MOCHA: A Dataset for Training and Evaluating Generative Reading Comprehension Metrics

Anthony Chen
UC Irvine
anthony.chen@uci.edu

Gabriel Stanovsky*
The Hebrew University
gabis@cse.huji.ac.il

Sameer Singh
UC Irvine
sameer@uci.edu

Matt Gardner
AI2 Irvine
mattg@allenai.org

Abstract

Posing reading comprehension as a generation problem provides a great deal of flexibility, allowing for open-ended questions with few restrictions on possible answers. However, progress is impeded by existing generation metrics, which rely on token overlap and are agnostic to the nuances of reading comprehension. To address this, we introduce a benchmark for training and evaluating generative reading comprehension metrics: ***MO**deling **C**orrectness with **H**uman **A**nnotations*. MOCHA contains 40K human judgement scores on model outputs from 6 diverse question answering datasets and an additional set of minimal pairs for evaluation. Using MOCHA, we train a ***L**earned **E**valuation metric for **R**eading **C**omprehension*, LERC, to mimic human judgement scores. LERC outperforms baseline metrics by 10 to 36 absolute Pearson points on held-out annotations. When we evaluate robustness on minimal pairs, LERC achieves 80% accuracy, outperforming baselines by 14 to 26 absolute percentage points while leaving significant room for improvement. MOCHA presents a challenging problem for developing accurate and robust generative reading comprehension metrics.[1]

1 Introduction

Reading comprehension (RC) has seen significant progress in the last few years, with a number of question answering (QA) datasets being created (Rajpurkar et al., 2016; Lai et al., 2017; Talmor et al., 2018). However, a majority of datasets are presented using a span-selection or multiple-choice (MC) format. Both formats are easy to evaluate, but in return, have restrictions placed on the questions that can be asked or the answers that can be returned. Furthermore, both formats hinge on distractor spans/choices for learning to be effective. Ensuring high quality distractors is a challenging task in and of itself, which can lead to models that exploit spurious correlations (Jia and Liang, 2017; Min et al., 2019; Geva et al., 2019). Posing RC as a generation task addresses the aforementioned issues. Generative RC does not require distractors, circumventing biases that could be introduced by them, and allows arbitrary questions and answers.

Passage: ...Behind one door is a lady whom the king has deemed an appropriate match for the accused; behind the other is a fierce, hungry tiger. Both doors are **heavily soundproofed to prevent the accused from hearing what is behind each one**...

Question: What feature do the doors have?
Reference: soundproofed
Candidate: They are **heavily soundproofed to prevent the accused from hearing what's behind each one**.

Human Judgement: 5 out of 5
LERC: 4.98 out of 5

BLEU-1: 0.07
ROUGE-L: 0.15
METEOR: 0.17

Figure 1: Generative reading comprehension example. Properly scoring the candidate requires access to the passage. Current metrics, such as BLEU, ROUGE and METEOR, are agnostic to the end-task while LERC is trained with the passage and question as input. As a result, LERC assigns a score that better reflects human judgement.

Unfortunately, existing metrics for evaluating text generation come with significant shortcomings. Many metrics score n-gram overlap, and it is well established that using token overlap as a measure of similarity has drawbacks (Chen et al., 2019; Edunov et al., 2019; Wang et al., 2020). Cur-

*Work done while at the Allen Institute for AI and the University of Washington.

[1]The dataset, code, a leaderboard, and a demo are available at https://allennlp.org/mocha.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 6521–6532,
November 16–20, 2020.

rent metrics also only consider the reference and are agnostic to the end-task being evaluated. Fig. 1 demonstrates that this is problematic for generative RC because scoring a candidate may require a metric to also consider the passage and the question. Without cheap and reliable evaluation, progress in generative reading comprehension has been extremely slow.

To address the need for better evaluation metrics tailored to reading comprehension, we present a dataset called MOCHA, aimed at developing *learned* metrics that **MO**del the **C**orrectness of candidates using **H**uman **A**nnotation scores. MOCHA contains human judgement scores on 40K candidates, an order of magnitude larger than prior work (Chen et al., 2019). The candidates come from six diverse QA datasets which test a wide range of RC phenomena such as commonsense reasoning and understanding narrative over movie scripts. After collecting all annotations, we follow work on creating more robust evaluation sets (Kaushik et al., 2020; Gardner et al., 2020) and augment the test set of MOCHA by manually writing a small set of minimal pairs (Table 3). The set of minimal pairs serve as a harder evaluation set for probing metric robustness.

Using MOCHA, we train a **L**earned **M**etric for **R**eading **C**omprehension which we abbreviate as LERC. We compare LERC against two sets of baselines: (1) existing metrics such as METEOR (Banerjee and Lavie, 2005) and BERTScore (Zhang et al., 2019); and (2) a sentence similarity model trained on STS-B (Cer et al., 2017). To ensure fair comparison, we evaluate LERC in an out-of-dataset setting: LERC is trained on all datasets *except* the one it is being evaluated on. On the test set, LERC outperforms baselines by as much as 36 Pearson correlation points and on the minimal pairs set, by as much as 26 accuracy points. Error analysis and minimal pair results indicate that there is substantial room to improve the robustness of LERC and its sensitivity to different linguistic phenomena. We hope that MOCHA and LERC enables a continual cycle of generative RC model and dataset developments that will enable easier collection of more diverse and useful candidates, allowing better learned metrics to be trained.

