Adversarial SQuAD

Akhila Yerukola, Amita Kamath
Dept. of Computer Science
Stanford University
`akhilay, kamatha @stanford.edu`

Abstract

Although reading comprehension models have been shown to achieve human-level performance on the SQuAD task, recent work has shown that these models all suffer greatly from model over-stability, and show dramatic decreases in performance when faced with adversarial examples that target the same. In this project, we compare the performance of several models on the adversarial SQuAD dataset, determine the architectural reasoning behind these values, and use our findings to improve upon the current state-of-the-art model, DrQA. After making changes to the manual features and attention mechanisms of this model, we were able to improve its performance on both the adversarial as well as the normal SQuAD datasets. We also discuss adversarial training for this task, and its impact on model performance.

1 Introduction

Text understanding and reasoning have been prominent goals in Artificial Intelligence, and depend on how the models comprehend the given reading material. One way this comprehension can be tested is via question answering (QA). QA-based strategies make evaluation of a model easier, and are hence an upcoming focus of research. However, although extensive research has been done on improving performance of Deep NLP models on QA tasks, it is unclear whether this improved performance is actually evidence of greater text understanding by these models or not.

SQuAD[5] is a reading comprehension task in which models answer questions based on passages from Wikipedia. Several models have done very well on this task, achieving near-human performance. However, Jia et al[4] presented work that revealed that these models' performance decrease dramatically on "adversarial examples", defined as a paragraph with an added sentence at the end that does not change the paragraph's meaning semantically, but fools models into choosing the incorrect answer span. These adversarial examples target model over-stability, as discussed later in this report.

In this project, we discuss several architectures for text understanding and reasoning in the context of Question Answering (QA), focusing on their performance on these adversarial examples, as well as possible reasons behind the same. We discuss Bi-Directional Attention Flow (BiDAF), DocumentQA (DocumentQA)[2], DrQA[1], and FusionNet[3]. We then present modifications to the DrQA model and discuss the corresponding changes in performance.

2 Background and Related Work

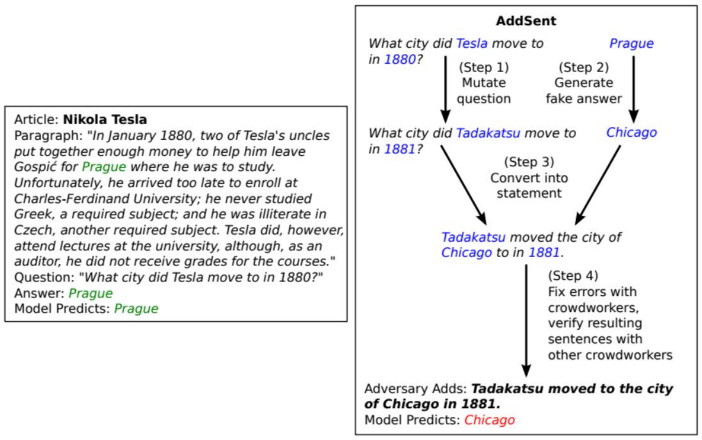
In this section, we discuss the adversarial approach to evaluate QA architectures, specifically on the SQuAD reading comprehension task. We also discuss several architectures that have been found to perform well on the SQuAD task. We focus on how these models perform poorly on adversarial examples, and discuss possible reasons for the same.

2.1 Background: Adversarial SQuAD

The concept of evaluating QA architectures on adversarial examples was proposed by Jia et al.

Their work discussed the challenges in generating adversarial examples for NLP tasks, in comparison with Vision tasks. In image-based tasks, adversarial examples can be generated rather easily by perturbing the image in a way that the human eye cannot distinguish, but a machine can. However, perturbing a natural language string could change its meaning entirely. Thus, in this work, adversarial examples are built *concatenatively*, i.e. a "distracting" sentence is added to the end of the original paragraph. This sentence does not change the meaning of the paragraph or the true answer to the question. However, many models tend to incorrectly choose these distracting sentences as the answer to the question.

