



Upgrade

Open in app



Published in Towards Data Science · Following ▾



Meraldo Antonio · Follow

Aug 29, 2019 · 14 min read · Listen



THE DEFINITIVE GUIDE TO BiDAF — PART 3 OF 4

Attention Mechanism in Seq2Seq and BiDAF — an Illustrated Guide

Sequence-to-sequence (seq2seq) and Bi-Directional Attention Flow (BiDAF) are influential NLP models. These models make use of a technique called “attention” that involves the comparison of two sequences. In this article, I explain how the attention mechanism works in these two models.

This article is the third in a series of four articles that aim to illustrate the working of **Bi-Directional Attention Flow (BiDAF)**, a popular machine learning model for question and answering (Q&A).

To recap, BiDAF is an *closed-domain, extractive* Q&A model. This means that to be able to answer a **Query**, BiDAF needs to consult the an accompanying text that contains the information needed to answer the Query. This accompanying text is called the **Context**. BiDAF works by extracting a substring of the Context that best answers the query — this is what we refer to as the **Answer** to the Query. I intentionally capitalize the words Query, Context and Answer to signal that I am using them in their specialized technical capacities.

Context:

Singapore is a small country located in Southeast Asia.

Query:

Where is Singapore situated?

BiDAF's Answer:



Southeast Asia.

An example of **Context**, **Query**, and **Answer**. Notice how the Answer can be found verbatim in the Context.

In the [first article](#) in the series, I presented a high level overview of BiDAF. In the [second article](#), I talked about how BiDAF uses 3 embedding layers to get vector representations of the Context and the Query. The final outputs of these embedding steps are two matrices — **H** (which represent words in the Context) and **U** (which represent words in the Query). **H** and **U** are the inputs of the attention layers, whose function is to combine their informational content.

These attention layers are the core component in BiDAF that differentiates it from earlier models and enables it to score highly in the SQuAD leaderboard. The workings of the attention layers will be the focus of this article. Let's dive in!

Conceptual Introduction to Attention Mechanism

Before delving into the details of the attention mechanism used in BiDAF, it behooves us to first have an understanding of what attention is. Attention was first introduced in 2016 as a part of a [sequence-to-sequence](#) (seq2seq) model. **As its name suggests, seq2seq is a neural network model whose aim is to convert one sequence to another sequence.** An example application of seq2seq is the translation of a  French sentence to an English  sentence, such as the one below:





Upgrade

Open in app

*Tu me manques beaucoup*

->

*I miss you a lot*

A sample translation task for seq2seq model.

A seq2seq model consists of two Recurrent Neural Networks (RNNs):

- The first RNN, called “*encoder*”, is responsible for understanding the input sequence (in our example, a French sentence) and converting its informational content into a fixed-size intermediary vector.
- The second RNN, called “*decoder*”, then uses this intermediary vector to generate an output sequence (in our example, the English translation of the French sentence).

Prior to the incorporation of attention mechanism, seq2seq models can only deal with short sequences. This restriction arises because the “vanilla” seq2seq models can only fit a limited amount of information into the intermediary vector and some informational content is lost in the process.

This situation is akin to an attempt to translate a French book by first reading the whole book, memorizing its content and *then* translating it into English just from memory. As you can imagine, such an attempt is bound to fail—our poor translator will forget most of the book by the time he starts writing the English translation!

The attention mechanism was developed to solve this information bottleneck. The core idea of attention is that on each step of the decoding process, we are to make a direct connection to specific parts of the encoder.

In the context of our French-English translation task, this means that at **every time our model is to generate the next English word, it will only focus on the most relevant portions of the input French sentence.**

Conceptually, a seq2seq translation model with attention works just like how a normal human translator would translate a French text. He would read the first paragraph of the French text and translate it into English, move on to the second paragraph and translate this one, and so on. By doing so, he doesn't have to commit the whole content of the book into his memory and run the risk of forgetting most of its content.