Instance	Score
Passage: With the aid of his daughter, Abigail, Barabas recovers his former assets. Barabas then uses his daughter's beauty to embitter Lodowick and Mathias against each other. **Q:** Why did Lodowick and Mathias fight? **Ref:** Over the affection of Abigail **Cand:** They fight over Barabas's daughter.	5
Passage: Miss Moppet ties a duster about her head and sits before the fire. The mouse thinks she looks very ill and comes down the bell-pull. **Q:** What does the mouse think when she sees the duster on Miss Moppet's head? **Ref:** that Miss Moppet is ill **Cand:** Miss Moppet thinks it is ill and is trying to sniff him.	2
Passage: Robin took a very long time to clean the windows of her house. **Q:** How would you describe Robin? **Ref:** a neat freak **Cand:** a clean person	5
Passage: The strangest thing that has happened was when they were singing the Chinese National Anthem she was standing in front of the TV swaying and singing. **Q:** What is probably true about this story? **Ref:** They are watching the Olympics **Cand:** The Olympics are watching	2

Table 1: Example instances with human judgement scores from MOCHA highlighting the diverse phenomenon that an evaluation metric needs to handle. These phenomenon include resolving coreference, dealing with factual correctness, understanding paraphrases, and understanding semantic roles.

2 A Description of MOCHA

Reading comprehension is the task of probing how well systems can understand passages of text. Framing reading comprehension as a generation problem provides a great deal of flexibility, but introduces the challenging problem of evaluation. These challenges are further amplified when applied to generative reading comprehension, where the introduction of a passage and a question can add to the complexity of evaluation (Table 1). To handle this challenge, we propose to *train* a generative reading comprehension metric. This first requires a large set of human judgement scores to be gathered.

In this section, we present MOCHA, a dataset that pairs reading comprehension instances, which consists of a passage, question, and reference, with candidates and human judgement scores. We describe the process of gathering candidates, collect-

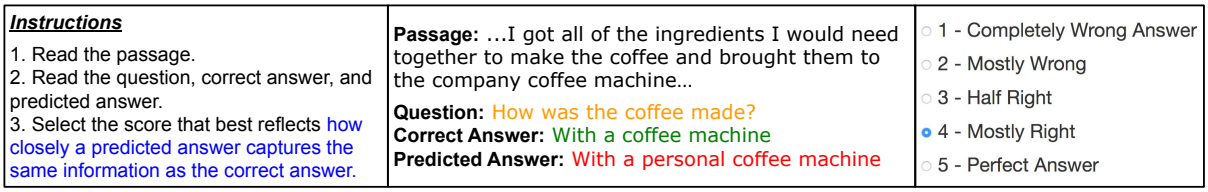

Figure 2: A compressed version of the Mechanical Turk interface for evaluating answer correctness. Workers were asked to score (1 to 5) how similar a candidate is to a reference **using** the passage and the question.

ing human judgement scores, and creating minimal pairs for evaluation.

2.1 Datasets

Candidates in MOCHA come from 6 constituent QA datasets that are diverse in their domains and answer types. This ensures that training and evaluation with MOCHA does not overfit to the characteristics of any constituent dataset.

NarrativeQA (Kociský et al., 2017) tests reasoning about events, entities, and their relations on movie scripts and book summaries.

MCScript (Ostermann et al., 2018) tests reasoning on stories written for a child-level reader.

CosmosQA (Huang et al., 2019) tests commonsense reasoning on blogs describing everyday events.

SocialIQA (Sap et al., 2019) tests social reasoning with passages constructed from a knowledge base.

DROP (Dua et al., 2019) tests predicate argument structure and numerical reasoning on Wikipedia articles concerning American football games, census results, and history.

Quoref (Dasigi et al., 2019) tests coreferential reasoning on Wikipedia articles.

NarrativeQA was created as a generative RC dataset. CosmosQA, MCScript, and SocialIQA were created as MC datasets which we re-purpose as generative datasets by using the correct choice as the reference. Our motivation for doing this is that the number of generative QA datasets is quite small, which we attribute to the quality of evaluation metrics.

The main focus of this work is in developing and evaluating metrics for generative RC. However, we wanted to see whether a learned metric could do well on span-selection datasets. We collected candidates on two span-based datasets, DROP and Quoref, to test this.

2.2 Collecting Candidates

Candidates on all four generative datasets are generated using backtranslation (Sennrich et al., 2016) and using a fine-tuned GPT-2 model (Radford et al., 2019). We also generate candidates for NarrativeQA and MCScript using a trained MHPG model (Bauer et al., 2018). We tried using MHPG for CosmosQA and SocialIQA but candidates were of poor quality. Unique to NarrativeQA, each question has two references. We treat the second reference as a candidate to be annotated if it has low n-gram overlap with the first reference. We use a span-selection BERT-based model to generate candidates for Quoref and NAQANET (Dua et al., 2019) and NABERT[2] models for DROP.

Models are trained on the training sets of each constituent dataset and candidates are produced on instances from the validation set (and test set if available). We filtered out candidates that exactly matched the reference. We also filtered out instances in DROP where the reference and the candidate are both numbers.[3]

In total, MOCHA contains 40K candidates, large enough for training a learned metric as well as for evaluating current and future metrics.

