Multi-grained Hypergraph Interest Modeling for Conversational Recommendation

Chenzhan Shang, Yupeng Hou, Wayne Xin Zhao, Yaliang Li, Jing Zhang

26 October 2023

摘要

为了解决传统对话推荐系统 (CRS) 中的数据稀缺性问题，MHIM 这篇论文使用超图来建模用户的历史会话以形成基于会话的超图，并使用外部知识图并构建一个基于知识的超图，从不同角度捕捉用户偏好，然后在这两种超图上进行多粒度超图卷积，并利用对比子图预训练来开发具有兴趣感知的 CRS。本次复现工作在其基础上进一步做出两方面的改进：1）改变超边扩展的条件；2）引入外部数据集进行数据增强，实验结果表明两种措施均在一定程度上带来了性能的提高。

关键词：对话推荐系统；超图学习；数据增强

1 引言

对话式推荐系统（Conversational Recommender System，CRS）是一个与用户通过自然语言的多轮对话进行交互的系统，其目标是为用户提供高质量的推荐以满足用户的即时信息需求。典型的 CRS 包括一个对话模块和一个推荐模块。对话模块的目标是理解用户的话语并生成有信息量的回应。推荐模块则专注于从文本信号中捕捉用户的偏好并推荐适当的物品。

尽管已经付出了很多努力来开发有效的 CRS，但大多数方法仍然集中于利用当前对话会话的有限上下文信息，通常受到数据稀缺问题的困扰。为了解决这个问题，作者基于用户在当前对话之前可能已经多次与 CRS 互动过提出解决方案。这些历史对话会话包含了捕捉用户偏好的关键信息，而且在实际系统中很容易收集。因此，他们考虑全面捕获隐藏在复杂历史数据中的用户兴趣，以丰富当前对话会话的有限上下文。

在提高 CRS 性能时，作者指出存在两个主要挑战：

会话内部和会话之间关系复杂。历史会话之间的关联关系既复杂又多样。一个会话通常集中在一个特定的主题上，涉及到少量重要的实体（例如，在电影推荐中可能是演员或导演）。由于对话上下文有限，准确捕捉会话内部实体之间的相关性以及建立会话之间关系都非常困难。

历史数据稀缺。在实际对话式推荐场景中，历史数据仍然很稀缺。用户的历史对话会话数量和物品的互动记录都遵循长尾分布。大多数用户只与系统互动几次，因此历史对话可能不足以进行准确的用户建模。此外，大多数物品可能只出现几次，这使得难以学习推荐所需的信息丰富的实体表示。

为了解决这些挑战，作者使用了超图来表示历史对话会话中的复杂语义关系。不同于标准图，超图中的一个超边可以连接多个顶点，这对于直接关联多个实体特别合适。超图的不同超边也共享通用的顶点，这对于建模反映用户兴趣的关键属性可能很有用。

作者提出了一种名为"Multi-grained Hypergraph Interest Modeling（MHIM）"的方法用于对话式推荐。为了改进 CRS，他们考虑两种方式来构建超图：

- 为了捕捉历史数据中的复杂关系，他们将每个会话中出现的项目连接一个超边，构建了一个基于会话的超图。
- 为了解决数据稀缺问题，他们引入了一个外部知识图（KG）并构建了一个基于知识的超图。

为了处理这两种超图，作者引入了多粒度超图卷积，以模拟历史用户兴趣。其中，基于会话的超图捕获历史数据中的粗粒度用户偏好，而基于知识的超图捕获了知识图中的细粒度用户兴趣。此外，为了学习信息丰富的实体表示，作者提出了通过对比子图鉴别来预训练知识图编码器。

为了进行准确的推荐，作者设计了一个超图感知的注意力模块来生成用户表示，该模块考虑了从历史数据中学到的多粒度用户偏好。对于对话任务，作者进一步利用历史对话会话来构建具有兴趣感知的回应生成器。作者表示这是第一个通过超图来利用历史数据对 CRS 中的用户兴趣建模的工作。通过与多个竞争性基线方法进行对比，实验证明了他们的方法在推荐和对话任务中的有效性。

2 相关工作

2.1 推荐系统

对话式推荐。对话式推荐系统通过多轮对话模拟用户兴趣并提供高质量的推荐。现有关于 CRS 的研究大致可以分为两类，即基于属性的 CRS 和基于生成的 CRS。**基于属性的 CRS** [6, 19, 20, 28, 33, 45, 51] 通常通过询问有关项目属性的查询并使用预定义模板生成响应来捕获用户偏好 [19, 33]。这些方法大多数逐渐缩小假设空间，以在更少的回合内搜索合适的项目。然而，这种 CRS 没有足够重视以自然语言生成类似人类的响应，这可能会降低用户体验。**基于生成的 CRS** [4, 7, 22–26, 29, 38, 47, 50, 52, 53] 通过采用 Seq2Seq 架构 [32, 34] 生成流畅的话语来缓解这个问题，从而构建了一个对话任务和推荐任务的端到端框架。有研究人员发布了一个基准数据集 ReDial [22]，其中包含有关电影推荐的对话。进一步的研究结合外部数据来改进用户偏好建模和推荐，包括面向实体的知识图谱 [4, 29, 50]、面向单词的知识图谱 [52] 和评论信息 [26]。为了有效利用外部数据，研究人员提出了一种从粗到细的对比学习框架 [26] 来改进数据语义融合。最近的一项研究 [23] 首先强调，用户的历史对话会话和相似的用户对于用户偏好建模至关重要。然而，复杂的历史数据背后的用户兴趣尚未得到全面捕捉。作者的工作通过利用历史对话和大规模外部知识扩展了第二类研究。关键的新颖之处在于通过多粒度超图卷积进行用户兴趣建模，以实现更好的推荐。

基于会话的推荐。基于会话的推荐侧重于动态捕获用户兴趣以基于短期会话进行推荐，其中会话是指在短时间内发生在多个用户之间的项目交互。GRU4Rec [13] 利用门控循环单

元（GRU）对用户行为进行顺序建模，这有助于利用复杂的会话内和会话间关系进行推荐。NARM [21] 提出将注意力机制纳入循环神经网络（RNN）中以准确捕获用户兴趣。最近，图神经网络（GNN）已被用于基于会话的推荐 [31, 40, 44]，因为它们在建模复杂的图结构上下文数据方面具有巨大的潜力。此外，研究人员建议将基于会话的场景扩展到基于多会话的场景 [39]，整合更多的上下文信息以获得准确的推荐。与基于会话的推荐相比，作者的工作将用户参与的多轮对话视为会话，并通过对会话推荐的短期序列进行建模来捕获用户兴趣。

2.2 图表示学习

图神经网络。使用图结构数据进行建模具有巨大的潜力，图神经网络（GNN）近年来引起了广泛关注。现有的 GNN 方法可分为谱方法 [17] 和空间方法 [11, 30, 35]。大多数方法遵循消息传递 [10] 方案来聚合来自邻居节点的结构信息。尽管 GNN 对于图数据建模非常有效，但它们通常需要大量特定于任务的数据来进行端到端训练。受自然语言处理 [3, 8] 和计算机视觉 [5, 12] 预训练最新进展的推动，研究人员致力于 GNN 的预训练 [14, 15, 27, 36, 48]。关键思想是在大量未标记图数据集或粗粒度监督数据集上预训练一个表达能力强的 GNN 编码器。现有的工作主要集中在设计适当的预训练任务，例如属性预测 [14]、图属性预测 [14]、图重建 [15] 和对比学习 [27, 36, 48]。在 CRS 领域，训练数据仍然不足。因此，作者建议通过对比学习 [27] 在大规模知识图谱 [18, 43] 上预训练图编码器。

