Improving a QA System for SQuAD using Attention- and Augmentation-based Methods

Stanford CS224N Default Project

Amelia Woodward
Department of Computer Science
Stanford University
ameliawd@stanford.edu

Angela Zhao
Department of Computer Science
Stanford University
angezhao@stanford.edu

Stone Yang
Department of Computer Science
Stanford University
yywstone@stanford.edu

Abstract

Question-Answering (QA) systems have been incredibly useful because they can find answers within given text. The SQuAD 2.0 dataset [1] is one of the most highly used for training Question-Answering systems (QA) systems. Our research goal is to improve the baseline by exploring self-matching attention, coattention, character-level embeddings, and data augmentation techniques to improve on the baseline Bidirectional Attention Flow (BiDAF) model and produce a QA system that works well on SQuAD. We find that character embedding methods are the most powerful of explored techniques in improving QA performance.

1 Key Information to include

- Mentor: Grace Lam
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

QA systems are a significant area of NLP from both a research and practical standpoint. From a research perspective, QA systems serve as a measure for how well NLP systems can 'understand' text. From a practical perspective, they are useful for better understanding any piece of text and to extract answers from text. To improve the QA system for SQuAD, we start from a baseline Bidirectional Attention Flow (BiDAF) model [2]. We first explore changes to attention, both replacing BiDAF attention with coattention [3] and adding an R-NET style self-matching attention layer on top of BiDAF attention [4]. Since the baseline model only include word level embeddings, we also experimented with adding character level embeddings. Finally we explored data augmentation techniques including span corruption and back-translation [5]. Our results show significant improvement using character embeddings. While we saw some lift with individual strategies like self-matching attention and span corruption, these did not translate to high performance when combined with character embeddings. We discuss some of the reasons these results in our analysis.

3 Related Work

The baseline BiDAF model we use was introduced by Bidirectional attention flow for machine comprehension [2]. This model is limited in data pre-processing, embedding layer construction and attention layer construction, which inspired us to look into literature in those three areas. **Data augmentation**: we are inspired by attempts to make robust QA systems by span corruption [6] and back-translation [5]. Existing literature aims to improve QA models' out-of-domain performance, while our work uses similar techniques to create augmented training data. **Embedding layer**: The BiDAF paper [2] includes character-level embeddings. As we learnt in lectures, character-level embeddings allow us to condition on the internal structure of words, and better handle out-of-vocabulary words. **Attention layer**: The Dynamic Coattention Network is a high-performing SQuAD model [3]. We take inspiration from its Coattention Layer. R-Net achieved state-of-the-art results on SQuAD and MS-MARCO datasets [4]. The self-matching attention mechanism from the paper is one of the main drivers for the good results, which motivated us to add a self-matching attention layer.

4 Approach

The architecture of our neural model is as follows:

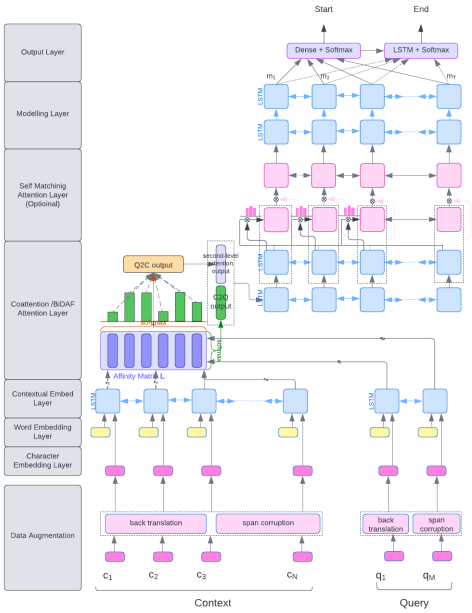

Figure 1: Overview of Model Used (larger version in Appendix)

There are three main improvement areas that we explore throughout the final project:
Data Augmentation We explored two areas of data augmentation.

- **Span Corruption**: We implement span corruption explored in the T5 paper [7]. We train the model on corrupted context only because we can better ensure that the answers remains valid. First, each example has a 30% chance of being chosen for a corruption attempt. We choose text span in the context to replace with OOV tokens. Then, if the text span that was corrupted contains the answer entirely, we replace the original answer indices with 0 to indicate that an answer no longer exists. If the corrupted span contains the answer only in part, we return the original example. This is because we want to ensure the model is training on valid answers, and it is difficult to tell if the partial answer remaining is still a valid answer.

