R-Net and Friends

Stanford CS224N Default Project

Amrita Palaparthi
Department of Computer Science
Stanford University
amritapv@stanford.edu

Ani Vegesana
Department of Computer Science
Stanford University
avegesan@stanford.edu

Megan Worrel
Department of Computer Science
Stanford University
mworrel@stanford.edu

Abstract

In our work, we examine how character-level embeddings, additional input features, and a modified attention mechanism, coupled with hyperparameter tuning, can enhance performance on the task of Question Answering. We perform these experiments on the SQuAD 2.0 dataset, building off of a baseline model of Bidirectional Attention Flow with two new input features from DrQA [1] and attention mechanisms described in R-Net[2]. Our findings indicate that our model boosts performance above the baseline, more effectively handling out-of-vocabulary words. Our highest-performing single model achieved scores of $F1 = 65.68$ and $EM = 62.48$ on the SQuAD 2.0 dev set, and our best ensemble model reached dev set scores of $(\mathbf{F1 = 68.32}, \mathbf{EM = 65.74})$ and scores of $(\mathbf{F1 = 65.67}, \mathbf{EM = 63.13})$ on the SQuAD 2.0 test set.

1 Key Information

- Mentor: Allan Zhou (ayz@stanford.edu)
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Reading Comprehension, a type of Question Answering (QA), is the task of locating the answer to a question in a context passage where both the question and context passage are presented as blocks of text and the answer is a phrase contained in the question verbatim. Models solving this task are used in web search engines to synthesize answers to user provided questions from crawlable web pages. It is also critical to determining how well models can understand and draw information from a piece of text. This is a difficult problem because the answer to a question is not always phrased exactly as it appears in the passage text or could be absent from the context entirely. Current methods for solving the QA task on the SQuAD dataset, such as BiDAF, DrQA, and R-Net, perform well with EM scores of 73.3, 70.0, and 76.7 and F1 scores of 81.1, 79.0, 83.7 and [3, 1, 2]. However, these three models have not been tested on the harder SQuAD 2.0, which includes unanswerable questions, by the original authors. This leaves room to adapt these models to increase speed and performance on SQuAD 2.0. These models also have disparate architectural components which have not previously been combined in a single model.

In our paper, we create a new hybrid architecture that starts with a BiDAF model, adds the exact match (EM) and aligned question embedding (AQE) features from DrQA, and adds the gated and

self-attention components of R-Net. We then perform an ablation study on our hybrid architecture and analyze how important each component is to the model. Finally, we use the "most confident model decides" strategy to ensemble different hybrid models to achieve better performance than any individual model could on its own. Our method resulted in EM=63.128 and F1=65.674 on the SQuAD 2.0 test dataset.

We were able to verify the result in the BiDAF paper that the addition of character embedding improves the performance of the model still holds on the SQuAD 2.0 dataset. We were also able to verify that EM doesn't yield additional meaningful information to the model if AQE features are present as well, paralleling DrQA. Unfortunately, we were unable to reproduce R-Net's SQuAD 1.1 results on the SQuAD 2.0 dataset, only reaching F1=61.75 and EM=64.36 without BiDAF attention features, but we were able to demonstrate R-Net's result that the self-attention block is able to improve the performance of the model in situations where context information that substantiates the answer is relatively far away from the answer.

3 Related Work

3.1 DrQA

Chen et al.'s DrQA model introduces the use of additional input features to boost performance on open-domain question answering. Rather than solely using word-level and character-level embeddings for both context and question inputs, DrQA leverage three types of additional features as part of their context encoding process: Exact Match, Aligned Question Embedding, and Token Features.

Described in detail in 4.2, Exact Match and Aligned Question Embedding (AQE) aim to improve performance by providing a measure of similarity between words in the question and context. These features serve similar roles; while Exact Match notes the presence of a single word in both the context and the question, AQE serves as a softer, more flexible indicator between similar question and context words. Token features, on the other hand, were manually selected and reflected characteristics of words such as part of speech and term frequency. In our approach, we choose to implement only Exact Match and AQE features, as Chen et al. found through ablation analysis that they contributed more to the F1 results achieved by DrQA. Since the removal of token features did not substantially decrease F1 score in the presence of Exact Match and AQE, we postulate that the latter two features would be more meaningful to include in our paragraph encoding method.