超图学习。超图 [2] 概括了边的概念，使其连接两个以上的节点，并提供了一种捕获高阶关系的自然方法。它是通过与深度学习技术相结合的方式进行探索的。HGNN [9] 和 HyperGCN [46] 首先设计了超图卷积运算来处理高阶相关性。进一步引入注意力机制来提高性能 [1]。还有几项研究将超图学习与推荐系统相结合 [16, 37, 41, 42, 49]。HyperRec [37] 使用超图来模拟下一项推荐的短期用户偏好。DHCF [16] 捕获用户和项目之间的高阶相关性，用于一般的协同过滤。DHCN [42] 进一步利用超边间信息进行基于会话的推荐。MHCN [49] 将自监督学习集成到用于社交推荐的超图卷积网络的训练中。作者的工作首次通过超图学习来建模用户兴趣以进行对话推荐。

3 本文方法

3.1 本文方法概述

本文的总体模型框架如图 1所示。首先作者对历史数据建立了两种不同类型的超图：基于会话的粗粒度超图和基于知识的细粒度超图。然后使用 R-GCN 对超图进行编码后送入超图卷积提取特征，将两个粒度的信息融合在一起得到历史会话的表示；同时，通过对比子图预训练任务增强 KG 编码器并得到用户最新会话的表示。最后，将历史信息和当前兴趣融合起来对用户进行项目推荐和对话任务。

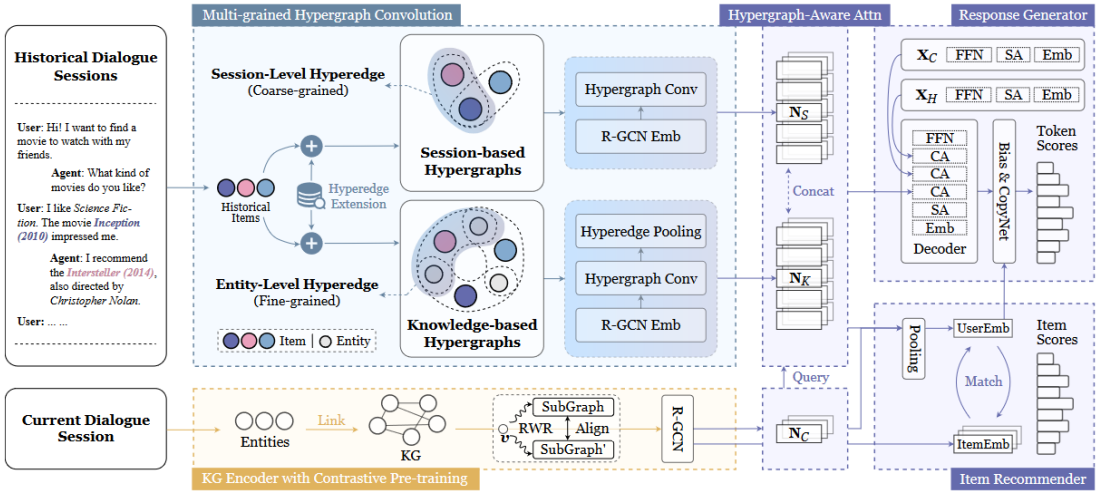

图 1. 总体模型框架

3.2 基于对比预训练的知识图编码

为了学习信息实体表示，本文提出通过对比子图判别来预训练 KG 编码器。由于 KG 为会话推荐任务提供了重要的外部信息，作者首先提出了针对特定任务的 KG 的表示学习方法。然后，通过引入对比子图预训练来描述增强训练方法。

3.2.1 特定任务的知识图编码

为了获取对话内容的语义，本文从话语中提取基本语义单元（即实体），并将其链接到 KG 条目，然后通过两跳搜索扩展这些实体。上述实体和扩展实体构成 CRS 的任务相关的知识图谱。采用 DBpedia 和 CN- DBpedia 作为外部知识图谱。考虑到这些关系对于学习实体表示至关重要，本文利用 R-GCN 来开发 KG 编码器，它通过显式地对关系语义进行建模来改进基本的 GCN 架构。

3.2.2 基于对比子图判别的改进 KG 编码器

受 CRS 数据集大小的限制，通常很难构建一个足够大的与任务相关的 KG 来训练 KG 编码器。因此，作者考虑利用大规模原始 KG 来改进 KG 编码器。为了减少不相关信息的影响，只保留 CRS 数据集中出现的关系，并利用它们从原始 KG 中跨越连接组件，称为扩展 KG。本文的想法是在扩展的 KG 上引入对比预训练来改进 KG 编码器。具体来说，利用子图作为对比实例，并使用子图实例判别作为预训练任务。该任务将每个子图视为一个不同的类别，并学习区分它们。

本文采用的是随机行走算法，从起始顶点 v 出发通过随机游走又回到顶点 v 作为一个新生成的子图。为了生成子图表示，首先使用 KG 编码器对子图实例中的节点进行编码，然后将节点嵌入求和并归一化为子图表示。为了构建对比样本，考虑从同一起始顶点派生的两个子图作为相似的实例对，其他子图视为不同的实例对。

考虑编码查询 q（即目标子图）和 $M+1$ 个编码键字典 $k_0, k_1 ... k_m$（即对比样本），作者假设字典中有一个单键 q 个匹配，用 k_+ 表示。在这项工作中采用了 InfoNCE 对比损失:

$$\mathcal{L} = -\log \frac{exp(q^T k_+/\tau)}{\sum_{i=0}^{M} exp(q^T k_i/\tau)} \tag{1}$$

3.3 用于历史用户兴趣建模的多粒度超图

3.3.1 超图卷积

本文开发了一个超图卷积网络来捕获高阶关系。通过引入了一个适当的归一化并以矩阵的形式重写卷积操作：

$$\mathrm{X}^{(l+1)} = \mathrm{D}^{-1}\mathrm{HB}^{-1}\mathbf{H}^{\top}\mathrm{X}^{(l)}\mathbf{W}^{(l)} \tag{2}$$

其中，$X^{(l)}$ 和 $X^{(l+1)}$ 分别是第 l 层和第 l+1 层的输入，D 和 B 分别是顶点和超边的度矩阵，H 为超图的关联矩阵，W 为权重参数。

超图卷积可以看作是一个两阶段的细化过程，对超图结构执行“顶点-超边-顶点”特征变换。首先，顶点特征通过权重矩阵 W 进行变换。然后，通过转置发生率矩阵 $H^{\top}$ 聚合顶点特征形成超边特征，并聚合相关超边特征以通过 H 生成细化顶点特征。引入顶点和超边的度矩阵（即 D,B）进行归一化。

3.3.2 基于会话的超图卷积

为了丰富当前对话会话的有限上下文，本文利用用户的历史对话会话进行偏好建模。作者观察到每个会话通常集中在单个主题上，例如，当用户正在寻找科幻小说时，出现在对话话语中的项目将与该主题相关。此外，来自某个用户的不同会话可能共享公共项目，这表明了内在的用户偏好。受这一观察的启发，作者将出现在同一会话中的项目提取为顶点以构建超边。然后，用户对应的所有超边通过共享项相互连接，构成基于会话的超图。构建两种粒度超图的方法示例如图 2所示。

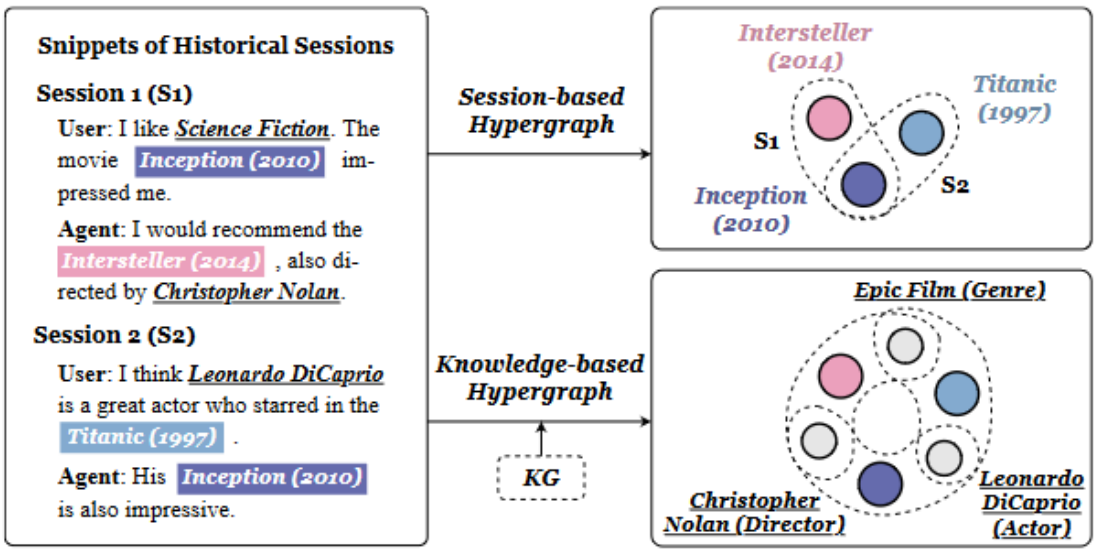

图 2. 构建两种粒度超图的方法示例

这两个会话包含两个电影项目，分别对应于基于会话的两个超边。此外，项目“Inception(2010)”同时被两个会话中的两个超边所连接。从形式上讲，历史对话中出现的物品构成