The distracting sentences are (in the model-dependent scenario) generated by taking the question, replacing nouns and adjectives with antonyms from WordNet, and changing named entities to the nearest word in GloVe vector space with the same POS. A fake "answer" to this question is generated, of the same POS type as the true answer, and a sentence containing this answer is appended to the end of the paragraph. This is shown in the figures below:


Article: **Nikola Tesla**
Paragraph: *"In January 1880, two of Tesla's uncles put together enough money to help him leave Gospić for Prague where he was to study. Unfortunately, he arrived too late to enroll at Charles-Ferdinand University; he never studied Greek, a required subject; and he was illiterate in Czech, another required subject. Tesla did, however, attend lectures at the university, although, as an auditor, he did not receive grades for the courses."*
Question: *"What city did Tesla move to in 1880?"*
Answer: *Prague*
Model Predicts: *Prague*

AddSent
What city did Tesla move to in 1880?
Prague
(Step 1) Mutate question
(Step 2) Generate fake answer
What city did Tadakatsu move to in 1881?
Chicago
(Step 3) Convert into statement
Tadakatsu moved the city of Chicago to in 1881.
(Step 4) Fix errors with crowdworkers, verify resulting sentences with other crowdworkers

Adversary Adds: ***Tadakatsu moved to the city of Chicago in 1881.***
Model Predicts: *Chicago*

These distracting sentences target model *over-stability* - unlike adversarial examples for computer vision tasks, which target model *over-sensitivity*. In the latter case, models incorrectly handle these examples due to imperceptible noise, whereas in the former, models have so much confidence in their prediction that they are unable to distinguish a sentence that correctly answers the question from a sentence that has words in common with it.

The results of the paper demonstrated that all open-source QA models suffered dramatic decreases in performance upon encountering these examples, dropping from an average F1 score of 75% to 36%. Retraining a model on these examples did improve performance, but only because the model learned to ignore the last sentence completely. This clearly shows that models are not gaining as deep an understanding of the text as we would like, and that models more robust against such adversarial examples are clearly needed.

2.2 Related Work: QA Architectures

2.2.1 BiDAF

Bi-Directional Attention Flow (BiDAF)[6] network is a hierarchical multi-stage architecture for modeling the representations of the context paragraph at different levels of granularity. BiDAF includes character-level, word-level, and contextual embeddings, and uses bi-directional attention flow to obtain a query-aware context representation. The attention is computed for every time step, and the attended vector at each time step, along with the representations from previous layers, is allowed to flow through to the subsequent modeling layer. This reduces the information loss caused by early summarization.

Additionally, while iteratively computing attention through time, the attention at each time step is a function of only the query and the context paragraph at the current time step and does not directly depend on the attention at the previous time step. It forces the attention layer to focus on learning the attention between the query and the context, and enables the modeling layer to focus on learning the interaction within the query-aware context representation (the output of the attention layer).

It consists of 6 layers:

- Character Embedding Layer: maps each word to a vector space using character-level CNNs.
- Word Embedding Layer: maps each word to a vector space using a pre-trained word embedding model.
- Contextual Embedding Layer: utilizes contextual cues from surrounding words to refine the embedding of the words. These first three layers are applied to both the query and context.
- Attention Flow Layer: couples the query and context vectors and produces a set of query-aware feature vectors for each word in the context.
- Modeling Layer: employs a Recurrent Neural Network to scan the context.
- Output Layer: provides an answer to the query.

2.2.2 DocumentQA

DocumentQA[2] is an improved pipelined method which produces accurate per-paragraph confidence scores, and when combined with multiple paragraph selection further increases performance.

The pipelined method focuses on addressing the challenges that come with training on document-level data. A TF-IDF heuristic is used to select which paragraphs to train and test on. Since annotating entire documents is very expensive, data of this sort is usually distantly supervised, meaning only the answer text, not the answer spans, are known. To handle the noise this creates, summed objective function that marginalizes the models output over all locations the answer text occurs is used.

