Textual Inference with Tree-structured LSTMs[1]

Kolawole J. Adebayo[a] Luigi Di Caro[a] Livio Robaldo[b] Guido Boella[a]

[a]University of Turin - {*kolawole.adebayo, luigi.dicaro, guido.boella*}@*unito.it*
[b]University of Luxembourg - *livio.robaldo@uni.lu*

Abstract

Textual Inference is a research trend in Natural Language Processing (NLP) that has recently received a lot of attention by the scientific community. Textual Entailment (TE) is a specific task in Textual Inference that aims at determining whether a hypothesis is entailed by a text. Usually tackled by machine learning techniques employing features which represent similarity between texts, the recent availability of more training data presupposes that Neural Networks that are able to learn latent feature from data for generalized prediction could be employed. This paper employs the *Child-Sum Tree*-LSTM for solving the challenging problem of textual entailment. Our approach is simple and able to generalize well without excessive parameter optimization. Evaluation done on SNLI, SICK and other TE datasets shows the competitiveness of our approach.

1 Introduction

Textual Inference stands at the heart of many NLP tasks. Presently, machines seem to be a bit far from reproducing human capability in reasoning and making semantic deductions from natural languages, e.g., spoken or written text. This is due to the phenomenon of variability and ambiguity in natural languages, since we have different ways of expressing similar ideas[2] [11]. The challenge is for machines to overcome these limitations for it to identify semantic connections such as similarity, entailment and paraphrasing etc., in a body of text. By Textual Inference, we mean the ability of Machines to recognize and quantify similarity between two text portions [1, 18], extract summary or paraphrases from a given text and most importantly, being able to infer the type of semantic connection between two text. The latter is generally referred to as Recognizing Textual Entailment (RTE), where, given two text fragments e.g., a text *T* and an hypothesis *H*, the machine's ability to determine whether T entails H is put to test [3]. This paper focuses on the Textual Entailment task.

Since Dagan et al [12] conceived the task of Recognizing Textual Entailment, it has continued to receive interest from a lot of researchers thus leading to projects and conferences dedicated to it [4, 21]. It has also inspired interests in similar NLP applications such as Factoid Question answering and information extraction [14].

TE tasks can be grouped into two i.e., the binary and multi-class categories. The former requires a Yes/No answer while the latter could be an n-way prediction[3], e.g., a 3-way task where label could be either of Entailed, Neutral or Contradict. The latter is a bit complicated than the former e.g., S_2 below contradicts S_1 while S_3 and S_4 are neutral and entailment with respect to S_1 respectively.

S_1: This church choir sings to the masses as they sing joyous songs from the book at a church.

---

[1]Kolawole J. Adebayo has received funding from the Erasmus Mundus Joint International Doctoral (Ph.D.) programme in Law, Science and Technology. Luigi Di Caro and Guido Boella have received funding from the European Union's H2020 research and innovation programme under the grant agreement No 690974 for the project "MIREL: MIning and REasoning with Legal texts". Livio Robaldo has received funding from the European Union's H2020 research and innovation programme under the grant agreement No 661007 for the project "ProLeMAS: PROcessing LEgal language in normative Multi-Agent Systems".

[2]E.g., synonymy, polysemy etc.

[3]n is in order of 3 and above. The PASCAL RTE 1-3 are examples of binary class, RTE 5-7 are examples of 3-way task and the Semeval track [13] is an example of a 5-way task

$\mathbf{S}_2$:A choir singing at a baseball game. (Contradiction)
$\mathbf{S}_3$: The church has cracks in the ceiling. (Neutral)
$\mathbf{S}_4$: The church is filled with song. (Entailment)

