Results of different models:

N	SAINT						LTMTI					
	2		3		4		2		3		4	
d_model	256	512	256	512	256	512	256	512	256	512	256	512
ACC	0.7349	0.7355	0.7360	0.7359	0.7362	**0.7368**	0.7339	0.7281	0.7317	0.7273	0.7333	0.7252
AUC	0.7784	0.7791	0.7799	0.7798	0.7803	**0.7811**	0.7762	0.7673	0.7726	0.7656	0.7755	0.7642

N	UTMTI						SSAKT					
	2		3		4		2		3		4	
d_model	256	512	256	512	256	512	256	512	256	512	256	512
ACC	0.7323	0.7220	0.7301	0.7163	0.7323	0.7204	0.6744	0.6699	0.7322	0.7300	0.7340	0.7221
AUC	0.7734	0.7577	0.7699	0.7493	0.7735	0.7565	0.6302	0.6057	0.7763	0.7722	0.7777	0.7579

Methods	ACC	AUC
MLP	0.7052	0.7363
NCF	0.7051	0.7341
NPA	0.7290	0.7656
SAKT	0.7271	0.7671
LTMTI	0.7339	0.7762
UTMTI	0.7323	0.7735
SSAKT	0.7340	0.7777
SAINT	**0.7368**	**0.7811**

Table 2. Comparison of the current state-of-the-art models and SAINT

Table 2. Evaluation metrics of different deep learning methods

Model	AUC	Model	AUC
BKT	0.6325 ± 0.0011	DKT	0.8324 ± 0.0031
EKTA	0.8384 ± 0.0036	EHFKT_S	0.8407 ± 0.0016
EHFKT_K	0.8371 ± 0.0022	EHFKT_D	0.8382 ± 0.0035
EHFKT_T	0.8445 ± 0.0025	EHFKT	$\mathbf{0.8505 \pm 0.0021}$

aixuexi.

Model	ASSIST09	ASSIST12	EdNet
BKT	0.6476	0.6159	0.5621
DKT	0.7356	0.7013	0.6909
DKVMN	0.7394	0.6752	0.6893
KTM	0.7500	0.6948	0.6855
DKT-Q	0.7244	0.6899	0.6876
DKVMN-Q	0.7405	0.6812	0.7152
DHKT	0.7544	0.7213	0.7245
PEBG+DKT	0.8287	0.7665	**0.7765**
PEBG+DKVMN	**0.8299**	**0.7701**	0.7757

Table 2: The AUC results over three datasets.

Table 2: Dataset Statistics

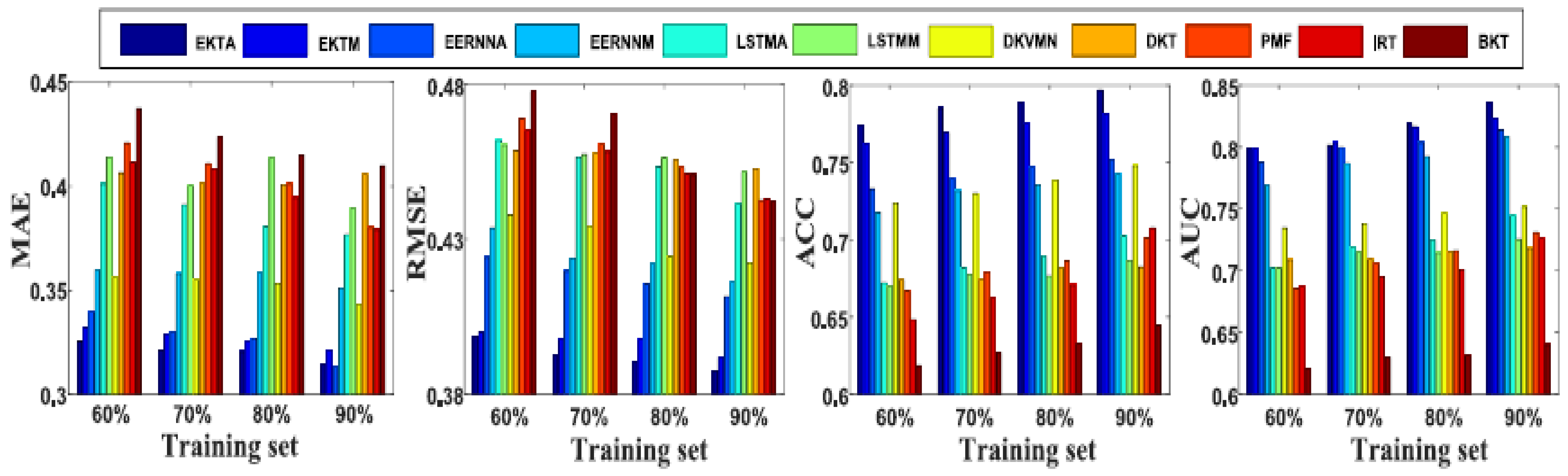
Datasets	#Users	#Skill tags	#Interactions	#Unique Interactions	Density
Synthetic-5	4000	50	200K	200K	1
ASSIST2009	4417	124	328K	35K	0.06
ASSIST2015	19917	100	709K	102K	0.05
ASSIST-Chall	686	102	943K	57K	0.81
STATICS	333	1223	190K	129K	0.31

