Capturing Pragmatic Knowledge in Article Usage Prediction using LSTMs

Jad Kabbara **Yulan Feng** **Jackie Chi Kit Cheung**
School of Computer Science
McGill University
Montreal, QC, Canada
{jad@cs, yulan.feng@mail, jcheung@cs}.mcgill.ca

Abstract

We examine the potential of recurrent neural networks for handling pragmatic inferences involving complex contextual cues for the task of article usage prediction. We train and compare several variants of Long Short-Term Memory (LSTM) networks with an attention mechanism. Our model outperforms a previous state-of-the-art system, achieving up to 96.63% accuracy on the WSJ/PTB corpus. In addition, we perform a series of analyses to understand the impact of various model choices. We find that the gain in performance can be attributed to the ability of LSTMs to pick up on contextual cues, both local and further away in distance, and that the model is able to solve cases involving reasoning about coreference and synonymy. We also show how the attention mechanism contributes to the interpretability of the model's effectiveness.

1 Introduction

Correctly performing pragmatic reasoning is at the core of many NLP tasks such as information extraction, automatic summarization, and machine translation. We focus in this paper on definiteness prediction, the task of determining whether a noun phrase should be definite or indefinite. In English, one instantiation of this task is to predict whether to use a definite article (*the*), indefinite article (*a(n)*), or no article at all. It has applications in machine translation (Heine, 1998; Netzer and Elhadad, 1998), and in L2 grammatical error detection and correction (Han et al., 2006).

Definiteness prediction is an interesting testbed for pragmatic reasoning, because both contextual and local cues are crucial to determining the acceptability of a particular choice of article. Consider the following example:

(1) *A/#the man entered the room. The/#a man turned on the TV.*

Factors such as discourse context, familiarity, and information status play a role in determining the choice of articles. Here, *man* is introduced into the discourse context by an indefinite article, then subsequently referred to by a definite article. On the other hand, non-context-dependent factors such as local syntactic and semantic restrictions may block the presence of an article. For example, demonstratives (e.g., *this*, *that*), certain quantifiers (e.g., *no*), and mass nouns (e.g., *money*) do not permit articles.

In this work, we investigate Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), a subclass of recurrent neural networks (RNNs) which have been popular recently in a variety of NLP tasks (Sutskever et al., 2014; Mikolov et al., 2010; Dyer et al., 2015). A number of reasons are often cited for the impressive performance gains of RNNs (Goldberg, 2015). First, they are able to take advantage of the patterns inherent in the data to learn features and representations suitable for complex interpretation tasks. Second, they can learn connections between processing units in the same layer, allowing the network to capture relations and patterns over an unbounded number of timesteps. Third, they provide an easy and natural way to inject external semantic knowledge by initializing the parameters of the model in an informed way. For example, the word embeddings can be initialized using pre-trained vectors such as `word2vec` (Mikolov et al., 2013).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2625–2634, Osaka, Japan, December 11-17 2016.

We compare several versions of LSTMs to explore how each of these factors affects article usage prediction. We also explore a version of LSTMs that employs an attention mechanism. The primary motivation for the use of an attention mechanism is to investigate whether such LSTM models focus on certain parts of the sentence when making predictions, and if so, to gain more insight into what parts of the sentence affect the model's prediction.

Our model achieves state-of-the-art performance on definiteness prediction, outperforming a previous, classification-based model by De Felice (2008). Our best model achieves 96.63% accuracy on the WSJ/PTB corpus, representing a relative error reduction of 51% compared to the previous state of the art. Each of the factors we examined (initializing with pre-trained vectors, giving more context, giving POS tags, attention) contributes to the performance of the model, though in different degrees. We perform a number of analyses to understand the behavior of the models, and show in particular how the attention mechanism can be useful for interpreting the model predictions.

The most interesting contribution of this paper is highlighting the suitability of LSTMs for tackling complex cases of article usage where there is no obvious local cue for prediction. We find evidence that LSTMs given an extended context window can resolve cases of article usage that seem to require reasoning about coreferent entities involving synonymy. Our results suggest that recurrent neural network models such as LSTMs are a promising approach to capturing pragmatic knowledge.

2 Related Work

Characterizing definite descriptions has been one of the first problems considered in semantics and pragmatics, and indeed in the philosophy of language. Russell (1905) analyzed definite descriptions as having quantificational force by asserting the existence and uniqueness of the definite NP, whereas Strawson (1950) emphasized their anaphoric nature, and their ability to trigger a presupposition about the existence of the noun phrase in the discourse context.