Most of the reported systems approached the problem as a classification task. They use hand crafted features fed into some machine learning algorithms and often relying on some knowledge resources e.g., WordNet, as well as syntactic and semantic resources, e.g., co-reference resolution, named entity recognizer, parts of speech-tagger and dependency parsing libraries. The features employed typify similarity between the text and hypothesis. The assumption is that common named entities, dependencies such as subject-verb-object-predicate agreement, presence of negation words as well as information retrieval (IR) based intuition such as TFIDF and cosine similarity might be good features for identifying entailment [17]. Other researchers approached the task simply as IR-based task without Machine learning. Reported systems in this category use the word alignment between the text and hypothesis, word overlap, word-word similarity and synonyms substitution using WordNet, surface string similarity e.g. levenhstein distance as well as other syntactic and semantic pointers. However, handcrafting features is usually time-consuming. Moreover, it is often uncertain the combination of features that might work best and ablation test is often done to weigh the features. Nevertheless, these systems rarely or slightly outperform simple baselines relying on just surface string similarity and word-overlap approaches [3].

Recently, Neural Networks such as Convolutional Neural Networks (CNN) [19], Recurrent Neural Networks (RNN) [22] and Long Short-Term Memory (LSTM) [16] have shown to possess the ability to autonomously identify latent features provided there is sufficient amount of data to learn from.

Moreover, it has been successfully applied to several NLP tasks while producing state of the art results, e.g., paraphrase detection [27], Question answering [14], text classification [30], and semantic relatedness [28]. In particular, the authors in [26] using attention-based LSTM have reported state of the art results in textual entailment.

This study builds on these works and show that a more simplified LSTM model which uses fixed raw embeddings learned from even the training data achieves similar result on some well known textual entailment datasets, even without any constraint on size of training data available as well as requiring little or no excessive hyper-parameter fine-tuning.

2 Background

The task of recognizing textual entailment (RTE) aims at making machines to mimic human inference capability, i.e., given a text T and some world knowledge H which could be a ground truth or partial truth (hypothesis), a human reading both the text and the hypothesis can recognize whether the meaning of the hypothesis can be directly inferred from the text [12], the goal is to make automated systems replicate this capability. As earlier pointed out, several authors used machine learning approaches with engineered features [15, 24].

The introduction of SNLI[4], a large dataset of 570K human generated text-hypothesis pairs by Bowman et al.,[9] and the encouraging results obtained from their LSTM-RNN neural networks model, there has been renewed interest in applying deep learning to textual entailment. Moreover, their model outperformed lexical classifiers which use word overlap, bigram and other features. The authors evaluated their models on SICK[5] and SNLI.

The authors in [26] proposed a word-by-word neural attention mechanism based on LSTM. The idea is to have two LSTMs, one reasoning over the sequence of tokens in the text while the other is reasoning on the hypothesis sequence. The second LSTM is conditioned by the first one as its memory is initialized by the output (i.e., the last cell state) of the first LSTM. This is different from the tree-structured LSTM networks proposed in [28] which being order insensitive, is able to capture semantics between two sentences. Notwithstanding, the authors evaluated their model on SNLI and obtained an accuracy of 83.7 and 83.5 on the development and test set respectively.

Baudis et al., [2] also reproduced an attention based model similar to the LSTM-based question answering model in [29]. Also, as in the former, the idea is to attend preferentially to token sequences in the sentence when building its representation. They proposed a RNN model, a CNN model as well as a hybrid RNN-CNN. The RNN captures the long-term dependencies and contextual representation of

[4] http://nlp.stanford.edu/projects/snli
[5] http://clic.cimec.unitn.it/composes/sick.html

words before being fed to the CNN. They report an accuracy of 0.829 and 0.774 respectively on SNLI train and test set respectively. Since this model also rely on embedding sequences, importance is also placed on word order. As pointed out in [28], order-insensitive models capture the semantics of natural language without recourse to syntactic structure differences. We therefore proposed an augmented tree-structure LSTM network which builds sentence representation from constituent subphrases of a text, but takes into account more compositional features for better generalization.

