Question Answering with BiDAF

LORENZO MARIO AMOROSA
ANDREA ESPIS
MATTIA ORLANDI
GIACOMO PINARDI

Contents

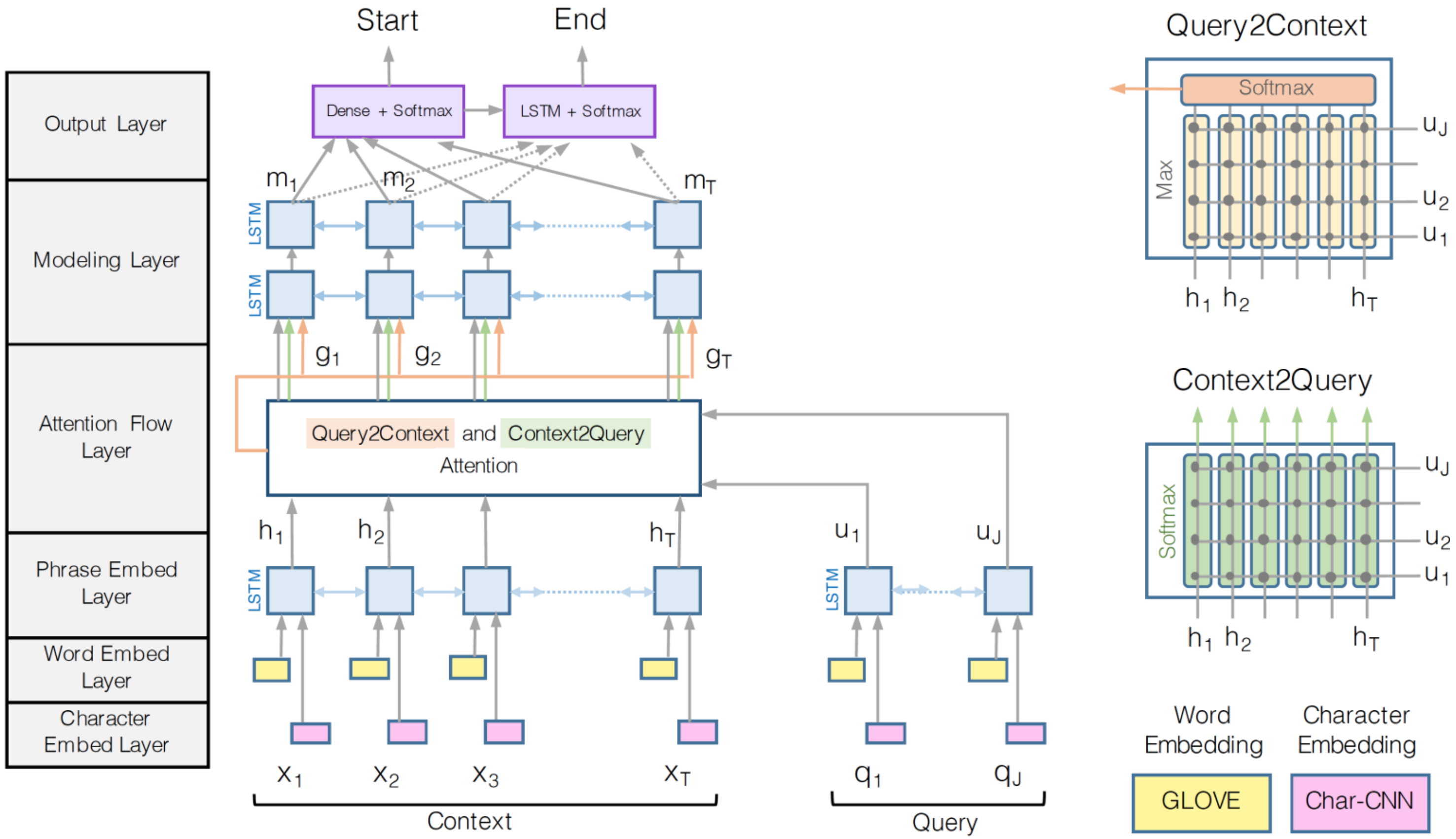
Brief description of the problem

- Predict the answer span given a context and a question
- SQuAD dataset
- BiDAF model:
  - Close-domain (works with query + context, no access to external KB)
  - Extractive
  - It answers to factoid questions

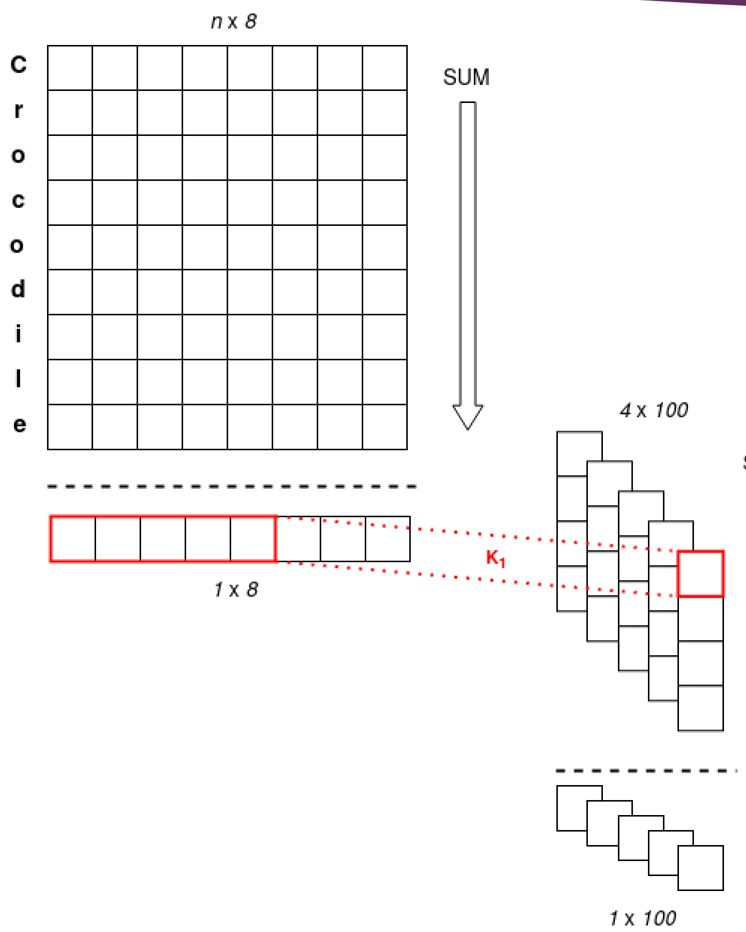
Pre-processing

- Tokenization using NLTK's TreebankWordTokenizer
- Conversion of answer span indexes from character-level to token-level
- Dataset split in the following way:
  - 64% reserved for training (57K samples)
  - 16% reserved for validation (13K samples)
  - 20% reserved for test (17K samples)

