Semantic Attention Networks for Intuitive Information Retrieval

Jonathan La[1]
McMaster University, 1280 Main Street West, Hamilton Ontario, Canada

(Dated: **12 January 2018**)

In this work we present a siamese LSTM-attention network capable of both learning semantic similarity between sentence pairs as well as highlighting significant tokens important for the prediction task. Highlighted tokens are then used to demonstrate simple comparison, summary and topic modelling methods which can be used by a human analyst for the purpose of information retrieval tasks with increased efficiency. The target application for the proposed network is for understanding general semantic trends and finding specific semantic information in a large text data set such as a social media network.

I. INTRODUCTION

The ability to accurately identify semantically similar text entities (eg. words, sentences and paragraphs) has been a growing research interest to the natural language processing community, as is evidenced by recent competitions[1]. Historically, comparing text semantics relied heavily on word-nets, bag-of-word implementations and token-count statistics, however neural network models have since been shown to be increasingly effective at learning semantics[2–4]. Many of these methods rely on the successes of the Long Short Term Memory (LSTM) network, capable of both converting an arbitrary-length sequences into a fixed-length vector as well as capturing long range dependencies over inputs[5].

Although neural networks offer great predictive power, it is often difficult for a human analyst to interpret how the model is learning. The ability to provide an analyst with an intuitive understanding of how the model is making predictions is useful not only for further model development and refinement, but can also serve as a simple interface for information compression. Precisely, we focus on the particular problem of navigation through a large data set composed of text elements (sentences, short paragraphs), where navigation entails both developing an understanding for general themes in the data set via cluster analysis as well as efficiently finding specific semantic information embedded within the data set.

II. APPROACH

Our approach combines the semantic predictive power of the Dependency Tree-LSTM (DTLSTM)[3] and the self-structured attention mechanism with penalization strategy[6]. The model takes as input a left and right sentence pair along with a label of semantic relatedness, and training is performed in a supervised fashion. The network is siamese (shared model weights) with respect to how the inputs are processed, although this could be changed for information retrieval tasks where the left and right application domains differ.

Formally, our model takes as input two text sequences together with a numerical label of semantic relatedness between them $y \in R$. Each sentence is then tokenized using the Stanford Neural Network Dependency Parser[7], and the label y is encoded as a vector $\bar{\mathbf{y}} \in R^{n_c}$ (n_c the number of target classes) allowing for the use of the KL divergence loss function. Sentences are then padded to a set length n with a padding token prior to being encoded using pre-trained Stanford Glove embedding vectors $\mathbf{x} \in R^{d_e}$, with d_e the embedding vector dimension (300 in our case)[8]. The embedding vector sequences are then passed into a single-layer DTLSTM, which updates its hidden state at each sequence-index of the input. Rather than keeping only the final hidden state of the DTLSTM[3], we keep all hidden states formed at each sequence-index update and form the matrix $H \in R^{n*d_m}$, where d_m is the hidden layer size of the DTLSTM.

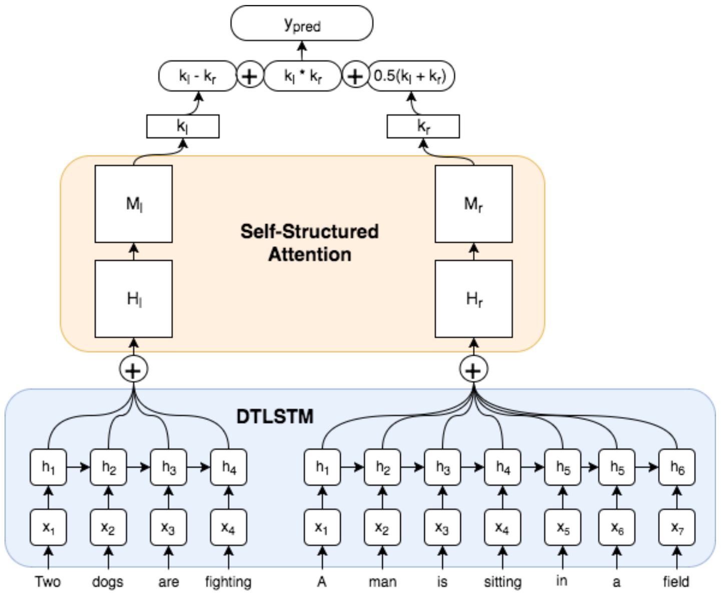

Figure 1: Our model combines the dependency tree-LSTM (DTLSTM) and the self-structured attention mechanism to create representations of each input sentence. Representations are then combined considering 3 different distance measures before being passed through a final MLP for prediction

