THE UNIVERSITY OF TEXAS AT DALLAS

Q&A with BiDAF+

Group 20

Sailesh Sriram
Vijay Anand Varma Indukuri
Rhugaved Narmade
Jahnavi Srividya Subramaniam

Q&A with BiDAF+

- One of the most fascinating application of Natural Language Processing is **Machine Comprehension**.
- Q&A entails answering questions about a certain text, context, or document
- Involves building systems that automatically answer questions posed by humans in a natural language
- Machine comprehension: Involves teaching models to read a passage of text(Context) and then answer questions(Query) about it

Goal: To improve the BiDAF model to effectively do Q&A tasks on machine comprehension given a context and query

Methodology (BiDAF Base-Model)

•*Closed-domain*, *extractive* Q&A model.

•Stands for Bi-Directional Attentional Flow (**BIDAF**)

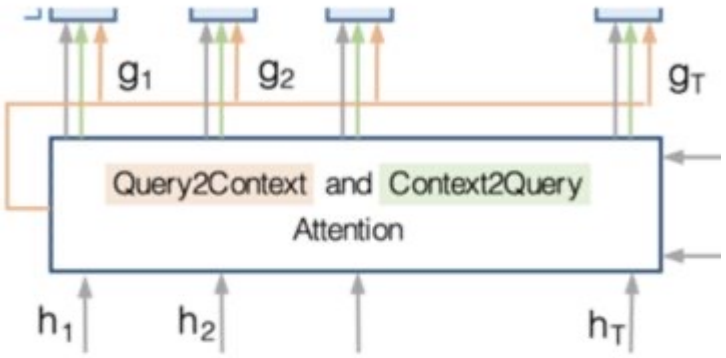
•Trained on **SQUAD 2.0**

•Uses four main layers: **encoding, attention, modeling, and output layers**

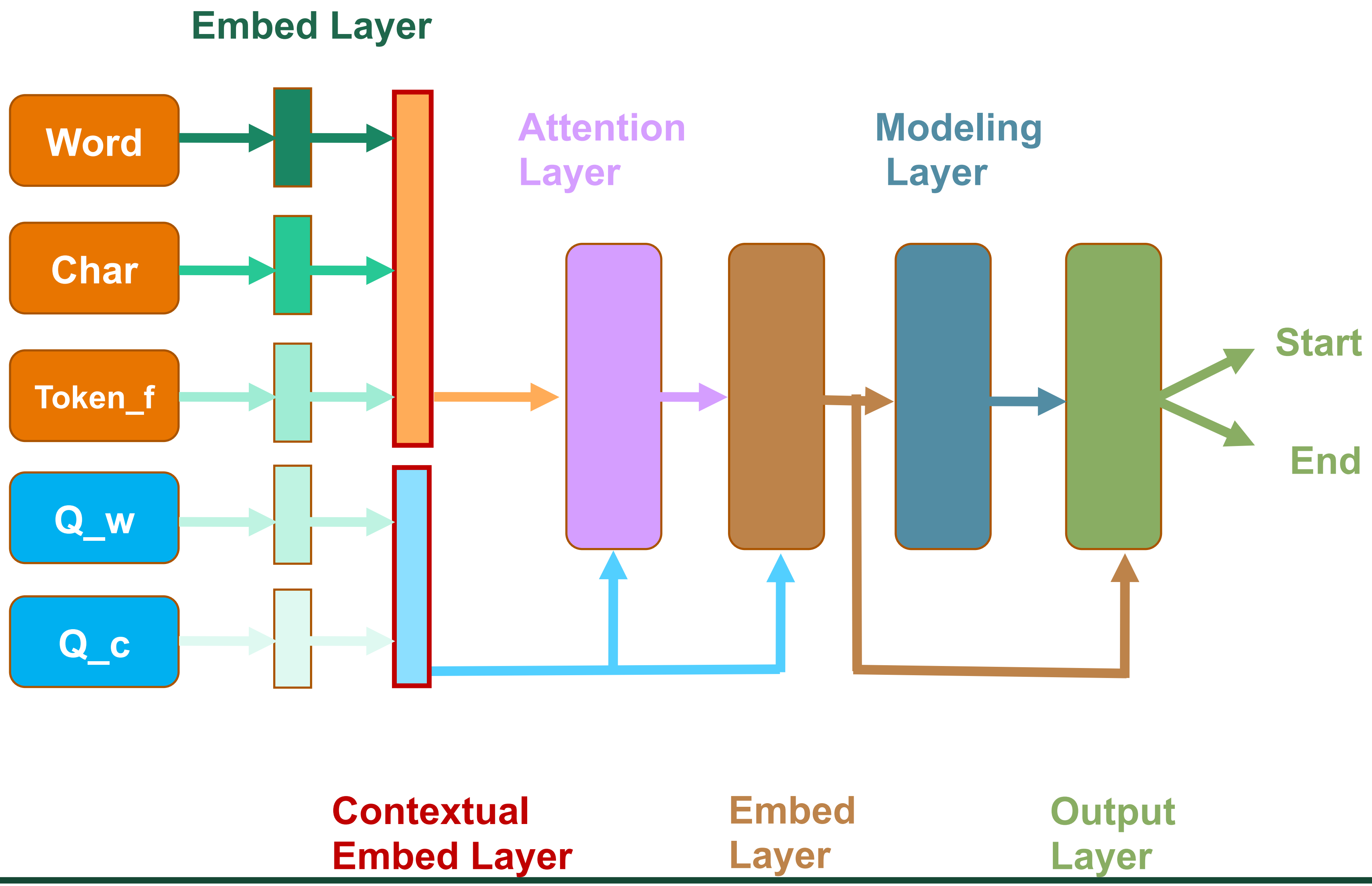
•Uses both **context-to-query** and **query-to-context attention**