2.3 Annotation Procedure

Annotations are collected with Mechanical Turk using the interface in Fig. 2. Workers are asked to score candidate answers on an ordinal scale from 1 to 5. We start by collecting a single annotation per candidate. Following this, candidates are split into training, validation, and test sets such that all candidates from a passage are contained within a dataset split. For instances in our validation and

[2]https://github.com/raylin1000/drop-bert

[3]From our inspection, if the reference and candidate are both numbers that are not equal, the candidate is always wrong.

Dataset	Avg Pass. Len	Avg Ques. Len	Avg Ref. Len	Avg Cand. Len	# Passages			# Ques./Ref. Pairs			# Candidates		
					Train	Dev	Test	Train	Dev	Test	Train	Dev	Test
NarrativeQA	333.0	9.6	5.8	5.9	85	11	18	2249	277	500	7471	890	1707
MCScript	197.1	7.8	4.3	4.1	462	61	93	2940	390	583	7210	978	1409
CosmosQA	72.8	10.8	7.5	8.8	1064	142	212	1139	156	226	5033	683	1017
SocialIQA	15.7	7.2	3.9	3.9	3075	414	611	3075	414	611	7409	1017	1527
DROP	213.4	11.6	3.6	5.1	80	10	17	542	76	117	687	97	152
Quoref	324.0	15.8	2.3	8.2	184	24	38	1098	123	180	3259	344	509
Total					4950	662	989	11043	1436	2217	31069	4009	6321

Table 2: Statistics for the human judgements per constituent dataset in MOCHA.

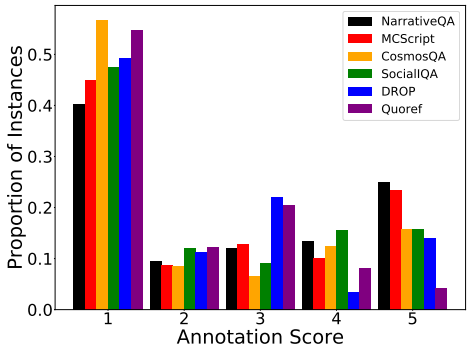

Figure 3: Human judgement score distribution on the training set of MOCHA, divided into the 6 constituent datasets. The distribution of scores is right-skewed because we did not annotate candidates that exactly matched a reference.

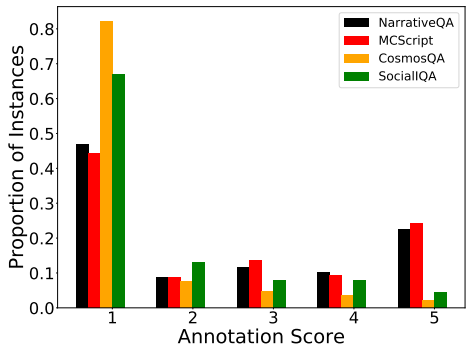

Figure 4: Score distribution on candidates from GPT-2. GPT-2 produces a *very* skewed score distribution for CosmosQA and SocialIQA, highlighting the difficulty of generative RC on commonsense questions.

test sets, we collect one additional human judgement score per candidate for span-based datasets, and two additional human judgement scores per candidate for generative datasets. Multiple annotations for a given candidate are averaged to form a gold annotation. More details such as payout and qualification testing are provided in Appendix D.

We calculated inter-annotator agreement using Krippendorff's Alpha-Reliability (Krippendorff, 2011) on the validation set of all 6 constituent datasets. We choose this metric because it applies to our setting, where there are multiple annotators per instance, and the annotators vary between instances. Agreement on our 6 datasets range from 0.71 to 0.92 (average = 0.82), indicating strong agreement.

2.4 Statistics for MOCHA

Statistics of instances and dataset splits in MOCHA are provided in Table 2. The number of unique passages varies considerably across datasets. NarrativeQA, which has the longest passages, has few unique passages, while SocialIQA has a unique passage for each question/reference pair. The number of candidates also varies across datasets. The most pronounced outlier is DROP, where we collected a tenth of the candidates compared to the other datasets. This is because we filtered out instances when both the candidate and reference were numbers, leaving much fewer candidates to annotate. The number of candidates outnumbers the question/reference pairs because for each pair, we generated multiple candidates using different generation sources (e.g. backtranslation, different model outputs).

Fig. 3 provides the annotation score distribution on the training set of MOCHA. Score distributions are right-skewed because we did not collect annotations when the reference exactly matched the candidate. The right-skew is most pronounced for Quoref because the number of ways a candidate can get a perfect score while not matching the reference is limited in a span extraction format.

