1. **对话**
   1. **End-to-End Neural Pipeline for﻿ Goal-Oriented Dialogue Systems using GPT-2**

用GPT2分别生成state和reponse,并加入了一个判断rsp是否是gold的分类目标函数,效果不如当时的pipeline模型.

拼接了所有历史的user和sys utterance, 但state没用GPT2生成, state输入GPT2时用的是gold

* 1. **SimpleTOD**

用GPT2分别生成state和reponse, 效果就比上一篇好, 超过了当时的pipeline模型.

state也是GPT2生成, 但目标函数是正常GPT2的目标函数, state输入GPT2时用的是gold

* 1. **SOLOIST**

用GPT2同时生成state和reponse,也加上了对比学习,目标函数为state是否正确,response是否正确,是否靠近正样本并远离负样本

state也是GPT2生成,目标函数是state和rsp都生成完后,再去做计算的, state输入GPT2时用的是gold

* 1. **UBAR**

使用了整个session的user/sys uterance, 包括state和act, 并且生成什么就输入什么到GPT2(state和act不用gold). 消融实验证明这两个优化都很有效.

* 1. **ConceptFlow**

本文把近距离概念和远距离概念区分处理, 因为近距离概念是中心概念central\_graph, 远距离的(2跳)为可考虑的对外延申概念outer\_graph, 本意是为了处理对话的多样性的同时又不偏离主题. 对central\_graph用GNN处理. 对outer\_graph用attention机制以对不同outer concept学习不同权重, 因为有些outer concept延申出去的意义更大些. 结合历史对话,生成当前对话.

1. **对话式推荐**
   1. **DuRecDial@ACL2020**

百度提出的类似DuConv的数据集,目的是训练出一个能通过对话引导对方购买的推荐模型. 不仅对实体跳转的推理路径进行标注,也对对话类型(闲聊/问答/推荐)进行了标注, 但对话语料也是简单和生硬. 基线算法于DuConv类似.

* 1. **SASRec:**

把用户历史行为(以前买过些啥)转化为一个seq2seq问题,预测下一次会买啥.感觉想的有点简单了,下一次会买啥不是简单地由购买历史决定的,但在公开数据集中当时表现还可以. 见:https://3ms.huawei.com/km/blogs/details/9631131

* 1. **TG\_CRS@COLING2020**

提出TG\_ReDial数据集, 目的是训练出一个能通过对话引导对方购买的推荐模型, 但与DuRecDial不同的是,数据集是基于豆瓣真实用户的访问历史构建.但依然依赖对推理路径的标注. 基线模型结合了SASRec和GPT2, 并分三个模块, 可分别训练和使用. 见:https://3ms.huawei.com/km/blogs/details/9631131

* 1. **EAR@WSDN2020**

提出了把对话式推荐分成Estimater, Action, Reflection三大模块. Estimator用于预测用户喜好, Action基于E的预测判断是要继续询问用户喜欢什么还是生成推荐的话语, Relection把用户的反应提供给Estimator进行训练. Action模块只能简单地询问, 而无法生成说服性的闲聊,对话属性弱一些; 而Reflection和Estimator又增加了推荐和RL的味道,但它们单独用于推荐,效果又不好.总的来说只是简单搭了个框架.

1. **对话式QA**
   1. **HAM@CIKM2019**

将对特定主题下的未知信息的探索作为一系列的问答对话来处理, 使用History Attention机制选择对话历史中对当前问答比较有作用的

* 1. **Stay Hungry, Stay Focused @EMNLP2020**

在不断提问探寻信息的任务中, 问正确的问题很关键, 本文认为正确的问题应该是能够挖掘更多信息的问题, 并且以该问题的答案中与问答对话中的词语重复的多少来衡量模型所生成的问题的好坏, 重复得越少, 生成的问题越好

* 1. **Unsupervised Question Decomposition @EMNLP2020**

将一个复杂问题(多跳)分解成多个简单的子问题(单跳),来达到解决复杂问题的目的. 从额外的简单问题数据集,搜索出与复杂问题相似的问题,作为虚拟子问题,然后训练一个从复杂问题转换成多个虚拟问题的seq2seq模型,以达到分解目的.

1. **多跳多文档QA**
   1. **CO-Search**

把下述各种技术(4.3-4.6)集成成一个系统,可作为工程参考.