•Output Layer predicts start and end positions within the context where the answer lies

•Foundation for our experiments



Overview



Embed Layer
Word
Char
Token_f
Q_w
Q_c
Attention Layer
Modeling Layer
Start
End
Contextual Embed Layer
Embed Layer
Output Layer

Experiments on BiDAF

In this work, besides the baseline model, we explore:

1.Embedding operations:

1. Character embeddings
2. Word Embeddings[Glove]
3. Token features[POS, NER, EM, TF] -> spaCy for extracting tags from text

2.Attention mechanisms:

1. Self-Attention
2. Coattention

3.Other Experiments

Evaluate different versions of our model with BiDAF(Baseline) and QANet on **EM and F1 Score**

Experimental Setup

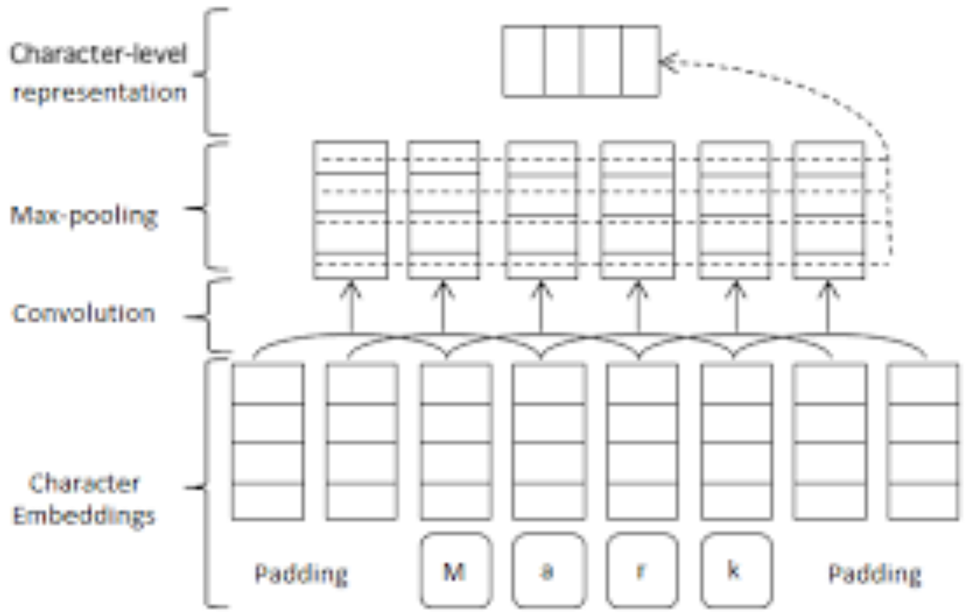
- All experiments are implemented in **Pytorch**
- Batch size **64**
- **30** Epochs
- Fixed Learning Rate of **0.5**
- Hidden size of **100**
- Default drop rate of **0.2**
- **Adadelta** optimizer
- Negative log likelihood optimizer
- Trained on **Google Colab**

1/2 SQUAD 2.0 description of files:

- **train-v2.0.json:**
  Total topics: 221
  Total paragraphs: 10035
  Total questions: 68319
- **dev-v2.0.json:**
  Total topics: 16
  Total paragraphs: 646
  Total questions: 6078
- **test-v2.0.json:**
  Total topics: 20
  Total paragraphs: 570
  Total questions: 5915

