Using Contextual Information for Neural Natural Language Inference

Billovits, C.
cjbillov@stanford.edu

Eric, M.
meric@cs.stanford.edu

1. Introduction

The advent of such commercial natural language interfaces as Siri, Google Now, and Cortana has made it clear that full semantic language understanding is one of the great contemporary problems of artificial intelligence. Developing a system with this power is immensely difficult because of the complexity of human text. A system with full semantic understanding would need to be capable of handling all the tricky aspects of language including named entity disambiguation, coreference resolution, and relation extraction. Moreover, it would need to have a grounded knowledge base of the world, being able interpret common sense facts and draw logical inferences among linguistic statements. Recognizing textual entailment relations is a cornerstone problem in the path to achieving true semantic understanding. Given two sentences S_1 and S_2, this problem seeks to determine the relationship between the two statements. Common classes of relationships include entailment (S_2 logically follows from S_1), contradiction (S_2 is not logically consistent with S_1), and neutrality (S_2 is logically independent of S_1).

This task is performed with ease by humans, who have years of practice as agents in real world scenarios and processors of common sense knowledge, but is one of the hardest problems in automated natural language understanding. For a machine to recognize textual entailment well, it has to have a nuanced understanding of the sentences and be able to figure out both general semantics (with reference to world knowledge) and also the particular logical structure of the sentences (so that, for example, the system can tell that the addition of the word "no" takes the sentence pair from entailment to contradiction). In this work, we explore novel neural architectures aimed at encoding both logical structure and general semantics.

The remainder of our paper is structured as follows: Section 2 discusses existing work in natural language inference (NLI) with an emphasis on recent neural network-based architectures. Section 3 discusses existing inference datasets and the nuances each bring to NLI. Section 4 enumerates the approaches we investigated. Section 5 quantifies our initial results and documents our efforts to improve on the initial results. Finally, Section 6 discusses next areas of exploration.

2. Background

Neural network-based models are a more recent approach for natural language inference. Bowman et. al. have investigated the use of tree-structured neural networks for extracting semantic meaning and have demonstrated through solid performance on the Sentences Including Compositional Knowledge dataset, that neural models show great promise for helping artificial systems learn logical semantics. [8], [2] Bowman et. al. also used recurrent networks with a long short-term (LSTM) memory unit to perform inference on the recently released Stanford Natural Language Inference (SNLI) dataset, achieving comparable performance to a strong lexicalized classifier system. [3]

More recently, Rocktaschel et. al. presented an end-to-end sequential LSTM model with a word-by-word attention mechanism that was able to achieve state-of-the-art on the SNLI dataset. [9] Forays with more general memory structures have also seen success on SNLI. Notably, Cheng et. al. use a novel LSTM unit that unwinds the cell and hidden states onto a dynamic memory tape at each time step, using attention to compute new hidden and cell states [6] This architecture holds state-of-the-art on SNLI as well as meeting close to state-of-the-art performance on language modeling and sentiment analysis.

A few papers have also focused using improved sentence embeddings for NLI. For example, Bowman et al. 2016 use a neural stack-based shift-reduce parser to efficiently incorporate tree-structured information. [4]

3. Datasets

3.1. SICK

Prior to the release of SNLI, the largest inference corpus available was the Sentences Involving Compositional Knowledge (SICK) dataset. At roughly 10,000 premise-hypothesis sentence pairs, SICK forms a moderately-sized testing bed for traditional statistical approaches for NLI. As Bowman et. al. demonstrated, pure neural architectures tend to perform poorly when trained with the SICK dataset alone.

However, since explicit use of world knowledge was obviated from SICK, comparing neural models against each other using SICK may shed light on the extent to which models learn compositional semantics. For example, many contradictory examples in SICK are formed synthetically with negation. In lexicalized machine-learning approaches, brittle negation features can significantly boost performance [1] [7]. Neural model founded on continuous vector representations would have to distinguish sentences based on just a negation word vector, which can also inherit general semantic ambiguities (e.g. prepositional attachment, coreference resolution).

