Orchestra: An Ensemble Approach to SQuAD 2.0

Aaron Levett
Department of Computer Science, Biology
Stanford University
Stanford, CA 94305
alevett@stanford.edu

Taide Ding
Department of Computer Science
Stanford University
Stanford, CA 94305
tding@stanford.edu

Abstract

The Stanford Question Answering Dataset (SQuAD) 2.0 task tests both a system's ability to answer reading comprehension questions and to determine when a question cannot be answered given the provided passage. In this report, we outline improvements we have made to a Bidirectional Attention Flow (BiDAF) model baseline, which was provided to us as described in the original BiDAF paper (Seo et al. 2016 [1]), though with the character embedding layer omitted. We developed several distinct improvements that resulted in increased EM and F1 scores relative to the baseline. Our first improvement used word embedding vectors that concatenate a given word's GLoVe and word2vec embeddings for use in the word embedding layer. Our second improvement added a character embedding layer to the provided baseline. Further improvement resulted from replacing Bidirectional LSTMs with Bidirectional GRUs in selected layers of the BiDAF model. Though experiments with variable learning rates proved inconclusive, an ensemble model we call ORCHESTRA consisting of four of our highest performing models yielded our best performance on the CS224N SQuAD 2.0 test set, with an F1 score of 67.00 and an EM score of 63.86. As of 3:45pm on Sunday, March 17, 2019, our model ranked second highest for both F1 and EM scores out of 42 submissions on the test non-PCE leaderboard.

1 Introduction

Question answering and reading comprehension are crucial areas of research in Natural Language Processing: a system that performs well in question answering and reading comprehension tasks must have the ability to parse natural language patterns in order to extract knowledge about the real world. Automated means of question answering have wide-reaching applications both impressive (e.g. IBM's Watson) and practical (e.g. digital voice assistants and search engines).

The Stanford Question Answering Dataset (SQuAD) 2.0 task evaluates a system's ability to answer reading comprehension questions given a context passage, as well as to determine when such a question cannot be answered given the context passage. Recent years have seen significant breakthroughs in performance on the SQuAD 2.0 dataset, exemplified by Devlin et al.'s Bidirectional Encoder Representations from Transformers (BERT) model, which uses pretrained contextual embeddings (PCEs) whose weights can be fine-tuned for different NLP tasks [2]. As such, almost all of the top 30 submissions on the global SQuAD 2.0 leaderboard make use of BERT in various configurations. However, there is still much to be learned from implementing and innovating upon non-PCE question answering models.

To this effect, we achieved improvements upon Seo et al.'s Bidirectional Attention Flow (BiDAF) model [1], provided as the baseline for the CS224N SQuAD 2.0 challenge with character embeddings omitted. We experimented with a variety of approaches, including implementing a character embedding layer, increasing the information contained in the word embeddings by using both GLoVe and

word2vec embeddings, as well as modifying BiDAF's architecture to consist of bidirectional GRUs instead of bidirectional LSTMs. We then attempted hyperparameter tuning, and finally experimented with ensembling of the different models generated through the experiments.

2 Related Work

The Bidirectional Attention Flow (BiDAF) model as described by its creators makes use of "a multi-stage hierarchical process that represents the context at different levels of granularity and uses bi-directional attention flow mechanism to obtain a query-aware context representation without early summarization" [1]. Unlike previous state-of-the-art question answering systems, which would often take the entire context paragraph and turn it into a single fixed-size vector via attention, BiDAF uses attention at each time step to help avoid premature context summarizing. Seo et al.'s original paper provides the authoritative description of BiDAF's innovative approaches [1].

Numerous examples in current literature propose extensions to BiDAF that can be successful. For instance, pair2vec capitalizes on relationships between neighboring words by creating pairwise embeddings that reflect contextual similarity of word pairs [3]. The authors of pair2vec add these embeddings in BiDAF's attention layer without modifiying the original model's word embeddings input layer. In addition, examples abound in neural network literature of ensemble models that combine the output of many neural networks trained on the same task in different ways (or even just with different random seeds) to improve the ensemble model's overall performance. Moreover, ensembling complementary models to mutually reinforce performance has been shown to be beneficial over indiscriminately ensembling all available models [4].

3 Approach

Our study focuses on augmenting the Bidirectional Attention Flow (BiDAF) model as described in Seo et al. 2016 [1] in order to improve its performance on the Stanford Question Answering Dataset (SQuAD) 2.0 task. We focused our experimentation on BiDAF in three chief directions:

1. Augmenting the representations for the input layer
2. Employing Gated Recurrent Units (GRUs) in place of the LSTMs used in the baseline BiDAF for different subsets of the encoder, modeling, and output layers
3. Ensembling to achieve better predictive performance than any one of the individual models

3.1 Baseline

As specified in the default project handout [5], the Bidirectional Attention Flow (BiDAF) model as described in Seo et al. 2016 [1] with character-level embeddings omitted was used as the baseline.