In terms of corpus-based studies, Poesio and Vieira (1998) analyzed a subset of definite descriptions found in the WSJ, and asked annotators to classify them according to their function. They found that 50% of definite descriptions were classified as discourse-new, 30% as anaphoric, and 18% as associative or bridging. Lee et al. (2009) investigated the role of contextual information in predicting article usage in a user study.

There has been a variety of previous models on article prediction (Knight and Chander, 1994; Minnen et al., 2000; Han et al., 2006; Gamon et al., 2008). De Felice (2008) framed it as a MaxEnt classification task, extracting a number of linguistically motivated features from the context of each head noun. Turner and Charniak (2007) took an approach more similar to our own, viewing article selection as a language modelling problem by training a parser-based language model on the WSJ and North American News Corpus.

More generally, the use of distributed representations in discourse processing is becoming more widespread, with applications in discourse parsing (Kalchbrenner and Blunsom, 2013; Ji and Eisenstein, 2014; Li et al., 2014) and implicit discourse relation detection (Ji and Eisenstein, 2015). There have also been recent efforts to build distributed representations of linguistic units larger than a sentence (Le and Mikolov, 2014; Li et al., 2015). Hill et al. (2016) investigated several variants of LSTMs to predict function and content words, but they did not consider determiners in their study.

3 Model

We introduce several variants of LSTM models for definiteness prediction. LSTMs extend standard RNNs by providing additional memory control that can capture long-term dependencies between data samples that would be difficult to capture with standard RNNs. The memory control allows the model to carefully regulate how much the current input affects the new memory state, how much the previous memory state affects the new memory state and what elements of the memory should play a role in generating the output. We first present the LSTM variants that we experimented with, then describe the input representations which we used for the models.

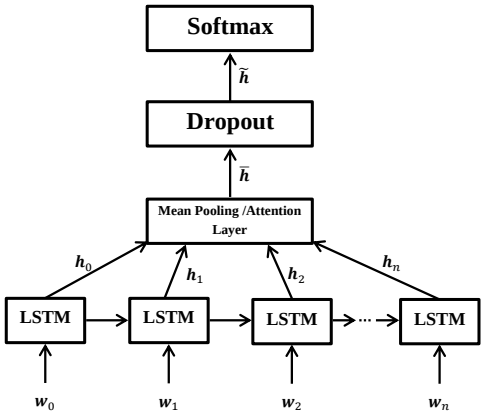

Figure 1: LSTM-based neural network model.

3.1 Vanilla Model

In a standard, "vanilla" LSTM (Figure 1), the model is given an input sequence, $w_0, w_1, \ldots, w_n$. Each w_i is a vector representation of an input word at timestep i, which is fed to an LSTM cell consisting of an input gate, an output gate, a forget gate, a cell state, and a hidden state (see (Graves, 2013) for more details). The outputs of the LSTM cells are fed to a mean pooling layer:

$$\overline{h} = \frac{1}{n} \sum_i h_i \tag{2}$$

Then, the dropout layer randomly suppresses output neurons of $\overline{h}$ during the forward pass of training with a pre-defined probability by setting them to zero. This in effect acts as a regularization mechanism (Srivastava et al., 2014). In the last stage, we perform classification using a three-input softmax unit.

3.2 Attention-based model

Attention mechanisms have recently attracted considerable interest in the deep learning community. Attention mechanisms are loosely inspired by theories of human visual attention in which specific regions of interest have high focus compared to other regions.

Compared to "vanilla" model, the attentive LSTM model replaces the mean pooling layer with a learned attention layer that leads to a weighted average of the timesteps. For each timestep i, the model learns:

$$c_i = \tanh(Mh_i + b) \tag{3}$$

$$a_i = \exp(c_i) / \sum_j \exp(c_j), \tag{4}$$

where M is a learned weight matrix, b is a bias term, c_i reflects the importance of the input at that timestep, and a_i is a normalized version of the importance over all timesteps in the sample. Then, the output is calculated as:

$$\overline{h} = \sum_i a_i h_i. \tag{5}$$

3.3 Input representation

We map each input token w_i to some learned embedding, which is then fed to the input layer of the LSTM. We investigate several different linguistically motivated factors for building this representation

Table 1: Number and distribution of noun phrases in the PTB corpus by the article type.