* 1. **RAG@NIPS2020**

把DPR用到了生成式模型上,在抽取式问答中也取得了较好成绩,但需要很多语料

* 1. **DPR@EMNLP2020**

不像ORQA和REALM那样对BERT进行基于doc retrieve的预训练,而仅把BERT作为encoder,但提出了CL目标函数以及较好的负采样方法,提升了效果.

* 1. **DrKit@ICLR2020**

把REALM的玩法扩展到了多跳QA领域,并且可微分的去训练多跳多文档的QA模型.具体是把多跳建模成不断猜测下一步要retrieve些什么实体相关的文档,retrieve过来后进行判断是否能回答,不能回答再retrieve.

* 1. **REALM@ICML2020**

把doc retrieve过程也加入到BERT的端到端预训练中,提出了适合训练doc retrieve模型得文档拼接并预测mask的任务,并用延迟更新doc embed的方式解决了训练数据量大的问题,同时提出了QA场景如何在REALM上进行fine tuning.

* 1. **ORQA@ACL2019**

提出了QA并不一定要基于一篇文档,而是可以从多文档中retrieve,在从retrieve得文档中查找答案,同时提出了反向完形填空SSL任务使得预训练模型更能为Q找到合适得包含A得文档或文段.

* 1. **HopRetriever@AAAI2021**

基于PathRetriever的代码改进而成,把PathRetriever中基于词共现的召回改成了基于语义的召回,把各个文档按照pagerank的玩法连成一张图,以链接为边,以包含该链接的上下文的embed以及链接到的那个文档的embed作为边特征,用attention机制有监督地计算从一个文档到另一个文档的权重. 最后把所权重高的文档与问题拼起来求答案.

* 1. **PathRetriever <- HGN <- CogQA**

粗略说一下这三个在HotPodQA上的老模型. CogQA是自己在阅读文档的同时构建了一个KG,实体是文档中相关实体的embed,边是一个实体(文档)到另一个实体的跳转(链接)embed,最后在该实时构建的KG上进行推理. HGN则是构建了由实体到句子到段落到文档的层级图(不是树,因为从实体可连接到文档等跨层级连接),基于该层级图进行推理. PathRetriver其实就是用类似HopRetriver的方法(但没有基于语义),把召回的文档拼在一起求答案. 最终PathRetriever的效果比前两者好,个人觉得是因为信息丰富,没有经过人为裁剪,以及BERT的效果的确香

1. **KG+QA和对话**
   1. **KG-BART@AAAI2021**

分两个graph, concept-reason graph对现有实体(Q中给出的实体)建模成图,学习现有实体间关系,用于encoder和decoder. concept-expand graph是每个现有实体基于其邻居延申出来的子图,学习实体自身延申出去的知识,用于在decoder时同时融入现有实体自身的扩展知识,也考虑现有实体间的关系(基于层级多头GAT)

* 1. **NSM+h@WSDM2021**

无需路径标注,用双向BFS找到若干路径,然后teacher网络基于正负样本路径做前向和后向的学习(类似一种约束,使得每一跳的相应实体在前向和后向计算都是分布一致的),以习得每一跳在每个实体上的概率.然后将该概率分布教给student网络. 虽然不用路径标注,但路径需要在KG中,并且每步计算都基于所有实体.

* 1. **OpenDialKG@ACL2019**

认为对话的话题的变化,是在一个KG上不同的实体间的跳转,提出了一个包含了实体跳转路径作为label的对话数据集,并提供了一个RNN+attention实现的baseline,用于基于KG推理的方式来进行对话. 但模型强依赖于数据集中推理路径的标注,同时数据集的语料和话题跳转既简单又生硬..

* 1. **AttnIO@EMNLP2020**

基于OpenDialKG数据集,用attention传递的方法提升了在OpenDialKG上的准确率.

* 1. **KE-Dialog@EMNLP2020**

基于KG用模板方法生成对话语料,用于训练GPT2,使其学习到KG中的知识.

* 1. **DuConv@ACL2019**

百度提出的类似OpenDialKG数据集,目的是训练出一个能引导对方将对话进行到特定话题的模型. 但也需要对推理路径进行标注,对话语料也是简单和生硬. 也提出了基线算法,基于Transformer去预测下一轮对话会涉及的实体,然乎用Seq2Seq+MemoryNetwor生成对话语句.

