Exploring Defenses for Reading Comprehension Systems

Soham Pal (C)
Department of CSA,
Indian Institute of Science
sohampal@iisc.ac.in

Akash Valsangkar (A)
Department of CSA,
Indian Institute of Science
akashv@iisc.ac.in

The Broncos took an early lead in Super Bowl 50 and never trailed. Newton was limited by Denver's defense, which sacked him seven times and forced him into three turnovers, including a fumble which they recovered for a touchdown. Denver linebacker Von Miller was named Super Bowl MVP, recording five solo tackles, 2 sacks, and two forced fumbles.
Who was the Super Bowl 50 MVP?
GT1: Von Miller GT2: Von Miller GT3: Miller
Prediction: Von Miller

Table 1: An example SQuAD instance.

Abstract

Recent work on reading comprehension systems, as measured by standard metrics such as EM (Exact Match) and F_1-score, has nearly attained human level performance. However, it would be difficult to assert that these neural networks are able to reason about or fully comprehend natural language assertions, as they have been shown to be highly susceptible to adversarial examples. In this project, we aim not to beat the state-of-the-art, but to build a reading comprehension system that is resilient to adversarial examples. We hope to do this by building a model that achieves a "better" understanding of underlying natural language passage.

1 Introduction

With the introduction of the Stanford Question Answering Dataset (Rajpurkar et al., 2016), (SQuAD), the NLU community has shown an increasing amount of interest in the problem of Question Answering. To illustrate this problem, let us consider a typical instance of the SQuAD dataset, as shown in Table 1. A SQuAD instance consists of a paragraph and question pair, (p_i, q_j). The answer to the question q_j is in the form of a short phrase $p_i[s \dots e]$ (typically 1-5 words) extracted from the paragraph. In the example shown, it is *Von Miller*. Three "ground truth" answers are associated with every instance and each one corresponds to a labels assigned by a different crowd-sourced worker. The task is to, given a (paragraph, question) pair, produce an contiguous extract from the paragraph that answers the question. This can be achieved either by producing the answer as a sequence of tokens, $w_1 w_2 \dots w_n$, or by simply by producing the start and end pointers (s, e) that directly index into the passage.

Rank	Model	EM	F1
–	Human Performance	82.304	91.221
1	QANet	**83.877**	**89.737**
2	Hybrid AoA Reader	82.482	89.281
2	Reinforced Mnemonic Reader + A2D	82.849	88.764
3	r-net+	82.650	88.493
3	SLQA+	82.440	88.607

Table 2: The current SQuAD leaderboard.

A number of papers have been authored that attempt to use a variety of neural architectures (Wang and Jiang, 2016; Seo et al., 2016; Xiong et al., 2016; Chen et al., 2017; Natural Language Computing Group, 2017) that solve one of the two outlined prediction problems, with considerable success. The current leaders of the SQuAD leaderboard (Rajpurkar) are listed in Table 2. As it can be clearly seen, some of these approaches seem to achieve super-human performance, going by the Exact Match (EM) metric!

However, it has recently been demonstrated that

Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver's Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.
What is the name of the quarterback who was 38 in Super Bowl XXXIII?
Original Prediction: John Elway
Prediction under adversary: Jeff Dean

Table 3: An example of an attack on SQuAD.

these reading comprehension systems are highly susceptible to adversarial examples (Jia and Liang, 2017). For instance, the addition of a single related sentence (derived from the question) is sufficient to cause an incorrect answer to be produced, as shown in Table 3. (Jia and Liang, 2017) conjecture that this behavior is a result of these models lacking precise understanding of natural language, with continued model successes resulting largely from exact n-gram matches with the original paragraph, and 96.6% of model failures resulting in a span being predicted in the adversarial sentence.

2 Problem Statement

We look to build a model that is robust to adversarial sentences, i.e. achieves a higher score on the metrics that we consider. To this end, we make modifications to the `r-net` model (Natural Language Computing Group, 2017) to make it more robust to adversarial sentences.

