MITRE at SemEval-2018 Task 11: Commonsense Reasoning without Commonsense Knowledge

Elizabeth M. Merkhofer, John Henderson, David Bloom,
Laura Strickhart and **Guido Zarrella**
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730-1420, USA
{emerkhofer,jhndrsn,dtbloom,lstrickhart,jzarrella}@mitre.org

Abstract

This paper describes MITRE's participation in SemEval-2018 Task 11: Machine Comprehension using Commonsense Knowledge. The techniques explored range from simple bag-of-ngrams classifiers to neural architectures with varied attention and alignment mechanisms. Logistic regression ties the systems together into an ensemble submitted for evaluation. The resulting system answers reading comprehension questions with 82.27% accuracy.

1 Introduction

Reading comprehension tasks measure the ability to answer questions that require inference from a free text story. This SemEval task, like many standardized tests (e.g., the SAT), provides multiple-choice answers. Reading comprehension may rely on information explicitly contained in the text, such as which actors are present, as well as elements of world knowledge, like understanding common scripts.

Early attempts at statistical reading comprehension include fill-in-the-blank questions and combine rich question categorization, information retrieval techniques, and entity recognition with type-specific, hand-crafted tactics (Hirschman et al., 1999; Anand et al., 2000).

More recent neural work uses continuous, distributed semantic space rather than n-gram overlap to find answers similar to the story. Trischler et al. (2016) compute a version of word overlap by finding cosine similarity of word representations between sections. Yin et al. (2016) restate the question and answer and incorporate a question-type classifier.

In this effort we explored neural distributed representations and lexicon-based machine learning approaches, especially attention and word overlap.

2 Task, Data and Evaluation

Machine Comprehension using Commonsense Knowledge was a shared task organized within SemEval-2018 (Ostermann et al., 2018b).

The task organizers released a dataset of 1,689 stories and 10,872 questions (up to 14 questions per story), split into training and development sets. The stories were first-person, English language narratives written by Mechanical Turk workers in response to prompts asking them to describe a scenario, like going on a date. Stories were up to 860 words long, with 90% under 273 words and a median length of 183 words. Questions are up to 22 words long, with a median length of seven. Each question has two possible answers. Answers vary in length from a single word, including *yes* or *no*, to 30 words, with a median length of three. Many questions are repeated between multiple stories. A sample story and questions are shown in Figure 1. Dataset construction is detailed in Ostermann et al. (2018a). The evaluation metric for the task is simple accuracy: the portion of correct answers.

Question types Table 1 quantifies the question types in the development set. We created this taxonomy to better understand the dataset. It also allowed us to direct our development toward model weaknesses.

Common Sense Knowledge and Inference Types Reading comprehension relies on inference. We explored an inference taxonomy outlined in Chikalanga (1992) to better characterize the types of questions and answers involved in this shared task. For a subset of the training dataset, we manually classified whether questions required a reader to make lexical, propositional, or pragmatic inferences about the story. We found that many questions could be answered with more than one type of inference and some questions required multiple inferences. We considered which infer-

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 1078–1082
New Orleans, Louisiana, June 5–6, 2018.

I went to the airport to go see my friend in North Carolina. So I drove up and had to go to the gate to go park. I paid my money to go park and went to park. From there I walked up into the airport to figure out where I need to check in. After much confusion, I found the place to check in and walked up. I gave them my boarding pass and they took my bags. From there they weighed them to make sure that they were not over the weight limit for carry on items. After that I went to security. I arrived super early so that I would have plenty of time in the line. I got in line for security and just waited. After getting to the front of the line, I took off my shoes and belt and put those on an x-ray scanner and got scanned myself. I put them back on and went to my flight.

Did they check in any baggage?

✓yes

X no

Where were they flying to?

✓North Carolina

X South Carolina

Why did they arrive to the airport early?

X They were not early, they were late.

✓To have enough time to wait in line

Why do they have to check in?

