基于知识图谱的推荐相关论文keypoint

笔记本:	snowcement的笔记本		
创建时间:	2018/10/16 15:12	**更新时间:**	2018/11/7 14:41
作者:	snowcement@126.com		
标签:	知识图谱		

知识图谱相关:

---

1 干货 | 大规模知识图谱的构建、推理及应用

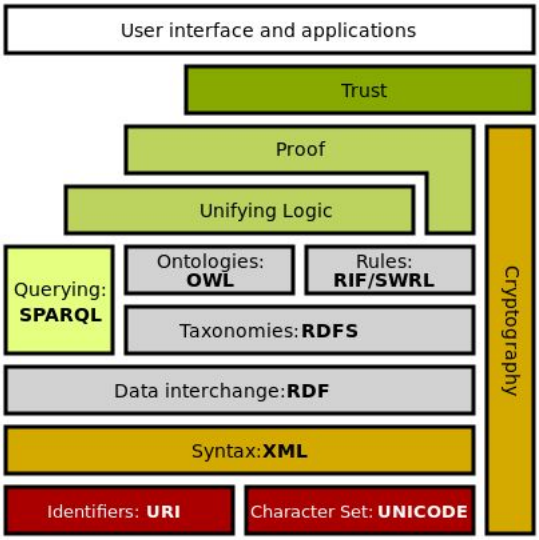
- https://mp.weixin.qq.com/s?__biz=MjM5MDI3MjA5MQ==&mid=2697266451&idx=1&sn=264e01bf70c410ee5cee9b3d95a08a15&chksm=8376fa27b401733198c913c9a2be6a6622394a58864d698bc57e
- 构建：将众多的实体和关系需要从原始数据（可以是结构化也可以是非结构化）中被抽取出来，并以图的方式进行结构化存储
  - 结构化数据：很容易转换为图结构
  - 非结构化数据构建方法：
    - NLP
    - DL:
      1. 可用于抽取AVP(属性-值对)
      2. 端到端的NER:
         - 从一段**非结构化文本**中找出相关实体（triplet中的主词和宾词），并标注出其位置以及类型
         - 是 NLP领域中一些复杂任务（如关系抽取、信息检索等）的基础
         - 实现技术：
           - 早期基于字典和规则的方法
           - 传统ML的方法：
             - NER:**一个序列标注问题**，不同于分类问题，序列标注问题中的预测标签不仅与输入特征有关，还与之前的预测标签有关，也就是预测标签之间存在相互依赖和影响
             - HMM
             - MEMM
             - CRF：条件随机场（Conditional Random Field，CRF）是序列标注的主流模型。它的目标函数不仅考虑输入的状态特征函数，还包含了标签转移特征函数。在训练的时候可以使用SGD学习参数。在预测时，可以使用Vertibi算法求解使目标函数最大化的最优序列
           - DL:
             - BiLSTM-CNN-CRF：主要由Embedding层（词向量、字向量等）、BiLSTM、tanh隐藏层以及CRF层组成（对于中文可以不需要CNN）
             - CNN-CRF
             - RNN-CRF：实验表明**BiLSTM-CRF**可以获得较好的效果，在特征方面，由于秉承了深度学习的优点，所以无需特征工作的铺垫，使用词向量及字向量就可以得到不错的效果
             - Attention机制：
               - BiLSTM-CRF+Attention机制，将原来的字向量和词向量的拼接改进为按权重求和，使用两个隐藏层来学习Attention的权值，这样使得模型可以动态地利用词向量和字向量的信息。同时加入NE种类的特征，并在字向量上使用Attention来学习关注更有效的字符。实验效果优于BiLSTM-CRF的方法
             - 仅需少量标注样本的半监督来进行相应的工作
               - https://www.cnblogs.com/robert-dlut/p/6847401.html
      3. 关系抽取：**一个序列标注问题，采用模型与NER相同**
      4. 关系补全：
         - 通过现有知识图谱来预测实体之间的关系，是对关系抽取的重要补充
         - 传统方法：
           - TransE和TransH: 假设实体和关系处于**相同的语义空间**，把关系作为从实体A到实体B的翻译来建立实体和关系嵌入
           - 一个实体是由多种属性组成的综合体，不同关系关注实体的不同属性，所以仅仅在一个空间内对他们进行建模是不够的
         - TransR：
           - 将实体和关系**投影**到不同的空间中，在实体空间和关系空间构建实体和关系嵌入
           - 特定的关系投影能够使得两个实体在这个关系下真实地靠近彼此，使得不具有此关系的实体彼此远离
      5. 知识融合，
         - 包含以下几部分：
           - 实体对齐
           - 属性对齐
           - 冲突消解
           - 规范化等
         - 对开放域很难，对**特定领域** 可以通过别名举证、领域知识等方法进行对齐和消解，从技术角度来看，这里会涉及较多的逻辑，所以偏传统机器学习方法，甚至利用业务逻辑即可覆盖大部分场景
  - **没有统一的方法**，因为其构建需要一整套知识工程的方法，知识的更新也是不可避免的，所以一定要重视快速迭代和快速产出检验
- 查询：
  - RDF->OWL->SPARQL
  - postgresql
- 存储：
  - 选**关系数据库**还是**NoSQL 数据库**（内存数据库、图数据库）？要不要用**内存数据库**（e.g.redis）？要不要用**图数据库**（Neo4J、graphsql、sparkgraphx（包含图计算引擎）、OrientDB、基于hbase的Titan、BlazeGraph等）？这些都需要根据数据场景慎重选择
    - nosql和传统关系型数据库的区别
      - 优点：灵活的数据模型，结构比后者更丰富、更易扩展、高可用，查询效率高，传统关系型数据库受限于磁盘io，所以在高并发的情况下，压力倍增，而像redis这种内存数据库每秒支持10w次读写、nosql成本也比较低
      - 缺点：不支持sql这样的工业标准查询（学习成本高）、大多数nosql都不支持事务、nosql只能保证数据相对一致性，尤其是在数据同步的时候，主从服务器的状态是不一致的
  - **CN-DBpedia** 实际上是基于 mongo 数据库，参与开发的谢晨昊提到，一般只有**在基于特定领域才可能会用到图数据库**，就知识图谱而言，基于 json(bson) 的 mongo 就足够了。用到图查询的领域如征信，一般是需要要找两个公司之间的关联交易，会用到最短路径/社区计算等
- 知识图谱的推理，将知识图谱表示为张量tensor形式，通过张量分解（tensor factorization）来实现对未知事实的判定：
  - 用途：
    - 链接预测（判断两个实体之间是否存在某种特定关系）
    - 实体分类（判断实体所属语义类别）
    - 实体解析（识别并合并指代同一实体的不同名称）
  - 模型：
    - RESCAL模型

