A Joint Model for Question Answering and Question Generation

Tong Wang [*1] **Xingdi Yuan** [*1] **Adam Trischler** [1]

Abstract

We propose a generative machine comprehension model that learns jointly to ask and answer questions based on documents. The proposed model uses a sequence-to-sequence framework that encodes the document and generates a question (answer) given an answer (question). Significant improvement in model performance is observed empirically on the SQuAD corpus, confirming our hypothesis that the model benefits from jointly learning to perform both tasks. We believe the joint model's novelty offers a new perspective on machine comprehension beyond architectural engineering, and serves as a first step towards autonomous information seeking.

1. Introduction

Question answering (QA) is the task of automatically producing an answer to a question given a corresponding document. It not only provides humans with efficient access to vast amounts of information, but also acts as an important proxy task to assess machine literacy via reading comprehension. Thanks to the recent release of several large-scale machine comprehension/QA datasets (Hermann et al., 2015; Rajpurkar et al., 2016; Dunn et al., 2017; Trischler et al., 2016; Nguyen et al., 2016), the field has undergone significant advancement, with an array of neural models rapidly approaching human parity on some of these benchmarks (Wang et al., 2017; Shen et al., 2016; Seo et al., 2016). However, previous models do not treat QA as a task of natural language generation (NLG), but of pointing to an answer span within a document.

Alongside QA, question generation has also gained increased popularity (Du et al., 2017; Yuan et al., 2017). The task is to generate a natural-language question conditioned on an answer and the corresponding document. Among its many applications, question generation has been used to improve QA systems (Buck et al., 2017; Serban et al., 2016; Yang et al., 2017). A recurring theme among previous studies is to augment existing labeled data with machine-generated questions; to our knowledge, the direct (though implicit) effect of asking questions on answering questions has not yet been explored.

In this work, we propose a joint model that both asks and answers questions, and investigate how this joint-training setup affects the individual tasks. We hypothesize that question generation can help models achieve better QA performance. This is motivated partly by observations made in psychology that devising questions while reading can increase scores on comprehension tests (Singer & Donlan, 1982). Our joint model also serves as a novel framework for improving QA performance outside of the network-architectural engineering that characterizes most previous studies.

Although the question answering and asking tasks appear symmetric, there are some key differences. First, answering the questions in most existing QA datasets is *extractive* — it requires selecting some span of text within the document — while question asking is comparatively *abstractive* — it requires generation of text that may not appear in the document. Furthermore, a (document, question) pair typically specifies a unique answer. Conversely, a typical (document, answer) pair may be associated with multiple questions, since a valid question can be formed from any information or relations which uniquely specify the given answer.

To tackle the joint task, we construct an attention-based (Bahdanau et al., 2014) sequence-to-sequence model (Sutskever et al., 2014) that takes a document as input and generates a question (answer) conditioned on an answer (question) as output. To address the mixed extractive/abstractive nature of the generative targets, we use the pointer-softmax mechanism (Gulcehre et al., 2016) that learns to switch between copying words from the document and generating words from a prescribed vocabulary. Joint training is realized by alternating the input data between question-answering and question-generating examples for the same model. We demonstrate empirically that this model's QA performance on *SQuAD*, while not state of the art, improves by about 10% with joint training. A key novelty of our joint model is that it can generate (partially) abstractive answers.

[*]Equal contribution [1]Microsoft Maluuba. Correspondence to: Tong Wang <tong.wang@microsoft.com>.

2. Related Work

Joint-learning on multiple related tasks has been explored previously (Collobert et al., 2011; Firat et al., 2016). In machine translation, for instance, Firat et al. (2016) demonstrated that translation quality clearly improves over models trained with a single language pair when the attention mechanism in a neural translation model is shared and jointly trained on multiple language pairs.