Character Embedding

- Vectors generated to represent characters in each word
- CNN Layers are built on character embeddings
- ReLU activation function, dropout, and max-pooling are appliec on the character embeddings
- Added batch normalization to every CNN Layer for regularization
- Tested three different scenarios:
    - 1 CNN Layer Without Batch Normalization
    - 2 CNN Layers Without Batch Normalization
    - 2 CNN Layers With Batch Normalization



Improvements with Character Embedding

	F1	EM	AvNA
BiDAF(base)	56.38	52.87	63.49
1 CNN Layer Without Batch Norm	56.74	53.67	63.54
2 CNN Layers Without Batch Norm	57.33	54.02	64.63
2 CNN Layers with Batch Norm	**60.34**	**56.75**	**67.45**

Token Features

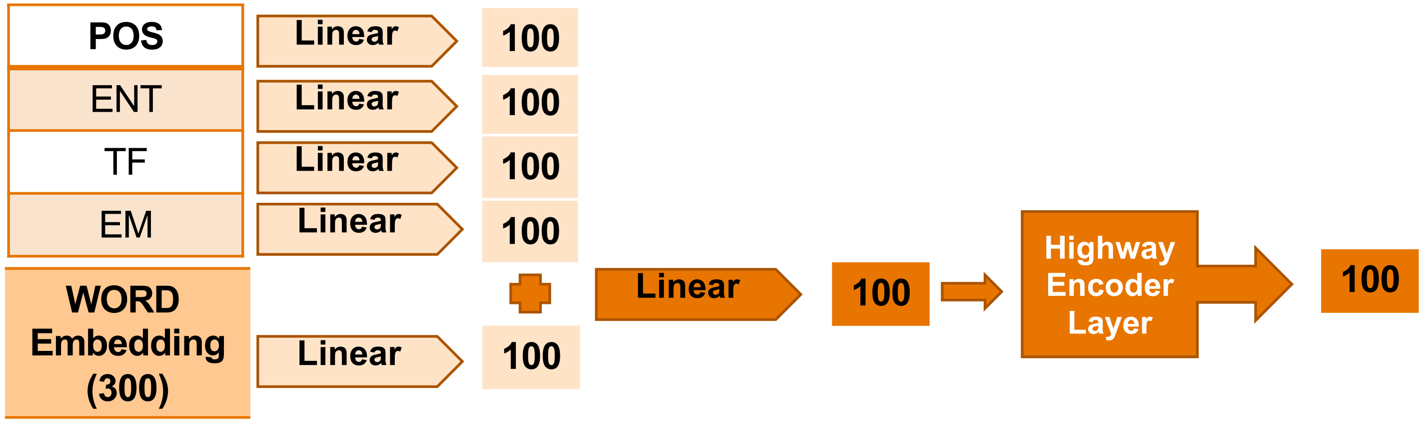
We experiment with ideas from **Chen et al.** on token features to create a latent vector

- **ENT** : Named Entities Recognized by the spaCy's small English model based on WordNet 3.0
  Eg. "Apple is looking at buying U.K. startup for $1 billion", "Apple" is tagged as an organization, "U.K." is tagged as a geological entity, and "$1 billion" is tagged as money.
- **POS** : Parts of Speech tags Recognized by spaCy
  Eg. "Apple" is tagged as "proper noun singular"
- **TF** : Frequency of the word in a context / Total words in context
- **EM** : Exact match vector for every word in Context vs Question
  comparision of each lowercase word in context with question, labled as 1 or 0

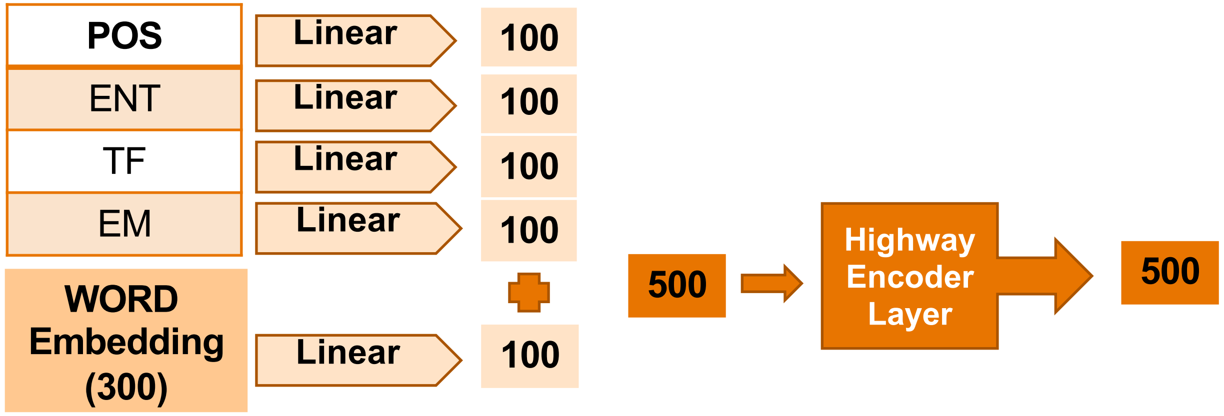
The four token features forms a **vector length of four for each word in the context**