了一个历史物品集 I_H，由此组成的基于会话的超图，关联矩阵用 H_S 表示, 顶点和边的度矩阵分别表示为 D_S 和 D_B，编码后实体的表示为 N_H，其被送到基于会话的超图卷积中:

$$\mathrm{N}_S = \mathrm{HConv}\left(\mathrm{H}_S, \mathrm{D}_S,\ \mathrm{B}_S,\ \mathrm{N}_H\right) \tag{3}$$

其中 HConv(·) 是由公式 2所定义的。并且 N_S 是通过历史项目集 I_H 的基于会话的超图卷积增强的嵌入矩阵，从而得到一个从粗粒度的角度聚合历史对话会话的会话级语义。

3.3.3 基于知识的超图卷积

由历史对话会话提供的上下文信息可能不足以发现固有的用户偏好。因此，作者尝试引入外部 KG。具体来说，对在外部 KG 中找到对话中出现的项目，将每个历史项目及其 N-跳邻居作为超边。**其动机是顶点和扩展邻居可能具有共同的语义**。然后，所有从历史项目派生的超边通过共享实体相互连接，这些共享实体构成了基于知识的超图。给定一个基于知识的超图，它包括由历史物品及其 N-跳邻居组成的顶点集 ε_K，关联矩阵用 H_K 表示，顶点和边的度矩阵分别用 D_K 和 B_K 表示，ε_K 的表示构成嵌入矩阵 $N_K^{'}$，随后送入基于知识的超图卷积:

$$\widetilde{\mathrm{N}}_K = \mathrm{HConv}\left(\mathrm{H}_S, \mathrm{D}_S,\ \mathrm{B}_S,\ \mathrm{N}_K^{'}\right) \tag{4}$$

其中 $\widetilde{N}_K$ 是 ε_K 的基于知识的超图卷积增强的嵌入矩阵。基于知识超图的消息传递从存储在 KG 中的三元组中聚合相似的语义，这促进了从这种细粒度的角度从历史对话会话中捕获实体级别的用户兴趣。

3.3.4 相似对话的超边扩展

为了进一步缓解用户历史对话的稀缺性，本文提出基于相似对话的公共项目进行超边扩展。基本思想是对话中的共同项目通常对应于用户之间的相似偏好。具体来说，从对话中提取项目并构建超边，形成超边集合。然后，基于当前对话与超边集合中超边之间的公共项目比率，选择一定数量的超边进行扩展。

3.4 用户兴趣感知的对话推荐

3.4.1 基于超图感知的注意力用户表示

作者的目标是利用相关的历史兴趣来增强当前用户兴趣的建模。为此，提出了一个超图感知多头注意力层将历史项目表示与当前实体表示相结合。上面已经学习了基于会话和基于知识的项目表示，即 N_S 和 N_K。接下来，作者从当前对话会话中获取实体表示，表示为 N_C，将其作为处理中的两个项目 N_S 和 N_K 的查询。形式上，计算历史项目的综合表示:

$$N_{SK} = MHA(N_C, [N_S; N_K], [N_S; N_K]) \tag{5}$$

其中 $MHA(Q, K, V)$ 定义了一个多头注意力函数，该函数以查询矩阵 Q、键矩阵 K 和值矩阵 V 作为输入。这种方式可以利用所构建的超图结构中的相关实体兴趣，同时考虑多粒度 (会话级和实体级) 的语义。因此，可以通过融合相关的历史偏好来增强当前用户的兴趣。然后，采用池化层 (平均池化) 来融合历史和当前用户兴趣，并获得最终的用户表示 $\mathbf{u}$:

$$\mathbf{u} = Pooling([Pooling(N_{SK}); N_C]) \tag{6}$$

3.4.2 物品推荐

在此基础上，作者从历史对话会话中同时考虑了会话级和实体级语义，获得了多粒度超图卷积增强的用户表示。进一步计算将项目推荐给用户 **u** 的概率：

$$P_{rec} = Softmax(\mathbf{u} \cdot N_I^{\top}) \tag{7}$$

为了学习模型的参数，作者采用交叉熵损失作为目标函数：

$$\mathcal{L}_{rec} = -\sum_{j=1}^{B}\sum_{i=1}^{|I|}\left[-\left(1-y_{ij}\right)\cdot\log\left(1-P_{\mathrm{rec}}^{(j)}(i)\right)+y_{ij}\cdot\log\left(P_{\mathrm{rec}}^{(j)}(i)\right)\right] \tag{8}$$

3.4.3 回复生成

作者采用 Transformer 开发了一个用于对话任务的编码器-解码器框架，还引入了两个独立的编码器来分别编码历史和当前对话。然后，学习的表示被输入解码器作为交叉注意力信号。形式上，在每个解码器层，作者首先在自注意力层和交叉注意力层之后获得文本表示：

$$A_0^n = MHA(R^{n-1}, R^{n-1}, R^{n-1}) \tag{9}$$

$$A_1^n = MHA(A_0^n, N_{SK}, N_{SK}) \tag{10}$$

其中 R^{n-1} 是第 (n - 1) 层的解码器中的入矩阵，N_{SK} 是由多粒度超图卷积增强的当前对话的项目表示。然后，作者将历史会话和当前对话会话的表示融合到解码器中。为了避免对历史对话的过度拟合，作者引入了一个超参数 β 来实现之间的权衡：

$$A_2^n = MHA(A_1^n, X_C, X_C) \tag{11}$$

$$A_3^n = MHA(A_1^n, X_H, X_H) \tag{12}$$

$$A_4^n = \beta \cdot A_2^n + (1-\beta)\cdot A_3 \tag{13}$$

其中 X_C 是当前对话会话编码器输出的嵌入矩阵，X_H 是历史对话会话编码器输出的嵌入矩阵。本文的方法与之前研究的主要区别在于，采用两个独立的编码器分别对当前对话和历史对话进行编码并将多粒度超图卷积增强的项目表示融合到解码器中。最后，通过前馈网络层获得解码器层输出：

$$R^n = FFN(A_4^n) \tag{14}$$

此外，生成的响应预计会反映用户的兴趣，并包含不同的推荐项目。因此，采用了用户兴趣感知偏见和另一种由复制机制产生的与项目相关的偏见。给定预测序列，下一个令牌概率计算为:

$$P_{gen}\left(y_i \mid y_1, \ldots, y_{i-1}\right) = P_1\left(y_i \mid \mathbf{R}_i\right) + P_2\left(y_i \mid \boldsymbol{u}\right) + P_3\left(y_i \mid \mathbf{R}_i, \boldsymbol{u}\right) \tag{15}$$

其中 $P_1(\cdot)$ 是以解码器输出 R_i 作为输入生成的词汇概率，$P_2(\cdot)$ 是用户兴趣感知偏见，指的是通过将用户表示融入到对话生成中来使模型偏向于生成那些与用户兴趣相关的词汇或句子；$P_3(\cdot)$ 表示由复制机制产生的与项目相关的偏见，复制机制指的是生成的对话并不是完全重新生成的，而是可以通过从用户历史对话中进行复制得到的，与项目相关的偏见指的就是在历

