知识图谱

一、综述：了解概况

先从总体了解知识图谱，找到了文章Named Entity Extraction for Knowledge Graphs: A Literature Overview（doi: 10.1109/ACCESS.2020.2973928.）从总体上讲述了知识图谱的来源、构建过程。

FIGURE 1. From natural language (NL) to knowledge graphs (KG).

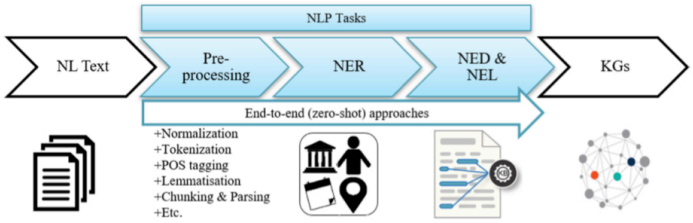


FIGURE 2. Natural-language processing (NLP) tasks.

在了解总体之后，依次阅读了关于**三元组提取（关系特征提取）**以及**知识图谱嵌入的若干综述**。选取了上述两过程目前接近或达到sota的两个模型：基于BERT的SpERT模型和Hierarchy-Aware Knowledge Graph（HAKE）模型作为知识图谱构建方面的了解。然后以一篇以**推荐作为下游任务目标**，用到了知识图谱并采用元路径寻找关系的模型作为知识图谱运用的学习。

二、具体方法深入

背景：自2012年，谷歌宣布推出知识图谱，这是一个大规模的知识库，旨在提供关于搜索查询的结构化信息。知识图谱的推出使得谷歌搜索结果更丰富、更精准，能够直接回答用户的问题，而不仅仅是返回相关链接。而由于三元组其简单易懂、表达能力强、易于存储和查询等特点，三元组的提取显得很重要。关系和实体提取技术的进步也为三元组的应用提供了技术支撑。三元组在知识图谱构建、信息检索、机器翻译、文本摘要、推荐系统等领域有着广泛的应用前景。

(一) 三元组构建———实体和关系的提取

❶ 模型：SpERT

（Span-based Joint Entity and Relation Extraction with Transformer Pre-training　DOI：https://doi.org/10.48550/arXiv.1909.07755）

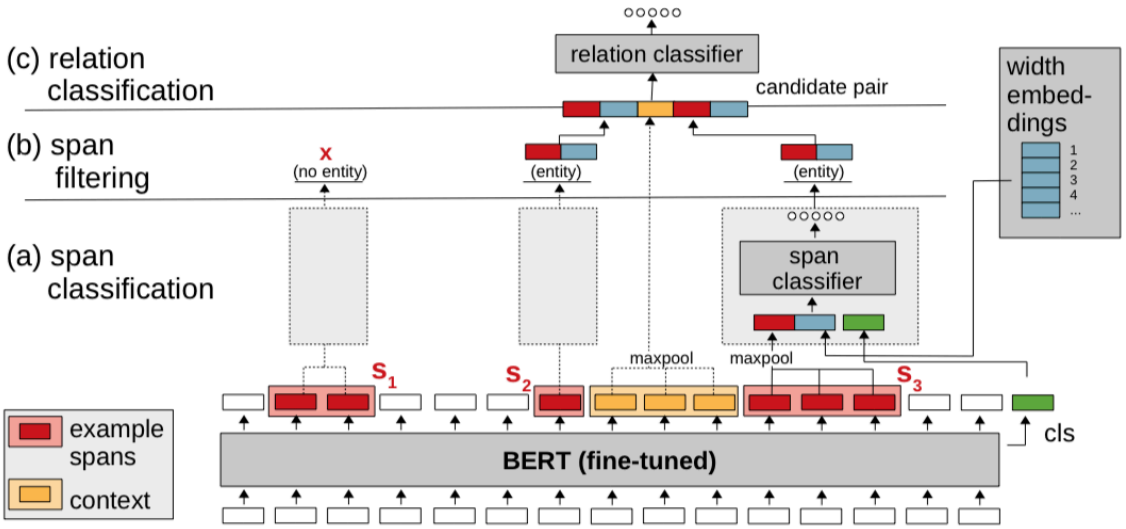
以往不足：

1. 关系和实体分步判断，泛化能力不足。

2. 嵌套实体的提取出现缺失，如Ford’s Chicago plant employs 4,000 workers”，实体存在的可能有Chicago和Chicago plant。