We then choose a linear combination of the hidden states in H using the self-structured attention mechanism[6] $\mathbf{a} \in R^n$:

$$\mathbf{a} = softmax(w_{s2} tanh(W_{s1} H^T))^6$$

With $W_{s1} \in R^{d_a * d_m}$, $\mathbf{w_{s2}} \in R^{d_a}$ and d_a representing the number of attention units. We allow for r different attention hops over H by extending $\mathbf{w_{s2}}$ to $W_{s2} \in R^{r*d_a}$ yielding the annotation matrix $A \in R^{r*n}$:

$$A = softmax(W_{s2} tanh(W_{s1} H^T))^6$$

By applying this attention mechanism to the original sentence embedding matrix H we obtain a new sentence representation $M \in R^{r*d_m}$:

$$M_l = A_l H_l{}^6$$
$$M_r = A_r H_r{}^6$$

We then apply a batched dot product between the 2-D matrix M and a 3-D weight tensor $W_f \in R^{1*d_m*d_{o1}}$ with relu non linearity, which can be understood as applying a different single-layer Multi-Layer Perceptron (MLP) with hidden layer size d_{o1} without bias to each row of M, yielding $F \in R^{r*d_{o1}}$:

$$F_l = relu(batcheddot(M_l, W_f))\ ^6$$
$$F_r = relu(batcheddot(M_r, W_f))^6$$

We then squash F_l, F_r into sentence vectors $\mathbf{k_l}$ and $\mathbf{k_r} \in R^{d_{o1}}$ using a single-layer MLP with relu nonlinearity, parametrized by weight tensor $W_s \in R^{r*1}$ and bias term $B_s \in R^{d_{o1}}$

$$\mathbf{k_l} = (F_l^T W_s + B_s)^T$$
$$\mathbf{k_r} = (F_r^T W_s + B_s)^T$$

We then combine $\mathbf{k_l}$ and $\mathbf{k_r}$ into a single vector $\Phi \in R^{3d_{o1}}$ which considers the relative difference, mean and element-wise product between the pair:

$$\begin{aligned}\Phi(\mathbf{k_l}, \mathbf{k_r}) &= \begin{bmatrix}\mathbf{u}, & \mathbf{v}, & \mathbf{w},\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{k_l} - \mathbf{k_r}, & \mathbf{k_l} \odot \mathbf{k_r}, & \frac{1}{2}(\mathbf{k_l} + \mathbf{k_r})\end{bmatrix}\end{aligned}$$

Where $\odot$ represents the Hadamard (element-wise) product. We then feed Φ through a final 2-layer MLP with leaky-relu and sigmoid non-linearities, followed by a softmax output. If the first and second MLP layers are parametrized by weight and bias tensors $W_{f1} \in R^{3d_{o1}*d_{o2}}, B_{f1} \in R^{d_{o2}}$ and $W_{f2} \in R^{d_{o2}*d_{o3}}, B_{f2} \in R^{d_{o3}}$ respectively, and the output layer by weight and bias tensors $W_{f3} \in R^{d_{o3}*n_c}, B_{f3} \in R^{n_c}$, then the prediction vector $\mathbf{y_{pred}}$ is given by:

$$Y_1 = leaky - relu(\Phi W_{f1} + B_{f1})$$
$$Y_2 = sigmoid(Y_1 W_{f2} + B_{f2})$$
$$\mathbf{y_{pred}} = softmax(Y_2 W_{f3} + B_{f3})$$

During training, the loss is then computed as the KL-divergence between $\mathbf{y_{pred}}$ and $\bar{\mathbf{y}}$. To constrain attention weights to eligible (non-pad) tokens, a penalization term for the self-structured attention matrix is introduced A given by $P = ||(AA^T - I)||_F^2$, where I is the identity matrix[6]. This penalization term is obtained for each input as it passes through the network, and is added to the total loss function.