3 Methods

We describe the general LSTM architecture. Specifically, this work employs the *Child-Sum Tree*-LSTMs proposed by [28]. We describe our modified version of the algorithm in this work.

Long Short-Term Memory Networks

A characteristic of deep networks is their ability to autonomously learn semantic representation from text without recourse to time-consuming feature construction. Recurrent Neural Networks (RNNs) have connections that have loops, adding feedback and memory to the networks over time. This memory allows this type of network to learn and generalize across sequences of inputs rather than individual patterns. LSTM Networks [16] are a special type of RNNs and are trained using backpropagation through time, thus overcoming the vanishing gradient problem. LSTM networks have memory blocks that are connected into layers, the block contains gates that manage the blocks state and output. These gates are the *input* gates which decides the values from the input to update the memory state, the *forget* gates which decides what information to discard from the unit and the *output* gates which decides what to output based on input and the memory of the unit. LSTMs are thus able to memorize information over a long period of time since this information are stored in a recurrent hidden vector which is dependent on the immediate previous hidden vector. A unit operates upon an input sequence and each gate within a unit uses the sigmoid activation function to control whether they are triggered or not, making the change of state and addition of information flowing through the unit conditional. We follow the definition and notation of LSTM in [28].

At each time step *t*, let an LSTM unit be a collection of vectors in $\mathbb{R}^d$ where *d* is the memory dimension: an *input gate* i_t, a *forget gate* f_t, an *output gate* o_t, a *memory cell* c_t and a *hidden state* h_t. The state of any gate can either be open or closed, represented as [0,1]. The LSTM transition can be represented with the following equations (x_t is the an input vector at time step *t*, σ represents sigmoid activation function and $\odot$ the elementwise multiplication. The u_t is a tanh layer which creates a vector of new candidate values that could be added to the state) :

$$
\begin{aligned}
i_t &= \sigma\Big(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}\Big),\\
f_t &= \sigma\Big(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}\Big),\\
o_t &= \sigma\Big(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}\Big),\\
u_t &= \tanh\Big(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}\Big),\\
c_t &= i_t \odot u_t + f_t \odot c_{t-1},\\
h_t &= o_t \odot \tanh c_t
\end{aligned}
\tag{1}
$$

Tree-Structured LSTMs

Tree-LSTMs are specialized LSTMs that adopt the tree-structure topology, i.e., at any given time step *t* the LSTM is able to compose its states from input vector and hidden states of its child-nodes simultaneously. This is unlike the standard LSTM that assumes a single child per unit since the gating vectors and memory cell updates are dependent on the states of all child-nodes while also maintaining a forget state for each child. A well-known variant of this structure, i.e., the *Child-Sum Tree*, has been proposed by Tai [28]. The *Child-Sum Tree* LSTMs transition is represented by the following equation, where C_j is the set of children in a node j and $k \in C_j$.

$$\widehat{h_j} = \sum_{k \in C_{(j)}} h_k \tag{2}$$

$$i_j = \sigma\left(W^{(i)} x_j + U^{(i)} \widehat{h_j} + b^{(i)}\right) \tag{3}$$

$$f_j k = \sigma\left(W^{(f)} x_j + U^{(f)} \widehat{h_k} + b^{(f)}\right) \tag{4}$$

$$o_j = \sigma\left(W^{(o)} x_j + U^{(o)} \widehat{h_j} + b^{(o)}\right) \tag{5}$$

$$u_j = \tanh\left(W^{(u)} x_j + U^{(u)} \widehat{h_j} + b^{(u)}\right) \tag{6}$$