3.2 Input Layer Augmentations

3.2.1 word2vec + GLoVe

For the first augmentation to the input layer, we concatenated the GLoVe word embeddings for each word (as originally used in the baseline) with the word's word2vec representation, providing strictly greater semantic and contextual information than the baseline. The final dimension of the word embeddings used was 600 (300 from each of the two sources). word2vec embeddings were obtained from the Google News dataset[1]. This model will be subsequently referred to as **w2v+GLoVe**.

3.2.2 baseline + character embeddings

For the second type of augmentation to the input layer, we implemented a CNN to obtain character-level embeddings to feed into the contextual embedding layer along with the GLoVe word embeddings, allowing for two levels of granularity in capturing semantic information. The CNN had a kernel size of 5 with 64 in channels and the number of out channels equivalent to the word vector size (300 or 600

[1] `https://code.google.com/archive/p/word2vec/`

depending on whether we combined this approach with w2v+GLoVe described in 3.2.1). This model will be subsequently referred to as **CharEmb**. The model that implements both character embeddings and the concatenation of word2vec and GLoVe will be referred to as **w2v+GLoVe+CharEmb**.

3.3 Gated Recurrent Units (GRUs)

Introduced by Cho et al. in 2014, Gated Recurrent Units offer a simpler alternative to LSTMs by way of fewer parameters, including using a reset gate that combines the functionality of the LSTM's input and forget gates [6]. Since their introduction, practitioners have frequently found GRUs to be computationally more efficient than LSTMs, as well as performing better on smaller datasets.

The baseline BiDAF model employs bidirectional LSTMs in three layers: 1) in the Encoder Layer, LSTMs encode temporal dependencies between the timesteps of the embedding layer's output; 2) in the Modeling Layer, LSTMs integrate temporal information between context representations conditioned on the question; 3) the output of the Modeling Layer then passes through LSTMs in the Output Layer before computing the softmax [5]. Our study tested three models, each of which incrementally replaces LSTMs with GRUs in each layer of the CharEmb BiDAF model (3.2.2). The table below shows the various permutations trained in our experiments.

Model	Encoding	Modeling	Output
CharEmb (LSTM)	LSTM	LSTM	LSTM
CharEmb (1GRU)	LSTM	GRU	LSTM
CharEmb (2GRU)	GRU	GRU	LSTM
CharEmb (3GRU)	GRU	GRU	GRU

3.4 Ensembling

Ensembling combines the outputs of multiple models trained on the dataset to obtain a final prediction. The technique allows for multiple model hypotheses to be taken account, reducing the impact of overfitting by any one constituent model on the ensemble model's overall performance on test data. In our experiments, we ensembled the following combinations of models described in the above sections.

Constituent Model	E1	E2	E3	E4	E5	E6
w2v+GLoVe	+	+		+	+	+
CharEmb (LSTM)	+		+	+	+	+
CharEmb (1GRU)						+
CharEmb (2GRU)		+	+	+	+	+
CharEmb (3GRU)						+
w2v+GLoVe+CharEmb (LSTM)					+	+

For each ensemble model, we weighted each constituent model equally, i.e. we took the arithmetic mean of the log-probability of the start and end positions across the context as determined by each constituent model, and used the resulting final distribution to determine the answer span.

4 Experimental Methods

4.1 Data

The CS 224N Squad 2.0 dataset [5] was used for training, consisting of 129,941 training examples (context/question/answer), and 6078 development examples. All of these examples come from the official SQuAD 2.0 dataset, with half of the official dataset's dev set used as the CS 224N dev dataset.

4.2 Evaluation Method

F1 and EM metrics were used for evaluation. For examples that lack an answer, F1 and EM are defined as 1 if the model correctly predicts no answer, and 0 if the model predicts there to be an answer.

4.3 Experimental Details

For all experiments discussed below, we allowed training to occur for 30 epochs, with a fixed learning rate of 0.50 unless a variable learning rate is specified. Batch gradient descent with batch size of 64 was employed, and dropout probability was set at 0.2 for all experiments.