	none	the	a(n)
Training set	150606	49117	23907
Development set	11987	3898	1867
Test set	14676	4848	2155

beyond the standard "vanilla" LSTM model. First, we experiment with incorporating or not POS tags as a form of syntactic knowledge (+/−**POS**). To incorporate such knowledge, we concatenate the POS to the aforementioned learned embedding before feeding it to the input layer. The other factor that we consider is how to initialize the embedding associated with each word-type. We experiment with initializing it randomly, or with pre-trained vectors, which could inject knowledge derived from a large, external corpus. The intuition behind incorporating pre-trained vectors is that they might help the model recognize bridging references of those involving synonyms (e.g., *a* ***house*** *... the* ***home***). Thus, we compare the following options:

- **Random**: The embedding is initialized randomly.
- **word2vec**: The embedding is initialized by the SkipGram vectors of Mikolov et al. (2013) trained on the Google News corpus (about 100 billion words).
- **GloVe**: The embedding is initialized by the global vectors Pennington et al. (2014) that are trained on the Common Crawl corpus (840 billion tokens).

Both word2vec and GloVe word embeddings consist of 300 dimensions. To test whether compressing those embeddings would lead to a better prediction performance, we investigated the use of PCA to reduce the dimensionality of the word embeddings, but found that this did not influence performance on the development set.

4 Dataset

We use the Penn Treebank (PTB) corpus (Marcus et al., 1993) with the standard section splits for training (01–23), development (00, 24) and testing (22, 23). We extract all of the noun phrases present in the parsed corpus whose head noun's POS tag is one of `NN`, `NNS`, `NNP`, or `NNPS`. Also, we do not lemmatize, and ignore case and punctuation. Finally, we remove any occurrence of the relevant determiners (*the*, *a*, *an*) from all the data sets (training, development, test). The number of samples in the dataset is shown in Table 1.

In our experiments, we adopt one of two different sample configurations: the first focusing on a local context and the second extending the context to encompass one or more sentences. Specifically, for the former, we define a sample to be the set of tokens from the previous head noun of a noun phrase up to and including the head noun of the current noun phrase. For example, take the following passage (head nouns indicated in bold):

(6) For six **years**, T. Marshall Hahn **Jr.** has made corporate **acquisitions** in the George Bush **mode**: kind and gentle.

The following samples –relying on local context– are shown, with their labels: *For six years* – 'none', *T. Marshall Hahn Jr* – 'none', *has made corporate acquisitions* – 'none', *in George Bush mode* – 'the'.

For the sample configuration relying on the extended context, the sample is constructed in the same way (as described above) and, in addition, tokens from the previous sample(s) are added sequentially (in reverse) until a pre-specified total number of tokens per sample is reached. That way, the sample not only reflects local information from the current noun phrase, but also information that is contained in a previous sentence (or more).

Having these two distinct sample configurations allows us to compare, in general, the learning performance of the LSTM network between the two cases and to investigate, in particular, whether those networks are able to resolve cases that exhibit long range dependencies and complex cases that require contextual clues that go beyond the local context of a given article.

5 Experiments

We compare our LSTM models against the following systems. The first is the most frequent baseline (**Baseline**), which labels all noun phrases with the most frequent class of none. The second is a reimplementation of the classification-based method of De Felice (2008) (**LogReg**), which extracts features from a fixed context window for a multinomial logistic regression classifier. Our implementation differs from the one described in that work in two respects. First, we could not gain access to the list of mass and count nouns that she did. We approximated this by crawling the British National Corpus (Burnard, 2007) for instances of nouns that appear after *a/many/few* to identify count nouns, and *much/little/bit of* for mass nouns. During feature extraction on the PTB, nouns that have not been previously encountered are given the label unknown.

5.1 LSTM training

Many variations of the LSTM cell have been proposed in the literature; however, since we implement our model using Lasagne[1], we use the LSTM cell implemented by Lasagne and presented in (Graves, 2013). We use the AdaDelta algorithm (Zeiler, 2012) for learning the optimal network parameters and word embeddings. The model has a number of hyperparameters that were tuned on a development set. We list below the range that we explored (min, max) and the final value that was used [val]:

- Word embedding and hidden-layer dimensionality: (50, 200), [100]
- Dropout probability: (0.2, 0.7), [0.6]
- Minibatch size: (20, 200), [100]