任务	指标	指标含义
Item 推荐 (K=10,50)	Recall@K	系统在前 K 个推荐结果中与用户兴趣相关的项目个数。 Recall@K= 正确检索的项目数 / 用户实际感兴趣的项目数
	MRR@K	系统在前 K 个推荐结果中，第一个用户真正感兴趣的项目的排名的倒数的平均值。 MRR@K = 1 / 第一个相关项目的排名
	NDCG@K	指系统在前 K 个推荐或检索结果中的归一化折损累积增益。 NDCG@K = DCG@K / IDCG@K
对话生成 (n=2,3, 4)	Distinct n-gram	衡量生成的文本中有多少不同的 n-gram（连续的 n 个单词或字符序列），以评估文本生成的多样性和丰富性。

表 1. 任务评估指标

Dataset	**#Dialogues**	**#Users**	**#Items**	**Sparsity**
ReDial	11,348	956	6,924	99.9843%
TG-ReDial	10,000	1,482	33,834	99.9973%

表 2. 数据集

史对话中选择复制时选择的是那些与项目有关的句子进行复制。最后，对话模块使用交叉损失进行训练:

$$\mathcal{L}_{gen} = -\sum_{i=1}^{B}\sum_{t=1}^{T}\log\left(P_{gen}\left(y_t \mid y_1, \ldots, y_{t-1}\right)\right) \tag{16}$$

3.5 评估指标

本文在两个任务，即项目推荐和对话生成上进行实验并评估模型性能，相关评估指标表 1所示。

4 复现细节

4.1 论文复现

本文相关代码已开源：源代码地址，本次复现工作在开源代码的基础上进行。

数据准备。原论文在 ReDial 和 TG-ReDial 两个数据集上进行训练和评估。ReDial 是一个英语会话式推荐数据集，是通过众包工作者在 Amazon Mechanical Turk 上按照一套详尽的指令构建的。TG-ReDial 是一个半自动创建的中文会话式推荐数据集。两个数据集的统计信息如表2所示。

从原文作者提供的数据集链接：数据集地址。下载处理后的数据集，解压缩 "*data_contrast.zip*" 并将其移动到 "*contrast/*" 文件夹下，解压缩 "*data_mhim.zip*" 并将其移到 "*mhim/*" 文件夹下。

复现。复现工作按作者的方法先在两个数据集上分别进行了对比子图预训练，以训练改进 KG 编码器：

```
cd Contrast
python run.py -d redial -g 0
python run.py -d tgredial -g 0
```

预训练得到的模型 save/dataset/#epoch-epoch.pth 复制到 MHIM/pretrain/dataset/下进行推荐和对话两个模型的训练。

在两个数据集 redial 和 hredial 上运行 *run_crslab.py* 进行模型的训练和测试：

```
cd ../MHIM
python run_crslab.py --config /mhim/hredial.yaml -g 0 -s 1 -p -e 10
python run_crslab.py --config /mhim/htgredial.yaml -g 0 -s 1 -p -e 10
```

值得注意的是原论文中提供的 redial 数据集的下载链接已不存在，于是我自行在网上进行搜索下载了相应的数据集放置在对应位置，并修改数据加载模块的代码使得当数据存在时跳过下载步骤。以上工作均是在完全保留原文参数的设置下进行的实验，为了验证部分参数对模型性能的影响，我对训练的轮次 *epoch*、随机种子 *seed* 和扩展策略 *extension_strategy* 做出修改并观察实验结果的变化。

4.2 模型改进

我主要对模型进行了以下两方面的改进，分别是引入电影评论外部数据集进行数据增强和改变扩展项的扩展条件。

4.2.1 数据增强

在 redial 数据集上引入外部数据集。redial 是英文对话推荐数据集。**RevCore** [26] 这篇论文中提供了电影评论数据集，该数据集爬取自IMDB, 它是最受欢迎和最权威的电影数据库之一。每条评论都可以根据相应的电影以及 IMDB 提供的评分进行查询。作者选取了每部电影有用性评分最高的 30 条评论，以保证收集到的评论质量较高。本次复现通过提取具有相同情感极性的评论加入到本文现有的数据集中来进行数据增强。

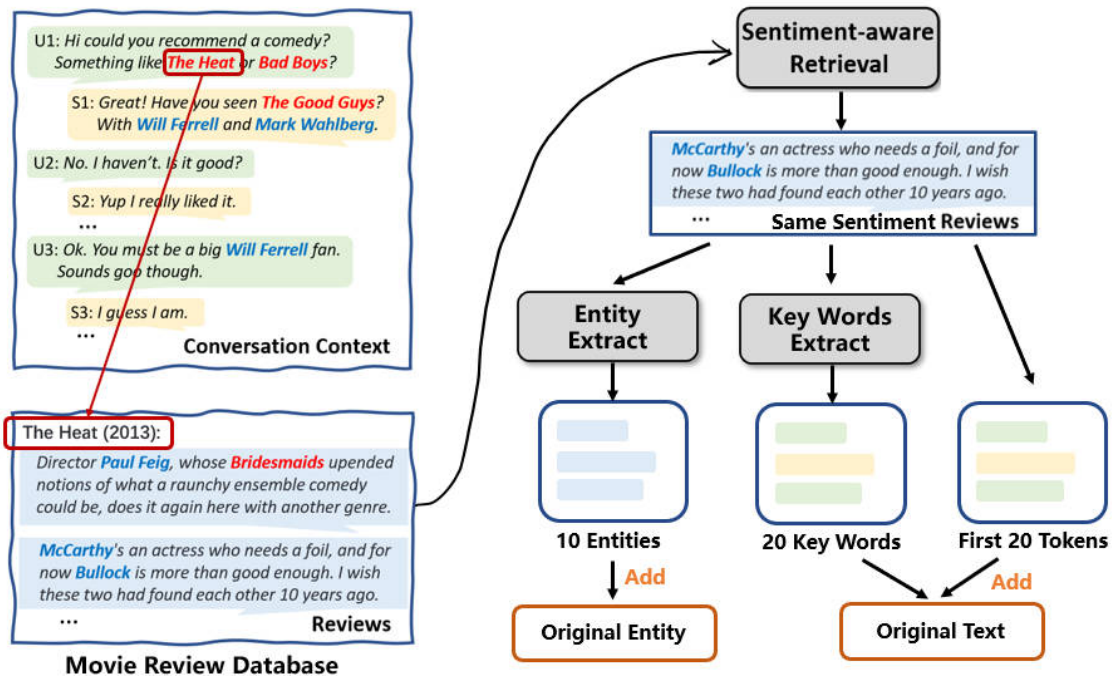

图 3. 改进模型框架

改进主要是在进行超边扩展和构建超图之前进行的，该部分改进后的模型总体如图3所示。

具体地，我通过遍历原始数据集中的每个对话，将每个对话中提到的电影项目提取出来再对其进行遍历，若某个电影项目在引入的电影评论数据集中有出现，则提取该电影的评论列表并找到与原对话情感极性相同的评论，得到评论列表后进行实体抽取和 tokens 抽取。

对于**实体抽取**，我将评论中出现的实体转换为 index，取前 10 个加入到原始对话的 entity 列表中。

对于 **tokens 抽取**，我采取了两种操作，第一是按原文 [26] 的方法直接截断前 20 个 tokens 加入到原数据集的 text tokens 中增加对话丰富度；第二是考虑到直接截断前 20 个 tokens 并不能反映评论的语义信息，因此我选择提取得到的评论中的 20 个关键词，将关键词加入到原数据集的 text tokens 中。将两种方法得到的 tokens 分别加入到原始对话的 text 集合中，进行两个实验验证加入外部 tokens 的有效性。

