

BiDAF Question Answering with Character Embedding, Self-Attention, and Weighted Loss

Stanford CS224N Default IID SQuAD Project; No collaborators; No mentor; Not sharing

Stephen Ren
Department of Computer Science
Stanford University
`rensteph@stanford.edu`

March 17, 2021

Abstract

Machine question answering remains a central problem in natural language processing. In this work, we build upon the default bidirectional attention flow model and explore the effect of adding character embeddings, self-attention, and a weighted loss function compared with the baseline. While character embeddings and self-attention have been demonstrated to improve the performance of language models, the motivation for a weighted loss function comes from the nature of the SQuAD dataset itself. We note that about half of the samples of the SQuAD dataset have no-answer, and is thus denoted by a start and end-pointer value of zero. Because the problem is effectively being treated as a classification problem (where the pointer locations are the classes to be predicted), this results in a ground truth distribution that is heavily skewed toward start and end-pointer class 0. To address this imbalance, we also propose the use of a weighted loss function, which down-weights no-answer examples, discouraging the model from simply guessing no-answer as a default choice. With a combined model, we achieve 62.11 EM and 65.54 F1 on the test set. We discover that a great deal of the error of the model comes from false-positives, and over-reliance on token matching.

1 Introduction

A key task for measuring machine-understanding of language is question answering, where given a passage text (referred to as the context) and a question (referred to as the query), a language model must find the answer to the question within the passage if it exists. Recent advances in deep learning have pushed the performance of such models, often employing the use of recurrent-neural

networks and the attention mechanism. One very successful architecture is the bidirectional attention-flow model (BiDAF^[1]), which is used as the baseline for this work.

However, our baseline BiDAF model does not incorporate character-level information as input, and so it loses out on valuable information regarding the structure of words, and also struggles with out-of-context words. Thus, we experiment with ways to include sub-word embeddings into our model. We also improve the attention mechanism by including self-attention, inspired by Microsoft’s R-Net^[2]. We also note that the task of question answering is proposed under the framework of classification, where the model must choose the indices (classes) of the start and end pointers. Because a significant portion of the SQuAD dataset has no answer, this makes the target distribution for start and end-pointers heavily weighted toward class zero (representing an empty string) compared to the other possible indices of the passage. We see this during training when the first few epochs actually show a dip in performance, as the model begins to stray away from relying on outputting no-answer. Thus, we also propose a weighted loss function to offset such an imbalance, which, when combined with a model that was trained without a weighted loss function. We demonstrate success in creating a model which performs well above the baseline for the task of question answering on the SQuAD dataset.

2 Related Work

In 2016, Rajpurkar et. al introduced the Stanford Question Answering Dataset (SQuAD)^[3] comprising of over 100,000 samples question-context passages for use toward the goal of machine reading comprehension. Each sample consists of a question (also referred to as query) text about a piece of context text (also referred to as a passage), and the ground-truth target is a span of text within the context that answers the question. The first dataset of its quality and scale, SQuAD opened the door to train and evaluate many state-of-the art question answering models, and we employ the same dataset in this project.

Recurrent Neural Networks and Attention

Recurrent neural networks (RNN) and long-short term memory (LSTM)^[4] models were among the most successful deep-learning language models for tasks ranging from machine translation to text comprehension. These models encode source and target sequences token by token into a hidden state, which is then decoded by another network (often another RNN) to complete a downstream task. While recurrent neural networks proved quite successful in language tasks, a large bottleneck of such a system was the fact the only information passed from the encoder to the decoder was the final hidden state, and so all of the information of a piece of text had to have been encoded in that final state. The introduction of the attention mechanism addressed this issue and is what really propelled the performance of such networks further, and subsequently formed the basis of state-of-the art transformer models, which are based purely on attention and convolution. Attention allows for the model to "look back" on the source

text and compare hidden states (often with a dot product) to see which context hidden state is most relevant to the current question, and use the information encoded at that timestep.

BiDAF The bidirectional attention-flow (BiDAF) model for question and answering is a modification of the attention mechanism commonly used with LSTM models, introduced in 2018 by Seo et. al. The main improvement is that the BiDAF Attention model computes the attention between both question-to-context as well as context-to-question, instead of just going one way, hence the name 'bidirectional', and thus information 'flows' between the query and context at each timestep. The baseline used in this work is a simple implementation of the BiDAF model. In the original BiDAF paper, the authors also used features of varying granularity (i.e. word embeddings, sub-word embeddings), but our baseline uses only the base word embedding.

R-Net In Microsoft's recent work with R-net, the authors introduced the idea of self-attention, in which a piece of context is attended with itself in addition to being attended with the query, hence the name self-attention. They found that this kind of attention helps the model gain more understanding of the overall context and results in less information loss. The authors of the paper found that this change (along with some gated attention-based mechanisms they also introduced) led to state-of-the art performance on both the SQuAD dataset and the MS-MACRO dataset, which is a question-answering dataset introduced by Microsoft.