        - TRESCAL模型
        - 路由排序算法（PRA算法） 常用来判断两个实体之间可能存在的关系
- 知识图谱的应用： **搜索、问答、推荐系统**、反欺诈、不一致性验证、异常分析、客户管理等。以上场景在应用中出现越来越多的**深度学习模型**
    - 知识图谱在深度学习模型中的应用， 利用大量先验知识，来大大降低模型对大规模标注语料的依赖
        - 将知识图谱的语义信息输入到深度学习模型中，将离散化的知识表示为连续化的向量， 使得知识图谱的先验知识能够称为深度学习的输入**[见Knowledge Graph Embedding相关文章]**
        - 利用知识作为优化目标的约束，指导深度学习模型的学习过程，通常是将知识图谱中的知识表示为**优化目标的后验正则项**
        - **知识图谱的表示学习**用于学习实体和关系的向量化表示，其关键是合理定义知识图谱中关于事实（三元组h,r,t）的**损失函数fr(h,t)**，其总和是三元组的两个实体h和t的向量化表示。通常情况下，当事实h,r,t成立时，期望最小化fr(h,t)，实现模型：
            - 基于距离的模型: SE模型： 当两个实体属于同一个三元组时，它们的向量表示在投影后的空间中也应该彼此靠近。损失函数定义为向量投影后的距离
            - 基于翻译的模型：TransE,TransH,TransR， 通过向量空间的向量翻译来描述实体与关系之间的相关性

---

2 知识图谱，知乎专栏

- https://zhuanlan.zhihu.com/p/31864048
- KG前身1：语义网络(Semantic Network)【与语义网（Semantic Web）是两个概念】：
    - 语义网络由相互连接的节点和边组成
        - 节点表示概念或者对象
        - 边表示他们之间的关系(**is-a关系**，比如：猫是一种哺乳动物；**part-of关系**，比如：脊椎是哺乳动物的一部分)
    - 优点：
        - 容易理解和展示。
        - 相关概念容易聚类
    - 缺点：
        - 节点和边的值没有标准，完全是由用户自己定义。————>**KG中提出RDF,解决该问题**
        - 多源数据融合比较困难，因为没有标准。————>**KG中提出RDF,解决该问题**
        - 无法区分概念节点和对象节点 ————> **RDF无法解决**， 它对具体事物的描述，缺乏抽象能力，无法对同一个类别的事物进行定义和描述。**W3C制定的另外两个标准RDFS/OWL解决了这个问题**
            - 即定义Class和Object(也称作Entity，Instance)
                - e.g. 熊是哺乳动物的一个实例： is-a关系
                - e.g. 熊是哺乳动物的一个子类： subClassOf关系
        - 无法对节点和边的标签(label，我理解是schema层，后面会介绍)进行定义————> **W3C制定的另外两个标准RDFS/OWL解决了这个问题**
    - 基于语义网络的思想建立的知名项目：
        - WordNet： 一个英语的词汇库
        - BabelNet：相对于WordNet，BabelNet是一个多语言的词汇库
        - HowNet：即知网，中文语义词典
- KG前身2：语义网（Semantic Web）



