Siamese Recurrent Architectures for Learning Sentence Similarity

Jonas Mueller
Computer Science
& Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Aditya Thyagarajan
Department of Computer Science
and Engineering
M. S. Ramaiah Institute of Technology

Abstract

We present a siamese adaptation of the Long Short-Term Memory (LSTM) network for labeled data comprised of pairs of variable-length sequences. Our model is applied to assess semantic similarity between sentences, where we exceed state of the art, outperforming carefully handcrafted features and recently proposed neural network systems of greater complexity. For these applications, we provide word-embedding vectors supplemented with synonymic information to the LSTMs, which use a fixed size vector to encode the underlying meaning expressed in a sentence (irrespective of the particular wording/syntax). By restricting subsequent operations to rely on a simple Manhattan metric, we compel the sentence representations learned by our model to form a highly structured space whose geometry reflects complex semantic relationships. Our results are the latest in a line of findings that showcase LSTMs as powerful language models capable of tasks requiring intricate understanding.

Introduction

Text understanding and information retrieval are important tasks which may be greatly enhanced by modeling the underlying semantic similarity between sentences/phrases. In particular, a good model should not be susceptible to variations of wording/syntax used to express the same idea. Learning such a semantic textual similarity metric has thus generated a great deal of research interest (Marelli et al. 2014). However, this remains a hard problem, because labeled data is scarce, sentences have both variable length and complex structure, and bag-of-words/tf-IDF models, while dominant in natural language processing (NLP), are limited in this context by their inherent term-specificity (c.f. Mihalcea, Corley, and Strapparava 2006).

As an alternative to these ideas, Mikolov et al. (2013) and others have demonstrated the effectiveness of neural word representations for analogies and other NLP tasks. Recently, interests have shifted toward extensions of these ideas beyond the individual word-level to larger bodies of text such as sentences, where a mapping is learned to represent each sentence as a fixed-length vector (Kiros et al. 2015; Tai, Socher, and Manning 2015; Le and Mikolov 2014).

Naturally suited for variable-length inputs like sentences, recurrent neural networks (RNN), especially the Long Short-Term Memory model of Hochreiter and Schmidhuber (1997), have been particularly successful in this setting for tasks such as text classification (Graves 2012) and language translation (Sutskever, Vinyals, and Le 2014). RNNs adapt standard feedforward neural networks for sequence data $(x_1, \ldots, x_T)$, where at each $t \in \{1, \ldots, T\}$, updates to a hidden-state vector h_t are performed via

text representation

$$h_t = \text{sigmoid}\left(Wx_t + Uh_{t-1}\right) \tag{1}$$

While Siegelmann and Sontag (1995) have shown that the basic RNN is Turing-complete, optimization of the weight-matrices is difficult because its backpropagated gradients become vanishingly small over long sequences. Practically, the LSTM is superior to basic RNNs for learning long range dependencies through its use of memory cell units that can store/access information across lengthy input sequences. Like RNNs, the LSTM sequentially updates a hidden-state representation, but these steps also rely on a memory cell containing four components (which are real-valued vectors): a memory state c_t, an output gate o_t that determines how the memory state affects other units, as well as an input (and forget) gate i_t (and f_t) that controls what gets stored in (and omitted from) memory based on each new input and the current state. Below are the updates performed at each $t \in \{1, \ldots, T\}$ in an LSTM parameterized by weight matrices $W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o$ and bias-vectors b_i, b_f, b_c, b_o:

$$i_t = \text{sigmoid}\left(W_i x_t + U_i h_{t-1} + b_i\right) \tag{2}$$
$$f_t = \text{sigmoid}\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{3}$$
$$\widetilde{c}_t = \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) \tag{4}$$
$$c_t = i_t \odot \widetilde{c}_t + f_t \odot c_{t-1} \tag{5}$$
$$o_t = \text{sigmoid}\left(W_o x_t + U_o h_{t-1} + b_o\right) \tag{6}$$
$$h_t = o_t \odot \tanh(c_t) \tag{7}$$

A more thorough exposition of the LSTM model and its variants is provided by Graves (2012) and Greff et al. (2015).