Implementation of Attention in Sequence-to-Sequence Model

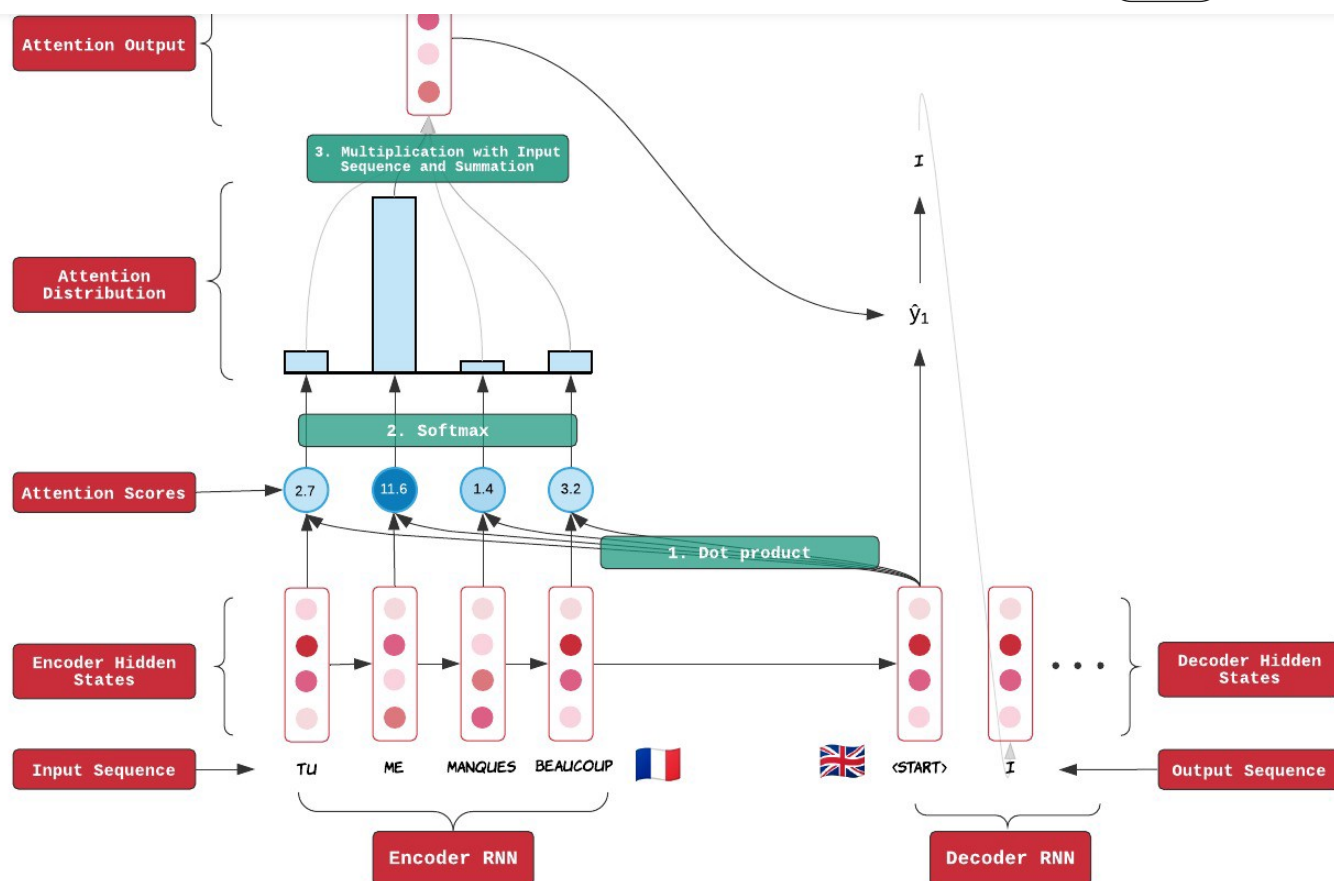
Practically, we can include an attention mechanism in a seq2seq model by performing the following three steps that are depicted in the diagram below:





Upgrade

Open in app



Attention mechanism in seq2seq

1. Comparison of Sequences and Calculation of Attention Scores

At each time step during the decoding process, we will compare the decoder hidden state with all of the encoder hidden states. This comparison can be done using any function that takes two vectors and outputs a scalar that reflects their similarity. The simplest of such function is a simple dot product. The scalar output of the similarity function is called an “**attention score**”; these attention scores are depicted by blue circles in the diagram above.

2. Conversion of Attention Scores to Attention Distribution

We then take the softmax of all these attention scores. The softmax function normalizes these attention scores into a probability distribution (a set of numbers that sum up to one). This probability distribution is called the “**attention distribution**”; it signals the parts of the input sequence that are most relevant to the decoding process at hand.

The blue bars in the diagram above show the attention distribution. We see that the bar corresponding to the second French word, “me”, is the tallest; this is because this word translates to “I” in English, which is the first word in our output sequence.

3. Multiplying Attention Distribution with Encoder Hidden States to Get Attention Output

We then multiply each element of the attention distribution with its corresponding encoder hidden states and sum up all of these products to produce a single vector called the “**attention output**”. You can think of the attention output as a selective summary of the input sequence. The attention output will then become the input to the next decoding step.

Although the three attention steps above were first applied to seq2seq, they are generalizable to other applications. As we will see later, BiDAF uses the same three steps in its implementation of attention, albeit with some minor modifications.

With this quick overview of the attention mechanism and its implementation in seq2seq, we are now ready to see how this concept is applied in BiDAF. *C’est parti!*





Upgrade

Open in app

information from \mathbf{H} and \mathbf{U} to create several matrix representations of the Context that also contain information from the Query.