Variations in using the Token Features

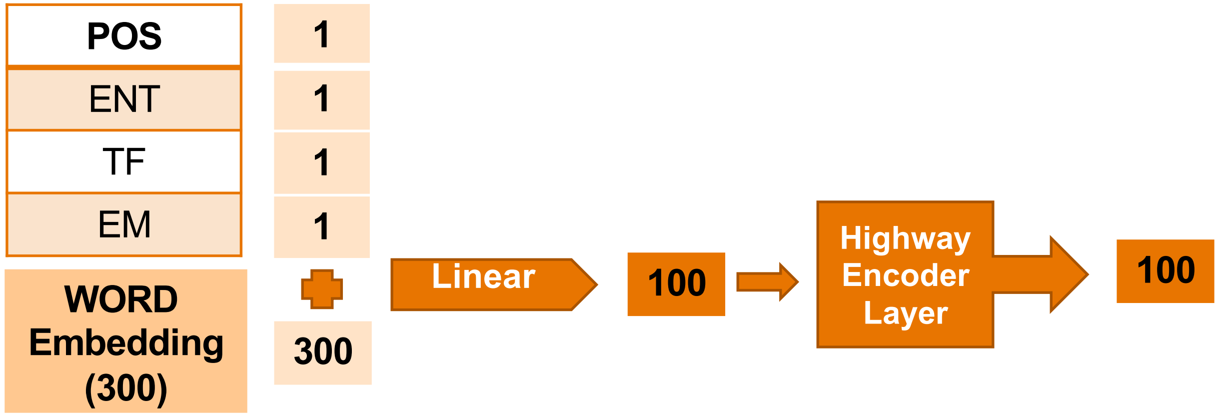
VARIATION 1



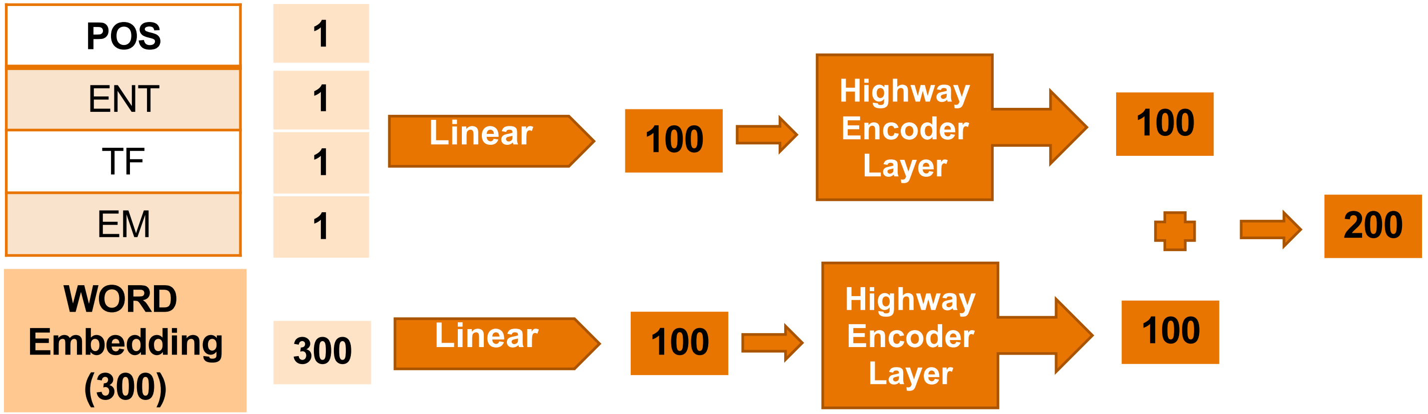
VARIATION 2



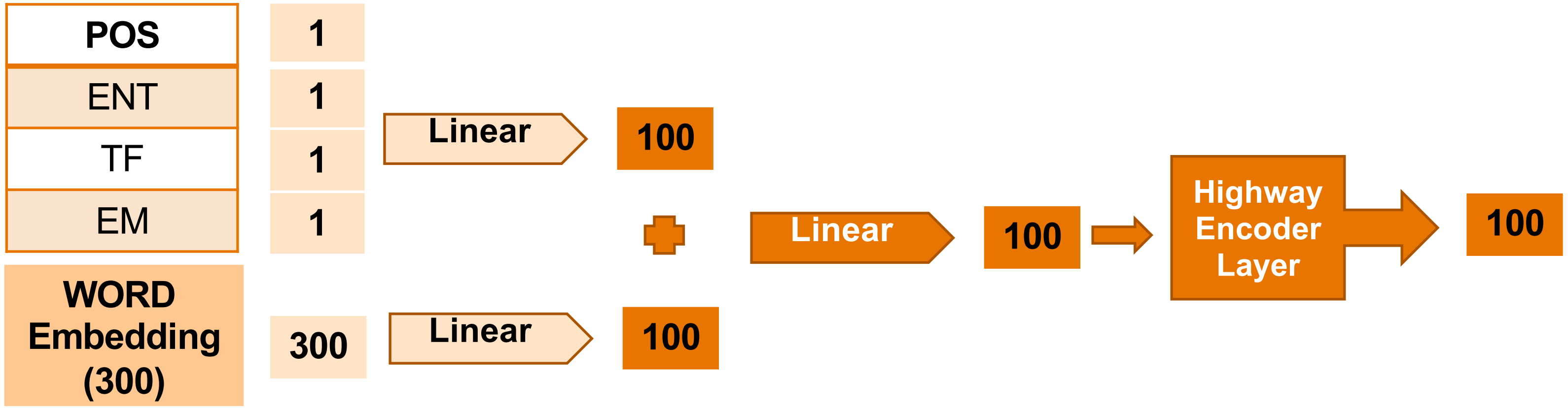
VARIATION 3



VARIATION 4



VARIATION 5



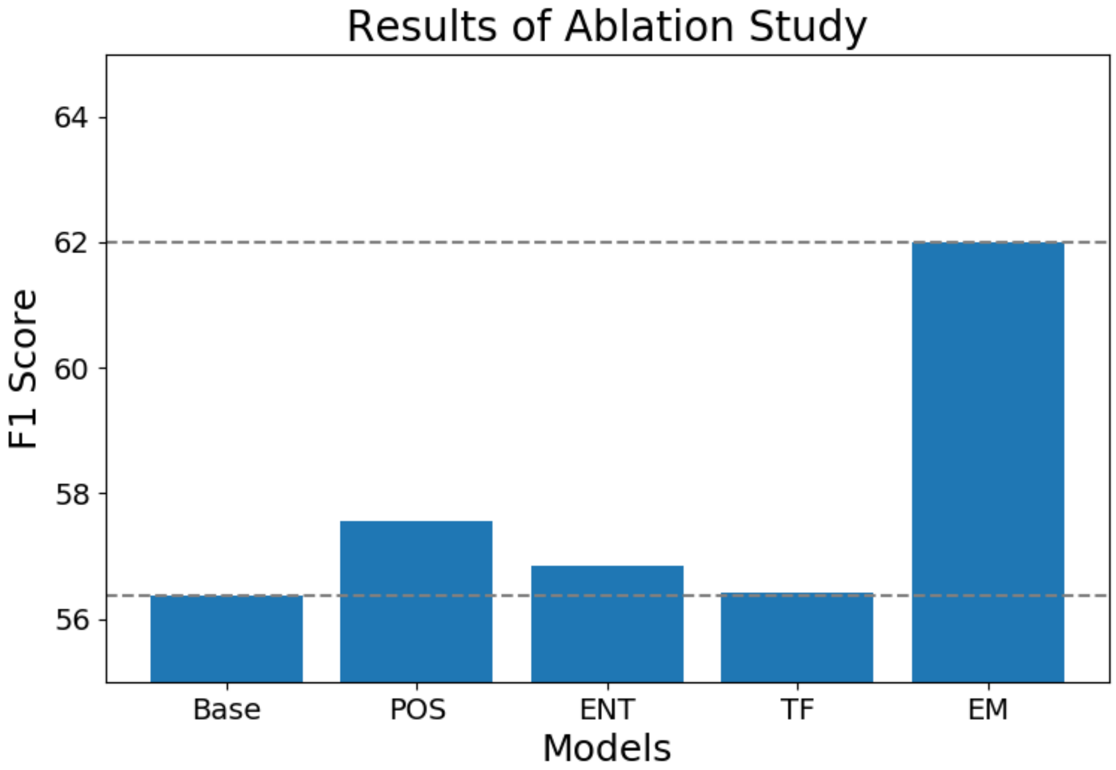
- 4 token features are pre-computed during setup for efficiency
- Results on the next slide showcase the performance jump compared to our baseline
- For variation 5, we form a vector of length four for each word in the context, pass it through a projection layer, concatenate with word embeds, pass through projection and finally pass through a small highway network