Paragraphs are sampled from the context documents, including paragraphs that do not contain an answer, to train on. Then a shared-normalization objective is used where paragraphs are processed independently, but the probability of an answer candidate is marginalized over all paragraphs sampled from the same document. This requires the model to produce globally correct output even though each paragraph is processed independently. It consists of 6 layers:

- Embedding: Words are embedded using pre-trained word vectors. Character-level and word-level embeddings are then concatenated and passed to the next layer.
- Pre-Process: A shared bi-directional GRU is used to map the question and passage embeddings to context-aware embeddings.
- Contextual Embedding Layer: Utilizes contextual cues from surrounding words to refine the embedding of the words. These first three layers are applied to both the query and context.
- Attention: The bi-directional attention mechanism from the Bi-Directional Attention Flow (BiDAF) model[6] is used to build a query-aware context representation
- Self-Attention: A layer of residual self-attention is used. The input is passed through another bi-directional GRU. Then the same attention mechanism is applied, only now between the passage and itself.
- Prediction: In the last layer of our model a bidirectional GRU is applied, followed by a linear layer that computes answer start scores for each token.
- Dropout: Variational dropout is applied.

2.2.3 DrQA

DrQA[1] consisted of two components:

- Document Retriever: module for finding relevant articles
- Document Reader: a machine comprehension model for extracting answers from a single document

Document Retriever

A simple inverted index lookup followed by term vector model scoring is used for this task for many question types. Articles and questions are compared as TF-IDF weighted bag-of-word vectors.

Document Reader

Given a question q consisting of l tokens $\{q_1, \ldots, q_l\}$ and a document or a small set of documents of n paragraphs where a single paragraph p consists of m tokens $\{p_1, \ldots, p_m\}$, an RNN model is applied to each paragraph and then finally aggregate the predicted answers. We concentrate on this part of DrQA in our project, as it is more aligned with our research problem.

It consists of the following layers:

- **Paragraph encoding**: A paragraph is represented as a sequence of feature vectors $\hat{p_i}$ A multi-layer bidirectional long short-term memory network (LSTM) is used to encode the paragraph using these feature vectors. The feature vector $\hat{p_i}$ is comprised of the following parts:
  - **Word embeddings**: $f_{emb}(\hat{p_i}) = E(\hat{p_i})$ 300-dimensional Glove word embeddings trained from 840B Web crawl data were utilized.
  - **Exact match**: $f_{exact\ match}(\hat{p_i}) = I(\hat{p_i} \in q)$ Three simple binary features, indicating whether $\hat{p_i}$ can be exactly matched to one question word in q, either in its original, lowercase or lemma form.
  - **Token features**: $f_{token}(\hat{p_i}) = (POS(\hat{p_i}), NER(\hat{p_i}), TF(\hat{p_i}))$ Part of speech tagging, named entity recognition and term frequency features were used.
  - **Aligned question embedding**: $f_{align}(pi) = \sum_j a_{i,j} E(q_j)$, where the attention score $a_{i,j}$ captures the similarity betweenp_i and each question words q_j. $a_{i,j}$ is computed by the dot products between nonlinear mappings of word embeddings.
- **Question encoding**: The question encoding is simpler, which has another recurrent neural network on top of the word embeddings of q_i and combine the resulting hidden units into one single vector.

$$b_j = \frac{exp(w.q_j)}{\sum_j exp(w.q_j)}$$

- **Prediction**: Two classifiers independently for predicting the two ends of the span. Concretely, a bilinear term to capture the similarity between p_i and q and compute the probabilities of each token being start and end as:

$$P_{start}(i) \propto exp(p_i W_s q)$$
$$P_{end}(i) \propto exp(p_i W_e q)$$

3 Approach

3.1 Analysis of Different Models

We analyzed several models to determine their performance on the normal development set as well as the adversarial development set, and determine reasons behind the results based on the architecture of each model. We ran the models discussed in the previous section. Our results are discussed below:

Rank (F1-Adv)	Model	EM (Normal)	F1 (Normal)	EM (Adv)	F1 (Adv)
1	DrQA	69.38	78.90	47.78	55.44
2	DocumentQA	71.59	80.7	43.67	51.42
3	BiDAF	67.60	77.28	41.48	49.72
4	DocumentQA with ELMo	70.84	79.80	41.32	48.94

We ranked these models based on their EM and F1 scores when validated on the adversarial SQuAD dataset. We chose EM and F1 scores as an evaluation metric to concur with existing research in this area.

DrQA obtained the best results out of these models. We did a more detailed analysis of the architecture to determine the reason for the same, as well as to see if we could improve its performance. This study is detailed later on in the report.