In question answering, Wang & Jiang (2016) proposed one of the first neural models for the SQuAD dataset. SQuAD defines an *extractive* QA task wherein answers consist of word spans in the corresponding document. Wang & Jiang (2016) demonstrated that learning to point to answer boundaries is more effective than learning to point sequentially to the tokens making up an answer span. Many later studies adopted this boundary model and achieved near-human performance on the task (Wang et al., 2017; Shen et al., 2016; Seo et al., 2016). However, the boundary-pointing mechanism is not suitable for more open-ended tasks, including abstractive QA (Nguyen et al., 2016) and question generation. While "forcing" the extractive boundary model onto abstractive datasets currently yields state-of-the-art results (Wang et al., 2017), this is mainly because current generative models are poor and NLG evaluation is unsolved.

Earlier work on question generation has resorted to either rule-based reordering methods (Heilman & Smith, 2010; Agarwal & Mannem, 2011; Ali et al., 2010) or slot-filling with question templates (Popowich & Winne, 2013; Chali & Golestanirad, 2016; Labutov et al., 2015). These techniques often involve pipelines of independent components that are difficult to tune for final performance measures. Partly to address this limitation, end-to-end-trainable neural models have recently been proposed for question generation in both vision (Mostafazadeh et al., 2016) and language. For example, Du et al. (2017) used a sequence-to-sequence model with an attention mechanism derived from the encoder states. Yuan et al. (2017) proposed a similar architecture but in addition improved model performance through policy gradient techniques.

Several neural models with a questioning component have been proposed for the purpose of improving QA models, an objective shared by this study. Yang et al. (2017) devised a semi-supervised training framework that trained a QA model (Dhingra et al., 2016) on both labeled data and artificial data generated by a *separate* generative component. Buck et al. (2017) used policy gradient with a QA reward to train a sequence-to-sequence paraphrase model to reformulate questions in an existing QA dataset (Dunn et al., 2017). The generated questions were then used to further train an existing QA model (Seo et al., 2016). A key distinction of our model is that we harness the *process* of asking questions to benefit question answering, without training the model to answer the generated questions.

3. Model Description

Our proposed model adopts a sequence-to-sequence framework (Sutskever et al., 2014) with an attention mechanism (Bahdanau et al., 2014) and a pointer-softmax decoder (Gulcehre et al., 2016). Specifically, the model takes a document (i.e., a word sequence) $D = (w_1^d, \ldots, w_{n_d}^d)$ and a condition sequence $C = (w_1^c, \ldots, w_{n_c}^c)$ as input, and outputs a target sequence $Y^{\{q,a\}} = (\hat{w}_1, \ldots, \hat{w}_{n_p})$. The condition corresponds to the question word sequence in answer-generation mode (a-gen), and the answer word sequence in question-generation mode (q-gen). We also attach a binary variable to indicate whether a data-point is intended for a-gen or q-gen. Intuitively, this should help the model learn the two modalities more easily. Empirically, QA performance improves slightly with this addition.

Encoder

A word w_i in an input sequence is first embedded with an embedding layer into vector $\mathbf{e}_i^w$. Character-level information is captured with the final states $\mathbf{e}_i^{ch}$ of a bidirectional Long Short-Term Memory model (Hochreiter & Schmidhuber, 1997) on the character sequences of w_i. The final representation for a word token $\mathbf{e}_i = \langle \mathbf{e}_i^w, \mathbf{e}_i^{ch} \rangle$ concatenates the word- and character-level embeddings. These are subsequently encoded with another BiLSTM into annotation vectors $\mathbf{h}_i^d$ and $\mathbf{h}_j^c$ (for the document and the condition sequence, respectively).

To better encode the condition, we also extract the encodings of the document words that appear in the condition sequence. This procedure is particularly helpful in q-gen mode, where the condition (answer) sequence is typically extractive. These extracted vectors are then fed into a condition aggregation BiLSTM to produce the *extractive condition encoding* $\mathbf{h}_k^e$. We specifically take the final states of the condition encodings $\mathbf{h}_J^c$ and $\mathbf{h}_K^e$. To account for the different extractive vs. abstractive nature of questions vs. answers, we use $\mathbf{h}_J^c$ in a-gen mode (for encoding questions) and $\mathbf{h}_K^e$ in q-gen mode (for encoding answers).