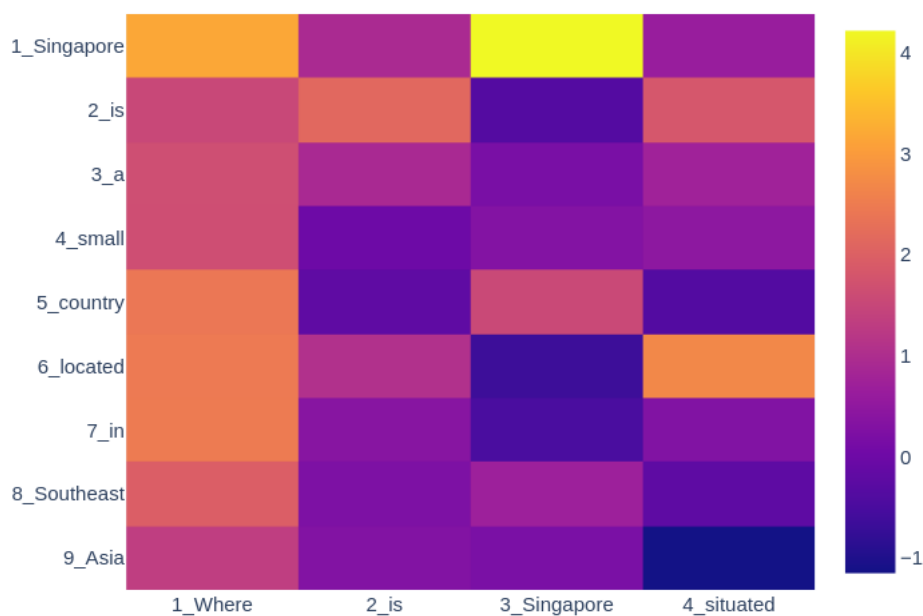
Our sixth step — the first attention-related step — is the formation of the so-called similarity matrix \mathbf{S} . \mathbf{S} is a tall skinny matrix with a dimension of \mathbf{T} -by- \mathbf{J} (number of words in Context by number of words in the Query).

The generation of the similarity matrix \mathbf{S} corresponds to the first step in the seq2seq attention mechanism discussed above. It entails applying a comparison function to each column in \mathbf{H} and each column in \mathbf{U} . *The value in row \mathbf{t} and column \mathbf{j} of the matrix \mathbf{S} represents the similarity of \mathbf{t} -th Context word and \mathbf{j} -th Query word.*

Let's take a look at an example of similarity matrix \mathbf{S} . Suppose we have this Query/Context pair:

- **Context:** "Singapore is a small country located in Southeast Asia." ($\mathbf{T} = 9$)
- **Query:** "Where is Singapore situated?" ($\mathbf{J} = 4$)

The similarity matrix \mathbf{S} produced from the above Query/Context pair is shown below:



An example of similarity matrix \mathbf{S}

We can make a couple of observations from the matrix \mathbf{S} above:

- As we expect, the matrix has a dimension of 9-by-4, 9 being the length of the Context(\mathbf{T}) and 4 being the length of the Query (\mathbf{J}).
- The cell in row 1, column 3 contains a relatively high value as indicated by its bright yellow color. This implies that the Query word and Context word associated with this coordinate are highly similar to each other. These words turn out to be the exact same word — "Singapore" — hence it makes sense that their vector representations are very similar.
- **Just because a Context word and a Query word are identical doesn't necessarily imply that their vector representations are highly similar!** Look at the cell in row 2, column 2 — this cell encodes the similarity of the Context word "is" and the identical Query word "is". However, its value is not as high as the "Singapore" pair above. This is because these vector representations also incorporate information from the surrounding phrases. This contextual contribution is especially important for small copulas such as "is".
- **On the other hand, we can see that the similarity value of two distinct words with close semantic and grammatical meaning, such as "situated" and "located" is relatively high.** This is thanks to our word and character embedding layers, which can generate vector representations that pretty accurately reflect a word's meaning.

Now let me tell you about how we calculate values in \mathbf{S} . *The comparison function used to perform this calculation is called α .* α is more