Although the success of LSTM language models eludes current theoretical understanding, Sutskever, Vinyals, and Le (2014) empirically validate the intuition that an effectively trained network maps each sentence onto a fixed-length vector which encodes the underlying meaning expressed in the text. Recent works have proposed many

other LSTM variants such as the simple gated recurrent unit (GRU) of Cho et al. (2014). After extensive empirical analysis, Greff et al. (2015) question whether any of the proposed modifications reliably outperforms the basic LSTM model. Broader architectural revisions of the sequential RNN have also been introduced for modeling text, including bidirectional, multi-layer (Graves 2012), and recursive tree-structures (Socher 2014).

In this work, we show that, given enough data, a simple adaptation of the LSTM may be trained on paired examples to learn a highly structured space of sentence-representations that captures rich semantics. Our results, like the language translation experiments of Sutskever, Vinyals, and Le (2014), demonstrate that standard LSTMs can perform remarkably well on seemingly complex NLP problems. Despite its simplicity, our approach exhibits superior performance than the current state of the art for evaluating similarity between sentences.

Formally, we consider a supervised learning setting where each training example consists of a pair of sequences $(x_1^{(a)}, \ldots, x_{T_a}^{(a)})$, $(x_1^{(b)}, \ldots, x_{T_b}^{(b)})$ of fixed-size vectors (each $x_i^{(a)}, x_j^{(b)} \in \mathbb{R}^{d_{in}}$) along with a single label y for the pair. Note that the sequences may be of different lengths $T_a \neq T_b$ and the sequence lengths can vary from example to example. While we treat the two sequences symmetrically, our methods may be easily extended to the case where the $(x_1^{(a)}, \ldots, x_{T_a}^{(a)})$ stem from one domain and $(x_1^{(b)}, \ldots, x_{T_b}^{(b)})$ from another. Assuming the given y values reflect an underlying measure of similarity, our algorithms produce a mapping from a general space of variable length sequences into an interpretably structured metric space of fixed dimensionality (which can be applied to new examples not present in the data unlike manifold embedding techniques such as multidimensional scaling).

Our motivating example is the task of scoring similarity between sentences, given example pairs whose semantic similarity has been labeled by humans as y. In this case, each $x_i^{(a)}$ denotes the vector representation of a word from the first sentence while the $x_j^{(b)}$ denote word vectors from the second. Thus, we employ LSTMs with the explicit goal to learn a metric that reflects semantics, in contrast to the work of Sutskever, Vinyals, and Le (2014), where such properties emerge in the learned representations as indirect effects of the translation task.

Related Work

Due to its importance across diverse applications, semantic similarity evaluation was selected as the first task of SemEval 2014, where numerous researchers applied methods to a labeled dataset containing pairs of sentences involving compositional knowledge (SICK) (Marelli et al. 2014). Competitive methods for this data have all utilized both heterogeneous features (e.g. word overlap/similarity, negation modeling, sentence/phrase composition) as well as external resources (e.g. Wordnet (Miller 1995)), and a wide variety of learning algorithms have been applied (e.g. SVM, Random forest, k-Nearest Neighbors, and model ensembles). One of the best original submissions, from Zhao, Zhu, and Lan (2014), learns vector-space representations using latent semantic analysis along with numerous other handcrafted features. Another high-performing system, from Bjerva et al. (2014), uses formal semantics and logical inference in conjunction with features derived from the popular *word2vec* neural language model of Mikolov et al. (2013), whose word vectors we employ as the sole input to our model.

Recently, three neural network methods more similar to our approach have produced marked improvements in performance. He, Gimpel, and Lin (2015) propose an elaborate convolutional network (ConvNet) variant which infers sentence similarity by integrating various differences across many convolutions at varying scales. These authors explain that their substantial architectural engineering is necessary due to the limited availability of labeled data, an issue we deal with in this work by augmenting the training set.

Kiros et al. (2015) propose the skip-thoughts model, which extends the skip-gram approach of *word2vec* from the word to sentence level. This model feeds each sentence into an RNN encoder-decoder (with GRU activations) which attempts to reconstruct the immediately preceding and following sentences. To adapt their approach to the sentence similarity task, Kiros et al. first pass a sentence through the RNN encoder (whose weights are fixed after training on the initial corpus) to obtain a *skip-thought* vector. Subsequently, a separate classifier is trained on the SICK data using features derived from differences and products between skip-thought vectors for the pair of sentences appearing in each training example. As in the encoder-decoder framework of Sutskever, Vinyals, and Le (2014), semantic properties emerge in the skip-thought representations as indirect effects rather than being explicitly targeted in the objective.