- **Back Translation**: Paraphrasing by back-translation is another method of data augmentation to prevent the model from learning brittle correlations. Given an example x, we create an augmented example x' where both the query and the context have been first translated into German and then back into English. Additionally, we translate the given answer as well into German and then back to English using the same translation model. If there is no exact string answer in the context, we will employ the same heuristic in Longpre et al. (2019) [5]. We find the closest span of text using character-level 2-gram overlap between the paraphrased context and the original answer. This span of text will only be the new answer if their Jaccard similarity score passes a certain threshold. Otherwise, we will discard the back-translated query and context pair. Due to compute costs and runtime limitations, we chose to backtranslate only the queries of 300 data samples.

Embedding Layer
We added a character-level embedding layer on the BiDAF model. This layer maps each word to a vector space using character-level CNNs. Following Kim (2014) [8], we obtain the character level embedding of each word using a 2D Convolutional Neural Network. Characters are embedded into vectors, which can be considered as 1D inputs to the CNN, and whose size is the input channel size of the CNN. Then, the character embedding goes through a drop-out layer with drop-out probability 0.05 and a ReLU layer. The outputs are max-pooled over the entire width to obtain a fixed-size vector for each word.

Attention Layer
We implemented both co-attention and self-matching attention to replace the baseline attention layer.

- **Coattention** refers to attending over representations that are themselves attention outputs. We implement the Coattention Layer following Xiong (2016)'s approach for Dynamic Coattention Networks, using the following steps:
  1. Assuming we have context hidden states $c_1, \ldots, c_N \in R^l$ and question hidden states $q_1, \ldots, q_M \in R^l$, apply a linear layer with tanh nonlinearity to the question hidden states to obtain projected question hidden states $q'_1, \ldots, q'_M$.
  $$q'_j = tanh(\mathbf{W}\mathbf{q}_j + \mathbf{b}) \in R^l, \forall j \in \{1, \ldots, M\}$$
  2. Add sentinel vectors $c_\emptyset \in R^l$ and $q_\emptyset \in R^l$ to both the context and question states, and use the aggregate of these vectors to compute the affinity matrix $L \in R^{(N+1)\times(M+1)}$, where each entry is an affinity score: $L_{ij} = c_i^\top q'_j \in R$.
  3. Use the affinity matrix L to compute attention outputs for both directions. For C2Q Attention, obtain C2Q attention outputs a_i and for Q2C attention obtain outputs b_j:
  $$\alpha_i = \text{softmax}(L_{i,:}) \in R^{M+1}; a_i = \sum_{j=1}^{M+1} \alpha_j^i q'_j \in R^l;$$
  $$\beta^j = \text{softmax}(L_{i,j}) \in R^{N+1}; b_j = \sum_{i=1}^{N+1} \beta_j^i c'_i \in R^l$$
  4. Use the C2Q attention distributions α^i to take the weighted sums of the Q2C attention outputs b_j, which gives us the second-level attention outputs s_i:
  $$s_i = \sum_{j=1}^{M+1} a_j^i b_j \in R^l, \forall i \in \{1, \ldots, N\}$$
  5. Concatenate second-level attention outputs s_i with the first-level C2Q attention outputs a_i and feed the sequence through a bidirectional LSTM. The resulting hidden states are u_i: $\{u_1, \ldots, u_N\} = \text{biLSTM}(\{[s_1; a_1], \ldots, [s_N; a_N]\})$
- **Self-matching attention** means the hidden state $\mathbf{h_t}$ attends to all the previous hidden states so far $\mathbf{h_1}, \ldots, \mathbf{h_{t1}}$. We implement the Self-Matching Attention Layer from R-Net [4]. Through gated attention-based recurrent networks, question-aware passage representation $\{v_t^P\}_{t=1}^n$ is generated to pinpoint important parts in the passage. There are two problems with such representation. First, it has very limited knowledge of context. One answer