Sub-Word Embeddings While word-embeddings such as GloVe^[7] are a natural way to encode input text, sub-wording embeddings^[6] have also been demonstrated to improve model performance. These embeddings allow for the model to learn a more granular representation of words, and also allows the model to better handle out-of-context words.

Retinanet / Focal Loss Finally, deep learning has also met resounding success with computer vision and object detection in images, and many of the techniques employed could also be applicable across domains. In 2017, Facebook research developed retinanet^[5] - a single-stage object detector that employs **focal loss** during training. Whereas two-stage object detectors first scan through an image to generate high-likelihood bounding boxes before passing into the classifier, single-stage object detectors generate and classify candidate bounding boxes across an image in parallel. Though this cuts down on computation, it also means many low-likelihood bounding boxes (such as background) contribute to the overall loss. In a two-stage detector, low-likelihood bounding boxes would have never made it to the classification stage. Because of this, these low-likelihood bounding boxes are very easily classified as background by the single-stage detector, which overwhelms the true-positives when evaluating loss. Lin et. al addressed this issue by using by down-weighting high-confidence detections, a mechanism which they dubbed focal-loss. Because the SQuAD dataset heavily favors the no-answer index (0) compared to any of the other candidate index, the same principle may apply here as well.

3 Approach

Baseline Architecture

The baseline model is the BiDAF model with word embeddings, which was already present with the starter code and configurations that came with the default project.

Character Embedding

The first improvement on the baseline was a change from word embedding layer in favor of character-level embedding. A sequence is represented as $X_i \in R^{l \times m \times c}$, where l is the max passage length (in words), m is the max word length, and c is the character embedding size. We flatten the last two dimensions into $m \cdot c$ before applying the projection layer. When masking over the data, we only mask entire words that are pad characters, but pad characters trailing after words are kept (this is done so that packing and unpacking is more streamlined with PyTorch). In a further iteration, we also concatenated the character embedding with the word embedding representations along the hidden dimension before feeding it further downstream in the model, which further improved the performance of the model. I implemented this code myself.

Self-Attention

To improve the attention mechanism even further, we incorporate the self-attention mechanism and described in R-Net to the model, where the context hidden state attends to itself. However, because the paper used a different overall architecture than BiDAF, there was some ambiguity that arose regarding the implementation details. One such ambiguity was whether the self-attention should be computed within the BiDAF layer, or if it should be a subsequent layer that takes as input the attention output from the BiDAF layer. As such, we experimented with both methods.

In the first method (referred to as Integrated Self-Attention), we modified the BiDAF layer to attend the context to itself using the same match multiplication it used between question and context. Let $c_1, c_2 \dots c_N \in R^{2H}$ represent the context hidden states, with N being the context length and H being a single-direction hidden state, and let $a_1, a_2, \dots a_N \in R^{2H}$ represent the context-to-question attention, and $b_1, b_2 \dots b_M \in R^{2H}$ represent the question-to-context attention with M being the question length. Then the output g_i of the original BiDAF attention layer is

$$g_i = [c_i; a_i; c_i \circ a_i; c_i \circ b_i] \in R^{8H} \quad (1)$$

where $;$ represents concatenation and $\circ$ is element-wise multiplication.

In the integrated self-attention, the modified BiDAF layer outputs

$$g_i = [c_i; a_i; c_i \circ a_i; c_i \circ b_i; k_i; c_i \circ k_i] \in R^{12H} \quad (2)$$

where k_i is the content-to-context self attention (computed in the same way as context to query or query to context attention, but in this case query is the same as context).

In the second approach (referred to as Subsequent Self-Attention), we kept the original BiDAF attention layer the same, but then the second layer would be another BiDAF attention layer, and would take in the a_i from the first layer as the input for both query and context. Let j_i be the self-attention between a_i and itself. The output of the second layer would then be

$$g'_i = [a_i; j_i; a_i \circ j_i; a_i \circ j_i] \in R^{8H} \quad (3)$$

the outputs of these two layers would then be concatenated barring redundant features, to get

$$g''_i = [c_i; a_i; c_i \circ a_i; c_i \circ b_i; j_i; a_i \circ j_i] \in R^{12H} \quad (4)$$

I implemented this code myself.

