Hybrid Embeddings with Attention Mechanism and Autoencoded Meta-Embedding

Final Presentation

Rameez Mohammed QURESHI - Asmaa DEMNY - Fatima HABIB - Tuan-Anh VO

26/01/2021

IDMC, University of Lorraine

Outline

Background

Seq2Seq architecture

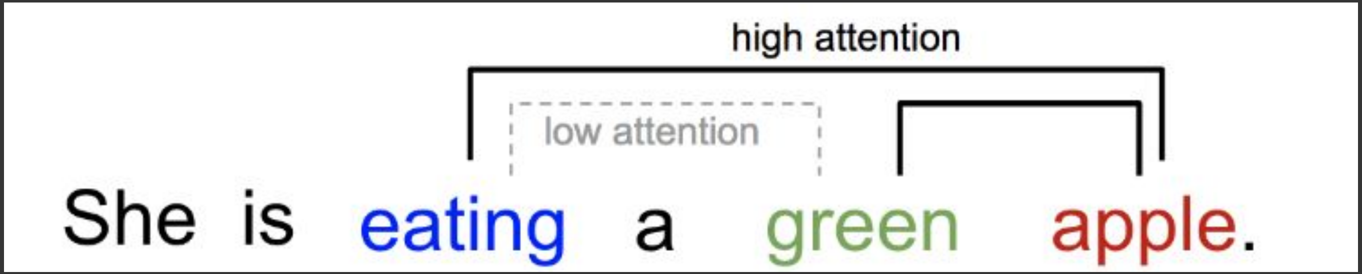
- Originally invented in Language Modeling
- Encoder-decoder architecture, the encoder compress all the information from the **input sequence (source)** into a context vector of fixed length, then the vector serve as initialiser to the decoder to generalise output sequence
- Inadequate when it comes to tasks with a long sequence,
- *Bahdanau et al. (2015)* suggested that not only all the input words be taken into account in the context vector, but also their relative importance

Attention mechanism

In the context of neural networks, it can be described as a vector of importance weights.

Allows the model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly.

With the help of the attention, **the dependencies** between source and target sequences are not restricted by the in-between distance anymore



Transformer and Multi-head-attention

Transformer architecture: first introduced in the paper "Attention is all you need" by Vaswani *et al.* 2017

It relying **entirely on self-attention** to compute representations of its input and output without using sequence-aligned RNNs or convolution

Self-attention: is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence."

Multi-headed Attention: or *"scale dot-product attention"*, in which self-attention is computed not once but multiple times in the Transformer's architecture, in parallel and independently

The independent attention outputs are concatenated and linearly transformed into the expected dimensions, through the mechanism of ***Key, Query and Value***

Key, Query and Value

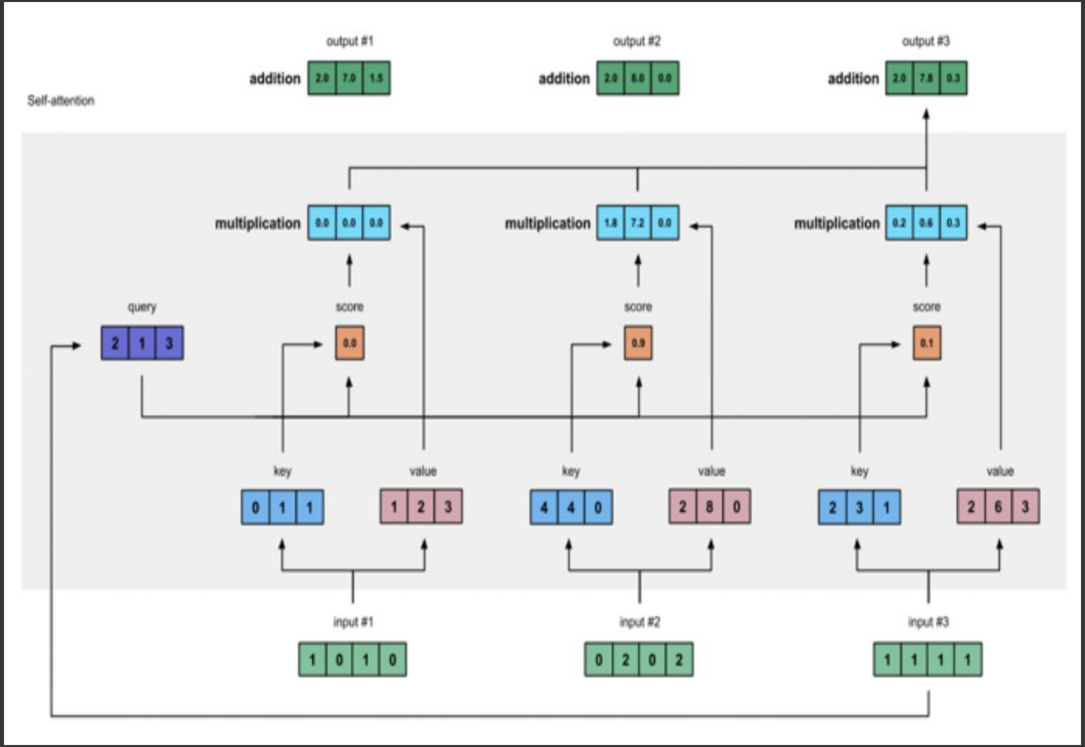
The query is from the decoder hidden state and the key and value are from the encoder hidden states

