法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院和讲者共同拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象
  - 新浪微博：ChinaHadoop

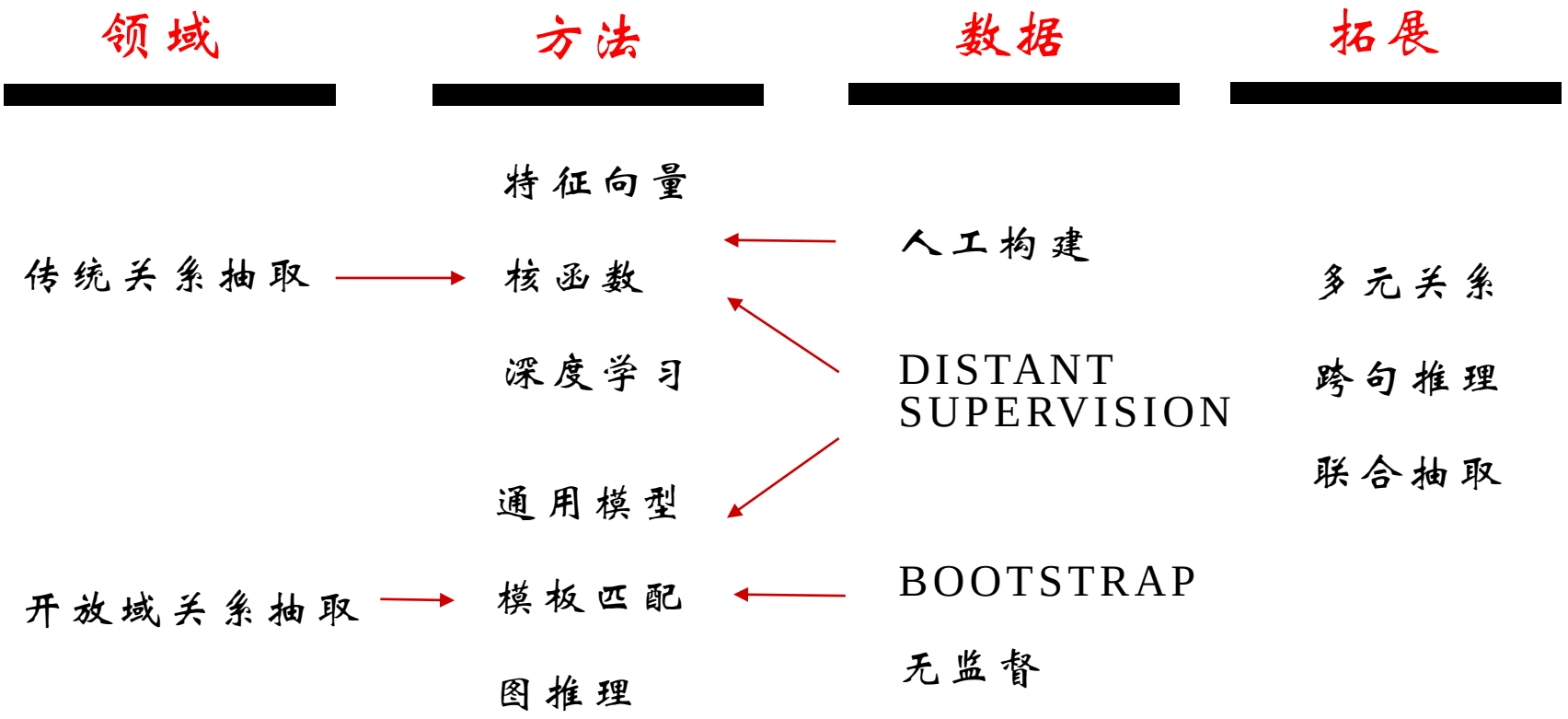
第四讲 知识抽取与挖掘II

大纲

- 面向文本的知识抽取
  - DeepDive关系抽取实战
  - 开放域关系抽取
- 知识挖掘
  - 实体消歧与链接
  - 知识规则挖掘
  - 知识图谱表示学习