总体增加的代码的关键部分如下所示：

```
def _merge_conv_data(self, dialog, user_id, conv_id):
    for utt in dialog:
        # ...数据加载代码...
        # 1. 判断对话中是否涉及到movie
        if len(movie_ids)>0:
            # 2. 是的话就将text中的word拼接起来变为text
            sentence = ' '.join(utt["text"])
            # 3. 找到相同感情倾向的评论
            movie_comment_text = self._find_same_sentiment_and_item_text(
                sentence, movie_ids)
            # 4. 简化movie_comment_text内容
```

```
            # 方法1: 提取20个关键词: 使用函数extract_keywords()得到关键词
            summary_of_movie_review = self._extract_keywords(
                movie_comment_text,20)
            # 方法2: 截取前20个token
            summary_of_movie_review = movie_comment_text.split()[:20]
            # 将summary_of_movie_review转为id加入text_token_ids中
            movie_comment_text_id = [self.tok2ind.get(word, self.tok2ind[
                '__unk__']) for word in summary_of_movie_review]
            # 5. 提取movie_comment_text中的entity, 加入到entity_ids中
            entitiesID_in_review = self._find_movie_review_entity(
                movie_comment_text)
        else:
            movie_comment_text_id = []
            entitiesID_in_review = []
        # ...将得到的movie_comment_text_id和entitiesID_in_review加入到原
            数据集...
    return augmented_convs
```

Listing 1: Add Movie Review in Redial

其中函数 _find_same_sentiment_and_item_text() 是为了**找到与当前对话情感极性相同的评论**，经过测试后我发现将极性设置为 3 个比设置为 5 个更好（尽管参考论文采用的是 5 个极性），因为当设置为 5 个极性时头尾两个极性的评论极少。

```
def _find_same_sentiment_and_item_text(self, text, movie_ids):
    # ...加载数据代码...
    current_text_sentiment = self._sentiment_of_text(text)
    for mov_id in movie_ids:
        # text中提到的电影在评论中出现了
        if mov_id in review_movie_token2id:
            review_list = review_items[review_movie_token2id.index(mov_id
                )][1]
            for i in review_list:
                # 情感极性一致
                if self._sentiment_of_text(i) == current_text_sentiment:
                    extended_sentiment_text += i
    return extended_sentiment_text
```

Listing 2: find same setiment reviews

情感极性函数 _sentiment_of_text() 简单采用 TextBlob 库得到：

```
    def _sentiment_of_text(self, text):
        analysis = TextBlob(text)
        # 获取文本的极性
        sentiment_polarity = analysis.sentiment.polarity
        if sentiment_polarity > 0:
```

```
            sentiment = 1
        elif sentiment_polarity < 0:
            sentiment = -1
        else:
            sentiment = 0
        return sentiment
```

Listing 3: sentiment of text

关键词提取算法采用的是经典的 TF-IDF 算法：

```
def _extract_keywords(self, text, num_keywords=20):
    # ...其他数据清洗操作...
    # 使用TF-IDF计算关键词权重
    tfidf_scores = {}
    for word in set(words):
        tf = word_freq[word] / len(words)
        idf = text_collection.idf(word)
        tfidf_scores[word] = tf * idf
    # 根据TF-IDF权重排序关键词
    sorted_keywords = sorted(tfidf_scores.items(), key=lambda x: x[1],
       reverse=True)
    # 提取前N个关键词
    top_keywords = [keyword for keyword, score in sorted_keywords[:
       num_keywords]]
    return top_keywords
```

Listing 4: extract keywords

对于寻找评论中出现的 entity 的方法，单词（或短语）与实体的映射关系对于评论实体的抽取相当重要，比如在训练数据中，若用户在对话中提到 “musical film” 这一短语，则该对话中将会附带一个对应该实体的相关链接（”<http://dbpedia.org/resource/Musical_film>”），但是由于论文没有提供该关系映射字典，因此我对原文的训练数据集（train_data.json）做出进一步的处理：

1）对每个对话内容（text）进行命名实体识别得到实体列表；2）提取实体链接的后半部分（”http://dbpedia.org/resource/” 之后）得到词组；3）遍历实体列表，若列表中的实体在词组中出现，则得到一对映射关系。如图所示是采用该方法提取到的字典，共 1448 条。

图 4. 实体字典

得到该字典后，对情感极性相同的评论进行扫描，若评论中出现字典的键，则将对应的实体链接通过论文中的”entity2id.json” 字典转为 index 加入到当前对话的 entity 中，以丰富 entity 的内容。

以下代码即为抽取评论中出现的实体的总函数：

```
    def _find_movie_review_entity(self, movie_comment_text):
        with open('/data/dataset/hredial/nltk/words2entityAdd.json', 'r')
            as file:
            entity_dictionary = json.load(file)
        entitiesADD_list_in_movie_review = self._find_entity_in_sentence(
           movie_comment_text, entity_dictionary)
        entitiesID_in_review = [self.entity2id[entityAdd] for entityAdd
           in entitiesADD_list_in_movie_review if entityAdd in self.
           entity2id]
        return entitiesID_in_review
```

Listing 5: start func of extracting entity

其中 “words2entityAdd.json” 即为提取后的字典，*__find_entity_in_sentence* 函数用于提取文本中的实体并得到相应的实体链接。

```
    def _find_entity_in_sentence(self, sentence, entity_dictionary):
    entityADD_list = []
    for key in entity_dictionary:
        if key in sentence:
            entityADD_list.append(entity_dictionary[key])
    return entityADD_list
```

Listing 6: extract entity

在 htgredial 数据集上引入外部数据集。 htgredial 是中文对话推荐数据集，为了找到合适的电影评论数据集，我从豆瓣电影数据集地址下载了电影数据集 Moviedata-10M，该数据集采集于豆瓣电影，电影与明星数据收集于 2019 年 8 月上旬，影评数据 (用户、评分、评论) 收集于 2019 年 9 月初，共 945 万数据，其中包含 14 万部电影，7 万演员，63 万用户，416 万条电影评分，442 万条影评。

复现工作主要采用数据集中的 *movies.csv* 和 *comments.csv*，其中 *movies.csv* 包含电影 id"MOVIE_ID"、电影名称"NAME" 和电影描述"STORYLINE" 等字段，*comments.csv* 包含电影 id"MOVIE_ID" 和电影评论"CONTENT" 等字段。

提取评论过程与在 redial 数据集上大致相同，差异在于我在其中添加了评论的前 20 个 tokens 之外，由于观察到数据集中包括每部电影的描述，于是我还尝试将电影描述加入到上下文中来丰富语义信息，值得注意的是我一共做了三个实验，分别是引入评论、引入描述、引入评论和描述。

同时，由于中文数据集中 entity 的提取更为复杂，加上在 redial 数据集上引入外部数据集中的 entity 并没有带来性能的提升，因此我没有在 htgredia 数据集上做引入 entity 的实验。总体增加的代码的关键部分如下所示：

```
def _merge_conv_data(self, dialog, user_id, conv_id):
    for utt in dialog:
        # . . . 数 据 加 载 代 码 . . .
        # 1. 将对话中的所有单词拼接成一句话
        sentence = ' '.join(utt["text"])
        # 2. 在相应的电影评论中获得前20个tokens
        review_20words = self._get_first_20tokens_of_review_(sentence,
           movie_ids)
        storyline_20words = self._get_first20_tokens_of_storyline(
           movie_ids)
        review_20tokens = [self.tok2ind.get(word, self.tok2ind['__unk__'
           ]) for word in review_20words]
        storyline_20tokens = [self.tok2ind.get(word, self.tok2ind['
           __unk__']) for word in storyline_20words]

        if utt["role"] == last_role:
            augmented_convs[-1]["text"] += text_token_ids +
               review_20tokens + storyline_20tokens
            augmented_convs[-1]["movie"] += movie_ids
            augmented_convs[-1]["entity"] += entity_ids
```

Listing 7: Add Movie Review in Redial

4.2.2 改变扩展项的扩展条件

为了提高推荐任务的准确率，我转而尝试改变超边的扩展条件。原文中提到，由于对话推荐数据集的稀缺性，作者考虑引入了一个原始的大规模的 KG 来作为增强知识。具体来说，

对于会话中的每一个项目（item），在引入的 KG 中找到该项目及其 n 跳邻居，并连接一个超边。代码中作者设置了扩展 1 跳邻居，我将其修改为 2 跳来观察实验结果是否有所提高。