    - 技术栈【指W3C制定的用于描述和关联万维网数据的一系列技术标准】：
    - **语义网的三大核心技术： RDF，OWL和SPARQL**。RDF，RDFS/OWL属于语义网技术栈，它们的提出，使得语义网克服了语义网络的缺点。 https://zhuanlan.zhihu.com/p/32122644
        - **RDF(Resource Description Framework)，即资源描述框架**，是W3C制定的，用于描述实体/资源的标准数据模型。
            - RDF图中一共有三种类型，International Resource Identifiers(IRIs)，blank nodes 和 literals:
                - IRI我们可以看做是URI或者URL的泛化和推广，它在整个网络或者图中唯一定义了一个实体/资源，和我们的身份证号类似。
                - literal是字面量，可以把它看做是带有数据类型的纯文本，比如，罗纳尔多原名可以表示为"Ronaldo Luís Nazário de Lima"^^xsd:string。
                - blank node简单来说就是没有IRI和literal的资源，或者说匿名资源。
                - SPO每个部分的类型约束：
                    - Subject可以是IRI或blank node。
                    - Predicate是IRI。
                    - Object三种类型都可以。
            - RDF对is-a关系进行了定义，即，rdf:type
            - 如何存储、传送RDF数据【序列化方法】：
                - RDF/XML： 用XML的格式来表示RDF数据； 对于RDF来说，XML的格式太冗长，也不便于阅读，通常不会使用这种方式来处理RDF数据
                - N-Triples： 用多个三元组来表示RDF数据集，是最直观的表示方法。 开放领域知识图谱DBpedia通常是用这种格式来发布数据的
                - **Turtle**：
                    - 使用得最多的一种RDF序列化方式。它比RDF/XML紧凑，且可读性比N-Triples好
                    - 会加上前缀（Prefix）对RDF的IRI进行缩写
                    - 同一个实体拥有多个属性（数据属性）或关系（对象属性），可以只用一个subject来表示，使其更紧凑， 将一个实体用一个句子表示（这里的句子指的是一个英文句号“.”）而不是多个句子，属性间用分号隔开
                - RDFa： 是HTML5的一个扩展，在不改变任何显示效果的情况下，让网站构建者能够在页面中标记实体，像人物、地点、时间、评论等等。也就是说，将RDF数据嵌入到网页中，搜索引擎能够更好的解析非结构化页面，获取一些有用的结构化信息
                - JSON-LD：即“JSON for Linking Data”，用键值对的方式来存储RDF数据
        - **RDFS【 Resource Description Framework Schema】和OWL【 Web Ontology Language】** 进行schema层的建模:

        - 模式语言/本体语言（schema/ontology language）解决了RDF表达能力有限的困境，**是用来描述RDF数据的，在概念、抽象层面对RDF数据进行定义**
        - RDFS/OWL本质上是一些预定义词汇（vocabulary）构成的集合，用于对RDF进行类似的类定义及其属性的定义
        - 在表现形式上，它们就是RDF。其常用的方式主要是RDF/XML，Turtle
        - RDFS:
            - RDFS本质上是RDF词汇的一个扩展，当中不区分数据属性和对象属性，词汇rdf:Property定义了属性，即RDF的“边”。常用词汇如下：
                - rdfs:Class. 用于定义类。
                - rdfs:domain. 用于表示该属性属于哪个类别。
                - rdfs:range. 用于描述该属性的取值类型。
                - rdfs:subClassOf. 用于描述该类的父类。比如，我们可以定义一个运动员类，声明该类是人的子类。
                - rdfs:subProperty. 用于描述该属性的父属性。比如，我们可以定义一个名称属性，声明中文名称和全名是名称的子类。
                - 其实rdf:Property和rdf:type也是RDFS的词汇，因为RDFS本质上就是RDF词汇的一个扩展。在这里不罗列进去，是避免混淆
        - OWL:
            - 主要功能：
                - 提供快速、灵活的数据建模能力。
                - 高效的自动推理。
                    - 基于本体的推理
                    - 基于规则的推理
            - owl区分数据属性和对象属性（对象属性表示实体和实体之间的关系）：
                - 词汇owl:DatatypeProperty定义了数据属性，
                - 词汇owl:ObjectProperty定义了对象属性。
            - 常用词汇：
                - 描述属性特征的词汇
                    - owl:TransitiveProperty. 表示该属性具有传递性质
                    - owl:SymmetricProperty. 表示该属性具有对称性
                    - owl:FunctionalProperty. 表示该属性取值的唯一性
                    - owl:inverseOf. 定义某个属性的相反关系
                - 本体映射词汇（Ontology Mapping）： 用于融合多个独立的Ontology（Schema）
                    - owl:equivalentClass. 表示某个类和另一个类是相同的
                    - owl:equivalentProperty. 表示某个属性和另一个属性是相同的
                    - owl:sameAs. 表示两个实体是同一个实体
    - **SPARQL(SPARQL Protocol and RDF Query Language)**，有两部分组成：
        - 协议： 通过HTTP协议在客户端和SPARQL服务器（SPARQL endpoint）之间传输查询和结果，这也是和其他查询语言最大的区别
        - 查询语言：基于图匹配的思想。把上述的查询与RDF图进行匹配，找到符合该匹配模式的所有子图，最后得到变量的值，分为以下三步：
            - 构建查询图模式，表现形式就是带有变量的RDF。
            - 匹配，匹配到符合指定图模式的子图。
            - 绑定，将结果绑定到查询图模式对应的变量上。
- KG前身3：链接数据（Linked Data）：
    - 被当做是语义网技术一个更简洁，简单的描述。
    - 当它指语义网技术时，它更强调“Web”，弱化了“Semantic”的部分。
    - 对应到语义网技术栈，它倾向于使用RDF和SPARQL（RDF查询语言）技术，对于Schema层的技术，RDFS或者OWL，则很少使用。
    - 链接数据应该是最接近知识图谱的一个概念，从某种角度说，知识图谱是对链接数据这个概念的进一步包装
        - Linked Data更强调不同RDF数据集（知识图谱）的相互链接
        - KG不一定要链接到外部的知识图谱，更强调有一个本体层来定义实体的类型和实体之间的关系。另外，知识图谱数据质量要求比较高且容易访问，能够提供面向终端用户的信息服务（查询、问答等等）
- 知识图谱：
    - 由本体（Ontology）作为Schema层，和RDF数据模型兼容的结构化数据集
        - **本体建模，**
            - 构建方式：
                - 自顶向下：常用于领域知识图谱的本体构建
                - 自底向上：常用于开放域知识图谱的本体构建
    - 由一些相互连接的实体和他们的属性构成的。由一条条知识组成，每条知识表示为一个SPO三元组(Subject-Predicate-Object)
    - 关系也称为属性（Property）
        - 根据是实体和实体之间的关系还是实体和数据值之间的关系分为:
            - 对象属性（Object Property）
            - 数据属性（Data Property）
    - 在知识图谱中，用RDF形式化地表示这种三元关系，数据来源：
        - 结构化数据：
            - RDB2RDF标准：
                - 直接映射,**缺点**【不能把数据库的数据映射到我们自己定义的本体上】：
                    - 数据库的表作为本体中的类（Class）
                    - 表的列作为属性（Property）
                    - 表的行作为实例/资源
                    - 表的单元格值为字面量
                    - 如果单元格所在的列是外键，那么其值为IRI，或者说实体/资源
                - R2RML【可以让用户更灵活的编辑和设置映射规则】：
                    -
        - 半结构化数据：数据有一定的组织形式，但较结构化数据而言更松散（属性名和属性值具有多样性，比如“生日”就有“出生日期”、“诞辰”等多种表达方式），例如百度百科、维基百科、互动百科等
        - 非结构化的数据：纯文本数据