DocumentQA performed quite well on the adversarial dataset. We believe that this is due to the question self-attention that the model implements: by incorporating this, the model places more importance on the question, which is an advantage against adversarial examples, which are not as closely related to the question as the true answer span.

BiDAF also performed quite well on the adversarial validation set. We believe that this is due to the attention mechanisms implemented in the architecture. However, these mechanisms did not put as much importance on the question, which could be why the BiDAF performance was not as high as DocumentQA.

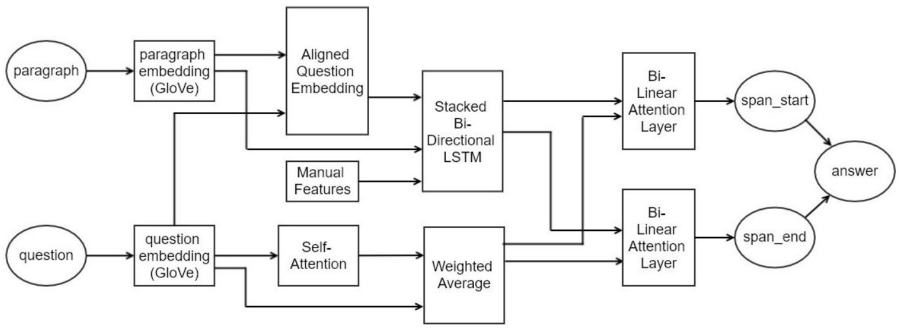
DocumentQA with ELMo surprisingly did not do as well on the datasets: ELMo usually improves model. We believe that since ELMo is a deep contextualized word representation, when the attention was calculated between the context words and the question words, most of the contribution of the attention scores were from the adversarially generated sentence in the paragraph. Hence the model makes an incorrect prediction.

Of these models, we decided to proceed with the one with the highest F1 on the adversarial development set, i.e. DrQA. We studied the architecture to determine the reason behind this, as well as to see whether we could make changes to the model to improve this performance further. We were able to improve the performance of this model, as discussed later in this report.

4 Experiments

4.1 Description

The model we focused our efforts on is DrQA. We observed from our experiments that this model did the best on adversarial examples, and wanted to determine why this occurred. We studied the architecture of the model, as depicted below:


paragraph
paragraph embedding (GloVe)
Aligned Question Embedding
Stacked Bi-Directional LSTM
Manual Features
Bi-Linear Attention Layer
span_start
answer
Bi-Linear Attention Layer
span_end
question
question embedding (GloVe)
Self-Attention
Weighted Average

We performed an ablative analysis of the architecture, removing various features to determine the relative importance of each one, with respect to the impact on the model's performance on both the normal as well as adversarial development sets. We determined that the "manual features" implemented in DrQA were the driving cause behind DrQA's high performance. These are simple word-level features that are concatenated to the other features and fed into the LSTM, and are discussed in detail below.

The word level features we determined to be the most important based on our ablative analysis are:

1. Context word in question (in cased form, uncased form or lemma form)
2. Term frequency (TF) of the context word

We realized that it is quite intuitive to realize why these features improve performance on adversarial examples: in adversarial examples, it is less likely that a context word would be in the question; similarly, the adversarial sentences contain swapped-out words that would not have as high a TF as other words originally belonging to the context.

Based on this analysis, we proposed other modifications to the DrQA model with the goal of improving performance on adversarial examples. In the Results section of this report, we show that these modifications actually improved model performance on normal examples as well.