第一部分:面向文本的知识抽取

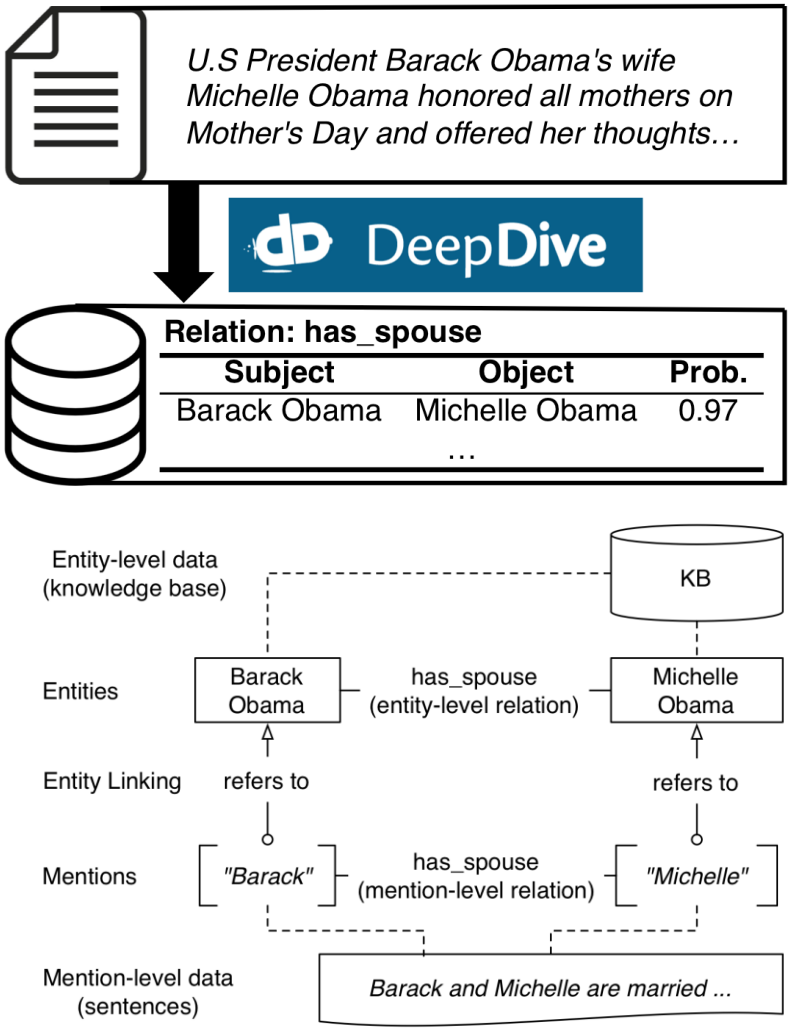
关系抽取分类


领域
方法
数据
拓展
传统关系抽取
开放域关系抽取
特征向量
核函数
深度学习
通用模型
模板匹配
图推理
人工构建
DISTANT SUPERVISION
BOOTSTRAP
无监督
多元关系
跨句推理
联合抽取