3.2. SNLI

The bulk of development for this paper was done using the recently released SNLI dataset. At 570,152 sentence pairs, it is the largest NLI corpus available. The data is divided into train/dev/test splits with 550,152/10,000/10,000 sentence pairs respectively. The splits also have vocabularies of size 42,382/6,861/7,012 unique tokens respectively. The 3 class labels are also all equally represented in the dataset. Below we present the distribution of sentence length in the corpus:

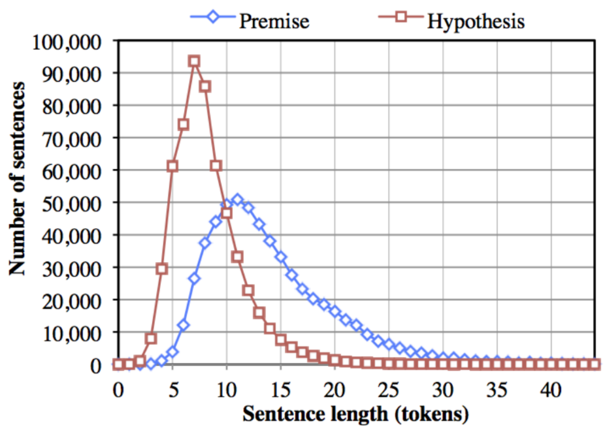


Figure 1: SNLI sentence length distribution

Sentence length is approximately Poisson distributed, with premise sentences generally longer than their hypothesis counterparts. As many premise sentences contain more than 20 words, models seeking performance must have the capacity to handle long-term lexical dependencies.

The SNLI dataset was collected through completion of a cleverly designed image-captioning task. Unlike SICK, SNLI does not explicitly obviate the necessity for world knowledge, and in fact derives much of its semantic complexity from the need to make use of common-sense grounding. This observation inspires the intuition that using an external memory tape or collection of external memories could be useful for keeping track of relevant world knowledge that complex neural NLI architectures can utilize to inform their understanding of sentence-level semantics.

Lexicalized classifiers perform moderately worse on SNLI than neural models: aggregate three-class F1 for a state-of-the art lexicalized classifier is 78.2, whereas state-of-the-art on neural models is the LSTMN in Cheng et al. 2016 [6], achieving 86.3.

4. Approach

4.1. Baselines

The first neural baseline we have considered and implemented is a simple sum of embeddings classifier. The model is organized as follows: sum 100-dimensional pre-trained GloVe embeddings of all the tokens in the premise. Do the same for the hypothesis of the data sample. Concatenate these two output vectors and then feed them through three fully-connected layers each of which is followed by a tanh nonlinearity and has a 200-dimensional hidden layer. Take the final output layer and apply a softmax transformation to get a three way probability distribution over the labels, and use this distribution to compute a cross-entropy loss. This simplistic model is not expected to do extraordinarily well, but with some fine-grained tuning, Bowman et. al. suggest that the model should be able to achieve up to 79.3%/75.3% train/test accuracy. Further it is the simplest example of a neural architecture that is expected to learn some of the semantic complexity of the SNLI dataset.

4.2. Initial Attempt at Recurrent Architectures

A natural extension to the preceding baseline is to replace the lowermost sum-of-word-embeddings layer with a premise LSTM layer and an LSTM layer [10], whose hidden states are concatenated and then fed through the fully-connected components as before. The GloVe embeddings fed into the LSTM are again 100-dimensional and produce 100 dimensional hidden-states. The fully-connected layers are also 200-dimensional. Bowman et. al. report a performance of 84.8%/77.6% on the train/test splits of SNLI. We reimplemented a similarly shallow architecture with different training parameters, with the intent of stacking the pre-trained LSTM inside our architecture to provide sentence embeddings for the input premise-hypothesis pairs. We refer to this model as the Pair-LSTM in the results below.

4.3. Memory Tape Architectures: Desiderata

In question-answering tasks, various memory tapes have been explored to emulate approximate information retrieval, with various levels of supervision. In its simplest form, question-answering gives a background series of sentences describing the world, and then asks a question, whose answer is a word or sequence of words in the vocabulary. Many earlier models have used supervision of supporting facts [13] to determine which memory cells are relevant. Suukbaatar et al. 2015 use a deep hierarchical attention model to discover relevant facts [11], utilizing content and address-based access to the memory tape.