Decoder

The RNN-based decoder employs the pointer-softmax mechanism (Gulcehre et al., 2016). At each generation step, the decoder decides adaptively whether (a) to generate from a decoder vocabulary or (b) to point to a word in the source sequence (and copy over). Recurrence of the pointing decoder is implemented with two LSTM cells c_1

and c_2:

$$\mathbf{s}_1^{(t)} = c_1(\mathbf{y}^{(t-1)}, \mathbf{s}_2^{(t-1)}) \quad (1)$$

$$\mathbf{s}_2^{(t)} = c_2(\mathbf{v}^{(t)}, \mathbf{s}_1^{(t)}), \quad (2)$$

where $\mathbf{s}_1^{(t)}$ and $\mathbf{s}_2^{(t)}$ are the recurrent states, $\mathbf{y}^{(t-1)}$ is the embedding of decoder output from the previous time step, and $\mathbf{v}^{(t)}$ is the context vector (to be defined shortly in Equation (3)).

The pointing decoder computes a distribution $\boldsymbol{\alpha}^{(t)}$ over the document word positions (i.e., a document attention, Bahdanau et al. 2014). Each element is defined as:

$$\alpha_i^{(t)} = f(\mathbf{h}_i^d, \mathbf{h}^c, \mathbf{h}^e, \mathbf{s_1}^{(t-1)}),$$

where f is a two-layer MLP with *tanh* and *softmax* activation, respectively. The context vector $\mathbf{v}^{(t)}$ used in Equation (2) is the sum of the document encoding weighted by the document attention:

$$\mathbf{v}^{(t)} = \sum_{i=1}^{n} \alpha_i^{(t)} \mathbf{h}_i^d. \quad (3)$$

The generative decoder, on the other hand, defines a distribution over a prescribed decoder vocabulary with a two-layer MLP g:

$$\mathbf{o}^{(t)} = g(\mathbf{y}^{(t-1)}, \mathbf{s}_2^{(t)}, \mathbf{v}^{(t)}, \mathbf{h}^c, \mathbf{h}^e). \quad (4)$$

Finally, the switch scalar $s^{(t)}$ at each time step is computed by a three-layer MLP h:

$$s^{(t)} = h(\mathbf{s}_2^{(t)}, \mathbf{v}^{(t)}, \boldsymbol{\alpha}^{(t)}, \mathbf{o}^{(t)}),$$

The first two layers of h use *tanh* activation and the final layer uses *sigmoid* activation, and highway connections are present between the first and the second layer. We also attach the entropy of the softmax distributions to the input of the final layer, postulating that the quantities should help guide the switching mechanism by indicating the confidence of pointing vs generating. The addition is empirically observed to improve model performance.

The resulting switch is used to interpolate the pointing and the generative probabilities for predicting the next word:

$$p(\hat{w}_t) \sim s^{(t)} \boldsymbol{\alpha}^{(t)} + (1 - s^{(t)}) \mathbf{o}^{(t)}.$$

4. Training and Inference

The optimization objective for updating the model parameters θ is to maximize the negative log likelihood of the generated sequences with respect to the training data $\mathcal{D}$:

$$\mathcal{L} = -\sum_{x \in \mathcal{D}} \log p(\hat{w}_t | w_{<t}, x; \theta).$$

Here, $w_{<t}$ corresponds to the embeddings $\mathbf{y}^{(t-1)}$ in Equation (1) and (4). During training, gold targets are used to teacher-force the sequence generation for training, i.e., $w_{<t} = w_{<t}^{\{q,a\}}$, while during inference, generation is conditioned on the previously generated words, i.e., $w_{<t} = \hat{w}_{<t}$.