3.2 R-Net

Beyond feature engineering, we augment our baseline model to include two additional forms of attention: gated attention and self-matching attention (self attention). Gated attention functions similarly to our BiDAF baseline's context-to-question attention mechanism, generating a representation of a context paragraph that is aware of relevant matching information in the question. We also add R-Net's self attention mechanism to our implementation. This component of the model, which has no analogue in BiDAF, bolsters performance by generating a refined, self-aligned representation of the context paragraph. We included this attention (described further in 4.3) in our approach since the results of R-Net indicate that it enabled the model to perform more effectively on questions where a large amount of passage context is required to determine the correct answer.

4 Approach

4.1 Character-Level Embeddings

Our first modification to the BiDAF baseline model was the addition of character-level embeddings to supplement lookup-based word embeddings. We experimented with passing our character-level embeddings through a 1D convolution – as is done in the original BiDAF implementation – and through a bidirectional RNN encoder. Character-level embeddings utilize the morphology of words and enable better generalization to words not contained in the provided vocabulary.

4.2 Feature Engineering

After successfully implementing character-level embeddings, we then computed the exact match f_{exact_match} and aligned question embedding f_{align} features from DrQA. We chose to implement all three exact match features from the DrQA model: exact match, lowercase form, and lemmatized form. For exact match, we return a binary value indicating whether each context word is found directly in the question. For lowercase and lemmatized form, we convert all words in the context and question to lowercase and corresponding lemma respectively before performing the same check as exact match. These input features can be summarized by $f_{exact\_match}(c_i) = \mathbb{I}(c_i \in q)$ for each context word c_i and question q.

Next, we computed f_{align} for each context word. The aligned question embedding feature plays a complementary role to f_{exact_match} in model performance. f_{align} is an attention mechanism between context and question embeddings, providing a soft alignment between context and question words. As in the DrQA model, we compute aligned question embeddings according to $f_{align}(p_i) = \sum_j a_{i,j}\mathbf{E}(q_j)$ where

$$a_{i,j} = \frac{\exp(\alpha(\mathbf{E}(c_i)) \cdot \alpha(\mathbf{E}(q_j)))}{\sum_{j'} \exp(\alpha(\mathbf{E}(c_i)) \cdot \alpha(\mathbf{E}(q_{j'})))} = \text{softmax}_{:,j}\left[\alpha(\mathbf{E}(c))\alpha(\mathbf{E}(q))^\top\right]$$

for a dense layer with ReLU nonlinearity α. All additional features were appended to our context embeddings.

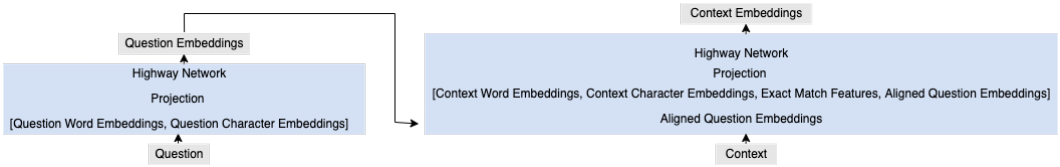

Figure 1: Embedding layer with feature engineering and separate Highway Networks

As seen in Figure 1, the final context embeddings were of the form $\mathbf{E}(c_i)) = [$ word embedding, character-level embeddings, EM, AQE $]$ and the final question embeddings were of the form $\mathbf{E}(q_i)) = [$ word embedding, character-level embeddings $]$. In the embedding layer, the baseline model passes context and question embeddings through both a shared projection and a Highway Network. After appending our additional features, we modified the baseline embedding layer to pass context and question embeddings through separate projections with identical construction. For the Highway Network, we chose to run two experiments – applying a shared and separate Highway Networks to the context and question embeddings respectively. The separate Highway Networks were also constructed identically.

4.3 R-Net and Extensions

After performing feature engineering, we began implementing the R-Net gated and self attention to replace and complement BiDAF attention, described in detail in [3].