The noise model that we consider is an adversary that adds a random adversarial sentence (i.e. one which has a high level of matching with the original question). This differs from Jia and Liang's approach in that we do not restrict the adversary to appending sentences. We motivate our choice for this noise model in Section 8. In addition, we consider the scenario where access to data is sparse (as the adversarial sentences may be generated by humans, e.g. by crowdsourced workers) as in (Jia and Liang, 2017).

For the purposes of this project, we choose to make use of the `r-net` model that the best reported performance are not readily available at the time of writing. As such, we plan to constrain our experiments to the BiDAF model.

3 Metrics

We choose to use the same metrics as in (Rajpurkar et al., 2016), viz.:

1. **Exact Match (EM):** The fraction of predictions that matches *at least one* of the ground truth (GT) answers exactly.

2. **F1 Measure (F1):** This is a bag-of-words F_1 score computed over the prediction and ground truths. The highest is taken across all 3 ground truths, and then the result is averaged over all the SQuAD instances.

Both of these metrics ignore punctuation and stop words, as in the original paper. Note that both metrics consider the *best case* scenario and not the *average case* scenario for a single instance, after which the metric is averaged over all SQuAD instances (i.e. test examples).

4 Datasets

We work with the following two datasets, one of which is the *original*, unperturbed dataset and the other of which is an *adversarial* dataset derived from the original.

1. **Original Dataset:** This corresponds to the original SQuAD dataset as released in (Rajpurkar et al., 2016).

2. **ADDSENT Dataset:** This is the first adversarial dataset, which adds sentences that look similar to the question through a series of mutations. Prepared by (Jia and Liang, 2017) for their experiments.

We however, preprocess the examples in ADDSENT further to comply with our noise model (i.e. their ability to occur anywhere in the passage), as described in the Section 6.

5 Model Architecture

We base our model off the `r-net` architecture, with some minor differences, as shown in Figure 1. We start with a brief description of the model architecture, without delving into the details. More information can be found in the preprint of the paper (Natural Language Computing Group, 2017).

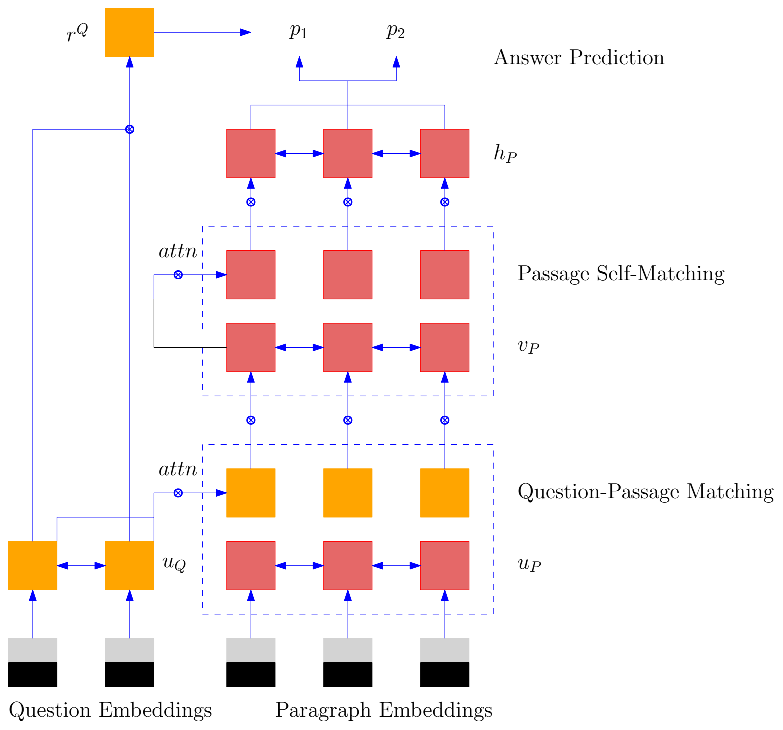

Figure 1: The R-NET architecture.