Every input (green) is multiplied with a set of weights for all the keys, queries, values

The weight assigned to each value is computed by a compatibility function (dot product) of the query with the corresponding key: the score

The scores then go through the softmax function to yield a set of weights whose sum equals 1.

Each weight multiplies its corresponding values to yield the context vector which utilizes all the input hidden states



Meta Embedding

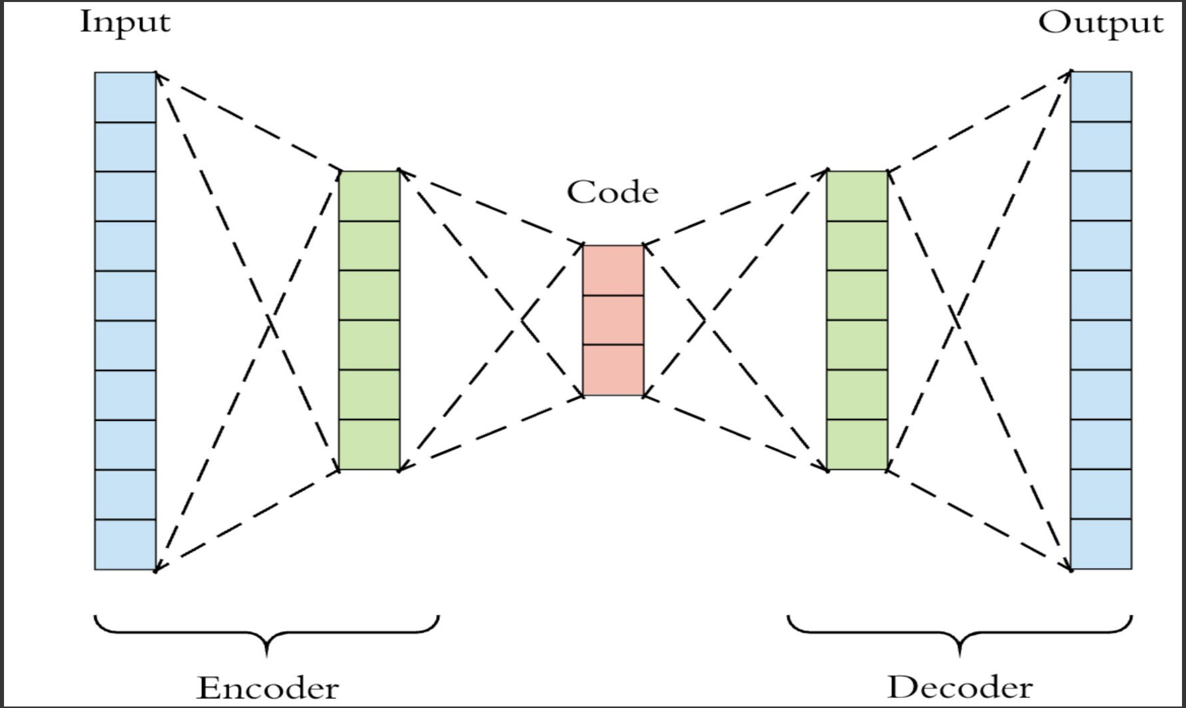
- Combine different approaches of embeddings .
- The goal of the meta-embedding learning is to learn a single (possibly lower-dimensional) common,multiple, pre-trained source word embeddings without retraining.