Deepdive
关系抽取实战

KBC系统

- 填充、融合不同来源的知识
- 输入
  - 非结构化的期刊文章
  - 半结构化的html、table等
- 输出
  - 结构化知识库



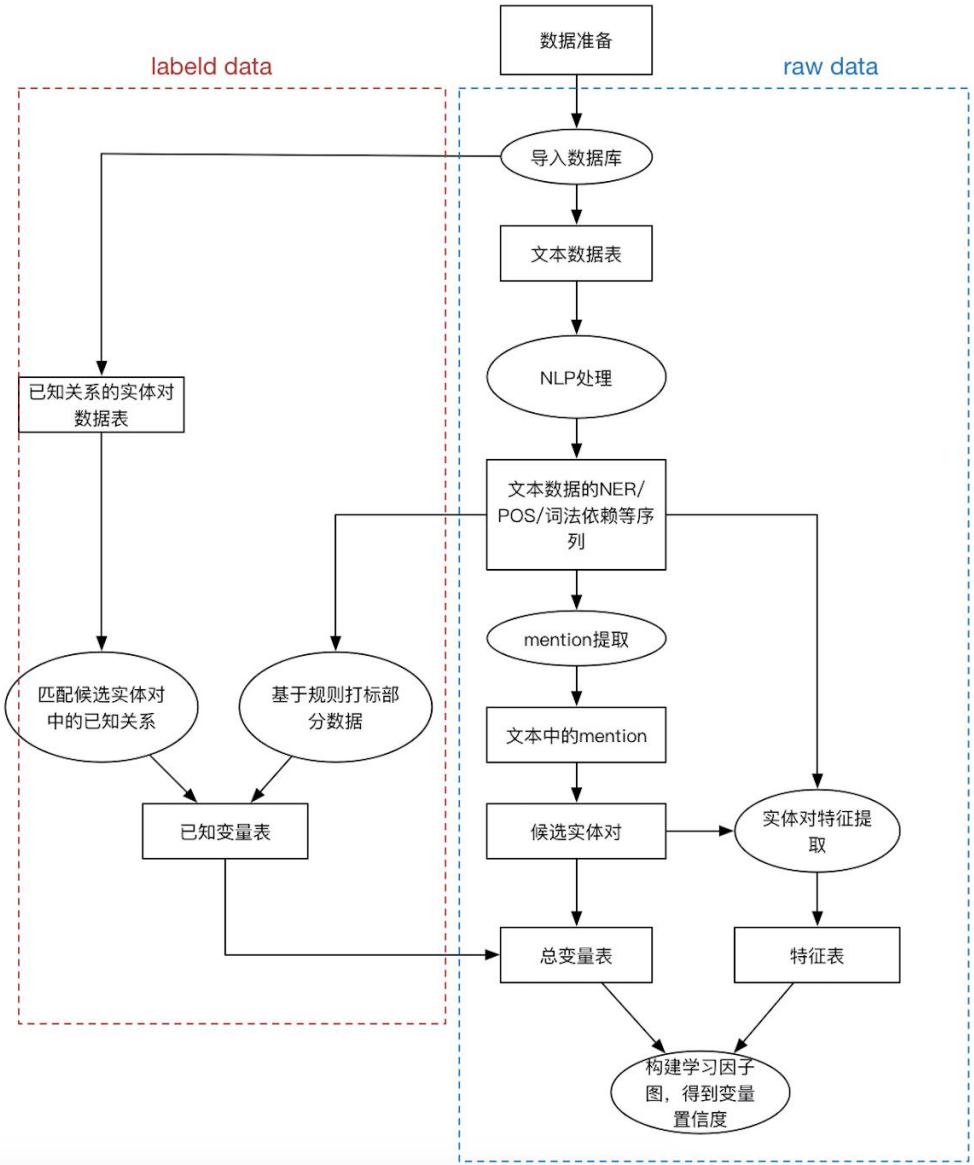
Deepdive

- KBC系统的自动搭建框架
  - 特征工程 + distant supervision + 图优化
  - 考虑全局最优而不是某个三元组最优
  - 帮助领域专家自主搭建KB，专家只需要填充领域知识，不用考虑算法性能等问题

- 技术难点
  - 设计一个KBC系统的工作流，包括文本预处理、特征抽取、统计推理与学习、迭代优化等。用户通过自定义datalog语言调控这些过程。
  - 利用分布式数据库大幅度提升系统性能

KBC流程

- 特征抽取
  - OCR、NLP工具等
  - 用户自定义脚本
- 专业知识融合
  - 在整个知识库上进行多种关
- 监督学习
  - distant supervision
- 迭代优化
  - 如何分析模型并提升最终结



框架实战：
抽取上市公司中的股权交易关系

工程组成

- 主程序文件
  - 数据表定义
  - 调用用户脚本 (udf)
    - 文本预处理
    - 特征抽取
  - 因子图定义
- 数据库文件
  - 定义数据表的存储位置
- 工程配置文件
  - 定义采样及训练方法
- 输入文件夹
  - 存储训练数据及文本
- udf
  - 用户自定义python脚本

先验数据导入

- 准备抽取所需的先验数据(国泰安)
  - 从知识库中获得已知具有交易关系的实体对
  - 命名为pos_transaction.csv，放在input文件夹下

pos_transaction.csv

科修达有限公司	万科企业股份有限公司
深圳市全新好股份有限公司	上海上海量宽技术有限公司
深圳市宝安宝利来实业有限公司	神州高铁技术股份有限公司
中国北方工业公司	中国南玻集团股份有限公司
前海人寿保险股份有限公司	中国南玻集团股份有限公司
深圳南玻显示器件科技有限公司	中国南玻集团股份有限公司
深圳南玻显示器件科技有限公司	中国南玻集团股份有限公司
深业沙河（集团）有限公司	沙河实业股份有限公司
深圳中恒华发股份有限公司	深圳市中恒华发科技有限公司
深圳能源集团股份有限公司	国泰君安

- 在app.ddlog中定义相应的数据表

```
@source
pos_transaction(
    @key
    company1_name text,
    @key
    company2_name text
).
```