3. 依赖标记的实体，本文基于transformer可以自动识别关系和实体。

❷ 技术细节



(a) Span Classification：

a.把句子中所有可能的spans词句通过BERT模型嵌入，通过max pooling（目前实验证明最佳） （红框）

b.通过学习获得特定长度的有跨度的span经过BERT嵌入。 （蓝框）

c.总体文段的分析，全部文字嵌入，达到通过上下文消除歧义的效果。 （绿框）

这三个结果喂入span分类器，用softmax多分类为不同实体。

(b) Span Filtering

过滤不是实体的输出，简单地把10 tokens以上的去除，把成本限制在 O(n)。

(c) Relation Classification

在两个确定的实体s1，s2之间的词句通过max-pooling寻找联系，由于关系通常的不对称性，分别用<s1,r,s2>和<s2,r,s1>计算sigmoid得分，高于阈值的为两实体间的关系

训练

① 采用有label训练。 ② 强副样本在模型中起到重要作用。

采用实体和关系的损失之和作为损失函数：

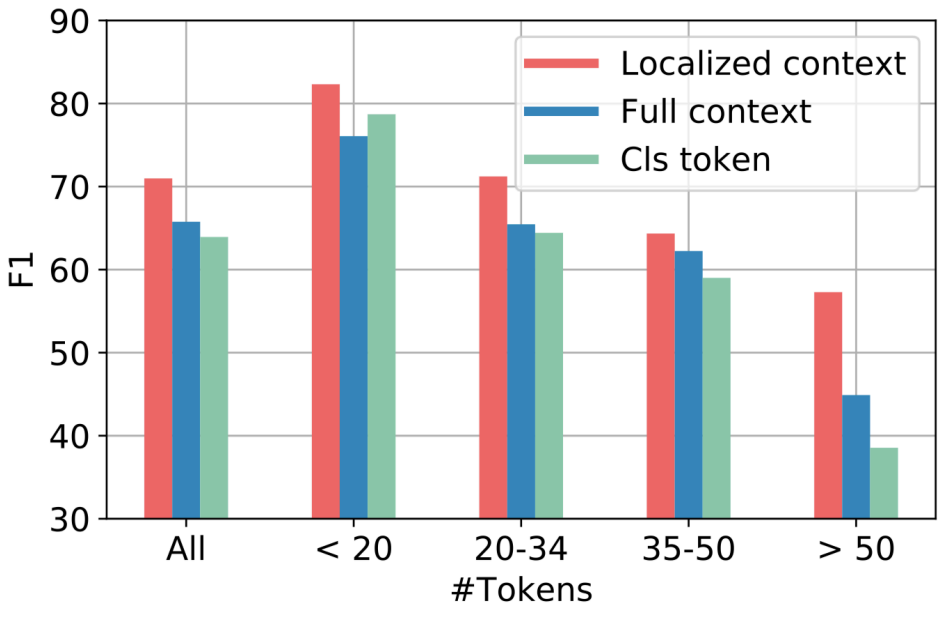
$$\mathcal{L} = \mathcal{L}^s + \mathcal{L}^r$$

❷ 实验

在众模型中做到了最佳：

Dataset	Model	Entity			Relation		
		Precision	Recall	F1	Precision	Recall	F1
CoNLL04	Multi-head + AT [2]†	-	-	83.61	-	-	61.95
	Multi-head [3]†	83.75	84.06	83.90	63.75	60.43	62.04
	Global Optimization [40]†	-	-	85.60	-	-	67.80
	Multi-turn QA [18]†	89.00	86.60	87.80	69.20	68.20	68.90
	Table-filling [23]*	81.20	80.20	80.70	76.00	50.90	61.00
	Hierarchical Attention [6]*	-	-	86.51	-	-	62.32
	Relation-Metric [31]*	84.46	84.67	84.57	67.97	58.18	62.68
	SpERT†	88.25	89.64	**88.94**	73.04	70.00	**71.47**
	SpERT‡	85.78	86.84	**86.25**	74.75	71.52	**72.87**
SciERC	SciIE [20]†	67.20	61.50	64.20	47.60	33.50	39.30
	DyGIE [21]†	-	-	65.20	-	-	41.60
	DyGIE++ [34]†	-	-	67.50	-	-	48.40
	SpERT† (using BERT)	68.53	66.73	67.62	49.79	43.53	46.44
	SpERT† (using SciBERT)	70.87	69.79	**70.33**	53.40	48.54	**50.84**
ADE	Multi-head [3]†	84.72	88.16	86.40	72.10	77.24	74.58
	Multi-head + AT [2]†	-	-	86.73	-	-	75.52
	CNN + Global features [17]*	79.50	79.60	79.50	64.00	62.90	63.40
	BiLSTM + SDP [16]*	82.70	86.70	84.60	67.50	75.80	71.40
	Relation-Metric [31]*	86.16	88.08	87.11	77.36	77.25	77.29
	SpERT (without overlap)	89.02† 89.26‡	88.87† 89.26‡	**88.94**† **89.25**‡	78.09	80.43	**79.24**
	SpERT (with overlap)	88.69† 88.99‡	89.20† 89.59‡	**88.95**† **89.28**‡	77.77	79.96	**78.84**