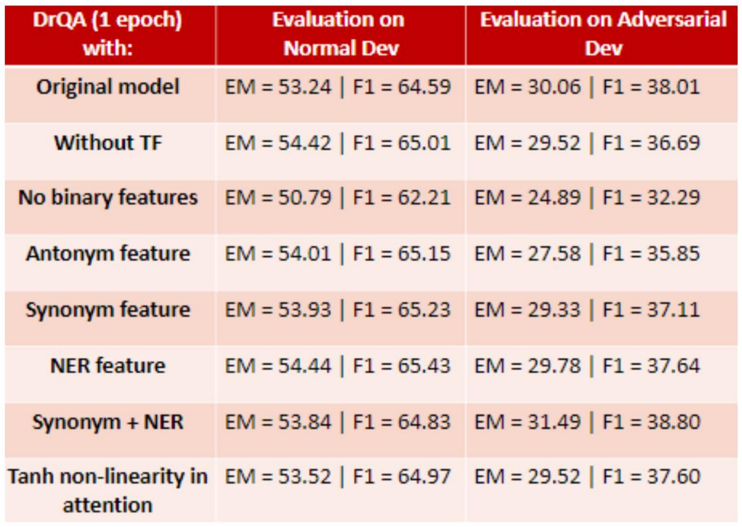
The proposed modifications are:

1. Adding new binary features:
   (a) Context word is an antonym of a word in the question in WordNet
   (b) Context word is a synonym of a word in the question in WordNet
   (c) Context word is a Named Entity and is NOT in the question
2. Switched the non-linearity from ReLU to tanh in the attention layer

The reason we proposed these modifications is primarily based on the intuition behind how adversarial examples are generated: Jia et al arrive at an adversarial sentence to append to the paragraph by switching a question word with its antonym and by replacing Named Entities with other named entities of the same type close to the original one in the GloVe vector space. Hence, we hypothesized that adding the features described above would improve the model's robustness against such scenarios.

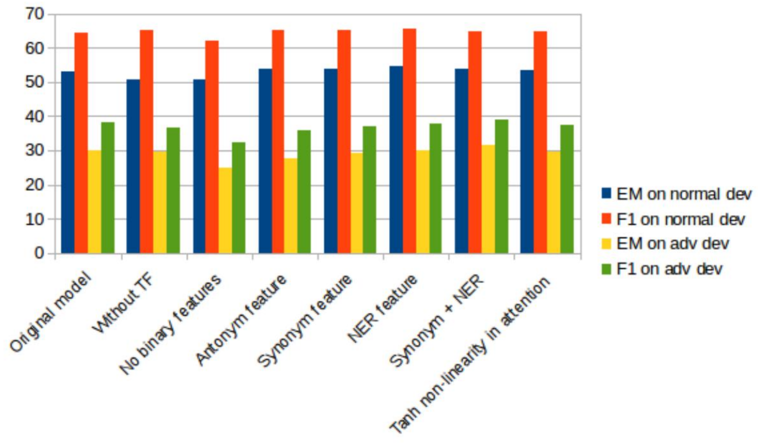
4.2 Results

We ran DrQA after implementing these features, separately and in combinations, and arrived at the following results:



DrQA (1 epoch) with:	Evaluation on Normal Dev	Evaluation on Adversarial Dev
Original model	EM = 53.24 | F1 = 64.59	EM = 30.06 | F1 = 38.01
Without TF	EM = 54.42 | F1 = 65.01	EM = 29.52 | F1 = 36.69
No binary features	EM = 50.79 | F1 = 62.21	EM = 24.89 | F1 = 32.29
Antonym feature	EM = 54.01 | F1 = 65.15	EM = 27.58 | F1 = 35.85
Synonym feature	EM = 53.93 | F1 = 65.23	EM = 29.33 | F1 = 37.11
NER feature	EM = 54.44 | F1 = 65.43	EM = 29.78 | F1 = 37.64
Synonym + NER	EM = 53.84 | F1 = 64.83	EM = 31.49 | F1 = 38.80
Tanh non-linearity in attention	EM = 53.52 | F1 = 64.97	EM = 29.52 | F1 = 37.60

This table has also been represented in the form of a graph:

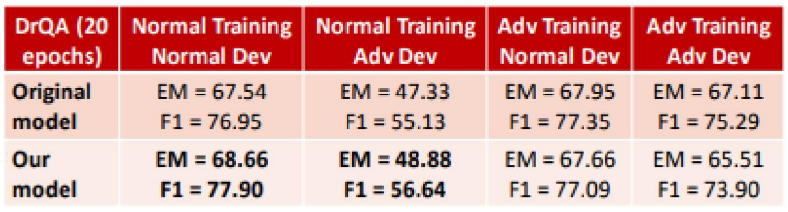


4.3 Inferences

We made the following inferences based on our findings. As expected, including the NER, TF and Synonym features improved model performance. This was expected: as discussed earlier in the report, the reason for including the NER and TF features was in order to tackle the intuition behind the distracting sentences having named entities not in the question, and infrequent in the document. The synonym feature also improved performance, as the questions posed in SQuAD usually contain a synonym of a word the answer span. This was of help in a more general sense, i.e. not with respect to adversarial examples specifically. Changing the non-linearity from ReLU to tanh allowed bounding of gradients.