BiDAF structure: Overview



Character Embedding Layer – Baseline

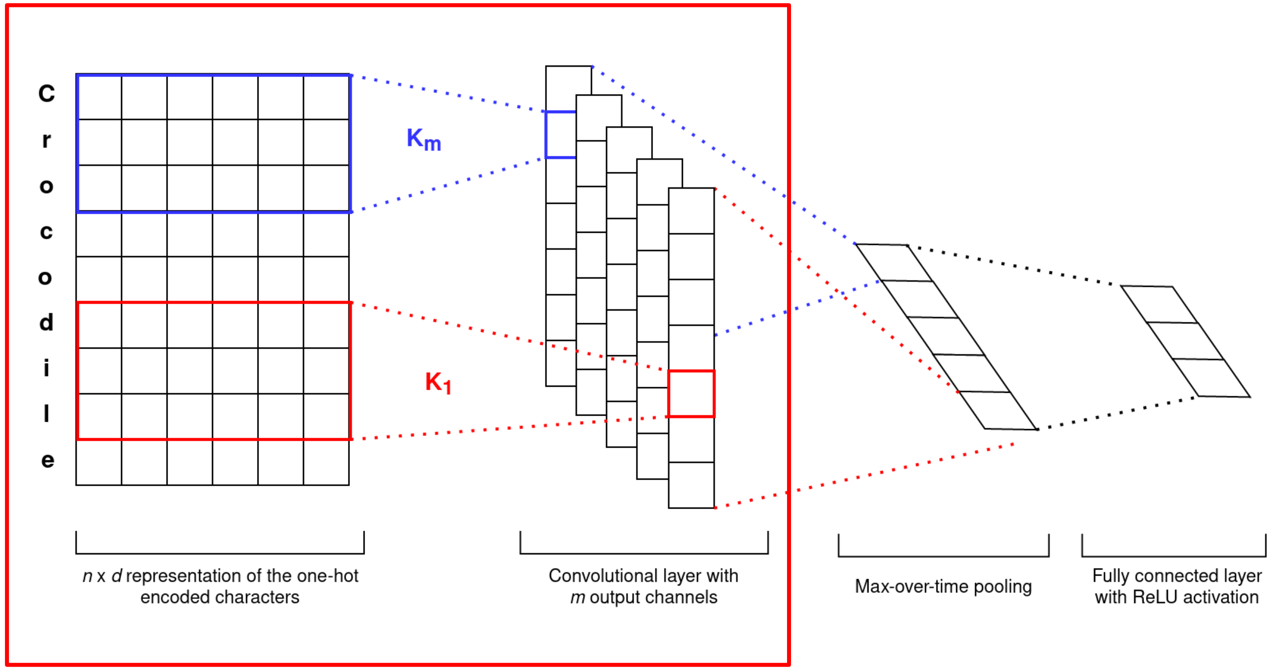


- Each character of the sentences is encoded using a trainable 8-dim embedding layer. The output has shape: batch size, sequence length, word length, char embedding dim (8); 2D slices in picture
- The embedding of each word is obtained by summing on the word length dimension (BS x SL x 8)
- Apply a convolution with kernel 1x5 and 100 output channels
- Sum on char embedding dimension obtaining a 3D tensor (BS x SL x 100)

Character Embedding Layer – Our improvements

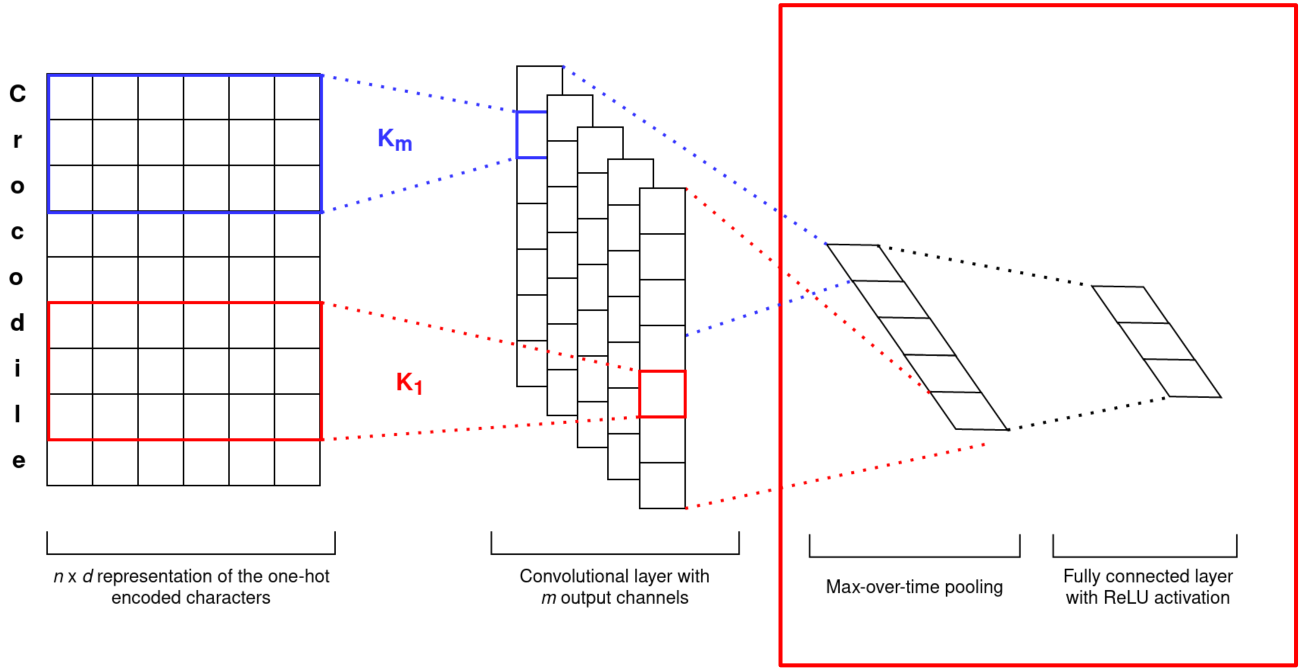
- 101-dimensional one-hot vector to each character using a non-trainable one-hot encoder
- 1100 distinct characters in the corpus → 99 most frequent + padding + unknown:
  - <a> = [1, 0, 0, …, 0]
  - <b> = [0, 1, 0, …, 0]
  - <PAD> = [0, 0, …, 0]
  - <UNK> = [0, …, 0, 1]
- The output of the one-hot encoder is a 4D tensor: batch size, sequence length, word length and 101-dim character embedding.