First, we substituted the BiDAF attention layer in the baseline model with the sequential computation of R-Net gated and self attention. Our initial implementation followed the R-Net specifications exactly. Gated attention is a soft alignment between context representations $\{u_t^C\}_1^m$ and question representations $\{u_t^Q\}_1^m$ resulting in $\{v_t^C\}_1^m$, paralleling context-to-question attention from the BiDAF model. Define additive attention as follows:

$$s_j^t(x_j^t) = v^T \tanh(x_j^t)$$

$$a_i^t(x_j^t) = \frac{\exp(s_i^t(x_j^t))}{\sum_{j=1}^m \exp(s_j^t(x_j^t))}$$

$$\ell_t(x_j^t) = \sum_{i=1}^m a_i^t(x_j^t) u_i^Q$$

Using the definition for additive attention, the equation for gated attention is $d_t = \ell_t(W_u^Q u_j^Q + W_u^C u_j^C + W_v^C v_{t-1}^C)$ where v_t^C is the RNN output at each time step

$$v_t^C = \text{RNN}(v_{t-1}^C, [u_t^C; d_t]^*)$$

An additional gate is applied to $[u_t^C; d_t]$ to compute the RNN input $[u_t^C; d_t]^*$. This gate is computed on the context representation u_t^C and context-question attention d_t. It is defined

$$g_t = \text{sigmoid}(W_g[u_t^C; d_t])$$
$$[u_t^C, d_t]* = g_t \odot [u_t^C; d_t]$$

g_t is used to capture the idea that select context words are more relevant to answering the question than other context words. The gate effectively directs the RNN to focus on those relevant context words.

The attention output d_t and LSTM output v_t^C are computed through mutual recursion. Our implementation for gated attention required explicitly providing the hidden state at each step of the LSTM, requiring manual iteration, and ultimately causing long training times. For more efficient training, we then implemented a non-recursive version of R-Net's gated attention, computing the attention in its entirety before passing the result into the LSTM. Thus, our s_j^t was now defined $d_t = \ell_t(W_u^Q u_j^Q + W_u^C u_j^C)$ with the rest of the gated attention computation remaining the same. This approach leveraged faster matrix multiplication and gpu processing, achieving much faster training times.

The second component of R-Net's attention, and the most significant contribution of the model, is self attention – a soft alignment from question-aware context representation v_t^C back to itself. Although theoretically the LSTM should be able to build up a hidden representation of the entire context, due to the small finite length of the hidden state vector, the LSTM is prone to catastrophic forgetting of far away contextual information [4]. While gated attention makes the context representation question-aware, the self attention mechanism makes the context representation more robust to catastrophic forgetting. Self attention h_t^C is computed similarly to gated attention v_t^C where self attention is defined

$$h_t^C = \text{BiRNN}(h_{t-1}^C, [v_t^C; d_t]^*)$$

with $d_t = \ell_t(W_v^C v_j^C + W_v^{C'} v_t^C)$ and $[v_t^C; d_t]^*$ computed with an identical gate as in gated attention. Input to the BiDAF modeling layer is thus of the form $g_i = [v_i^C; h_i^C] \in \mathbb{R}^{4H}$ for each context location $i \in \{1, ..., N\}$ where H is the hidden size from the embedding layer projection.

Hoping to better understand the comparative strengths of BiDAF and R-Net, our next substitution was to remove the modeling and output layers from the baseline model and use the R-Net output layer instead. The R-Net output layer is computed as an unidirectional RNN with attention (similar to gated attention) where the timestep is not the word position, but is instead just two timesteps (start and end token indices of the answer) and the input to the attention is the output of self-attention, identical to [2]. This produces two predicted log-probabilities, corresponding to the start and end indices of the answer.

Finally, we decided to experiment with a parallel model architecture, computing both BiDAF and R-Net attention in parallel and concatenating the results for input into the BiDAF modeling and output layers in order to create a more expressive model architecture that can represent models of both families.

Shown in Figure 2, embeddings are fed directly into the computation for BiDAF attention and R-Net gated and self attention.

Further, since R-Net gated and self attention are no longer computed sequentially, we modified the mechanism for computing self attention to be computed directly on the context representations $\{u_t^C\}_1^m$ (seen in Figure 2). s_j^t in self attention is now defined

$$s_j^t = v^T \tanh(W_v^C u_j^C + W_v^{C'} u_t^C)$$

with the rest of the self attention computation remaining the same. Input to the BiDAF modeling layer is thus of the form $g_i = [c_i; a_i; c_i \circ a_i; c_i \circ b_i; v_i^C; h_i^C] \in \mathbb{R}^{12H}$ for each context location $i \in \{1, ..., N\}$.