$$c_j = i_j \odot u_j + \sum_{k \in C_{(j)}} f_{jk} \odot c_k \tag{7}$$

$$h_j = o_j \odot \tanh c_j \tag{8}$$

Tree-LSTM for Textual Entailment

We use the *Child-Sum Tree*-LSTM to generate sentence representation for both the text and hypothesis by committing a tree to each of text and hypothesis. Assuming both the text and hypothesis are sentences, each with constituent words that are well connected. The structural connection between the words form a deep branching graph, with elements and their dependencies (in case of dependency parsing) where each connection in principle unites a superior term and an inferior term. The inferior term defers to its superior, thus distinguishing between semantically useful words like nouns and verbs to say, a determinative word. With constituency parsing, a phrase-like one-to-one correspondence between the words is observed. The *Child-Sum Tree*-LSTM works better for dependency parse tree representation of a sentence, where each child is a node in the representation. For each node, the LSTM unit takes as input the vectors of its superior word to which it is dependent. In the case of constituency parsing, an LSTM unit takes as input the exact vector of the node.

Considering our n-way classification, given a set of classes Y whose label cardinality corresponds to n, i.e., y_1, y_2, y_3...y_n are the given labels. First, we obtain a representation r_j = (h_{txt}, h_{hyp}) from both the text and hypothesis using the dependency tree representation. Recall that an inferior node takes the fixed raw vectors of its superior while a superior node takes the sum of its vectors and that of all its dependents. At each superior node of the text and hypothesis tree, we encode the entailment relationship as the distance and angle between their element-wise summed vectors and angle of their vectors product. We use the * operator to denote vectors product. We predict the label $\hat{y}_j$, given the input representation r_j observed which encodes the entailment relationship at nodes in the subtree rooted at j. The classifier takes the hidden state h_j at the node as input. The process is represented by the equations below:

$$h_{\times} = h_{txt} \odot h_{hyp} \tag{9}$$

$$h_{+} = h_{txt} + h_{hyp} \tag{10}$$

$$h_{\curlywedge} = |h^{*}_{txt} \odot h^{*}_{hyp}| \tag{11}$$

$$h_s = \sigma\left(W^{(\times)} h_{\times} + W^{(\curlywedge)} h_{\curlywedge} + W^{(+)} h_{+} + b^{(h)}\right) \tag{12}$$

$$\widehat{p}\theta(y|\{r\}_j) = softmax\left(W^{(p)} h_s + b^{(p)}\right) \tag{13}$$

$$\widehat{y_j} = \arg\max_{y} \widehat{p}\theta(y|r_j) \tag{14}$$

4 Evaluation

RTE PASCAL challenge [12] is an important avenue for researchers to submit TE systems for public evaluation. We evaluated our system on the PASCAL RTE3 dataset which consists of 800 sentence pairs both for development and test set. The RTE3 dataset has only two classes, i.e., the entailment relation can either be true or false. The SEMEVAL track offering similarity and entailment task also makes use of the SICK dataset[6]. SICK consists of 10000 sentence pairs annotated for use in both sentence similarity and 3-way entailment task. Finally we evaluated our system on the SNLI corpus [9] which till date is the biggest manually constructed entailment dataset publicly available. However, we only use half of the training data in our experiment. Nevertheless, the result obtained shows that our model is able to generalize quite well.

In the context of our ongoing work in the legal domain[7] [5, 8, 7], we explored the option of evaluating our models on a dataset deeply rooted in legal nuances. The three datasets cited above contain sentences that are domain independent and thus have no technical jargons. Our goal is to see how our model would perform within the complex legal domain. Legal texts seem intuitive as they have some peculiarities which set them apart from day-to-day natural language since they employ legislative terms. For instance, a sentence could reference another sentence (e.g., an article) without any explicit link to its text from inside the quoting text. Also, sentences could be long with several clausal dependencies, that is notwithstanding of its inter and intra-sentential anaphora resolution complexity. We opined that a system that is able to achieve good result in this scenario would generalize well given other domain dependent texts.

We used the COLIEE dataset[8] which is a Question-Answering legal corpus made available in the context of COLIEE Legal Information Extraction/Retrieval challenge. Task 2 of the challenge addresses TE, such that, given a sentence and a Query, a system identifies if there is an entailment or not. We provide our evaluation on the 2015 training and test set.