The columns corresponding to #Users, #Skill tags and #Interactions

Table 3: Student Performance prediction comparison.

Datasets	AUC				
	DKT	DKT+	DKVMN	SAKT	Gain%
Synthetic	0.823	0.824	0.822	**0.832**	0.97
ASSIST2009	0.820	0.822	0.816	**0.848**	3.16
ASSIST2015	0.736	0.737	0.727	**0.854**	15.87
ASSISTChall	**0.734**	0.728	0.689	**0.734**	0.00
STATICS	0.815	0.835	0.814	**0.853**	2.16
Average	0.786	0.789	0.773	**0.824**	4.43

	ASSIST2012		POJ		Junyi	
	AUC	ACC	AUC	ACC	AUC	ACC
DKT	0.712	0.679	0.656	0.691	0.814	0.744
SAKT	0.735	0.692	0.696	0.705	0.834	0.757
DKVMN	0.701	0.686	0.704	0.700	0.822	0.751
DKT+Forget	0.722	0.685	0.662	0.700	0.840	0.759
EERNN	0.748	0.698	0.733	0.720	0.837	0.758
EKT	0.754	0.702	0.737	0.729	0.842	0.759
RKT	**0.793**	**0.719**	**0.827**	**0.774**	**0.860**	**0.770**
Gain%	5.172	2.422	12.212	6.173	1.775	1.050



Zhixue.com

From:

Assistments 2009: We can see how PEBG improves previous models, showing importance of context and time similarity.
At the same time using SAKT (Self Attentive KT) improves results, allowing to infer that transfomers have better results than previous models.
There is an incoherence between results of DKVM on this dataset, which is much bigger in SAKT paper, so probably results of SAKT are as good as PEBG ones.

Assistments 2012: We can observe that RKT (Relation Aware Self Attentive KT) and PEBG improves performances. In particular RKT uses features generated by PEBG, so it can be considered as an application of self-attention to PEBG features using temporal similarity too.
RKT is in general a very good model to try.

Ednet: This dataset does not have textual information of problems, so it will not be useful for our application of NLP to KT.
Despite it, we can still use this dataset to compare previous models and infer which are the best (non-textual) prediction models:

- We can see how SAINT is best model for predictions between itself, LTMtI, UTMTI, SSAKT.
- At the same time similar performances are reached by PEBG+DKVM too, proving how this model produces better features than other ones.

In conclusion it can be clever trying using both PEBG, SAINT and try combining them.

aixuexi: This dataset is used only in this paper, so it is not worth the attention, but we can see how EHFKT improves results of DKT, despite using the same structure of DKT for prediciton.
This method they used is the same used by me, despite they tried to predict difficulty and knowledge distribution. Since my results are better than their, I think they took wrong choice to try predicting Knowledge Distribution, Difficulty and Semantic Features, losing the power of Bert Encodings.
Anyway the idea of generating encodings of text using BERT is clever and I already tried in March, I think I should include it to compare the power of representation of BERT with Countvectorizer and Word2Vec.

Zhixue.com:

This results show how EKT is actually a clever model, infact it is based on attention mechanisms with inputs which are embeddings obtained from word2vec.

POJ and Junyi: The results on this datasets are given only for RKT. They do not add any information relevant not given already by assistment 2012, but they can be used to compare effectiveness of models with RKT.

Availability of code for each model:

- RKT:

    the code is available, I have already modified it and it should require some days to make it   work on my dataset.

- SAINT - LTMTI - UMTI - SSAKT:

    I have found a pytorch implementation, I have adapted it to Assistments 2012 during these days, but I am not sure about its effectiveness, because it does not reach results described by paper "Toward an appropriate…."