candidate is often oblivious to important cues in the passage outside its surrounding window. Second, there exists some sort of lexical or syntactic divergence between the question and passage in the majority of SQuAD dataset [9]. Passage context is necessary to infer the answer. To address this problem, self-matching attention directly matches the question-aware passage representation against itself. It dynamically collects evidence from the whole passage for words in passage and encodes the evidence relevant to the current passage word and its matching question information into the passage representation h_t^P:

$$h_t^P = BiRNN(h_{t-1}^P, [v_t^P, c_t])$$

where $c_t = att(v^P, v_t^P)$ is an attention-pooling vector of the whole passage (v^P):

$$s_j^t = v^\top tanh(W_v^P v_j^P + W_v^{\tilde{P}} v_t^P)$$

$$a_i^t = exp(s_i^t) / \sum_{j=1}^{n} exp(s_j^t)$$

$$c_t = \sum_{i=1}^{n} a_i^t v_i^P$$

An additional gate as in gated attention-based recurrent networks is applied to $[v_t^P, c_t]$ to adaptively control the input of RNN.
Self-matching extracts evidence from the whole passage according to the current passage word and question information.

5 Experiments

5.1 Data

We use the official SQuAD 2.0 dataset with three splits: train, dev and test as described in the default SQuAD handout. The three splits in detail:

- train (129,941 examples): All taken from the official SQuAD 2.0 training set.
- dev (6078 examples): Roughly half of the official dev set, randomly selected.
- test (5915 examples): Remaining examples from official dev set, plus hand-labeled examples.

5.2 Evaluation method

We evaluate based on two key metrics.

- Exact Match: a binary measure. It is equal to 1 if exactly matches the ground truth.
- F1: calculates the harmonic mean of precision and recall. Harmonic mean: the reciprocal of the arithmetic mean of the reciprocals of the given set of observations.

When evaluating on the validation or test sets, we take the **maximum** F1 and EM scores across the three human-provided answers for that question. This makes the evaluation more forgiving. Additionally, we look at the negative log-likelihood (NLL) and AvNA. AvNA measures the classification accuracy of answer vs no answer predictions.

5.3 Experimental details

For the baseline model, we follow the provided configurations, where:

- Learning rate: 0.5
- Training time: 30 epochs, or about 3 hrs

5.4 Results

Figure 2 shows the best performing runs of our different implemented models. As we had limited test-submissions, not all the results include the test F1/EM scores.
Based on these results, surprisingly, we saw that the character embedding-only model performed the highest. Please see Appendix for tensorboard result for training results versus the baseline.
In addition, the combined methods performed similarly or lower than each individual single-model, both when we combined character embeddings with self-matching attention and character embeddings with span corruption..
We observed a slight (approximately -0.6F1/-1EM) decrease of the test set evaluation compared to the dev set evaluation.

Model	Dev F1	Dev EM	Dev AvNA	Test F1	Test EM
Baseline	61.47	58.04	68.26	-	-
Span Corrupt Only	60.64	57.42	67.95	-	-
Back-translation*	61.74	59.21	71.22	-	-
Coattention	55.28	51.60	63.65	-	-
Self-matching attention	62.55	59.11	69.04	-	-
Character Embedding	**64.10**	**60.80**	**70.78**	**63.58**	**59.81**
Self-matching attention + character embedding	62.55	59.23	68.54	-	-
Span corruption + character embedding	59.97	56.49	67.11	-	-

Figure 2: Scores for Models and their modifications

* *Back-translation's scores were predicted based on preliminary training of 2-3 epochs. However, after training for 10 epochs, back-translation does not converge.*

We expect the score to be higher when we combine character embedding with self-matching attention, but the result is contrary to this expectation. The two improvements both individually achieve the same or higher F1 scores than combined. We suppose that this is because the hyperparameters that gave character embeddings the best performance are not the same as those that gave self-matching attention the best performance; i.e. while the hyperparameters such as 30 epochs and initial learning rate of 0.5 are suitable for both the baseline and baseline with character embeddings, however the self-matching attention layer would be trained more effectively with lower learning rates. For instance, while we ran shorter training runs for different hyperparameters to see the initial trajectory of training, we noticed that self-matching attention tends to converge more effectively with a learning rate closer to 0.25.