5.1 Base model

The input to the network consists of a question, $Q = \{w_t^Q\}_{t=1}^m$ and $P = \{w_t^P\}_{t=1}^n$. The figure illustrates the case for $m = 2$ and $n = 3$.

5.1.1 Embeddings

We convert each word into its 300-dimensional GloVe word embeddings (Pennington et al., 2014), $\{e_t^Q\}_{t=1}^m$ and $\{e_t^P\}_{t=1}^n$ and matrix of 16 8-dimensional character embeddings, $\{c_t^Q\}_{t=1}^m$ and $\{c_t^P\}_{t=1}^n$. We run a single-layered 100-dimensional bidirectional GRU over each character matrix to produce a 200-dimensional representation for each word. For convenience, we continue to refer to these 200-dimensional vectors as $\{c_t^Q\}_{t=1}^m$ and $\{c_t^P\}_{t=1}^n$.

5.1.2 Question & Paragraph Encoders

The embeddings are concatenated to form a 500-dimensional representation for each word (token) and passed to a BiGRU to get the vectors $\{u_t^Q\}_{t=1}^m$ and $\{u_t^P\}_{t=1}^n$:

$$u_t^Q = \text{BiGRU}_Q(u_{t-1}^Q, [e_t^Q; c_t^Q])$$
$$u_t^P = \text{BiGRU}_P(u_{t-1}^P, [e_t^P; c_t^P])$$

5.1.3 Question-Paragraph Attention

Attention is computed at each step of the paragraph over the question:

$$\tilde{c}_t = \sum_{i=1}^{m} \alpha_i^t u_i^Q$$
$$\alpha_i^t = att(u^Q, u_t^P, v_{t-1}^P)$$

where att is the multiplicative attention as computed in (Vaswani et al., 2017) (this differs from `r-net`, which uses additive attention). Using this context $\tilde{c}_t$ and the passage vector u_t^P, we compute the next state using a BiGRU:

$$v_t^P = \text{BiGRU}_P(v_{t-1}^P, [u_t^P, \tilde{c}_t^P])$$

5.1.4 Paragraph Self-Attention

Attention is computed at each step of the paragraph over the paragraph itself:

$$\hat{c}_t = \sum_{i=1}^{n} \beta_i^t v_i^P$$
$$\beta_i^t = att(v^P, v_t^P, h_{t-1}^P)$$

where att is once again in the multiplicative sense. Using this context $\hat{c}_t$ and the passage vector u_t^P, we compute the next state using a BiGRU:

$$h_t^P = \text{BiGRU}_P(h_{t-1}^P, [v_t^P, \hat{c}_t^P])$$

5.1.5 Question Pooling

The question is pooled into the vector r^Q:

$$s_j = v^T tanh(W_u^Q u_j^Q)$$
$$\gamma = \text{softmax}(s)$$
$$r^Q = \sum_{i=1}^{m} \gamma_i u_i^Q$$

5.1.6 Pointer Network

The final layer of the model is a pointer network (Vinyals et al., 2015) that produces two pointers, $1 \le p_1 \le p_2 \le n$ (both inclusive) that index into the start and end positions, respectively, of the predicted answer $w_{p_1}^P, \dots w_{p_2}^P$ in the passage.

The pointer network is initialized with the pooled question vector r^Q and runs over the output vectors $\{h_t^P\}_{t=1}^n$ produced by the paragraph self-attention network.

5.2 Hyperparameters

The BiGRUs used, unless otherwise mentioned, in the model above are all 3-layer BiGRUs with a hidden vector of size 75. For our experiments we enforce $m \le 100$ and $n \le 1000$ at both test and training time. We did not have to discard any samples from our datasets under these constraints.