Experiment

We implemented our adaptable Tree-Structured LSTM as proposed by [28]. We obtained dependency tree of both text and hypothesis using Stanford dependency parser [10]. Instead of encoding both text and hypothesis as one-hot encoding representation of the token sequences, we used trained word embedding vectors with fixed weight throughout the experiments. We used 300-dimensional Glove vectors [25]. However, we also generated our own 300-dimensional word embeddings using the popular Word2Vec algorithm[9] [23] on a minimal set of text including SNLI, RTE2, RTE3 and SICK datasets. For the experiment on COLIEE corpus, we included its text in those used to generate the first Word2Vec word embeddings, using the same parameter for training, we obtained a separate embeddings to use for this particular run. An oddity is that even with the embeddings generated from a rather small text, the obtained vectors still capture the semantics of the sentences. We therefore compare the result obtained when we use Glove as well as our trained embeddings.

We used the Keras[10] Deep Learning library to construct two models which only differ based on their configuration depth. Model1 is quite basic, it has a single hidden layer with 300 LSTM units. Model2 mimics the feed-forward MLP networks in that we maintain a stack of layers to increase the depth and scale-up the performance. The first hidden layer size = 300, the second hidden layer size = 200, the third hidden layer size = 150. Depending on the cardinality of classification in each dataset, the sigmoid output layer predicts the right label based on the class distribution. We observed that applying a heavy dropout between 0.4 and 0.5 in the hidden layers lowers the performance. Hence, a moderate dropout between 0.1 and 0.2 was used depending on the performance on the validation data.

As earlier pointed out, our goal is to avoid excessive parameter fine-tuning specifically for each dataset. Because of this, we maintain a uniform batch-size of 25. We used ADAM, a stochastic optimizer with learning rate set at 0.01 and a decay value of 1e-4 as well as momentum of 0.9. Typically, we set the

[6] `http://clic.cimec.unitn.it/composes/sick.html`

[7] Specifically, the MIREL project: `http://www.mirelproject.eu`, which is drawn from our past project EUCases [6].

[8] `http://webdocs.cs.ualberta.ca/~miyoung2/COLIEE2016`

[9] No stopword removal or stemming was done, all terms were lemmatized. We used the skip-gram representation with context window of 10, dimension of 300 and min-count of 5 using the gensim package.

[10] `https://github.com/fchollet/keras`

number of training epochs to 250. However, throughout the experiments, we track the model for over-fitting by monitoring the peak accuracy on the validation data as well as the epoch where the model does not seem to learn sufficiently again and the accuracy begins to drop or does not increase anymore over a period of time. Usually, we pick this epoch for our test runs.

Tables 1, 2 and 3 display the results obtained on the datasets used. For table 1, we used the results from Rocktaschel et al.,[26], Baudis et al., [2] and Bowman et al., [9] as the baseline systems on SNLI and SICK respectively. For PASCAL-RTE3, we did not include any baseline system since there is no recent work which use similar deep learning approach on that dataset. For the result on COLIEE dataset in table 3, we include as our baseline, the best and the baseline results as reported by the authors in [20]. Note that we only use a randomly sampled half of the SNLI data for training due to computation time for our experiments were conducted on the CPU and not the GPU. We also give a comparison of the performances of the models with Glove vectors and our generated Word2Vec embeddings from the training data. It turns out that there is no clear distinction between the results obtained in this respect. In fact, instances exist where we obtained higher accuracy with our trained embeddings. We also observed that the depth/configuration complexity of the network influences accuracy by some order of magnitude since we obtained marginally improved accuracy with *model2* that has more hidden LSTM layers stacked.

