Deep Learning based Recommender System

Jakub Kuciński

Why would we use Deep Neural Networks for recommendations?

- Suitable inductive biases to the input data type
- Can adapt or use advances from other domains
- End-to-end differentiable
- Nonlinear transformations
- Representation learning
- Sequence modelling
- Flexibility

Classic Factorization Machines

	Feature vector **x**																				Target y	
	User				Movie					Other Movies rated					Time	Last Movie rated						
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...	Time	TI	NH	SW	ST	...		
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...	1	$y^{(2)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...	4	$y^{(3)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...	5	$y^{(4)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...	1	$y^{(5)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...	5	$y^{(6)}$

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i\, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle\, x_i\, x_j$$

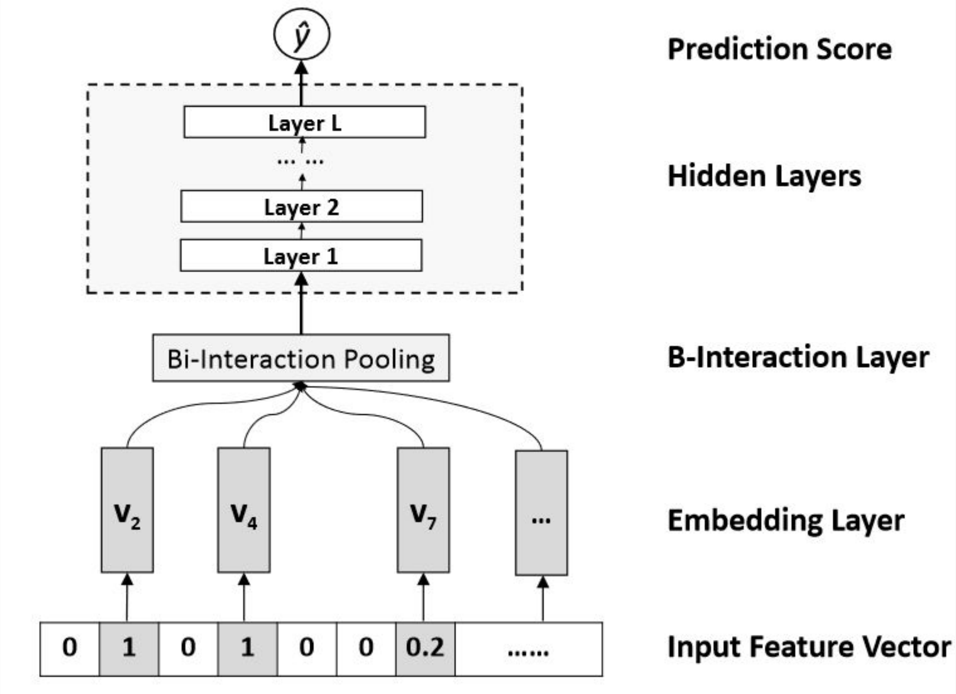
Classic Factorization Machines

$$
\begin{aligned}
&\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j \\
=&\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j - \frac{1}{2} \sum_{i=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_i \rangle \, x_i \, x_i \\
=&\frac{1}{2} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{f=1}^{k} v_{i,f} \, v_{j,f} \, x_i \, x_j - \sum_{i=1}^{n} \sum_{f=1}^{k} v_{i,f} \, v_{i,f} \, x_i \, x_i \right) \\
=&\frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f} \, x_i \right) \left(\sum_{j=1}^{n} v_{j,f} \, x_j \right) - \sum_{i=1}^{n} v_{i,f}^2 \, x_i^2 \right) \\
=&\frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f} \, x_i \right)^2 - \sum_{i=1}^{n} v_{i,f}^2 \, x_i^2 \right)
\end{aligned}
$$

Neural Factorization Machines

$$\hat{y}_{FM}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathbf{v}_i^T \mathbf{v}_j \cdot x_i x_j$$

$$\hat{y}_{NFM}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + f(\mathbf{x})$$



$$f(\mathbf{x}) = \mathbf{h}^T \mathbf{z}_L$$

$$\begin{aligned}
\mathbf{z}_1 &= \sigma_1(\mathbf{W}_1 f_{BI}(\mathcal{V}_x) + \mathbf{b}_1), \\
\mathbf{z}_2 &= \sigma_2(\mathbf{W}_2 \mathbf{z}_1 + \mathbf{b}_2), \\
&\ldots\ldots \\
\mathbf{z}_L &= \sigma_L(\mathbf{W}_L \mathbf{z}_{L-1} + \mathbf{b}_L),
\end{aligned}$$

$$f_{BI}(\mathcal{V}_x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i \mathbf{v}_i \odot x_j \mathbf{v}_j$$

$$\mathcal{V}_x = \{x_i \mathbf{v}_i\} \text{ where } x_i \neq 0$$

$$f_{BI}(\mathcal{V}_x) = \frac{1}{2}\left[\left(\sum_{i=1}^{n} x_i \mathbf{v}_i\right)^2 - \sum_{i=1}^{n} (x_i \mathbf{v}_i)^2\right]$$

Neural Factorization Machines

$$\hat{y}_{NFM-0} = w_0 + \sum_{i=1}^{n} w_i x_i + \mathbf{h}^T \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i \mathbf{v}_i \odot x_j \mathbf{v}_j$$

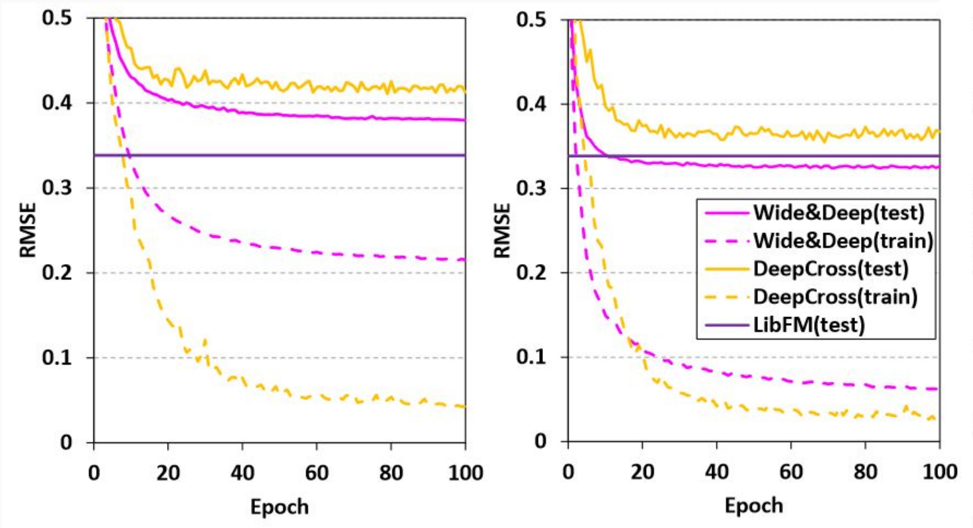
$$= w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{f=1}^{k} h_f v_{if} v_{jf} \cdot x_i x_j.$$

$$\hat{y}_{FM}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathbf{v}_i^T \mathbf{v}_j \cdot x_i x_j$$

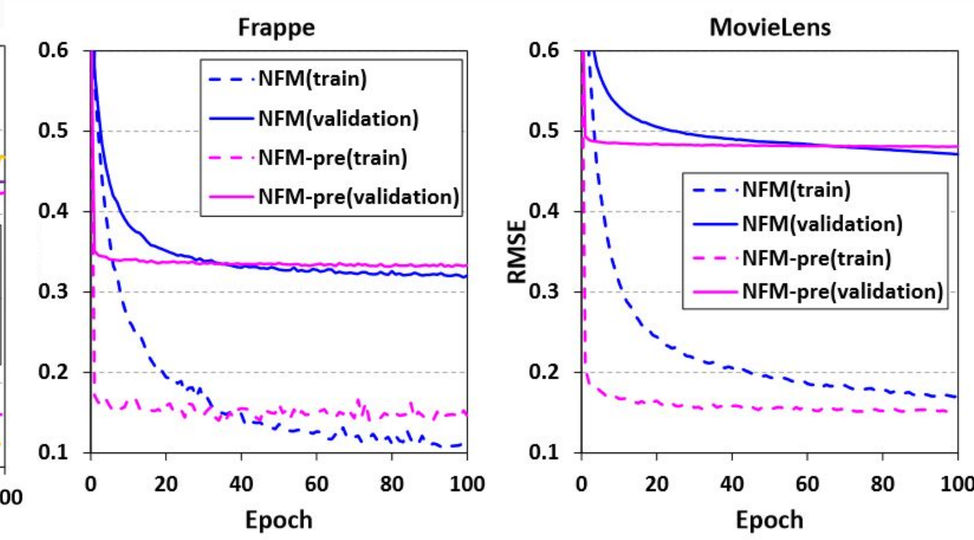
$$L_{reg} = \sum_{\mathbf{x} \in \mathcal{X}} (\hat{y}(\mathbf{x}) - y(\mathbf{x}))^2,$$