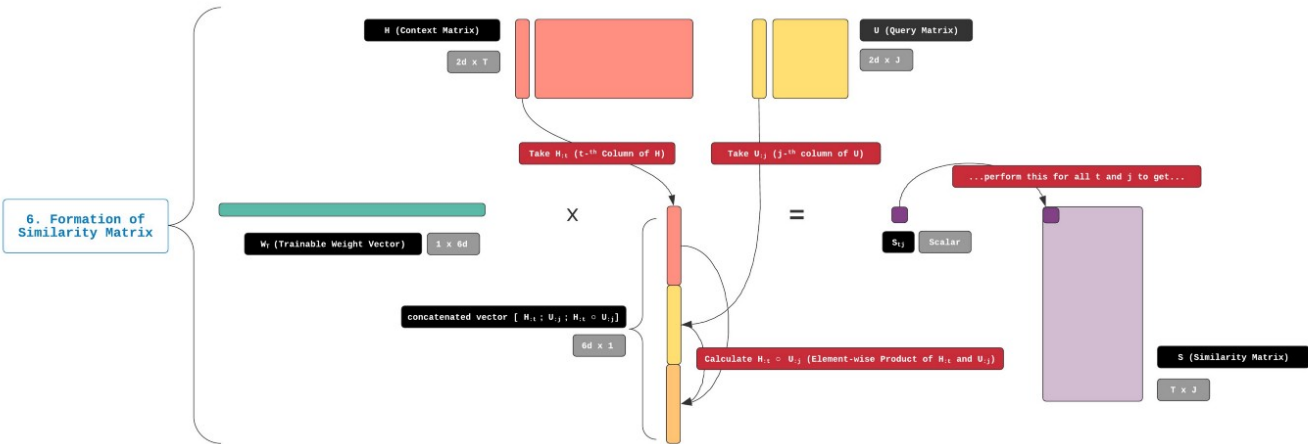
Upgrade

Open in app

- \mathbf{a} and \mathbf{b} are the two input vectors
- $\mathbf{w}^T \in \mathbb{R}^{6d}$ is a trainable weight vector
- \odot is element wise multiplication
- $[\ ;]$ is vector concatenation across row

As the function α comprises a multiplication of a row vector and a equally-sized column vector, it always returns a scalar.

The diagram below shows all the matrix operations performed in this step.



Step 6. Using the similarity function α , we combine context matrix H and query matrix U to form similarity matrix S .

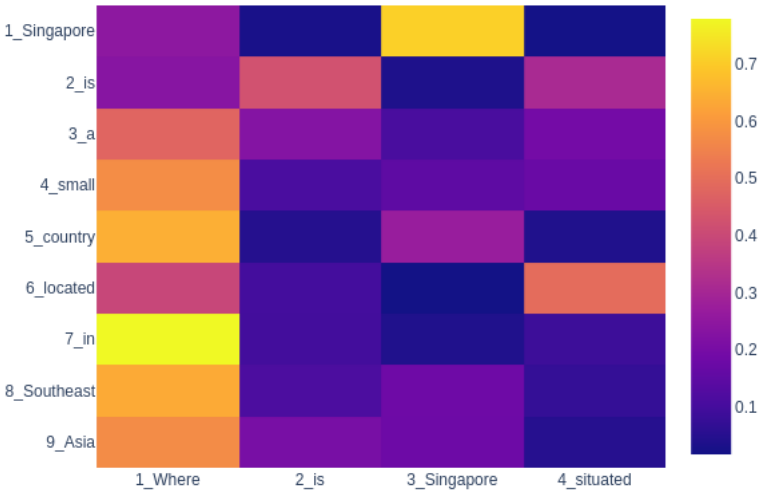
Step 7. Context-to-Query Attention (C2Q)

The similarity matrix S serves as an input to the next two steps: Context-to-Query attention (C2Q) and Query-to-Context attention (Q2C).

In this section, we will focus on C2Q. The goal of this step is to find which Query words are most relevant to each Context words.

Performing C2Q is similar to performing the second and the third steps of the seq2seq attention. First, we use the scalar values in S to calculate the attention distribution. This is done by taking the row-wise softmax of S . The result is another matrix. This matrix is not explicitly named in the BiDAF paper, but let's call it matrix A .

The matrix A , whose dimension is the same as S , indicates which Query words are the most relevant to each Context word. Let's look at an example of A :