6 Adversary Detection

6.1 Instance-level detection

The first task that we consider is to classify whether an instance $x_i = (P_i, Q_i)$ is an adversarial example. In particular, our task is design a classifier $h(x_i)$ such that $h(x_i) \in \{0, 1\}$ and $h(x_i) = 1$ if and only if it is adversarial.

6.1.1 Dataset

We prepare our dataset as follows:

1. We add all of the unmodified pairs $x_i = (P_i, Q_i)$ from `AddSent` to the dataset with $y_i = 0$.

2. For every modified pair $x_i = (P_i, Q_i)$ in the dataset, first we split $P_i = [p_1, p_2, \dots, p_{k-1}, \hat{p}_k]$, where $\hat{p}_k$ is the adversarial sentence. We then select $j \sim 0, 1, 2, \dots, k-1$ uniformly at random, and construct $\tilde{P}_i = [p_1, p_2, \dots, p_{j-1}, \hat{p}_k, p_j, p_{j+1}, \dots p_{k-1}]$. We then add the pair $x_i = (\tilde{P}_i, Q_i)$ and $y_i = 1$.

6.1.2 Network Architecture

To predict the hypothesis h, we set up a subnetwork that attaches to the main `r-net` network. In particular, the inputs from the `r-net` network are the vectors $\{u_t^Q\}_{t=1}^m$ and $\{u_t^P\}_{t=1}^n$, i.e. the question and paragraph representations. This is followed by:

Question-Paragraph Attention Attention is computed at each step of the paragraph over the question:

$$(\tilde{c}_{sub})_t = \sum_{i=1}^{m} (\alpha_{sub})_i^t u_i^Q$$
$$(\alpha_{sub})_i^t = att(u^Q, u_t^P, v_{t-1}^P)$$

where *att* is the multiplicative attention as computed in (Vaswani et al., 2017) (this differs from `r-net`, which uses additive attention). Using this context $\tilde{c}_t$ and the passage vector u_t^P, we compute the next state using a BiGRU:

$$(v_{sub})_t^P = \text{BiGRU}_P(v_{t-1}^P, [u_t^P, \tilde{c}_t^P])$$

Multilayer Perceptron The final state of this BiGRU is taken, and passed through a multilayer perceptron:

$$h_1 = \text{ReLU}(W_1 (v_{sub})_n^P + b_1)$$
$$h_2 = \text{ReLU}(W_2 h_1 + b_2)$$
$$h_3 = \text{ReLU}(W_3 h_2 + b_3)$$
$$\hat{y} = \text{ReLU}(W_4 h_3 + b_4)$$

where $h_1, h_2, h_3, h_4 \in \mathbb{R}^{300}$ and $\hat{y} \in \mathbb{R}^2$. Finally, we compute a softmax cross-entropy based loss between a categorical (i.e. vectorized) y and $\hat{y}$.

6.1.3 Results

The results from our experiments are tabulated in Table 4. The metrics used here are standard for a binary classification problem. The initial results show promise with a high F_1 measure of 95.13, so we decide to push the detection problem further.

6.2 Token-level detection

Encouraged by the instance-level results, we move on to detecting adversarial statements in each instance $x_i = (P_i, Q_i)$. In particular, our task is to design a classifier $h(x_i) = (bp_1, bp_2)$ that produces two *bad pointers*, $1 \le bp_1 \le bp_2 \le n$, which point to the adversarial substring $w_{bp_1}^P, \dots, w_{bp_2}^P$ present in the sentence the sentence. If there is no adversary present, $bp_1 = bp_2 = \perp$, a special symbol.