Character Embedding Layer – Our improvements



- We consider 2D slices of the 4D tensor having shape *word length* x *char embedding dim* (n x d)
- Each slice is processed through a 2D convolution with m output channels and with kernel width equal to d, so the filter is slid only vertically

Character Embedding Layer – Our improvements

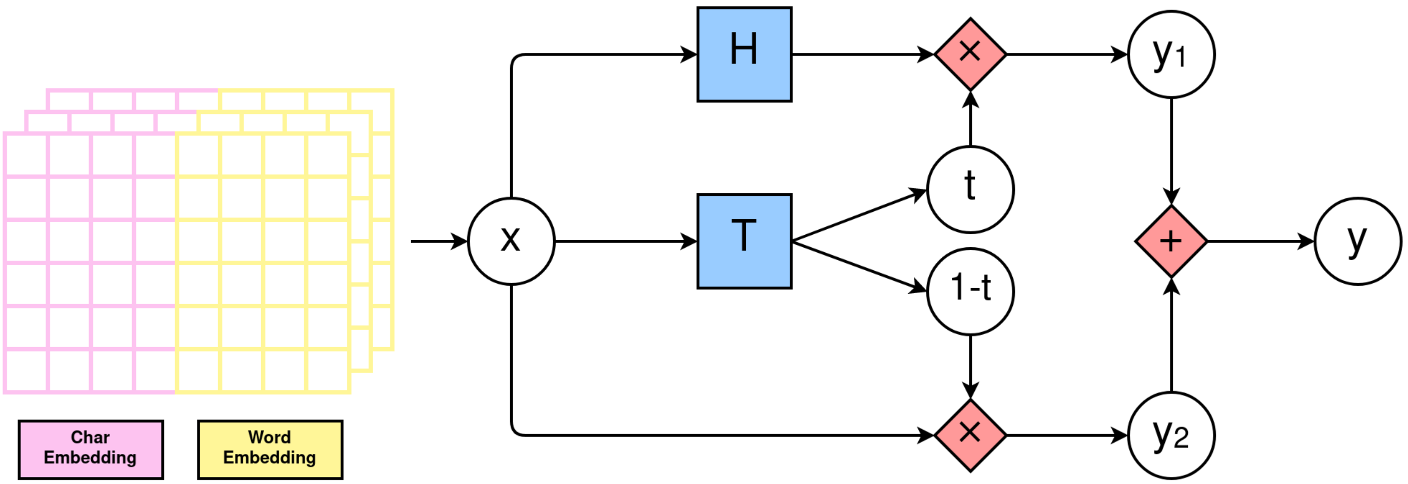


- From each of the m channels the max value is extracted using max-pooling
- The obtained vector is processed by a dense 2-layer neural network
- Output: 3D tensor (batch size, sequence length, out emb dim)
- Advantage: every word has fixed representation

Word Embedding Layer

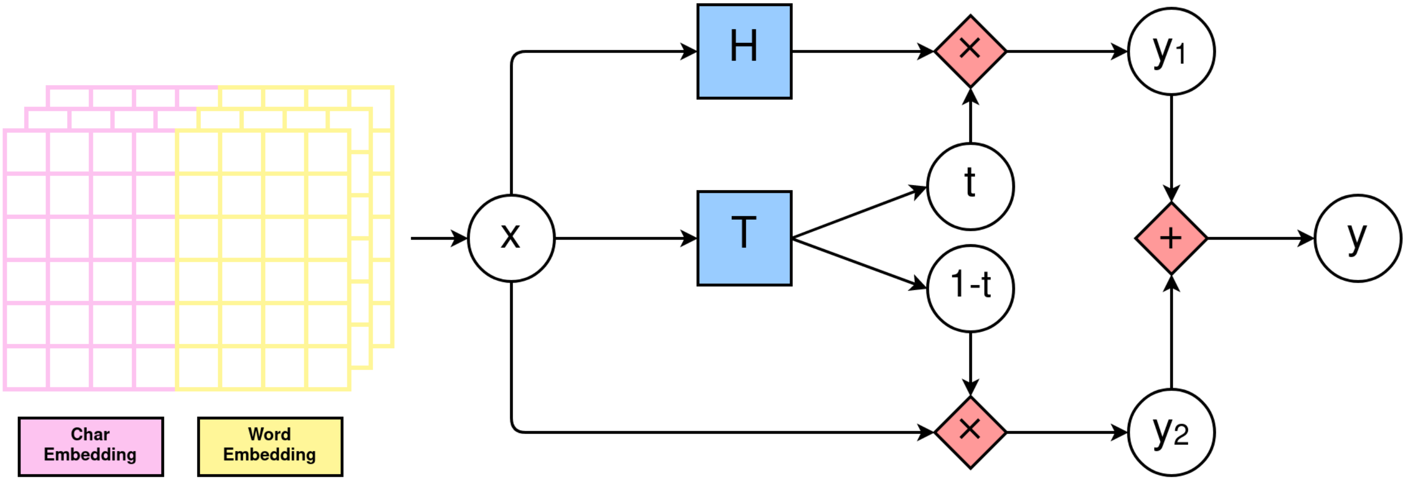
- Embedding at the word level
- Implemented as PyTorch Embedding layer
- Weights initialized using GloVe with same embedding dimension as Char Embedding Layer to give them same importance
- The output 3D tensor has shape: batch size, sequence length, output embedding dim