For words with multiple occurrence, since their exact references in the document cannot be reiabled determined, we aggregate the probability of these words in the encoder and the pointing decoder (similar to Kadlec et al. 2016). At test time, beam search is used to enhance fluency in the question-generation output.[1] The decoder also keeps an explicit history of previously generated words to avoid repetition in the output.

5. Experiments

5.1. Dataset

We conduct our experiments on the SQuAD corpus (Rajpurkar et al., 2016), a machine comprehension dataset consisting of over 100k crowd-sourced question-answer pairs on 536 Wikipedia articles. Simple preprocessing is performed, including lower-casing all texts in the dataset and using *NLTK* (Bird, 2006) for word tokenization. The test split of SQuAD is hidden from the public. We therefore take 5,158 question-answer pairs (self-contained in 23 Wikipedia articles) from the training set as validation set, and use the official development data to report test results. Note that answers in this dataset are strictly extractive, and we therefore constrain the pointer-softmax module to point at all decoding steps in answer generation mode.

5.2. Baseline Models

We first establish two baselines without multi-task training. Specifically, model `A-gen` is trained only to generate an answer given a document and a question, i.e., as a conventional QA model. Analogously, model `Q-gen` is trained only to generate questions from documents and answers. Joint-training (in model `JointQA`) is realized by feeding answer-generation and question-generation data to the model in an alternating fashion between mini-batches.

In addition, we compare answer-generation performance with the *sequence model* variant of the match-LSTM (`mLSTM`) model (Wang & Jiang, 2016). As mentioned earlier, in contrast to existing neural QA models that point to the start and end boundaries of extractive answers, this model predicts a sequence of document positions as the answer. This makes it most comparable to our QA setup. Note, however, that our model has the additional capacity

[1] The effectiveness of beam search can be undermined by the generally diminished output length. We therefore do not use beam search in a-gen mode, which also saves training time.

Table 1. Model evaluation on question- and answer-generation.

Models	*Answer Generation* F1	EM	*Question Generation* QA_{F1}	PPL	$BLEU_4$
A-gen	54.5	41.0		–	
Q-gen		–	72.4	260.7	10.8
JointQA	63.8	51.7	71.6	262.5	10.2
mLSTM	68.2	54.4		–	

to generate abstractively from the decoder vocabulary.

5.3. Quantitative Evaluation

We use *F1* and *Exact Match* (*EM*, Rajpurkar et al. 2016) against the gold answer sequences to evaluate answer generation, and *BLEU*[2] (Papineni et al., 2002) against the gold question sequences to evaluate question generation. However, existing studies have shown that the task of question generation often exhibits linguistic variance that is semantically admissible; this renders it inappropriate to judge a generated question solely by matching against a gold sequence (Yuan et al., 2017). We therefore opt to assess the quality of generated questions Y^q with two pretrained neural models as well: we use a language model to compute the perplexity of Y^q, and a QA model to answer Y^q. We measure the *F1* score of the answer produced by this QA model.

We choose mLSTM as the pretrained QA model and train it on SQuAD with the same split as mentioned in Section 5.1. Performance on the test set (i.e., the official validation set of SQuAD) is 73.78 *F1* and 62.7 *EM*. For the pretrained language model, we train a single-layer LSTM language model on the combination of the *text8* corpus[3], the *Quora Question Pairs* corpus[4], and the gold questions from SQuAD. The latter two corpora were included to tailor to our purpose of assessing *question* fluency, and for this reason, we ignore the semantic equivalence labels in the Quora dataset. Validation perplexity is 67.2 for the pretrained language model.

5.4. Analysis and Discussion

Evaluation results are provided in Table 1. We see that A-gen performance improves significantly with the joint model: both *F1* and *EM* increase by about 10 percentage points. Performance of q-gen worsens after joint training, but the decrease is relatively small. Furthermore, as pointed out by earlier studies, automatic metrics often do not correlate well with the generation quality assessed by humans (Yuan et al., 2017). We thus consider the overall outcome to be positive.