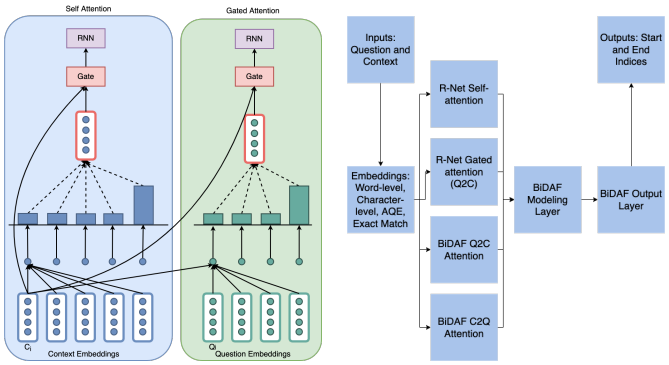

Figure 2: "Parallel" model architecture

5 Experiments

5.1 Data

Our model used GLoVe 300-dimensional word embeddings of 2.2M words and 64-dimension character embeddings for 1373 characters (a combination of Latin letters with and without diacritics, numerical digits, punctuation, and characters used in the native names of foreign places or people) pretrained on the 840B token Common Crawl dataset as well as the EM and AQE features from Chen et al.'s DrQA. We used the Spacy small English pipeline to lemmatize words in the EM calculation.

The model takes in dictionary IDs for each word and character (corresponding to the ID of the words and characters in the dictionary used to train GLoVe) in the context and passage texts and the EM and AQE features as input. At each word position in the passage, the model output a logit, a predicted log confidence (i.e. probability), for the answer starting at that position as well as a logit for it ending at that position. The model was trained on the SQuAD 2.0 dataset which has 100,000 answerable and 50,000 unanswerable questions. Finding the highest valued start and end logit for any given model is equivalent to finding the predicted start and end token indices for the answer. If the model predicts that there is no answer to the question, it will return logits such that both the start and stop indices are 0. The answer is the substring of passage text between the start and end indices following the Python convention (the substring is inclusive of the start token, but exclusive of the end token). The logits are used directly in ensembling where we always defer to the model with the highest ending logit on any given question.

5.2 Evaluation method

We use the two official metrics used by the SQuAD dataset: EM (exact match) and F1. EM is the fraction of the predicted answers in which the start and stop indices match the ground truth answer exactly (including the case where there is no answer.) This a harsher metric than F1, which is based on the number of overlapping word locations (not unique words) in the predicted and correct substring and doesn't give a zero score for partially correct answers. The F1 score for a single answer is the harmonic mean of the precision, the fraction of ground truth word locations that are in the predicted answer, and the recall, the fraction of predicted word locations that are in the ground truth answer. The F1 score for a model is the mean over the entire dataset of the F1 scores of the model's predictions.

5.3 Experimental details

Prior to hyperparameter tuning, all experiments were run with a learning rate of 1, hidden size of 100, batch size of 64, and dropout probability of 0.1, as used in the BiDAF baseline. Batch size was reduced to 16 when training R-net models in order to adhere to memory constraints.

We built upon our model incrementally, choosing configurations of each component that yielded the best performance. For instance, after implementing character-level embeddings, all subsequent exper-

Model	F1	EM	AvNA
Baseline	60.65	57.37	67.82
+ Character-Level Embeddings	**63.76**	60.14	70.29
+ EM	64.34	60.85	71.40
+ AQE	64.52	61.35	72.76
+ EM & AQE (shared highway)	65.04	61.75	72.68
+ EM & AQE (separate highways)	**65.16**	61.00	71.92
Mutually Recursive R-Net Attn + R-Net Output	64.36	61.75	70.63
Non-Recursive R-Net Attn + R-Net Output	61.61	51.18	70.36
Non-Recursive R-Net Attn + BiDAF Modeling/Output	62.32	58.86	69.03
Non-Recursive R-Net Gated Attn + BiDaf Modeling/Output	63.68	59.90	71.09
R-Net Self Attn + BiDaf Attn/Modeling/Output ($\star$)	63.35	60.94	71.33
Parallel Attn + BiDaf Attn/Modeling/Output (†)	**65.02**	61.20	71.22
$\star$; LR Decay (factor = 0.5, patience = 2) + Dropout = 0.30	**65.68**	62.48	72.69
†; LR Decay (factor = 0.5, patience = 2)	65.30	61.93	71.95
9 Model Ensemble	**68.32**	65.74	74.32
12 Model Ensemble	68.08	65.47	74.11