句子在不同tokens的性能比较：



总结

此模型利用BERT对词，小语段，整体上下文进行训练，符合直观而且性能最佳。 良好的预训练模型为后续各种任务的实现提供了很好的起点。 个人认为美中不足的是关系提取的时候，实体两端（左实体左端以及右实体右端）的文段被忽视，如“Gina is Tom’s cat”的从属关系可能就被忽略了。

(二) 知识图谱构建（已得三元组）

Table 3. Global H@1, H@10, MR, and MRR Results for All LP Models on Each Dataset

		FB15k				WN18				FB15k-237				WN18RR				YAGO3-10			
		H@1	H@10	MR	MRR	H@1	H@10	MR	MRR	H@1	H@10	MR	MRR	H@1	H@10	MR	MRR	H@1	H@10	MR	MRR
Matrix Decomposition Models	DistMult	73.61	86.32	173	0.784	72.60	94.61	675	0.824	22.44	49.01	199	0.313	39.68	50.22	5913	0.433	41.26	66.12	1107	0.501
	ComplEx	**82.32**	**91.04**	**33**	**0.855**	94.43	96.15	**190**	0.951	**27.18**	**55.80**	**144**	**0.367**	44.27	**58.06**	2867	0.489	**50.09**	**71.29**	**793**	**0.577**
	ANALOGY	65.59	83.74	126	0.726	92.61	94.42	808	0.934	12.59	35.38	476	0.202	35.82	38.00	9266	0.366	19.21	45.65	2423	0.283
	SimplE	66.13	83.63	138	0.726	93.25	94.58	759	0.938	10.03	34.35	651	0.179	38.27	42.65	8764	0.398	35.76	63.16	2849	0.453
	HolE	75.85	86.78	211	0.800	93.11	94.94	650	0.938	21.37	47.64	186	0.303	40.28	48.79	8401	0.432	41.84	65.19	6489	0.502
	TuckER	72.89	88.88	39	0.788	94.64	95.80	510	0.951	25.90	53.61	162	0.352	42.95	51.40	6239	0.459	46.56	68.09	2417	0.544
Geometric Models	TransE	49.36	84.73	45	0.628	40.56	94.87	279	0.646	21.72	49.65	209	0.31	2.79	49.52	3936	0.206	40.57	67.39	1187	0.501
	STransE	39.77	79.60	69	0.543	43.12	93.45	208	0.656	22.48	49.56	357	0.315	10.13	42.21	5172	0.226	3.28	7.35	5797	0.049
	CrossE	60.08	86.23	136	0.702	73.28	95.03	441	0.834	21.21	47.05	227	0.298	38.07	44.99	5212	0.405	33.09	65.45	3839	0.446
	TorusE	68.85	83.98	143	0.746	94.33	95.44	525	0.947	19.62	44.71	211	0.281	42.68	53.35	4873	0.463	27.43	47.44	19455	0.342
	RotatE	73.93	88.10	42	0.791	94.30	96.02	274	0.949	23.83	53.06	178	0.336	42.60	57.35	3318	0.475	40.52	67.07	1830	0.498
	HAKE	74.49	88.44	43	0.796	94.34	**96.19**	230	0.950	24.91	54.20	184	0.347	45.28	**58.06**	3300	**0.497**	46.27	69.54	2068	0.546
Deep Learning Models	ConvE	59.46	84.94	51	0.688	93.89	95.68	413	0.945	21.90	47.62	281	0.305	38.99	50.75	4944	0.427	39.93	65.75	2429	0.488
	ConvKB	11.44	40.83	324	0.211	52.89	94.89	202	0.709	13.98	41.46	309	0.230	5.63	52.50	3429	0.249	32.16	60.47	1683	0.420
	ConvR	70.57	88.55	70	0.773	94.56	95.85	471	0.950	25.56	52.63	251	0.346	43.73	52.68	5646	0.467	44.62	67.33	2582	0.527
	InteractE	72.58	88.68	623	0.786	94.63	95.55	403	0.950	26.35	53.73	185	0.355	42.72	51.96	5056	0.458	46.61	68.51	2182	0.543
	CapsE	1.93	21.78	610	0.087	84.55	95.08	233	0.890	7.34	35.60	405	0.160	33.69	55.98	**720**	0.415	0.00	0.00	60674	0.000
	RSN	72.34	87.01	51	0.777	91.23	95.10	346	0.928	19.84	44.44	248	0.280	34.59	48.34	4210	0.395	42.65	66.43	1339	0.511
	AnyBURL	81.47	87.96	217	0.837	**94.72**	95.62	358	**0.953**	26.92	51.83	276	0.353	**45.77**	57.63	4035	**0.497**	49.17	68.48	1818	0.560