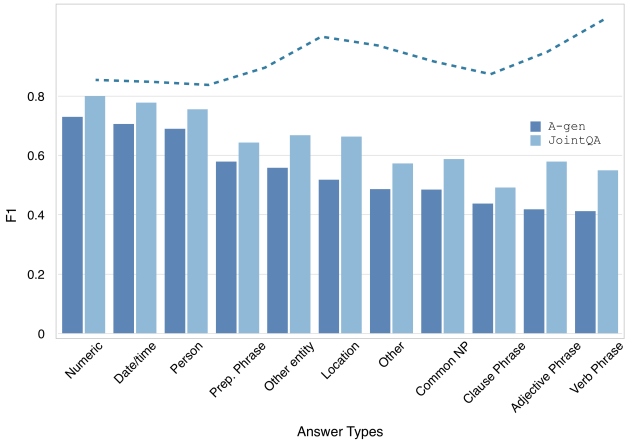

Figure 1. Comparison between A-gen and JointQA stratified by answer types. The dashed curve indicates period-2 moving average of the performance difference between the models.

Meanwhile, although our model does not perform as well as mLSTM on the QA task, it has the added capability of generating questions. mLSTM uses a more advanced encoder tailored to QA, while our model uses only a bidirectional LSTM for encoding. Our model uses a more advanced decoder based on the pointer-softmax that enables it to generate abstactively and extractively.

For a finer grained analysis, we first categorize test set answers based on their entity types, then stratify the QA performance comparison between A-gen and JointQA. The categorization relies on *Stanford CoreNLP* (Manning et al., 2014) to generate constituency parses, POS tags, and NER tags for answer spans (see Rajpurkar et al. 2016 for more details). As seen in Figure 1, the joint model significantly outperforms the single model in all categories. Interestingly, the moving average of the performance gap (dashed curve above bars) exhibits an upward trend as the A-gen model performance decreases across answer types, suggesting that the joint model helps most where the single model performance is weakest.

5.5. Qualitative Examples

Qualitatively, we have observed interesting "shifts" in attention before and after joint training. For example, in the positive case in Table 2, the gold question asks about the direct object,*Nixon*, of the verb *endorse*, but the A-gen model predicts the indirect object, *Kennedy*, instead. In contrast, the joint model asks about the appositive of *vice president* during question generation, which presumably

[2]We use the *Microsoft COCO Caption Evaluation* scripts (https://github.com/tylin/coco-caption) to calculate *BLEU* scores.

[3]http://mattmahoney.net/dc/textdata

[4]https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

Table 2. Examples of QA behaviour changes possibly induced by joint training. Gold answers correspond to text spans in green. In both the positive and the negative cases, the answers produced by the joint model are highly related (and thus presumably influenced) by the generated questions.

Positive	Document	*in the 1960 election to choose his successor , eisenhower endorsed his own vice president , republican richard nixon against democrat john f. kennedy .*
	Q_{gold}	*who did eisenhower endorse for president in 1960 ?*
	Q_{gen}	*what was the name of eisenhower 's own vice president ?*
	Answer	`A-gen:` *john f. kennedy* `JointQA:` *richard nixon*
Negative	Document	*in 1870 , tesla moved to karlovac , to attend school at the higher real gymnasium , where he was profoundly influenced by a math teacher martin sekulić*
	Q_{gold}	*why did tesla go to karlovac ?*
	Q_{gen}	*what did tesla do at the higher real gymnasium ?*
	Answer	`A-gen:` *to attend school at the higher real gymnasium*
		`JointQA:` *he was profoundly influenced by a math teacher martin sekulić*

"primes" the model attention towards the correct answer *Nixon*. Analogously in the negative example, QA attention in the joint model appears to be shifted by joint training towards an answer that is incorrect but closer to the generated question.

