GRU model and attention explanation:

High level overview of the model:

- Input layer: input sentence to the model;
- Embedding layer: map each word into a low dimension vector
- GRU layer: utilize bi directional GRU to get high level features from the second step above
- Attention layer: produce a weight vector, and merge word-level features from each time step into a sentence-level feature vector, by multiplying the weight vector
- Output layer: the sentence-level feature vector is finally used for relation classification

The GRU network used contains two sub-networks for the left and right sequence context, which are forward and backward pass respectively. The output of the i th word is given by:

$$h_i = [\overrightarrow{h_i} \oplus \overleftarrow{h_i}]$$

I used element-wise sum to combine the forward and backward pass outputs.

- Used inbuilt *tf.keras.layers.GRU* from keras to define the GRU layer with a hidden size of 128.
- For bi-directional GRU used *tf.keras.layers.Bidirectional* from keras with merge mode as concat.

Attention:

The GRU produced an output matrix having output vectors as [h1, h2, . . . , hT]. Let us say the matrix as H and where T is the sentence length in the last output vector. The representation *r* of the sentence is formed by a weighted sum of these output vectors as shown by equations below:

$$M = \tanh(H)$$
$$\alpha = softmax(w^T M)$$
$$r = H\alpha^T$$

In the above equations, *H* is the dimension of the word vectors, *w* is a trained parameter vector and *w^T* is its transpose. The dimension of *w is d^w, α is T and r is d^w*. To get the final sentence-pair representation the below form is used:

$$h^* = \tanh(r)$$

Used inbuilt tensorflow functions to implement the above equations in my model.

Observations for experiment 1, 2 and 3

experiment number	experiment_name	epoch_num	train_loss	val_loss	f1_score
1	**Only word embeddings**	1	2.09	1.95	0.34
		2	1.23	1.64	0.47
		3	0.5	1.78	0.48
		4	0.19	2.01	**0.48**
		5	0.09	2.4	**0.48**
2	**word+Pos embeddings**	1	2.4	2.29	0.34
		2	1.48	1.81	0.42
		3	0.83	1.72	0.48
		4	0.39	1.87	0.51
		5	0.17	2.07	**0.52**
3	**word+Dep structure**	1	1.85	1.99	0.32
		2	1.06	1.68	0.51
		3	0.54	1.67	0.58
		4	0.27	1.85	0.58
		5	0.18	2.02	**0.59**

Observation for the baseline: using word + embedding + dependency structure

experiment_name	epoch_num	train_loss	val_loss	f1_score
Baseline (word embeddings + POS embeddings + dep structure)	1	1.99	2.25	0.37
	2	1.28	1.84	0.45
	3	0.74	1.76	0.51
	4	0.38	1.87	0.56
	5	0.22	2.05	**0.6**

From the experiments conducted above, it is observed that the **baseline model performs the best**. Word embeddings, position embeddings and dependency structure combined give the best results.

As expected dependency structure is a better feature than position embeddings as it helps establish relations between terms which is the primary task for semval i.e relation extraction.

- Word + dependency structure performs second best.
- Word embedding used: glove, of dimension 100
- Batch size: 5
- Epochs: 5

Advanced Model:

Motivation: Previous basic models defined rely on high-level lexical and syntactic features obtained by NLP tools such as WordNet, dependency parser, part-of-speech (POS) tagger. These neural models do not fully utilize information of entity that may be the most crucial features for relation classification. To address these issues, I used a novel end-to-end recurrent neural model defined by [1] which incorporates an entity-aware attention mechanism with a latent entity typing (LET) method. This model not only utilizes entities and their latent types as features effectively but also is more interpretable by visualizing attention mechanisms.

The model consists of four main components:

- Word Representation that maps each word in a sentence into vector representations;
- Self Attention that captures the meaning of the correlation between words based on multi-head attention
- BLSTM which sequentially encodes the representations of self attention layer
- Entity-aware Attention that calculates attention weights with respect to the entity pairs, word positions relative to these pairs, and their latent types obtained by latent entity typing.

Attention component:

One additional component in this model used is multihead attention. In multi-head attention, the scaled dot-product attention with linear transformations are performed on *r* parallel heads to pay attention to different parts. Then formulation of multi-head attention is defined by the following:

$$M = \tanh(H)$$
$$\alpha = softmax(w^T M)$$
$$r = H\alpha^T$$

Because this work requires self attention, the input matrices of multi-head attention, Query, Key, and Value are all equivalent to X, the word representation vectors. The output of self attention layer is the sequence of representations whose include informative factors in the input sentence

README for advanced:

LSTM network:

For sequentially encoding the output of self attention layer, I have used a bi-directional LSTM that consists of two sub LSTM networks: a forward LSTM network which encodes the context of an input sentence and a backward LSTM network which encodes one of the reverse sentences. More formally, BLSTM works as follows:

$$\overrightarrow{h_t} = \overrightarrow{LSTM}(m_t)$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}(m_t)$$

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$$

The representation vectors obtained from self attention layer are forwarded into the network step by step. At the time step t, the hidden state of a BLSTM is obtained by concatenating, the hidden state of forward LSTM network and the backward one.