Tai, Socher, and Manning (2015) propose Tree-LSTMs which generalize the order-sensitive chain-structure of standard LSTMs to tree-structured network topologies. Each sentence is first converted into a parse tree (using a separately trained parser), and the Tree-LSTM composes its hidden state at a given tree node from the corresponding word as well as the hidden states of all child nodes. The hope is that by reflecting syntactic properties of a sentence, the parse tree structured network can propagate necessary information more efficiently than a sequentially restricted architecture. This model is adapted for sentence similarity just as by Kiros et al., where representations of the input sentences are now produced by Tree-LSTMs rather than skip-thoughts (Tree-LSTM representations are however jointly trained together with the final classifier on the SICK dataset).

Our proposed model also represents sentences using neural networks whose inputs are word vectors learned separately from a large corpus. However, our representation learning objective directly reflects the given semantic similarity labels unlike the approach of Kiros et al. While these aforementioned neural networks utilize complex learners to predict semantic similarity from the sentence representations, we impose stronger demands: namely, a semantically structured representation space should be learned such that simple metrics suffice to capture sentence similarity. This perspective also underlies the siamese architec-

ture for face verification developed by Chopra, Hadsell, and LeCun (2005), which utilizes symmetric ConvNets where we use LSTMs. Siamese neural networks been proposed for a number of metric learning tasks (Yih et al. 2011; Chen and Salman 2011), but to our knowledge, recurrent connections remain largely unexplored in this context.

Manhattan LSTM Model

The proposed Manhattan LSTM (MaLSTM) model is outlined in Figure 1. There are two networks LSTM_a and LSTM_b which each process one of the sentences in a given pair, but we solely focus on siamese architectures with tied weights such that $\text{LSTM}_a = \text{LSTM}_b$ in this work. Nevertheless, the general untied version of this model may be more useful for applications with asymmetric domains such as information retrieval (where search queries are stylistically distinct from stored documents).

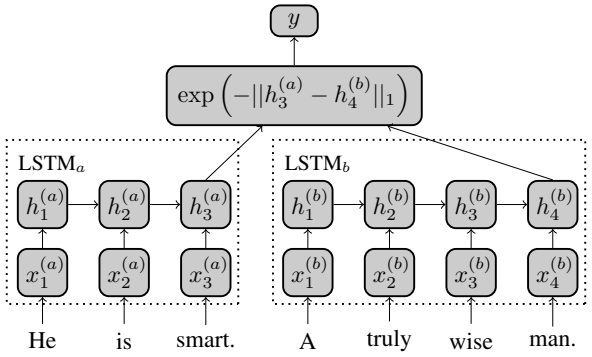

Figure 1: Our model uses an LSTM to read in word-vectors representing each input sentence and employs its final hidden state as a vector representation for each sentence. Subsequently, the similarity between these representations is used as a predictor of semantic similarity.

The LSTM learns a mapping from the space of variable length sequences of d_{in}-dimensional vectors into $\mathbb{R}^{d_{rep}}$ ($d_{in} = 300, d_{rep} = 50$ in this work). More concretely, each sentence (represented as a sequence of word vectors) $x_1, \ldots, x_T$, is passed to the LSTM, which updates its hidden state at each sequence-index via equations (2)-(7). The final representation of the sentence is encoded by $h_T \in \mathbb{R}^{d_{rep}}$, the last hidden state of the model. For a given pair of sentences, our approach applies a pre-defined similarity function $g : \mathbb{R}^{d_{rep}} \times \mathbb{R}^{d_{rep}} \rightarrow \mathbb{R}$ to their LSTM-representations. Similarities in the representation space are subsequently used to infer the sentences' underlying semantic similarity.

Note that unlike typical language modeling RNNs, which are used to predict the next word given the previous text, our LSTMs simply function like the encoder of Sutskever, Vinyals, and Le (2014). Thus, the sole error signal backpropagated during training stems from the similarity between sentence representations $h_{T_a}^{(a)}$, $h_{T_b}^{(b)}$, and how this predicted similarity deviates from the human annotated ground truth relatedness. We restrict ourselves to the simple similarity function $g(h_{T_a}^{(a)}, h_{T_b}^{(b)}) = \exp(-||h_{T_a}^{(a)} - h_{T_b}^{(b)}||_1) \in [0, 1]$. This forces the LSTM to entirely capture the semantic differences during training, rather than supplementing the RNN with a more complex learner that can help resolve shortcomings in the learned representations as done by Kiros et al. (2015) and Tai, Socher, and Manning (2015).