Results

Models	F1	EM
BiDAF(base)	56.38	52.87
Variant 1	59.60	56.12
Variant 2	57.39	54.07
Variant 3	57.45	52.46
Variant 4	58.30	55.59
Variant 5	**60.31**	**57.25**

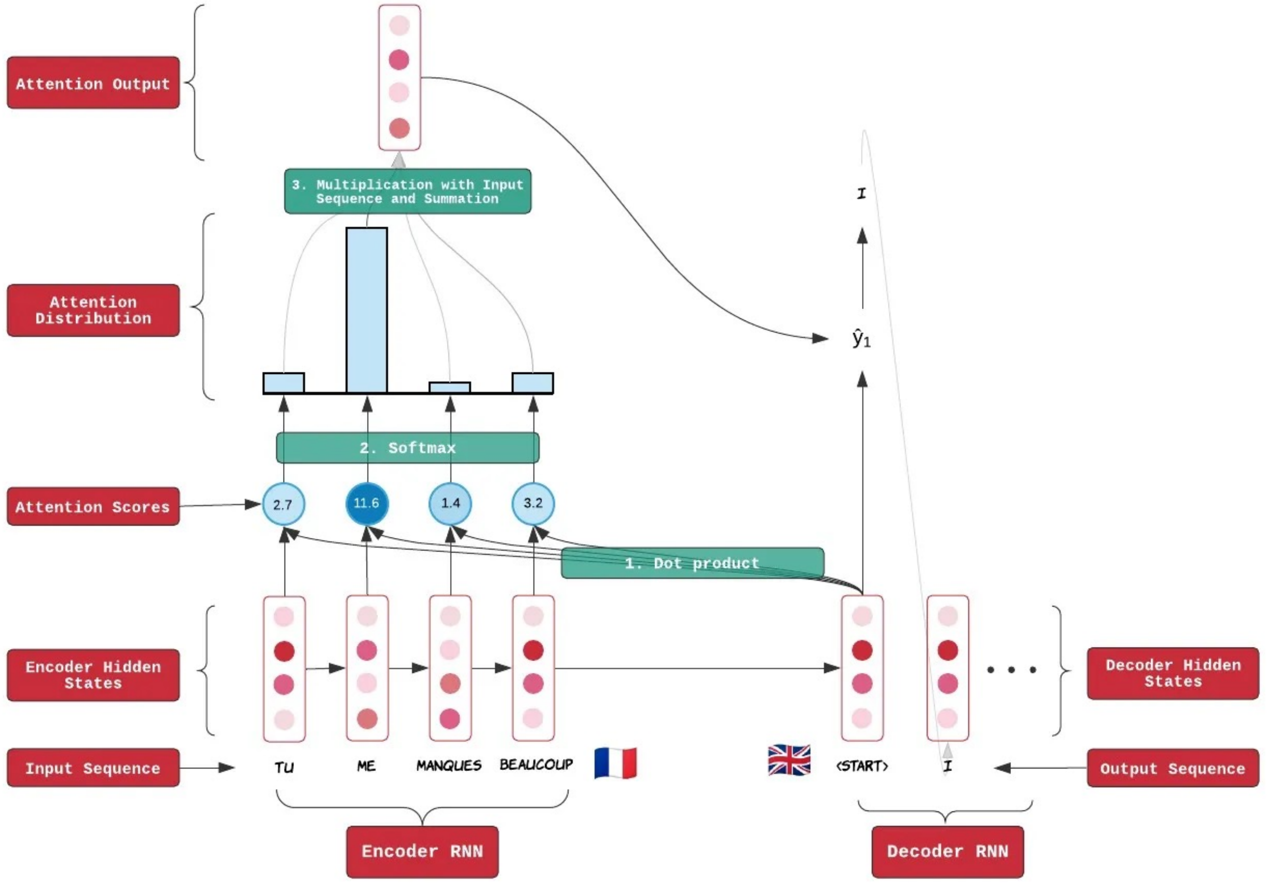
Ablation Study on Token Features

- Token features bring such a significant jump in F1 and EM metric
- Single Token Feature experiment
- All other features replaced by zeros, while the rest of the model is kept identical