Diagram for the bi-directional LSTM network:

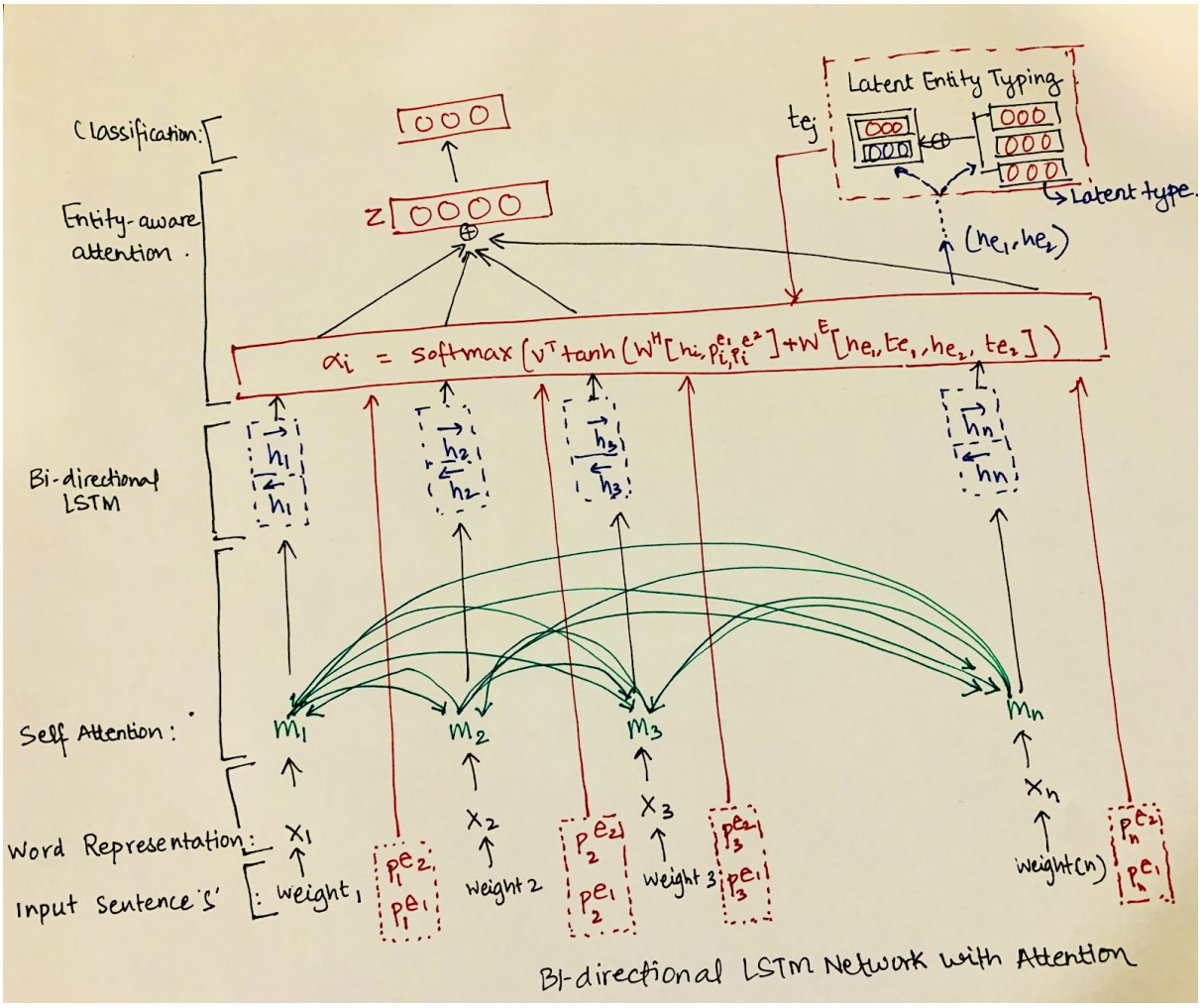

Figure 1: Bi-directional LSTM network with attention

Hyperparameters used in the model

parameters	value
word embedding used	glove.840B.300d.txt
embedding dimension used	300
number of heads	4
size of hidden layer	300
batch size	20
L2 regularization coefficient	0.00001

Evaluation:
The model was performing well for word + dependency structure embeddings and the configurations mentioned above. Below is one of the evaluation results obtained by the model.

experiment_name	epoch_num	train_loss	val_loss	f1_score
word+Dep structure	1	1.87	2.01	0.3
	2	1.12	1.8	0.49
	3	0.56	1.77	0.54
	4	0.29	1.96	0.51
	5	0.2	1.91	**0.55**