Upgrade

Open in app

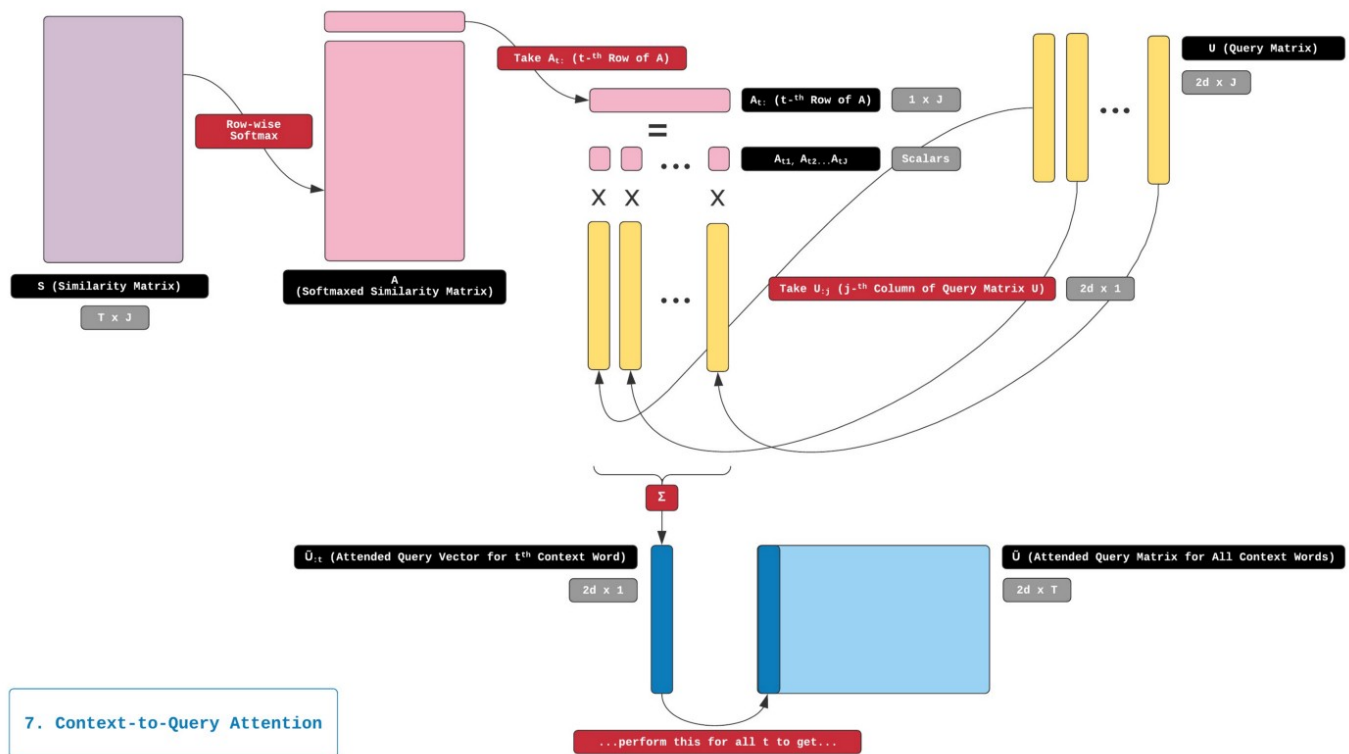
- **Semantic similarity strongly contributes to relevance.** We see that for the Context words [“Singapore”, “is”, “located”], the most relevant Query words are [“Singapore”, “is”, “situated”]. These are also words with which they share strong semantic similarity.
- **Context words “understand” the information requested by query words.** We see that the Query word “Where” is the most relevant query word for the Context words [“a”, “small”, “country”, “in”, “Southeast”, “Asia”]— words related to geographical location.

We then take every row of \mathbf{A} to get the attention distribution \mathbf{A}_t : which has a dimension of 1-by- \mathbf{J} . \mathbf{A}_t : reflects the relative importance of each Query word for the t -th Context word.

We then calculate the weighted sum of the query matrix \mathbf{U} with respect to each element in the attention distribution \mathbf{A}_t . The result of this step is the attention output matrix called $\tilde{\mathbf{U}}$, which is a $2d$ -by- \mathbf{T} matrix.

$\tilde{\mathbf{U}}$, just like \mathbf{H} , is a matrix representation of the Context. However, $\tilde{\mathbf{U}}$ contains different information from \mathbf{H} ! Whereas \mathbf{H} encapsulates the semantic, syntactic and contextual meaning of each Context word, $\tilde{\mathbf{U}}$ encapsulates the information about the relevance of each query word to each Context word.

The whole process to generate $\tilde{\mathbf{U}}$ from similarity matrix \mathbf{S} and query matrix \mathbf{U} is depicted below:



Step 7. Context-to-Query attention

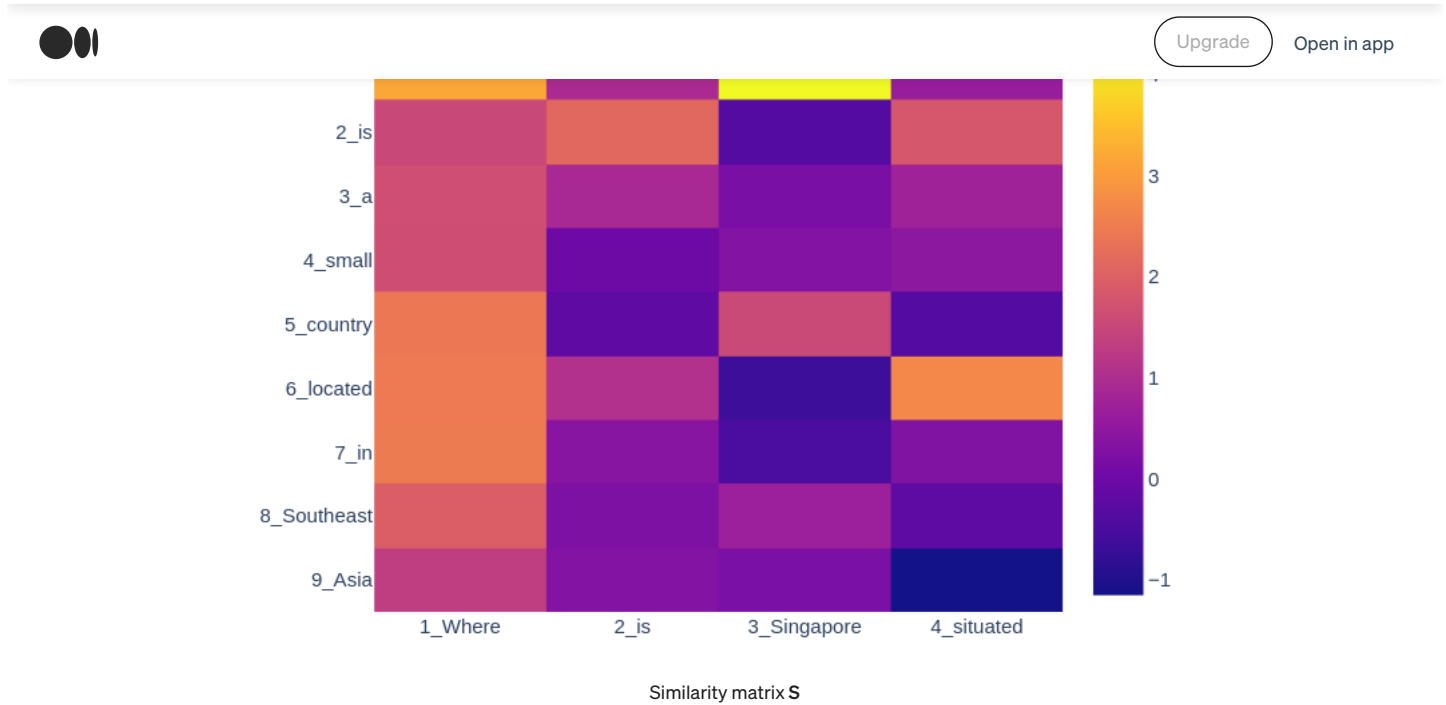
Step 8. Query-to-Context (Q2C) Attention