As Chopra, Hadsell, and LeCun (2005) point out, using a ℓ_2 rather than ℓ_1 norm in the similarity function can lead to undesirable plateaus in the overall objective function. This is because during early stages of training, a ℓ_2-based model is unable to correct errors where it erroneously believes semantically different sentences to be nearly identical due to vanishing gradients of the Euclidean distance. Empirically, our results are fairly stable across various types of simple similarity function, but we find that g utilizing the Manhattan distance slightly outperforms other reasonable alternatives such as cosine similarity (used in Yih et al. 2011).

Semantic relatedness scoring

The SICK data contains 9927 sentence pairs with a 5,000/4,927 training/test split (Marelli et al. 2014). Each pair is annotated with a relatedness label $\in [1, 5]$ corresponding to the average relatedness judged by 10 different individuals. Although their skip-thoughts RNN is trained on a vast corpus for two weeks, Kiros et al. (2015) point out that it is unable to distinguish between many of the test-set sentences shown in Table 1, highlighting the difficulty of this task.

Sentence pair	**G**	**S**	**M**
A little girl is looking at a woman in costume. A young girl is looking at a woman in costume.	4.7	4.5	4.8
A person is performing tricks on a motorcycle. The performer is tricking a person on a motorcycle.	2.6	4.4	2.9
Someone is pouring ingredients into a pot. A man is removing vegetables from a pot.	2.4	3.6	2.5
Nobody is pouring ingredients into a pot. Someone is pouring ingredients into a pot.	3.5	4.2	3.7

Table 1: Example sentence pairs from the SICK test data. G denotes ground truth relatedness $\in [1, 5]$, S = skip-thought predictions, and M = MaLSTM predictions.

To enable our model to generalize beyond the limited vocabulary present in the SICK training set, we provide the LSTM with inputs that reflect relationships between words beyond what can be inferred from the small number of training sentences. LSTMs typically require large datasets to achieve good generalization due to their vast numbers of parameters, and we thus augment our dataset with numerous additional training examples, a common practice in SemEval systems (Marelli et al. 2014) as well as high-performing neural networks. Like many top performing semantic similarity systems, our LSTM takes as input word-vectors which have been pre-trained on an external corpus. We use the 300-dimensional *word2vec* embeddings[1] which Mikolov et al. (2013) demonstrate can capture intricate inter-word relationships such as $vec(\text{king}) - vec(\text{man}) + vec(\text{woman}) \approx$

[1]Publicly available at: code.google.com/p/word2vec

vec(queen). We encourage invariance to precise wording and expand our dataset by employing thesaurus-based augmentation in which 10,022 additional training examples are generated by replacing random words with one of their synonyms found in Wordnet (Miller 1995). A similar strategy is also successfully adopted by Zhang, Zhao, and LeCun (2015). Unlike the SemEval 2014 submissions, our methods do not require extensive manual feature generation beyond the separately trained *word2vec* vectors.

The MaLSTM predicts relatedness for a given pair of sentences via $g(h_{T_a}^{(a)}, h_{T_b}^{(b)})$, and we train the siamese network using backpropagation-through-time under the mean-squared-error (MSE) loss function (after rescaling the training-set relatedness labels to lie $\in [0, 1]$). SemEval evaluates predicted similarities against the given human-annotated similarities on three metrics: Pearson correlation, Spearman correlation, and MSE. Due to the simple construction of our similarity function, the predictions of our model are constrained to follow the $\exp(-x)$ curve and are thus not suited for these evaluation metrics. After training our model, we apply an additional nonparametric regression step to obtain better-calibrated predictions (with respect to MSE). Over the training set, the given labels (under original $[1, 5]$ scale) are regressed against the univariate MaLSTM g-predicted relatedness as the sole covariate, and the fitted regression function is evaluated on the MaLSTM-predicted relatedness of the test pairs to produce adjusted final predictions. We use the classical local-linear estimator discussed in Fan and Gijbels (1992) with bandwidth selected using leave-one-out cross-validation. This calibration step serves as a minor correction for our restrictively simple similarity function (which is necessary to retain interpretability of the sentence representations).