2.5 Limitations and Robust Evaluation with Minimal Pairs

Candidates in MOCHA come from existing models, so that a metric learned on this data will be most applicable to current research. However, as

Phenomenon	Original Instance	Minimal Pairs
Coreference	**Passage:** Norman is the supposed son of Frenchman de Vac . . . As de Vac dies, he reveals Norman is Richard, the king's son and Edward's brother, who he kidnapped. **Q:** Who is the Frenchman de Vac? **Ref:** a fencing master who kidnapped Norman	**Cand. 1:** a fencing master who kidnapped Richard **(5)** **Cand. 2:** a fencing master who kidnapped Edward **(3)**
Hyponymy	**Passage:** With the electric rifle, Tom and friends bring down elephants, rhinoceroses, and buffalo. **Q:** What does Tom bring down with his rifle? **Ref:** Rhinoceroses, buffalo, and elephants.	**Cand. 1:** Animals **(4)** **Cand. 2:** Humans **(1)**
Negation	**Passage:** skylar told quinn's friend about a secret that quinn wanted to keep hidden. **Q:** What will Quinn want to do next? **Ref:** be angry	**Cand. 1:** Quinn will be mad at Skylar **(5)** **Cand. 2:** Quinn will not be mad at Skylar **(1)**
Semantic Role	**Passage:** Taylor gave a raise and promotion to Kendall. **Q:** How would you describe Taylor? **Ref:** As someone who appreciates what Kendall does	**Cand. 1:** Taylor appreciates Kendall **(5)** **Cand. 2:** Kendall appreciates Taylor **(1)**
Syntax	**Passage:** Taylor looked around in Robin's cupboards and peeked inside Robin's drawers and medicine cabinet. **Q:** How would you describe Taylor? **Ref:** intrusive	**Cand. 1:** I would describe Taylor as intrusive **(5)** **Cand. 2:** Would I describe Taylor as intrusive **(3)**
Word Sense	**Passage:** Taylor got married but kept her last name. **Q:** How would you describe Taylor? **Ref:** independent	**Cand. 1:** individualistic **(5)** **Cand. 2:** nonpartisan **(1)**
Other	**Passage:** The Princess stuffs her ears with cotton and begins her journey. **Q:** What does the Princess put in her ears? **Ref:** She puts cotton in her ears.	**Cand. 1:** Her ears have cotton **(4)** **Cand. 2:** Her ears are cotton **(2)**

Table 3: Minimal pairs categorized by the linguistic phenomena. Given a passage, question, and reference, we create two new candidates, $\mathbf{c_1}$ and $\mathbf{c_2}$, with associated human judgement scores $\mathbf{s_1}$ and $\mathbf{s_2}$. In total, we wrote 200 minimal pairs (50 for each generative QA dataset).

research in generative reading comprehension models is presently limited, the strength of these models can be low. Fig. 4 shows that generative QA models struggle to produce quality answers when asked about commonsense scenarios. The majority of 5's in CosmosQA and SocialIQA are produced via backtranslation, while GPT-2 struggles to produce "correct" candidates. This raises an issue with the evaluation; a metric can look strong when evaluated on current model outputs, but may in-fact struggle in the future when QA systems produce better answers. Thus, using only these candidates for evaluation could lead to overconfidence in a learned metric's capabilities.

We take inspiration from from recent work creating more robust evaluations (Kaushik et al., 2020; Gardner et al., 2020) and augment the test set of MOCHA with a small number of minimal pairs created by the authors. Given a passage, question, and reference from the test set, we manually create two new candidates, $\mathbf{c_1}$ and $\mathbf{c_2}$, which form a minimal pair. Accompanying $\mathbf{c_1}$ and $\mathbf{c_2}$ are human judgement scores, $\mathbf{s_1}$ and $\mathbf{s_2}$, collected using the same interface in Fig. 2. The minimal pair is created so that $\mathbf{c_1}$ has a higher score (i.e. is a better answer) than $\mathbf{c_2}$. Each minimal pair is designed to capture a particular linguistic phenomenon (see Table 3). Using this set of minimal pairs, we can study how often a metric prefers the better candidate. We create 200 minimal pairs (50 for each generative QA dataset), which we use for evaluation *separately* from the original test set.

3 A Learned Metric

We provide details on LERC, our learned metric. LERC is initialized using BERT-base (Devlin et al., 2019) We define as input a tuple consisting of a passage, $\mathbf{p}$, a question, $\mathbf{q}$, a reference answer, $\mathbf{a}$, and a candidate answer, $\hat{\mathbf{a}}$. The input to BERT is

Metric	NarrativeQA		MCScript		CosmosQA		SocialIQA		DROP		Quoref		Avg. *r*	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
BLEU-1	0.403	0.472	0.181	0.260	0.660	0.670	0.595	0.549	0.409	0.387	0.674	0.578	0.487	0.486
METEOR	0.605	0.615	0.461	0.502	0.696	0.711	0.644	0.637	0.664	0.568	**0.729**	0.716	0.633	0.624
ROUGE-L	0.434	0.495	0.224	0.297	0.701	0.701	0.599	0.558	0.480	0.366	0.712	0.604	0.525	0.503
BERTScore	0.419	0.534	0.172	0.194	0.803	0.779	0.604	0.584	0.174	0.328	0.207	0.286	0.396	0.450
BERT STS-B	0.711	0.686	0.364	0.449	0.803	0.789	0.663	0.666	0.690	**0.715**	0.690	**0.750**	0.653	0.676
LERC	**0.772**	**0.738**	**0.666**	**0.694**	**0.852**	**0.824**	**0.777**	**0.799**	**0.760**	0.712	0.704	0.741	**0.755**	**0.751**

Table 4: Pearson correlation to human judgement scores on the validation and test sets of MOCHA. LERC results are from a model trained in an out-of-dataset fashion, averaged across three runs.