Contextual Embedding Layer – Highway Network



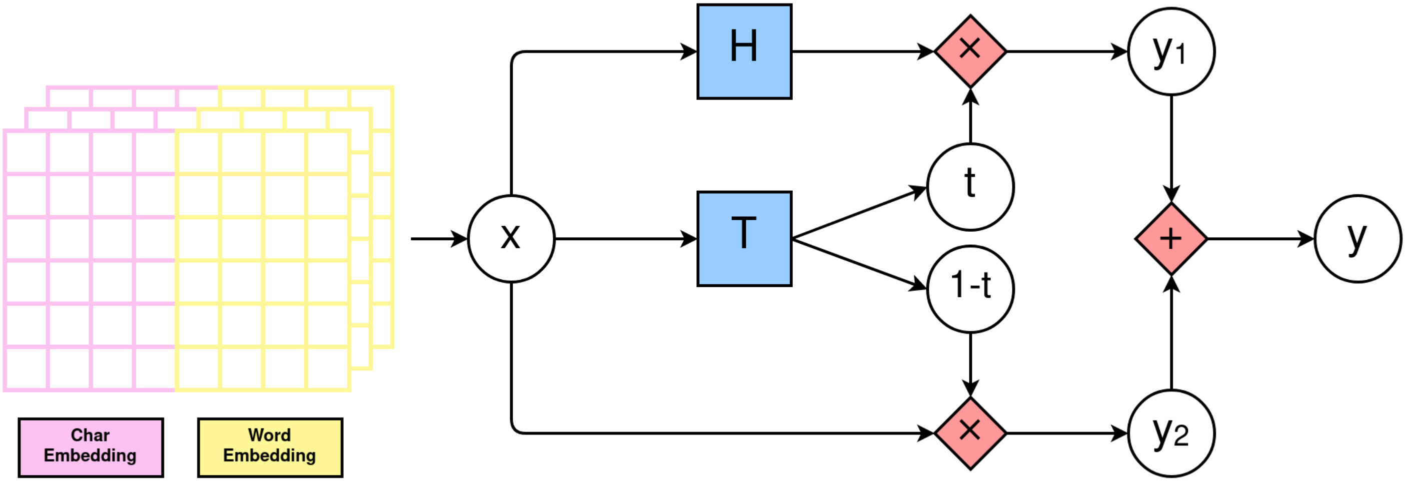
- Generalization of the residual block
- Highway Network input: Char and Word Embeddings concatenation along the embedding dimension
- $y = H(x) \cdot t + x \cdot (1 - t)$
- t is computed by the Transform gate T
- Two branches:
  - First branch: the input x is transformed using the transform function H and re-weighted according to t
  - Second branch: the input x is carried out as it is and re-weighted according to $1 - t$

Contextual Embedding Layer – Baseline



- Baseline:
  - Two Dense Highway Network blocks
  - Transform gate T and transform function H are both fully connected neural networks
  - Weighting value t is distinct for each element of the activation tensor

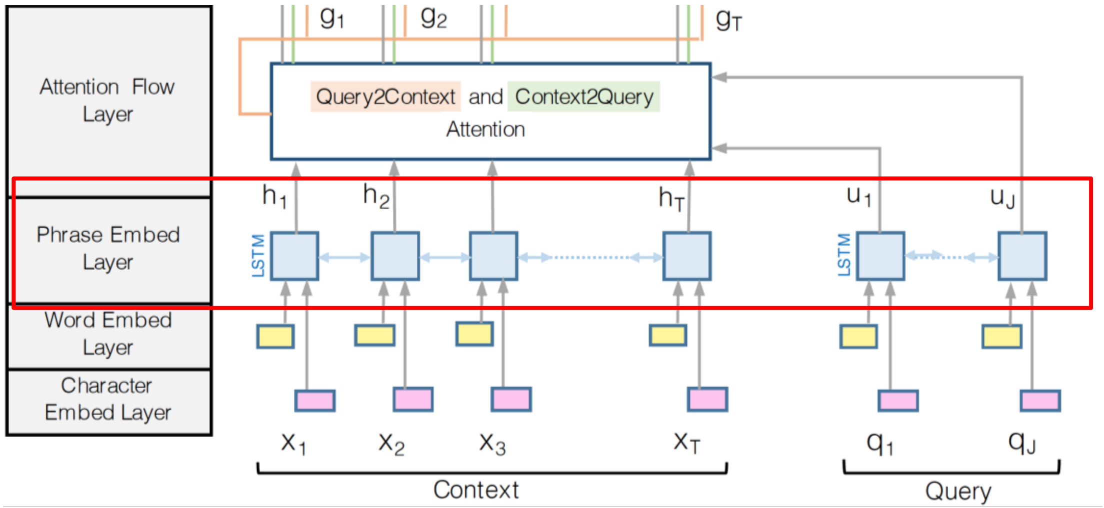
Contextual Embedding Layer – Our improvements