Table 1: Performance measured by the F1 and EM scores on the dev set. The AvNA column is the fraction of questions for which the model responded "No answer" when the . The parameters that we chose for evaluation for each model were the ones with the highest F1 score on the dev set.

iments were run with character-level embeddings included. After implementing feature engineering, we chose to use both input features and separate highways for following experiments.

We noticed a substantial increasing in training time after implementing the recursive R-net gated and self attention layers. The resulting model took 15.38 hours to train for 1 million steps, dwarfing the corresponding 0.8 hours for our BiDAF-based models. To improve training efficiency, we shifted to a non-recursive implementation and additionally utilized GRU rather than LSTM for the RNN components of our model, reducing training time to 1.1 hours.

5.4 Results

Table 1 displays our experiment results on the development set of the SQUAD 2.0 dataset. The addition of character-level embeddings improved F1 score by about 3 points. This was the greatest improvement in F1 score of all changes made to the baseline. Improvement was expected as the learned character-level embeddings allow the model to learn subwords and out-of-vocabulary words and even perform better on infrequently seen vocabulary words. There was a negligible change in scores when the character embedding 1D convolution was replaced with a bidirectional RNN (A).

The addition of EM features increased F1 by about 0.6, while AQE features increased F1 by about 0.8. We were expecting similar increases in F1 score from each feature individually as EM and AQE compute similar metrics of similarity between context and question words. We also see that implementing both features together leads to a further increase in F1 by about 0.5. This is consistent with the DrQA observation that the two features serve "similar but complementary role[s]."

As shown in Figure A, the additional input features gave the model a boost early on, but leveled out to match the performance of the model without feature engineering as training progressed. Intuitively, the additional input features enabled the model to paraphrase context and drew attention to potentially relevant context words; however, this benefit waned with further training of the original model. Overall, this indicates that additional input features convey information that can be beneficial to speed up training and boost model accuracy slightly, but can likely also be learned by a sophisticated model on its own with enough training.

None of our R-Net experiments were able to surpass BiDAF performance in terms of F1 score *prior to hyperparameter tuning*. This was a surprising result as R-Net reported higher F1 scores on the SQUAD 1.0 development and test sets than the origial BiDAF model. A possible explanation for this is that R-Net uses a variation of additive attention which is less expressive than the BiDAF

multiplicative attention. Additionally, this could simply be due to a lack of appropriate hyperparameter selection, as implementing learning rate decay on the R-net parallel implementation eventually boosted performance to beyond the F1 score achieved by BiDAF with feature engineering alone. In addition, the mutually recursive R-Net implementation performed about three F1 points better than the non-recursive version. This was expected because the mutually recursive implementation allows the RNN and attention mechanism in gated attention to learn simultaneously, passing information to each other at each time step. If we had the resources to train all our R-Net experiments with the recursive implementation, we are confident that the R-Net results would have exceeded the BiDAF results by a greater amount.

5.4.1 Hyperparameter Tuning

After creating our parallel model architecture using the hyperparameters used by BiDaF, we ran more experiments with tweaked hyperparameters to attempt to improve performance. We lacked the resources and time to run a thorough random or grid search on the hyperparameters, but we ran the eight configurations in Table A. Based on that small set of experiments, we know that a larger hidden size improves model performance and a different initial learning rate and changing the dropout parameter only make relatively minor improvements within about one F1 point. Learning rate decay yields substantial improvements, but did not provide a uniform boost in F1 score across all models. This means that the correct scheduling hyperparameters are highly dependent on the architecture. In experiments, we used a learning rate that multiplies the current learning rate by a factor every time that the model's F1 score has not improved for n evaluation steps, where n is called the patience. Further fine-tuning of the LR scheduler and dropout are needed to reach optimal performance of the architecture.