Training details

Our LSTM uses 50-dimensional hidden representations h_t and memory cells c_t. Optimization of the parameters is done using the Adadelta method of Zeiler (2012) along with gradient clipping (rescaling gradients whose norm exceeds a threshold) to avoid the exploding gradients problem (Pascanu, Mikolov, and Bengio 2013). We employ early-stopping based on a validation set containing 30% of the training examples.

It is well-known that the success of LSTMs depends crucially on their initialization, and often parameters transferred from neural networks trained for a different task can serve as a strong starting point for the optimization (c.f. Bengio 2012). We first initialize our LSTM weights with small random Gaussian entries (and a separate large value of 2.5 for the forget gate bias to facilitate modeling of long range dependence). Then, our MaLSTM is (pre)trained as previously described on separate sentence-pair data provided for the earlier SemEval 2013 Semantic Textual Similarity task (Agirre and Cer 2013). The weights resulting from this pre-training thus form our starting point for the SICK data, which is markedly superior to a random initialization.

Results

The MaLSTM is able to accurately score the Table 1 examples which Kiros et al. highlight as difficult for their skip-thoughts model. Despite being calibrated for MSE, our approach performs better than existing systems for the semantic relatedness task across all three evaluation metrics (see Table 2). Note that because all results shown in Table 2 rely on additional feature generation (e.g. dependency parses) or data augmentation schemes, this is only an evaluation of complete relatedness-scoring systems rather than a fair comparison of the different learning algorithms employed. Nonetheless, we perform ablation experiments to better understand our methods finding that the Pearson-correlation (the primary SemEval performance metric) of our approach worsens by: 0.01 without regression calibration, 0.02 without pre-training, and 0.04 without synonym augmentation. Due to the limited available training data, we do not realize performance gains by switching to multi-layer or bidirectional LSTMs.

Method	r	ρ	**MSE**
Illinois-LH (Lai and Hockenmaier 2014)	0.7993	0.7538	0.3692
UNAL-NLP (Jimenez et al. 2014)	0.8070	0.7489	0.3550
Meaning Factory (Bjerva et al. 2014)	0.8268	0.7721	0.3224
ECNU (Zhao, Zhu, and Lan 2014)	0.8414	–	–
Skip-thought+COCO (Kiros et al. 2015)	0.8655	0.7995	0.2561
Dependency Tree-LSTM (Tai, Socher, and Manning 2015)	0.8676	0.8083	0.2532
ConvNet (He, Gimpel, and Lin 2015)	0.8686	0.8047	0.2606
MaLSTM	**0.8822**	**0.8345**	**0.2286**

Table 2: Test set Pearson correlation (r), Spearman's ρ, and mean squared error for the SICK semantic textual similarity task. The first group of results are top SemEval 2014 submissions and the second group are recent neural network methods (best result from each paper shown).

In Table 3, Tai, Socher, and Manning show the most similar test-set examples found by their Tree-LSTM for three given sentences as well as its inferred similarity scores. We apply our model to these same examples, determining that while the sequential MaLSTM is slightly worse at identifying active-passive equivalence, our approach is better at distinguishing verbs and objects than the compositional Tree-LSTM which often infers seemingly over-estimated relatedness scores in Table 3. For example, the ground truth labeling between "Tofu is being sliced by a woman" and "A woman is slicing butter" is only 2.7 in the SICK test set (and substituting "potatoes" for "butter" should not greatly increase relatedness between the two statements).

Ranking by Dependency Tree-LSTM Model	Tree	M
a woman is slicing potatoes		
a woman is cutting potatoes	4.82	4.87
potatoes are being sliced by a woman	4.70	4.38
tofu is being sliced by a woman	4.39	3.51
a boy is waving at some young runners from the ocean		
a group of men is playing with a ball on the beach	3.79	3.13
a young boy wearing a red swimsuit is jumping out of a blue kiddies pool	3.37	3.48
the man is tossing a kid into the swimming pool that is near the ocean	3.19	2.26
two men are playing guitar		
the man is singing and playing the guitar	4.08	3.53
the man is opening the guitar for donations and plays with the case	4.01	2.30
two men are dancing and singing in front of a crowd	4.00	2.33

Table 3: Most similar sentences (from 1000-sentence subsample) in the SICK test data according to the Tree-LSTM. Tree / M denote relatedness (with the sentence preceding each group) predicted by the Tree-LSTM / MaLSTM.