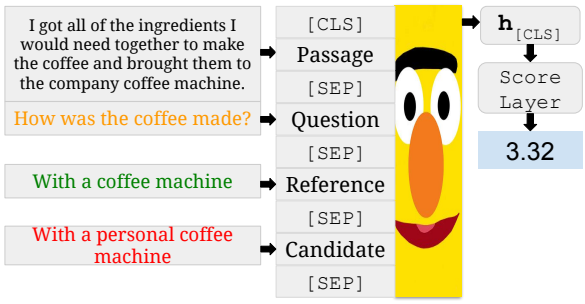

Figure 5: LERC is a BERT model that has been fine-tuned on human judgment scores. LERC takes as input a passage, question, reference, and candidate, and returns a score rating the "correctness" of the candidate.

structured as:

$$[\text{CLS}]\ \mathbf{p}\ [\text{SEP}]\ \mathbf{q}\ [\text{SEP}]\ \mathbf{a}\ [\text{SEP}]\ \hat{\mathbf{a}}\ [\text{SEP}]$$

BERT returns a hidden state for each input token. We use the first hidden state $\mathbf{h}_{[CLS]}$, as the pooled representation of the input.

3.1 Fine-Tuning with Human Judgements

Our goal is to train BERT to mimic the human judgements given a set of input tuples, $\{(\mathbf{p}, \mathbf{q}, \mathbf{a}, \hat{\mathbf{a}})\}_{i=1}^{n}$, and a set of human judgment scores, $\{y\}_{i=1}^{n}$, We apply a regression layer on top of our pooled representation (Fig. 5) and train with a MSE loss.

$$\hat{y}_i = \mathbf{W}\, \mathbf{h}_{i\,[CLS]}$$
$$\text{loss}_i = (y_i - \hat{y}_i)^2$$

3.2 Pre-Training the Learned Metric

Learning the interactions between the input components can be difficult with only human judgement fine-tuning. To overcome this, we *pre-train* on four multiple-choice QA datasets: BoolQ (Clark et al., 2019a), MCTest (Richardson et al., 2013), RACE (Lai et al., 2017), and MultiRC (Khashabi et al., 2018). We use the same input structure as fine-tuning, but the reference and candidate are replaced by two answer choices, $\mathbf{a_1}$ and $\mathbf{a_2}$:

$$[\text{CLS}]\ \mathbf{p}\ [\text{SEP}]\ \mathbf{q}\ [\text{SEP}]\ \mathbf{a_1}\ [\text{SEP}]\ \mathbf{a_2}\ [\text{SEP}]$$

We pre-train BERT via 3-way classification to predict whether: $\mathbf{a_1}$ is the correct answer, $\mathbf{a_2}$ is the correct answer, or $\mathbf{a_1}$ and $\mathbf{a_2}$ are both correct. MultiRC has multiple correct answers per question and we create additional instances where both $\mathbf{a_1}$ and $\mathbf{a_2}$ are correct by duplicating the correct answer for all three datasets.

4 Experiments

Training LERC: We use the PyTorch (Paszke et al., 2019), HuggingFace Transformers (Wolf et al., 2019), and AllenNLP (Gardner et al., 2017) libraries to implement LERC. We pre-train LERC before fine-tuning on MOCHA. We evaluate LERC in two settings, an out-of-dataset (OOD) setting and an all-datasets (AD) setting. In the OOD setting, we train and tune LERC on all datasets in MOCHA *except* the dataset we are evaluating on. This reflects the use case where we want to apply LERC to evaluate a new dataset where we do not have human judgement scores. In the AD setting, we train on all datasets in MOCHA and evaluate on all datasets. All results reported for LERC are the average of three runs using the best set of hyperparameters found on the validation set of MOCHA.

Baselines: We compare LERC against BLEU-1 (Papineni et al., 2001), ROUGE-L (Lin, 2004), METEOR (Banerjee and Lavie, 2005), and BERTScore (Zhang et al., 2019). We also compare LERC against a BERT-base model fine-tuned on the sentence similarity task, STS-B (Cer et al., 2017). Results for BERT STS-B are the average of three runs using the best set of hyperparameters found on the

Dataset	Dev *r*
NarrativeQA	0.805
MCScript	0.816
CosmosQA	0.864
SocialIQA	0.820
DROP	0.796
Quoref	0.794

Table 5: Pearson correlation on the validation set of MOCHA with LERC trained on all constituent datasets.

Ablation	Avg. Dev *r*
Ref. Only	0.081
Cand. Only	0.093
Ref. & Cand.	0.742
Ques. & Ref. & Cand.	0.723
Pass. & Ques. & Ref. & Cand.	0.726
LERC (with pre-training)	0.755

Table 6: Partial-input ablations of LERC trained in an out-of-dataset fashion. Results are Pearson correlation on the validation set, averaged across all constituent datasets.

validation set of STS-B. All baselines are agnostic to the passage and the question.

4.1 Correlation Results

We evaluate the baselines and OOD LERC in Table 4 using Pearson correlation. LERC outperforms the baseline metrics despite being trained in a out-of-dataset situation. METEOR does surprisingly well despite relying on *n*-gram overlap to do evaluation. Interestingly, the sentence similarity model does better than the baseline metrics while falling behind LERC.