The best results of each metric for each dataset are marked in bold and underlined. The hyperparameters used for each model to obtain the reported results can be found in Appendix G.

相关工作：将三元组表示为知识图谱，关键是找到一种映射以便表示出实体和关系。

基于复数域的ComplEx也可以达到很不错的性能：ComplEx关键思想是使用复值嵌入来表示知识图谱中的实体和关系。在 ComplEx 中，每个实体和关系都用一个复数向量表示。这些复数向量包含了两个部分：实数部分（实部）和虚数部分（虚部）。通过使用复数嵌入，ComplEx 能够更好地捕捉实体和关系之间的语义信息，特别是在处理对称和反对称关系时。ComplEx 模型的训练目标是通过最大化正确三元组得分与错误三元组得分之间的差异来学习实体和关系的复数嵌入。通过学习到的复数嵌入，模型可以有效地进行链接预测任务，包括实体预测和关系预测。

❶ 模型：HAKE

（Learning Hierarchy-Aware Knowledge Graph Embeddings for Link Prediction DOI：https://doi.org/10.1609/aaai.v34i03.5701）

本文模型HAKE基于极坐标来表示实体和关系，达到了：有关系的实体的模量相同，没有关系的实体的模量不同；有关系的实体的表示差异在于相位差。达到了实体-关系的唯一嵌入（相比之下，如RotatE模型等仅采用平移表示关系，会造成在流形上产生若干符合要求的点位，产生不固定性）。

❷ 技术细节：

模的关系：

$$\mathbf{h}_m \circ \mathbf{r}_m = \mathbf{t}_m, \text{ where } \mathbf{h}_m, \mathbf{t}_m \in \mathbb{R}^k, \text{ and } \mathbf{r}_m \in \mathbb{R}_+^k.$$

由此得到由模评判关系好坏程度的分数：

$$d_{r,m}(\mathbf{h}_m, \mathbf{t}_m) = \left\| \mathbf{h}_m \circ \mathbf{r}_m - \mathbf{t}_m \right\|_2$$

三者联系越正确dr,m越小，反之越大。

相位关系：

$$(\mathbf{h}_p + \mathbf{r}_p) \bmod 2\pi = \mathbf{t}_p, \text{where } \mathbf{h}_p, \mathbf{r}_p, \mathbf{t}_p \in [0, 2\pi)^k$$

由此得到由相位评判关系好坏程度的分数：

$$d_{r,p}(\mathbf{h}_p, \mathbf{t}_p) = \| \sin((\mathbf{h}_p + \mathbf{r}_p - \mathbf{t}_p)/2) \|_1$$

同时考虑的距离以及分数函数：

The distance function of HAKE is:

$$d_r(\mathbf{h},\mathbf{t}) = d_{r,m}(\mathbf{h}_m,\mathbf{t}_m) + \lambda d_{r,p}(\mathbf{h}_p,\mathbf{t}_p),$$

where $\lambda \in \mathbb{R}$ is a parameter that learned by the model. The corresponding score function is

$$f_r(\mathbf{h},\mathbf{t}) = d_r(\mathbf{h},\mathbf{t}) = -d_{r,m}(\mathbf{h},\mathbf{t}) - \lambda d_{r,p}(\mathbf{h},\mathbf{t}).$$

损失函数：

To train the model, we use the negative sampling loss functions with self-adversarial training (Sun et al. 2019):