While question-answering tasks generally have a small fixed context per example, our goal for NLI is to allow for unbounded memory tape sizes, and potentially leverage various inter-sentence facts about the world. We decided that the weakly supervised end-to-end approach in [11] is most fitting, as the candidate memory set may not always contain relevant memories, or even encode mutually reconcilable worlds. Thus, some sort of attention model between sentences is appropriate.

Another prime consideration is the granularity of attention on memories. State-of-the-art recurrent models employ word-level granularity, but cannot use world knowledge effectively. To ensure memories do not depend on hidden elements of their derivative sentence, we restrict our memory tape to represent contiguous premise and hypothesis sentence pairs, instead of clauses or predicates.

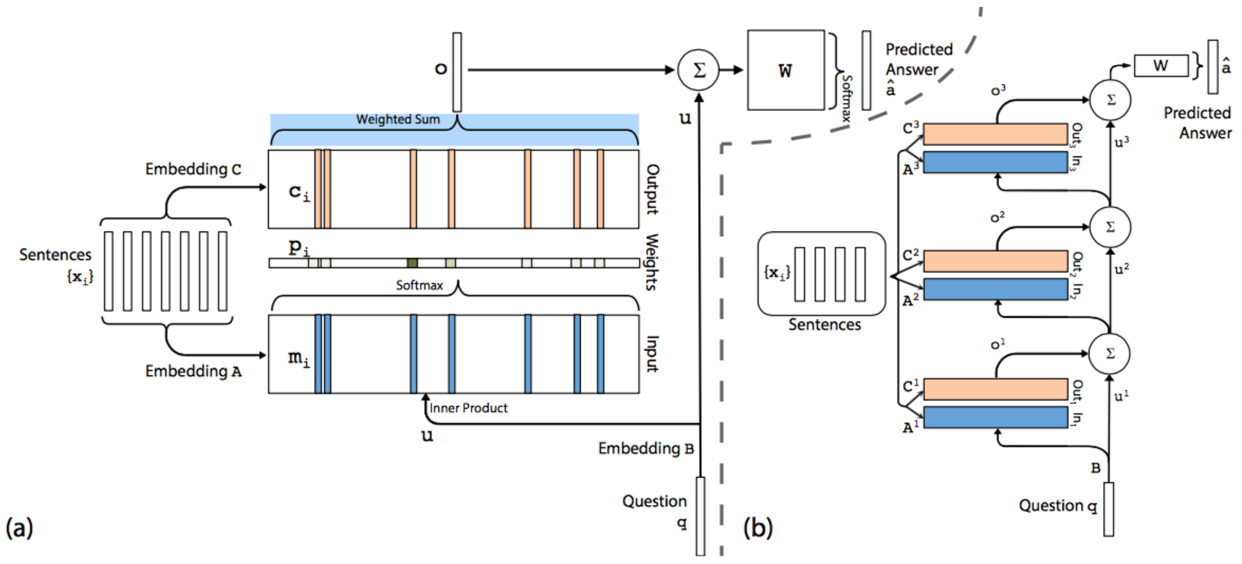


Figure 2: a) A single memory lookup layer. b) Three phases of memory lookup, followed by a classification layer

We built an end-to-end memory network following the standard positional encoding and temporal encoding strategy in Suukbaatar 2015 [11], as well as the hierarchical soft attention model that selects relevant memories in sequential phases (hops). Each layer uses tied word embeddings between layers, and tied word embeddings for both memories and training examples alike.

Temporal encoding is restricted to distinct premise-hypothesis pairs, with no attempt to induce temporal dependencies between training set examples. We force premise and hypotheses sentences to be an atomic unit in the memory tape, so that the temporal encoding can learn arbitrary inter-sentence positional importance via backpropagation. Currently, one temporal encoding strategy is shared across layers and memory / question processing. A promising future direction is to generalize this idea and form a ”lexicalized” temporal encoding tensor, which selects the appropriate temporal filter with attention.

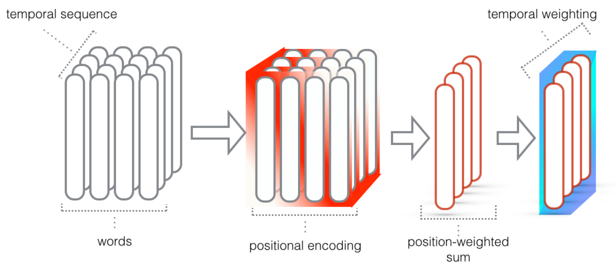


Figure 3: Positional/Temporal Encoding

Finally, the output of the memory network is modified with a separate affine layer taking the output vector to the three labels. The model was fit with a softmax prediction function and multi-class cross-entropy loss function.