5 Results

5.1 Input Layer Augmentations

5.1.1 word2vec + GLoVe

Two different versions of w2v+GLoVe were run. In the first version, denoted "W2V OOV-NULL", tokens that were not in the word2vec vocabulary were treated in the same manner as out-of-vocabulary tokens were treated in the baseline code, i.e. the zero vector is used as the embedding. In the second version, denoted "W2V OOV-GLoVe", a token's corresponding GLoVe embedding was used for tokens that were not in the word2vec vocabulary. All other aspects of the model were carried over from the baseline. As the following charts show, W2V OOV-GLoVe performs sizeably better overall than the baseline and W2V OOV-NULL.

Figure 1: Tensorboard outputs for w2v+GLoVe experiments: **Orange** = Baseline; **Blue** = W2V OOV-GLoVe; **Red** = W2V OOV-NULL

Table 1: Out of Vocabulary w2v+GLoVe results: F1 and EM Scores for model epoch with Peak F1

Model	F1	EM
Baseline	61.28	57.87
W2V OOV-NULL	61.59	58.43
W2V OOV-GLoVe	62.92	59.75

All subsequent references to the **w2v+GLoVe** model will denote the W2V OOV-GLoVe model above.

5.1.2 Character Embeddings (CharEmb)

BiDAF with character embedding layer, denoted "CharEmb (LSTM)", was implemented as described in 1.3. Seeing significant improvements over the baseline, we took the more successful word embed-

ding concatenation approach (W2V OOV-GLoVe) described in 5.1.2 and used its associated word embeddings as input to the BiDAF implementation along with the added character embedding layer, running a combined model "**w2v+GLoVe+CharEmb (LSTM)**" to determine if both improvements to the model would result in even greater overall performance.

Figure 2 shows the Tensorboard output for the two runs with the character embedding added (plotted against baseline and w2v+GLoVe for reference). Table 2 displays F1 and EM scores for the best produced model during each of the runs.

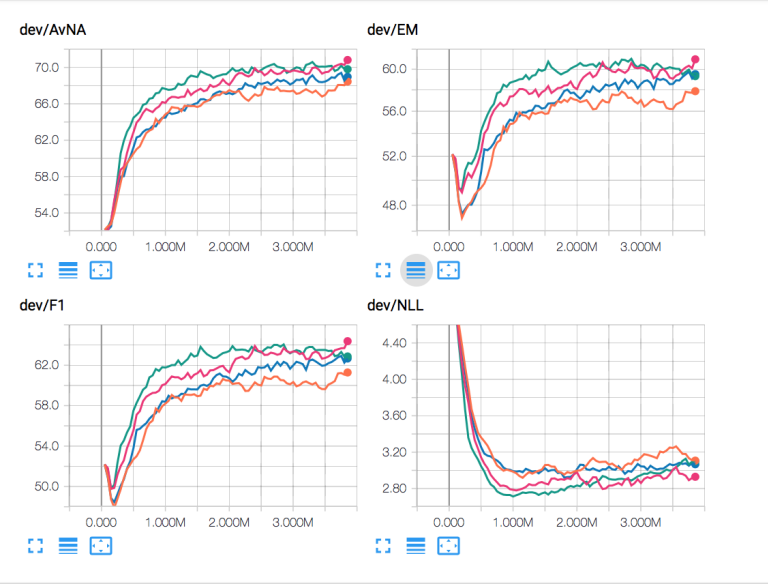

Figure 2: Character Embedding Experiment Results: **Orange** = Baseline; **Blue** = w2v+GLoVe; **Pink** = CharEmb (LSTM), **Green** = w2v+GLoVe+CharEmb (LSTM)

Table 2: F1 and EM Scores for model epochs with best F1

Model	F1	EM
Baseline	61.28	57.87
w2v+GLoVe	62.92	59.75
CharEmb (LSTM)	64.38	60.98
w2v+GLoVe+CharEmb (LSTM)	64.06	61.03

Interestingly, while both w2v+GLoVe and CharEmb improved performance over the baseline, combining the two improvements did not lead to substantial changes in F1 and EM performance beyond the single embedding models. However, initial performance gains are much quicker in the combined word and character embedding model, possibly due to the greater number of parameters being learned in the combined model compared to either of the models that used a single embedding format. However, the plateauing EM and F1 scores may indicate overfitting in the combined model; the combined model's training error trended slightly lower than that of any of the single embedding models across all epochs.

5.2 Implementing GRUs

Replacing bidirectional LSTMs with bidirectional GRUs led to comparable and somewhat better results on the dev set. That being said, no appreciable increase in convergence speed or computational efficiency was observed compared to LSTMs, belying the conventionally touted advantages of GRUs. Overall, CharEmb (2GRU), i.e. the model in which the Encoder and Modeling Layers were converted to GRUs but the Output layer remained as LSTMs, was able to outperform CharEmb (LSTM) as well as the other two GRU implementations.