Sentence representations

We now investigate the geometry of the sentence representation-space learned by the MaLSTM network. As the ℓ_1 metric is the sum of element-wise differences, we hypothesize that by using specific hidden units (i.e. dimensions of the sentence representation) to encode particular characteristics of a sentence, the trained MaLSTM infers semantic similarity between sentences by simply aggregating their differences in various characteristics.

Some examples supporting this idea are shown in Figure 2, which depicts the values that particular sentences take along specific dimensions of h_T. It is evident that the hidden unit shown at the top has learned to detect negation, separating sentences containing words like "no" or "not" from the rest, regardless of the other content in the text. The hidden unit in the middle plot is particularly sensitive to categorization of the direct objects, separating sentences describing actions on balls, grass, cosmetics, and vegetables. The hidden unit depicted at the bottom of Figure 2 clearly separates sentences based on their subject, imposing an interesting ordering that reflects broader similarity between the subject categories: cats, general animals, dogs, boys, general people (someone), and men. Unlike the ConvNet of He, Gimpel, and Lin which measures similarity explicitly across multiple scales and locations in the sentence, the delineation of these underlying characteristics emerges naturally in the MaLSTM representations, which are guided solely by the ℓ_1 metric and overall semantic similarity labels.

Next, we shift our attention from local characteristics of different hidden units toward the global geometry of the sentence representation space. Due to our training criterion, this space is naturally endowed with the ℓ_1 metric and avoids being highly warped. While analysis of neural network representations typically requires nonlinear dimensionality reduction like t-SNE (van der Maaten and Hinton 2008), we can simply use principal components analysis (PCA) for informative visualization of the MaLSTM representations due to their simple structure.

Figure 2: MaLSTM representations of test set sentences depicted along three different dimensions of h_T (indices 1, 2, and 6). Each number above the axis corresponds to a sentence representation and its location represents the value this particular hidden unit assigns to the sentence (shown below).

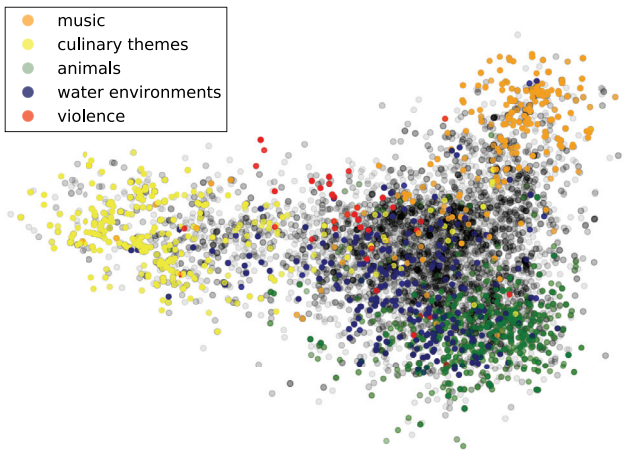

Figure 3: MaLSTM representations for all sentences from the SICK test set, projected onto two principal components.

Figure 3 depicts an overview of the SICK dataset from the perspective of the MaLSTM model (after PCA dimension reduction). For interpretability, we color many of the sentences based on distinct concepts/themes under which they fall. The geometric coherence of the sentences in the representation space exists across numerous categories: from sentences about animals (ranging from cats to lemurs), culinary themes (like slicing vegetables), music (like guitar playing), water environments (e.g. the ocean or swimming pools), etc. In fact, the sentence representations cluster along nearly all additional meaningful semantic categorizations we could come up with (not depicted due to coloring constraints).

One peculiar aspect of this representations space is the low-density region that separates the culinary themed examples from the other sentences. Around this area, there are numerous violence and gun-related sentences in the representations, for example: "A man is fixing a silencer to a gun". We find that these violent texts are likely to receive much lower similarity scores when paired with more mundane sentences typically found in SICK (the average violent-nonviolent pair only has similarity 1.88 compared with an average of 3.41 for all test-set pairs). This explains why the MaLSTM representations have learned to become sparse in the vicinity of these violent examples (depicted in red in Figure 3).