---

3 Ontop:

- http://ontop.inf.unibz.it/
- http://wenda.chinahadoop.cn/question/10246
    - Ontop和D2RQ都可以把关系数据库转为虚拟RDF数据，二者区别?
        - D2RQ基本上就不推荐了：
            - 一方面他的mapping的表达能力和文档有一些不清晰的地方，相比R2RML的文档（W3C推荐的映射语言）清晰很多，且表达能力更强
            - D2RQ翻译SPARQL2SQL的过程缺乏query optimization，在大数据情况下性能比较差，经常会出现timeout或OOM（内存消耗比较严重的情况）
            - 此外，他不支持推理。虽然D2RQ可以将KG映射到Jena Model上，通过Jena Reasoner在其上做各种inference，但是效率可想而知。而Ontop是支持推理的（backward chaining）
- 怎样使ontop把关系数据库和图数据库中的数据联合映射成一个virtualRDFgraph以便用SPARQL统一查询?
    - 关系数据库的用ontop的Mapping映射为virtual RDF，然后用RDF4J等通过SPARQL federation来管理这个映射的virtual RDF和原本的RDF数据即可。

论文：
1 KB4Rec:ADatasetforLinkingKnowledgeBaseswith RecommenderSystems

- 构建一个开放的linked KB（knowledge base）数据集用于推荐系统(RS):knowledge-aware recommender systems
  - Freebase，将事实存储为三元组(head,relation,tail).使用的是2015年3月的版本【目前为最新发布的版本】
- RS选用三个广泛使用的数据集
  - MovieLens->movie
  - LFM-1b->music
  - Amazon book->book
- Related work for knowledge-aware recommender systems
  - 第一阶段
    - context-aware recommendation algorithms，利用社交信息【Epinions数据集】、POI属性信息【Yelp数据集】、电影属性信息【MovieLens数据集】、用户档案信息【Microblogging数据集】..
    - 利用原始RS平台的辅助信息（上下文数据）做推荐，这些数据往往辅助信息种类少，且辅助信息间关系relation patterns被忽略了
  - 第二阶段
    - HIN: ,Heterogeneous InformationNetworks，能有效学习relation patterns
    - 依赖图搜索算法，很难处理大规模关系查找
  - 第三阶段
    - KB,用于组建只是和领域事实
    - **通过连接RS items和KB entities**，但现有文献使用的均为private KB,无法获取
- 如何构建RS到KB的连接(Linkage):
  - 通过调用离线Freebase search API，用item titles作为queries,检索KB entities
  - 如果没有KB entity返回，说明RS item在linkage process被拒绝了
  - 如果至少一个KB entity返回，使用一种辅助信息作为准确连接间的一种精确的约束。e.g. IMBD ID,artist name and writer name are used for the three domains of movie, music and bookrespectively
- Linkage ratio和哪些因素可能有关?
  - KB的构建经常包含人力工作，无法忽视人类注意力的偏差
  - 流行度，成正相关，即item越流行，Linkage ratio可能越高
  - 新颖度，成负相关，即item发布时间越晚，Linkage ratio可能越低
- KB4Rec数据集的使用，选用哪些推荐算法进行试验，性能指标选取：
  - 指标：
    - MRR：mean reciprocal rank
    - HR: hit ratio
    - NDCG: normalized discounte cumulative gain
  - 推荐算法：
    - 与文献【7】相似，采用last-item recommendation task for evaluation
    - 算法：
      - BPR
      - SVDFeature
      - mCKE:第一篇文献提出利用KB和其他信息来提高推荐性能
      - KSR：Knowledge-enhanced Sequential Recommender,利用KB信息提高语义表示memory networks的性能【结果最好】
- KB4Rec数据集地址: https://github.com/RUCDM/KB4Rec#Datasets