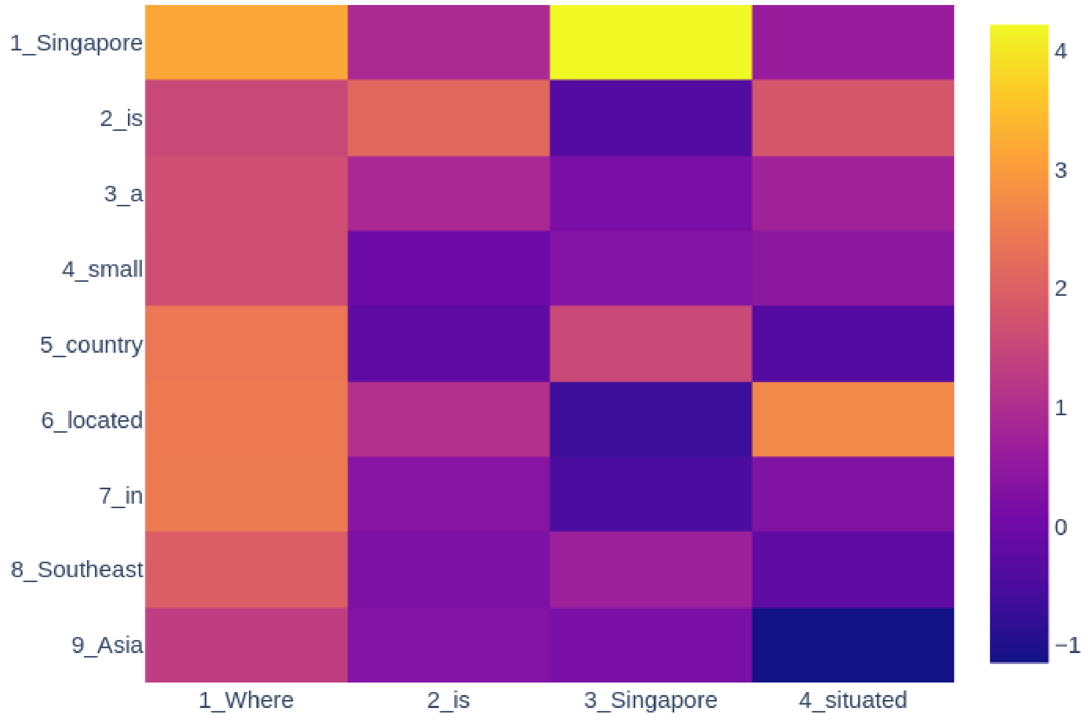
- Our improvements:
  - Two Convolutional Highway Network blocks
  - Transform gate T uses a 2D convolution (vertical slid) and mean operation to obtain t
  - Weighting value t is shared among all the elements of the same batch
  - Transform function H processes input x using 2D convolution with 5x5 kernel

Contextual Embedding Layer - RNN

- The activations from the two branches are added together
- The Highway Network output is processed by a by a bidirectional Recurrent Neural Network (LSTM or GRU)



Attention Flow Layer



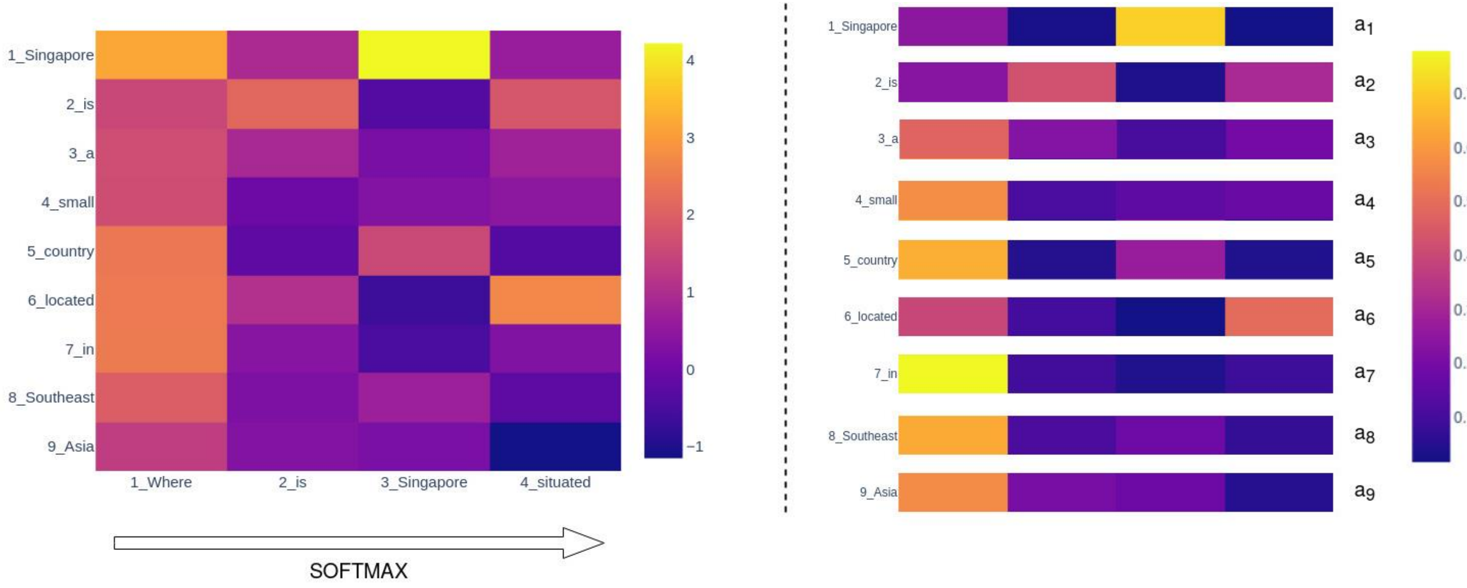
- It merges information between context and query through the **bidirectional attention flow** mechanism.
- Attention is computed both from context to query and vice versa by using a **shared similarity matrix** $\mathbf{S} \in \mathbb{R}^{c \times q}$.
- Given the contextual embeddings of the context $\mathbf{H} \in \mathbb{R}^{2d \times c}$ and the query $\mathbf{U} \in \mathbb{R}^{2d \times q}$, each element of $\mathbf{S}$ is computed as:
  $$\mathbf{S}_{ij} = \mathbf{w}_{(\mathbf{S})}^{\mathrm{T}} \cdot \mathrm{concat}\left(\mathbf{H}_{:i}, \mathbf{U}_{:j}, \mathbf{H}_{:i} \odot \mathbf{U}_{:j}\right)$$
  where $\mathbf{w}_{(\mathbf{S})} \in \mathbb{R}^{6d}$ is a **learnable** weight vector.