以下是代码的关键部分。

```
def _build_adjacent_matrix(self):
    # ......
    # 【修改部分】遍历一定次数，即选择几跳邻居
    for _ in range(1):
        buffer = set()  # 用于存储当前跳的实体集合
        for source in last_hop:
            # 将与上一跳实体相连的实体添加到邻接集合中
            adj[entity].update(graph[source])
            buffer.update(graph[source])
        last_hop = buffer  # 更新上一跳的实体集合
```

Listing 8: n hop neighborhood

另外，在 3.3.4 相似对话的超边扩展这一节中，作者提出为了进一步减轻用户历史对话的稀缺性，他们提出根据共同项目进行超边扩展。具体来说，他们从对话中提取项目并构建超边，形成一个超边集合。然后根据当前对话的项目和超边集合中共同项目比例，选择超过一定阈值的超边进行扩展。

论文中设置的是所有会话中与目标超边中的项目的项目重叠数排名前 50 且重叠率高于 40% 的会话，将这些会话超边中包括的项目作为扩展项加入到目标超边中。

```
def _search_extended_items(self, conv_id, context_items):
    # ......
    #【修改部分】排名前50且比率大于40%的会话
    for i in range(50):
        if conv_and_ratio[i][1] < 0.40:
            break
        extended_items.append(self.conv2items[conv_and_ratio[i][0]])

    return extended_items
```

Listing 9: search extended items

4.3 实验环境搭建

创建虚拟环境。实验需要在 python3.8 的环境下运行，创建成功后下载所需的库和对应的 pytorch。

```
conda create -n py38 python=3.8 # 创建python3.8的虚拟环境
conda activate py38 # 激活环境
conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch
    -c nvidia # 下载torch
```

服务器。本次实验提供的服务器及其参数如表3所示。

服务器	服务器参数
201	CPU: Intel(R) Xeon(R) Platinum 8383C*2 （共 80 核 160 线程） GPU: NVIDIA A100-PCIE-40GB * 4 内存：256GB
184	CPU: Intel(R) Xeon(R) Platinum 8358P * 2 （共 64 核 128 线程） GPU: NVIDIA GeForce RTX 4090 24GB * 4 内存：125GB

表 3. 实验使用的服务器参数

	ReDial						TG-ReDial					
	R@10	R@50	M@10	M@50	N@10	N@50	R@10	R@50	M@10	M@50	N@10	N@50
MHIM	0.1966	0.3832	**0.0742**	**0.0830**	**0.1027**	**0.1440**	0.0300	0.0783	**0.0108**	**0.0129**	**0.0152**	0.0256
复现	**0.2001**	**0.3865**	0.0713	0.0802	0.1013	0.1426	**0.3039**	**0.0825**	0.0105	0.0128	0.0151	**0.0263**

表 4. 在推荐任务中原论文与复现结果的比较

4.4 创新点

本次复现工作在两个方面上进行创新：

- **改变超图扩展的扩展条件**。通过改变扩展超图的扩展条件，包括具有相似偏好的用户之间的超边扩展和在外部 KG 上的邻居跳数，来对扩展条件做出一定的约束。两个数据集上的实验结果表明改变扩展方法有助于模型性能的提高，特别是在项目推荐任务上。
- **数据增强**。通过引入外部数据集并在一些约束条件下将与本任务本数据集有关的数据添加到模型中，以此来进一步缓解数据稀疏性的问题。在两个数据集上的实验结果表明数据增强确实有助于模型性能的提高，特别是在对话生成任务上。

5 实验结果分析

5.1 论文复现

首先我按照和原论文完全相同的设置进行实验，实验结果如表4和5所示，其中”MHIM”一行是原论文中的实验结果，” 复现” 是复现后的结果。可以看到实验结果与原论文结果有一定的波动和起伏。

同时为了验证部分参数对模型性能的影响，我对训练的轮次 *epoch*、随机种子 *seed* 和扩展策略 *extension_strategy* 做出修改并观察实验结果的变化。实验设置如表6所示。相应的实验结果如表8和7所示。注意，在后续实验中都是与我复现的结果进行比较，若实验结果相比

	ReDial			TG-ReDial		
	Dist_2	Dist_3	Dist_4	Dist_2	Dist_3	Dist_4
MHIM	0.3278	0.6204	0.9629	**1.1100**	**2.3520**	3.8200
复现	**0.4012**	**0.7532**	**1.1620**	1.1020	2.3430	**3.8230**

表 5. 在对话任务中原论文与复现结果的比较

复现结果有提升，则将该实验结果（单元格）高亮为蓝色，同时对于每个复现或改进部分将具有最佳结果的实验结果加粗显示。

从表8可以看出对于推荐任务，这些参数的更改在 redial 数据集上的多个指标有一定的提升，且将随机种子设置为 2022 时，无论是在 redial 数据集还是在 htgredial 数据集上性能提升最高。对于对话生成任务，这些参数的更改在 htgredial 数据集上的所有指标都得到了提升，且在将扩展策略 *extension_strategy* 修改为 Random 时性能提升最高，同时，将预训练轮次提升到 20 轮对 redial 数据集性能提升帮助最大。

实验	实验设置
原论文	seed1、epoch100+5、预训练 epoch11、extension_strategy: Adaptive
实验 1	预训练 epoch20
实验 2	epoch6+6
实验 3	seed8
实验 4	seed2022
实验 5	extension_strategy: Random

表 6. 论文复现实验设置

	ReDial			TG-ReDial		
	Dist-2	Dist-3	Dist-4	Dist-2	Dist-3	Dist-4
MHIM	0.3278	0.6204	0.9629	1.11	2.352	3.82
复现	0.4012	0.7532	1.162	1.102	2.343	3.823
实验 1	**0.4419**	**0.8448**	**1.325**	1.339	2.801	4.547
实验 2	0.3818	0.6656	0.9495	1.589	3.211	5.022
实验 3	0.3478	0.6858	1.107	1.249	2.651	4.298
实验 4	0.3542	0.6736	1.03	1.367	2.833	4.486
实验 5	0.2644	0.4996	0.7902	**1.666**	**3.547**	**5.742**

表 7. 改变部分参数在对话任务上的实验结果

5.2 数据增强

5.2.1 在 redial 数据集上引入外部数据集

根据 4.2.2 节的介绍，我在 redail 这个英文对话推荐数据集上引入了外部数据集，实验设置如表9所示。在推荐任务和对话生成任务上的实验结果分别如表10和表11所示。

从实验结果可以看出引入外部数据集对推荐任务没有任何帮助，特别是加入 entity 就是为了提高推荐任务的性能，但是引入后反而降低了实验结果，我猜想可能的原因是我提取的 entity 与单词映射有问题。按照我在 4.2.2 节提到了提取 entity 的做法，一个单词可能在 entity 的后缀部分出现不止一次，也就是说一个 entity 可能对应多个单词，但是我提取的单词和 entity 是一对一的，这会造成模型判断的歧义，混淆了模型的 entity 选择导致性能的下降。但是原论文中没有提到单词和 entity 如何进行映射，也没有相应的字典，因此无法对这一问题做出进一步的解决。

	ReDial						TG-ReDial					
	R@10	R@50	M@10	M@50	N@10	N@50	R@10	R@50	M@10	M@50	N@10	N@50
MHIM	0.1966	0.3832	0.0742	0.083	0.1027	0.144	0.03	0.0783	0.0108	0.0129	0.0152	0.0256
复现	0.2001	0.3865	0.0713	0.0802	0.1013	0.1426	0.03039	**0.0825**	0.01054	0.01277	0.01513	0.02629
实验 1	0.1982	0.3844	0.07204	0.08108	0.1014	0.1428	0.3039	**0.0825**	0.1053	0.1277	0.01513	0.02629
实验 2	0.1964	0.3817	0.07619	0.08506	0.1043	0.1454	0.02939	0.07415	0.009335	0.01121	0.01394	0.02346
实验 3	0.1936	0.3863	0.0773	0.0864	0.1046	0.1471	0.02939	0.07348	0.01202	0.01404	0.01604	0.02567
实验 4	**0.2018**	**0.3883**	**0.07942**	**0.08828**	**0.108**	**0.1493**	**0.03373**	0.07816	**0.01216**	**0.01403**	**0.01713**	**0.02659**
实验 5	0.201	0.3789	0.07601	0.08425	0.1052	0.1443	0.3039	**0.0825**	0.01054	0.01277	0.01513	0.02629