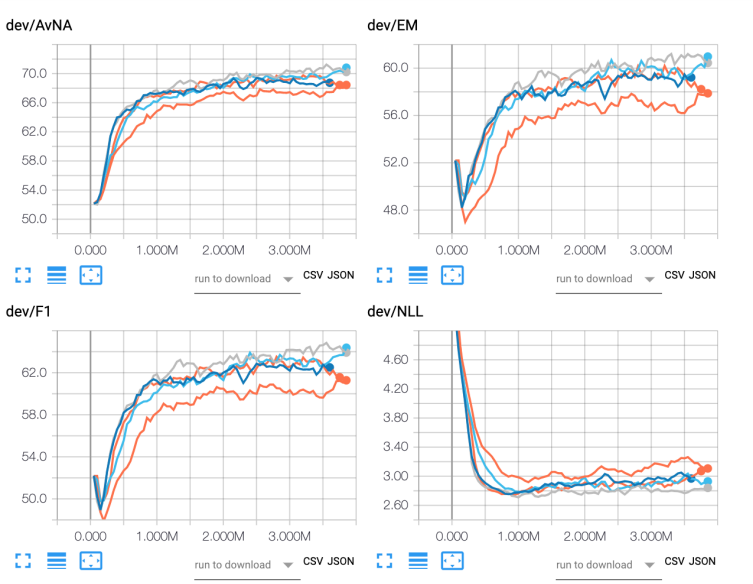

Figure 3: Tensorboard Output for GRU implementations: **Lower Orange** = Baseline; **Light Blue** = CharEmb (LSTM); **Upper Orange** = CharEmb (1GRU), i.e. GRU in Modeling Layer only; **Gray** = CharEmb (2GRU), i.e. GRU in Encoder and Modeling Layers; **Dark Blue** = CharEmb (3GRU), i.e. all LSTMs in model changed to GRUs.

Table 3: F1 and EM Scores for model epochs with best F1

Model	F1	EM
Baseline	61.28	57.87
CharEmb (LSTM)	64.38	60.98
CharEmb (1GRU)	63.52	60.16
CharEmb (2GRU)	62.87	59.62
CharEmb (3GRU)	64.84	61.20

5.3 Hyperparameter Tuning

We experimented with learning rate decay, but results were underwhelming compared to the corresponding models with constant learning rate, suggesting that the decaying learning rate caused the model to simply converge to suboptimal local minima. Experiments were terminated prematurely to save computing resources. See Figure 4 on the following page for details.

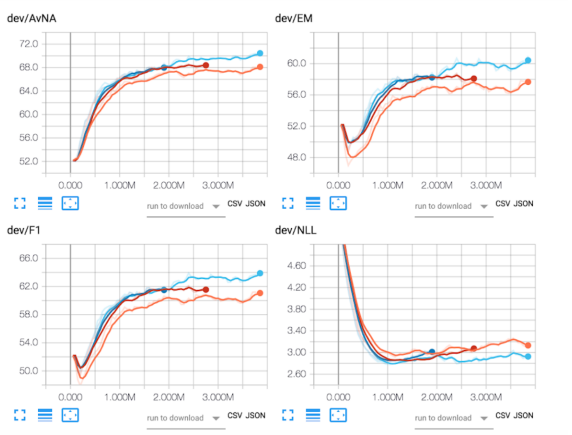

Figure 4: Tensorboard Output for learning rate decay experiments: **Lower Orange** = Baseline; **Light Blue** = CharEmb (LSTM) with baseline learning rate (LR = 0.5); **Dark Blue** = CharEmb (LSTM) with initial LR = 0.5 and decay rate per epoch = 0.95; **Dark Red** = CharEmb (LSTM) with constant learning rate (LR = 0.4)

5.4 Ensemble Models

Table 4: F1 and EM Scores for Ensemble Models: scores on test set were reported only for models submitted to leaderboard.

Constituent Model	E1	E2	E3	E4	E5	E6
w2v+GLoVe	+	+		+	+	+
CharEmb (LSTM)	+		+	+	+	+
CharEmb (1GRU)						+
CharEmb (2GRU)		+	+	+	+	+
CharEmb (3GRU)						+
w2v+GLoVe+CharEmb (LSTM)					+	+
Scores	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
F1 (Dev)	65.52	65.61	66.54	66.57	67.27	67.06
F1 (Test)			64.80	65.91	67.00	
EM (Dev)	62.43	62.64	63.32	63.54	64.24	64.17
EM (Test)			61.52	62.72	63.86	

Incorporating GRU models into the ensemble led to noticeably higher FM and EM scores than for LSTM-only ensembles, suggesting that the slightly different temporal dependencies encoded by the two types of recurrent units reinforce each other in the ensemble. The top performing model (E5), combining w2v+GLoVe, CharEmb (LSTM), CharEmb (2GRU), and the concatenated input model w2v+GLoVe+CharEmb (LSTM), was submitted to the Non-PCE Leaderboard as **ORCHESTRA**.