- 命令行生成postgresql数据表
  - $ deepdive do pos_transaction
  - 对于定义输入的表格，deepdive会自动去input文件夹下找到同名csv文件，在postgresql里建表导入

```
        company1_name          |        company2_name
-------------------------------+-----------------------------
 科修达有限公司                | 万科企业股份有限公司
 深圳市全新好股份有限公司      | 上海上海量宽技术有限公司
 深圳市宝安宝利来实业有限公司 | 神州高铁技术股份有限公司
 中国北方工业公司              | 中国南玻集团股份有限公司
 前海人寿保险股份有限公司      | 中国南玻集团股份有限公司
 深圳南玻显示器件科技有限公司 | 中国南玻集团股份有限公司
 深圳南玻显示器件科技有限公司 | 中国南玻集团股份有限公司
 深业沙河（集团）有限公司      | 沙河实业股份有限公司
 深圳中恒华发股份有限公司      | 深圳市中恒华发科技有限公司
 深圳能源集团股份有限公司      | 国泰君安
(10 rows)
```

待抽取文章导入

- 准备待抽取的文章，命名为articles.csv，放在input文件夹下(上市公司公告)

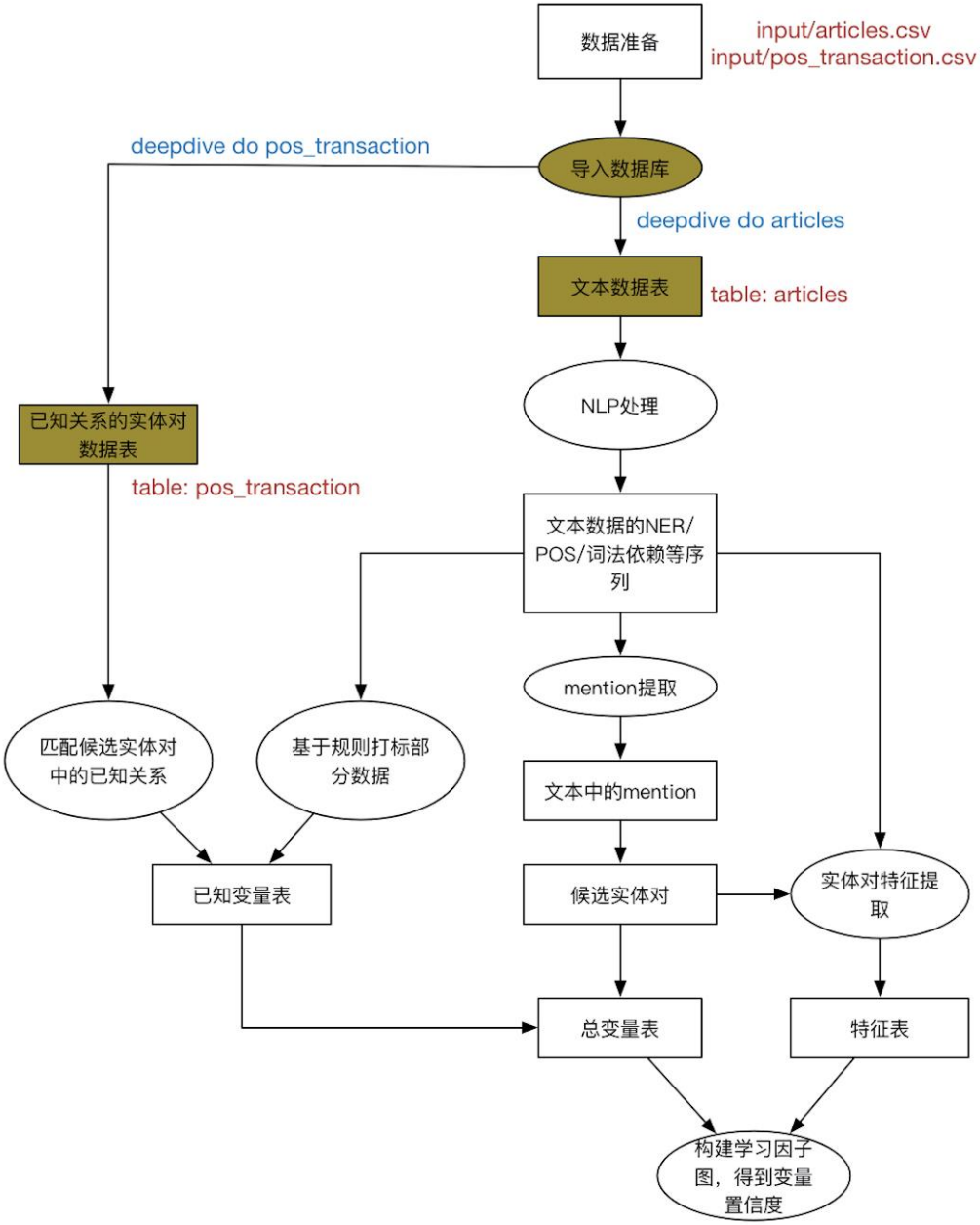
1201734370	证券代码:600969 证券简称:郴电国际 编号:公告临 2015-033　湖南郴电国际发展股份有限公司　为郴州市城市建设投资发展集团有限公司
1201734454	证券代码:600108 证券简称:亚盛集团 公告编号:临2015-067　甘肃亚盛实业(集团)股份有限公司　第七届监事会第四次会议决议公告甘肃亚
1201734455	证券代码:600108 证券简称:亚盛集团 公告编号:2015-068　甘肃亚盛实业(集团)股份有限公司　关于黑河黄藏寺水利枢纽工程　征占用
1201734457	证券代码:600108 证券简称:亚盛集团 公告编号:2015-072　甘肃亚盛实业(集团)股份有限公司　关于全资子公司间股权转让的公告　本公
1201734458	证券代码:600108 证券简称 :亚盛集团 编号:临 2015-071　甘肃亚盛实业(集团)股份有限公司　关于为全资子公司提供担保的公告　本公司
1201734460	证券代码:600079 证券简称:人福医药 编号:临 2015-100 号人福医药集团股份公司 关于控股子公司通过 FDA 认证的公告近日，人福医药集
1201734461	证券代码:600108 证券简称:亚盛集团 公告编号:2015-069　甘肃亚盛实业(集团)股份有限公司 关于转让工业用地土地使用权的关联交易公
1201734464	证券代码:600108 证券简称 :亚盛集团 公告编号:2015-070　甘肃亚盛实业(集团)股份有限公司　关于为控股子公司提供担保的公告重要内
1201738707	证券代码:600483 证券简称:福能股份 编号:2015-088　福建福能股份有限公司　关于变更为控股子公司提供担保金额的公告本公司董
1201738753	证券代码:600642 股票简称:申能股份 编号:临 2015－025　申能股份有限公司关于 向中天合创能源有限责任公司提供担保的　关联交