Neural Factorization Machines

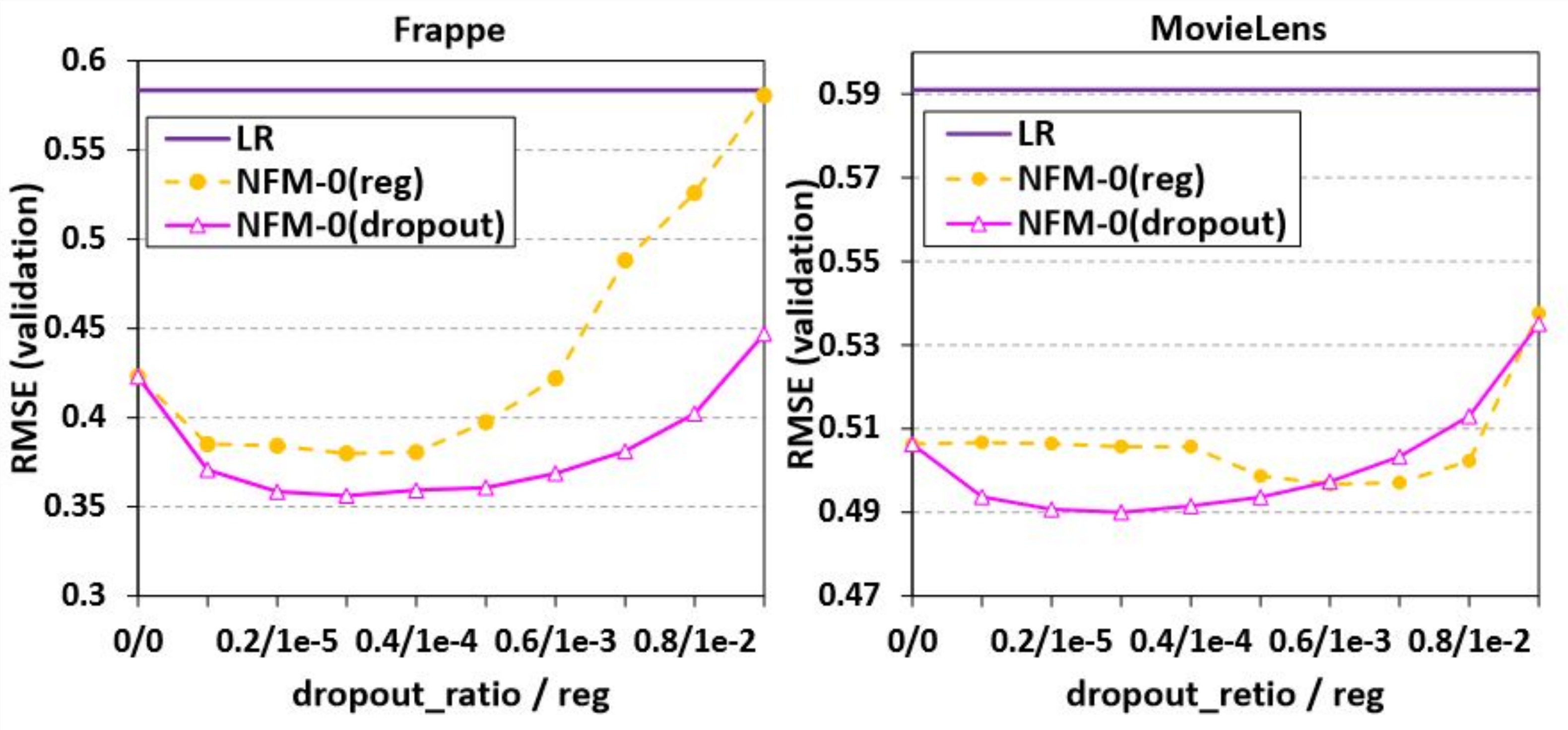
Dataset	Instance#	Feature#	User#	Item#
Frappe	288,609	5, 382	957	4,082
MovieLens	2,006,859	90, 445	17,045	23,743



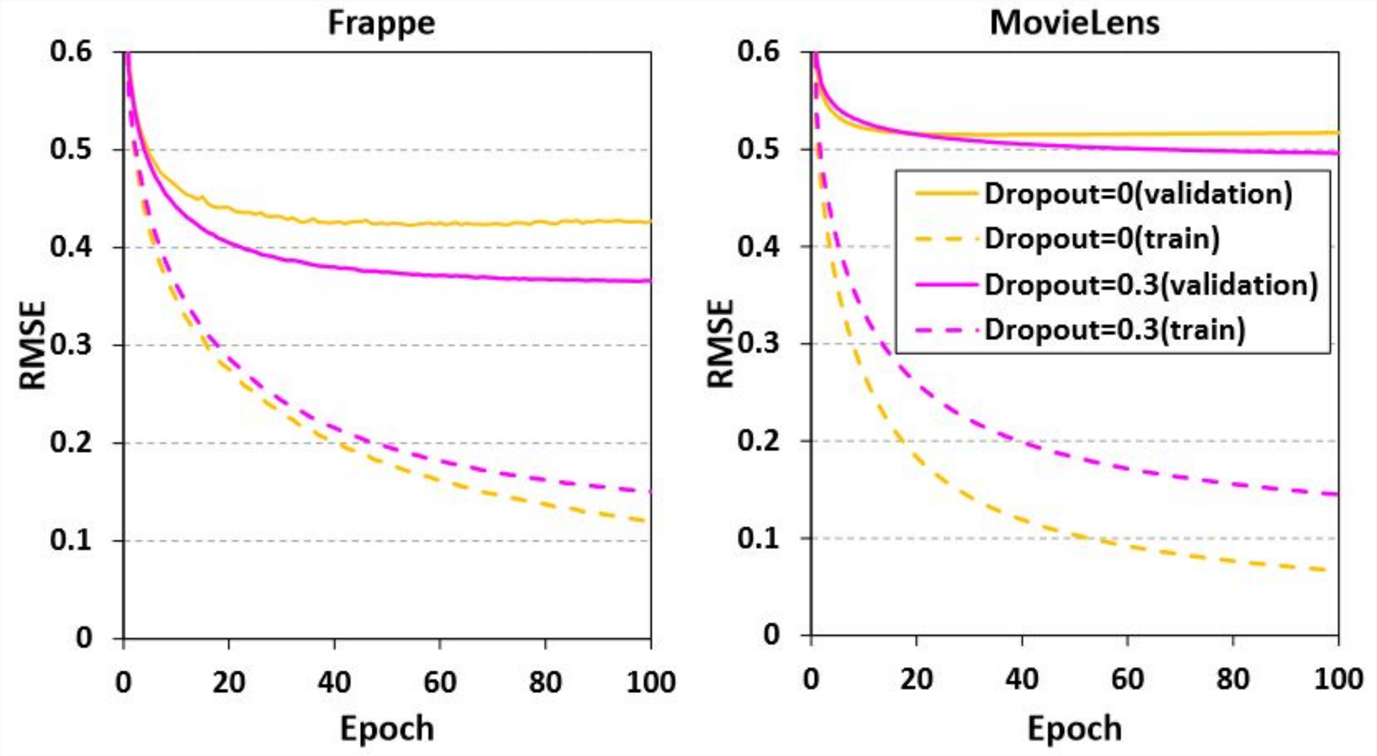
(a) Random initialization (b) FM as pre-training