- Exercise Hierarchical Feature Enhanced Framework (EHFKT)

    No code is available online. As previosuly said, BERT encodings are worth, while a general (and not detailed) describtion of KDES (knowledge Distribution Extraction System), DFES, and SFES is given. Then they are given as input to a DKT.   In my opinion these methods are not worth, but we can use BERT encodings as inputs to other models.

-Exercise Aware KT:

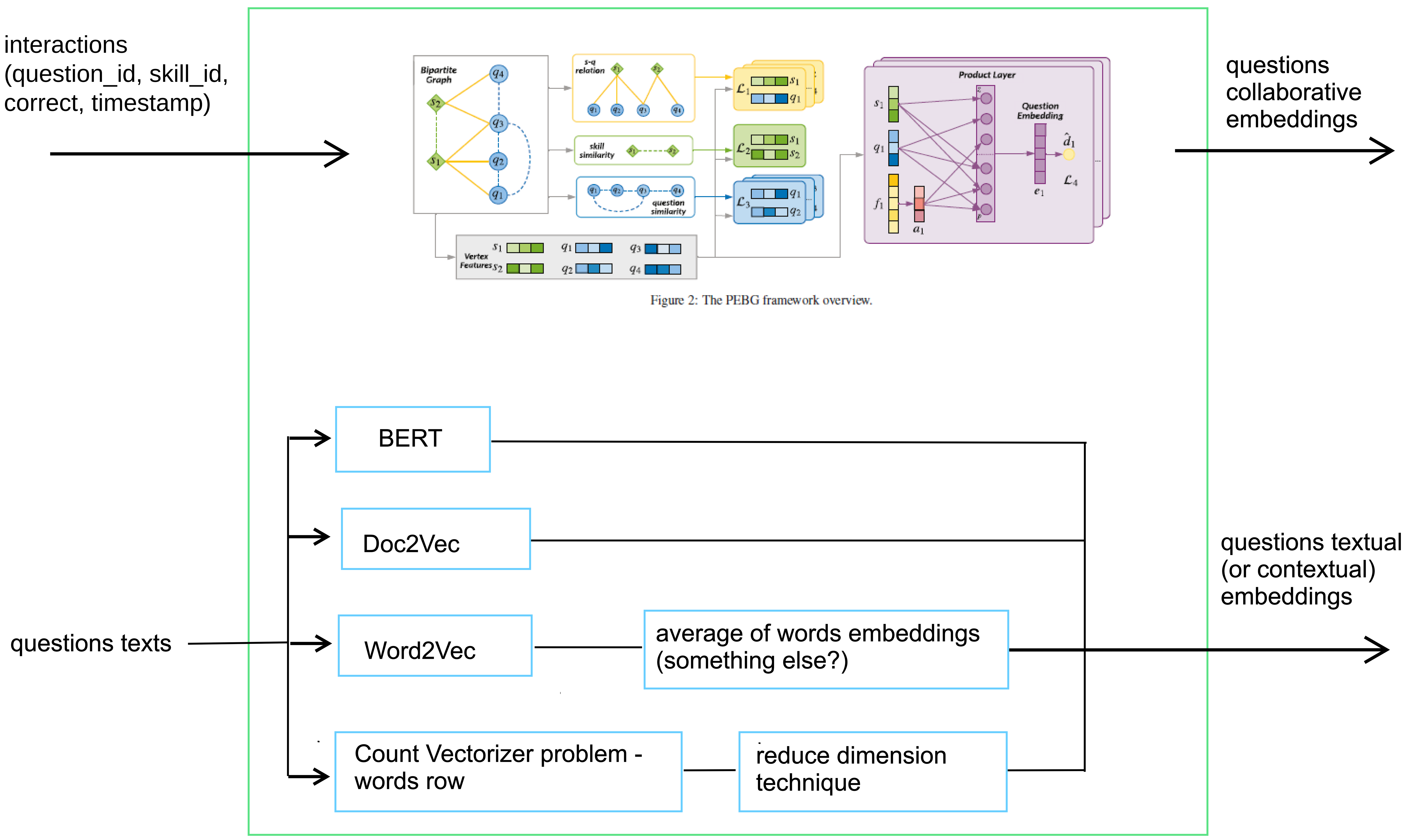
This model uses word2vec encodings as input to attention mechanisms or RNN, which is what i tried doing.   The main limitation of this paper is the use of word2Vec only and of