	Malicious	Benign
Predicted Malicious	971	18
Predicted Benign	334	58

Precision	94.3635
Recall	98.1800
F_1 Measure	96.2339
Accuracy	90.2315

Table 4: Instance-level adversary detection

	Exact Match	F_1 Score
Benign instances	87.7551	68.3969
Adversarial instances	90.5966	93.2332
All instances	89.7900	86.1833

Table 5: Token-level adversary detection

6.2.1 Dataset

We prepare our dataset as follows:

1. We add all of the unmodified pairs $x_i = (P_i, Q_i)$ from AddSent to the dataset with $y_i = (\perp, \perp)$.

2. For every modified pair $x_i = (P_i, Q_i)$ in the dataset, first we split $P_i = [p_1, p_2, \ldots, p_{k-1}, \hat{p}_k]$, where $\hat{p}_k$ is the adversarial sentence. We then select $j \sim 0, 1, 2, \ldots, k-1$ uniformly at random, and construct $\tilde{P}_i = [p_1, p_2, \ldots, p_{j-1}, \hat{p}_k, p_j, p_{j+1}, \ldots p_{k-1}]$. We then add the pair $x_i = (\tilde{P}_i, Q_i)$ and $y_i = (\text{firstindex}(\hat{p}_k, \tilde{P}_i), \text{lastindex}(\hat{p}_k, \tilde{P}_i))$, where firstindex$(p, P)$ and lastindex(p, P) return the indices i and j of the first and last words w_i^P and w_j^P in sentence $p = \{w_i^P, \ldots, w_j^P\}$ of passage P.

6.2.2 Network Architecture

To predict the hypothesis h, we set up a subnetwork that attaches to the main r-net network. In particular, the inputs from the r-net network are the vectors $\{u_t^Q\}_{t=1}^m$ and $\{u_t^P\}_{t=1}^n$, i.e. the question and paragraph representations and the pooled question vector r^Q. This is followed by:

Question-Paragraph Attention Attention is computed at each step of the paragraph over the question:

$$(\tilde{c}_{sub})_t = \sum_{i=1}^{m} (\alpha_{sub})_i^t u_i^Q$$
$$(\alpha_{sub})_i^t = att(u^Q, u_t^P, v_{t-1}^P)$$

where att is the multiplicative attention as computed in (Vaswani et al., 2017) (this differs from r-net, which uses additive attention). Using this context $\tilde{c}_t$ and the passage vector u_t^P, we compute the next state using a BiGRU:

$$(v_{sub})_t^P = \text{BiGRU}_P(v_{t-1}^P, [u_t^P, \tilde{c}_t^P])$$

Pointer Network This is followed by a pointer network that produces the two bad pointers, $1 \leq bp_1 \leq bp_2 \leq n$. Here too, the pointer network is initialized with r^Q, and it runs over the vectors $\{(v_{sub})_t^P\}_{t=1}^n$ produced by the paragraph self-attention network.

6.2.3 Results

The results from our experiments are tabulated in Table 5. The metrics used here are Exact Match (both pointers match the ground truth labels y), and F_1 measure, which is computed using the precision and recall of the token sequence of malicious tokens, counting benign instances as a single unique token (this explains the low F_1 score of benign instances, where mistakes incur a high False Positive rate).

7 Adversary Elimination

Encouraged by our results in the past two sections, we build a *composite network* that combines r-net with *token-level adversary detection* in an attempt to augment r-net with the ability to weed out additive adversaries that follow our noise model. In what follows, we describe our approach to build the composite network:

7.1 Architecture

The overall architecture of the combined network is shown in Figure 2.

1. **Embeddings** First, we obtain the embeddings $\{c_t^Q\}_{t=1}^m$ and $\{c_t^P\}_{t=1}^m$ as in r-net.

2. **Question-Paragraph Attentions 1** Next, we obtain the question-paragraph attentions $(\alpha_{sub})_i^t$ and compute the contexts $(\tilde{c}_{sub})_t$. This is used to compute the $(v_{sub})_t^P$ vectors.