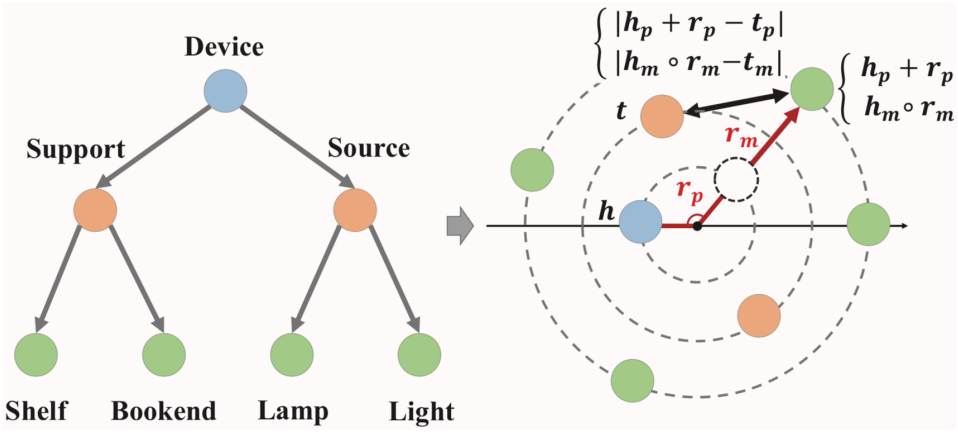
$$L = -\log\sigma(\gamma - d_r(\mathbf{h},\mathbf{t})) - \sum_{i=1}^{n} p(h'_i,r,t'_i)\log\sigma(d_r(\mathbf{h}'_i,\mathbf{t}'_i) - \gamma),$$

where γ is a fixed margin, σ is the sigmoid function, and (h'_i,r,t'_i) is the ith negative triple. Moreover,

$$p(h'_j,r,t'_j|\{(h_i,r_i,t_i)\}) = \frac{\exp\alpha f_r(\mathbf{h}'_j,\mathbf{t}'_j)}{\sum_i \exp\alpha f_r(\mathbf{h}'_i,\mathbf{t}'_i)}$$

is the probability distribution of sampling negative triples, where α is the temperature of sampling.

嵌入图示：

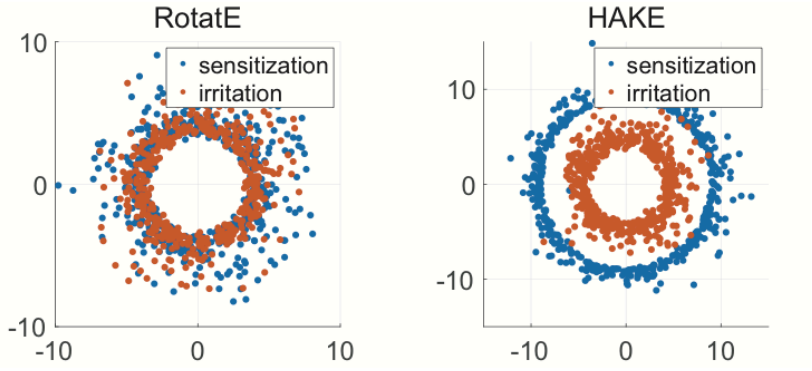


❷ 实验

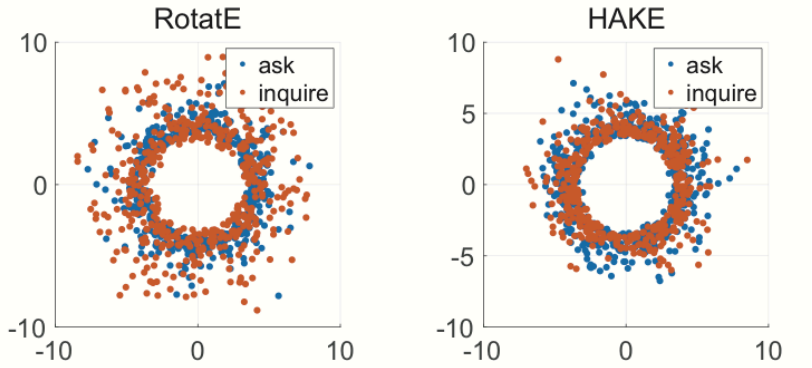
	WN18RR				FB15k-237				YAGO3-10			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE	.226	-	-	.501	.294	-	-	.465	-	-	-	-
DistMult	.43	.39	.44	.49	.241	.155	.263	.419	.34	.24	.38	.54
ConvE	.43	.40	.44	.52	.325	.237	.356	.501	.44	.35	.49	.62
ComplEx	.44	.41	.46	.51	.247	.158	.275	.428	.36	.26	.40	.55
RotatE	.476	.428	.492	.571	.338	.241	.375	.533	.495	.402	.550	.670
ModE	.472	.427	.486	.564	.341	.244	.380	.534	.510	.421	.562	.660
HAKE	**.497**	**.452**	**.516**	**.582**	**.346**	**.250**	**.381**	**.542**	**.545**	**.462**	**.596**	**.694**