Note that the examples from Table 2 come from the validation set, and it is thus not possible for the joint model to memorize the gold answers from question-generation mode — the priming effect must come from some form of knowledge transfer between q-gen and a-gen via joint training.

5.6. Implementation Details

Implementation details of the proposed model are as follows. The encoder vocabulary indexes all words in the dataset. The decoder vocabulary uses the top 100 words sorted by their frequency in the gold questions in the training data. This encourages the model to generate frequent words (e.g. *wh*-words and function words) from the decoder vocabulary and copy less frequent ones (e.g., topical words and entities) from the document.

The word embedding matrix is initialized with the 300-dimensional *GloVe* vectors (Pennington et al., 2014). The dimensionality of the character representations is 32. The number of hidden units is 384 for both of the encoder/decoder RNN cells. Dropout is applied at a rate of 0.3 to all embedding layers as well as between the hidden states in the encoder/decoder RNNs across time steps.

We use *adam* (Kingma & Ba, 2014) as the step rule for optimization with mini-batch size 32. The initial learning rate is $2e-4$, which is decayed at a rate of 0.5 when the validation loss increases for two consecutive epochs.

The model is implemented using *Keras* (Chollet et al., 2015) with the *Theano* (Al-Rfou et al., 2016) backend.

6. Conclusion

We proposed a neural machine comprehension model that can jointly ask and answer questions given a document. We hypothesized that question answering can benefit from synergistic interaction between the two tasks through parameter sharing and joint training under this multitask setting. Our proposed model adopts an attention-based sequence-to-sequence architecture that learns to dynamically switch between copying words from the document and generating words from a vocabulary. Experiments with the model confirm our hypothesis: the joint model outperforms its QA-only counterpart by a significant margin on the SQuAD dataset.

Although evaluation scores are still lower than the state-of-the-art results achieved by dedicated QA models, the proposed model nonetheless demonstrates the effectiveness of joint training between QA and question generation, and thus offers a novel perspective and a promising direction for advancing the study of QA.

References

Agarwal, Manish and Mannem, Prashanth. Automatic gap-fill question generation from text books. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 56–64. Association for Computational Linguistics, 2011.

Al-Rfou, Rami, Alain, Guillaume, Almahairi, Amjad, Angermueller, Christof, Bahdanau, Dzmitry, Ballas, Nicolas, Bastien, Frédéric, Bayer, Justin, Belikov, Anatoly, Belopolsky, Alexander, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

Ali, Husam, Chali, Yllias, and Hasan, Sadid A. Automation of question generation from sentences. In *Proceed-*

ings of QG2010: The Third Workshop on Question Generation, pp. 58–67, 2010.

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL http://arxiv.org/abs/1409.0473.

Bird, Steven. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pp. 69–72. Association for Computational Linguistics, 2006.

Buck, Christian, Bulian, Jannis, Ciaramita, Massimiliano, Gesmundo, Andrea, Houlsby, Neil, Gajewski, Wojciech, and Wang, Wei. Ask the right questions: Active question reformulation with reinforcement learning. *arXiv preprint arXiv:1705.07830*, 2017.

Chali, Yllias and Golestanirad, Sina. Ranking automatically generated questions using common human queries. In *The 9th International Natural Language Generation conference*, pp. 217, 2016.

Chollet, François et al. Keras. https://github.com/fchollet/keras, 2015.

Collobert, Ronan, Weston, Jason, Bottou, Léon, Karlen, Michael, Kavukcuoglu, Koray, and Kuksa, Pavel. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

Dhingra, Bhuwan, Liu, Hanxiao, Cohen, William W, and Salakhutdinov, Ruslan. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*, 2016.

Du, Xinya, Shao, Junru, and Cardie, Claire. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*, 2017.

Dunn, Matthew, Sagun, Levent, Higgins, Mike, Guney, Ugur, Cirik, Volkan, and Cho, Kyunghyun. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*, 2017.