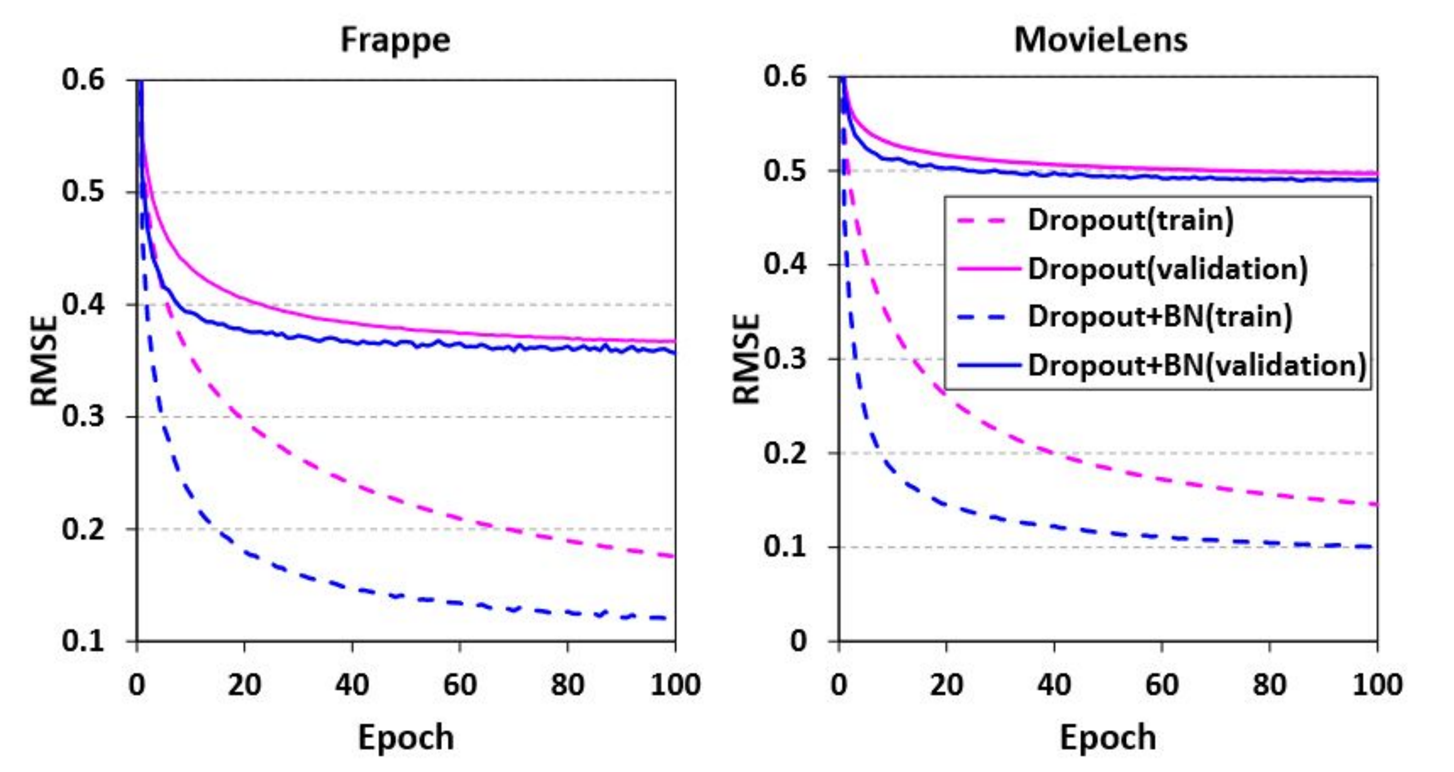
Neural Factorization Machines



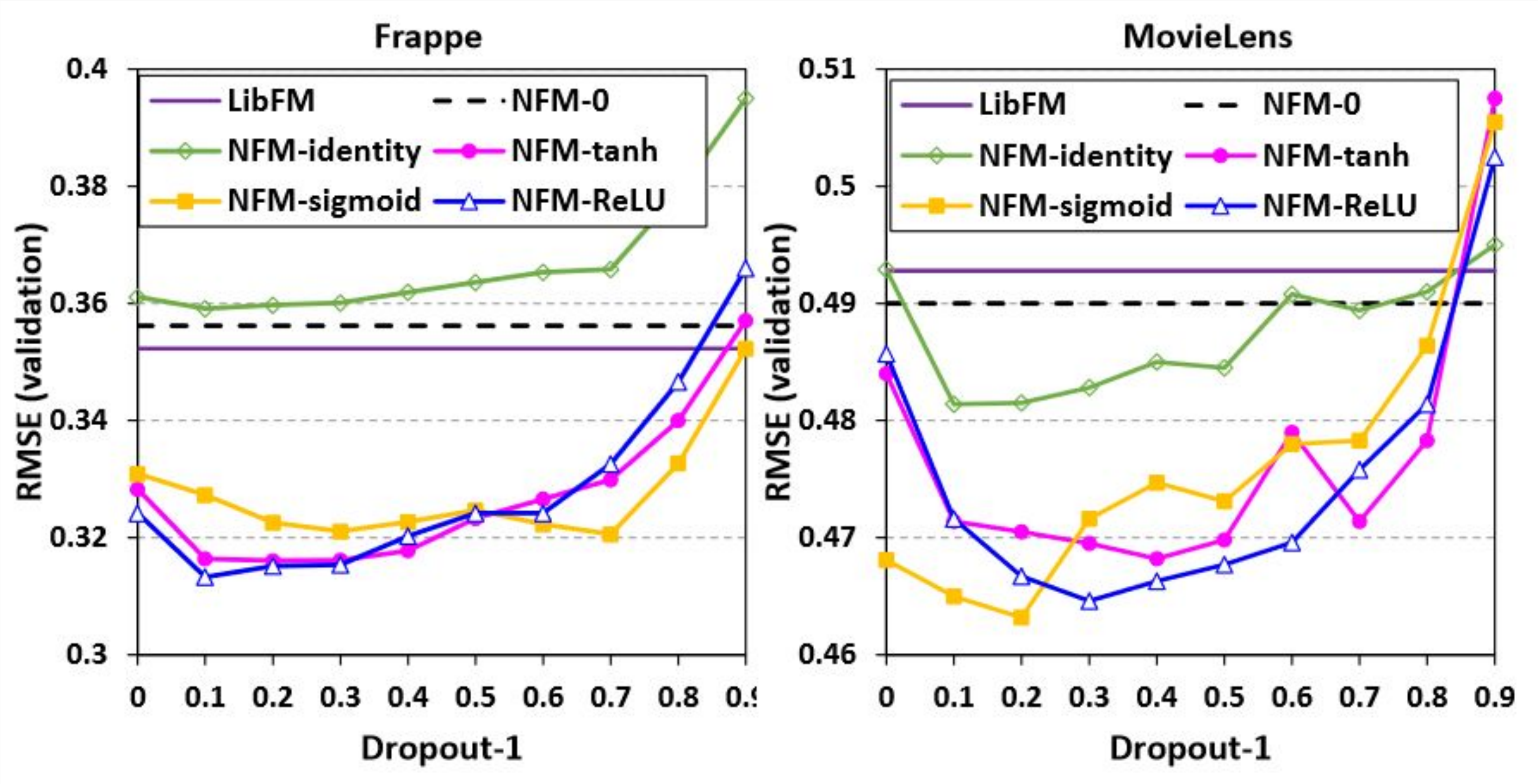
Neural Factorization Machines



Neural Factorization Machines



Neural Factorization Machines



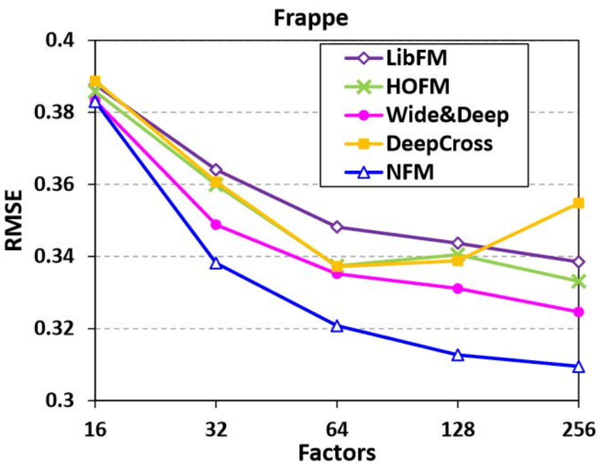
Neural Factorization Machines

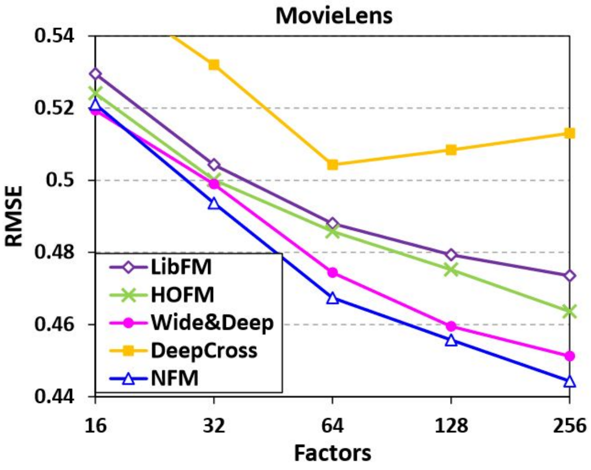
Table 2: NFM *w.r.t.* different number of hidden layers.

Methods	Frappe	MovieLens
NFM-0	0.3562	0.4901
NFM-1	**0.3133**	**0.4646**
NFM-2	0.3193	0.4681
NFM-3	0.3219	0.4752
NFM-4	0.3202	0.4703

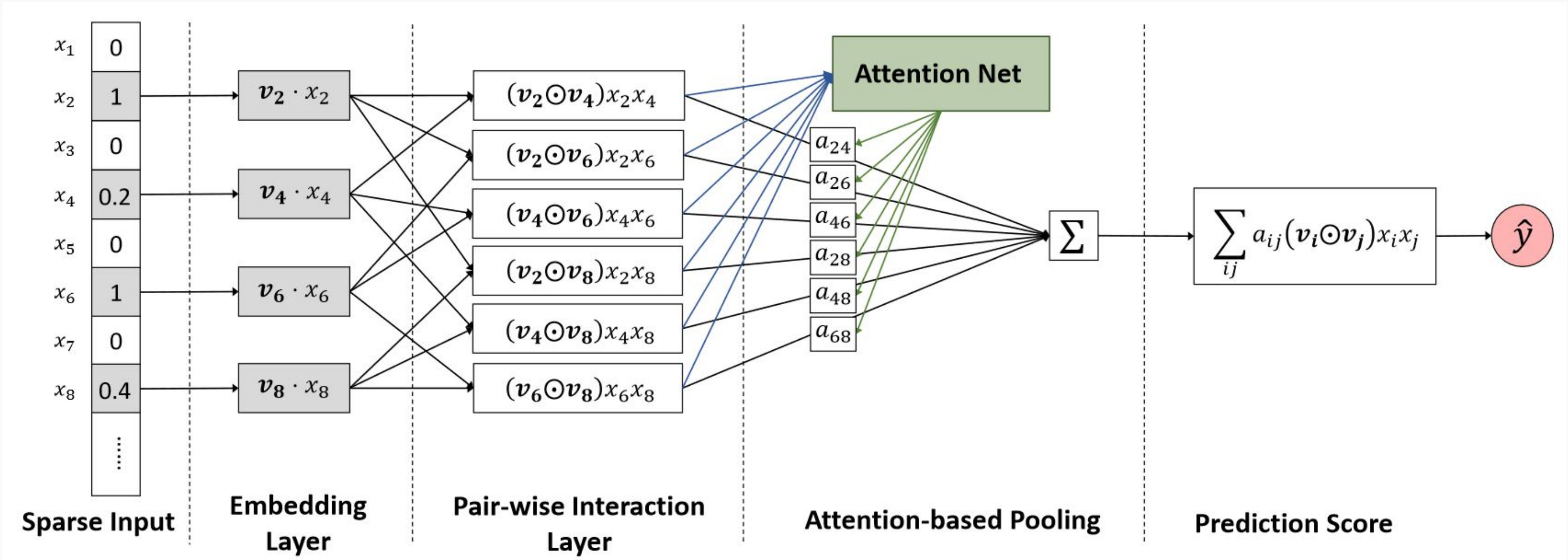
Neural Factorization Machines

	Frappe				MovieLens			
	Factors=128		Factors=256		Factors=128		Factors=256	
Method	**Param#**	**RMSE**	**Param#**	**RMSE**	**Param#**	**RMSE**	**Param#**	**RMSE**
LibFM [28]	0.69M	0.3437	1.38M	0.3385	11.67M	0.4793	23.24M	0.4735
HOFM	1.38M	0.3405	2.76M	0.3331	23.24M	0.4752	46.40M	0.4636
Wide&Deep [9]	2.66M	0.3621	4.66M	0.3661	12.72M	0.5323	24.69M	0.5313
Wide&Deep (pre-train)	2.66M	0.3311	4.66M	0.3246	12.72M	0.4595	24.69M	0.4512
DeepCross [31]	4.47M	0.4025	8.93M	0.4071	12.71M	0.5885	25.42M	0.5907
DeepCross (pre-train)	4.47M	0.3388	8.93M	0.3548	12.71M	0.5084	25.42M	0.5130
NFM	**0.71M**	**0.3127****	**1.45M**	**0.3095****	**11.68M**	**0.4557***	**23.31M**	**0.4443***