6 Results

We present the results of our experiments in Table 2. Our LSTM models outperform LogReg and the baseline in terms of accuracy despite not having access to an extensive set of hand-crafted linguistic features. Incorporating pre-trained word vectors into the initialization results in a small but consistent improvement in performance, for both the version without and with POS tags. We notice, however, that the particular choice of the embedding, does not seem to make much of a difference, with GloVe very slightly outperforming word2vec. In all cases, using an attention mechanism in the learning model led to an improvement in the accuracy. We also contrast the learning performance of the network when fed samples relying on local context versus samples relying on the extended context. Most interestingly, POS tags do not seem as necessary for high performance in the extended case, as the discrepancy between using or not POS tags is almost negligible. This could be explained by the fact that an extended context allows the network to learn relevant syntactic cues from context which were only available with explicit POS tags in the case of local context. Overall, the best setting is using extended contexts and GloVe embeddings with POS tags and attention in the LSTM model, at 96.63% accuracy.

Our results improve upon different previously reported accuracies on this task. De Felice (2008) reported the best previous result of 92.15% accuracy, though this was on the BNC corpus. Our reimplementation of this work (LogReg) achieved 93.07 % on the WSJ-PTB corpus. Turner and Charniak (2007) reported an accuracy of 86.63% on ten-fold cross-validation over WSJ with additional training on an external corpus. We are able to achieve a higher accuracy on comparable data using a smaller amount of training data, though the models initialized with pre-trained word embeddings could arguably be said to be trained on large amounts of text.

[1] `http://lasagne.readthedocs.org/`

Method			**Accuracy** (%)	
Baseline			67.70	
LogReg			93.07	
	Init.	**POS**	**Local contexts**	**Extended contexts**
LSTM	Random	−POS	83.94 - 83.96	95.82 - 96.08
LSTM	word2vec	−POS	84.91 - 84.93	96.40 - 96.53
LSTM	GloVe	−POS	85.35 - 85.75	96.37 - 96.43
LSTM	Random	+POS	94.11 - 94.12	95.95 - 96.08
LSTM	word2vec	+POS	94.50 - 94.52	96.20 - 96.25
LSTM	GloVe	+POS	94.64 - 94.67	96.38 - **96.63**

Table 2: Accuracy results for article prediction on the WSJ/PTB corpus. The LSTM models are distinguished by their initialization method (**Init.**), whether or not they used POS tags (**POS**), and whether they were give local or extended context. For each result for the LSTM models, we show the result for the model without attention (left) and with attention (right).

Method	**Class**	**P**	**R**	**F1**
Baseline	none	67.70	100.0	80.74
LogReg	a	75.05	70.49	72.70
	none	98.63	98.41	98.52
	the	84.10	86.94	85.50
Local LSTM+a	a	76.30	73.04	74.63
+ GloVe	none	99.55	99.42	99.49
+ POS	the	87.60	89.60	88.59
Extended LSTM+a	a	86.88	91.28	89.02
+ GloVe	none	98.02	96.99	97.50
+ POS	the	95.34	96.25	95.79

Table 3: Precision, Recall, and F1 results by class. We only show the 'none' class results of the baseline, as the baseline is to label everything as 'none'. LSTM+a represents the attention-based model.

These numbers conceal the large differences in performance between the different classes. Table 3 shows the breakdown of the results for each model by class label in terms of precision, recall, and F1. Because of space limitations, we only include results for selected models. Overall, the "none" class is the easiest to predict. This is expected, as various syntactic and semantic cues are highly indicative of the "none" class (e.g., presence of a demonstrative in the context or the head noun being a named entity).

Also, we consider the performance of the models on named entities versus non-named entities, as identified by the POS tag on the head noun. Table 4 shows the accuracy results divided into the two classes. As expected, named entities are easier for the model to predict, due to conventions about article usage related to named entities in English. The LogReg model already achieves high performance on named entities, and it is conceivable that with a larger amount of training data, it can approach the performance of the LSTM models, because it will see more conventions about named entities or classes of named entities. Both LSTM models improve on performance on both classes, and actually obtain a greater absolute improvement on the non-named entities. This could be seen as evidence that they actually are making better predictions for those cases that require long-range dependencies.