4.4. Memory Tape Architectures: Memory Contents

Soft, dense attention models used for averaging tend to scale poorly over large inputs. Because the expected relevance per example is low, for any quasi-exponential underlying relevance distribution, most of the probability mass will be allocated to irrelevant vectors. Further, as the softmax probability distribution is stretched across many candidates, gradients are attenuated significantly, especially in a multi-hop network. Thus, the working candidate set of memories must remain small in order to use attention effectively.

Again, the tradeoffs between global contexts and local contexts comes into play. We use fixed context sized n corresponding to arbitrary n premise-hypothesis pairs, which encodes a dependency graph of degree n. Our rationale is that fixing the degree of this dependency graph between training examples and their support should aid in convergence for the word embeddings, and reduce variability between runs. Any sort of attention or parametric computation to rank a very large corpus of memories quickly becomes computationally intractable, so subsampling is necessary if each example should be able to draw support from any and all facts.

Since as mentioned in the previous section, disjoint examples needn't represent mutually compatible worlds, our proposed method to globally select a candidate memory set was reinforcement learning. According to an exploration-exploitation hyperparameter η, explore by sampling according to a multinomial distribution (policy) over all facts. During backpropagation, update the multinomial distribution probabilities, preserving the expectation across the selected examples; that is, constrain

$$E[p^{(i)}_{before}] = E[p^{(i)}_{after}] \tag{1}$$

and append the highly-attended memories to an LRU cache, using a threshold parameter τ). Under exploitation, just take memories from the cache.

The specified algorithm retains locally relevant contexts in the cache if there is semantic locality within a batch, and retains globally relevant contexts when there is no such locality. Further, it is stable because unseen examples would not be penalized during exploration. Finally, it is piecewise continuous, and thus differentiable, so it can be included in an end-to-end network easily, unlike a nearest-neighbors algorithm.

Over time, as η is annealed, globally relevant contexts are likely to be sampled and to appear in the cache. Unfortunately, due to time constraints, we are not able to report results of selecting contexts globally using this algorithm.

4.5. Miscellaneous

We built our models and test infrastructure using Theano and Lasagne [12]. Our models were trained on a single Nvidia GTX750M GPU and an Nvidia GeForce GTX Titan X. Source code can be found at https://github.com/mihail911/NNLI.

For the memory networks, we first built a replica of the model and hyperparameters found in [11], and verified its performance on the bAbI synthetic question-answering set with the paper's results to ensure correctness, before using components for NLI. To this end, we were able to pass all tasks that Suukbaatar et al. pass, and achieve accuracy within 5% for the tasks that do not.

During training, we ended up using Adam to optimize all of our published models. The learning rate was annealed exponentially every tenth epoch. We terminated learning early once dev accuracy failed to improve after 2 epochs, and report our best train and development set F1 scores.

5. Results and Discussion

5.1. Experiments with SICK

As discussed in Section 2.1, we initially thought SICK would be a useful prototyping dataset to assess the extent to which our neural models might learn compositional semantics without world knowledge. We found that of our baselines, the sum-of-embeddings model achieved our best results, while the memory network lagged significantly. We modified the number of memory selection layers from a single hop to 9 hops, and tried contexts varying from 1 sentence to 100, but none of the models could reach 0.60 F1 on the development set.

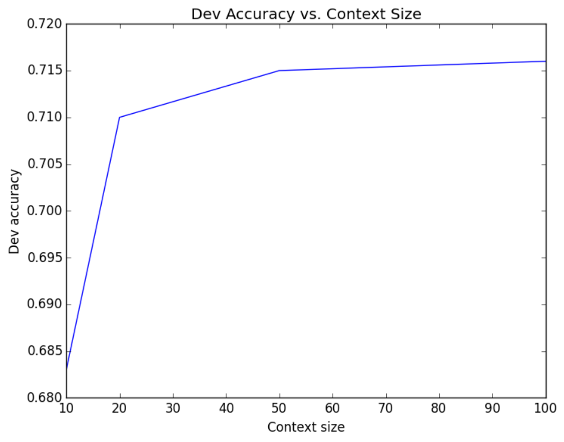
Architecture	SICK Train F1	SICK Dev F1
SumEmbeddings	0.95	**0.75**
SentLSTM	0.96	0.74
MemNet	**1.00**	0.58
RNN	0.98	0.77 (test)
LaiMaxEnt	–	**0.845** (test)