Focal / Weighted Loss

There are many examples in the SQuAD dataset where the answer is simply not found in the context, and thus target is simply no answer. Because of this, the model very quickly learns to simply guess no-answer as a default behavior. This problem is analogous to many 'easy negatives' for dense object detection in computer vision, where background proposals are very easily classified as not having an object. In Retinanet^[1], Lin et al addressed this problem using focal loss - a simple change to the loss mechanism to down the weight of high confidence scores, thus preventing easy-negatives from overwhelming the loss function, making training faster and more effective.

$$FL = (1 - p_t)^\gamma nLL \quad (5)$$

where p_t is the predicted probability of the correct class (pointer location in our case). The hyperparameter γ represents the strength of the modification. I modified the implementation of focal loss by mbsariyildiz^[8].

However, as the project progressed, focal loss proved to not be particularly useful, and so we realized the main issue was one of class imbalance, rather than overconfident predictions. Thus, we also ran experiments where we down-weighted the loss contributions from examples where the end pointer was 0. Initially we used $1/\ln(D/N)$ as the strength of the weight, where D was the length of the dataset and N is the number of classes (possible pointer locations), but this led to sub-par results. We then simply used a hyperparameter of $\frac{1}{3}$. Though this model did not perform quite as well as the regular weighted loss, it did achieve slightly better AvNA scores than the other models, which was worth reporting.

4 Experiments

Data and Objective We use the SQuAD dataset for the task of question answering. In particular, given a question and a piece of passage text referred to as the context, the goal is to pick out the section of the context that contains the answer for the question.

Evaluation Method We evaluate on the SQuAD dataset dev split outlined in the default project. We use the F1 and exact-match scores, as well as AvNA (answer vs no-answer). We also report the test set results, but these are limited as we are only allowed three submissions.

Experimental Details

Baseline: Same learning-rate and configurations that came with the default code.
Character Embedding: Using a character embedding with embed-size 32. Other hyperparameters were the same as the baseline.

Character Embedding with word embedding: Same configurations as the *Character Embedding* experiment, but the word-level and character-level embedding for each token were concatenated together along the hidden dimension.

Character Embedding + Word Embedding with Focal Loss: Same configurations as the *Character Embedding + Word embedding* experiment, but this time the objective function is focal loss instead of straightforward negative log-likelihood, with $\gamma = 2$.

Character + Word Embedding, Self-attention Version 1 (Integrated): Same configurations as the *Character + Word Embedding* experiment, but this model also integrated self-attention into the BiDAF attention layer.

Character + Word Embedding, Self-attention Version 2 (Separate): Same configurations as the *Character + Word Embedding* experiment, but this model had a subsequent self-attention layer that takes as input the C2Q attention output from the previous attention layer.

Character + Word Embedding, Self Attn V2 Weighted: Same configurations as the *Character + Word Embedding, Self Att V2 (Separate)* experiment, but had the loss from pointer index 0 down-weighted by a factor of 1/3

5 Results

The first 3 columns are for the best recorded dev set performance during training, while the last two are for the test set.

	AvNA	EM	F1	Test EM	Test F1
Baseline	67.65	57.4	60.64	-	-
Char-Embedding	69.99	60.38	63.67	-	-
Char+Word Embedding	69.79	61.13	64.16	-	-
Char+Word Emb +Focal Loss	69.16	60.36	63.59	-	-
Char+Word Emb +Self Attention V1	71.37	61.64	65.09	62.11	65.65
Char+Word Emb +Self Attention V2	71.5	<u>62.43</u>	<u>65.56</u>	60.85	64.26
Char+Word+Self Att V2, Weighted	<u>71.69</u>	60.44	64.11	59.04	63.03

We find that adding a character embedding results in a notable improvement

over the baseline, achieving a 61.13 EM score and 64.46 F1 score compared to 56.8 and 60.64, respectively, for the baseline. The integrated self-attention layer (V1) further improved performance, for a best performing model of **62.43** EM and **65.56** F1 on the test set. Interestingly, the subsequent self-attention layer (V2) did better on the dev set at its best step, although further on in training it overfit a bit and fell behind V1. Adding focal loss did not seem to impact the performance too much, in fact there was a slight decrease overall, but still well above the baseline. Incorporating focal loss into the models did not lead to a notable improvement in performance. The most likely reason for this is probably due to the fact focal loss mainly addresses high-confidence predictions, which is not necessarily the case with no-answer passages during training. Down-weighting non-answer samples did not quite help overall performance either, but did lead to a slightly higher AvNA score on the dev set. Training takes about 40% longer per epoch compared to the baseline when using character embeddings, and about 50% longer compared to the baseline when adding in both character embeddings and self-attention.

6 Analysis

In examining the qualitative results from the model, one can broadly characterize the errors into three categories: False positive, false negative, and wrong match. A false positive is when the passage did not contain an answer to the question, but the model gives an answer. A false-negative is when the question did have an answer in the passage but the model gave no answer. Finally, a wrong match is when the answer exists in the context but the model gave a wrong answer (that was not empty). Examining the way each of these errors arise gives great insight into how the model is coming up with its answers. Overall, we find the model to be over-reliant on finding close matches of the query within the context, and is not capturing the full of effect of antonyms or mismatched objects/subjects between the query and context.