Model	SNLI Train	SNLI Test	SICK Train	SICK Test
Model1 (Glove)	84.76	78.30	85.10	76.00
Model1 (w2vec)	84.40	78.00	-	-
Model2 (Glove)	85.30	79.40	88.00	80.10
Model2 (w2vec)	85.90	78.30	-	-
Neural Attention[26]	85.30	83.50	-	-
attn1511 [2]	82.90	77.40	85.80	76.70
LSTM-RNN [9]	84.80	77.60	99.90	80.80

Table 1: Evaluation on SNLI and SICK datasets

Model	Train	Test
Model1 (Glove)	95.76	86.90
Model1 (w2vec)	94.80	82.24
Model2 (Glove)	95.30	87.20
Model2 (w2vec)	93.20	87.60

Table 2: Evaluation on PASCAL-RTE3 dataset

Model	Train	Test
Model1 (w2vec)	76.00	64.19
Model2 (w2vec)	81.33	70.10
Baseline [20]	-	55.87
Best [20]	-	63.87

Table 3: Evaluation on COLIEE dataset

Discussion and Conclusions

In this paper, we have presented a *Child-Sum Tree*-LSTM model for solving the challenging problem of textual entailment. We showed that our approach performs competitive with other state of the art deep learning systems applied to textual entailment. We reported our evaluations on SNLI, SICK, and relatively small PASCAL-RTE3 datasets. We evaluated our models on a domain specific Question-Answering corpus with background in the field of Law. Results are shown in Tables 1, 2, and 3.

The table 1 reveals slight over-fitting in our models since it did not generalize well when compared to the baselines, i.e., we seem to obtain better accuracy on the training data than the authors in [26] while their system generalize better on unseen data. However, observing deeply, we see only slight variations between train and test accuracy. Probably, the model is not able to minimize error or learn better. For the SNLI dataset, it is possible that there is an imbalance in class distribution when we randomly shuffled the data in order to pick one-half of the original training data used in our experiments. Class imbalance can skew the learning curve, thus leading to improper generalization. Note that for SICK, we split the data into train, validation and test set at the ration 60:20:20 respectively. Likewise, COLIEE has no official validation set, we therefore split the test set into validation and test in the ratio 50:50.

The seemingly poor result obtained on the COLIEE corpus is quite noticeable. Two issues might be connected to this. First, the general consensus is that with deep neural networks, more data gives a better generalization. In fact, a lexicalized classifier with carefully handcrafted features or a simple n-gram based classifier might generalize better with little training data in comparison to a very complex deep network architecture with little data to learn from. The COLIEE corpus with less than 500 training sentence pairs is some order of magnitude smaller than the PASCAL-RTE3 corpus which is generally agreed to be insufficient for training neural networks[11]. Secondly, we believe that the result on PASCAL-RTE3 is not totally bad, given that the training corpus is of similar size. Another factor may be considered. For instance, we believe that the technicality of the text and the fact that it is domain specific pose some issues. First, the embedding was not obtained from the training of a large legal text from which the semantics of legal jargons are best captured. Furthermore, compared to SNLI and SICK, both the text and hypothesis in COLIEE are unusually long, which as pointed out earlier, is a clear characteristic of legal text. However, we posit that more training data and the inclusion of more legal data for generating word embedding might improve results. Nevertheless, we report improved result to the baselines from [20].

Compared to all the benchmarked baselines, our approach is simple and required little or no hyperparameter fine-tuning. Obtaining a better generalization should be possible by optimizing various parameters, e.g., a grid search optimization of some parameters might lead to a more accurate model. Also, introducing more layers in our model might achieve better generalization even though it can also lead to model memorizing features without any significant improvement in the learning curve.

References

[1] E. Agirrea, C. Baneab, D. Cerd, M. Diabe, A. Gonzalez-Agirrea, R. Mihalceab, G. Rigaua, J. Wiebef, and B. Donostia. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. *Proceedings of SemEval*, pages 497–511, 2016.

[2] Petr Baudiš and Jan Šedivỳ. Sentence pair scoring: Towards unified framework for text comprehension. *arXiv preprint arXiv:1603.06127*, 2016.