6 Analysis

6.1 Error analysis

To get a better sense of where our model is making mistakes, we look through individual questions ourselves and try to identify common sources of error.
The first category of error is that the model sometimes fails to recognize paraphrases. Even though "When" questions should be the easiest to predict, in the example below, the model fails to predict as it does not recognize the equivalence of meaning between "German Invasion of Poland" and "Germany invaded Poland". This could be solved by a more robust data augmentation technique that prevents the model from learning the text patterns rigidly.

Error Type: **Model fails to recognize paraphrases of words**

Question	When did Germany invade Poland and in doing so start World War II?
Context	After the **German Invasion of Poland on 1 September 1939** began the Second World War, Warsaw was defended till September 27. Central Poland, including Warsaw, came under the rule of the General Government, a German Nazi colonial administration.
Model Answer	N/A
Ground Truth	September 1939

Figure 3: Error Type 1 - Misunderstanding paraphrasing

The second category of error is that the model fails to capture causality through keywords such as "due to". In the example below, the model outputs the incorrect answer "the flora of the city" since it is pattern-matching the phrase "very rich in species". This could potentially be improved by pre-processing the model on a causality term vocabulary, including phrases such as "due to", "because", "the reason being", etc.

Error Type: **Model fails to understand causality**

Question	Why is Warsaw's flora very rich in species?
Context	**The flora of the city may be considered very rich in species**. The species richness is mainly **due to the location of Warsaw** within the border region of several big floral regions comprising substantial proportions of close-to-wilderness areas (natural forests, wetlands along the Vistula) as well as arable land, meadows and forests. Bielany Forest, located within the borders of Warsaw, is the remaining part of the Masovian Primeval Forest.
Model Answer	The flora of the city
Ground Truth	location of Warsaw

Figure 4: Error Type 2 - Causal relationships

The third type of error is that the model fails to understand the conjunction relationship (i.e. "and"). This is showcased by the following example. The model correctly predicted p_{start}, but did not capture the conjunctive relationship expressed by "and also". This is probably because the model usually see commas as a sentence separator, and stops the answer text span when it sees one.

Error Type: **Model fails to recognize text spans with conjunction**

Question	What tribes supported British?
Context	Context: The British were supported in the war **by the Iroquois Six Nations, and also by the Cherokee** – until differences sparked the Anglo-Cherokee War in 1758. In 1758 the Pennsylvania government successfully negotiated the Treaty of Easton, in which a number of tribes in the Ohio Country promised neutrality in exchange for land concessions and other considerations.
Model Answer	Iroquois Six Nations
Ground Truth	Iroquois Six Nations, and also by the Cherokee

Figure 5: Error Type 3 - Difficult understanding Conjunctions

6.2 Categorized questions

Figure 6 below shows how the models perform on different types of questions. The "when" questions perform the best across the board. We hypothesize that it is because it is relatively easy to predict which numbers/words represent a point in time compared to extracting text without numeric characters. By contrast, answering "how/why" or other questions requires the model to understand the intent in the context, which is a deeper level of understanding and thus a more difficult task.