✓To board their flight

X They had to check in because they were going overseas.

Figure 1: Sample story with questions.

%	description
22.8	yes/no questions – One answer began with the string *yes* and the other answer began with the string *no*
9.4	yes/no only questions – answers were entirely the word *yes* or *no*
13.0	*who* questions
12.8	*what* questions
8.9	*where* questions
5.4	*when* questions
17.6	*how* questions
12.1	*why* questions
0.4	*which* questions
23.7	all correct answer words are in the story
30.7	none of the correct answer words are in the story
13.7	all incorrect answer words are in the story
37.6	none of the incorrect answer words are in the story
15.6	none of the correct or incorrect answer words are in the story

Table 1: Question types in the development set.

ences could require knowledge and data sources external to the story, like temporal-spatial relationships or physical properties such as temperature. Our analysis determined that this was unnecessary for task completion.

3 System Overview

We created an ensemble of three systems, each of which independently predicted the correct answer. Two of the systems use a neural attention architecture, and the third is a logistic regression.

3.1 Neural Attention

A recurrent neural system uses attention to create a representation for each answer that considers the question and story. Our ensemble includes two versions of this model with word embeddings trained on different corpora.

The words of each section (story, question, and answer) are embedded using a frozen projection layer, a fully-connected layer, and a recurrent layer. Another Long Short-Term Memory (LSTM) layer operates over the words in each story sentence to represent the story as a sequence of sentence embeddings. Each embedded answer sequence attends the words of the question using a parameterized attention mechanism (described below). This output attends the story sentences using a similar attention mechanism, and an LSTM reduces the sequence. A fully-connected layer produces a prediction for each answer output, and a softmax converts these to probabilities over the two answers. All hidden layers are of size 128, 50% dropout follows on the embedding and RNN layers, and the adam optimizer is used for training.

Components of this neural system include: pretrained distributed word representations, word overlap/hard attention features, and an attention mechanism.

For one model, **NN-T**, we used `word2vec` (Mikolov et al., 2013) to learn distributed representations of words from the text of English tweets collected from 2011 to 2016. We applied `word2phrase` twice to identify phrases of up to four words, and trained a skip-gram model of size 256, using a context window of 10 words and 15 negative samples per example. Twitter embeddings were chosen because at least some of the corpus matches the informal, first-person register of the task dataset. Our other set of word vectors, used in **NN-GN**, was released by Google alongside the tool and is made up of 300-vectors trained on a billion words of Google News (GoogleCode, 2013 (accessed March 3, 2018). For both vector sets, we used only the 100,000 most frequent vocabulary items, since the shared data vocabulary was quite limited.

Word-overlap features were concatenated to the pretrained word embeddings. These consist of four channels that compare the present section (question or answer) with the story and question. The first two channels are binary overlap: whether

	$\sigma\|w\|$	w	description
1	1.99	1.2814	$\|S \cap A\|$
2	1.18	-0.5591	answer length (words)
3	0.49	0.0394	answer length (chars)
4	0.15	-0.1728	$\|Q \cap A\|$
5	0.11	-0.0005	story length (chars)
6	0.09	0.0026	story length (words)
7	0.04	0.0041	question length (chars)
8	0.03	-0.0219	$\|S \cap Q\|$
9	0.02	0.0114	question length (words)

Table 2: Length and count features in the logistic regression model, ranked by influence ($\sigma|w|$).

the word in this position appears in the compared section. The cosine similarity channels contain the similarity of the present word with the closest word in the other section, computed using the same set of pretrained word vectors used by the neural model.

Our model adapts the attention mechanism described in Vaswani et al. (2017). The scaled dot product attention mechanism takes a memory and a query representation, e.g., the embedded words of the question and answer, respectively. Linear transformations are used to create a key and value from the former, and a query from the latter. Compatibility is computed as the scaled dot product of the key and query. This determines the weights over each timestep in the value, for each timestep of the query. Our model did not improve using parallel multi-head memory mechanisms nor by including fully-connected layers on top of them.