Finally, we conducted two-tailed paired sign tests (with a level of significance $\alpha = 0.05$) to examine whether the best performing LSTM model (GloVe + POS with attention and extended context) significantly outperformed the LogReg baseline, as well as other versions of the LSTM model, namely the best LSTM with local context, the best LSTM with random initialization and the best LSTM without

Method	NE	non-NE
Baseline	86.98	61.76
LogReg	97.27	91.77
Local LSTM+a + GloVe + POS	**98.88**	93.44
Extended LSTM+a + GloVe + POS	97.62	**96.48**

Table 4: Test set accuracy results for named entities (N = 5100) and non-named entities (N = 16579).

	Simple Cases		Complex Cases	
	fixed	dup.	syn.	semantics
	86	6	8	100
Total	92		108	

Table 5: Classification of 200 samples incorrectly predicted by the best performing LSTM model on the local context but correctly predicted by the best performing model on the extended context. With categories of simple cases including fixed expressions(*fixed*) and duplication of the head noun(*dup.*), complex cases of synonyms(*syn.*), and cases require semantic understanding(*semantics*)

POS tags. We found that, for the single case of the best LSTM model without POS tags, the difference between that LSTM model and the best performing LSTM model was not statistically significant (p= 0.41). For all the remaining models, we found a highly significant difference ($p < 10^{-6}$) between the best performing LSTM model and each of those models.

7 Analysis

We perform some additional analysis to understand the behavior of the models.

7.1 Local context versus extended context

In order to gain an insight into the possible reasons behind the large performance gains obtained when using the extended context, we compare the best performing LSTM model that uses local context to the best model relying on the extended context and investigate 200 samples out of the 957 samples that were incorrectly predicted by the former (local context) but correctly predicted by the latter (extended context).

We grouped the samples into two main categories: (1) simple cases where the decision can be made based on the noun phrase itself (e.g. fixed expressions such as "the other day", and named entities), or the same head noun was introduced in the earlier discourse context; (2) complex cases where contextual knowledge involving pragmatic reasoning is required (e.g., entity coreference involving synonymy, bridging reference).

Table 5 shows the break-down into those categories. Complex cases accounted for over half of the cases with 108 cases involving synonymy or other complex cases such as bridging references. The model using local context failed to predict those cases correctly, which can be explained by the fact that, with local context only, no obvious cues for predictions were available. This suggests that LSTM networks constitute learning models that can learn to predict complex cases given an appropriate contextual window.

7.2 Qualitative analysis using the attention weights

In order to gain a better understanding of how the model makes its predictions and whether the model is able to resolve complex cases, we consider the attention weights of several samples (i.e., the set of a_i weights). All of the following cases were incorrectly predicted using local context but correctly predicted when the LSTM network was fed samples with extended context. Note although the original sentences (shown below) include the determiners (for the classes 'a', and 'the'), the determiners are removed from

the samples that are fed to the neural network model as mentioned in Section 4. The samples have a fixed length of 50 tokens. To emphasize the article that is the focus of the prediction task in that particular sample, we present such article in bold in each of the examples. Finally, note that since a sample consists of 50 tokens, the average weight for a token is 0.02.

Consider the example:

(7) *... net income for the third quarter of 16.8 million or 41 cents a share reflecting [a] broad-based improvement in the company's core businesses. Retail profit surged but the company it was only a modest contributor to third-quarter results. A year ago, net, at* ***the*** *New York investment banking firm ...*

In this example, in addition to the tokens in the noun phrase "New York investment banking firm" receiving some of the highest weights, both "contributor" and "company" were among the 10 tokens with the highest weight (with all of them receiving a weight of more than 0.04). While the LSTM with local context incorrectly predicted such a sample, high weights on tokens such as "contributor" and "company" suggest that the LSTM had potentially made use of the extended context paying higher "attention" to those relevant tokens while mostly ignoring the contents of the rest of the sentence.

In the following example:

(8) *...companies. In a possible prelude to the resumption of talks between Boeing Co. and striking Machinists union members, a federal mediator said representatives of the two sides will meet with him tomorrow. It could be a long meeting or it could be a short one, said Doug Hammond,* ***the*** *mediator...*

in addition to "mediator" receiving the highest weight (approx. 0.05), both "Doug" and "Hammond" received weights of approx. 0.049 and 0.05. While resolving the case of "the mediator", the network correctly paid attention to the relevant tokens "Doug" and "Hammond".