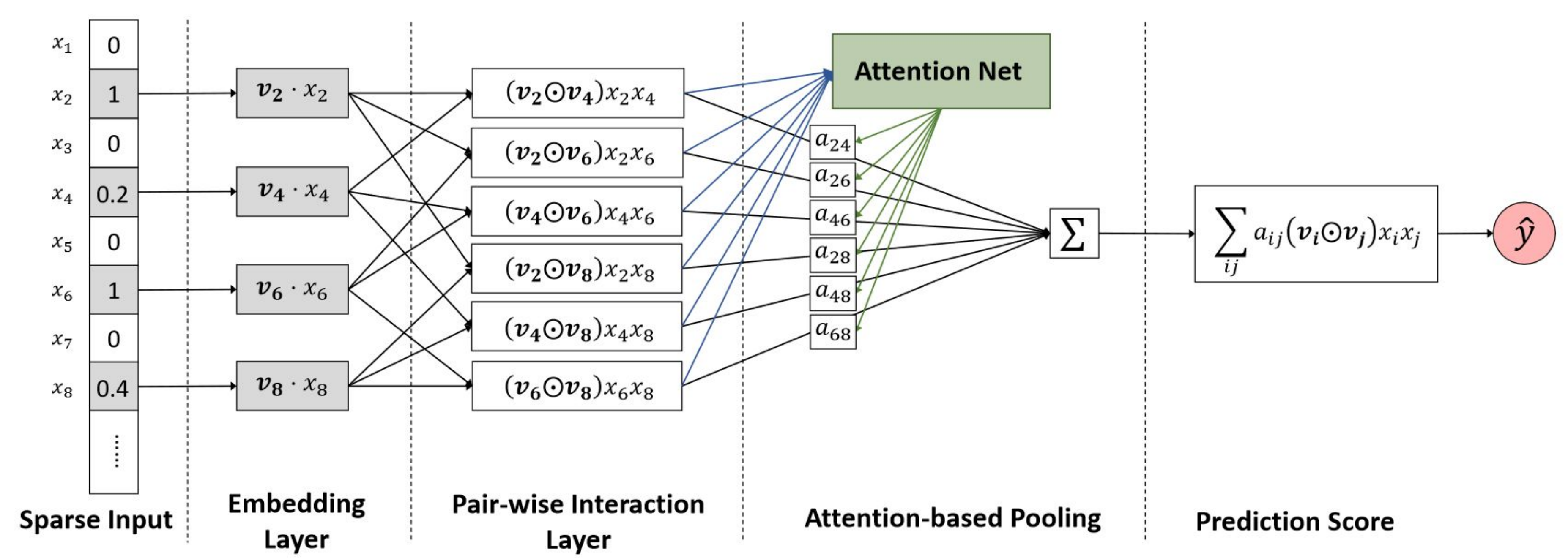
Attentional Factorization Machines



$$f_{BI}(\mathcal{V}_x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i \mathbf{v}_i \odot x_j \mathbf{v}_j$$

$$f_{Att}(f_{PI}(\mathcal{E})) = \sum_{(i,j)\in\mathcal{R}_x} a_{ij}(\mathbf{v}_i \odot \mathbf{v}_j)x_ix_j$$

Attentional Factorization Machines



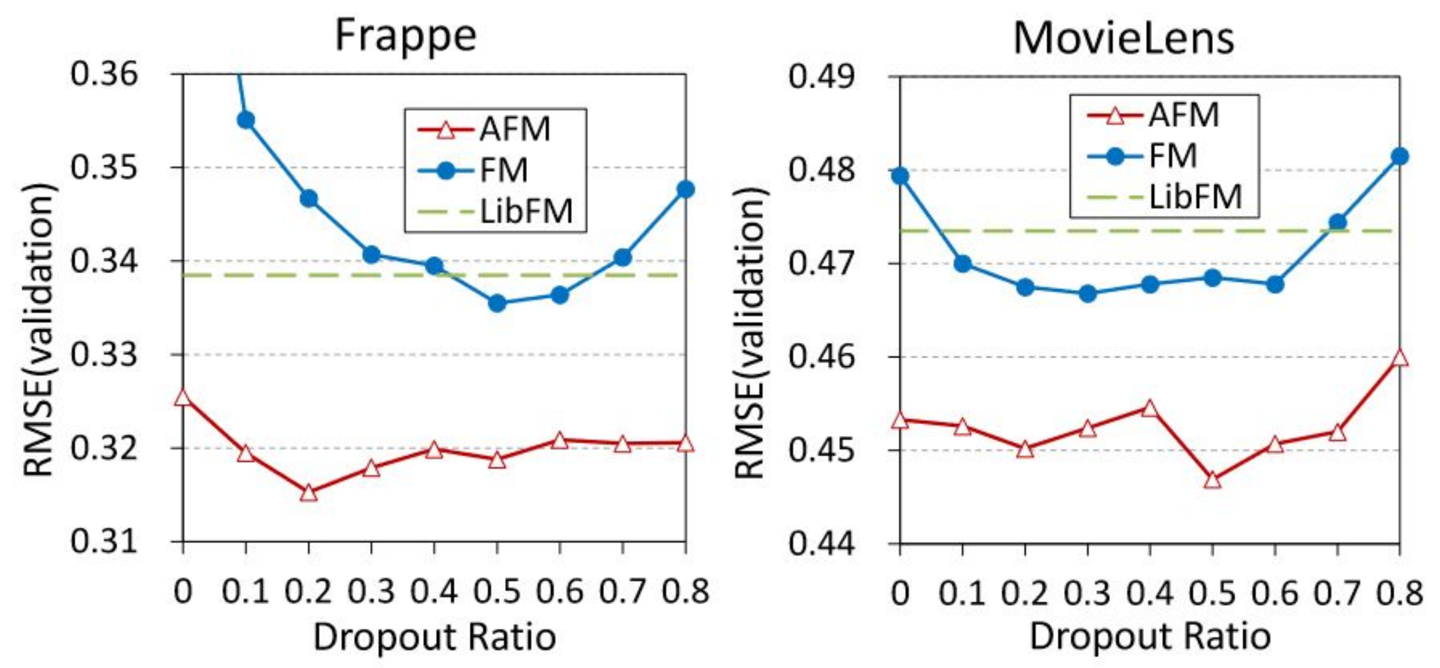
$$f_{Att}(f_{PI}(\mathcal{E})) = \sum_{(i,j)\in\mathcal{R}_x} a_{ij}(\mathbf{v}_i \odot \mathbf{v}_j)x_i x_j$$

$$a'_{ij} = \mathbf{h}^T ReLU(\mathbf{W}(\mathbf{v}_i \odot \mathbf{v}_j)x_i x_j + \mathbf{b}),$$

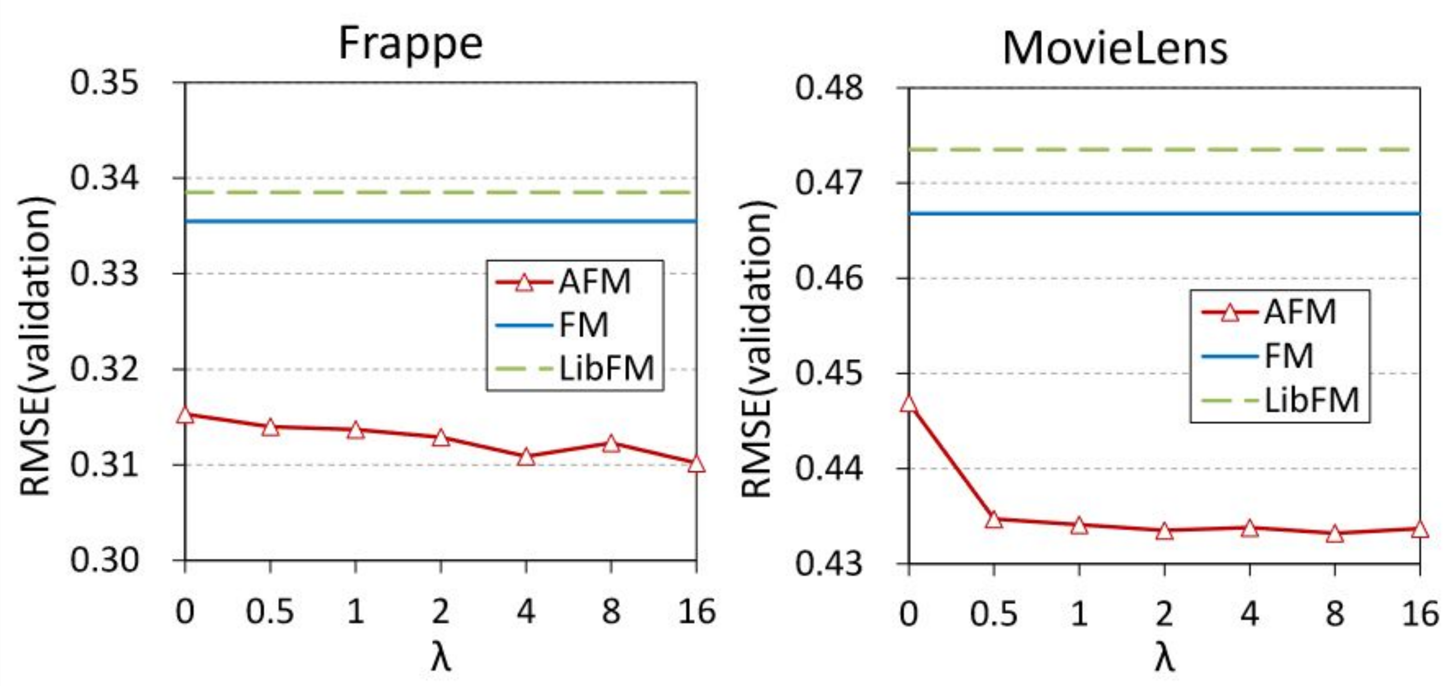
$$a_{ij} = \frac{\exp(a'_{ij})}{\sum_{(i,j)\in\mathcal{R}_x} \exp(a'_{ij})},$$