5.4.2 Ensembling

As mentioned in section 5.1, the ensembling method chooses the model with the highest ending logit. Due to time constraints, we ensembled models from all twelve experiments that performed better than our character level embeddings model in our final evaluation as opposed to retraining our best model from scratch. By doing this, however, we still hoped and observed that slight differences in model architectures would gave better collective understanding and still improve performance above any single model. We also noticed that our simple ensembling produces poorer results that what we would expect if it has too many models because it suffers from maximization bias, i.e. models that produce confidently incorrect answers easily skew (bias) the answer of the whole ensemble due to the maximum used in computation [5], suggesting that there is room for improvement. Ultimately, we chose to evaluate our model on the complete ensemble instead of a subset of the highest scoring 9 because it was unclear whether choosing a subset of the models would lead to overfitting. Our final 12 model ensemble scored $(\mathbf{F1} = \mathbf{65.67}, \mathbf{EM} = \mathbf{63.13})$ on the SQuAD 2.0 test set, scoring 17th on the non-PCE leaderboard without use of transformers.

6 Analysis

Out-of-Vocabulary Words

Question: Who designed the garden for the University Library?

Context: Another important library – the University Library, founded in 1816, is home to over two million items. The building was designed by architects Marek Budzyński and Zbigniew Badowski and opened on 15 December 1999. It is surrounded by green. The University Library garden, designed by Irena Bajerska, was opened on 12 June 2002. It is one of the largest and most beautiful roof gardens in Europe with an area of more than 10,000 m2 (107,639.10 sq ft), and plants covering 5,111 m2 (55,014.35 sq ft). As the university garden it is open to the public every day.

Answer: Irena Bajerska

Prediction (baseline): Marek Budzyński and Zbigniew Badowski

Prediction (char embeddings): Irena Bajerska

The above question requires understanding of out-of-vocabulary words to answer correctly. The context contains multiple proper nouns, in addition to specific numbers, that were not encountered during training. With just the addition of character embeddings, which give knowledge at a deeper level of granularity, the new model is able to predict the right answer when the baseline was not.

Long-range Dependencies

Question: Disruptions in sleep can lead to increase in what chronic conditions?

Context: When suffering from sleep deprivation, active immunizations may have a diminished effect and may result in lower antibody production, and a lower immune response, than would be noted in a well-rested individual. Additionally, proteins such as NFIL3, which have been shown to be closely intertwined with both T-cell differentiation and our circadian rhythms, can be affected through the disturbance of natural light and dark cycles through instances of sleep deprivation, shift work, etc. As a result, these disruptions can lead to an increase in chronic conditions such as heart disease, chronic pain, and asthma.

Answer: heart disease, chronic pain, and asthma

Prediction (gated): heart disease

Prediction (self-attention): heart disease, chronic pain, and asthma

In this question, the sentence containing the answer includes only the word "disruptions," while the question itself refers to sleep disruptions. In order to infer that these disruptions refer to sleep, knowledge of the previous sentence referring to "sleep deprivations" is required. The R-Net model including self attention is able to capture this long-range dependency by placing emphasis on the entirety of the text. However, the R-Net model that solely uses gated attention without self attention lacks this capability and therefore only predicts "heart disease" rather than all affected conditions.

Question Word	what	which	who	where	why	when	how	whose	other
Prevalence	3651	218	630	256	87	439	555	28	87
BiDAF									
TNR	0.7691	0.7273	0.8192	0.8191	0.7250	0.8291	0.8111	0.8571	0.7742
FPR	0.2309	0.2727	0.1808	0.1809	0.2750	0.1709	0.1889	0.1429	0.2258
TPR	0.6784	0.8000	0.6797	0.6914	0.7234	0.7438	0.6361	0.6429	0.5714
FNR	0.3216	0.2000	0.3203	0.3086	0.2766	0.2562	0.3639	0.3571	0.4286
R-Net									
TNR	0.7076	0.6667	0.7341	0.7652	0.6471	0.8034	0.7266	0.8125	0.6316
FPR	0.2924	0.3333	0.2659	0.2348	0.3529	0.1966	0.2734	0.1875	0.3684
TPR	0.6966	0.7869	0.6856	0.7234	0.7500	0.7701	0.6354	0.6667	0.5102
FNR	0.3034	0.2131	0.3144	0.2766	0.2500	0.2299	0.3646	0.3333	0.4898