3.2 Logistic Regression

A logistic regression (LR) system was developed as a baseline against which the neural approach would be compared. This system was competitive enough to be included in the final ensemble.

The vocabulary of the LR system was limited to the training set. The Porter stemmer (Porter, 1980) was used to strip the non-information bearing suffixes from all of the words. Standard stopword lists removed words like *yes* and *no* that were important to the questions and answers, so the stopword list was reduced to just {*a*,*an*,*the*}. Various lengths were included as features. Taking S, Q, and A as the sets of words in the story, question, and answer respectively, the following three counts were added to the feature set: $|S \cap A|$, $|Q \cap A|$ and $|S \cap Q|$. Table 2 shows the full list of length and word count features.

The rest of the features were lexicalized patterns. Table 3 shows examples. Features include

	$\sigma\|w\|$	w	story	question	answer
1	0.61	1.85			ye
2	0.50	-1.89	it		it
3	0.45	-2.22	they		they
4	0.35	1.00			they
5	0.34	-1.64	wa		wa
6	0.33	-1.42	in		in
7	0.28	-2.25	at		at
8	0.27	-1.97			friend
9	0.26	1.98			narrat
10	0.24	-1.66	for		for
11	0.22	-1.60	of		of
12	0.22	2.03	friend		friend
13	0.21	-1.39	on		on
14	0.20	-1.90	with		with
15	0.20	0.82		did	no
16	0.19	-0.57			no
17	0.19	-1.78	were		were
18	0.19	-1.88	them		them
19	0.18	1.84			author
20	0.17	1.86		who	author
21	0.16	-1.75	one		one
22	0.16	-1.85	had		had
23	0.15	-1.09			hour
24	0.14	-1.82			neighbor
25	0.14	1.12		who	narrat
26	0.13	2.10		long	not
27	0.13	-1.73	out		out
28	0.13	1.20			home
29	0.13	1.50		mani	one
30	0.13	-1.22	minut		minut
31	0.13	-1.49	after		after
32	0.13	0.57		did	it
33	0.12	1.80	morn		morn
34	0.12	1.76		who	they
35	0.12	0.74		they	their
36	0.12	2.00			speaker

Table 3: Stemmed word factors in the logistic regression model ranked by influence ($\sigma|w|$).

the set of words in the answer (A), the words common to the story and the answer ($S \cap A$) and the Cartesian product of the question and answer ($Q \times A$).

A bias term was added and `Liblinear` (Fan et al., 2008) was used to compute the model. L2 regularization was used to encourage generalization. The best value for the regularization parameter was the default, 1. The target variable for the LR model to predict was the distinction between a correct answer and an incorrect answer. At decode time, the answer with the higher probability of being correct was chosen. This simple logistic regression model performed surprisingly well, less than 1% off from our best neural model.

Tables 2 and 3 show each feature's influence on fitting the training set. The rank ordering is standard deviation times the magnitude of the feature weight, $\sigma|w|$. This figure of merit balances features with high weights that were rare and features with low weights that were common.

3.3 Ensemble

An L2-regularized logistic regression weights the predictions of the above systems. MITRE's offi-

cial submission is an ensemble of the subsystems' binary class predictions. We submitted this system because it is simpler than an ensemble of continuous predictions and found it had the same accuracy on the development data, though more than 50 test set predictions were different.

4 Additional Experiments

We applied several approaches to the problem that did not generalize as well to the development data and were not included in the final ensemble.

Baselines We trained two baseline classifiers with incomplete information to gauge the difficulty of the task and to measure the relative importance of the story, question, and answers separately. Each baseline was a recurrent neural network with a layer of pretrained word embeddings and a stack of two 128-dimensional GRU layers. The *QA-only* baseline received as input only a concatenation of a question and its candidate answers, separated by special tokens. The *A-only* baseline received only a concatenation of the two candidate answers. *A-only* scored at 71.9% accuracy on the dev set, while *QA-only* was slightly higher at 73.2% accuracy. Other attempts to augment these models with attention over a lengthy story sequence frequently failed to eclipse the *QA-only* baseline, leading us to investigate hierarchical attention models and explicit overlap features.