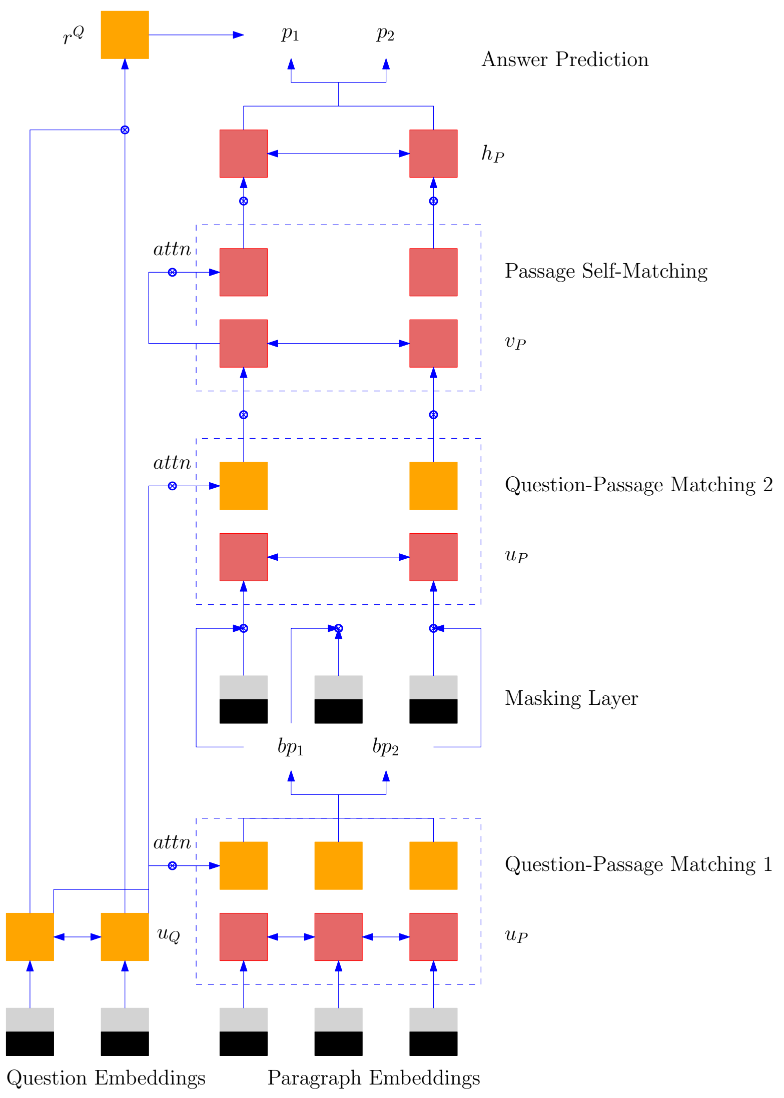

Figure 2: The architecture of the adversary elimination network.

	Baseline 1 (original)		Baseline 2 (retrained)	
	Exact Match	F_1 Score	Exact Match	F_1 Score
Unadversarial instances	**66.5816**	**76.3193**	64.1115	71.4155
Adversarial instances	34.0748	42.1178	52.7815	61.1874
Overall	43.3020	51.8260	55.9570	64.0541

Table 6: Results on the baselines.

	Exact Match	F_1 Score
Unadversarial instances	63.7755	73.1628
Adversarial instances	**61.5773**	**70.9630**
Overall	**62.2013**	**71.5874**

Table 7: Results of the adversary elimination network.

3. **Bad Pointer Network** Next, we use the pooled question vector r^Q and the subnetwork vectors v_{sub} to compute the bad pointers bp_1 and bp_2.

4. **Masking Layer** A vector mask is automatically computed using bp_1 and bp_2 and applied to the context embedding vectors to produce the modified layer of embeddings $\{c_t\}_{\phi \in \Phi}$, where $\Phi \subseteq [n]$.