Table 2: True and False Positive and Negative Rates of BiDAF and RNet Prediction of Answer vs No Answer Depending on the Question Type

In an effort to understand why the R-Net model performed worse than the BiDAF model before hyperparameter tuning, we looked at each model's true negative rate TNR, false positive rate FPR, etc. when predicting if each question type was answerable. Shown in Figure 2, R-Net had a worse TNR (higher FPR) across the unanswerable questions for all nine question types, indicating that R-Net consistently incorrectly predicted answers for unanswerable questions. This indicates that the sophisticated, self-attending R-Net model may be learning nonexistent patterns. Further, R-Net appears to have performed especially poorly on "how" and "why" questions which likely require more advanced comprehension than the "when" questions on which its performance was stronger.

7 Conclusion

Implementing R-Net attention mechanisms in conjunction with DrQA's additional input features results in a substantial increase in performance over our baseline model on Question Answering for SQuAD 2.0. Each of the four main components of our approach - DrQA additional input features, R-Net attention mechanisms, hyperparameter tuning, and ensembling - built upon one another to provide an incremental increase in F1 score, which, through our work, we were able to quantify. We noticed that numbers are not in our vocabulary, so creating an "is numeric" binary input feature might help boost performance. Given enough compute resources, we would have trained recursive R-Net with each of the combinations in Table 1 to get a fairer estimate of performance. We would also run R-Net without the DrQA input features to better segregate the improvements made by each. Future work would also implement a bias aware sampling technique [6] or clustering [5] to choose the most likely answer less naively to avoid maximization error in ensembling. Finally, in future work we would have done a random search to tune the hyperparameters. We didn't have the resources or time to run enough experiments where we would be likely to get close to the best hyperparameters of our model, so we just selected several hand-fabricated settings, leading to suboptimal performance.

References

[1] Jason Wetson Antoine Bordes Danqi Chen, Adam Fisch. Reading wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*, 2017.

[2] Microsoft Research Asia Natural Language Computing Group. R-net: Machine reading comprehension with self-matching networks. In *Association for Computational Linguistics (ACL)*, 2017.

[3] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[4] Gaurav Arora, Afshin Rahimi, and Timothy Baldwin. Does an LSTM forget more than a CNN? an empirical study of catastrophic forgetting in NLP. In *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association*, pages 77–86, Sydney, Australia, 4–6 December 2019. Australasian Language Technology Association.

[5] Charu C. Aggarwal and Saket Sathe. Theoretical foundations and algorithms for outlier ensembles. *SIGKDD Explor. Newsl.*, 17(1):24–47, sep 2015.

[6] Fábris Kossoski and Mario Barbatti. Nuclear ensemble approach with importance sampling. *Journal of Chemical Theory and Computation*, 14(6):3173–3183, 2018. PMID: 29694040.

