weaknesses of the learned representations. The second reason is that reproducibility now becomes very straightforward.

3.1 Details of training

To induce skip-thought vectors, we train two separate models on our book corpus. One is a unidirectional encoder with 2400 dimensions, which we subsequently refer to as uni-skip. The other is a bidirectional model with forward and backward encoders of 1200 dimensions each. This model contains two encoders with different parameters: one encoder is given the sentence in correct order, while the other is given the sentence in reverse. The outputs are then concatenated to form a 2400 dimensional vector. We refer to this model as bi-skip. For training, we initialize all recurrent matrices with orthogonal initialization [16]. Non-recurrent weights are initialized from a uniform distribution in [-0.1,0.1]. Mini-batches of size 128 are used and gradients are clipped if the norm of the parameter vector exceeds 10. We used the Adam algorithm [17] for optimization. Both models were trained for roughly two weeks. As an additional experiment, we also report experimental results using a combined model, consisting of the concatenation of the vectors from uni-skip and bi-skip, resulting in a 4800 dimensional vector. We refer to this model throughout as combine-skip.

After our models are trained, we then employ vocabulary expansion to map word embeddings into the RNN encoder space. The publicly available CBOW word vectors are used for this purpose. The skip-thought vectors are trained with a vocabulary size of 20,000 words. After removing multiple word examples from the CBOW model, this results in a vocabulary size of 930,911 words. Thus even though our skip-thoughts model was trained with only 20,000 words, after vocabulary expansion we can now successfully encode 930,911 possible words.

Since our goal is to evaluate skip-thoughts as a general feature extractor, we keep text pre-processing to a minimum. When encoding new sentences, no additional preprocessing is done other than basic tokenization. This is done to test the robustness of our vectors. As an additional baseline, we also consider the mean of the word vectors learned from the uni-skip model. We refer to this baseline as bow. This is to determine the effectiveness of a standard baseline trained on the BookCorpus.

3.2 Semantic relatedness

Our first experiment is on the SemEval 2014 Task 1: semantic relatedness SICK dataset [30]. Given two sentences, our goal is to produce a score of how semantically related these sentences are, based on human generated scores. Each score is the average of 10 different human annotators. Scores take values between 1 and 5. A score of 1 indicates that the sentence pair is not at all related, while
The next task we consider is paraphrase detection on the Microsoft Research Paraphrase Corpus. On this task, two sentences are given and one must predict whether or not they are a score of 5 indicates they are highly related. The dataset comes with a predefined split of 4500 training pairs, 500 development pairs and 4927 testing pairs. All sentences are derived from existing image and video annotation datasets. The evaluation metrics are Pearson’s $r$, Spearman’s $\rho$, and mean squared error.

Given the difficulty of this task, many existing systems employ a large amount of feature engineering and additional resources. Thus, we test how well our learned representations fair against heavily engineered pipelines. Recently, [22] showed that learning representations with LSTM or Tree-LSTM for the task at hand is able to outperform these existing systems. We take this one step further and see how well our vectors learned from a completely different task are able to capture semantic relatedness when only a linear model is used on top to predict scores.

To represent a sentence pair, we use two features. Given two skip-thought vectors $u$ and $v$, we compute their component-wise product $u \cdot v$ and their absolute difference $|u - v|$ and concatenate them together. These two features were also used by [22]. To predict a score, we use the same setup as [22]. Let $r^T = [1, \ldots, 5]$ be an integer vector from 1 to 5. We compute a distribution $p$ as a function of prediction scores $y$ given by $p_i = y - [y]$ if $i = [y] + 1$, $p_i = [y] - y + 1$ if $i = [y]$ and 0 otherwise. These then become our targets for a logistic regression classifier. At test time, given new sentence pairs we first compute targets $\hat{p}$ and then compute the related score as $r^T \hat{p}$. As an additional comparison, we also explored appending features derived from an image-sentence embedding model trained on COCO (see section 3.4). Given vectors $u$ and $v$, we obtain vectors $u' = u \cdot v$ and $v' = v \cdot u$ from the learned linear embedding model and compute features $u' \cdot v'$ and $|u' - v'|$. These are then concatenated to the existing features.