4.4 Evaluation

In order to evaluate, we compared performance of our best model (obtained from above) against the baseline model proposed by [1] trained on both the original SQuAD dataset and the adversarially perturbed SQuAD dataset[4]. We chose EM and F1 as our metrics, similar to other research in the area.



DrQA (20 epochs)	Normal Training Normal Dev	Normal Training Adv Dev	Adv Training Normal Dev	Adv Training Adv Dev

Original model	EM = 67.54
	F1 = 76.95
		EM = 47.33
		F1 = 55.13
			EM = 67.95
			F1 = 77.35
				EM = 67.11
				F1 = 75.29

Our model	**EM = 68.66**
	F1 = 77.90
		EM = 48.88
		F1 = 56.64
			EM = 67.66
			F1 = 77.09
				EM = 65.51
				F1 = 73.90

Original SQuAD dataset training:

We notice that the features which were incorporated additionally into DrQA to combat the effect of the adversarial examples not only improved the F1 and EM score from 55.13 and 47.33 to 56.64 and 48.88 on the adversarial development evaluation dataset, but also improved the F1 and EM score from 76.95 and 67.54 to 77.90 and 68.66 on the normal development evaluation dataset. This shows that the features we added to improve model robustness against adversarial examples improved model performance on the normal validation set as well: many of the features we added (for example: synonym) were intuitive in the context of question answering in general, not just adversarial question answering.

Adversarial SQuAD dataset training: We also trained our model and the baseline model[1] on the adversarially perturbed SQuAD dataset released[4]. This was done to validate the model stability. As expected, we notice that the performance on the adversarial development dataset evaluated using these models increased as compared to the models trained on the original SQuAD dataset. We also notice that when these adversarially trained models were evaluated on the normal SQuAD development dataset, the F1 and EM scores dropped very little, which implies that the performance of the model on normal examples wasn't affected significantly by training the model on adversarial examples.

5 Conclusion and Future Work

Recent work[4] has shown that reading comprehension models suffer greatly from model overstability, and show dramatic decreases in performance when faced with adversarial examples that target the same. Our goal in this project was to compare the performance of several existing reading comprehension models on the normal SQuAD validation set as well as the adversarial validation set, determine the architectural reasoning behind these values, and use our findings to improve upon the current best model.

In accordance with this aim, we studied DrQA[1], a model which (to the best of our knowledge), currently has the highest performance on adversarial SQuAD. We studied its architecture and determined that its manual features and attention mechanism were the cause behind its high performance, and made changes to the model by incorporating several features and making modifications to the attention mechanism. We saw that this did indeed improve model performance, on both the adversarial validation set as well as the normal validation set - implying that the model robustness was improved not only for adversarial examples, but for normal examples as well.

We also retrained the models on adversarial examples, and obtained an increase in performance on the adversarial validation set (as expected), and an insignificant drop in performance on the normal validation set (due to improved model robustness). We are happy to report an increase in the current state of the art in model performance on adversarial validation set, both with and without adversarial training.

For future work, we would like to determine whether the features we incorporated that improved DrQA performance would improve the performance of other models as well: in particular, we are interested in FusionNet[3] and BiDAF[6]. We also plan to determine whether more architectural changes to the DrQA model, such as changing the attention mechanisms in a more substantial way, would improve performance further.

Acknowledgements

We would like to thank Robin Jia for his advice and insights regarding this project, as well as Abigail See, for her encouragement and guidance.

References

[1] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

[2] Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*, 2017.

[3] Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. *arXiv preprint arXiv:1711.07341*, 2017.

[4] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017.

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

[6] Ramon Tuason, Daniel Grazian, and Genki Kondo. Bidaf model for question answering.