在所有模型（包括ComplEx）的所有指标上sota（和综述的性能表有区别）

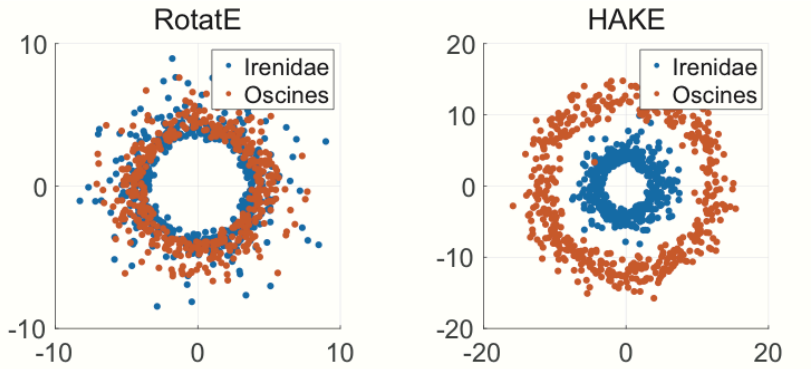
与相似模型RotatE的比较以及实例数据集嵌入图：



(a) *(sensitization, _hypernym, irritation)*



(b) *(ask, _verb_group, inquire)*



(c) *(Irenidae, _member_meronym, Oscines)*

总结：

坐标系有很多，映射空间也是多样的，可以尝试诸多数学里的工具来实现世界物体关系和数字向量之间的映射。

（三）基于知识图谱，元路径的推荐系统下游任务应用

❶ 模型：TEMR（Temporal Meta-path Guided Explainable Recommendation　　DOI：https://doi.org/10.1145/3437963.3441762）

突破：

基于时间序列，在全局知识图谱的体系下增加了时间的信息，考虑了用户的动态行为，既增强了性能，也增加了可解释性。

涉及到了序列信息的提取，采用了transformer，有很好的性能。

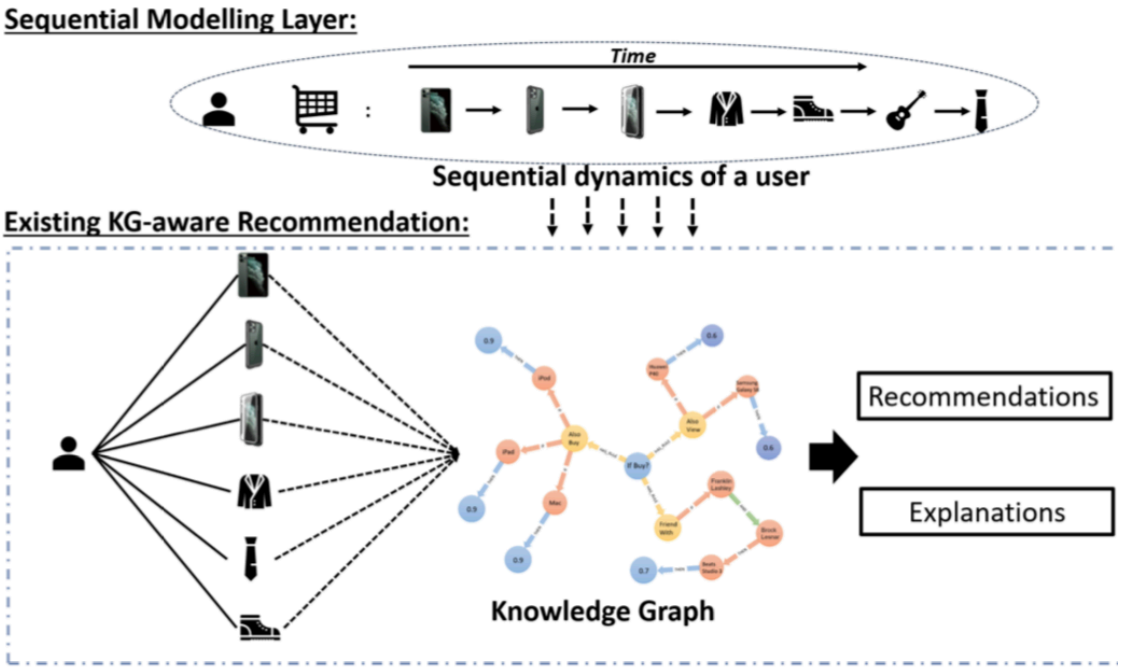


Figure 1: The concept of TEMR. TEMR contributes a sequential dynamic modelling layer on top of existing knowledge-aware explainable recommendation (the dashed box on the bottom).