- The most straightforward meta-embeddings approaches are **Concatenation (CONC)** and **Averaging (AV)**.

- Concatenation was performed in [1] ,simple concatenation between (BigGraph and ELMo) (our baseline).

Auto-Encoder

- An autoencoder is a feedforward neural network
- Generalization of **PCA**.
- It is used in **Representation Learning** .


Input
Output
Code
Encoder
Decoder

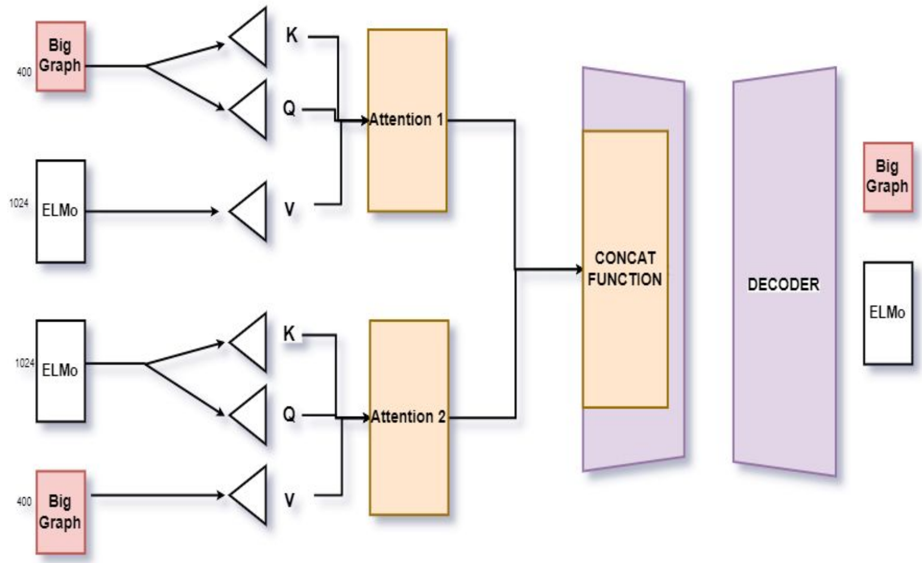
Methodology

“Parallel” Multi Head Self Attention

In the encoder, we add two different multihead attention layers

Function as a kind of “filter” that would hypothetically help us to ‘attend’ to more relevant information from the input vectors, in our case are two source embeddings ELMo and BigGraph.

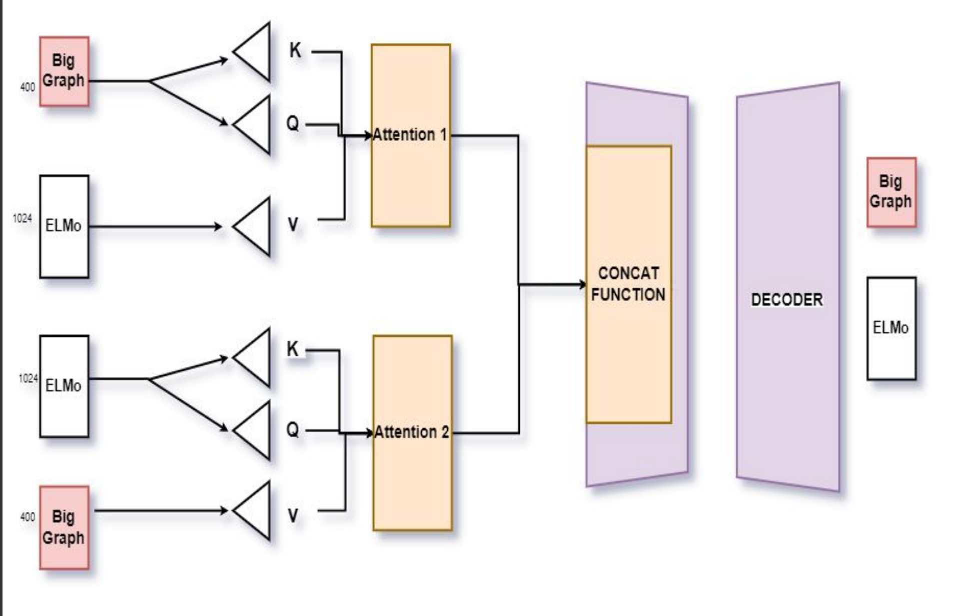
The “parallel mechanism” : arrange exact opposite orders, when assigning the key, query and value of each attention layer to the source embeddings



“Parallel” Multi Head Self Attention

The first layer, the score comes from Big Graph Embedding. It then goes through the softmax function to yield an attention weight, which then be multiplied by the corresponding value come as input from ELMo embedding, to yield a context vector.