III. TRAINING PROCEDURE

The SICK[1] data set contains 9927 sentence pairs with a 4500/500/4927 training/development/test split[13], respectively. The padding length is set to $n = 40$, and calibration is done using random search over the following parameters: learning rate, weight decay, d_m, d_{o1}, d_{o2}, d_{o3}, r, d_a and dropout, as well as some model components such as MLP nonlinearities. The optimization is performed using the Adagrad optimizer[9], and KL divergence is used as a loss function. Dropout was applied to each layer of the final predicting MLP, excluding the output layer, and calibration yielded the following parameters: learning rate=0.0045, weight decay=0.007, d_m=350, d_{o1}=1000, d_{02}=280, d_{o3}=280, r=7, d_a=40, $dropout$=0.07. After calibrating our model on the development set, we apply the best-performing model to the test set.

IV. RESULTS

After calibration we train our model with the above parameters for 20 epochs, obtaining a SICK[1] test set accuracy (Pearson correlation) of 86.79%, compared to the test-set accuracy of the dependency tree-LSTM[3] alone achieving a Pearson correlation of 86.76 %. We note that better performing models use significantly more data via transfer learning, with state of the art transfer learning methods achieving 88.5%[10].

The addition of the self-structured attention mechanism does not significantly add to performance results, however it does allow us to visualize which elements of each sentence are most important for predicting semantic relatedness. Next, we investigate three simple methods for enhancing the efficiency of an analyst during information retrieval tasks. To demonstrate, we randomly sample 500 sentences from the entire SICK data set and compute the $O(n^2)$ pairwise similarity between each sentence pair. For each sentence-pair comparison, the attention weightings are stored for each sentence for later inspection. The resulting similarity matrix is then clustered using spectral clustering, allowing the nearest neighbours (and clusters) of sentences to be identified. In the figures below we rank sentence tokens based on their learned attention weightings, where we add colour to emphasize the top 3 vales (more than 3 highlighted tokens indicates that multiple tokens received the same attention weight). The highest attention weight is indicated in red, and all others in orange for simplicity.

The first and most obvious way to use the attention weights is during the direct comparison of two sentences. By highlighting the parts of the sentence deemed most relevant to the prediction task, an analyst is able to gain valuable insight into how the algorithm is learning. This could be used by an analyst to determine how the model is lacking, and could provide an interface for teacher training via semi-supervised learning.

s1:	A woman who is wearing a pink boa is riding a bicycle down a bridge built for pedestrians
s2:	The cyclist is performing a trick in the air
s1:	The ice skating rink placed outdoors is full of people
s2:	A lot of people are in an ice skating park
s1:	The young boys are playing outdoors and the man is smiling nearby
s2:	The kids are playing outdoors near a man with a smile

Figure 2: Attention weightings used to compare two sentences s1 and s2. Predicted similarity between s1 and s2 are 0.44 (top), 0.83 (middle) and 0.93 (bottom)

The tokens receiving the highest attention can also be used as an effective summarizing method, which can be used to compress larger text objects into a short-list of significant keywords. Using such a method could allow an analyst to increase efficiency in information retrieval tasks by scanning summary methods rather than reading raw text. To demonstrate this we choose a reference sentence from the original 500 sentences and determine its nearest neighbours (neighbour members of corresponding cluster). For all summaries, stop words[11] are ignored.

Ref. Summary	**Reference**
Five, front, child, hut	Five wooden stands are in front of each child 's hut

NN Summary	**Nearest Neighbours**
Five, standing, hut	Five children are standing in a wooden hut
angels, snow, children	Two angels are making snow on the lying children
amphitheatre, talking, boy	There is no adult in the amphitheater talking to a boy
children, playing, waiting	There are no children playing and waiting
boy, standing, water	A boy is standing in the water

Figure 3: Demonstration of a simple summary method using attention weights between a reference and its nearest neighbours. By comparing summaries rather than original text, an analyst can more efficiently search through clusters and find elements of interest. Corresponding predicted similarities (from top to bottom) are 0.56, 0.49, 0.49, 0.49 and 0.49.

The same method can be extrapolated further to a simple topic model over clusters. Beginning with a cluster of semantically similar text, we can obtain keywords (a summary) of each member. These tokens can be extended to a topic model using heuristic methods (such as common hypernyms), by augmenting existing topic modelling systems (eg. Latent Dirichlet Allocation), or used directly. For demonstration purposes we simply rank the summary tokens by frequency, adding colour for effect.