Table 3 (left) presents our results. First, we observe that our models are able to outperform all previous systems from the SemEval 2014 competition. It highlights that skip-thought vectors learn representations that are well suited for semantic relatedness. Our results are comparable to LSTMs whose representations are trained from scratch on this task. Only the dependency tree-LSTM of [22] performs better than our results. We note that the dependency tree-LSTM relies on parsers whose training data is very expensive to collect and does not exist for all languages. We also observe using features learned from an image-sentence embedding model on COCO gives an additional performance boost, resulting in a model that performs on par with the dependency tree-LSTM. To get a feel for the model outputs, Table 4 shows example cases of test set pairs. Our model is able to accurately predict relatedness on many challenging cases. On some examples, it fails to pick up on small distinctions that drastically change a sentence meaning, such as tricks on a motorcycle versus tricking a person on a motorcycle.

### 3.3 Paraphrase detection

The next task we consider is paraphrase detection on the Microsoft Research Paraphrase Corpus [31]. On this task, two sentences are given and one must predict whether or not they are...
Table 4: Example predictions from the SICK test set. GT is the ground truth relatedness, scored between 1 and 5. The last few results show examples where slight changes in sentence structure result in large changes in relatedness which our model was unable to score correctly.

<table>
<thead>
<tr>
<th>Sentence 1</th>
<th>Sentence 2</th>
<th>GT</th>
<th>pred</th>
</tr>
</thead>
<tbody>
<tr>
<td>A little girl is looking at a woman in costume</td>
<td>A young girl is looking at a woman in costume</td>
<td>4.7</td>
<td>4.5</td>
</tr>
<tr>
<td>A little girl is looking at a woman in costume</td>
<td>The little girl is looking at a man in costume</td>
<td>3.8</td>
<td>4.0</td>
</tr>
<tr>
<td>A little girl is looking at a woman in costume</td>
<td>A little girl in costume looks like a woman</td>
<td>2.9</td>
<td>3.5</td>
</tr>
<tr>
<td>A sea turtle is hunting for fish</td>
<td>A sea turtle is hunting for food</td>
<td>4.5</td>
<td>4.5</td>
</tr>
<tr>
<td>A sea turtle is not hunting for fish</td>
<td>A sea turtle is hunting for fish</td>
<td>3.4</td>
<td>3.8</td>
</tr>
<tr>
<td>A man is driving a car</td>
<td>The car is being driven by a man</td>
<td>5</td>
<td>4.9</td>
</tr>
<tr>
<td>There is no man driving the car</td>
<td>A man is driving a car</td>
<td>3.6</td>
<td>3.5</td>
</tr>
<tr>
<td>A large duck is flying over a rocky stream</td>
<td>A duck, which is large, is flying over a rocky stream</td>
<td>4.8</td>
<td>4.9</td>
</tr>
<tr>
<td>A large duck is flying over a rocky stream</td>
<td>A large stream is full of rocks, ducks and flies</td>
<td>2.7</td>
<td>3.1</td>
</tr>
<tr>
<td>A person is performing acrobatics on a motorcycle</td>
<td>A person is performing tricks on a motorcycle</td>
<td>4.3</td>
<td>4.4</td>
</tr>
<tr>
<td>A person is performing tricks on a motorcycle</td>
<td>The performer is tricking a person on a motorcycle</td>
<td>2.6</td>
<td>4.4</td>
</tr>
<tr>
<td>Someone is pouring ingredients into a pot</td>
<td>Someone is adding ingredients to a pot</td>
<td>4.4</td>
<td>4.0</td>
</tr>
<tr>
<td>Nobody is pouring ingredients into a pot</td>
<td>Someone is pouring ingredients into a pot</td>
<td>3.5</td>
<td>4.2</td>
</tr>
<tr>
<td>Someone is pouring ingredients into a pot</td>
<td>A man is removing vegetables from a pot</td>
<td>2.4</td>
<td>3.6</td>
</tr>
</tbody>
</table>

Table 5: COCO test-set results for image-sentence retrieval experiments. R@K is Recall@K (high is good). Med r is the median rank (low is good).