---

2 Knowledge Graph Embedding: A Survey of Approaches and Applications

- KG embedding研究的出发点：KG的表示一般基于三元组（head entity, relation, tail entity），尽管能够有效的表示结构化数据，但是底层的本质上是符号表示，使得KG很难操作; KG embedding将KG中的成分映射到一个连续的矢量空间中，不仅保留KG中的固有结构，同时简化了处理
- KG embedding研究主要分为两个阶段：
  - 阶段1：仅利用KG中的fact构建embedding, embedding只需要和每个单独fact匹配，对下游的一些任务not predictive enough
  - 阶段2：在阶段1的基础上，利用更多的信息形式，e.g.实体类型（entity type）、关系路径(relation path)、文本描述(textual description)、逻辑规则(logical rules)，得到more predictive embeddings
- **只基于facts的KG embedding 构建由3步组成：**
  - step1: 表示entities和relations:
    - entity表示形式：
      - 矢量
      - 考虑entity的不确定性，利用多元高斯分布对entity进行建模
    - relation常被看成在矢量空间中的操作，表示形式：
      - 矢量
      - 矩阵
      - 张量
      - 多元高斯分布
      - 混合高斯（mixtures of Gaussians）
  - step2: 定义第一个打分函数
    - 每个fact(h,r,t)均对应一个score func **f_r(h, t)**，在KG中观察到的facts得分高于未观察到的，根据score function定义方式不同，这种只基于facts的KG embedding技术可被分为以下两类：
      - translational distance models：使用**基于距离的score func【目标优化函数】，这些模型均包含约束（e.g.强制vector embedding至少L2范数），这些约束在优化问题中被转化为正则项【正则项】**
        - TransE及其扩展，**实体/关系都是矢量空间中确定的点**
          - TransE：
            - 简单高效，通过学习分布式的词表示来捕捉语言规律，e.g.JamesCameron + DirectorOf ≈ Avatar
            - 处理一对多，多对一，多对多关系时有问题，e.g.一对多为例，AlfredHitchcock + DirectorOf ≈Psycho，Rebecca，RearWindow，一个导演对应多部电影，虽然这些电影属于不同实体，但是学到的矢量表示都是非常相似的，这是有问题的
          - TransE改进策略：
            - 引入 Relation-Specific Entity Embeddings:
              - TransH：
                - 改进TransE：引入 **Relation-Specific Entity Embeddings**，允许实体在不同的关系中有明显不同的表示。e.g.即使Psycho，Rebecca，RearWindow在给定DirectorOf 关系时，表示很相似，但给定其他关系时，表示可能相差很大
                - 引入relation-specific超平面，每个关系r用矢量**r**表示，在一个以**w_r**为法向量的超平面上，实体**h,t**投射到该超平面上
              - TransR：

                - 引入relation-specific 空间，而不是超平面；实体表示为实体空间的向量，每个关系关联到另外的关系空间，定义投影矩阵M_r（实体空间到关系空间）
                - 每个关系都需要引入投影矩阵，不如TransE,TransH简单高效
            - TransD:
                - 简化TransR,比TransR更高效。将投影矩阵分解为两个矢量乘积，引入额外的映射向量**w_h，w_t, w_r**
            - TranSparse:
                - 简化TransR,强制投影矩阵的稀疏性
        - relaxing translational requirement:放松**h+r≈t**的限制
            - TransM: 每个事实（三元组）关联一个权重，通过降低一对多，多对一，多对多关系的权重，TransM允许**t**在这些关系中远离**h+r**
            - ManifoldE: 放松约束关系，**t**约束在以**h+r**为质心，权重值为半径的超球体中
            - TransF: 放松约束关系，**t**约束在与**h+r**为同向即可
            - TransA: 为每个关系r引入对称非负矩阵，使用自适应Mahalanobis距离定义score
    - Gaussian Embeddings，**实体/关系被看做随机变量**
        - KG2E：将实体和关系表示成从多元高斯分布中提取的随机向量
            - 使用 Kull-back-Leibler散度计算得分
            - 使用概率内积计算得分
        - TransG: 实体**h,t**利用高斯分布建模，关系**r**认为可能有多重语义信息，被表示为混合高斯分布
    - 其他距离模型：
        - UM(unstructured model)
            - TransE的简化版本，令**r** = 0
            - 不能区分不同的关系
        - SE(structured embedding):
            - 对每个关系r，使用两个不同的投影矩阵，分别用于head entity, tail entity
- semantic matching models: 使用**基于相似度的score func**，通过匹配实体、关系见的潜在语义来衡量事实的合理性
    - RESCAL及其扩展
        - RESCAL:
            - 也叫双线性模型，将实体**h,t**与一个vector关联来捕捉潜在语义，关系**r**与一个matrix关联来建模latent factors间的两两交互
            - 其score func捕捉到了所有**h,t所有成分间**的两两交互
        - TATEC:
            - 不仅建模了h,r,t间3者交互，还定义了h,r/t,r间2者交互
        - DisMult
            - 简化了RESCAL，将矩阵**Mr**限制为对角阵
            - 其score func捕捉到了**h,t**中**相同维度上成分间**的两两交互，减少了每个关系**r**所有的参数数量
            - 模型过于简单【对角矩阵使得实体可交换】，只能处理对称关系，对于一般的KG功能不够强大
        - HolE(Holographic Embeddings)全息嵌入：
            - 将RESCAL的表现力与DisMult的简洁高效结合
            - 将实体、关系均表示为vector，进行Circular correlation，对pairwise interactions进行压缩，减少了每个关系**r**所有的参数数量，比RESCAL高效；且Circular correlation不能交换，可以像RESCAL一样，对非对称关系进行建模
        - ComplEx(Complex Embeddings)复数嵌入：
            - 对DisMult的扩展，引入复数嵌入，可以更好建模非对称关系
            - **h，r，t**不再依赖实数空间，而是依赖复数空间，非对称关系最终得到的事实会得到不同的score，这依赖相关实体对应的orders、
            - 共轭对称施加在embeddings时，HolE被视为ComplEx的一种特殊情况
        - **ANALOGY**:
            - 扩展RESCAL，进一步对实体、关系中相似的属性建模
            - 已被证明DisMult、HolE、ComplEx均属于ANALOGY的一种特殊情况
    - 利用神经网络进行匹配
        - SME: Semantic Matching Energy，
            - 在Input layer: 将fact三要素**h,r,t**映射为vector embeddings
            - 在Hidden layer：将关系**r**与head entity **h**结合得到**g_u(h,r)**;将关系r与tail entity h结合得到**g_v(r, t)**
            - score定义为**g_u,g_v**的点积
            - 根据**g_u,g_v**的形式不同，SME有两个版本：
                - SME(linear):
                - SME(bilinear)
        - NTN: neural tensor network:
            - 在Input layer: 将fact三要素**h,r,t**映射为vector embeddings
            - 在Hidden layer：将h，t，和二者与特定关系张量Mr结合三者映射到一个非线性hidden layer
        - SLM:single layer model：
            - NTN的简化形式，将h，t对应的权重矩阵，bias置零，只保留NTN中的最后一个要素
        - MLP: multi-lalyer perceptron：
            - h，r，t均映射为单vector
            - 在Input layer上将三者拼接，映射到非线性hidden layers
        - NAM:neural association model:（多隐层，其他都是单隐层）
            - 在Input layer: 将fact中**h,r**映射为vector embeddings后进行拼接，经过多隐层（激活函数：Relu）和**t**生成的embeddings乘积得到score