Attention Flow Layer – C2Q

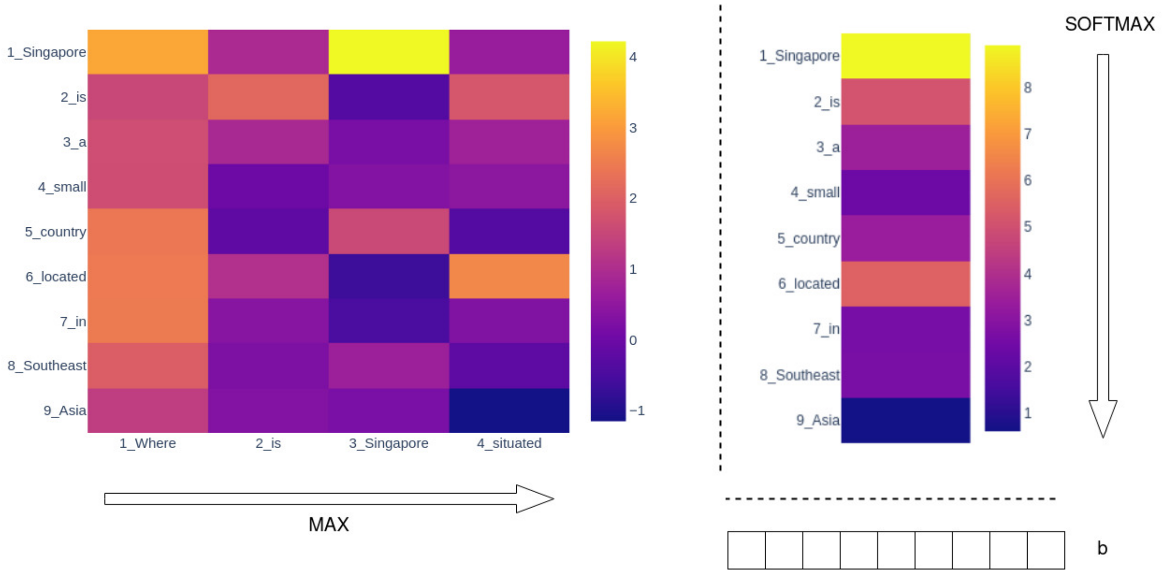
- It determines **which query words are more relevant** w.r.t. each context word.
- For each i-th context word, a vector of attention weights $\mathbf{a}_i \in \mathbb{R}^{q}$ is computed by applying a softmax on the i-th row of $\mathbf{S}$.
- Each column i of the attended query matrix $\tilde{\mathbf{U}} \in \mathbb{R}^{2d \times c}$ is computed as:

$$\tilde{\mathbf{U}} = \sum_{j} \mathbf{a}_{ij} \cdot \mathbf{U}_{:j}$$

- $\tilde{\mathbf{U}}$ contains the **attended query vectors** for the entire context.



Attention Flow Layer – Q2C



- It determines **the context words which are most similar to the query words**, and thus more likely to contain the answer.
- For each context word, we compute the maximum similarity w.r.t. all query words and apply a softmax to obtain the vector of attention weights $\mathbf{b} \in \mathbb{R}^{c}$.
- The vector $\tilde{\mathbf{h}} \in \mathbb{R}^{2d}$ is computed as:

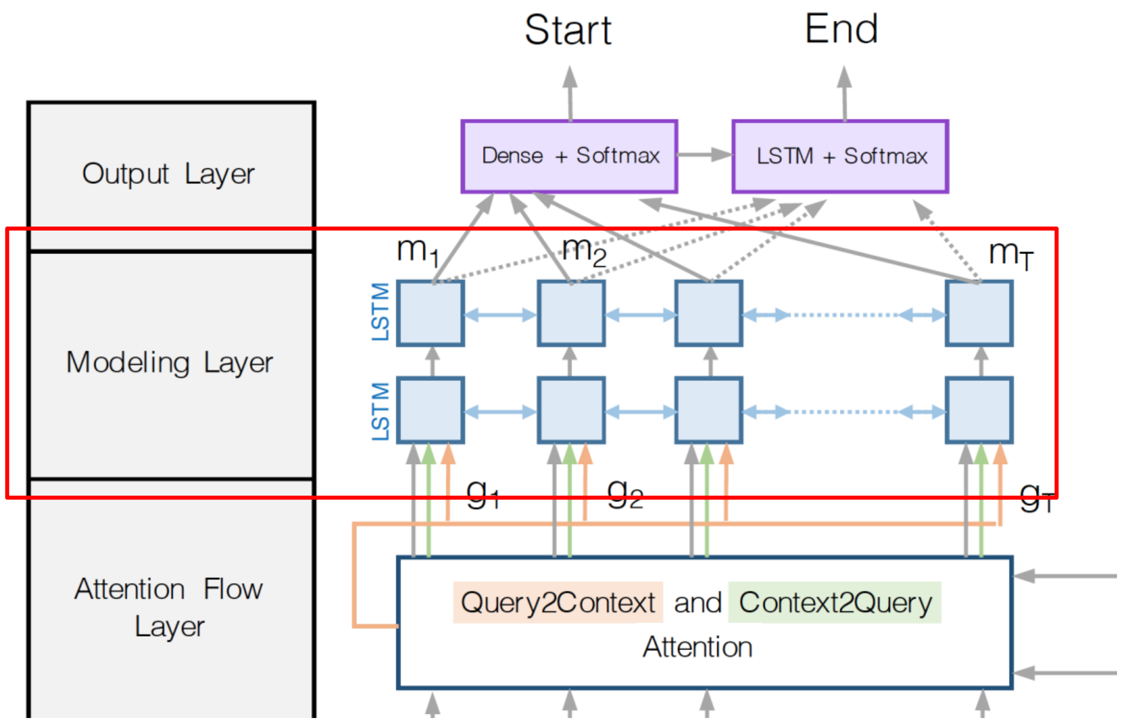
$$\tilde{\mathbf{h}} = \sum_{i} \mathbf{b}_i \cdot \mathbf{H}_{:i}$$

  and it indicates the **weighted sum of the most important words in the context w.r.t. the query**.
- $\tilde{\mathbf{h}}$ is then replicated c times across the columns to obtain the matrix $\tilde{\mathbf{H}} \in \mathbb{R}^{2d \times c}$.