[3] Luisa Bentivogli, Peter Clark, Ido Dagan, Hoa Dang, and Danilo Giampiccolo. The seventh pascal recognizing textual entailment challenge. *Proceedings of TAC*, 2011, 2011.

[4] Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. The fifth pascal recognizing textual entailment challenge. *Proceedings of TAC*, 9:14–24, 2009.

[5] G. Boella, L. Di Caro, L. Humphreys, L. Robaldo, R. Rossi, and L. van der Torre. Eunomos, a legal document and knowledge management system for the web to provide relevant, reliable and up-to-date information on the law. *Artificial Intelligence and Law*, to appear, 2016.

[6] Guido Boella, Luigi Di Caro, Michele Graziadei, Loredana Cupi, Carlo Emilio Salaroglio, Llio Humphreys, Hristo Konstantinov, Kornel Marko, Livio Robaldo, Claudio Ruffini, Kiril Simov, Andrea Violato, and Veli Stroetmann. Linking legal open data: Breaking the accessibility and language barrier in european legislation and case law. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law*, ICAIL '15, New York, NY, USA, 2015. ACM.

[7] Guido Boella, Luigi Di Caro, Daniele Rispoli, and Livio Robaldo. A system for classifying multi-label text into eurovoc. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law*, ICAIL '13, pages 239–240, New York, NY, USA, 2013. ACM.

[8] Guido Boella, Luigi Di Caro, and Livio Robaldo. *Semantic Relation Extraction from Legislative Text Using Generalized Syntactic Dependencies and Support Vector Machines*, pages 218–225. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[9] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[10] Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750, 2014.

[11] e.g., compared to 570,000 sentence pairs from SNLI

[11] Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. Recognizing textual entailment: Rational, evaluation and approaches–erratum. *Natural Language Engineering*, 16(01):105–105, 2010.

[12] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer, 2006.

[13] Myroslava O Dzikovska, Rodney D Nielsen, Chris Brew, Claudia Leacock, Danilo Giampiccolo, Luisa Bentivogli, Peter Clark, Ido Dagan, and Hoa T Dang. Semeval-2013 task 7: The joint student response analysis and 8th recognizing textual entailment challenge. Technical report, DTIC, 2013.

[14] Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. Applying deep learning to answer selection: A study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 813–820. IEEE, 2015.

[15] M. Gaona, A. Gelbukh, and S. Bandyopadhyay. Recognizing textual entailment using a machine learning approach. In *Mexican International Conference on Artificial Intelligence*. Springer, 2010.

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8), 1997.

[17] S. Jimenez, G. Duenas, J. Baquero, A. Gelbukh, J. Bátiz, and A. Mendizábal. Unal-nlp: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 732–742, 2014.

[18] Adebayo Kolawole John, Luigi Di Caro, and Guido Boella. Normas at semeval-2016 task 1: Semsim: A multi-feature approach to semantic text similarity. *Proceedings of SemEval*, 2016.

[19] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[20] M.Y. Kim, Y. Xu, and R. Goebel. A convolutional neural network in legal question answering.

[21] Bernardo Magnini, Roberto Zanoli, Ido Dagan, Kathrin Eichler, Günter Neumann, Tae-Gil Noh, Sebastian Pado, Asher Stern, and Omer Levy. The excitement open platform for textual inferences. In *ACL (System Demonstrations)*, pages 43–48, 2014.

[22] LR Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 2001.

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

[24] Shachar Mirkin, Ido Dagan, and Eyal Shnarch. Evaluating the inferential utility of lexical-semantic resources. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 558–566. Association for Computational Linguistics, 2009.

[25] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43, 2014.

[26] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[27] R. Socher, E. Huang, J. Pennin, C. Manning, and A. Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809, 2011.

[28] 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.

[29] Ming Tan, Bing Xiang, and Bowen Zhou. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*, 2015.

[30] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657, 2015.