- 在app.ddlog中定义文章数据表，包括doc_id和content

```
@source
articles(
    @key
    id      text,
    @searchable
    content text
).
```

- 同理，用deepdive do articles导入文章到postgresql里

工作流


数据准备
input/articles.csv
input/pos_transaction.csv
deepdive do pos_transaction
导入数据库
deepdive do articles
文本数据表
table: articles
NLP处理
已知关系的实体对数据表
table: pos_transaction
文本数据的NER/POS/词法依赖等序列
mention提取
匹配候选实体对中的已知关系
基于规则打标部分数据
文本中的mention
已知变量表
候选实体对
实体对特征提取
总变量表
特征表
构建学习因子图，得到变量置信度

文章数据预处理

- 对数据库中的文章数据进行NLP解析，为后续特征抽取做准备

- 在app.ddlog中定义sentences表，用于存放POS、NER等字段

```
@extraction
sentences(
    doc_id  text, sentence_index int, sentence_text  text,
    tokens  text[], lemmas text[], pos_tags text[],
    ner_tags   text[], doc_offsets int[], dep_types text[],
    dep_tokens int[]
).
```

- 定义NLP处理的函数nlp_markup
  - 输入为doc_id和content，输出按sentence表的字段格式
  - 函数调用nlp_markup.sh这个脚本实现NLP的处理，可自由发挥

```
function nlp_markup over (
        doc_id text,
        content text
    ) returns rows like sentences
    implementation "udf/nlp_markup.sh" handles tsv lines.
```

文章数据预处理

- nlp_markup.sh
  - 这里是调用udf/bazaar/parser下的run.sh进行实现的，run.sh调用了stanford nlp集成好的jar包。
  - 这一步要逐句做NER、词法分析等，耗时比较久。

```
set -euo pipefail
cd "$(dirname "$0")"

: ${BAZAAR_HOME:=$PWD/bazaar}
[[ -x "$BAZAAR_HOME"/parser/target/start ]] || {
    echo "No Bazaar/Parser set up at: $BAZAAR_HOME/parser"
    exit 2
} >&2

[[ $# -gt 0 ]] ||
    # default column order of input TSV
    set -- doc_id content

# convert input tsv lines into JSON lines for Bazaar/Parser


# start Bazaar/Parser to emit sentences TSV
tsv2json "$@" |
"$BAZAAR_HOME"/parser/run.sh -i json -k doc_id -v content
```