Attentional Factorization Machines

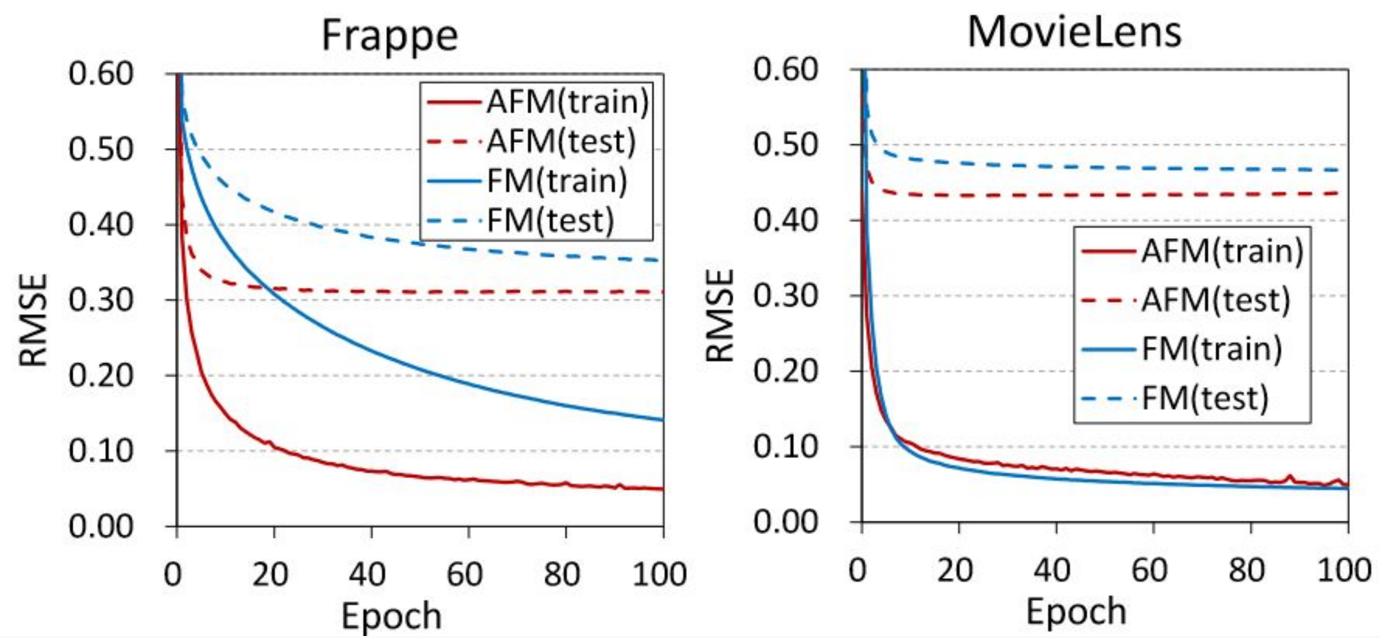
$$L = \sum_{x \in \mathcal{T}} (\hat{y}_{AFM}(\mathbf{x}) - y(\mathbf{x}))^2 + \lambda ||\mathbf{W}||^2$$



Attentional Factorization Machines



Attentional Factorization Machines



Attentional Factorization Machines

Table 2: Test error and number of parameters of different methods on embedding size 256. M denotes "million".

	Frappe		MovieLens	
Method	**Param#**	**RMSE**	**Param#**	**RMSE**
LibFM	1.38M	0.3385	23.24M	0.4735
HOFM	2.76M	0.3331	46.40M	0.4636
Wide&Deep	4.66M	0.3246	24.69M	0.4512
DeepCross	8.93M	0.3548	25.42M	0.5130
AFM	**1.45M**	**0.3102**	**23.26M**	**0.4325**

	Frappe				MovieLens			
	Factors=128		Factors=256		Factors=128		Factors=256	
Method	Param#	RMSE	Param#	RMSE	Param#	RMSE	Param#	RMSE
NFM	0.71M	0.3127^{**}	1.45M	0.3095^{**}	11.68M	0.4557^{*}	23.31M	0.4443^{*}

Item2Vec

$$\frac{1}{K}\sum_{i=1}^{K}\sum_{-c\le j\le c, j\ne 0}\log p(w_{i+j} | w_i)$$

$$p(w_j | w_i) = \frac{\exp(u_i^T v_j)}{\sum_{k\in I_W}\exp(u_i^T v_k)}$$

$$p(w_j | w_i) = \sigma(u_i^T v_j)\prod_{k=1}^{N}\sigma(-u_i^T v_k)$$

$$p(discard | w) = 1 - \sqrt{\frac{\rho}{f(w)}}$$

$$\frac{1}{K}\sum_{i=1}^{K}\sum_{j\ne i}^{K}\log p(w_j | w_i)$$

Item2Vec

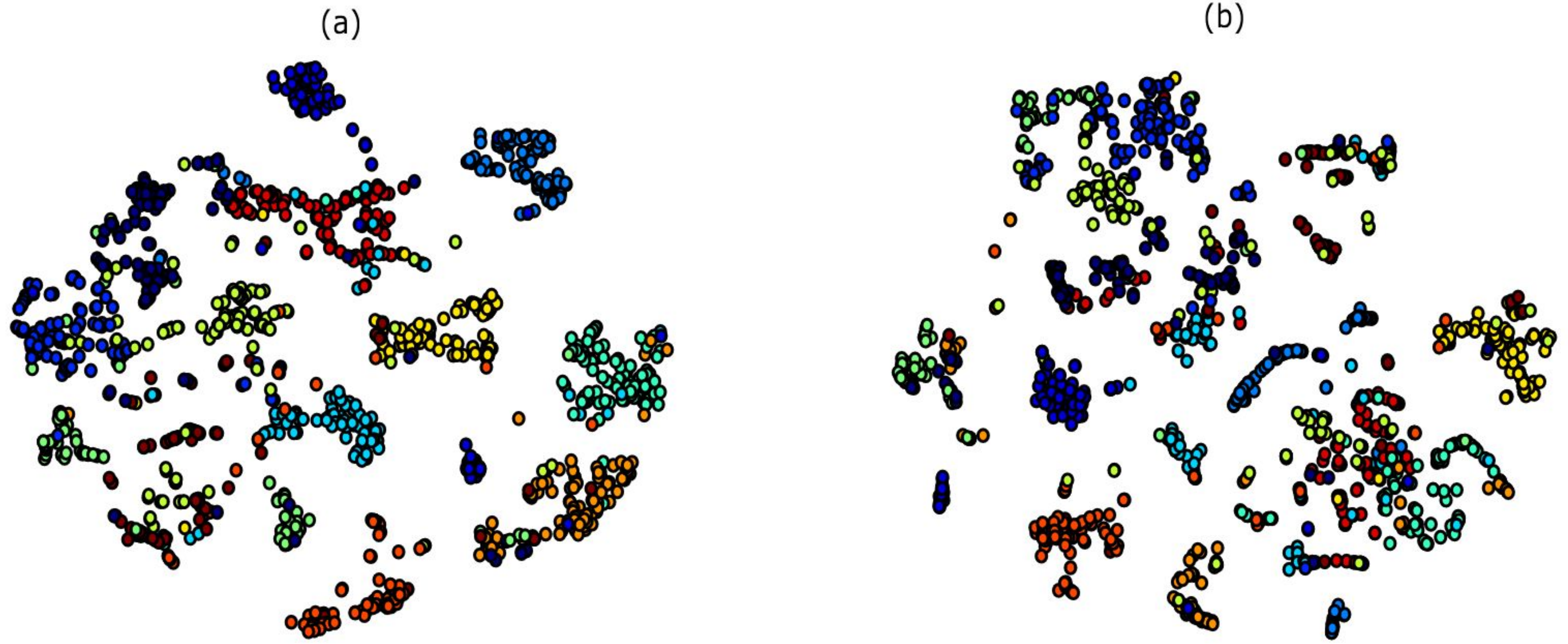


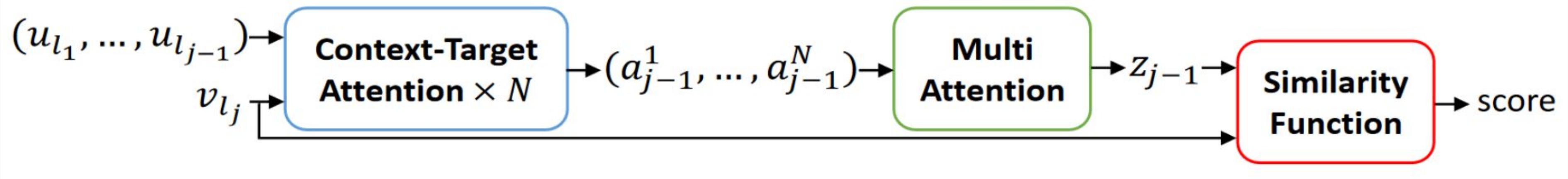
Fig.2: t-SNE embedding for the item vectors produced by item2vec (a) and SVD (b). The items are colored according to a web retrieved genre metadata.