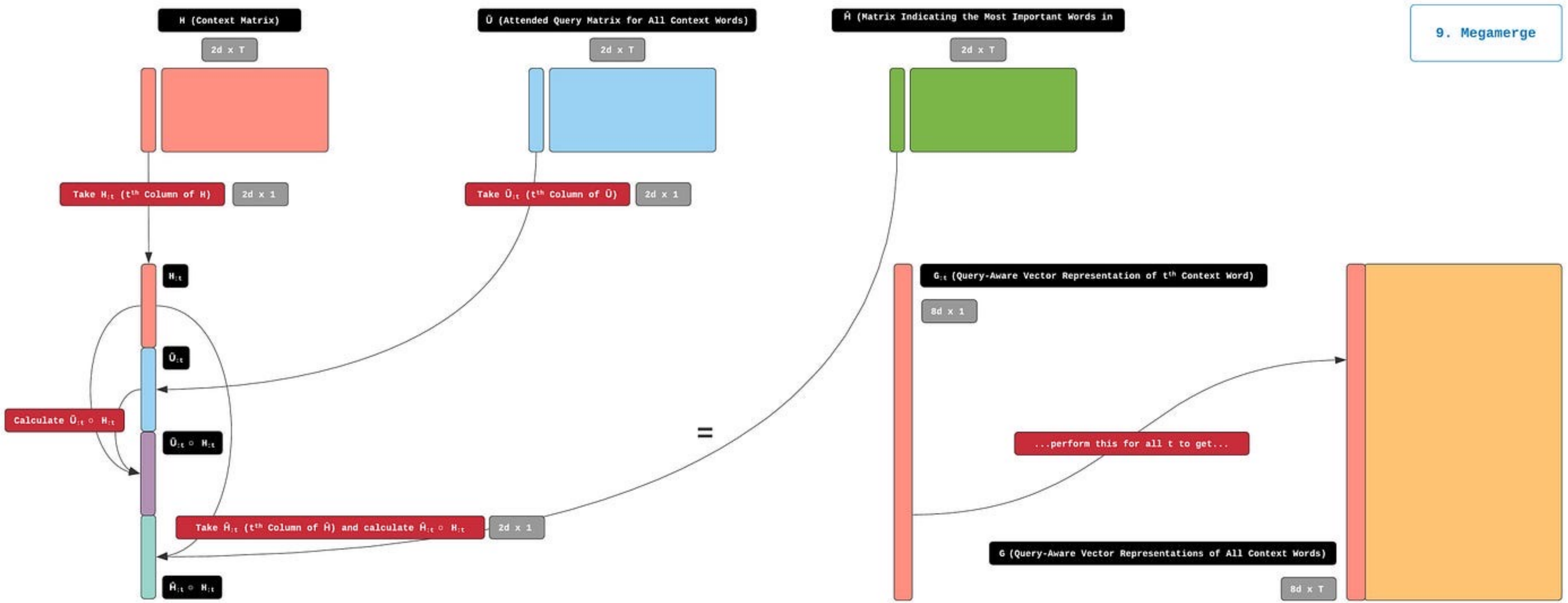


Attention

- Context matrix
- Similarity matrix
- Context-to-Query attention
- Query-to-Context attention
- Mega-merge