* 1. **KnowHRL@AAAI2020**

基于DuConv使用RL与KG的方式,由粗到细规划(先规划topic如何跳转,再规划在聊topic时如何让用户感觉更好)推理路径,由于数据集用的是DuConv,因此也依赖于推理路径的标注.

1. **多模态QA**
   1. **LayoutLM**

用于处理文档图片,既把它当作图片,也把它当作文档处理.加入了文字图在整张图上的绝对和相对的坐标信息,既有局部图(文字图)的信息(与token对齐),也有整张图(文档图)的信息(与文章[CLS]对齐),.然后就是加入了各种有利于多模态对齐的SSL任务(用图去猜被遮挡的文字,用文字去猜被遮挡的图之类的).

* 1. **iPerceiver**

加入了音频,speech,subtitle,QA等信息做融合

* 1. **HERO@EMNLP2020**

处理video+txt的.使用层级Transformer,先在local层面把subtitle和video clip对齐与融合,然后把所有得到的video frame embed传入另一个transformer做全局的处理.

* 1. **ERNIE-ViL@AAAI2021**

基于百度ERNIE,本来就是个很强的骨干,然后基于scene graph做各种mask based的预训练任务, 比如mask掉实体, 属性, 关系啥的.

* 1. **OSCAR@ECCV2020**

在UNITER基础上加入region的tag信息, 针对这些tag信息做了基于mask和match的预训练.效果就比UNITER好了.怀疑是不是一个研究分几篇paper发.

* 1. **UNITER@ECCV2020**

基于Transformer.可用于多种img+txt的下游任务的预训练模型. 图片中抽取region embed和txt embed拼在一起(用特殊segment分开???)作为输入.然后新加了与多模态相关的SSL任务: MRM(还原mased region), ITM(判断img与txt是否匹配). 然后还讲了一些加速训练的trick(但貌似都听说过,混合精度,边长btach之类的).然后就是在一堆数据上训练.

1. **IDEA**
   1. **对话QA预测意图后规划推理路径**

基于历史问答,预测当前问题的答案的大概范围, 从KG中找到相关节点, 然后规划推理路径

采用e2e对话模型的思想,concat多轮问答的历史,去预测当前论的问题和答案.

主题是否变化

根据历史生成问题,判断是否与当前问题相差太大,是则主题变化

结合其它方法

生成式意图预测:

根据真实问题生成若干答案,找到能提供信息最多的top n

根据生成的问题生成答案,找top n

抽取生成答案中的实体及其上位词, 从KG中寻找与它们语义相似的节点

基于n跳的约束规划推理路径

与问题实体n跳就能到达答案实体的路径

可加上其它约束

* 1. **第三方KG扩展问题为子图**

现有的kg融入很多都是用实体各有各子图的方式,尽管不同问题实体的子图都来自一个kg且很有可能重合

一个问题对于一个kg只有一个子图,减少冗余,增加有意义的关系连接,加快计算效率

* 1. **基于视频轨与音轨的多模态**

解决的问题:

提出效果更好,能处理更多模态的多模态模型. 最好是基于现有的预训练模型去做.

用于多模态意图识别/对话/QA/搜索/推荐/电视剧描述生成等

企业内可用于会议和培训视频的搜索, 会议助手, 销售助手等

方法:

划分event, 加入caption, 场景图等信息

捕捉caption和场景图变化所包含的信息

除了模态segment,

timestamp作为position embed

较近的一段时间的所有信息输入全连接transf

较远的历史, 先切分event族, 然后embed, 作为一种模态输入

风险:

可能会涉及对视频音频数据的预训练,算力需求大

* 1. **与AutoML结合的动态Transf**

从Transf不同层得到的embed不一样,对于一些简单的问题,层数太多,反而会导致’过度推理’而产生错误结果.

因此可以考虑采用可微分automl的方法,建立一个方法论模型,负责判断什么样的问题应该用那层产生的结果,以得到更好结果的同时减少计算量

* 1. **DeepSpeed+MS**

把deepspeed用在对mindspore的优化上

* 1. **编译器计算图优化**

使用图生成和automl结合的方法优化编译器计算图,进一步提高性能