Attention Flow Layer – Q2C

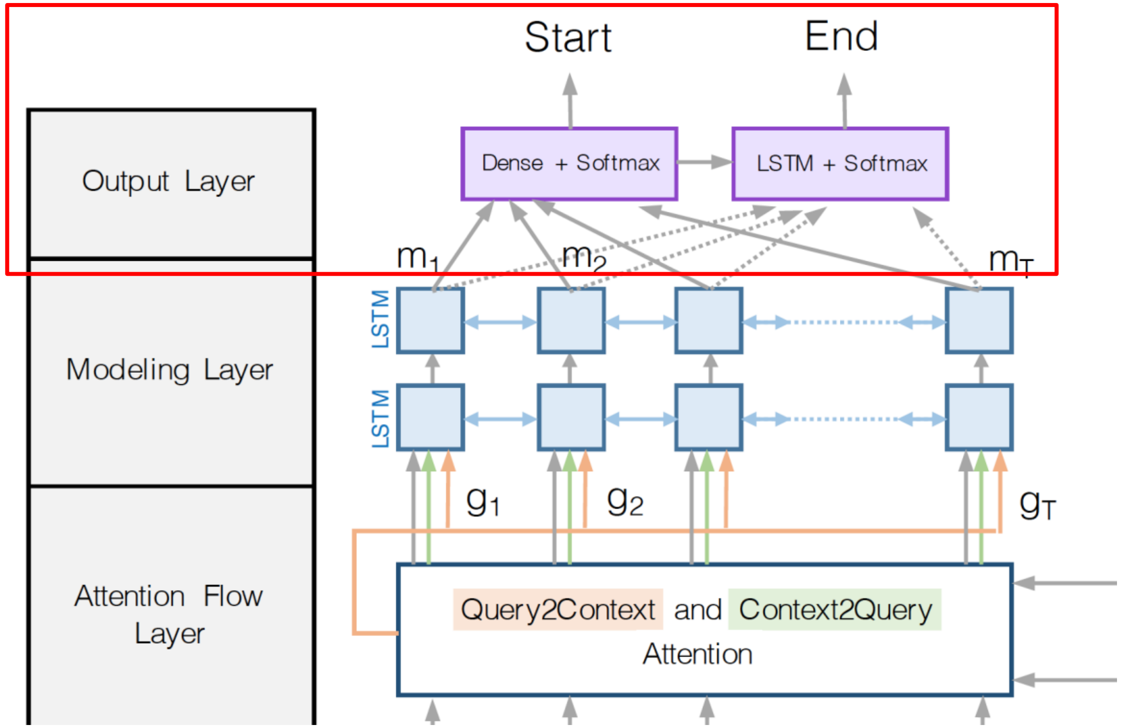
- The contextual embeddings and the attention vectors are combined to yield the matrix $\mathbf{G} \in \mathbb{R}^{8d \times c}$, where each column is the **query-aware representation** of each context word.
- Each column of $\mathbf{G}$ is computed as:
  $\mathbf{G}_{:i} = \mathrm{concat}\left(\mathbf{H}_{:i}, \widetilde{\mathbf{U}}_{:i}, \mathbf{H}_{:i} \odot \widetilde{\mathbf{U}}_{:i}, \mathbf{H}_{:i} \odot \widetilde{\mathbf{H}}_{:i}\right)$

Modelling Layer



- The **input** to the modelling layer is $\mathbf{G}$, which encodes the **query-aware representations of context words**.
- The **output** of the modelling layer captures the **interaction among the context words conditioned on the query**.
- We use a **2-layer bi-directional RNN**, with output size d for each direction.
- Hence we obtain a matrix $\mathbf{M} \in \mathbb{R}^{\mathbf{2}d\times c}$, which is passed onto the output layer.