表 8. 改变部分参数在推荐任务上的实验结果

实验	实验设置
实验 10	引入评论的 20 个 keywords
实验 11	引入评论的前 20 个 tokens
实验 12	引入评论的 20 个 entity
实验 13	引入评论的 10 个 entity
实验 14	epoch6+6、前 10 且超 60%、keywords

表 9. 在 redial 上进行数据增强实验设置

	ReDial					
	R@10	R@50	M@10	M@50	N@10	N@50
MHIM	0.1966	0.3832	0.0742	0.083	0.1027	0.144
复现	**0.2001**	**0.3865**	0.0713	0.0802	0.1013	0.1426
实验 10	0.2001	0.3865	0.07127	0.08022	0.1013	0.1426
实验 11	0.2001	0.3865	0.07127	0.08022	0.1013	0.1426
实验 12	0.1934	0.3808	0.07016	0.07912	0.09893	0.1405
实验 13	0.1968	0.3844	0.07057	0.07944	0.09999	0.1414
实验 14	0.1972	0.3833	**0.07603**	**0.08488**	**0.1044**	**0.1456**

表 10. 在推荐任务上的实验结果

	ReDial		
	Dist-2	Dist-3	Dist-4
MHIM	0.3278	0.6204	0.9629
复现	0.4012	0.7532	1.162
实验 10	0.4158	0.7927	1.241
实验 11	0.5349	1.052	1.648
实验 12	0.319	0.589	0.8867
实验 13	0.5059	0.9839	1.543
实验 14	**0.6138**	**1.14**	**1.724**

表 11. 在对话任务上的实验结果

5.2.2 在 htgredial 数据集上引入外部数据集

根据 4.2.2 节的介绍，我在 htgredail 这个中文对话推荐数据集上引入了电影评论外部数据集，实验设置如表12所示。需要注意的是，由于在 redial 数据集上引入外部数据集中的 entity 并没有带来性能的提升，并且中文数据集中的 entity 更难提取，因此在 htgredial 数据集上我没有进行 entity 引入实验。引入电影评论后在推荐任务和对话生成任务上的实验结果分别如表13和表14所示。

可以看到引入电影评论后对对话生成任务带来了不同程度的性能提升。同时我观察到外部电影评论数据集中的 *movies.csv* 具有 “STORYLINE” 字段，即电影描述，考虑到电影描述和电影评论都是电影的描述性信息，因此我也对电影描述单独进行引入，之后同时引入电影评论和电影描述进行实验，实验设置和实验结果如表15和表16、17所示。但令人遗憾的是引入电影描述并没有提高对话生成任务的性能。

实验	实验设置
实验 15	引入评论的前 20 个 tokens
实验 16	引入评论的前 20 个 tokens、seed2022

表 12. 在 redial 上进行数据增强实验设置

	TG-ReDial					
	R@10	R@50	M@10	M@50	N@10	N@50
MHIM	0.1966	0.3832	0.0742	0.083	0.1027	0.144
复现	0.03039	**0.0825**	0.01054	0.01277	0.01513	0.02629
实验 15	0.03039	0.0825	0.01053	0.01277	0.01513	0.02629
实验 16	**0.0344**	0.07849	**0.01221**	**0.01402**	**0.0173**	**0.02663**

表 13. 在推荐任务上的实验结果

	TG-ReDial		
	Dist-2	Dist-3	Dist-4
MHIM	1.11	2.352	3.82
复现	1.102	2.343	3.823
实验 15	1.124	2.347	3.774
实验 16	**1.39**	**2.823**	**4.443**

表 14. 在对话任务上的实验结果

实验	实验设置
实验 17	引入电影描述
实验 18	引入电影评论 + 电影描述

表 15. htgredial 引入电影描述实验设置

5.3 改变超边扩展的条件

根据 4.2.1 节，我对超边扩展的条件进行了两个方面的修改，实验设置如表18所示。由于原文在两个数据集上的设置不同，因此我分别进行了修改。在推荐任务和对话生成任务上的实验结果分别如表20和表19所示。

从实验结果可以看出在推荐任务上，改变扩展项的扩展条件在 ReDial 和 htgredia 数据集上均有不同程度的贡献，在这里注意到在 htgredial 数据集上实验 6 和实验 9 没有结果，这是因为在 htgredial 数据集对应的代码上将邻居跳数改为 2 跳后，可能由于数据量太大的原因多次在运行过程中被终止。在对话生成任务上，可以看到改变扩展条件完全没有帮助，而在 htgredial 数据集上在所有指标上性能都得到了提升，且将扩展条件改为 “前 20 且超 60%” 时对 TG-ReDial 数据集的性能提升贡献最大。

6 总结与展望

对话式推荐系统（CRS）是一个与用户多轮对话的交互系统，其目标是为用户提供高质量的推荐以满足用户的即时信息需求。作者主要提出了两个**创新点**：1）使用超图来表示历史会话中的复杂语义关系；2）建立了两个粒度的超图获取不同层面的用户兴趣。

本次复现工作首先对论文内容进行仔细研读之后按照作者的原始设置进行代码复现，并探讨了论文中的参数——训练的轮次 *epoch*、随机种子 *seed* 和扩展策略 *extension_strategy* 是否会对实验结果造成影响。之后我对模型在两个方面做出了改进，分别是改变超边扩展的条件和引入外部电影评论数据集进行数据增强。改变超边扩展的条件为对话生成任务带来了一定程度的性能提升，并且在 redial 数据集的推荐任务上的所有指标的性能都得到了提高。引入外部电影评论数据集进行数据增强虽然不能提高推荐任务的性能，但在对话生成任务上效果不错，**大部分都得到了性能提升**。

复现工作中有一个**明显的不足**是引入外部数据集进行实体丰富的部分，由于原数据集没有提供相应的词语/短语到实体的映射关系，因此获得单词到实体的映射关系的方法可能不太妥当导致生成的字典质量不高，这也造成了实验结果的下降。

	TG-ReDial					
	R@10	R@50	M@10	M@50	N@10	N@50
MHIM	0.03	0.0783	0.0108	0.0129	0.0152	0.0256
复现	0.03039	0.0825	0.01054	0.01277	0.01513	0.02629
实验 17	0.03039	0.0825	0.01053	0.01277	0.01513	0.02629
实验 18	0.03039	0.0825	0.01053	0.01277	0.01513	0.02629

表 16. 在推荐任务上的实验结果

	TG-ReDial		
	Dist-2	Dist-3	Dist-4
MHIM	1.11	2.352	3.82
复现	1.102	2.343	3.823
实验 17	0.9805	2.151	3.591
实验 18	1.012	2.114	3.424

表 17. 在对话任务上的实验结果

实验	实验设置	
	英文数据集 REDIAL	中文数据集 TG-REDIAL
MHIM	一跳邻居、前 50 且超 40%	一跳邻居、前 50 且超 5%
实验 6	扩展两跳邻居	扩展两跳邻居
实验 7	前 20 且超 60%	前 20 且超 5%
实验 8	前 50 且超 60%	前 40 且超 5%
实验 9	扩展两跳邻居、前 20 且超 60%	扩展两跳邻居、前 20 且超 5%

表 18. 改变超边扩展条件实验设置

未来可在以下四个方面进行改进：

1）**改变引入实体的方法**。如找到与当前对话中出现的电影最相关的电影，相关性前 k 个加入到原有的 entity 中丰富实体信息。

2）**用知识增强代替数据增强**。利用知识图谱捕获实体间的关系，引入先验知识，使用更少的数据量获得更多的知识，缓解数据稀缺问题。

3）使用**网格搜索或随机搜索**等方法确定最优超参数。

4）**改变 KG 编码器的预训练方法**。比如链接预测和实体预测等等来观察是否有性能上的提高。

	ReDial			TG-ReDial		
	Dist-2	Dist-3	Dist-4	Dist-2	Dist-3	Dist-4
MHIM	0.3278	0.6204	0.9629	1.11	2.352	3.82
复现	**0.4012**	**0.7532**	**1.162**	1.102	2.343	3.823
实验 6	0.343	0.6336	0.9635	-	-	-
实验 7	0.3933	0.7412	1.139	**1.768**	**3.561**	**5.486**
实验 8	0.3368	0.6575	1.043	1.572	3.227	5.034
实验 9	0.3134	0.5935	0.9142	-	-	-