The process is reversed in second attention layer, the score now comes from ELMo, and the value from BigGraph.

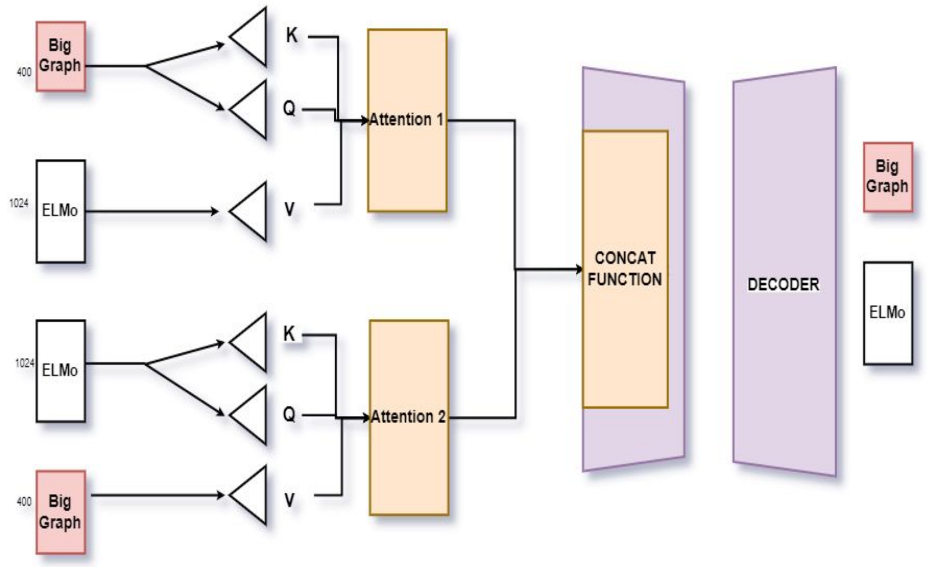


“Parallel” Multi Head Self Attention

The two different context vectors of the same size then serve as the input of a simple concatenation function that we have seen in Dieudonat et al (2020) , which is then embedded in the encoder, before further implemented in different auto encoded-meta embedding models.

Major inspiration: the capability of the Transformer’s multi-head attention mechanism to have key, query, and value come from different sources inputs

Changing the order could yield different attention weights which hypothetically could attend the relevant information from both textual and knowledge graph resources, via our source embeddings.



Dataset (FreeBase15k)