Output Layers

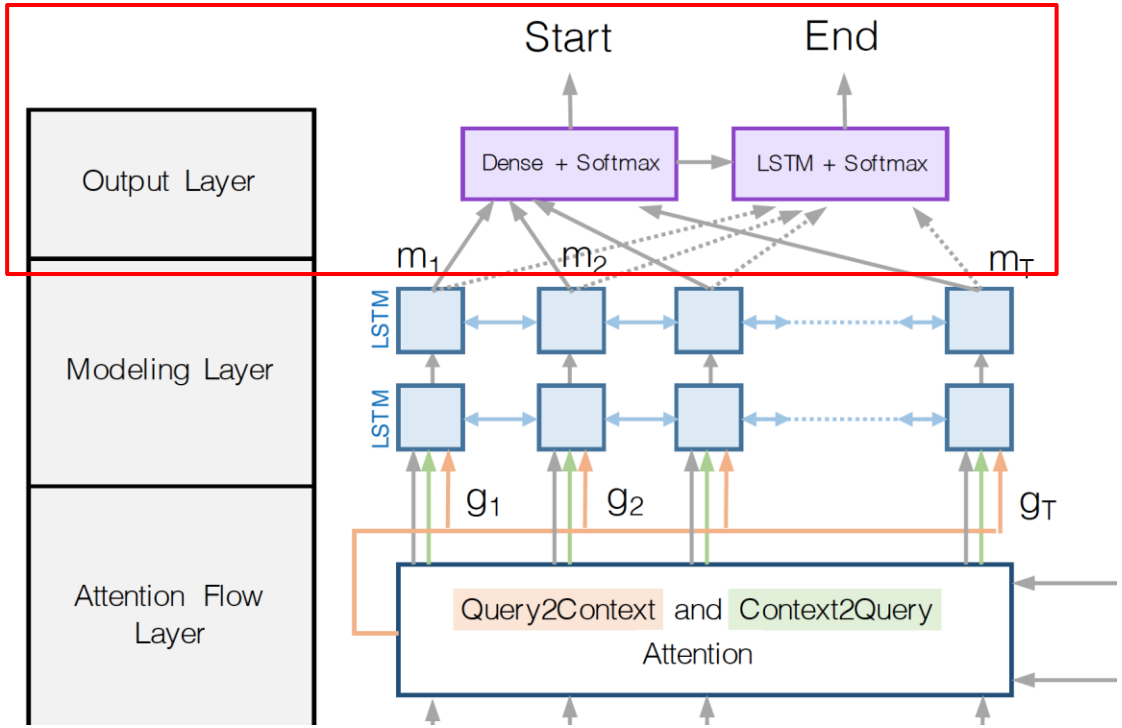


- The probability distribution for the **START** of the answer span is obtained as follows:

$$\mathbf{p}^1 = \mathrm{softmax}\left(\mathbf{w}_{(\mathbf{p}^1)}^{\mathrm{T}} \cdot \mathrm{concat}(\mathbf{G}, \mathbf{M})\right)$$

- $\mathbf{w}_{(\mathbf{p}^1)}^{\mathrm{T}} \in \mathbb{R}^{10d}$ is a **learnable** weight vector.

Output Layers



- The probability distribution for the **END** of the answer span is obtained in two steps:
  - the matrix $\mathbf{M}$ is fed into an additional bi-directional RNN which produces the matrix $\mathbf{M}^2 \in \mathbb{R}^{2d \times c}$;
  - $\mathbf{p}^2 = \mathrm{softmax}\left(\mathbf{w}_{(\mathbf{p}^2)}^{\mathrm{T}} \cdot \mathrm{concat}(\mathbf{G}, \mathbf{M}^2)\right)$
- $\mathbf{w}_{(\mathbf{p}^2)}^{\mathrm{T}} \in \mathbb{R}^{10d}$ is a **learnable** weight vector.

Loss and optimizer

- The loss function sums the negative log probabilities of the true start and end indices:

$$L(\theta) = -\frac{1}{N}\sum_{i}^{N} \log\left(\mathbf{p}^1_{y_i^1}\right) + \log\left(\mathbf{p}^2_{y_i^2}\right)$$

  where:

  - θ is the set of trainable parameters;
  - N is the number of samples;
  - y_i^1 and y_i^2 are the true start and end indices of the answer span for the i-th sample;
  - $\mathbf{p}^1_k$ and $\mathbf{p}^2_l$ are the probability that the k-th token is the start and that the l-th token is the end of the answer span.

- We trained the model using Adam optimizer with a learning rate of $5 \cdot 10^{-3}$ and mini-batches of size 8.

Baseline and variants

- **Baseline**: dense highway network, learnable character embedding layer and dropout
- **Variant 1**: convolutional highway network and concatenation of $\mathbf{p}^{\text{start}}$ when computing $\mathbf{p}^{\text{end}}$
- **Variant 2**: as the variant 1, but with a non-trainable character embedding layer based on the one-hot encoding of the most frequent characters
- **Variant 3**: as the variant 2, but with no dropout
- **Variant 4**: as the variant 3, but with the additional constraint $\mathbf{p}^{\text{end}} > \mathbf{p}^{\text{start}}$ (which acts only at inference)

Performance on test set					
Model	Baseline	Variant 1	Variant 2	Variant 3	**Variant 4**
Loss	5.07	4.73	3.64	3.67	**3.65**
Exact score	0.266	0.288	0.402	0.412	**0.421**
f1 score	0.432	0.443	0.587	0.599	**0.603**

Analysis of results

Error type 1

- Context: *Beyoncè Giselle Knowles-Carted (born September* ***4****,* ***1981****) is an American singer, […] rose to fame in the late* ***1990s*** *[…] Dangerously in Love (****2003****), which established her [...] featured the Billboard Hot* ***100*** *number-one […].*
- Query: *When did Beyoncè start becoming popular?*
- Correct answer: *in the late 1990s.*
- Our answer: *2003).*

Analysis of results

Error type 2

Example1

- Query: *In what city did Beyoncè grow up?*
- Correct answer: *Houston.*
- Our answer: *Houston, Texas.*

Example 2

- Query: *Which three countries did Beyonce's song "Work It Out" achieve top ten status?*
- Correct answer: *UK, Norway,and Belgium.*
- Our answer: *Belgium.*

Analysis of results

Error type 3

Example 1

- Context: *The latest study using magnetic resonance imaging (MRI) to humans and dogs together proved that […].*
- Query: *What technology was used to show that dogs respond to voices in the same brain parts as people?*
- Correct answer: *MRI.*
- Our answer: *magnetic resonance imaging.*

Example 2

- Context: *In Islam dogs are viewed as unclean because they are viewed as scavenger. In 2015 [...].*
- Query: *How are dogs viewed in Islam?*
- Correct answer: *as unclean.*
- Our answer: *as scavengers.*

Discussion and possible improvements

- Evidence shows that exploiting high-level constraints (like start span index < end span index) can lead to improvements
- The output layer can learn to constraint the value of p^{end} based on the value of p^{start}
- Possible improvement: merge matrices (eg. word and char embedding) with other approaches instead of concatenation, such as sum or weighted sum
- The models could be improved by better exploring the hyperparameters' space

Discussion and possible improvements

- Major weakness: models not suited for parallelized training due to RNN
- Baseline model slowed down by Dense Highway Network in the Contextual Embedding Layer

- We introduced the more efficient Convolutional Highway Network

Discussion and possible improvements

- Other architectures:
  - Transformers which do not employ recurrent modules would be trained more efficiently
  - XLNet seems the most suited solution because it does not have the limitations of BERT regarding the maximum input length, it matters since in SQuAD sometimes (context + query) > 512

Thank you for your bidirectional attention