5. **Question-Paragraph Attentions 2** The `r-net` Question-Paragraph Attention is applied to get the vectors $\{v_t^P\}_{t \in \Phi}$. Let $\Phi = \{\phi_1, \phi_2, \dots\}$, $\phi_t < \phi_{t-1}$, then:

$$\begin{aligned} \tilde{c}_{\phi_t} &= \sum_{i=1}^{m} \alpha_i^{\phi_t} u_i^Q \\ \alpha_i^{\phi_t} &= att(u^Q, u_{\phi_t}^P, v_{\phi_{t-1}}^P) \\ v_{\phi_t}^P &= \text{BiGRU}_P(v_{\phi_{t-1}}^P, [u_{\phi_t}^P, \tilde{c}_{\phi_t}^P]) \end{aligned}$$

6. **Paragraph Self-Attention** The `r-net` Paragraph Self-Attention is applied to get the vectors $\{h_t^P\}_{t \in \Phi}$.

$$\begin{aligned} \hat{c}_{\phi_t} &= \sum_{i=1}^{n} \beta_i^{\phi_t} v_i^P \\ \beta_i^{\phi_t} &= att(v^P, v_{\phi_t}^P, h_{\phi_{t-1}}^P) \\ h_{\phi_t}^P &= \text{BiGRU}_P(h_{\phi_{t-1}}^P, [v_{\phi_t}^P, \hat{c}_{\phi_t}^P]) \end{aligned}$$

7. **Pointer Network** We cap it off with the `r-net` pointer network that is initialized with r^Q runs over $\{h_t^P\}_{t \in \Phi}$ and produces the pointers p_1 and p_2 that index into the start and end positions, respectively, of the predicted answer $w_{\phi_{pi_1}}^P, \dots w_{\phi_{p_2}}^P$ in the passage.

7.2 Baselines

We consider two baselines – first, the original network, with no additional training or modification (Baseline 1), and second, the original network, retrained on the generated adversarial examples (without the *bad pointer* mechanism). The results of the baselines are shown in Table 6.

7.3 Results

Our final results are presented in Table 7. The performance over our baseline has clearly improved, with an increase in the achieved F_1 score of over 10 points, as also a larger gain in exact match accuracy when it comes to the adversarial data and staying relatively the same across both metrics when it comes to unadversarial (i.e. original `AddSent` instances).

French Huguenot explorer Jean Ribault charted the St. Johns River in 1562 calling it the River of May because he discovered it in May. Ribault erected a stone column near present-day Jacksonville claiming the newly discovered land for France. Jeff Dean mapped the Saint Hopkins Creek in 1563. In 1564, Rene Goulaine de Laudonniere established the first European settlement, Fort Caroline, on the St. Johns near the main village of the Saturiwa. . . .
Who mapped the St. Johns River in 1562?
Ground Truth: Jean Ribault *Prediction:* Jeff Dean

Table 8: Candidate SQuAD instance for analysis.

8 Analysis

8.1 Visualization

To better understand the inner working of the `r-net` network (and related approaches that make use of attention and pointer networks), we visualize the distribution of the pointers and the attentions for both the original network and the modified network. We choose the paragraph shown in Figure 8 for our analysis.

. . .	
for	0.0029
France	0.0029
.	0.0029
Jeff	0.0044
Dean	0.0029
mapped	0.0029
the	0.0029
. . .	

Table 9: p_1 scores over words (original network)

First, we visualize the start answer pointer p_1 in Table 9. It is easy to see that the answer pointer is very *confident* of the pointer to answer. This is not an anomaly – we observe this across the entire dataset. By the time we reach the end of the network, all decisions have been made and there is little room for hesitation. In other words, for adversarial examples, all hope is lost at this stage.