把输入的articles的行转化为json，传给stanford nlp

文章数据预处理

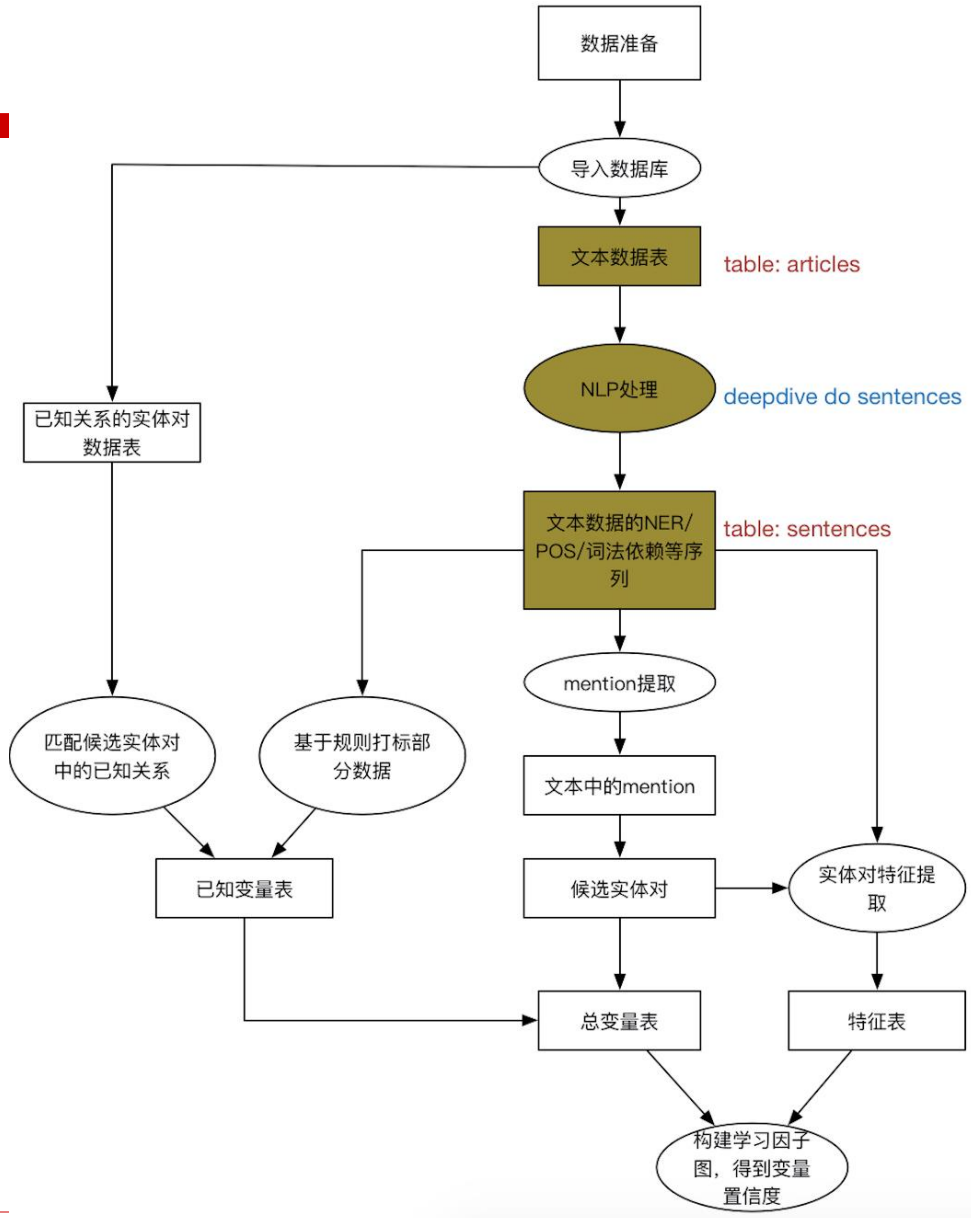
- 函数调用，从articles表中读取输入，输出存放在sentences表中

```
sentences += nlp_markup(doc_id, content) :-
    articles(doc_id, content).
```