6 Analysis

For error analysis, we examined our final ORCHESTRA model's dev output for the first 35 examples. Out of the 35 examples, 19 were exactly correct, 23 were either exactly correct or almost correct (i.e. the same semantic meaning was captured, but the span differed slightly from the provided answers), and 12 were completely incorrect.

Out of the 16 that were not perfectly correct exact matches, we noticed a few distinct categories of errors:

1. **Correct Semantic Content, Span Differs from Provided Answers**
   Example Context (Truncated):
   `For a precise definition of what it means to solve a problem using a given amount of time and space, a computational model such as the deterministic Turing machine is used. The time required by a deterministic Turing machine M on input x is the total number of state transitions, or steps, the machine makes before it halts and outputs the answer...`
   Example Question:
   `The time required to output an answer on a deterministic Turing machine is expressed as what?`
   ORCHESTRA output: `the total number of state transitions`
   Closest Correct Answer: `the total number of state transitions, or steps`
   This error type is the most 'excusable' for the model, so to say, and ocurred a few times in the sampled responses. The difference between the closest 'correct' answer and the provided answer is insignificant, and it is conceivable that a human with knowledge about Turing machines would have provided this answer given the context.
2. **Answering Unanswerable 'When' Questions**
   Example Context (Truncated):
   `The war was fought primarily along the frontiers between New France and the British colonies, from Virginia in the South to Nova Scotia in the North...The dispute erupted into violence in the Battle of Jumonville Glen in May 1754, during which Virginia militiamen...`
   Example Question:
   `When did violence end in war?`
   ORCHESTRA output: `May 1754`
   Correct Answer: `N/A (Unanswerable)`
   This error occurred twice in the sampled incorrect answers. It is possible that our model performed well in learning what counts as a possible 'when' answer (i.e. dates and times), but not of learning the association between the time and the event in the context that the time refers to.

7 Conclusion

Overall, our experiments illuminated several methods to improve upon the baseline BiDAF model, including adding a character embedding layer, modifying the input word embeddings to include both GLoVe and word2vec embeddings, and replacing the model's bidirectional LSTM's with bidirectional GRUs. Ensembling the start and end word probabilities computed by multiple models with different augmentations in order to determine the answer span greatly increased our model's performance on the CS224N dev set, culminating in our ORCHESTRA model. ORCHESTRA performed almost equally well on the CS224N test set as on the dev set, achieving the second highest F1 and EM score out of 42 submissions on the CS224N non-PCE test leaderboard as of report submission time (3:45pm on Sunday, March 17, 2019).

8 Future work

While our existing developments and performance so far on the CS224N leaderboard are exciting, there is still much room for improvement. For example, hyperparameter tuning, specifically of learning rates, did not prove fruitful. In the future, implementing a cyclic learning rate scheduler may benefit our model's performance: under such a scheme, the learning rate would decrease over time until performance plateaus, at which point the learning rate would be reset to a higher value in order to leap to another part of the domain and explore for additional local optima, which can then be ensembled [7]. Additional hyperparameter tuning, particularly of dropout rates, optimizer types and layer sizes, may also benefit the model's performance. In particular, increasing the dropout probability could help mitigate the overfitting that could be inferred in some of our models. Finally, additional architectural changes such as the implementation of a self-attention layer may be fruitful.

References

[1] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. *Bidirectional attention flow for machine comprehension*. *arXiv preprint arXiv:1611.01603*, 2016.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding arXiv:1810.04805*, 2018.

[3] Mandar Joshi, Eunsol Choi, Omer Levy, Daniel S. Weld, Luke Zettlemoyer. *pair2vec: Compositional Word-Pair Embeddings for Cross-Sentence Inference*. *arXiv:1810.08854*, 2018.

[4] Zhi-Hua Zhou, Jianxin Wu, Wei Tang. *Ensembling neural networks: Many could be better than all*, 2002.

[5] CS224N Course Staff. *CS224N Default Final Project: Question Answering on SQuAD 2.0*. Last updated on February 28, 2019.

[6] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. *arXiv:1406.1078*, 2014.

[7] Gao Huang, Yixuan Li, and Geoff Pleiss. *Snapshot Ensembles: Train 1, Get M for Free*. International Conference on Learning Representations (ICLR), 2017.