<table>
<thead>
<tr>
<th>Model</th>
<th>COCO Retrieval</th>
<th>Image Annotation</th>
<th>Image Search</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R@1</td>
<td>R@5</td>
<td>R@10</td>
</tr>
<tr>
<td>Random Ranking</td>
<td>0.1</td>
<td>0.6</td>
<td>1.1</td>
</tr>
<tr>
<td>DVSA [12]</td>
<td>38.4</td>
<td>69.6</td>
<td>83.3</td>
</tr>
<tr>
<td>GMM+HGLMM [33]</td>
<td>39.4</td>
<td>67.9</td>
<td>80.9</td>
</tr>
<tr>
<td>m-RNN [34]</td>
<td>41.0</td>
<td>73.0</td>
<td>83.5</td>
</tr>
<tr>
<td>bow</td>
<td>33.6</td>
<td>65.8</td>
<td>79.7</td>
</tr>
<tr>
<td>uni-skip</td>
<td>30.6</td>
<td>64.5</td>
<td>79.8</td>
</tr>
<tr>
<td>bi-skip</td>
<td>32.7</td>
<td>67.3</td>
<td>79.6</td>
</tr>
<tr>
<td>combine-skip</td>
<td>33.8</td>
<td>67.7</td>
<td>82.1</td>
</tr>
</tbody>
</table>

Table 3 (right) presents our results, from which we can observe the following: (1) skip-thoughts alone outperform recursive nets with dynamic pooling when no hand-crafted features are used, (2) when other features are used, recursive nets with dynamic pooling works better, and (3) when skip-thoughts are combined with basic pairwise statistics, it becomes competitive with the state-of-the-art which incorporate much more complicated features and hand-engineering. This is a promising result as many of the sentence pairs have very fine-grained details that signal if they are paraphrases.

3.4 Image-sentence ranking

We next consider the task of retrieving images and their sentence descriptions. For this experiment, we use the Microsoft COCO dataset which is the largest publicly available dataset of images with high-quality sentence descriptions. Each image is annotated with 5 captions, each from different annotators. Following previous work, we consider two tasks: image annotation and image search. For image annotation, an image is presented and sentences are ranked based on how well they describe the query image. The image search task is the reverse: given a caption, we retrieve images that are a good fit to the query. The training set comes with over 80,000 images each with 5 captions. For development and testing we use the same splits as . The development and test sets each contain 1000 images and 5000 captions. Evaluation is performed using Recall@K, namely the mean number of images for which the correct caption is ranked within the top-K retrieved results.
(and vice-versa for sentences). We also report the median rank of the closest ground truth result from the ranked list.

The best performing results on image-sentence ranking have all used RNNs for encoding sentences, where the sentence representation is learned jointly. Recently, [33] showed that by using Fisher vectors for representing sentences, linear CCA can be applied to obtain performance that is as strong as using RNNs for this task. Thus the method of [33] is a strong baseline to compare our sentence representations with. For our experiments, we represent images using 4096-dimensional OxfordNet features from their 19-layer model [36]. For sentences, we simply extract skip-thought vectors for each caption. The training objective we use is a pairwise ranking loss that has been previously used by many other methods. The only difference is the scores are computed using only linear transformations of image and sentence inputs. The loss is given by:

\[ \sum \sum \max \{0, \alpha - s(Ux, Vy) + s(Ux, Vy_k)\} + \sum \sum \max \{0, \alpha - s(Vy, Ux) + s(Vy, Ux_k)\}, \]

where \(x\) is an image vector, \(y\) is the skip-thought vector for the groundtruth sentence, \(y_k\) are vectors for constrastive (incorrect) sentences and \(s(\cdot, \cdot)\) is the image-sentence score. Cosine similarity is used for scoring. The model parameters are \(\{U, V\}\) where \(U\) is the image embedding matrix and \(V\) is the sentence embedding matrix. In our experiments, we use a 1000 dimensional embedding, margin \(\alpha = 0.2\) and \(k = 50\) contrastive terms. We trained for 15 epochs and saved our model anytime the performance improved on the development set.

Table 5 illustrates our results on this task. Using skip-thought vectors for sentences, we get performance that is on par with both [32] and [33] except for R@1 on image annotation, where other methods perform much better. Our results indicate that skip-thought vectors are representative enough to capture image descriptions without having to learn their representations from scratch. Combined with the results of [33], it also highlights that simple, scalable embedding techniques perform very well provided that high-quality image and sentence vectors are available.

### 3.5 Classification benchmarks

For our final quantitative experiments, we report results on several classification benchmarks which are commonly used for evaluating sentence representation learning methods.

We use 5 datasets: movie review sentiment (MR), customer product reviews (CR), subjectivity/objectivity classification (SUBJ), opinion polarity (MPQA) and question-type classification (TREC). On all datasets, we simply extract skip-thought vectors and train a logistic regression classifier on top. 10-fold cross-validation is used for evaluation on the first 4 datasets, while TREC has a pre-defined train/test split. We tune the L2 penalty using cross-validation (and thus use a nested cross-validation for the first 4 datasets).