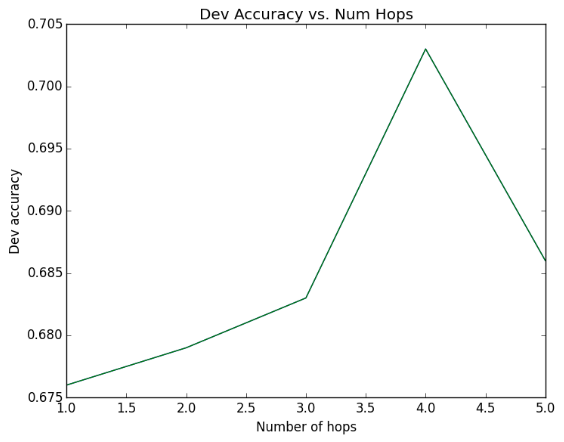
Figure 4: *SICK evaluation results. Memnet achieved best results with n_hops = 5 and context_size = 6. RNN, the recursive neural network detailed in [5], is the best neural architecture on SICK. LaiMaxEnt is the current state-of-the-art classifier on SICK for RTE [7]*

The heavy reliance of the Memory network on custom word embeddings and thus accurate counting statistics is a likely culprit for its especially poor performance on SICK. In the 4500 train examples, there are ~2400 distinct tokens, for around 3.9 sentences per token. By contrast, the train set for SNLI has 25.95 sentences per token, nearly an order of magnitude higher, which differing sentence lengths do not compensate for. Simply, the SNLI train set provides more support examples for a given token, and can produce word representations that are more likely to generalize to unseen examples. The rest of our experiments were performed on SNLI.

5.2. SNLI Memory Network Results

We performed a series of experiments of our end-to-end memory network implementation to the SNLI dataset. In particular, we were interested in seeing how modifying a few of the fundamental parameters of the memory network architecture would impact the performance on SNLI. The number of hops of inference as well as the number of context sentences used as part of the memory both may greatly influence the model's ability to learn accurate representations of the dataset semantics. The results on the SNLI dev set as we varied the number of hops of inference and size of the memory are shown below:





As we vary the number of hops our performance on dev increases from around 67.5% F1 to a maximum of around 70.3%. Our results indicate that the number of hops of inference will tend to monotonically improve the performance of the model to a local optimum after which additional number of hops only degrade the performance. As we endow the model with additional hops, we are allowing for more opportunity for the model to use the weight-representations of the context premise-hypothesis pairs to interact with query input and thereby incorporate more information from the model memory.

In general, we would expect additional hops to help the model to learn more complex semantic information over the memories and further inform its understanding of the relationship between the input premise and hypothesis. However, in this scheme, the extent to which this semantic information is useful to the model's understanding of the premise-hypothesis is very sensitive to the relevance of the chosen context. Consider that our most basic context selection scheme chooses a fixed set of contexts from the training set which could potentially provide little semantically relevant information. If the context is not relevant, then our model will essentially add noise to the query and additional hops will only add extra layers of noise that could obfuscate the model's understanding of the input. These observations are corroborated by our results as we varied the number of hops parameter. Recall that the nonlinearity in our architecture comes from the attention mechanism; if the context were equivalent to the question, then the network is a essentially glorified log-linear model.

As we vary the size of the context set, our performance on dev increases from around 68.3% F1 to around 71.5%. Our results indicate a fairly monotonic increase in F1 as context size is increased with an eventual plateau in performance. It appears that endowing the model with a larger memory can offset the somewhat simplistic context selection scheme so that even if the context sentences chosen are not especially relevant to the input, the fact that there are many of them, the model can simply have its attention mechanism learn appropriate relative weighting of the context sentences. In this way, the model is less susceptible to having its understanding of the query input compromised by noisy or insufficient semantic representations. However, after a certain point, we see less salient gains in F1 as simply having a boundless number of irrelevant sentences will not tend to better inform the model's understanding of the relationship between the input premise and hypothesis. Hence the increased benefit of more memories becomes less prominent.