A Appendix

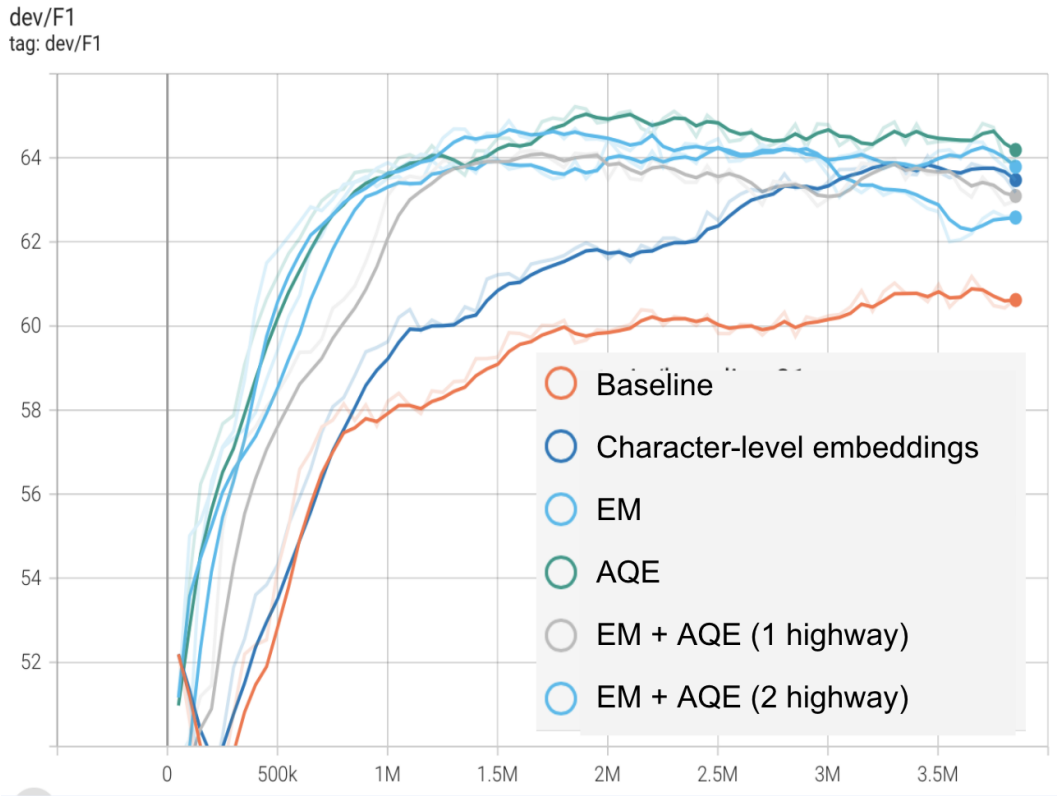

Figure A.1: Feature engineering ablation for BiDAF

Model	F1	EM	AvNA
Baseline	60.65	57.37	67.82
+ Character-Level Embeddings	**63.76**	60.14	70.29
+ RNN Character-Level Embeddings	62.52	59.00	70.34
+ EM	64.34	60.85	71.40
+ AQE	64.52	61.35	72.76
+ EM & AQE (shared highway)	65.04	61.75	72.68
+ EM & AQE (separate highways)	**65.16**	61.00	71.92
Mutually Recursive R-Net Attn + R-Net Output	64.36	61.75	70.63
Non-Recursive R-Net Attn + R-Net Output	61.61	51.18	70.36
Non-Recursive R-Net Attn + BiDAF Modeling/Output	62.32	58.86	69.03
Non-Recursive R-Net Gated Attn + BiDaf Modeling/Output	63.68	59.90	71.09
R-Net Self Attn + BiDaf Attn/Modeling/Output ($\star$)	63.35	60.94	71.33
Parallel Attn + BiDaf Attn/Modeling/Output (†)	**65.02**	61.20	71.22
$\star$; LR = 1.0 + Hidden Size = 75 (R-Net recommendation)	58.82	55.49	67.06
$\star$; LR = 1.0 + Hidden Size = 100	63.19	60.73	70.63
$\star$; LR = 0.5 + Hidden Size = 75	59.54	56.44	67.10
$\star$; LR = 0.5 + Hidden Size = 100 (default)	**63.35**	60.94	71.33
$\star$; LR Decay (factor = 0.1, patience = 10)	65.02	62.10	72.25
$\star$; LR Decay (factor = 0.5, patience = 2) [Default Dropout of 0.2]	65.39	62.34	72.31
$\star$; LR Decay (factor = 0.5, patience = 2) + Dropout = 0.30	**65.68**	62.48	72.69
$\star$; LR Decay (factor = 0.5, patience = 2) + Dropout = 0.52	64.06	61.00	70.81
†; LR Decay (factor = 0.5, patience = 2)	65.30	61.93	71.95
2 Model Ensemble	65.98	63.10	72.32
5 Model Ensemble	67.60	64.42	73.82
8 Model Ensemble	68.22	65.54	74.31
9 Model Ensemble	**68.32**	65.74	74.32
10 Model Ensemble	68.30	65.70	74.24
11 Model Ensemble	68.21	65.52	74.16
12 Model Ensemble	68.08	65.47	74.11

Table A.1: More complete version of Table 1