❷ 技术细节：

TEMR的主要由四个部分组成。首先，初始化用户和物品，使用DeepWalk将用户和物品实体嵌入。其次，不仅仅利用元路径，还挖掘和提取有意义的顺序（时间）和非顺序的元路径实例，以改善推荐性能和个性化。在这一步中，获取了连续物品之间的物品–物品实例路径。在使用注意力机制嵌入实例和用户/物品向量之后，将实例的权重视为特定用户的推理路径的权重。利用不同的推理路径权重来更新物品嵌

入，使物品包含推理信息。这一步还对用户的顺序购买信息进行建模，将前一个物品的特征传递给下一个物品。最后，将物品嵌入、用户嵌入和实例输入推荐网络进行推荐。

① 初始化用户和物品表示

通过在用户–物品二部图中进行随机截断的随机游走，将涉及的用户和物品学习为潜在表示，这相当于DeepWalk中句子的等价物。

DeepWalk通过基于word2vec的skipgram来优化游走中实体之间的共现概率。

② 考虑元路径上下文

仅考虑用户–物品路径对于推荐解释性是有限的，因为用户–物品路径仅代表用户的一般购物兴趣。相比之下，物品–物品路径更具表现力，可以通过探索物品之间的高阶关系反映出不同的原因，例如互补产品（例如手机和手机壳）、已知物品的可替代品（例如 iPhone – 手机膜 – 华为手机）、与其他人共同购买的产品等。此外，物品–物品路径有时还充当顺序建模信号，捕获用户购买的每个连续物品之间的时间依赖性，这对于顺序可解释的推荐至关重要。

尽管元路径在探索基于知识的推荐中具有强大的表达能力，但主要挑战仍然存在于因为元路径的数量太大而难以处理。我们需要简化和挑选，如果下一个节点和当前节点的嵌入相似，那么到下一个节点的路径更有可能连接。因此，根据这一假设，使用基于相邻节点相似性的方法来抽样路径中的下一个节点。具体来说，对于每个指定的元路径模式，我们通过计算当前节点和候选出节点之间的相似性来衡量优先度，这样的优先分数直接反映了两个节点之间的关联程度。通过这种方法，我们对每个用户–物品对和物品–物品对抽样并获取一些元路径实例参与推荐模块的训练。

③ 多元路径的组合的表示，揭示高阶、复杂原因

将用户–物品元路径实例和物品–物品元路径实例视为句子，将节点视为句子中的标记。然后，他们使用Word2Vec方法和Mean(·)操作来学习路径嵌入。接下来，采用了多头自注意力机制来学习基于元路径的上下文。连续购买两个物品的原因并不是简单唯一的。比如顾客在购买新手机后立即购买手机壳的原因可能是a.他/她的朋友拥有一款类似的手机并购买了这款手机壳。 b.手机壳是购买的新手机最流行的搭配。 c. 手机壳的颜色符合顾客的喜好。潜在的原因可能不止列举的几个，而且它们可以通过使用各种元路径来表示。

基于这一观察，利用了Transformer模型的多头自注意力机制，从多个路径实例中学习组合特征，以更好地描述知识图谱中每个连接实体对之间的复杂原因。

$$Attention(Q_\phi, K_\phi, V_\phi) = Softmax(\frac{Q_\phi K_\phi^T}{\sqrt{d_k}})V_\phi,$$

$$MultiHead(Q_\phi, K_\phi, V_\phi) = Concat(head_1, ..., head_m)W^O$$

④ 基于物品–物品元路径实例来建模时序依赖关系

通过物品关注单元传递前一个项目的信息：TMER框架通过物品关注单元来捕获前一个物品传递的信息。这个关注单元聚合了前一个物品–物品之间的连接信息，以及任何其他可能的关联信息。

通过特定的采样路径实例捕获物品–物品之间的连接，从而达到找寻物品之间的关系的目的，同时也利用了时序信息。

⑤ 参数更新

由于上一个物品也有重要的参考价值，采用了两层注意力机制。

$$\mathbf{h}_i^{(1)} = ReLU(W_{i-1}\mathbf{h}_{i-1} + W_{\phi_{i-1\to i}}^{(1)}\mathbf{h}_{\phi_{i-1\to i}} + \mathbf{b}_i^{(1)}) \odot \mathbf{h}_{i-1}$$

$$\mathbf{h}_i^{(2)} = ReLU(W_i\mathbf{h}_i + W_{\phi_{i-1\to i}}^{(2)}\mathbf{h}_{\phi_{i-1\to i}} + \mathbf{b}_i^{(2)}) \odot \mathbf{h}_i$$