We also study whether having human judgements for a particular dataset helps. We present results in Table 5 on the validation set of MOCHA when LERC is trained in an AD setting. Having human judgements for the target dataset is always helpful.

4.2 Error Analysis of LERC

We gather the 10 validation instances per generative dataset (40 instances total) with the highest absolute difference between the human judgement score and LERC score. We categorize the errors made by LERC in Table 7. A large source of error is the inability to leverage the passage correctly as well as handling large lexical gaps between references and correctly paraphrased candidates. The "Other" category includes understanding semantic roles and misspellings of the reference.

Error Source	Example
Passage Use *(22.5%)*	**Passage:** Edward takes charge and the children develop and expand the farmstead, aided by the entrepreneurial spirit of the younger brother Humphrey. They are assisted by a gypsy boy, Pablo, who they rescue from a pitfall trap. **Q:** Who do the children rescue from a trap? **Ref:** Pablo **Cand:** A gypsy kid **Human Score:** 4.6 **LERC:** 1.0
Same Meaning *(35%)*	**Passage:** The story centres on the relationship between Mrs Kitty Warren and her daughter, Vivie. Mrs. Warren, a former prostitute. **Q:** What did Mrs. Warren previously do for work? **Ref:** Prostitution **Cand:** She was an escort. **Human Score:** 4.6 **LERC:** 1.06
Opposite Meaning *(15%)*	**Passage:** Sasha hated her neighbours dog as it barked all day and night so after going to the shop and buying poisonous slug pellets, Sasha gave the dog some pills. **Q:** How would you describe Sasha? **Ref:** mean **Cand:** kind **Human Score:** 1 **LERC:** 4.32
Other *(27.5%)*	**Passage:** The train was slow and ambling, so much so that we were 2 hours late when we arrived in Montreal, missing our connection. **Q:** What might be true if the freight trains didn't cause a delay ? **Ref:** They wouldn't have missed their connection **Cand:** they couldn't help noticing their connection **Human Score:** 1 **LERC:** 4.2

Table 7: Error analysis of LERC. We take the 10 validation instances per *generative* dataset (40 total) with the largest difference between the score assigned by LERC and the score assigned by humans. We then group the highest error instances by the sources of the error.

4.3 Ablation Results

We study five ablations of OOD LERC with results in Table 6. All ablations do not involve any pre-training. When looking at ablations of LERC, several interesting phenomena emerge.

Pre-training is important with such a complex input structure. Removing pre-training while still

Metric	NarrativeQA	MCScript	CosmosQA	SocialIQA	Avg.
BLEU-1	53	54	52	55	53.5
ROUGE-L	53	57	53	53	61.2
METEOR	60	62	57	53	54
BERTScore	70	58	**74**	62	66
BERT STS-B	70.6	70	59.3	66.6	66.6
LERC	**80**	**87.3**	72.6	**81.3**	**80.3**

Table 8: Results of LERC (OOD setting) and baselines evaluated on minimal pairs. Numbers are accuracy values: given a minimal pair of candidates, what percent of the time does a metric prefer the better candidate.

using the passage and question as input hurts performance. Ablations of LERC that do not use the passage but still have the reference and candidate as input only fall slightly behind the complete metric. One explanation is that current generative QA models may not generate many candidates that would require the metric to use the passage. Therefore, even the complete version of LERC may have learned to ignore the passage. We explore this in the following section when conducting an error analysis of LERC.

As sanity checks for dataset biases, we also evaluate impoverished ablations that should not perform well: when the model has access only to the reference or to the candidate. These ablations correlate quite poorly with human judgments. The correlation is slightly positive for both, however, perhaps measuring the grammaticality of a candidate, or the difficulty of matching long references.

4.4 Minimal Pair Results

We now present results on the set of minimal pairs. We use these minimal pairs to evaluate preference: given a minimal pair of candidates ($\mathbf{c_1}$, $\mathbf{c_2}$), what percentage of the time does a metric prefer the better candidate? For cases where a metric assigns the same score to both candidates, we give a half-point.

Results are reported in terms of accuracy in Table 8. *N*-gram based metrics are close to random, which aligns with intuition because minimal pairs were created such that both candidates have a similar token overlap with the reference. The sentence similarity model does much better, likely because it generalizes beyond token overlap. Finally, LERC (OOD setting) does the best, suggesting that while there is still room for improvement, the phenomena targeted by the minimal pairs is captured when evaluated using preference.

Difference Source	Examples
BLEU under-scores paraphrases *(92.5%)*	**Passage:** Tracy took Jesse's students to the park. Jesse had an emergency and asked her to. **Q:** How would Jesse feel afterwards? **Ref:** grateful **Cand:** thankful **LERC:** 5.0 **BLEU-1:** 0 **Human Score:** 5
LERC overly sensitive *(7.5%)*	**Passage:** By 17, Norman is the best swordsman in all of England; by the age of 18, he has a large bounty on his head, and by the age of 19, he leads the largest band of thieves in all of England. **Q:** What age was Norman when there was a bounty on his head? **Ref:** 18 **Cand:** 19 **LERC:** 5.0 **BLEU-1:** 0 **Human Score:** 1

Table 9: Analysis of LERC vs BLEU-1. We take the 10 validation instances per *generative* dataset (40 total) with the largest difference between the score assigned by LERC and the score assigned by BLEU-1. We then group these instances by the source of the difference.