Thus, Figure 3 shows that human-determined semantic relatedness heavily depends on the occurrence of such themes. These discoveries about the SICK dataset are enabled by the interpretability of the MaLSTM representations, unlike the other proposed neural networks which rely on complex operations over their learned representations. In addition to providing model insight, informative representations can provide a useful tool for exploratory data analysis.

Entailment classification

To evaluate the broader utility of our sentence representations, we leverage them for a different application: the SemEval 2014 textual entailment task (Marelli et al. 2014). In addition to the relatedness scores, each of the SICK sentence pairs has also been labeled as one of three classes: entailment, contradiction, or neutral, which are to be predicted for the test examples. For this task, we solely rely on the same representations learned for predicting semantic relatedness (fixed without additional fine-tuning), and simply apply standard learning methods to do the entailment classification.

Specifically, from the MaLSTM representations $h^{(a)}_{T_a}, h^{(b)}_{T_b}$ of each pair of sentences, we compute the following simple features (also successfully used by Tai, Socher, and Manning 2015): element-wise (absolute) differences $|h^{(a)}_{T_a} - h^{(b)}_{T_b}|$ and element-wise products $h^{(a)}_{T_a} \odot h^{(b)}_{T_b}$. Using only these features, we train a radial-basis-kernel SVM to classify the entailment labels. The one-versus-all approach to multi-class problems is employed with hyperparameters optimized in 5-fold cross-validation.

Table 4 shows that such an approach outperforms all other textual-entailment systems except for the Illinois-LH system of Lai and Hockenmaier (2014). Thus even though the features provided to the SVM are learned for the distinct goal of semantic relatedness scoring (with no supervised information regarding contradictions or the neutral threshold), they capture enough relevant characteristics of the sentences to be highly useful for entailment-classification. In contrast to the MaLSTM representations, the Illinois-LH system employs many features specially constructed for this task such as hypernym counts and occurrences of "no" and "not". Interestingly, a useful feature like "no"-occurrence, which Lai and Hockenmaier manually selected, has been automatically learned by our model and is encoded by the first hidden unit shown in Figure 2.

Method	**Accuracy**
Illinois-LH (Lai and Hockenmaier 2014)	**84.6**
ECNU (Zhao, Zhu, and Lan 2014)	83.6
UNAL-NLP (Jimenez et al. 2014)	83.1
Meaning Factory (Bjerva et al. 2014)	81.6
Reasoning-based n-best (Lien and Kouylekov 2015)	80.4
LangPro Hybrid-800 (Abzianidze 2015)	81.4
SNLI-transfer 3-class LSTM (Bowman et al. 2015)	80.8
MaLSTM features + SVM	84.2

Table 4: Test set accuracy for the SICK semantic entailment classification. The first group of results are top SemEval 2014 submissions and the second are more recently proposed methods.

Discussion

This work demonstrates that a simple LSTM is capable of modeling complex semantics if the representations are explicitly guided. Leveraging synonym augmentation and pretrained word-embeddings, we circumvent the size-limitations of existing labeled datasets. Analysis of the learned model reveals that it utilizes diverse hidden units to encode different characteristics of each sentence. Admitting efficient test-time inference, our model can be deployed in real-time applications. Not only useful for scoring semantic relatedness/entailment, trained MaLSTM sentence representations can produce interesting insights in exploratory data analysis thanks to their interpretable structure.

Since our approach relies on pre-trained word-vectors as the LSTM inputs, it will benefit from improvements in word-embedding methods such as those of Li et al. (2015), especially as these word-vectors more comprehensively capture synonymity and entity-relationships. We also foresee significant gains as the amount of labeled semantic similarity data grows, both for statistical reasons and because sufficiently large sample sizes enable learning of de novo word-vectors tailored to this model.

References

Abzianidze, L. 2015. A Tableau Prover for Natural Logic and Language. *EMNLP* 2492–2502.

Agirre, E., and Cer, D. 2013. SEM 2013 shared task: Semantic Textual Similarity. *SemEval 2013*.