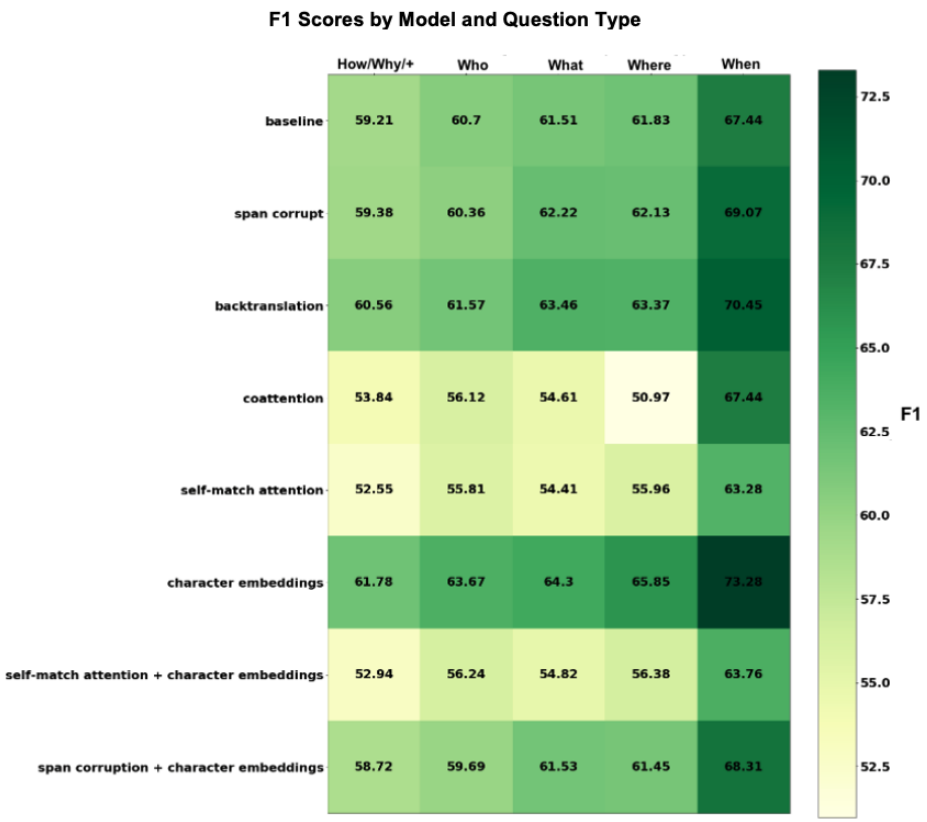

Figure 6: Heat map of model performance on different question categories.

Most of the models performed similarly across different categories of questions.
Character embedding provided a lift across all question types compared to the baseline. It performs the highest and relatively high compared to other categories on "when" questions out of all models, probably because there are lots of morphemes that encode meanings about time. For example, pre-fixes such as "pre" and "post" give hints about the order in time. They can be captured by using character embedding.
Coattention has particularly poor performance on "where" and "what" questions but stands out at the when questions. Since the dataset has more "what" questions, this may help it do better than the "where" questions. Adding self-matching attention also led to surprisingly poor comparative performance on some categories like "how and why" questions, "what" and "where" questions (though as mentioned in the footnotes, the performance is stronger across the board than appears in Figure 6). This feeds into the hypothesis that coattention and self-matching attention are better handling shorter phrases and worse at longer phrases; i.e. it is harder for these techniques to 'pay attention' to longer sections of context/harder to pay attention to a larger number of factors. As we saw in the error analysis section, many common areas were related either to longer phrases and paraphrasing or causal relationships which are more likely to be present when answering "how/why", "what" and "where" questions. [1]

[1]Note on Figure 5: self-match attention and self-match attention with character embeddings are indicative of the ratios of performance rather than overall performance as we lost access to our full 30-epoch trained model before conducting the by-category data analysis; we partially retrained within time and compute constraints to get these categorizations. Additionally, back-translation was extrapolated using a ratio due to compute constraints.

6.3 Data Augmentation: Span Corruption and Back-Translation

The current iteration of span corruption performed slightly worse than the baseline. We hypothesize that this is because a minority of the data was corrupted with short spans of OOV tokens - an average of 30% of examples had an average of 15% of the context corrupted, leading these examples to be too similar to the original data. This most likely lead to overfitting which is why the span corruption has lower performance than the baseline. Additionally, masking many different words as OOV may have confused the model when run with character embeddings, leading to worse quality embeddings and subsequent poorer performance.

Another variation of span corruption lead to half of the training dataset consisting of span corrupted data. The span corrupted data had an average of 20% the model was unable to learn effectively and showed no improvement beyond several epochs. With span corruption, we see the difficulty of data augmentation. Corrupting too much data leads to an inability for the model to converge, while too little data can lead to overfitting, since the model is seeing repeats of the same data. We hypothesize that our approach may have been naive and that better results require more time to process and test out different data corruption rates.

On the other hand, back-translation's performance began to decline after 5 epochs after rising quickly. One hypothesis for such behavior is that the translated queries did not boost data robustness in the desired manner. Most frequently, the translated sentences switched "what" and "which", introduced contractions, and removed commas. These were all changes that the models do not perform well on, as covered above. As such, these changes that made it deliberately harder for the model to learn may explain its poor performance.