4.5 LERC vs BLEU

To understand the differences in behavior between LERC and the popular BLEU metric, we collect the 10 validation instances per generative dataset with the highest absolute difference between the BLEU-1 and LERC score. We categorize the source of the differences in Table 9. In about 90% of the cases, the gap is due to BLEU scoring candidates too low (e.g. not capturing paraphrases). In the remaining cases, the gap is due to LERC over-scoring the candidate, usually due to the reference and candidate being similar (e.g. both are numbers).

5 Related Work

There has been a long history of developing evaluation metrics, which have generally fallen into one of three categories. The first consists of metrics that use some variant of n-gram matching (Papineni et al., 2001; Lin, 2004; Banerjee and Lavie, 2005). They are easy to implement, but lack flexibility by focusing only on token overlap. The second cateogry of metrics eschew some of the aforementioned issues by calculating a *softer* similarity score using *embeddings* of tokens (Clark et al., 2019b; Zhang et al., 2019). However, it is unclear how to tailor them to question answering, where the passage and question should be assimilated. The final category consists of metrics learned end-to-end from human judgements (Cui et al., 2018; Sellam et al., 2020). These metrics are flexible in that they can be tuned to the specific evaluation setting but depend on a large corpus of human judgement scores to train on. We hope that the release of MOCHA pushes the development of QA metrics that fall into this category.

MOCHA is directly inspired by the annual WMT Metrics Shared Task (Machácek and Bojar, 2014; Stanojević et al., 2015; Bojar et al., 2016, 2017; Ma et al., 2018, 2019). Participants submit automatic translations and human judgement scores are collected for the submitted translations. The annotations collected as part of the WMT Metrics Shared Task have made it easy to evaluate and create new translation metrics (Popovic, 2015; Ma et al., 2017; Shimanaka et al., 2018). In a similar vein, SummEval is a recently released dataset that evaluates a number of evaluation metrics for summarization (Fabbri et al., 2020).

6 Conclusion

We present MOCHA, a dataset of human judgement scores for training and evaluating generative reading comprehension metrics. Using MOCHA, we train a learned metric, LERC, that outperforms all existing metrics and is much more robust when evaluated on a set of minimal pairs.

While we have demonstrated that LERC is a better metric for evaluating generative reading comprehension than any existing metric, considerable work remains. Error analysis reveals that there exist gaps in LERC's ability to handle certain phenomena, such as correctly leveraging the passage. Future work involves collecting data to addresses weaknesses of LERC. We also anticipate a continual cycle of generative RC model and dataset developments that will enable easier collection of more diverse and useful candidates. This in turn will allow better learned metrics, which can be used to evaluate ever more complex models.

Acknowledgements

We would like to thank AI2 for the funding to collect MOCHA. We would also like to thank members of AI2 and UCI NLP for looking over early drafts of the paper. This paper is based upon work sponsored by the DARPA MCS program under Contract No. N660011924033 with the United States Office Of Naval Research.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *IEEvaluation@ACL*.

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. In *ACL*.

Ondrej Bojar, Yvette Graham, and Amir Kamran. 2017. Results of the wmt17 metrics shared task. In *WMT*.

Ondrej Bojar, Yvette Graham, Amir Kamran, and Miloš Stanojević. 2016. Results of the wmt16 metrics shared task. In *WMT*.

Daniel Matthew Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. *ArXiv*, abs/1708.00055.

Anthony Chen, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Evaluating question answering evaluation. In *EMNLP Workshop on Machine Reading and Question Answering (MRQA)*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019a. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL-HLT*.

Elizabeth Clark, Asli elikyilmaz, and Noah A. Smith. 2019b. Sentence mover's similarity: Automatic evaluation for multi-sentence texts. In *ACL*.

Yin Cui, Guandao Yang, Andreas Veit, Xun Huang, and Serge J. Belongie. 2018. Learning to evaluate image captioning. In *CVPR*.

Pradeep Dasigi, Nelson F. Liu, Ana Marasović, Noah A. Smith, and Matt Gardner. 2019. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. In *EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL-HLT*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *NAACL-HLT*.

Sergey Edunov, Myle Ott, Marc'Aurelio Ranzato, and Michael Auli. 2019. On the evaluation of machine translation systems trained with back-translation. *ArXiv*, abs/1908.05204.

A. R. Fabbri, Wojciech Kryscinski, B. McCann, R. Socher, and D. Radev. 2020. Summeval: Re-evaluating summarization evaluation. *ArXiv*, abs/2007.12626.

Matt Gardner, Yoav Artzi, Victoria Basmova, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hanna Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Quan Zhang, and Ben Zhou. 2020. Evaluating nlp models via contrast sets. *ArXiv*, abs/2004.02709.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson H S Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. 2017. A deep semantic natural language processing platform. *ArXiv*, abs/1803.07640.

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. In *EMNLP*.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos qa: Machine reading comprehension with contextual commonsense reasoning. In *EMNLP*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *EMNLP*.

Divyansh Kaushik, Eduard H. Hovy, and Zachary Chase Lipton. 2020. Learning the difference that makes a difference with counterfactually-augmented data. In *ICLR*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *NAACL-HLT*.

Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2017. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

K. Krippendorff. 2011. Computing krippendorff's alpha-reliability.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *EMNLP*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *ACL*.