- step3: 学习entities和relations的表示
    - 解决对所有观测facts的合理性最大化(maximize plausibility)的最优化问题
    - **模型的训练:**
        - 一些先验知识：http://www.sohu.com/a/144575100_464088
            - 封闭世界假设(Closed World Assumption, CWA)
                - 即如果我们在知识库中推不出来P或P的否定，就把P的否定加入知识库。有两种情况, CWA很有用. 一是可以**当假设知识库中的知识是完全的时候**. 例如, 在数据库中, 如果学生表中没有Peter, 则认为Peter不是学生. 二是当知道知识库的知识是不完全的, 如不足于回答一些问题, 但我们**必须在不完全知识的情况下做出决定**, 这时候CWA就有用了
            - 开放世界假设(Open World Assumption, OWA)
                - 对推不出来的命题就很诚实地当作不知道这个命题的正确与否, 这样的后果就是知识库中能推导出来的结论大大减少
            - 在语义Web环境下, 因为Web的开放性, 相关的知识很可能分布在Web上不同的场所, 因此在语义Web上推理, 用CWA是很不恰当的. 例如, 如果在一个知识库中只说了hasFriend(Peter, Tom), 如果采用CWA, 就会得到结论: Peter只有一个朋友. 这当然是不合理的, 因为很可能在别的地方说了Peter还有其他的朋友. 所以, 如果要在语义Web中聚集不同来源的知识, 应该采用OWA. (有一种中庸之道: **局部封闭世界(Local Closed World)**, 这里不多说). 描述逻辑中的推理刚好是采用**OWA的, 所以它的确适合作为语义Web的逻辑基础**
        - 基于OWA的训练：
            - 样本形式：
                - 正样本集：
                - 负样本集：
            - 目标函数：
                - logistic loss最小化
                    - 优势：对一些复杂的关系模式（如transitive relations）能得到一些紧凑的表达方式
                - pairwise ranking loss最小化
                    - 优势：不假设负样本一定是(命题)错误的，只是和正样本相比可能性小，使得positive facts得分要尽可能高于negative ones
                - 以上的目标函数中均包含约束项/正则项【不同的embedding模型不同】：

                - 已证明：logistic loss+semantic matching models（DisMult、ComplEx等）性能更好；
                - pairwise ranking loss+translational distance models（TransE）性能更好
            - 优化方法：SGD+minibatch
                - 初始化entity和relation embeddings后，每轮迭代，从正样本集中抽取positive facts集合，对每个positive fact依次产生一个或多个 negative facts，这些正负样本构成一个minibatch中的训练样本，经过一次minibatch，embeddings通过一个gradient step(具有恒定的或自适应的学习率)更新
                - 负样本构建策略有很多，不做总结了
                - 负采样对性能影响：越多越好，k=50是一个对准确率和训练时间很好的tradeoff
    - 基于CWA的训练：
        - 目标函数：
            - 最小均方误差，squared loss
            - logistic loss
            - absolute loss
        - 缺点：
            - 不适用于不完整的KG，即观测数据中有很多的missing facts
            - 已被证明，基于CWA的模型性能要比基于OWA的模型差
            - 引入大量的负样本，在模型训练时会引入扩展性问题
- **模型的比较：**
    - 模型的性能与具体的任务和采用的数据都有关系
        - 表现力丰富的模型性能不一定会更好，往往需要大量参数，在小型/中型数据集上容易过拟合
        - 在link preidiction任务中，采用WordNet和Freebase数据集，**ANALOGY**性能最好
        - 除了在step2中引入的模型：事实（三元组）还可利用**paired formulation**进行建模：
            - 将entity pair->(h,t)表达为矢量**p**,r表达为矢量**r**，fact的合理性有两个矢量内积决定，**这种entity pair的表示经常在关系抽取中使用**
            - 将head entity h表达为**h**，（r,t）->表达为矢量**p**
            - 缺点：
                - 未成对的实体间关系不易发现
                - 空间复杂度增加
                - 对于两个首尾实体对h1,t和h2,t，如果共享tail entity t,但是h1,t与 h2,t采用不同的vector representation，那么二者共享的tail entity信息就会丢**失了**