Firat, Orhan, Cho, Kyunghyun, and Bengio, Yoshua. Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv preprint arXiv:1601.01073*, 2016.

Gulcehre, Caglar, Ahn, Sungjin, Nallapati, Ramesh, Zhou, Bowen, and Bengio, Yoshua. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*, 2016.

Heilman, Michael and Smith, Noah A. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 609–617. Association for Computational Linguistics, 2010.

Hermann, Karl Moritz, Kocisky, Tomas, Grefenstette, Edward, Espeholt, Lasse, Kay, Will, Suleyman, Mustafa, and Blunsom, Phil. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pp. 1693–1701, 2015.

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

Kadlec, Rudolf, Schmid, Martin, Bajgar, Ondrej, and Kleindienst, Jan. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*, 2016.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Labutov, Igor, Basu, Sumit, and Vanderwende, Lucy. Deep questions without deep understanding. In *ACL (1)*, pp. 889–898, 2015.

Manning, Christopher D, Surdeanu, Mihai, Bauer, John, Finkel, Jenny Rose, Bethard, Steven, and McClosky, David. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pp. 55–60, 2014.

Mostafazadeh, Nasrin, Misra, Ishan, Devlin, Jacob, Mitchell, Margaret, He, Xiaodong, and Vanderwende, Lucy. Generating natural questions about an image. *arXiv preprint arXiv:1603.06059*, 2016.

Nguyen, Tri, Rosenberg, Mir, Song, Xia, Gao, Jianfeng, Tiwary, Saurabh, Majumder, Rangan, and Deng, Li. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Papineni, Kishore, Roukos, Salim, Ward, Todd, and Zhu, Wei-Jing. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics, 2002.

Pennington, Jeffrey, Socher, Richard, and Manning, Christopher D. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1532–1543, 2014.

Popowich, David Lindberg Fred and Winne, John Nesbit Phil. Generating natural language questions to support learning on-line. *ENLG 2013*, pp. 105, 2013.

Rajpurkar, Pranav, Zhang, Jian, Lopyrev, Konstantin, and Liang, Percy. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Seo, Minjoon, Kembhavi, Aniruddha, Farhadi, Ali, and Hajishirzi, Hannaneh. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

Serban, Iulian Vlad, García-Durán, Alberto, Gulcehre, Caglar, Ahn, Sungjin, Chandar, Sarath, Courville, Aaron, and Bengio, Yoshua. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *arXiv preprint arXiv:1603.06807*, 2016.

Shen, Yelong, Huang, Po-Sen, Gao, Jianfeng, and Chen, Weizhu. Reasonet: Learning to stop reading in machine comprehension. *arXiv preprint arXiv:1609.05284*, 2016.

Singer, Harry and Donlan, Dan. Active comprehension: Problem-solving schema with question generation for comprehension of complex short stories. *Reading Research Quarterly*, pp. 166–186, 1982.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pp. 3104–3112, 2014.

Trischler, Adam, Wang, Tong, Yuan, Xingdi, Harris, Justin, Sordoni, Alessandro, Bachman, Philip, and Suleman, Kaheer. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*, 2016.

Wang, Shuohang and Jiang, Jing. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*, 2016.

Wang, Wenhui, Yang, Nan, Wei, Furu, Chang, Baobao, and Zhou, Ming. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

Yang, Zhilin, Hu, Junjie, Salakhutdinov, Ruslan, and Cohen, William W. Semi-supervised qa with generative domain-adaptive nets. *arXiv preprint arXiv:1702.02206*, 2017.

Yuan, Xingdi, Wang, Tong, Gulcehre, Caglar, Sordoni, Alessandro, Bachman, Philip, Subramanian, Sandeep, Zhang, Saizheng, and Trischler, Adam. Machine comprehension by text-to-text neural question generation. *arXiv preprint arXiv:1705.02012*, 2017.