表 19. 改变扩展条件在对话任务上的实验结果

	ReDial						TG-ReDial					
	R@10	R@50	M@10	M@50	N@10	N@50	R@10	R@50	M@10	M@50	N@10	N@50
MHIM	0.1966	0.3832	0.0742	0.083	0.1027	0.144	0.03	0.0783	0.0108	0.0129	0.0152	0.0256
复现	0.2001	0.3865	0.0713	0.0802	0.1013	0.1426	0.03039	0.0825	0.01054	0.01277	0.01513	0.02629
实验 6	**0.2035**	0.3865	0.07314	0.08184	0.1034	0.1439	-	-	-	-	-	-
实验 7	0.2012	0.3881	0.07903	**0.08787**	0.1076	**0.1489**	**0.03073**	0.07582	**0.01256**	**0.01458**	**0.01686**	**0.02663**
实验 8	0.2012	0.3881	0.07903	**0.08787**	0.1076	**0.1489**	0.02939	0.07348	0.01201	0.01404	0.01604	0.02567
实验 9	0.2033	**0.3886**	**0.0792**	0.08786	**0.1081**	**0.1489**	-	-	-	-	-	-

表 20. 改变扩展条件在推荐任务上的实验结果

参考文献

[1] Song Bai, Feihu Zhang, and Philip H.S. Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110:107637, February 2021.

[2] Alain Bretto. *Hypergraph Theory: An Introduction*. Springer International Publishing, 2013.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[4] Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang, and Jie Tang. Towards knowledge-based recommender dialog system. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

[6] Yang Deng, Yaliang Li, Fei Sun, Bolin Ding, and Wai Lam. Unified conversational recommendation policy learning via graph-based reinforcement learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ' 21. ACM, July 2021.

[7] Yang Deng, Wenxuan Zhang, Weiwen Xu, Wenqiang Lei, Tat-Seng Chua, and Wai Lam. A unified multi-task learning framework for multi-goal conversational recommender systems. *ACM Transactions on Information Systems*, 41(3):1–25, February 2023.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North*. Association for Computational Linguistics, 2019.

[9] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3558–3565, July 2019.

[10] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry, 2017.

[11] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2017.

[12] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollar, and Ross Girshick. Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022.

[13] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks, 2015.

[14] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks, 2019.

[15] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD ' 20. ACM, August 2020.

[16] Shuyi Ji, Yifan Feng, Rongrong Ji, Xibin Zhao, Wanwan Tang, and Yue Gao. Dual channel hypergraph collaborative filtering. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD ' 20. ACM, August 2020.

[17] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2016.

[18] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. Dbpedia –a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[19] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM ' 20. ACM, January 2020.

[20] Wenqiang Lei, Gangyi Zhang, Xiangnan He, Yisong Miao, Xiang Wang, Liang Chen, and Tat-Seng Chua. Interactive path reasoning on graph for conversational recommendation. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[21] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM ' 17. ACM, November 2017.

[22] Raymond Li, Samira Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. Towards deep conversational recommendations, 2018.

[23] Shuokai Li, Ruobing Xie, Yongchun Zhu, Xiang Ao, Fuzhen Zhuang, and Qing He. User-centric conversational recommendation with multi-aspect user modeling. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ' 22. ACM, July 2022.

[24] Zujie Liang, Huang Hu, Can Xu, Jian Miao, Yingying He, Yining Chen, Xiubo Geng, Fan Liang, and Daxin Jiang. Learning neural templates for recommender dialogue system. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021.

[25] Lizi Liao, Ryuichi Takanobu, Yunshan Ma, Xun Yang, Minlie Huang, and Tat-Seng Chua. Deep conversational recommender in travel, 2019.

[26] Yu Lu, Junwei Bao, Yan Song, Zichen Ma, Shuguang Cui, Youzheng Wu, and Xiaodong He. Revcore: Review-augmented conversational recommendation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics, 2021.

[27] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD ' 20. ACM, August 2020.

[28] Xuhui Ren, Hongzhi Yin, Tong Chen, Hao Wang, Zi-Liang Huang, and Kai Zheng. Learning to ask appropriate questions in conversational recommendation. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.

[29] Rajdeep Sarkar, Koustava Goswami, Mihael Arcan, and John P. McCrae. Suggest me a movie for tonight: Leveraging knowledge graphs for conversational recommendation. In *Proceedings of the 28th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, 2020.

[30] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. *Modeling Relational Data with Graph Convolutional Networks*, page 593–607. Springer International Publishing, 2018.

[31] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM ' 19. ACM, January 2019.

[32] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM' 15. ACM, October 2015.

[33] Yueming Sun and Yi Zhang. Conversational recommender system. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[35] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2017.

[36] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax, 2018.

[37] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. Next-item recommendation with sequential hypergraphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ' 20. ACM, July 2020.

[38] Ting-Chun Wang, Shang-Yu Su, and Yun-Nung Chen. Barcor: Towards a unified framework for conversational recommendation systems, 2022.

[39] Zihan Wang, Gang Wu, and Yan Wang. Effectively using long and short sessions for multi-session-based recommendations. *arXiv preprint arXiv:2205.04366*, 2022.

[40] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):346–353, July 2019.

[41] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy Huang. Hypergraph contrastive collaborative filtering. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ' 22. ACM, July 2022.

[42] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. Self-supervised hypergraph convolutional networks for session-based recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4503–4511, May 2021.

[43] Bo Xu, Yong Xu, Jiaqing Liang, Chenhao Xie, Bin Liang, Wanyun Cui, and Yanghua Xiao. *CN-DBpedia: A Never-Ending Chinese Knowledge Extraction System*, page 428–438. Springer International Publishing, 2017.

[44] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. Graph contextualized self-attention network for session-based recommendation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, IJCAI-2019. International Joint Conferences on Artificial Intelligence Organization, August 2019.

[45] Kerui Xu, Jingxuan Yang, Jun Xu, Sheng Gao, Jun Guo, and Ji rong Wen. Adapting user preference to online feedback in multi-round conversational recommendation. *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021.

[46] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Pratim Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. In *Neural Information Processing Systems*, 2018.

[47] Bowen Yang, Cong Han, Yu Li, Lei Zuo, and Zhou Yu. Improving conversational recommendation systems' quality with context-aware item meta-information. In *Findings of the Association for Computational Linguistics: NAACL 2022*. Association for Computational Linguistics, 2022.

[48] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations, 2020.

[49] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. Self-supervised multi-channel hypergraph convolutional network for social recommendation. In *Proceedings of the Web Conference 2021*, WWW ' 21. ACM, April 2021.

[50] Tong Zhang, Yong Liu, Boyang Li, Peixiang Zhong, Chen Zhang, Hao Wang, and Chunyan Miao. Toward knowledge-enriched conversational recommendation systems. In *Proceedings of the 4th Workshop on NLP for Conversational AI*. Association for Computational Linguistics, 2022.

[51] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W. Bruce Croft. Towards conversational search and recommendation: System ask, user respond. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018.

[52] Kun Zhou, Wayne Xin Zhao, Shuqing Bian, Yuanhang Zhou, Ji-Rong Wen, and Jingsong Yu. Improving conversational recommender systems via knowledge graph based semantic fusion. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD ' 20. ACM, August 2020.

[53] Yuanhang Zhou, Kun Zhou, Wayne Xin Zhao, Cheng Wang, Peng Jiang, and He Hu. C^2-crs: Coarse-to-fine contrastive learning for conversational recommender system. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, WSDM ' 22. ACM, February 2022.