Item2Vec

TABLE 2: A COMPARISON BETWEEN SVD AND ITEM2VEC ON GENRE CLASSIFICATION TASK FOR VARIOUS SIZES OF TOP POPULAR ARTIST SETS

Top (q) popular artists	SVD accuracy	item2vec accuracy
2.5k	85%	**86.4%**
5k	83.4%	**84.2%**
10k	80.2%	**82%**
15k	76.8%	**79.5%**
20k	73.8%	**77.9%**
10k unpopular (see text)	58.4%	**68%**

Attentive Item2Vec



$$x = (l_1, \dots, l_K)$$

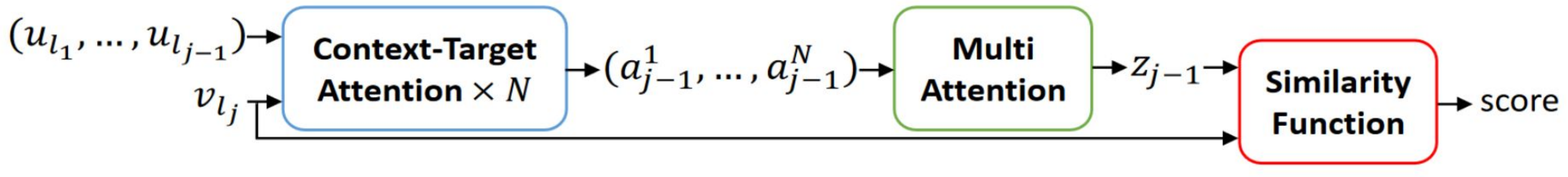
$$x_{1:j-1} = (l_1, \dots, l_{j-1})$$

$$a_{j-1} = \sum_{m=1}^{j-1} \alpha_{jm} B_c u_{l_m}$$

$$\alpha_{jm} = \frac{\exp\left(h(A_c u_{l_m}, A_t v_{l_j})\right)}{\sum_{n=1}^{j-1} \exp\left(h(A_c u_{l_n}, A_t v_{l_j})\right)}$$

$$h(u_i, v_j) = \frac{u_i^T v_j}{|u_i||u_j|}.$$

Attentive Item2Vec



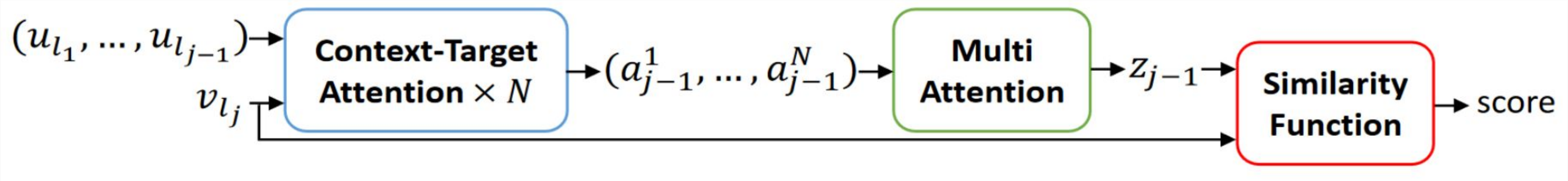
$$a_{j-1} = \sum_{m=1}^{j-1} \alpha_{jm} B_c u_{l_m} \qquad w_{j-1} = \left[(a_{j-1}^1)^T, \dots, (a_{j-1}^N)^T\right]^T$$

$$z_{j-1} = R w_{j-1}$$

$$o\left(z_{j-1}, v_{l_j}\right) = \psi_o\left(z_{j-1}, B_t v_{l_j}\right) + b_{l_j}$$

$$\psi(u, v) = W_1 \phi(W_0([u, v, u \circ v, |u - v|])),$$

Attentive Item2Vec



$$L_x = \sum_{j=2}^{K} - \log p(\, l_j | l_{1:j-1}),$$

$$p(l_j | l_{1:j-1}) = \frac{\exp\left(o\left(z_{j-1}, v_{l_j}\right)\right)}{\sum_{k \in \mathcal{N}} \exp\left(o\left(z_{j-1}, v_k\right)\right)}.$$

Attentive Item2Vec

Table 1. Datasets statistics

Dataset	#items	#users	#training examples	#test examples
MovieLens	6,040	3,577	778,679	6,040
MS	2,138	10,000	95,753	10,000
Yahoo!	4,222	9,362	241,896	9,362

Table 2. Performance Evaluation on MS dataset ($\cdot 10^{-2}$)

Model	HR@5	HR@10	HR@20	MRR@5	MRR@10	MRR@20
AI2V	**20.81**	**26.95**	**34.89**	**12.88**	**13.70**	**14.24**
I2V	11.06	15.53	22.37	4.53	5.13	5.59
NCF	11.42	20.08	32.28	5.59	6.70	7.55

Attentive Item2Vec

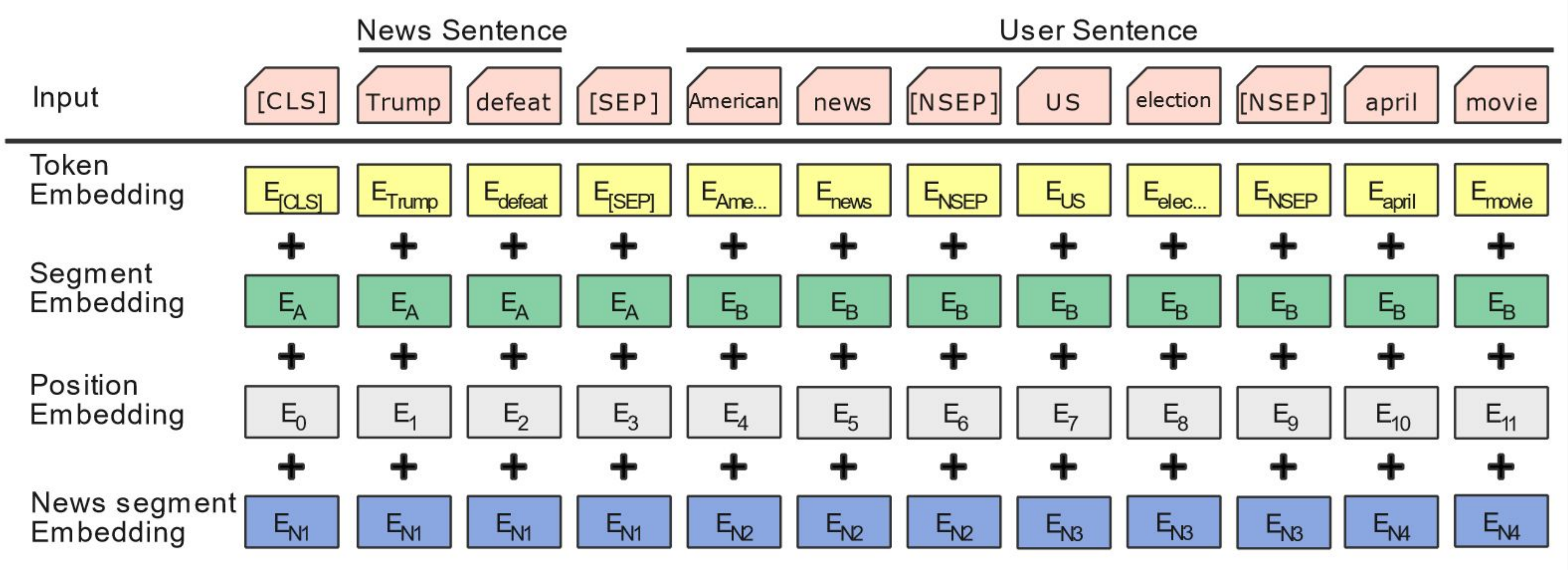
Table 3. Performance Evaluation on MovieLens dataset ($\cdot 10^{-3}$)

Model	HR@5	HR@10	HR@20	MRR@5	MRR@10	MRR@20
AI2V	**12.92**	**28.06**	**58.56**	**5.56**	**7.39**	**9.52**
I2V	11.30	27.05	55.73	3.75	5.87	7.75
NCF	8.48	19.99	45.84	3.25	4.78	6.56

Table 4. Performance Evaluation on Yahoo! Music dataset ($\cdot 10^{-1}$)

Model	HR@5	HR@10	HR@20	MRR@5	MRR@10	MRR@20
AI2V	**8.42**	**14.81**	**24.98**	**3.93**	**4.77**	**5.46**
I2V	5.65	10.94	18.07	2.57	3.26	3.75
NCF	5.64	11.09	19.41	2.57	3.28	3.85

UNBERT


News Sentence
User Sentence
Input
[CLS]
Trump
defeat
[SEP]
American
news
[NSEP]
US
election
[NSEP]
april
movie
Token Embedding
$E_{[CLS]}$ E_{Trump} E_{defeat} $E_{[SEP]}$ $E_{Ame...}$ E_{news} E_{NSEP} E_{US} $E_{elec...}$ E_{NSEP} E_{april} E_{movie}
Segment Embedding
E_A E_A E_A E_A E_B E_B E_B E_B E_B E_B E_B E_B
Position Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10} E_{11}
News segment Embedding
E_{N1} E_{N1} E_{N1} E_{N1} E_{N2} E_{N2} E_{N2} E_{N3} E_{N3} E_{N3} E_{N4} E_{N4}