This is most apparent when studying the false-positives. In general, the model struggles a lot with 'trick' questions, where a subject or object within the question is deliberately changed. For example:

step 2,657,684

- **Question:** Who proposed that water displaced through the projectile's path carries the projectile to its target?
- **Context:** Aristotle provided a philosophical discussion of the concept of a force as an integral part of Aristotelian cosmology. In Aristotle's view, the terrestrial sphere contained four elements that come to rest at different 'natural places' therein. Aristotle believed that motionless objects on Earth, those composed mostly of the elements earth and water, to be in their natural place on the ground and that they will stay that way if left alone. He distinguished between the innate tendency of objects to find their 'natural place' (e.g., for heavy bodies to fall), which led to 'natural motion', and unnatural or forced motion, which required continued application of a force. This theory, based on the everyday experience of how objects move, such as the constant application of a force needed to keep a cart moving, had conceptual trouble accounting for the behavior of projectiles, such as the flight of arrows. The place where the archer moves the projectile was at the start of the flight, and while the projectile sailed through the air, no discernible efficient cause acts on it. Aristotle was aware of this problem and proposed that the air displaced through the projectile's path carries the projectile to its target. This explanation demands a continuum like air for change of place in general.
- **Answer:** N/A
- **Prediction:** Aristotle

The question is very similar to this line in the passage "Aristotle... proposed that *air* displaced through the projectile's path carries the projectile toward's its target". In the query, the question almost word for word matches this sentence, except *air* was replaced with *water*. Though this simple change in just one word renders the entire question nonsensical in relation to the passage, the model will still have a very high attention score (as most of the words are the same), leading

it to the wrong answer of Aristotle, when it should have returned no answer.

A similar phenomenon occurs when examining many wrong match samples as well. For example, take the following sample:

step 3,051,939

- **Question:** Which country's cars became more highly sought after as they were more fuel efficient?
- **Context:** The crisis reduced the demand for large cars. Japanese imports, primarily the Toyota Corona, the Toyota Corolla, the Datsun B210, the Datsun 510, the Honda Civic, the Mitsubishi Galant (a captive import from Chrysler sold as the Dodge Colt), the Subaru DL, and later the Honda Accord all had four cylinder engines that were more fuel efficient than the typical American V8 and six cylinder engines. Japanese imports became mass-market leaders with unibody construction and front-wheel drive, which became de facto standards.
- **Answer:** Japanese imports
- **Prediction:** Honda Accord

Of the nine different car models mentioned in the passage, the model went with *Honda Accord*. This is again likely due to text-matching between the context and query - namely, the phrase *fuel efficient*, which was in the query, is closest to the mention of *Honda Accord* within the passage. This again hints that the attention mechanism is dominating the overall decision-making process of the model (which should not be too surprising, as state-of-the-art models have run well on attention alone).

I found that the majority of errors are from trick question false positives, and so perhaps a good next step in improving the model would be to find a better way to address those examples specifically.

7 Conclusion

We find that character embeddings and self-attention both improve the overall performance of QA models. Based on my analysis of the model outputs, I believe one major struggle these models face is over-reliance on attention, allowing small changes to the query to confuse the model. We think a model would benefit from being able to first distinguish if an answer was indeed within the passage. One such solution would be a two stage model, where the first model discerns if the answer is in the context or not, and only pass the problem onto the second stage if the answer is in the passage. The second stage would be the actual question answering, conditioned on the fact that the answer exists in the passage. Both of these problems are simpler than trying to both at the same time, and could make for interesting future work.

Code can be found at <https://github.com/stephrenny/squad> (better-commented than code submitted to Gradescope)

References

- [1] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. arXiv preprint arXiv:1611.01603, 2016.
- [2] <https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf>

- [3] Rajpurkar, Pranav, et al. “SQuAD: 100,000+ Questions for Machine Comprehension of Text.” Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, doi:10.18653/v1/d16-1264.
- [4] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8),1735–1780
- [5] Lin, Tsung-Yi, et al. “Focal Loss for Dense Object Detection.” 2017 IEEE International Conference on Computer Vision (ICCV), 2017, doi:10.1109/iccv.2017.324.
- [6] Bojanowski, Piotr, et al. “Enriching Word Vectors with Subword Information.” *Transactions of the Association for Computational Linguistics*, vol. 5, 2017, pp. 135–146., doi:10.1162/tacl_a_00051.
- [7] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.
- [8] <https://github.com/mbsariyildiz/focal-loss.pytorch/blob/master/focalloss.py>