- 编译并执行$ deepdive do sentences，生成sentences表

```
 24     |              1 | 泛海控股股份有限公司以272，961.98万元的价格收购控股股东中国泛海控股集团有限公司所持有的中国民生信托有限公司59.65%股权 | {泛海,控股,股份,有限,公司,以,272,,,961.98万,元,的,价格,收购,控股,股东,中国,泛海,控股,集团,有限,公司,所持,有的,中国,民生,信托,有限,公司,59.65%,股权} | {泛海,控股,股份,有限,公司,以,272,,,961.98万,元,的,价格,收购,控股,股东,中国,泛海,控股,集团,有限,公司,所持,有的,中国,民生,信托,有限,公司,59.65%,股权} | {NR,NN,NN,JJ,NN,P,CD,PU,CD,M,DEC,NN,NN,VV,NN,NR,NR,NN,NN,JJ,NN,VV,PN,NR,NN,NN,JJ,NN,CD,NN} | {ORG,ORG,ORG,ORG,ORG,O,O,O,MISC,MISC,O,O,O,O,O,ORG,ORG,ORG,ORG,ORG,ORG,ORG,O,O,ORG,ORG,ORG,ORG,ORG,O,O} | {0,2,4,6,8,10,11,14,15,22,23,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,64} | {nn,nn,nn,amod,nsubj,case,dep,"",nummod,relcl,mark,nn,prep,"",nn,nn,nn,nn,nn,amod,dobj,conj,nn,nn,nn,nn,amod,nn,nummod,dobj} | {3,3,5,5,14,13,10,0,10,13,10,13,14,0,21,19,19,19,21,21,14,14,30,26,26,28,28,30,30,22}
(1 row)
```

工作流


数据准备
导入数据库
文本数据表
table: articles
NLP处理
deepdive do sentences
已知关系的实体对数据表
文本数据的NER/POS/词法依赖等序列
table: sentences
mention提取
匹配候选实体对中的已知关系
基于规则打标部分数据
文本中的mention
已知变量表
候选实体对
实体对特征提取
总变量表
特征表
构建学习因子图，得到变量置信度

候选实体抽取

- 抽取文本中的候选实体
- 在app.ddlog中定义候选实体表

```
@extraction
company_mention(
    mention_id  text, mention_text   text,
    doc_id      text, sentence_index int,
    begin_index int,end_index      int
).
```

- 定义候选实体抽取的函数map_company_mention
  - 输入为sentence的token和对应的NER标签
  - 函数调用map_company_mention.py抽取NER为ORG的标签，并返回实体和在句中的位置

```
function map_company_mention over (
    doc_id         text,
    sentence_index int,
    tokens         text[],
    ner_tags       text[]
) returns rows like company_mention
implementation "udf/map_company_mention.py" handles tsv
lines.
```

候选实体抽取

- map_company_mention.py
  - 找到连续的ORG标签作为实体，并定位实体位置。
  - 可在此处对一些误识别的非实体进行过滤。

```
def extract(
        doc_id         = "text",
        sentence_index = "int",
        tokens         = "text[]",
        ner_tags       = "text[]",
    ):
    """
    Finds phrases that are continuous words tagged with company_name.
    """
    num_tokens = len(ner_tags)
    # find all first indexes of series of tokens tagged as company_name
    first_indexes = (i for i in xrange(num_tokens) if ner_tags[i] == "ORG" and (i == 0 or ner_tags[i-1] != "ORG"))
    for begin_index in first_indexes:
        # find the end of the company_name phrase (consecutive tokens tagged as company_name)
        end_index = begin_index + 1
        while end_index < num_tokens and ner_tags[end_index] == "ORG":
            end_index += 1
        end_index -= 1
        # generate a mention identifier
        mention_id = "%s_%d_%d_%d" % (doc_id, sentence_index, begin_index, end_index)
        mention_text = "".join(map(lambda i: tokens[i], xrange(begin_index, end_index + 1)))
        # Output a tuple for each company_name phrase
        yield [
            mention_id,
            mention_text,
            doc_id,
            sentence_index,
            begin_index,
            end_index,
        ]
```