Below we have our best performing models for the various architectures and design decisions explored:

Model	**Train Acc.**	**Dev Acc.**
Sum-of-embeddings	0.65	0.68
Pair-LSTM	0.875	0.795
Mem Net - Context Size	0.76	0.716
Mem Net - # Hops	0.79	0.703

Figure 5: *SNLI evaluation results. Note that the Mem Net train accuracies are artificially low because automatic termination was invoked after only a few epochs.*

We note that the sum-of-embeddings model does not perform as well as the results reported by Bowman et. al. This is not particularly surprising as we spent no time doing extensive hyperparameter searches of the model, and we do not know the exact training details of Bowman's model. Further, as it was a baseline, we were not especially concerned with maximizing the performance of this architecture on the SNLI.

Our Pair-LSTM architecture performed comparably to the results reported by Bowman. Here we were more interested in getting good performance of the model on SNLI, as we considered the possibility of using the sentence embeddings of LSTM layer of the model to inform a memory network layer that would be stacked on top of the LSTM.

We also report the best performing memory network architectures with the optimally selected context size and number of hops respectively. Though these memory networks do perform above baseline, we can attribute the somewhat subpar overall performance to the lack of powerful sentence embeddings. For example, all memory architectures we trained were able to eventually achieve very good performance on the train set, getting above 95% within 20 epochs. However, the general trend we observed was that the performance on the dev set would usually climb at first and then peak around the 2-4 epochs, indicating that the models were struggling to transfer over their knowledge from the train set to the dev set. Below we propose a number of additions to the model that can be explored in future work.

6. Future Work

Future work should investigate the use of enhanced sentence embeddings. Given that the our PairLSTM model nearly achieves 80% dev set accuracy with a rudimentary feedforward network stacked on top of sentence embeddings, we expect to see similar results from using LSTM-generated sentence embeddings to enhance the hierarchical memory attention of the memory network to similar levels. In addition, we expect this to improve the performance of deeper memory networks with multiple hops.

Given that any end-to-end memory network architecture will be sensitive to the selection of contexts that form its memory, in the future we may also investigate gating mechanisms to ensure that the model's learning is not tarnished through weighted combinations of noisy context sentences with the query premise-hypothesis. A simple first-pass at this idea may involve using a forget-gate at the uppermost layer of the memory network right before the final summation with the query that is fed as input to the softmax transformation. The hope is that if there are particularly noisy contexts, a forget-gate will learn to put less emphasis on the context in the weighted combination with the query.

Lastly, future work should experiment with selecting contexts globally using our sampling algorithm or a similar annealed exploration approach. Overall, we find little benefit in the convergence of locally determined contexts. Exploring sparse attention measures could be useful when using a large candidate memory set (in the hundreds or thousands of entries).

References

[1] Chris Billovits, Mihail Eric, and Guthrie Chris. Wordwise inference and entailment now. *Proceedings of CS224U Spring 2015*, 2015. 2

[2] Sam Bowman, Christopher Potts, and Christopher Manning. Recursive neural networks can learn logical semantics. USA. Association for Computational Linguistics. 1

[3] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Confer-*

ence on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, 2015. 1

[4] Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. A fast unified model for parsing and sentence understanding. *CoRR*, abs/1603.06021, 2016. 2

[5] Samuel R. Bowman, Christopher Potts, and Christopher D. Manning. Recursive neural networks for learning logical semantics. *CoRR*, abs/1406.1827, 2014. 5

[6] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *CoRR*, abs/1601.06733, 2016. 1, 2

[7] Alice Lai and Julia Hockenmaier. Illinois-lh: A denotational and distributional approach to semantics. *Proc. SemEval*, 2014. 2, 5

[8] 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 *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 1–8. Association for Computational Linguistics, 2014. 1

[9] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664, 2015. 1

[10] Hochreiter Sepp and Jurgen Schmidhuber. Long short-term memory. 3

[11] Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc., 2015. 3, 4, 5

[12] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. 5

[13] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *International Conference on Learning Representations*, 2015. 3