Negative sampling We explored negative sampling to augment the training data, to improve our models' ability to exclude wrong answers. We selected the 10 nearest neighbors for each question and supplemented the original positive and negative answer with the other questions' answers. These were deduplicated after minimal preprocessing (normalizing case and punctuation), increasing the number of answers per question to between four and 20. Our nearest neighbors calculation is based on the average of word vectors for the in-vocabulary words in the questions.

Negative sampling did not improve accuracy. We tested conditions where 1) the original negative answer was sampled with equal probability or 2) always kept, and considered different values of N, where N is the total number of answers the model considered. We found no accuracy gain from using negative sampling beyond normal variance when the original negative was always included. When N was small, not necessarily including the original negative seems to hurt accuracy, suggesting that the randomly drawn negatives were not as plausible as the original negatives. For larger values of N, both conditions hurt performance.

We experimented with condition 1 and the LR model. The best value of N was 5 for this model, but accuracy was still below the model trained with the original dataset.

	Factored		Ablated	
Component	dev	test	dev	test
NN-T	81.93	80.23	81.72	79.76
NN-GN	81.01	80.12	83.06	79.51
LR	81.36	79.66	81.64	79.87
All In			**85.12**	**82.27**

Table 4: Factored and ablated system components evaluated on our dev set and the official test set.

5 Experiment Details

The systems included in our model were trained only on the data released for this task, aside from word vector pretraining. The dev set was used to select hyperparameters for individual components and final ensemble.

6 Results

The columns of Table 4 show the accuracy of each system in isolation on the dev and test data ("Factored") and the performance of the ensemble when the individual system was removed ("Ablated"). The final line shows the overall accuracy of the submission.

7 Conclusion

An ensemble of models was used to answer multiple-choice reading comprehension questions about informal, first person narratives. The resulting official system ranked second in the shared task. Our system relies heavily on lexical and overlap features, without an explicit reasoning component or external sources of world knowledge. Word embeddings trained on larger corpora contribute a semantic space that supports some inference beyond simple word overlap.

Acknowledgments

Approved for Public Release; Distribution Unlimited. Case Number 18-1298.

References

Pranav Anand, Eric Breck, Brianne Brown, Marc Light, Gideon Mann, Ellen Riloff, Mats Rooth, and Michael Thelen. 2000. Fun with reading comprehension. In *Final report, Reading Comprehension group, Johns Hopkins Center for Language and Speech Processing Summer Workshop*.

Israel Chikalanga. 1992. A suggested taxonomy of inferences for the reading teacher. *Reading in a Foreign Language*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

GoogleCode. 2013 (accessed March 3, 2018). Word2vec. https://code.google.com/archive/p/word2vec/.

Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. 1999. Deep read: a reading comprehension system. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 325–332, Stroudsburg, PA, USA. Association for Computational Linguistics.

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*.

Simon Ostermann, Ashutosh Modi, Michael Roth, Stefan Thater, and Manfred Pinkal. 2018a. MCScript: A Novel Dataset for Assessing Machine Comprehension Using Script Knowledge. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan.

Simon Ostermann, Michael Roth, Ashutosh Modi, Stefan Thater, and Manfred Pinkal. 2018b. SemEval-2018 Task 11: Machine Comprehension using Commonsense Knowledge. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA.

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

Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He, Phillip Bachman, and Kaheer Suleman. 2016. A parallel-hierarchical model for machine comprehension on sparse data. *arXiv preprint arXiv:1603.08884*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Wenpeng Yin, Sebastian Ebert, and Hinrich Schütze. 2016. Attention-based convolutional neural network for machine comprehension. *arXiv preprint arXiv:1602.04341*.