The next step is Q2C, which like C2Q also starts with the similarity matrix \mathbf{S} . In this step, our goal is to find which Context word is most similar to either one of the Query words hence are critical for answering the Query.

We first take the maximum across row of the similarity matrix \mathbf{S} to get a column vector. This column vector is not explicitly named in the paper, but let's call it \mathbf{z} .

Let's now step back and think about what \mathbf{z} symbolizes. Remember, our similarity matrix \mathbf{S} records the similarity between each Context word and each Query word. Let's take a second look at the example \mathbf{S} we created above.

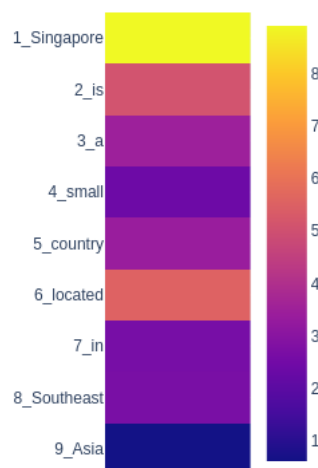




Now let's focus our attention to the the fourth row of this matrix, which corresponds to the Context word “small”. *One can see that there isn't any bright cell across this row!* This indicates that there is no word in the query that is similar in meaning to the Context word “small”. When we take the maximum across this row, the maximum value obtained will be close to zero.

Contrast this with the word “Singapore” and “situated”, in whose rows we do find at least one bright cell, indicating the existence of query word(s) similar to these words. When we take the maximum of these two rows, the corresponding value in the vector \mathbf{z} for these rows will also be relatively high.

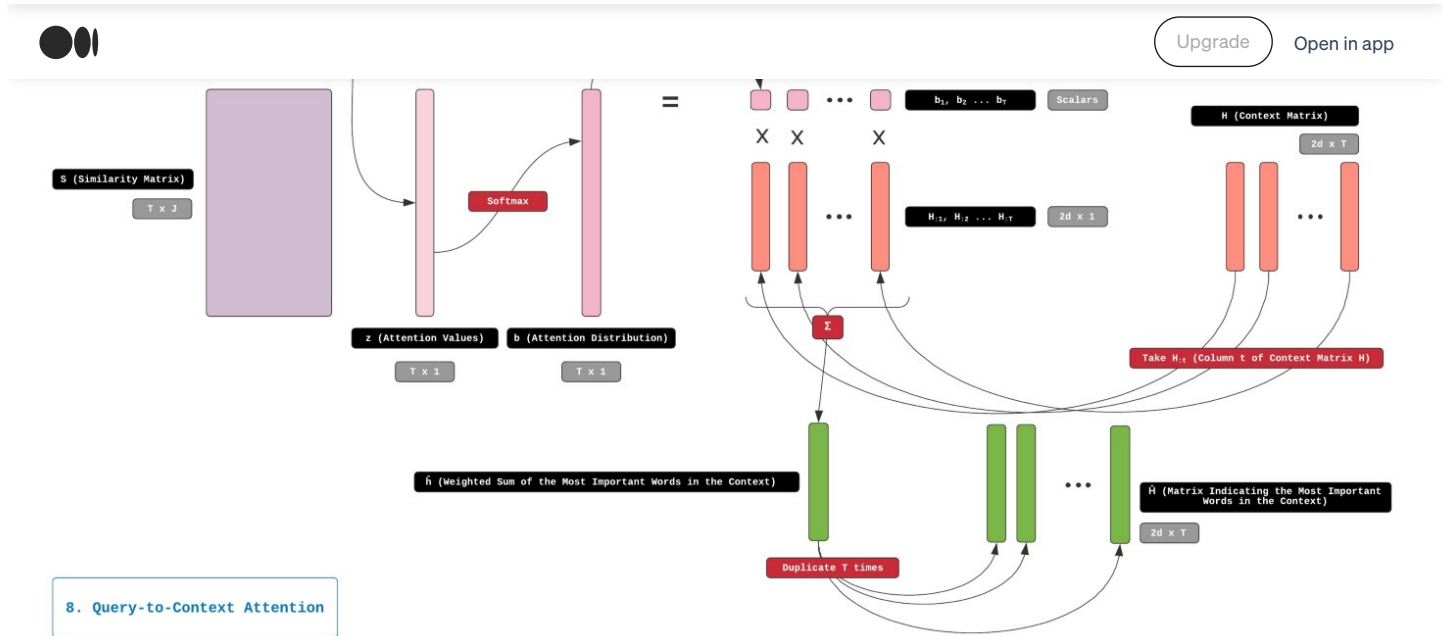
Here is the \mathbf{z} obtained for our example:



In Q2C, the values in \mathbf{z} serve our attention values. We apply softmax on \mathbf{z} to get an attention distribution called \mathbf{b} . We then use \mathbf{b} to take a weighted sum of the Context matrix \mathbf{H} . The resulting attention output is a $2d$ -by-1 column vector called $\hat{\mathbf{h}}$.

The last step of Q2C is copying-and-pasting $\hat{\mathbf{h}}$ T times and combine these copies into a $2d$ -by- T matrix called $\hat{\mathbf{H}}$. $\hat{\mathbf{H}}$ is yet another representation of the Context that encapsulates the information about the most important words in the Context with respect to the Query.

The whole process to generate $\hat{\mathbf{H}}$ from similarity matrix S and Context matrix \mathbf{H} is depicted below:



Step 8. Query-to-Context attention

Step 9. “Megamerge”

The matrices produced in steps 5, 7 and 8 are then combined to form a giant matrix \mathbf{G} . To refresh your memory, these three matrices are as follows:

- \mathbf{H} : the original Context matrix that encapsulates *the semantic, syntactic and contextual meaning of Context words*.
- $\tilde{\mathbf{U}}$: Context matrix that encapsulates *the relevance of each Query word to each Context word*.
- $\hat{\mathbf{H}}$: Context matrix that encapsulates *the information about the most important words in the context with respect to the Query*.

These three matrices have the same dimension — $2d$ -by- T .

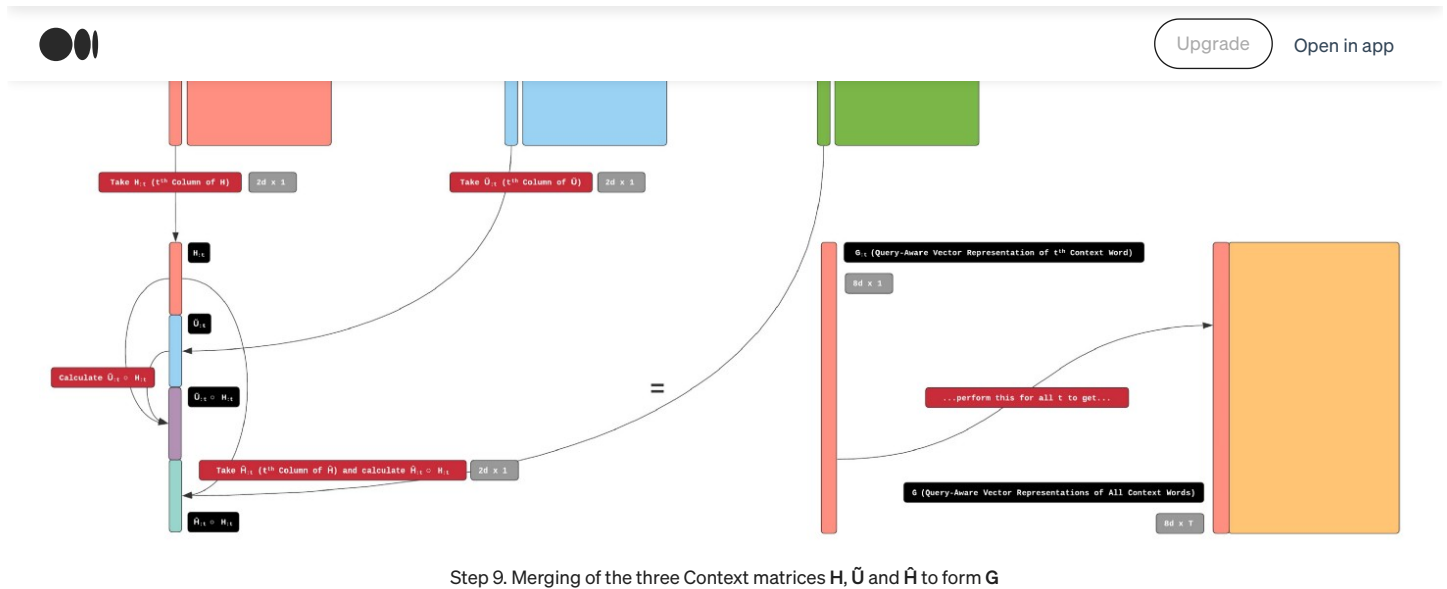
This “megamerging”, unfortunately, is not as simple as stacking these three matrices vertically! Here, we make use of another custom function called β . Here is the equation for β :