UNBERT

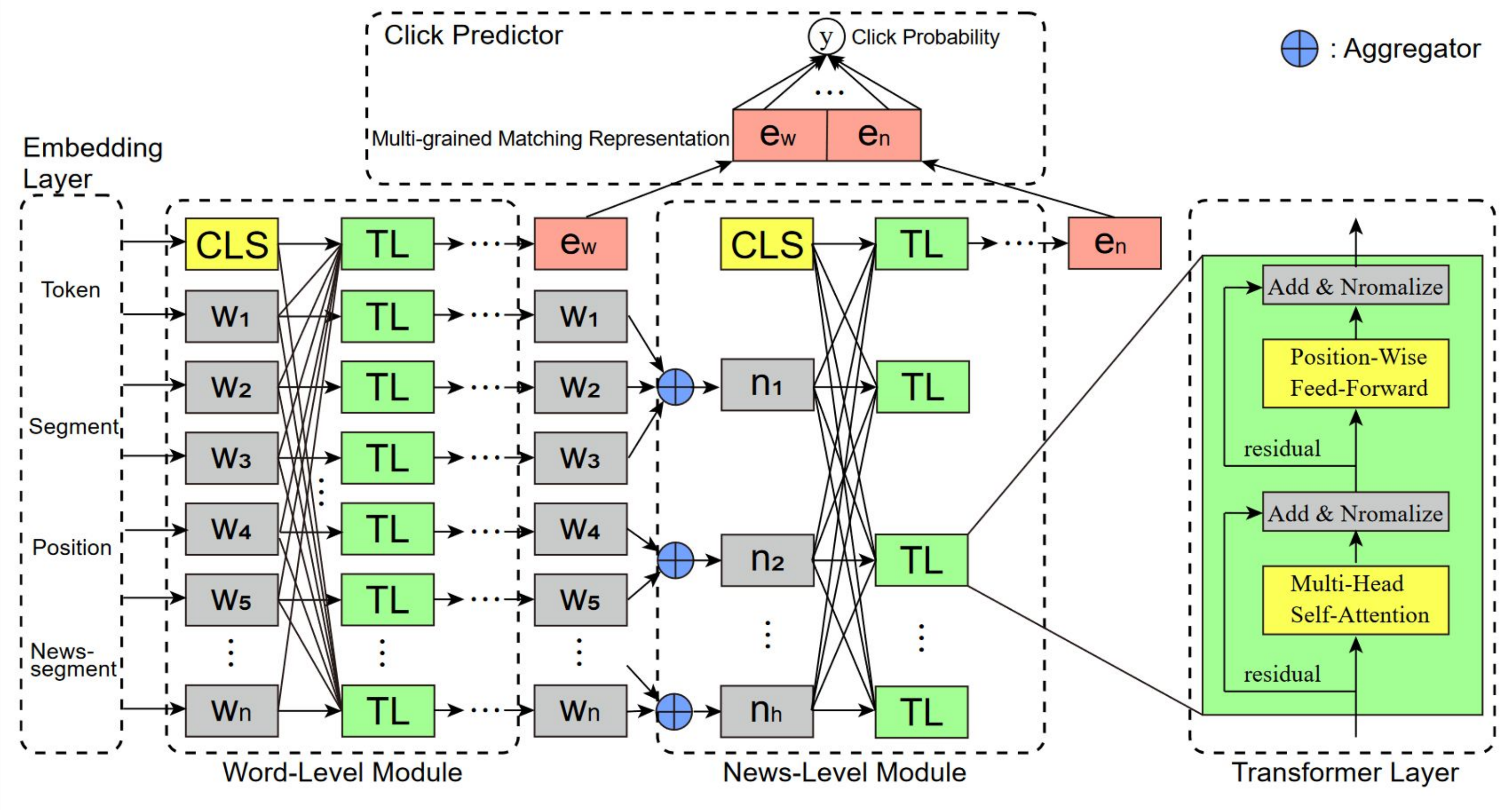


Figure 3: The overall architecture of our UNBERT approach.

UNBERT

$$Attention(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$head_i = Attention\left(E_t W_i^Q, E_t W_i^K, E_t W_i^V\right)$$

$$MultiHead(Q, K, V) = [head_1; \ldots; head_h]\, W^O$$

$$FFN(x) = \mathrm{ReLU}\left(xW_1 + b_1\right)W_2 + b_2$$

UNBERT

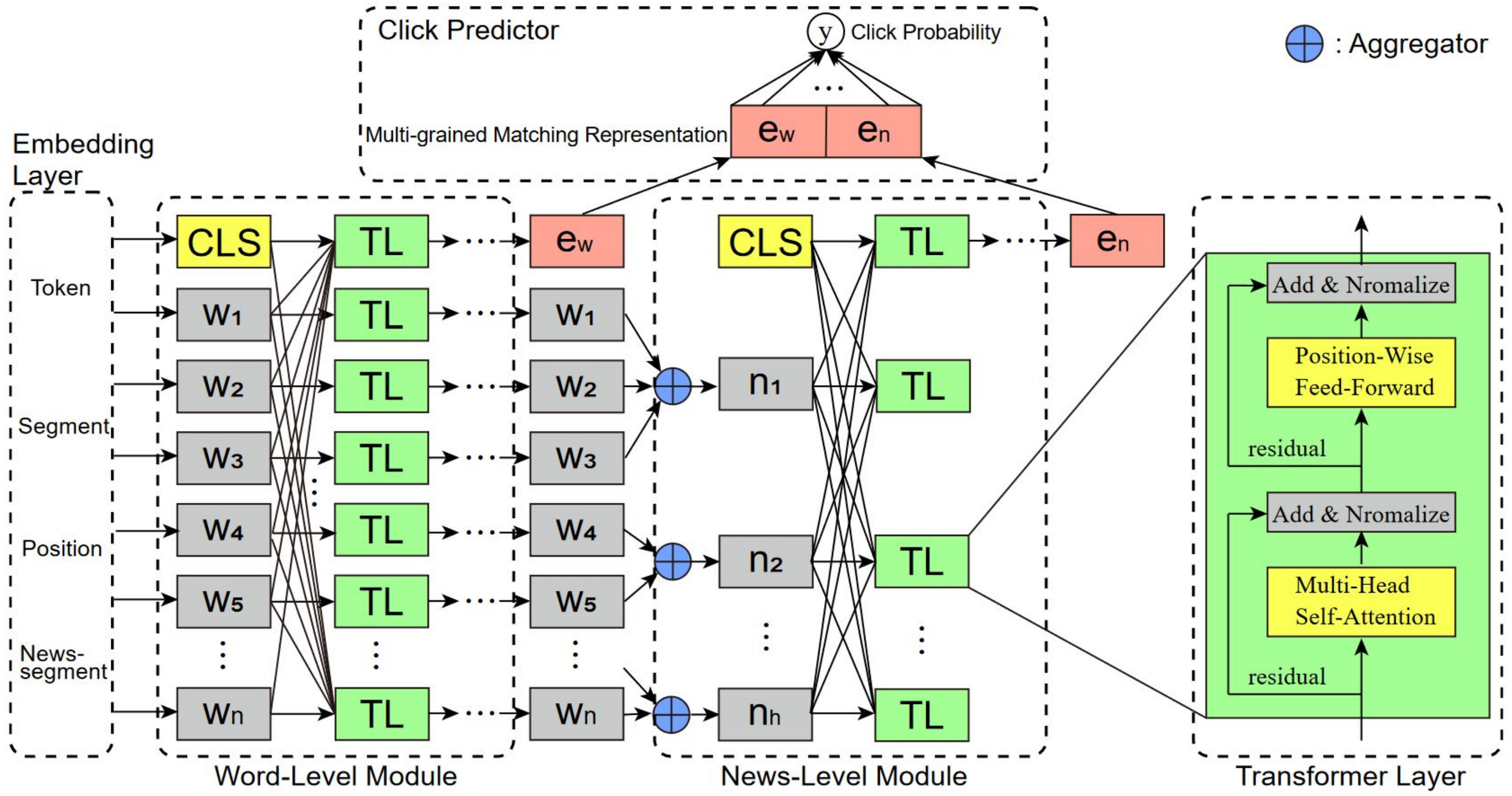


Figure 3: The overall architecture of our UNBERT approach.