7 Conclusion

Our main findings are as follows:

- **Character embeddings**, implemented alone, produce the greatest improvement in baseline.
- Improvement methods, when combined, did not always lead to an improvement.
  - Character embedding did not train well with data augmentation or our two implementation designs of attention, coattention or self-matching attention. We did, however, see a 1 point lift in F1 using self-matching attention alone.
  - Different methods require different hyperparameters to perform well.
- **Model performance varies** across different question categories.
  - All models performed the best on "when" questions
  - Coattention is particularly bad at "where" and "what" questions but stands out at the "when" questions. Since the dataset has more what questions, this may help it do better than the where questions. This feeds into the hypothesis that coattention is better handling shorter phrases and worse at longer phrases.

Some limitations and further directions of our research are as follows: (1) we would like to use more compute resources to tune hyperparameters of our models, especially attention based methods as we believe we did not find the best combination of learning rates and weight decay for these methods while we did see improved training performance with learning rate's lower than 0.5; (2) we want to try more variations of data augmentation since we found the model to be highly sensitive to changes in the percentage of augmented data and small adaptions like choice of corrupt tokens and span of corruption. Furthermore, while the techniques we explored led to discovering improvement in performance on the baseline, we are also curious to explore different architectures like QANet [10] and other transformer-based [11] methods which have been known to significantly increase performance.

References

[1] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

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

[3] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*, 2016.

[4] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, 2017.

[5] Shayne Longpre, Yi Lu, Zhucheng Tu, and Chris DuBois. An exploration of data augmentation and sampling techniques for domain-agnostic question answering. *arXiv preprint arXiv:1912.02145*, 2019.

[6] Siddhant Garg and Goutham Ramakrishnan. Bae: Bert-based adversarial examples for text classification. *arXiv preprint arXiv:2004.01970*, 2020.

[7] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[8] Yahui Chen. Convolutional neural network for sentence classification. Master's thesis, University of Waterloo, 2015.

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

[10] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.

[11] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Language modeling with longer-term dependency. 2018.