⑥ 完成模型

用户、项目和实例信息（在3.4.2中计算）连接成一个向量，通过具有可解释实例的多层感知机（MLP）获得用户–项目的预测分数。投入多层感知机（其中MLP包含一个具有ReLU作为激活函数和sigmoid函数作为输出层的两隐藏层神经网络）

将隐式反馈损失和负采样损失作为损失函数：

$$loss_{u,i} = -E_{j\sim P_{neg}}\left[\log\left(1 - r_{u,j}\right)\right]$$

❸ 实验

① baseline：

· GRU4Rec是一种基于GRU的基于会话的推荐方法。对于每个用户，我们将训练项视为一个会话。

· NARRE[2]: NARRE利用神经注意机制构建可解释的推荐系统。

· MCRec[16]: MCRec利用协同注意机制开发了一个深度神经网络，用于学习丰富的基于元路径的上下文信息进行推荐。

· NFM[12]: NFM有效地结合了线性因子分解机（FM）和非线性神经网络，用于在稀疏环境下的预测。

· FMG[43]: FMG使用FM和矩阵分解（MF）进行推荐，并且在性能上胜过了最先进的FM和其他基于HIN的推荐算法。

② 评估指标

· HR@k (Hit Ratio@k): 表示在前k个推荐结果中命中用户真实交互的比例。例如，HR@5表示在前5个推荐结果中命中用户真实交互的比例。

· NDCG@k (Normalized Discounted Cumulative Gain@k): 衡量推荐结果的排序质量，考虑了命中位置的贡献以及位置的折扣因素。NDCG@k值越高，表示推荐结果排序的质量越好。

③ 结果

Table 3: Performance Comparison with Baselines.

Datasets	Metrics	GRU4Rec	NARRE	MCRec	NFM	FMG	TMER
Amazon Musical Instruments	HR@1	0.6493	0.6324	0.5922	0.3929	0.4174	**0.8467**
	HR@5	0.6493	0.6408	0.6289	0.4169	0.4328	**0.9507**
	HR@10	0.6502	0.6764	0.6522	0.4631	0.6466	**0.9739**
	HR@20	0.7647	0.7988	0.6798	0.5484	0.8799	**0.9865**
	NDCG@1	0.0490	0.1025	0.1584	0.0549	0.1003	**0.2165**
	NDCG@5	0.0596	0.1224	0.1590	0.0736	0.1221	**0.2316**
	NDCG@10	0.0632	0.1572	0.1590	0.1013	0.1221	**0.2432**
	NDCG@20	0.0664	0.1603	0.1592	0.1432	0.1597	**0.2614**
Amazon Automotive	HR@1	0.6537	0.6633	0.6360	0.3362	0.4032	**0.9072**
	HR@5	0.6537	0.7063	0.6385	0.4063	0.4645	**0.9634**
	HR@10	0.6576	0.7064	0.6412	0.6024	0.8324	**0.9697**
	HR@20	0.8484	0.7963	0.6487	0.6203	0.9596	**0.9751**
	NDCG@1	0.0440	0.3343	0.1984	0.1549	0.3242	**0.4064**
	NDCG@5	0.0642	0.4024	0.2080	0.2742	0.3241	**0.4691**
	NDCG@10	0.0743	0.4536	0.2138	0.3527	0.3360	**0.4953**
	NDCG@20	0.0773	0.4761	0.2218	0.3533	0.3897	**0.5253**
Amazon Toys and Games	HR@1	0.7776	0.8235	0.6580	0.3563	0.4562	**0.8339**
	HR@5	0.7777	0.8644	0.6619	0.4794	0.5536	**0.9442**
	HR@10	0.7792	0.8933	0.6668	0.7335	0.8704	**0.9580**
	HR@20	0.8882	0.9025	0.6754	0.7544	0.8978	**0.9662**
	NDCG@1	0.1863	0.2065	0.1709	0.0954	0.1124	**0.2154**
	NDCG@5	0.2046	0.2564	0.1800	0.1543	0.1238	**0.2901**
	NDCG@10	0.2947	0.2933	0.1855	0.2566	0.1462	**0.3274**
	NDCG@20	0.2947	0.3364	0.1928	0.2594	0.1514	**0.3711**

总结：

基于时间序列的动态信息提取是十分有效的，利用良好的预训练模型transformer（和上篇文章一样）可以提供很好的性能，尤其是在序列信息的处理上，本模型的多头，多关系路径的处理利用transformer是合适的。除此之外，可解释性也是研究的重要方向，如本文的时间序列的加入以及高阶信息的捕获（如认识的人购买手机也会造成影响）使得模型的可解释性大大提高，这在推荐的任务上是很好的。