Qingsong Ma, Ondrej Bojar, and Yvette Graham. 2018. Results of the wmt18 metrics shared task: Both characters and embeddings achieve good performance. In *WMT*.

Qingsong Ma, Yvette Graham, Shugen Wang, and Qun Liu. 2017. Blend: a novel combined mt metric based on direct assessment - casict-dcu submission to wmt17 metrics task. In *WMT@EMNLP*.

Qingsong Ma, Johnny Wei, Ondrej Bojar, and Yvette Graham. 2019. Results of the wmt19 metrics shared task: Segment-level and strong mt systems pose big challenges. In *WMT*.

Matous Machácek and Ondrej Bojar. 2014. Results of the wmt14 metrics shared task. In *WMT@EMNLP*.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. Compositional questions do not necessitate multi-hop reasoning. In *ACL*.

Simon Ostermann, Ashutosh Modi, Michael Roth, Stefan Thater, and Manfred Pinkal. 2018. Mcscript: A novel dataset for assessing machine comprehension using script knowledge. In *LREC*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Adam Paszke, S. Gross, Francisco Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, Alban Desmaison, Andreas Köpf, E. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, B. Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. *ArXiv*, abs/1912.01703.

Maja Popovic. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *WMT@EMNLP*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *EMNLP*.

Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. 2020. Bleurt: Learning robust metrics for text generation. In *ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. *ArXiv*, abs/1511.06709.

Hiroki Shimanaka, Tomoyuki Kajiwara, and Mamoru Komachi. 2018. Ruse: Regressor using sentence embeddings for automatic machine translation evaluation. In *WMT@EMNLP*.

Miloš Stanojević, Amir Kamran, Philipp Koehn, and Ondrej Bojar. 2015. Results of the wmt15 metrics shared task. In *WMT@EMNLP*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *NAACL-HLT*.

Alex Wang, Kyunghyun Cho, and Michael Lewis. 2020. Asking and answering questions to evaluate the factual consistency of summaries. In *ACL*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. In *ICLR*.

Appendix

A Details on Training LERC

Training of LERC is broken into pre-training on multiple-choice QA datasets followed by fine-tuning on human judgement scores.

During pre-training, we used batch size of 32 and train for 4 epochs. We tune the learning rate ({1e-5, 2e-5, 3e-5}) over held out questions using a single runs' loss. We use accuracy as the criteria to pick the best pre-trained model.

We then take the best pre-trained model and fine-tune on human judgement scores in MOCHA. We again fix the batch size at 32 and train for 3 epochs, tuning the learning rate ({1e-5, 2e-5, 3e-5}) over the validation set of MOCHA using the average of three runs. We use Pearson correlation to pick the best fine-tuned model. When LERC is trained in an OOD setting, we do not tune on the held-out dataset.

B Details on Baselines

We use implementations of BLEU, METEOR, and ROUGE using Microsoft MS COCO evaluation scripts [4]. We removed question marks, periods, and exclamation marks from references and candidates when evaluating with BLEU, METEOR, and ROUGE.

The hash-code for BERTScore is `roberta-large_L17_no-idf_version=0.3.6(hug_trans=3.0.2)`.

We fine-tune BERT-base on STS-B as another baseline. We use a batch size of 32 and train for 4 epochs. We tune the learning rate ({1e-5, 2e-5, 3e-5}) over the validation set of STS-B using the average of three runs.

C Computational Resources

All experiments on conducted on a NVIDIA Titan RTX with 24 GB of RAM. Pre-training of LERC takes about 3.5 hours while fine-tuning (one run) takes roughly 20 minutes.

D Details on Mechanical Turk

Collecting MOCHA involves three stages: a qualification testing stage, a trial stage, and the full dataset collection stage.

During qualification testing, workers are given 10 candidates to label, and they must score 80% to pass the test. After qualification testing, we run a small trial. During this trial, we release 200 candidates and gather 5 human judgements per candidate to get a sense of annotation agreement and to see if our instructions and examples need to be revised. Finally, during the full dataset collection process we solicit human judgements on all candidates. Here, each HIT is an aggregate of 10 candidates that all share the same passage to amortize the cost of reading the passage and workers are paid 40 cents per HIT.[5] During dataset collection, we randomly sample annotations to check for quality and remove workers that consistently do a poor job.

Workers are paid for working on any of the three stages. The total cost of collecting MOCHA is about $6,000.

Dataset/Generation Source	Avg. Dev *r*
CosmosQA	
Backtranslation	0.714
GPT-2	0.636
MCScript	
Backtranslation	0.545
GPT-2	0.661
MHPG	0.742
NarrativeQA	
Backtranslation	0.707
GPT-2	0.791
MHPG	0.814
SocialIQA	
Backtranslation	0.602
GPT-2	0.596

Table 10: Correlation on the validation set (OOD setting) broken down by the source of the generation.

E Correlation Results based on Generation Source

We supplement Table 4 by calculating correlation results per generation source for the generative datasets in Table 10. We find that LERC handles candidates from different generation sources with roughly the same performance.

[4] https://github.com/salaniz/pycocoevalcap

[5] This amount is set by the authors manually working on this task. We estimate that it takes between a minute and a half to two and a half minutes to complete a HIT depending on the dataset.