#	Count
Train samples	11607
Test samples	1268
Sentences	13,874
Entities	798
Relations	16
Samples	29,492

Experiments

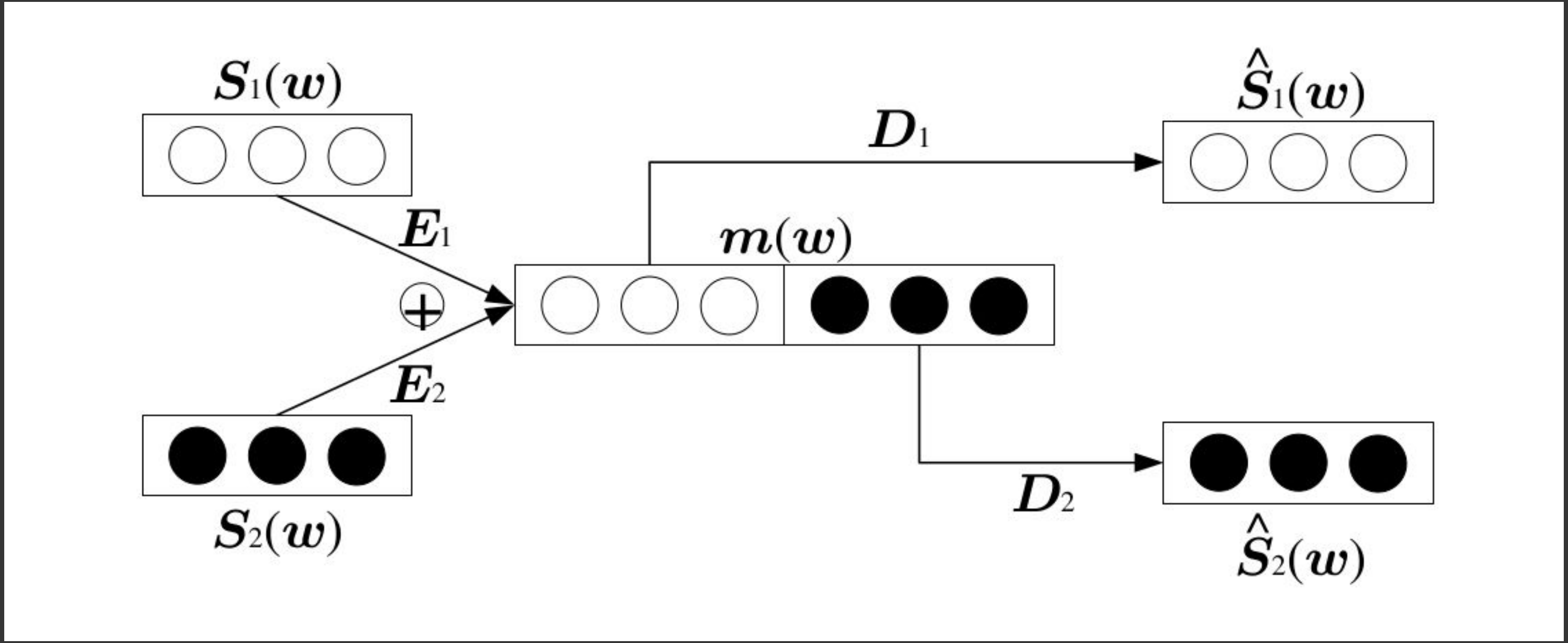
Auto-Encoded Meta-embedding

- Representing the meaning of individual words
- Generating a single (meta) word embedding from a given set of pre-trained input (source) word embeddings
- Three Autoencoded methods : DAEME, CAEME,AAEME