<table>
<thead>
<tr>
<th>Method</th>
<th>MR</th>
<th>CR</th>
<th>SUBJ</th>
<th>MPQA</th>
<th>TREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB-SVM [37]</td>
<td>79.4</td>
<td>81.8</td>
<td>93.2</td>
<td>86.3</td>
<td></td>
</tr>
<tr>
<td>MNB [37]</td>
<td>79.0</td>
<td>80.0</td>
<td>93.6</td>
<td>86.3</td>
<td></td>
</tr>
<tr>
<td>cBoW [38]</td>
<td>77.2</td>
<td>79.9</td>
<td>91.5</td>
<td>86.4</td>
<td>87.3</td>
</tr>
<tr>
<td>GConv [3]</td>
<td>76.3</td>
<td>81.3</td>
<td>89.5</td>
<td>84.5</td>
<td>88.4</td>
</tr>
<tr>
<td>RNN [6]</td>
<td>77.2</td>
<td>82.3</td>
<td>93.7</td>
<td>90.1</td>
<td>90.2</td>
</tr>
<tr>
<td>BRNN [6]</td>
<td>82.3</td>
<td>82.6</td>
<td>94.2</td>
<td>90.3</td>
<td>91.0</td>
</tr>
<tr>
<td>CNN [3]</td>
<td>81.5</td>
<td>85.0</td>
<td>93.4</td>
<td>89.6</td>
<td>93.6</td>
</tr>
<tr>
<td>AdaSent [6]</td>
<td>83.1</td>
<td>86.3</td>
<td>95.5</td>
<td>93.3</td>
<td>92.4</td>
</tr>
<tr>
<td>Paragraph-vector [3]</td>
<td>74.8</td>
<td>78.1</td>
<td>90.8</td>
<td>74.2</td>
<td>91.8</td>
</tr>
</tbody>
</table>

Table 6: Classification accuracies on several standard benchmarks. Results are grouped as follows: (a) bag-of-words models; (b) supervised compositional models; (c) Paragraph Vector (unsupervised learning of sentence representations); (d) ours. Best results overall are **bold** while best results outside of group (b) are *underlined.*

On these tasks, properly tuned bag-of-words models have been shown to perform exceptionally well. In particular, the NB-SVM of [37] is a fast and robust performer on these tasks. Skip-thought vectors potentially give an alternative to these baselines being just as fast and easy to use. For an additional comparison, we also see to what effect augmenting skip-thoughts with bigram Naive Bayes (NB) features improves performance.

Table 6 presents our results. On most tasks, skip-thoughts performs about as well as the bag-of-words baselines but fails to improve over methods whose sentence representations are learned directly for the task at hand. This indicates that for tasks like sentiment classification, tuning the representations, even on small datasets, are likely to perform better than learning a generic unsupervised
3.6 Visualizing skip-thoughts

As a final experiment, we applied t-SNE [38] to skip-thought vectors extracted from TREC, SUBJ and SICK datasets and the visualizations are shown in Figure 2. For the SICK visualization, each point represents a sentence pair, computed using the concatenation of component-wise and absolute difference of features. Even without the use of relatedness labels, skip-thought vectors learn to accurately capture this property.

4 Conclusion

We evaluated the effectiveness of skip-thought vectors as an off-the-shelf sentence representation with linear classifiers across 8 tasks. Many of the methods we compare against were only evaluated on 1 task. The fact that skip-thought vectors perform well on all tasks considered highlight the robustness of our representations.

We believe our model for learning skip-thought vectors only scratches the surface of possible objectives. Many variations have yet to be explored, including (a) deep encoders and decoders, (b) larger context windows, (c) encoding and decoding paragraphs, (d) other encoders, such as convnets. It is likely the case that more exploration of this space will result in even higher quality representations.

Acknowledgments

We thank Geoffrey Hinton for suggesting the name skip-thoughts. We also thank Felix Hill, Kelvin Xu, Kyunghyun Cho and Ilya Sutskever for valuable comments and discussion. This work was supported by NSERC, Samsung, CIFAR, Google and ONR Grant N00014-14-1-0232.

References


[23] Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. Grounded compositional semantics for finding and describing images with sentences. IACL, 2014.