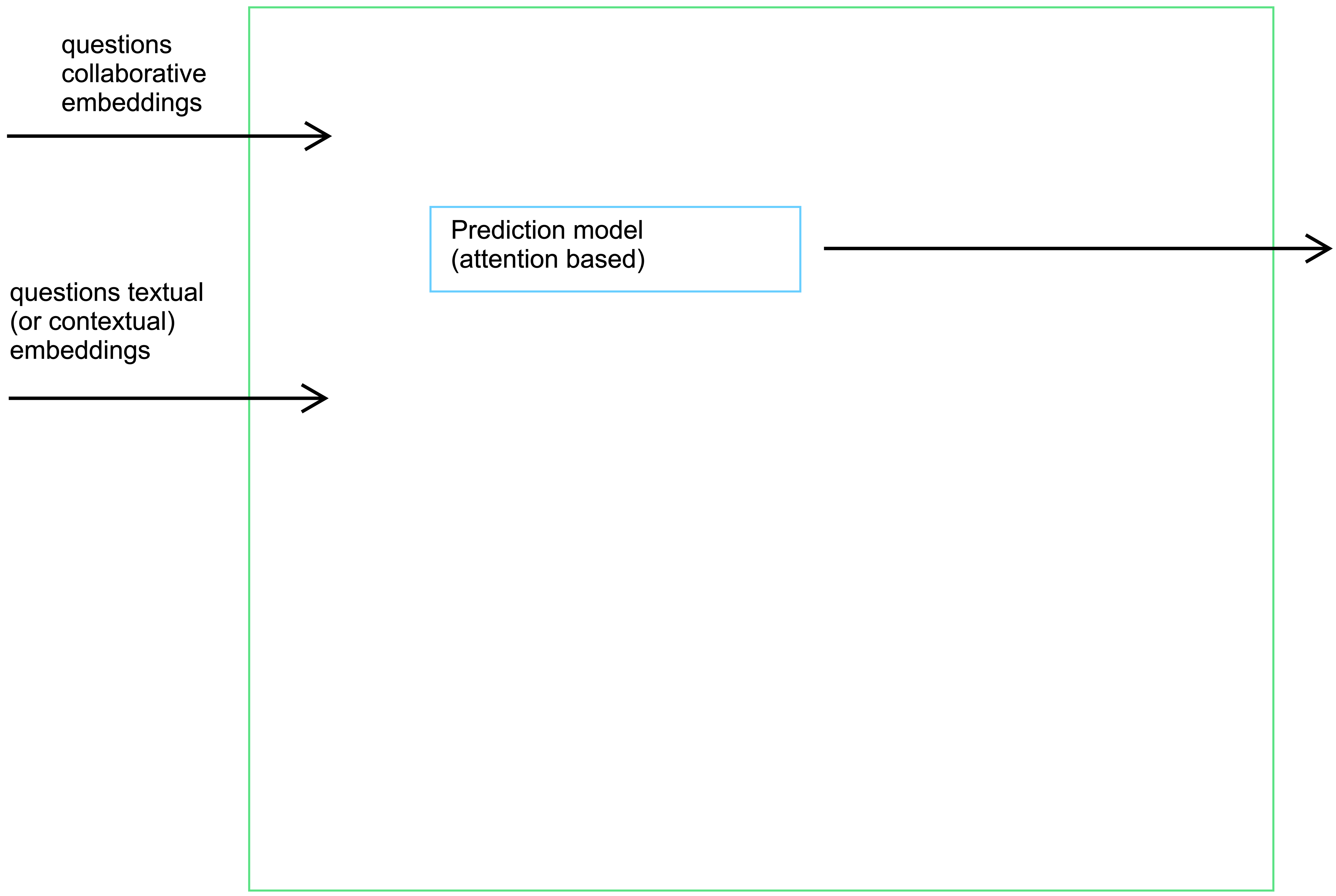
Personal opinion:

Prediction layer: (final layer which uses inputs to predicts correctness)
I think that the best models for predictions to try are transformer based models, since they outperform others by a lot.
SAINT is actually the best, so I think it is wise to use it as prediciton model. I am not sure if the code found online is actually well done, so I will use RKT too.

Inputs from collaborative filters: (similarity in performance between users and between items)
- I can take from RKT the use of PEBG generated features and the use of time decaying similarity model.

Inputs from text:
- Compare different methods:
    1) Count vectorizer /TF_IDF
    2) gensim (different variants as Word2Vec, pretrained Word2Vec, Doc2Vec, Doc2Vec based on pretrained Word2Vec)




questions collaborative embeddings
Prediction model (attention based)
questions textual (or contextual) embeddings