$$\beta(\mathbf{a}, \mathbf{b}, \mathbf{c}) = [\mathbf{a} ; \mathbf{b} ; \mathbf{a} \circ \mathbf{b} ; \mathbf{a} \circ \mathbf{c}], \text{ where...}$$

- \mathbf{a} , \mathbf{b} and \mathbf{c} are the three input vectors
- \circ is element wise multiplication
- $[\ ;]$ is vector concatenation across row

We define our megamerged matrix \mathbf{G} by $\mathbf{G}:\mathbf{t} = \beta(\mathbf{H}:\mathbf{t}, \tilde{\mathbf{U}}:\mathbf{t}, \hat{\mathbf{H}}:\mathbf{t})$ where $\mathbf{G}:\mathbf{t}$ is the t -th column vector of \mathbf{G} that corresponds to t -th Context word. \mathbf{G} has a dimension of $8d$ -by- T .

The whole process to generate \mathbf{G} from \mathbf{H} , $\tilde{\mathbf{U}}$ and $\hat{\mathbf{H}}$ is depicted below:



The giant matrix \mathbf{G} contains all information in \mathbf{H} , $\tilde{\mathbf{U}}$ and $\hat{\mathbf{H}}$. That is, you can think of each column vector in \mathbf{G} as a vector representation of a Context word that is “aware” of the existence of the Query and has incorporated relevant information from it.

So that's all there is to it about the attention layers in BiDAF! I know this might be a lot to take in, especially with the myriad of symbols and matrix operations involved. If you really want to study BiDAF in detail, I recommend you to print out the glossary below as well as all the diagrams above and study them side-by-side.

In the [next article](#), which will be the last article of the series, I will discuss about how **G** serves as an input to the modeling and output layer. The output layer will be the one that gives out probabilities for each Context word being included in the Answer span. I hope to see you in that [last article](#)!

Glossary

- **Context** : the accompanying text to a query that contains an answer to that query
- **Query** : the question to which the model is supposed to give an answer
- **Answer** : a substring of the Context that contains information that can answer the query. This substring is to be extracted out by the model.
- **T** : the number of words in the Context
- **J** : the number of words in the Query
- **U** : the original Context matrix that encapsulates the semantic, syntactic and contextual meaning of query words. **U** has a dimension of **2d-by-J**
- **H** : the original Context matrix that encapsulates the semantic, syntactic and Contextual meaning of Context words. **H** has a dimension of **2d-by-T**
- **S** : the similarity matrix that records the similarity between each Context word and each query word. **S** has a dimension of **T-by-J** (number of words in the Context by number of words in the Query).
- **α** : the comparison function used to get similarity values in **S**.
- **A** : the matrix that results from the row-wise softmax of **A** and indicates which query words are the most relevant to each Context word. **A** has a dimension of **T-by-J**.
- **z** : the column vector obtained by taking the maximum across row of the similarity matrix **S**. **z** has a dimension of **T-by-1**.
- **b** : an attention distribution vector that comes as the result of applying softmax to **z**. **b** has a dimension of **T-by-1**.



Upgrade

Open in app

- \mathbf{H} : Context matrix that encapsulates the information about the most important words in the context with respect to the query. \mathbf{H} has a dimension of $2d$ -by- T .
- \mathbf{G} : a big, $8d$ -by- T matrix that contains all information in \mathbf{H} , $\tilde{\mathbf{U}}$ and $\hat{\mathbf{H}}$. \mathbf{G} is the input to the modeling layer.
- β : a custom concatenation function used to build \mathbf{G}

References

[1] [Bi-Directional Attention Flow for Machine Comprehension](#) (Minjoon Seo *et. al*, 2017).

[2] [Neural Machine Translation by Jointly Learning to Align and Translate](#) (Bahdanau *et. al*, 2015).

If you have any questions/comments about the article or would like to reach out to me, feel free to do so either through [LinkedIn](#) or via email at meraldo.antonio AT gmail DOT com.

Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. [Take a look](#).

Get this newsletter

Emails will be sent to ammaarahmad1999@gmail.com.
[Not you?](#)