- **利用额外的信息进行KG embedding，包括 entity types, relation paths, textual descriptions, as well as logical rules，对只基于facts的KG embedding 模型进行优化**
    - 基于实体类型，即实体属于哪个语义范畴：
        - e.g. AlfredHitchcock(阿尔弗莱德·希区柯克)的type为Person，这种信息在多数KGs中都是可用的
        - 通常以特定关系编码，并以triples形式保存，e.g.（Psycho, **IsA**, CreativeWork). 即使用IsA作为普通的关系，相应的三元组作为普通的训练样本
        - SSE(semantically smooth embedding)
            - 假设：具有相同type的实体在embedding space中stay close
            - 利用 Laplacian eigenmaps和locally linear embedding构建正则化项，对embedding task进行约束
            - 缺点：认为实体的语义范畴不是非层次结构的，每个实体直属一个种类，这与真实世界中的KGs是矛盾的
        - TKRL(type-embodied knowledge representation learning)
            - 能够处理分层的实体类别和多类别标签，解决了SSE的不足
            - 是一个特定实体类型预测（type-specific projection ）的翻译距离模型（translational distance models），它对打分函数进行了优化
                - 利用type-specific projection matrices对h,t进行映射
                - 多类别标签：投影矩阵中元素表示为所有可能的type matrices的加权和
                - 分层的类别：将所有子领域的project matrices矩阵进行组合
                    - 加性组合
                    - 乘性组合
            - 缺点：尽管在一些任务中如 link prediction和triple classification性能良好，但空间复杂度相对高
        - 实体类型还能用来对不同关系中的 head and tail positions进行**类型约束**
            - e.g. 具有关系DirectorOf的head entity类型应为Person
    - 基于关系路径(relation path)，即实体间的多跳关系：
        - 包含丰富的语义线索，对KGs的补全有重要作用
        - 在多元关系数据中被广泛研究，e.g.路径排名算法：
            - 利用两实体间的路径作为特征去预测二者潜在的关系
            - 应用于KG embedding，如何将这些paths与entities和relations表示在相同的向量空间中?
                - **使用组合策略**：将path表示为组成它的关系的表示形式的组合，常用的组合策略：
                    - 加性组合
                    - 乘性组合
                    - RNN
        - PTransE(path-based TransE): TransE的扩展，
            - PTransE考虑了以上所有的组合策略（3种）
            - 与TransE相比，性能有大幅度提高
        - 文献【28】提出另一种相似的架构：
            - 构建三元组：using entity pairs connected not only with relations but also with relation paths，使用（h,p,t）p=r1->r1->r2->r3->...->rl
            - 模型是对 TransE和RESCAL的扩展：
                - 对二者的score function进行扩展:
                    - TransE中r->p,采用加性模型，r->r1+r2+...+rl
                    - RESCAL中r->p,采用乘性模型，M->M1*M2*...*Ml
            - 这种方法在answering path queries on KGs上性能很好
        - 尽管性能提升了，但是大量的paths会引入复杂度的挑战，部分文献通过对现有方法进行采样、剪枝、动态规划算法等改善存在的问题
    - 基于文本描述(textual description)，KGs中对实体简洁的描述包含了丰富的语义信息：
        - 原始模型：NTN，仅利用文本信息去初始化实体表示，**将KG facts和文本信息隔离，不能很好利用二者间的交互**
        - 文献【30】提出模型，解决了NTN中的问题：
            - 将制定KG与一个附加的文本语料对齐，将KG embedding和word embedding共同创建
            - 实体/关系与词的表示在同一矢量空间中，更有意义
            - 联合建模包含3个部分：
                - knowledge model, 使用TransE的变种
                - text model, 使用Skip-gram的变种
                - alignment model，保证实体/关系向量与词向量在同一矢量空间中，可能会利用很多机制：
                    - 通过实体名称对齐
                    - 通过 Wikipedia anchors对齐
                    - 通过实体描述对齐
            - 联合embedding利用了KGs的结构化信息和文本的非结构化信息，能够彼此增强，此外，这两类信息的对齐能实现**KG外实体的预测**，e.g. phrases 出现在web text，但还未出现在KGs
        - DKRL(description-embodied knowledge representationlearning)
            - 也是对TransE的扩展，实验结果表面在zero-shot scenario with out-of-KG entities场景，DKRL性能优于TransE
            - 用两种矢量表达描述实体e:
                - 基于结构化的es，捕捉了KG中facts反映的结构化信息
                - 基于描述的ed，捕捉了实体描述中的文本信息