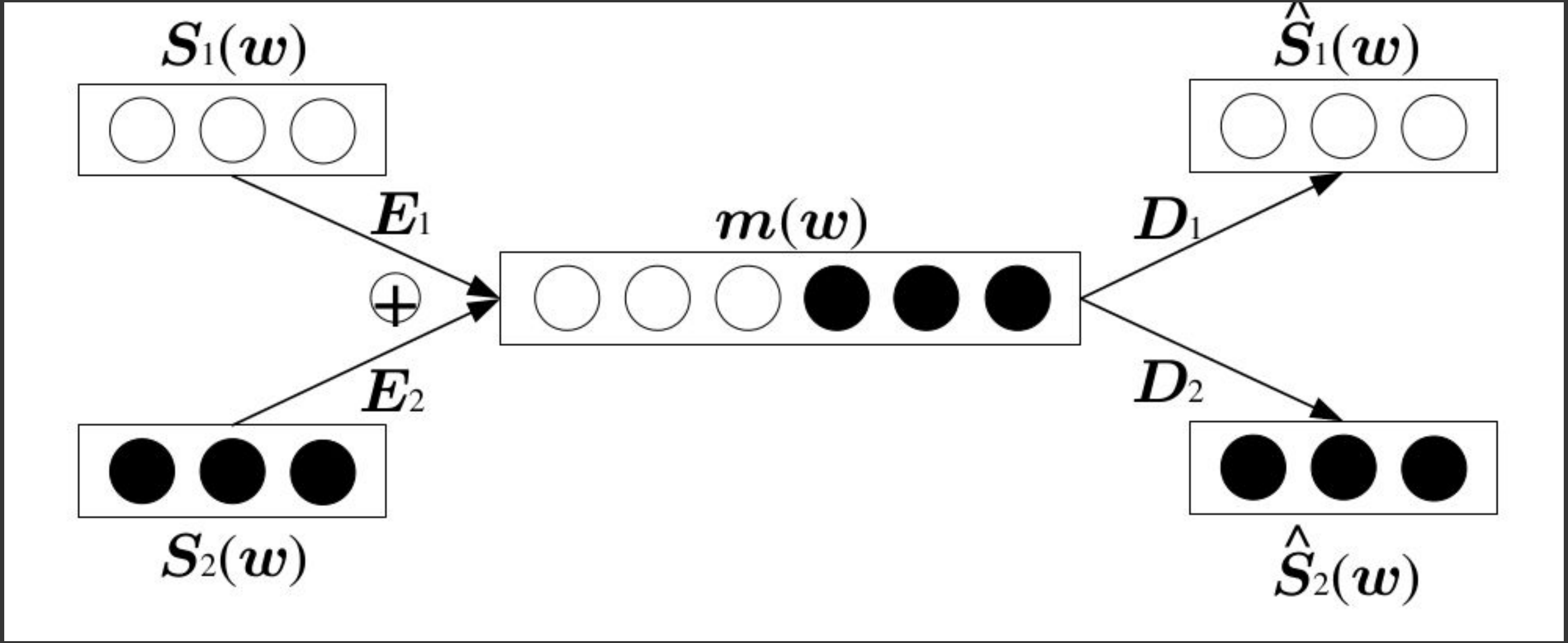
Chosen Embeddings

- ELMo (Contextual word embeddings)
- ComplEX (knowledge graph embedding)

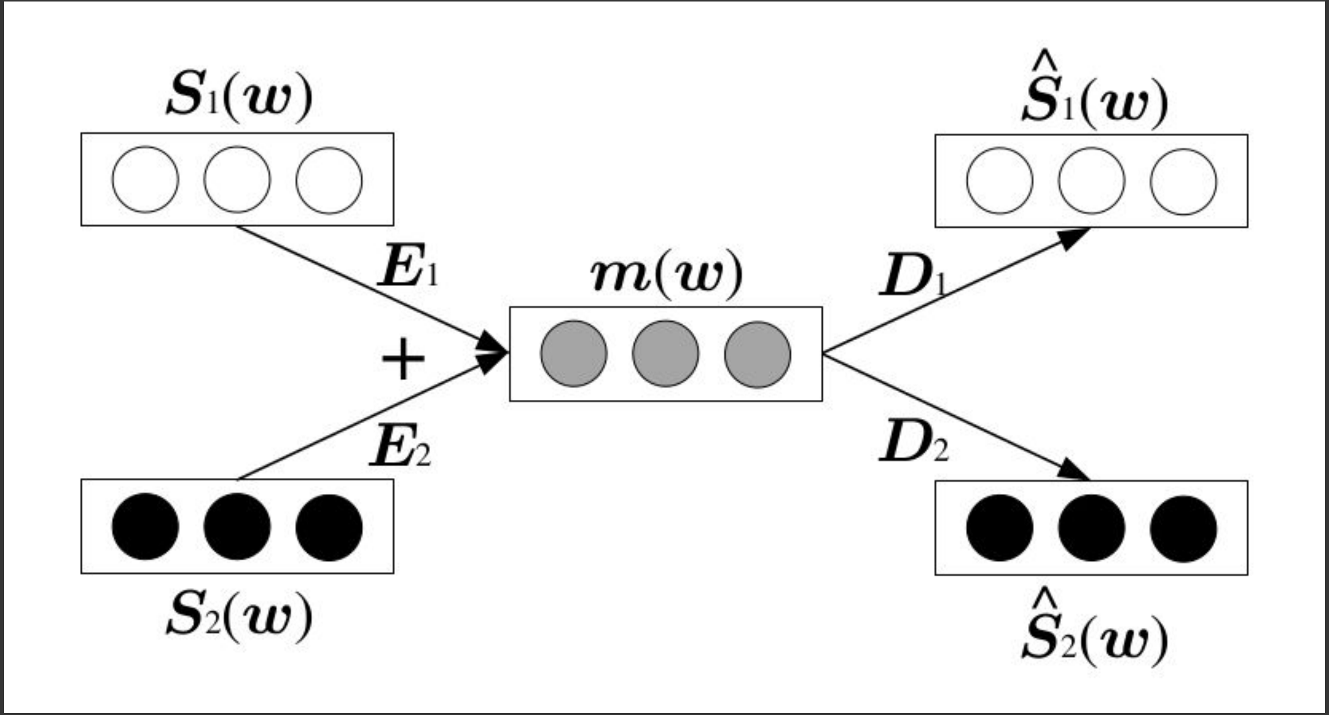
Decoupled Auto Encoded Meta-Embedding (DAEME)



Concatenated Auto Encoded Meta-Embedding (CAEME)



Averaged Auto Encoded Meta-Embedding (AAEME)