UNBERT

NSEP Aggregator

$$n_j = w_i \quad \text{where} \quad i = [\texttt{NSEP}]_j$$

Mean Aggregator

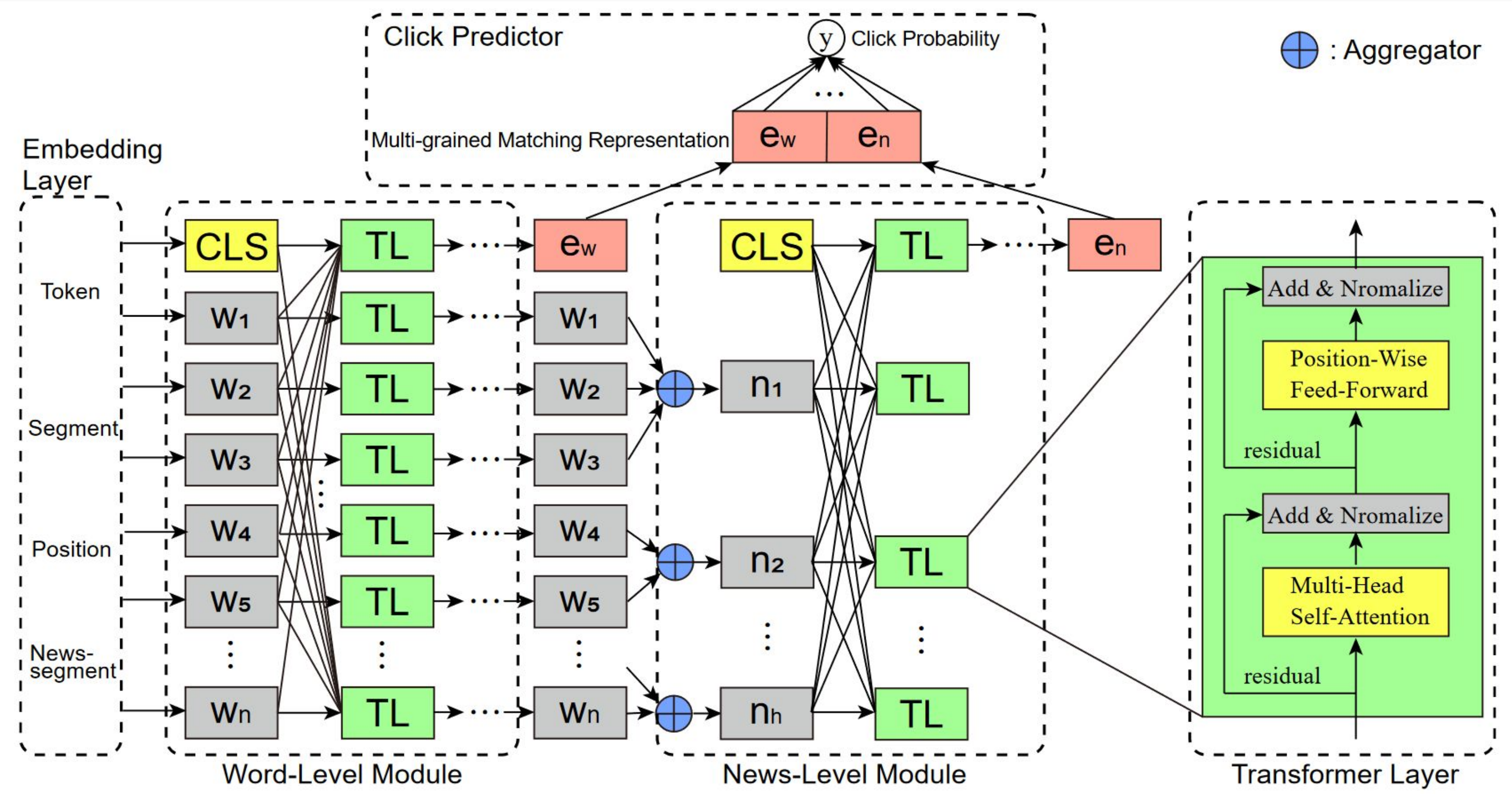
$$n_j = \sum_{i \in S_j} w_i \; / \; |S_j|$$

Attention Aggregator

$$f = \tanh\left(wW_h + b_h\right)W_o + b_o$$

$$n_j = \sum_{i \in S_j} f_i w_i \; / \sum_{i \in S_j} f_i$$

UNBERT



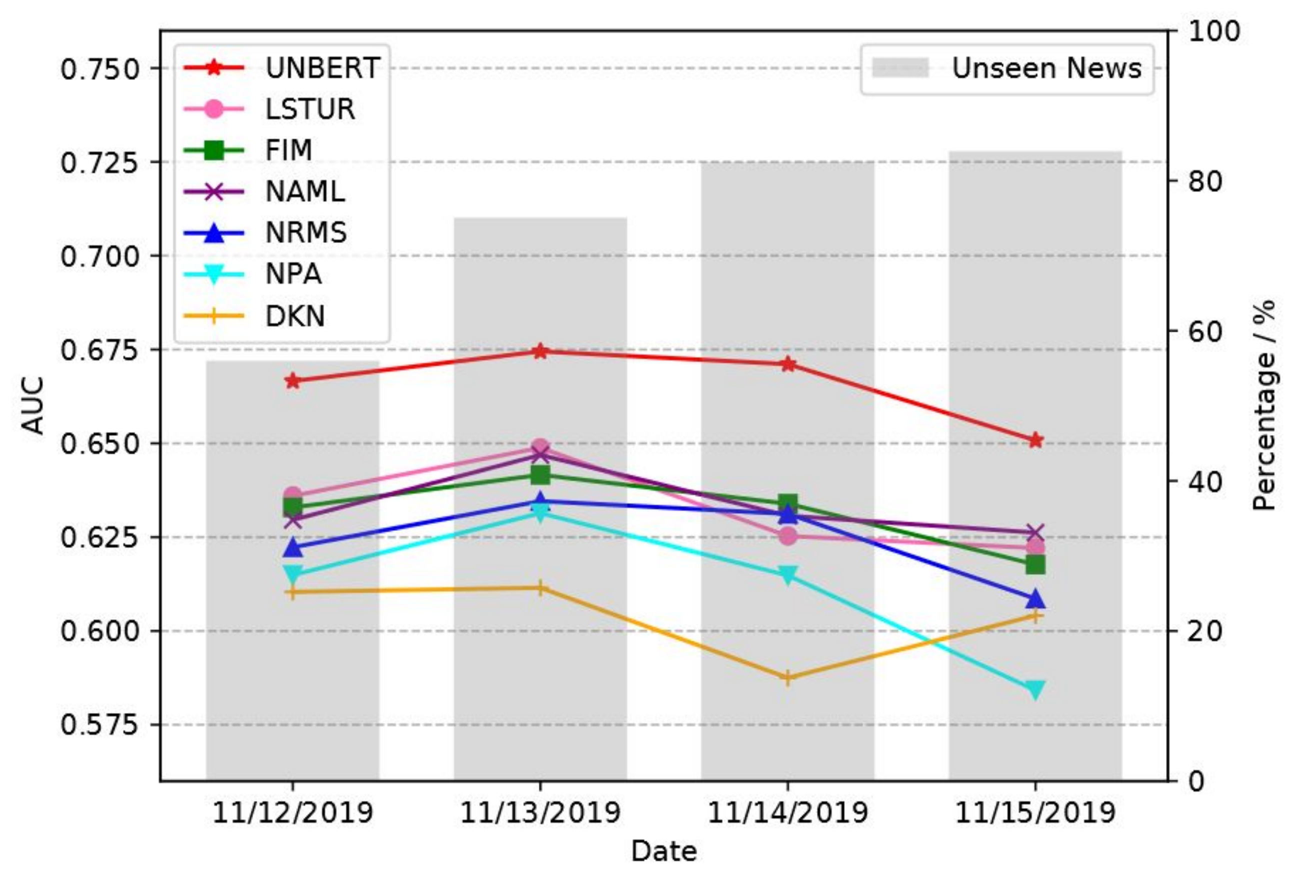
$$y = \mathrm{softmax}([e_w; e_n]\, W^c + b^c)$$

UNBERT

Method	MIND-small				MIND-large			
	AUC	MRR	nDCG@5	nDCG@10	AUC	MRR	nDCG@5	nDCG@10
LibFM	0.5974	0.2633	0.2795	0.3429	0.6185	0.2945	0.3145	0.3713
DeepFM	0.5989	0.2621	0.2774	0.3406	0.6187	0.2930	0.3135	0.3705
DKN	0.6175	0.2705	0.2890	0.3538	0.6407	0.3042	0.3292	0.3866
NPA	0.6321	0.2911	0.3170	0.3781	0.6592	0.3207	0.3472	0.4037
NAML	<u>0.6550</u>	<u>0.3039</u>	<u>0.3308</u>	<u>0.3931</u>	0.6646	0.3275	0.3566	0.4140
LSTUR	0.6438	0.2946	0.3189	0.3817	0.6708	0.3236	0.3515	0.4093
NRMS	0.6483	0.3001	0.3252	0.3892	0.6766	0.3325	0.3628	0.4198
FIM	0.6502	0.3026	0.3291	0.3910	<u>0.6787</u>	<u>0.3346</u>	<u>0.3653</u>	<u>0.4221</u>
UNBERT	**0.6762**	**0.3172**	**0.3475**	**0.4102**	**0.7068**	**0.3568**	**0.3913**	**0.4478**
%Improv.	2.12	1.33	1.67	1.71	2.81	2.22	2.60	2.57
UNBERT-en$^{\triangle}$	-	-	-	-	0.7183	0.3659	0.4020	0.4581

Boldface indicates the best results (the higher, the better), while the second best is underlined. UNBERT-en$^{\triangle}$ represents the ensemble score based on UNBERT which is at the top of https://msnews.github.io/#leaderboard.

UNBERT



UNBERT
LSTUR
FIM
NAML
NRMS
NPA
DKN
Unseen News
AUC
Percentage / %
Date
0.750
0.725
0.700
0.675
0.650
0.625
0.600
0.575
100
80
60
40
20
0
11/12/2019
11/13/2019
11/14/2019
11/15/2019

Resourses

- Shuai Zhang et al. (2019), Deep Learning based Recommender System: A Survey and New Perspectives.
- Steffen Rendle (2010), Factorization Machines.
- Xiangnan He and Tat-Seng Chua (2017), Neural Factorization Machines for Sparse Predictive Analytics.
- Jun Xiao et al. (2017), Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks.
- Oren Barkan and Noam Koenigstein (2016), Item2Vec: Neural Item Embedding for Collaborative Filtering.
- Oren Barkan et al. (2020), Attentive Item2Vec: Neural Attentive User Representations.
- Qi Zhang et al. (2020), UNBERT: User-News Matching BERT for News Recommendation.
- Interesting blog post (but unfortunately without benchmarks) - OLX Item2Vec. Authors claims that they integrated this NN architecture into OLX recommendation system (Oct 26, 2021).