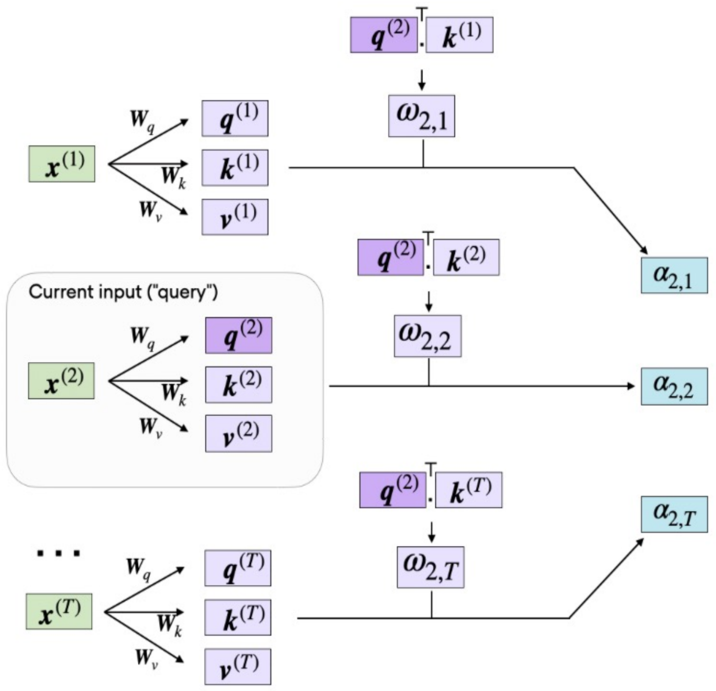


Attention


H (Context Matrix)
2d x T
Ũ (Attended Query Matrix for All Context Words)
2d x T
Ĥ (Matrix Indicating the Most Important Words in
2d x T
9. Megamerge
Take H:t (tth Column of H)
2d x 1
Take Ũ:t (tth Column of Ũ)
2d x 1
H:t
Ũ:t
Calculate Ũ:t ∘ H:t
Ũ:t ∘ H:t
=
Take Ĥ:t (tth Column of Ĥ) and calculate Ĥ:t ∘ H:t
2d x 1
Ĥ:t ∘ H:t
G:t (Query-Aware Vector Representation of tth Context Word)
8d x 1
...perform this for all t to get...
G (Query-Aware Vector Representations of All Context Words)
8d x T

Self-Attention

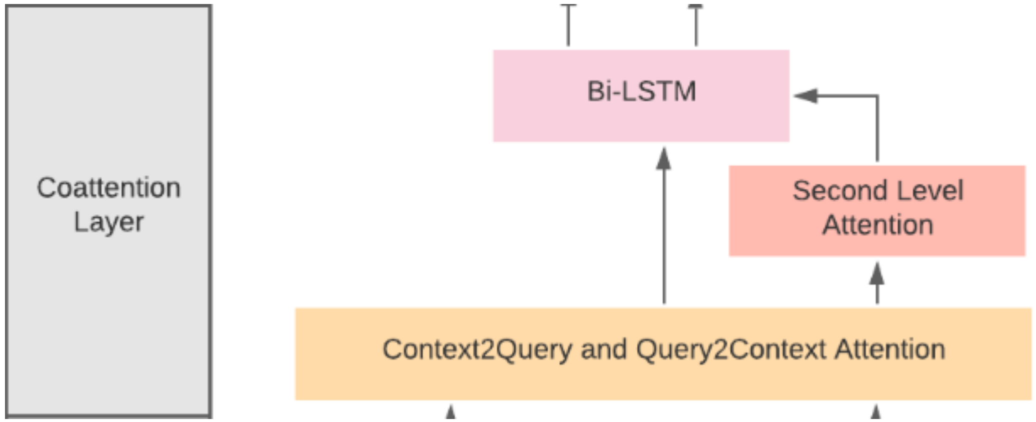
- Weight matrices for Query, Key and Value
- Unnormalized attention weights
- Attention scores
- On Q2C and C2Q attentions
- On BiDAF attention



where $\alpha_{2,i} = \text{softmax}\left(\frac{\omega_{2,i}}{\sqrt{d_k}}\right)$

Co-Attention

- Projected query hidden state
- Affinity matrix – Product of context and projected query hidden states
- Attention distributions(SoftMax) and vectors for C2Q and Q2C
- Weighted sum of Q2C with attention distributions of C2Q
- Feed this sequence to Bi-LSTM



Improvements with Attention

	F1	EM
BiDAF(base)	56.38	52.87
BiDAF(base) + Self-Attention	56.96	53.74
Self-Attention	**57.31**	**54.76**
Co-Attention	51.71	51.66

Conclusion

- Addition of Character Embeddings provides a big step up in performance in the base model(~4%)
- Token Features(ENT, POS, TF, EM) also increase the performance by a large margin(~4%)
- Self-attention on Q2C and C2Q matrixes performs better than co-attention
- We hope to see a considerable improvement in performance post integrating Character Embeddings, Token Features and Self Attention to our base model

Future work

- Integrate all experiments
- Train the final model on entire SQUAD 2.0 dataset
- Comparision with QANet
- Report final results

References

- Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. arXiv preprint arXiv:1611.01603, 2016.
- Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. arXiv preprint arXiv:1611.01604, 2016.
- Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. arXiv preprint arXiv:1704.00051, 2017.
- Mehryary, F., Bjerva, J., Dybå, T., & Gjøen, J. A. (2018). Comparing CNN and LSTM character-level embeddings in BiLSTM-CRF models for chemical and disease named entity recognition. arXiv preprint arXiv:1808.08450.
- https://sebastianraschka.com/blog/2023/self-attention-from-scratch.html
- https://towardsdatascience.com/the-definitive-guide-to-bidaf-part-3-attention-92352bbdcb07
- Base Code: https://github.com/michiyasunaga/squad

Thank you