Evaluation Metrics

1. **Precision @n : n is the number of the predictions**

$$\frac{no.\ of\ \ correct\ predictions\ up\ to\ n}{n}$$

2.Mean Average Precision@k

$$AP@\mathrm{K} = \frac{1}{\mathrm{m}} \sum_{\mathrm{i}=1}^{\mathrm{k}} \mathrm{P}\left(\mathrm{label}_{\mathrm{k}}\right),\ \text{if K}^{\text{th}}\ \text{item is relevant}$$

3. Mean Reciprocal Rank

$$MRR = \frac{1}{N} \sum_{i=1}^{N} RR\left(\text{output}_i\right)$$

where N is the total number of samples in the dataset.

Results

Entity Typing

Model	MAP@k=10	Precision@k=10 (mean ± std)
Previous Work		
Contextual Embeddings (1)	0.631	0.449 ± 0.271
KG Embeddings (2)	0.825	0.528 ± 0.269
Concatenation (1) + (2)	**0.828**	0.527 ± 0.268

Entity Typing

Model	MAP@k=10	Precision@k=10 (mean ± std)
PCA reduction to 200 dimension		
Contextual Embeddings	0.190	0.208 ± 0.249
KG Embeddings	<u>0.673</u>	0.476 ± 0.261
Concatenation	0.297	0.306 ± 0.277
Auto Encoded Meta Embeddings (AEMEs) with 400 dimensions		
DAEME	0.101	0.200 ± 0.282
CAEME	0.137	0.246 ± 0.299
AAEME	0.125	0.198 ± 0.212

Entity Typing

Model	MAP@k=10	Precision@k=10 (mean ± std)
Previous Work		
Contextual Embeddings (1)	0.631	0.449 ± 0.271
KG Embeddings (2)	0.825	0.528 ± 0.269
Concatenation (1) + (2)	**0.828**	0.527 ± 0.268

Attention Based Auto Encoder Model		
DAEME (400)	0.602	0.461 ± 0.283
CAEME (400)	0.775	0.514 ± 0.271
AAEME (400)	0.277	0.283 ± 0.271

Relation Prediction

Model	MRR	Precision (MAP@k=1)
Previous Work		
Contextual Embeddings (1)	0.750	0.554
KG Embeddings (2)	**0.817**	0.663
Concatenation (1) + (2)	0.738	0.533

Relation Prediction

Model	MRR	Precision (MAP@k=1)
Previous Work		
Contextual Embeddings (1)	0.750	0.554
KG Embeddings (2)	**0.817**	0.663
Concatenation (1) + (2)	0.738	0.533
PCA reduction with dimension 400		
Contextual Embeddings	0.656	0.433
KG Embeddings	0.750	0.557
Concatenation	0.708	0.481

Relation Prediction

Model	MRR	Precision (MAP@k=1)
Previous Work		
Contextual Embeddings (1)	0.750	0.554
KG Embeddings (2)	**0.817**	0.663
Concatenation (1) + (2)	0.738	0.533
Attention Based Auto Encoder Model		
DAEME (400)	**0.829**	0.687
CAEME (400)	0.816	0.666
AAEME (200)	**0.820**	0.671
AAEME (400)	0.809	0.653

Conclucion

- PCA did not offer significant changes .
- Concatenation still the best model in Entity Typing.
- **Multi-head self-attention** in Relation Typing.
- Able to solve the pertaining problem with relation prediction!

Future Work!

Focusing on the entity typing task to understand substandard results

Hyperparameter tuning

- Random initialisation.
- Transfer learning.
- Pretrained model.
- Different dataset.

Additive attention instead of averaging and concatenation of encoder outputs.

Thank you!

Appendix

Dimension	MRR	MAP@1
400	0.708	0.481
800	0.716	0.493
1200	0.710	0.476
2848 (w/o PCA)	0.738	0.533

Table: Results for the relation prediction task with PCA applied to the concatenation model.

References

1. Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.
2. Vaswani, A., et al. (2017). Attention is all you need. In Advances in neural information
3. processing systems (pp. 5998–6008).
4. Bank, D., Koenigstein, N., & Giryes, R. (2020). Autoencoders.
5. Dieudonat, L., Han, K., Leavitt, P., & Marquer, E. (2020). Exploring the Combination of Contextual Word Embeddings and Knowledge Graph Embeddings.
6. Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.
7. Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. Advances in neural information processing systems, 27, 3104–3112.