8 Appendix

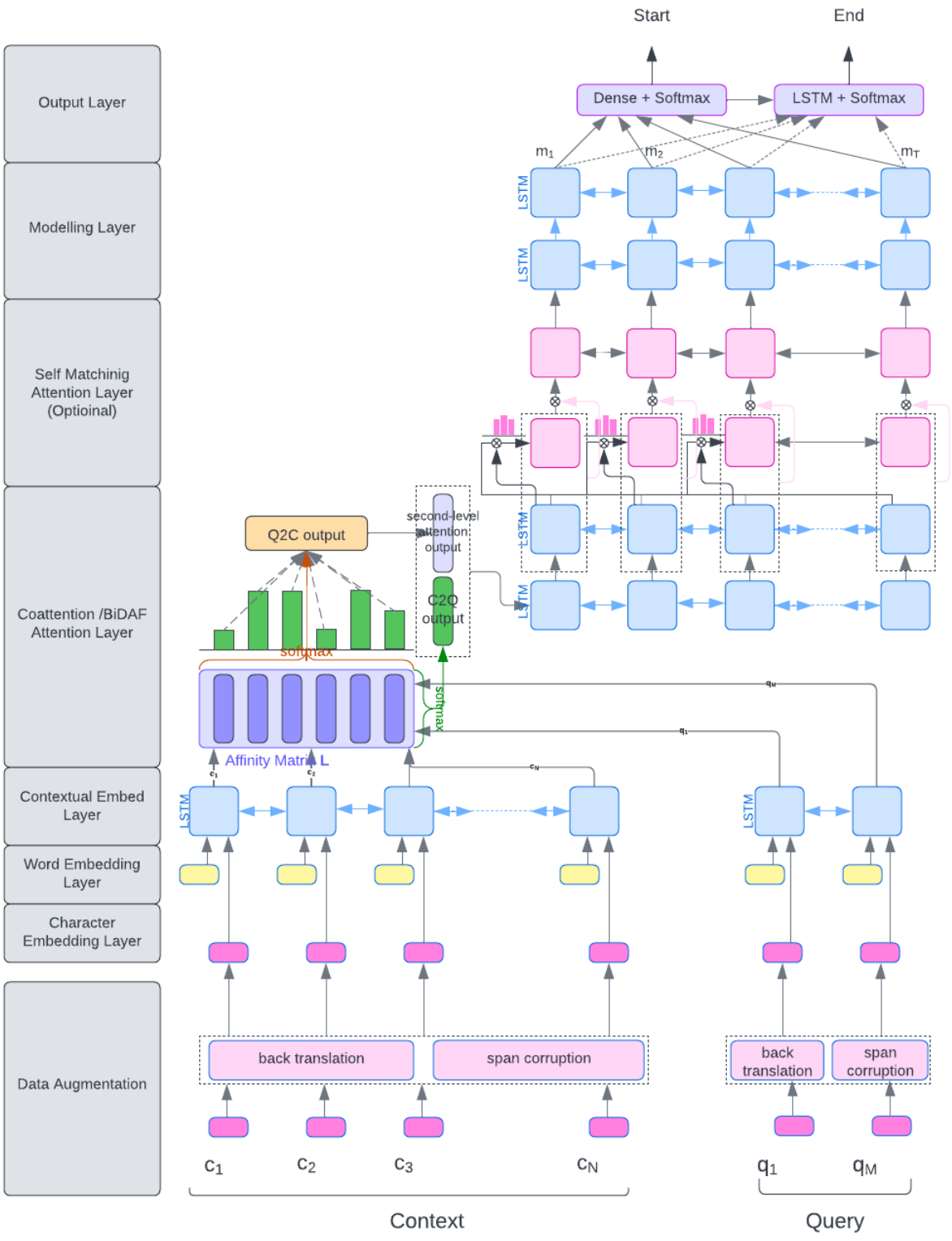

Figure 7: Full Scale Version of Model Architecture

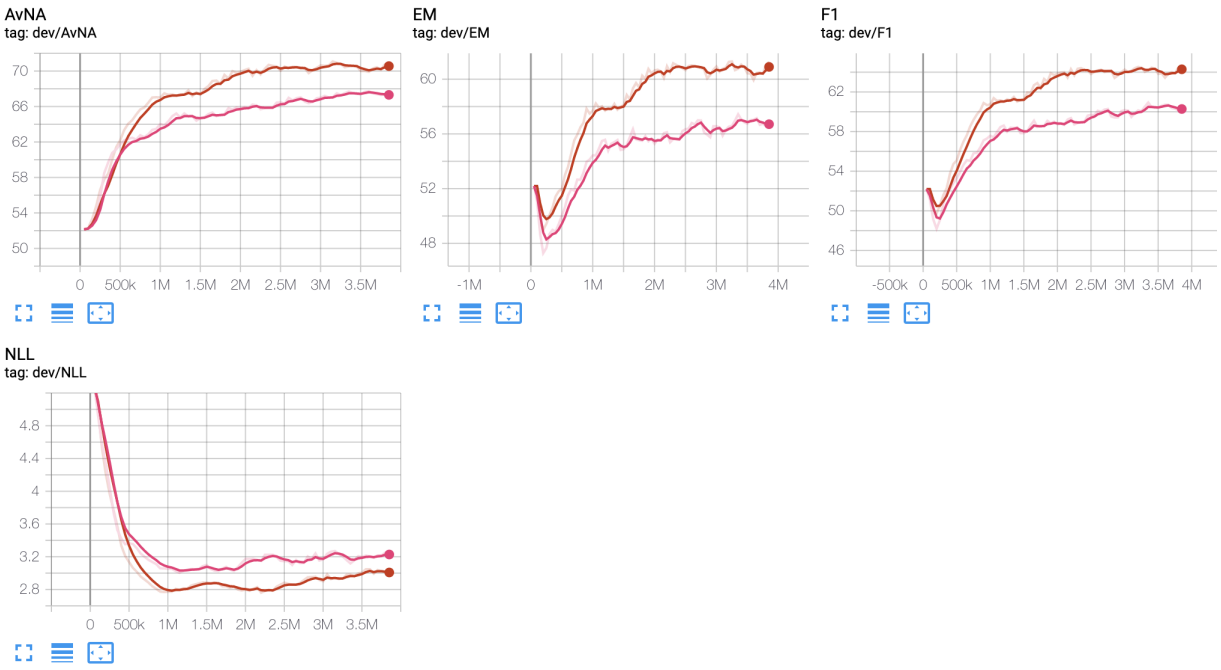

Figure 8: Character embedding versus baseline. Red is character embedding, pink is baseline