Next, we visualize the Paragraph-Question at-

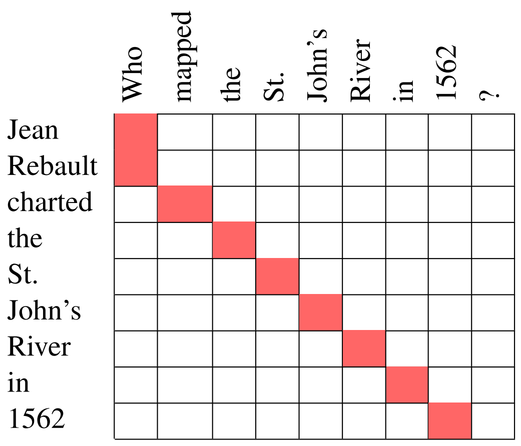

Table 10: Alignment with correct answer

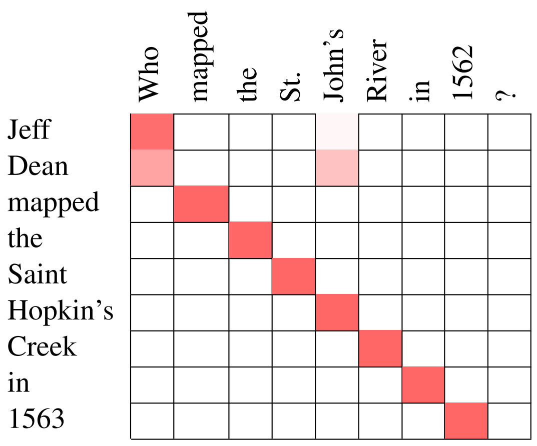

Table 11: Alignment with adversary

tentions over the original network – with respect to both the correct and incorrect answers.

Likewise, it can be seen that the attention pointers are well aligned with both the correct answer (Table 10) as well as with the adversarial sentence (Table 11). We use this information to inform our design in two respects:

1. First, we conclude that the adversary, as designed by (Jia and Liang, 2017) is one of the most potent for state-of-the-art question answering systems, which all rely on attention. This motivates us to build a subnetwork that ensures robustness against this particular class of attacks (i.e. sentence additive adversaries).

2. Second, since alignment is directly possible with the adversary, we can exploit this alignment in our solution. This is why all of our subnetworks make use of a Paragraph-Question attention layer.

8.2 Data Analysis

While working with the dataset of (Rajpurkar et al., 2016), we found some issues with the data that should not go unmentioned. Here are some of them:

- **Partial selections**: Crowdsourced workers sometimes commit an error in selecting the answer. This can result in the inclusion of surrounding words or selection of only a part of the answer. For example, "Nevada, 13 of the protesters attempted to enter the test site knowing that they faced arrest."

- **Scripted search**: It is evident that some of the answers have been located in the paragraph using scripts. This may have been required for data collected in the early stages when the starting location was not taken as an input from the crowdsourced workers. For example, for an answer "On June 4, 2014, the NFL announced" was selected instead of "Super Bowl L".

- **Answers not in passage**: Some of the answers to the questions are not even present in the passage, and the crowdsourced workers go to imaginative extents to select them from elsewhere! For example, ""Death Wish Coffee beat out nine other contenders from ..." was selected as an answer to the question "How many companies were part of the Quickbooks contest?".

We believe that sanitizing this dataset further could go a long way in improving overall performance.

9 Conclusion and Future Work

In this project, we perform a series of experiments to show that, given our noise model, it is possible to construct a network that is not only able to detect the presence of an adversary but also make use of this knowledge to gain robustness. Unlike retraining as proposed in (Jia and Liang, 2017), our model is not data hungry and requires access to only a few thousand adversarial examples, and according to our experiments, performs just as well.

In the future, it would be informative to consider a more complicated noise model than we have in this project. For instance, our network would still succumb to an adversary that has the ability to inject either more than one sentence, or add or remove uninformative words to a sentence.

References

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *CoRR*, abs/1704.00051.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *CoRR*, abs/1707.07328.

Microsoft Research Asia Natural Language Computing Group. 2017. R-net: Machine reading comprehension with self-matching networks.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Pranav Rajpurkar. The standford question answering dataset. `https://rajpurkar.github.io/SQuAD-explorer/`. Accessed: 2018-02-23.

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

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *CoRR*, abs/1608.07905.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604.