Topic
Dog
Grass
Green
Jockeys
Field
Horse

Summary	**Originals**
dog, running, grass	A dog , which is brown , and a black one are running in the grass
dog, excitedly, grass	A dog is excitedly playing with water in the grass
jockeys, field, green	The Jockeys are riding horses on the field, which is completely green
field, horse jockeys	The green field for horse races is completely full of jockeys
dog, green, grass	A dog , which is small , is playing tirelessly on the green grass
dog, green, grass	A dog , which is small , is playing on the green grass
	A dog , which is small , is running out on the green grass

Figure 4: Simply topic model over a cluster. Topics are sorted by descending frequency with colour added for emphasis (left). Summary methods (middle) and original (right) are also shown

V. DISCUSSION

Although the addition of the self-structured attention mechanism to the DTLSTM does not improve performance on this particular data set, it does provide a method for an analyst to gain valuable insight into how the algorithm is learning and a method for information compression. Due to the large number of parameters in the network (significantly more than the original DLSTM), we expect that performance improvements can be obtained by increasing the amount of training data through transfer learning or by generating more training data (eg. with thesaurus-based information[12]). The self-structured attention mechanism is understood to help an LSTM network learn long range dependencies by learning dependencies itself, relaxing some of the burden from the LSTM[6]. However, due to the short sentence lengths (i.e. lack of long range dependencies) comprising the SICK data set, we postulate that the DTLSTM itself (without the self-structured attention mechansim) is sufficient for learning the necessary long range dependencies. Evidence of this phenomena can be seen in the way that the attention weightings accumulate towards the end of the sentence, explainable by considering that the DTLSTM accumulates information to its last hidden state and thus the last token receives the most significance. Therefore, an obvious next step for this work is to apply the model to larger data sets (or to the SICK data set after transfer learning) and to data sets comprising larger sentences (increased necessity for maintaining long range dependencies).

The main emphasis of this work was to show how attention mechanisms could be used with semantic similarity networks in order to improve the efficiency of information retrieval tasks. To do this, we had to compute the pairwise ($O(n^2)$) similarity between sentences, which is computationally infeasible for engineering solutions. To address this issue, networks[12] have been developed to compute sentence embeddings ($O(n)$) which can then be compared with simple distance metrics, drastically reducing the computational requirements for computing similarities and forming clusters.

VI. CONCLUSION

In this work we demonstrated how the high-performing dependency-tree LSTM[3] model can be combined with the self-structured attention mechanism[6] to create a network that both learns semantic relatedness between sentence pairs, as well as provides an interface for condensing information for an analyst. Information compression is obtained by identifying the tokens which receive the highest attention weights, and has been shown in this work to be useful for direct sentence comparisons and developing intuition for how the model is making predictions, comparison of summary methods, rather than raw text, for the purpose of quick information retrieval, and for a simple topic model over clusters. Although the methods shown in this work rely on simple thresholding and frequency ranking techniques, learned attention weightings can be incorporated into other systems (eg. Latent Dirichlet Allocation for topic modelling) for improved performance. The next steps for this work include applying the network to larger data sets comprising longer sentence lengths (increased need for long range dependencies), as well as investigating methods for reducing the need for $O(n^2)$ pairwise comparisons in order to form clusters.

The end goal for this work was to present a method for partitioning a space of text objects in terms of semantic relatedness, as well as providing a mechanism for efficiently navigating the data set in order to find general trends and specific information of interest. Together, the methods discussed allow us to derive semantic clusters from a data set, to look across cluster topics to understand general trends in semantics, and to investigate within clusters efficiently through summary methods. Specific information can be found by looking at the nearest (semantic) neighbours to a given search query, followed by manual investigation of nearest neighbour summary methods. We expect that applications including social media navigation and survey-result analysis could benefit from such a method.

VII. ACKNOWLEDGEMENTS

I would like to especially thank Riddhiman Dasgupta for his open source Pytorch implementation of the dependency-tree LSTM[13] and Haoyue Shi for his open source Pytorch implementation of the self-structured attention mechansism[14]. The code for this work will be released at `https://github.com/lajd/`.

[1] Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *SemEval@ COLING*, pages 1–8, 2014.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[3] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[4] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

[5] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735.

[6] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[7] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[8] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[9] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[10] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

[11] Martin F Porter. An algorithm for suffix stripping. *Program*, 14 (3):130–137, 1980.

[12] Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, pages 2786–2792, 2016.

[13] An open-source implementation of the paper "a structured self-attentive sentence embedding" published by ibm and mila. `https://github.com/ExplorerFreda/Structured-Self-Attentive-Sentence-Embedding`. Accessed: 2018-01-10.

[14] Tree lstm implementation in pytorch. `https://github.com/dasguptar/treelstm.pytorch`. Accessed: 2018-01-10.