Consider this example:

(9) *BMA's investment banker Alex Brown & Sons Inc. has been authorized to contact possible buyers for the unit. Laidlaw Transportation Ltd. said it raised its stake in ADT Ltd. of Bermuda to 29.4% from 28%. A spokesman for Laidlaw declined to disclose the price* ***the*** *Toronto transportation...*

In this example, for correctly predicting that "Toronto transportation" should have a label "the", the network focused on the tokens "Toronto" and "transportation" while also attributing high weights to the tokens in "Laidlaw Transportation Ltd." (with all of them receiving a weight of more than 0.04 and figuring in the 10 highest weights).

These samples demonstrate that the attention mechanism can give us useful feedback on why the model is making the predictions that it is, adding interpretability to deep models. Secondly, they provide evidence that RNNs are able to learn complex features in order to place importance to syntactically and semantically relevant cues for this pragmatic reasoning task. By potentially allocating high weights to relevant tokens in an extended context, the LSTM network could also learn to predict complex cases of article usage, explaining its performance gains.

8 Conclusion

We have shown that English article usage, a task requiring reasoning about both local and non-local cues, can be successfully predicted using an LSTM recurrent neural network. Despite using generic features, our model outperforms previous methods for article prediction that rely on limited context. Our model is very successful in predicting the 'none' class and in making predictions for named entities. We perform a series of analyses to investigate whether the performance gains are due to factors that are traditionally cited for RNN models. By examining the attention weights, we find evidence that the models are actually learning complex features such as various syntactic and semantic restrictions, which would have been encoded by manually constructed features in previous models. As for initializing with pre-trained word embeddings, this improves performance consistently across multiple models but only marginally. The

LSTM can also take advantage of long-ranged dependencies in making its prediction, because it can place high attention on any part of the input sequence from an arbitrary distance before the head noun. We also have shown that LSTM networks show strong performance in resolving complex semantic cases when given an extended contextual window spanning two or more sentences.

Our model does not rely on task-specific features, only task-specific training. Thus, the exact same model should be applicable to other tasks involving semantic and pragmatic knowledge. We are currently planning to conduct further experiments on predicting other linguistic constructions involving contextual awareness and presupposition.

References

Lou Burnard. 2007. Reference guide for the british national corpus (XML edition).

Rachele De Felice. 2008. *Automatic error detection in non-native English*. Ph.D. thesis, University of Oxford.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China, July. Association for Computational Linguistics.

Michael Gamon, Jianfeng Gao, Chris Brockett, Alexandre Klementiev, William B. Dolan, Dmitriy Belenko, and Lucy Vanderwende. 2008. Using contextual speller techniques and language modeling for ESL error correction. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*, pages 449–456.

Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *arXiv preprint arXiv:1510.00726*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2006. Detecting errors in english article usage by non-native speakers. *Natural Language Engineering*, 12(2):115–129.

Julia E. Heine. 1998. Definiteness predictions for japanese noun phrases. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 519–525. Association for Computational Linguistics.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of the 2016 International Conference on Learning Representations*, January.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13–24, Baltimore, Maryland, June. Association for Computational Linguistics.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-Augmented distributed semantics for discourse relations. *Transactions of the Association for Computational Linguistics*, 3:329–344.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 119–126, Sofia, Bulgaria, August. Association for Computational Linguistics.

Kevin Knight and Ishwar Chander. 1994. Automated postediting of documents. In *AAAI*, volume 94, pages 779–784.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1188–1196.

John Lee, Joel Tetreault, and Martin Chodorow. 2009. Human evaluation of article and noun number usage: Influences of context and construction variability. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 60–63. Association for Computational Linguistics.

Jiwei Li, Rumeng Li, and Eduard H. Hovy. 2014. Recursive deep models for discourse parsing. In *EMNLP*, pages 2061–2069.

Jiwei Li, Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1106–1115, Beijing, China, July. Association for Computational Linguistics.

Mitchell P. Marcus, Mary A. Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH 2010*, volume 2, page 3.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Guido Minnen, Francis Bond, and Ann Copestake. 2000. Memory-based learning for article generation. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 43–48. Association for Computational Linguistics.

Yael D. Netzer and Michael Elhadad. 1998. Generating determiners and quantifiers in hebrew. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 89–96. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Massimo Poesio and Renata Vieira. 1998. A corpus-based investigation of definite description use. *Computational Linguistics*, 24(2):183–216.

Bertrand Russell. 1905. On denoting. *Mind*, pages 479–493.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Peter F. Strawson. 1950. On referring. *Mind*, 59(235):320–344.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Jenine Turner and Eugene Charniak. 2007. Language modeling for determiner selection. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 177–180. Association for Computational Linguistics.

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