        - TEKE(text-enhanced KG embedding model)
            - 首先在语料库中注释实体，创建一个由实体和单词组成的共现网络
            - 对每个实体e,在共现网络中定义一个文本上下文n(e)作为其邻居，文本上下文的向量**n(e)**定义为在n(e)中词向量的加权平均
            - 对三元组中的每个关系r，TEKE定义了一个文本上下文作为h和t的公共邻居n(h, t) = n(h) 交集 n(t)，用与上面相似的方法得到向量**n(h, t)，A,B**为权重矩阵，更新三元组
                - **r_new = Bn(h,t)+r**
                - **h_new = An(h)+h**
                - **t_new = An(t)+t**
            - 能学到更丰富的实体/关系表达，性能胜过原始模型TransE，TransH，TransR
    - 基于逻辑规则(logical rules)，依据一阶**霍恩子句**:
        - e.g. 任意x;y : HasWife(x;y)=>HasSpouse(x;y)，任意通过关系HasWife连接的两个实体，也应该被关系HasSpouse连接
        - 逻辑规则包含丰富的背景知识，**广泛在知识获取和知识推理等方面被研究**，常基于马尔科夫逻辑网络
        - 现有的系统如 WARMR , ALEPH , 和AMIE能自动从KGs中提取逻辑规则
        - 文献【23,24】提出利用规则改善embedding模型，但是规则建模与embedding建模是隔离的，只是增加了一个后期加工个的步骤，性能提升有限
        - 文献【34，35】提出将二者同时建模，提出KALE，使得facts和rules在一个统一的框架中建模
            - 学习到的embedding不仅与facts兼容，还与rules兼容，在**知识获取和知识推理**上更有效
            - facts被当做一个 ground atom，定义其truth value
            - 逻辑规则首先需要实例化为ground rules
                - e.g. 任意x;y : HasWife(x;y)=>HasSpouse(x;y)实例化结果：任意x;y : HasWife(AlfredHitchcock;AlmaReville)=>HasSpouse(AlfredHitchcock;AlmaReville)
                - 需要执行grounding procedure，缺点是当KGs中存在大量的实体，或者它们的规则非常复杂时会很耗时耗空间
            - 实例化后，ground rules会被解释为由ground atoms和logical connectives构成的**复杂规则**，可通过 t-norm fuzzy logics 建模
        - 文献【111】作为文献【35】扩展，解决其存在问题：
            - 避免grounding，不会实例化x,y，但这种策略只适用于简单规则，**不能推广到复杂的规则**

- **利用其它信息**
    - 利用实体属性
        - KGs中的关系不仅能表示实体间的关系【对象属性】还能表示实体的属性【数据属性】
            - 但大多数KGs embedding技术并未对二者进行区分，以RESCAL为例，不区分的结果导致tensor维度大幅度增加
            - 文献【22】提出区分二者，relations仍用tensor进行编码，attributes用一个单独的entity-attribute矩阵
    - 利用时间信息
        - KGs对时间是敏感的，对模型添加时间顺序约束，比如BORNIN和DIEIN是有先后顺序的
        - time-aware embedding model
    - 利用图结构
        - graphaware embedding model
        - 利用三种图结构学习entity和relation的表达
            - neighbor context：类似KGs中的三元组
            - path context：类似之前介绍的 relation paths
            - edge context：给定一实体，其edge context定义为所有的relations连接到该实体或者从该实体引出的（入度+出度）
    - Evidence from Other Relational Learning Methods
        - 结合pathranking algorithm (PRA)
            - MLP+PRA
            - RESCAL+PRA
- **KG embedding在KG中的应用**
    - link prediction，也叫作entity prediction或者entity ranking
        - 预测一个实体是否与另一实体间存在特定的关系
            - （？，r, t）或者（h,r,?）
            - 本质上就是一个KG completion任务，添加graph中确实的knowledge
        - 类似的任务，（h,?，t）即 relation prediction
        - 在完成实体和关系的表示后，link prediction只需要在进行一个排名步骤ranking
    - triple classification
        - 用于验证一个unseen triple fact （h,r,t）是否为true
        - 在完成实体和关系的表示后，每个三元组都有一个得分，triple classification可在score的基础上进行，三元组得分越高的越可能为true fact
        - 对unseen triple fact进行预测，对score设置门限，门限以上为true，否则为false
    - entity classitication
        - 将实体划分到不同的语义范畴
        - 可以当做link prediction特例，（x, IsA，？），是一个KG completion任务
    - entity resolution:实体解析
        - 验证两个实体是否属于同一个对象
        - 文献【18】指定这样的场景：
            - KG已经包含了relation用于说明两个实体间是否是等价的，即表示为'EQUALTO', embedding也已经从relation中学到
            - 这样，实体解析退化为triple classification，即判断（x;EqualTo;y）是否保留，或有多少可能保留
            - 这种直观策略并不总能work，因为不是所有KG都会对关系'EQUALTO'进行encode
        - 文献【13】提出只在实体表达的基础上进行实体解析
- **KG embedding在非KG中的应用**
    - 关系抽取
        - 在纯文本中在实体已经被识别出之后抽取relational facts
        - 文献【20】提出将一个text-based extractor和TransE结合，可以更好的进行关系抽取
        - 【缺】
    - 问答系统【缺】
    - 推荐系统
        - 文献【133】提出一种混合的推荐架构，利用了KG中的异构信息去提升协同过滤的性能
            - user的latent vector【**u_i**】
            - **使用了KG中存储的三种信息**，去构建item的latent vector【**e_j**】
                - 包含结构化知识（triple facts）、文本知识（一本书或者电影的textual summary）、视觉知识（一本书的封面或者电影的海报图片）去导出items的语义表达
                    - 结构化知识（triple facts）：可使用典型的KG embedding技术，e.g. TransR
                    - 文本知识：使用堆叠的去噪自编码器抽取
                    - 视觉知识：使用堆叠的卷积自编码器抽取
            - user i对item j的偏好定义为两个latent vector的乘积