Bengio, Y. 2012. Deep Learning of Representations for Unsupervised and Transfer Learning. *JMLR W&CP: Proc. Unsupervised and Transfer Learning challenge and workshop* 17–36.

Bjerva, J.; Bos, J.; van der Goot, R.; and Nissim, M. 2014. The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. *SemEval 2014*.

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. *EMNLP* 632–642.

Chen, K., and Salman, A. 2011. Extracting Speaker-Specific Information with a Regularized Siamese Deep Network. *NIPS* 298–306.

Cho, K.; Gulcehre, B. v. M. C.; Bahdanau, D.; Schwenk, F. B. H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *EMNLP* 1724–1734.

Chopra, S.; Hadsell, R.; and LeCun, Y. 2005. Learning a similarity metric discriminatively, with application to face verification. *Computer Vision and Pattern Recognition* 1:539–546.

Fan, J., and Gijbels, I. 1992. Variable bandwidth and local linear regression smoothers. *The Annals of Statistics* 20:2008–2036.

Graves, A. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*. Studies in Computational Intelligence, Springer.

Greff, K.; Srivastava, R. K.; Koutnik, J.; Steunebrink, B. R.; and Schmidhuber, J. 2015. LSTM: A Search Space Odyssey. *arXiv: 1503.04069*.

He, H.; Gimpel, K.; and Lin, J. 2015. Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks. *EMNLP* 1576–1586.

Hochreiter, S., and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation* 9(8):1735–1780.

Jimenez, S.; Duenas, G.; Baquero, J.; Gelbukh, A.; Bátiz, A. J. D.; and Mendizábal, A. 2014. Unal-nlp: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. *SemEval 2014*.

Kiros, R.; Zhu, Y.; Salakhutdinov, R.; Zemel, R. S.; Torralba, A.; Urtasun, R.; and Fidler, S. 2015. Skip-Thought Vectors. *NIPS* to appear.

Lai, A., and Hockenmaier, J. 2014. Illinois-lh: A denotational and distributional approach to semantics. *SemEval 2014*.

Le, Q., and Mikolov, T. 2014. Distributed Representations of Sentences and Documents. *ICML* 1188–1196.

Li, Y.; Xu, L.; Tian, F.; Jiang, L.; Zhong, X.; and Chen, E. 2015. Word Embedding Revisited: A New Representation Learning and Explicit Matrix Factorization Perspective. *IJCAI*.

Lien, E., and Kouylekov, M. 2015. Semantic Parsing for Textual Entailment. *International Conference on Parsing Technologies* 40–49.

Marelli, M.; Bentivogli, L.; Baroni, M.; Bernardi, R.; Menini, S.; and Zamparelli, R. 2014. SemEval-2014 Task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval 2014*.

Mihalcea, R.; Corley, C.; and Strapparava, C. 2006. Corpus-based and Knowledge-based Measures of Text Semantic Similarity. *AAAI Conference on Artificial Intelligence*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; and Dean, J. 2013. Distributed Representations of Words and Phrases and their Compositionality. *NIPS* 3111–3119.

Miller, G. A. 1995. WordNet: A Lexical Database for English. *Communications of the ACM* 38(11):39–41.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the difficulty of training recurrent neural networks. *ICML* 1310–1318.

Siegelmann, H. T., and Sontag, E. D. 1995. On the Computational Power of Neural Nets. *Journal of Computer and System Sciences* 50:132–150.

Socher, R. 2014. *Recursive Deep Learning for Natural Language Processing and Computer Vision*. Phd thesis, Stanford University.

Sutskever, I.; Vinyals, O.; and Le, Q. 2014. Sequence to sequence learning with neural networks. *NIPS* 3104–3112.

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. *ACL* 1556–1566.

van der Maaten, L., and Hinton, G. 2008. Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research* 9:2579–2605.

Yih, W.; Toutanova, K.; Platt, J.; and Meek, C. 2011. Learning Discriminative Projections for Text Similarity Measures. *Proceedings of the Fifteenth Conference on Computational Natural Language Learning* 247–256.

Zeiler, M. D. 2012. ADADELTA: An Adaptive Learning Rate Method. *arXiv:1212.5701*.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level Convolutional Networks for Text Classification. *arXiv:1509.01626*.

Zhao, J.; Zhu, T. T.; and Lan, M. 2014. Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. *SemEval 2014*.