找到每个ORG标签起始位置

从起始位置往后遍历，找到结束位置

候选实体抽取

- 函数调用，从sentences表中读取输入，输出到company_mention中

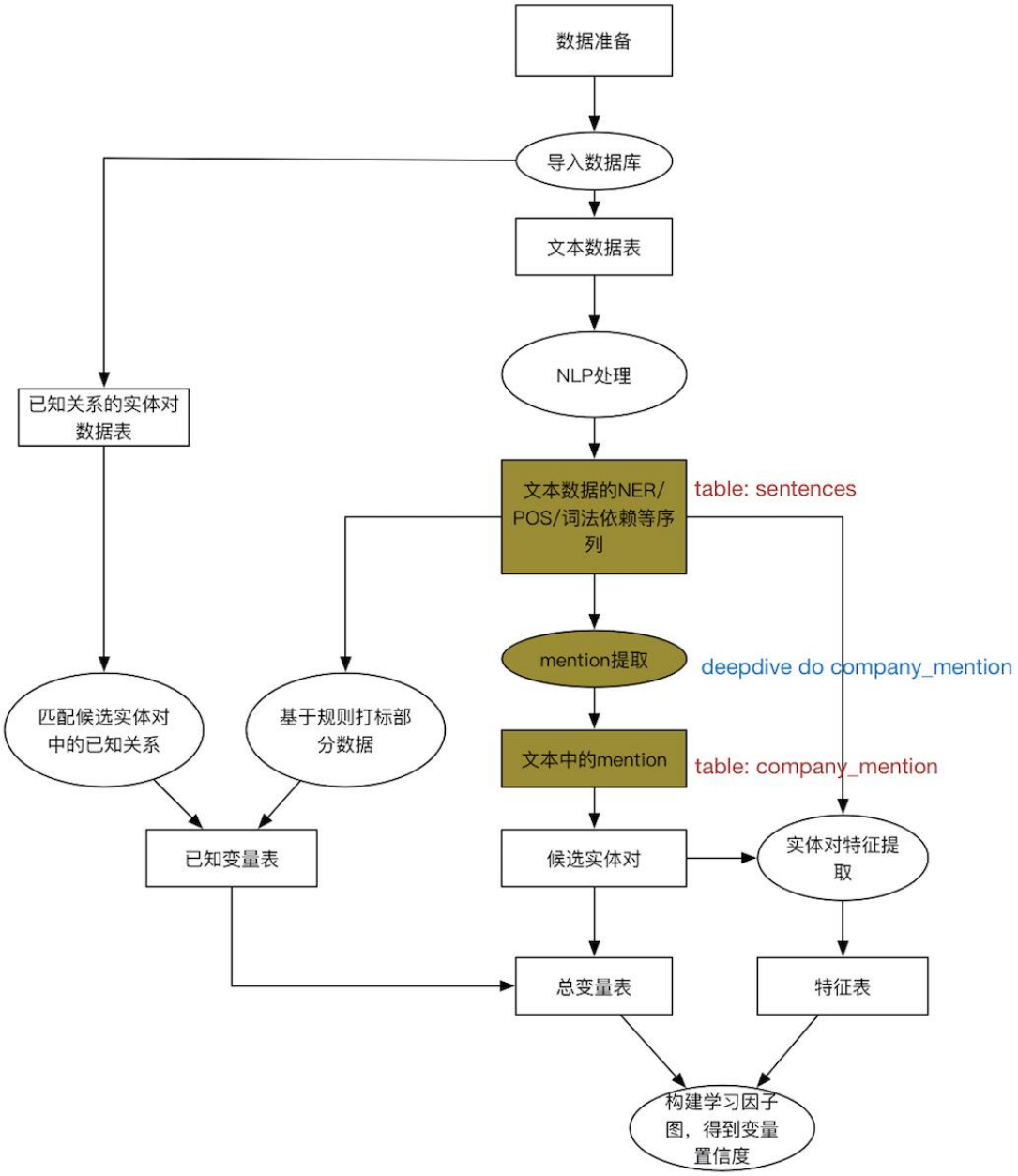
```
company_mention += map_company_mention(
doc_id, sentence_index, tokens, ner_tags
) :-
sentences(doc_id, sentence_index, _, tokens, _, _,
ner_tags, _, _, _).
```

- 编译并执行$ deepdive do company_mention，生成候选实体表

mention_id	mention_text	doc_id	sentence_index	begin_index	end_index
1201841277_4_3_3	前本公司	1201841277	4	3	3
1201748452_7_0_4	宁波港股份有限公司	1201748452	7	0	4
1201748457_5_0_4	天津松江集团有限公司	1201748457	5	0	4
1201773382_17_0_4	中电广通股份有限公司	1201773382	17	0	4
1201748457_11_1_5	天津松江股份有限公司	1201748457	11	1	5
1201764903_23_1_6	上海医药集团股份有限公司	1201764903	23	1	6
1201755138_6_2_4	上海丰华	1201755138	6	2	4
1201766320_21_0_2	天津市房地产发展	1201766320	21	0	2
1201782465_50_3_5	阳泉煤业集团	1201782465	50	3	5
1201763494_2_10_10	国旅	1201763494	2	10	10

(10 rows)

工作流



候选实体对生成

- Join实体表，筛选出在同句中的不同实体，生成候选实体对

```
@extraction
transaction_candidate(
    p1_id   text,
    p1_name text,
    p2_id   text,
    p2_name text
).
```

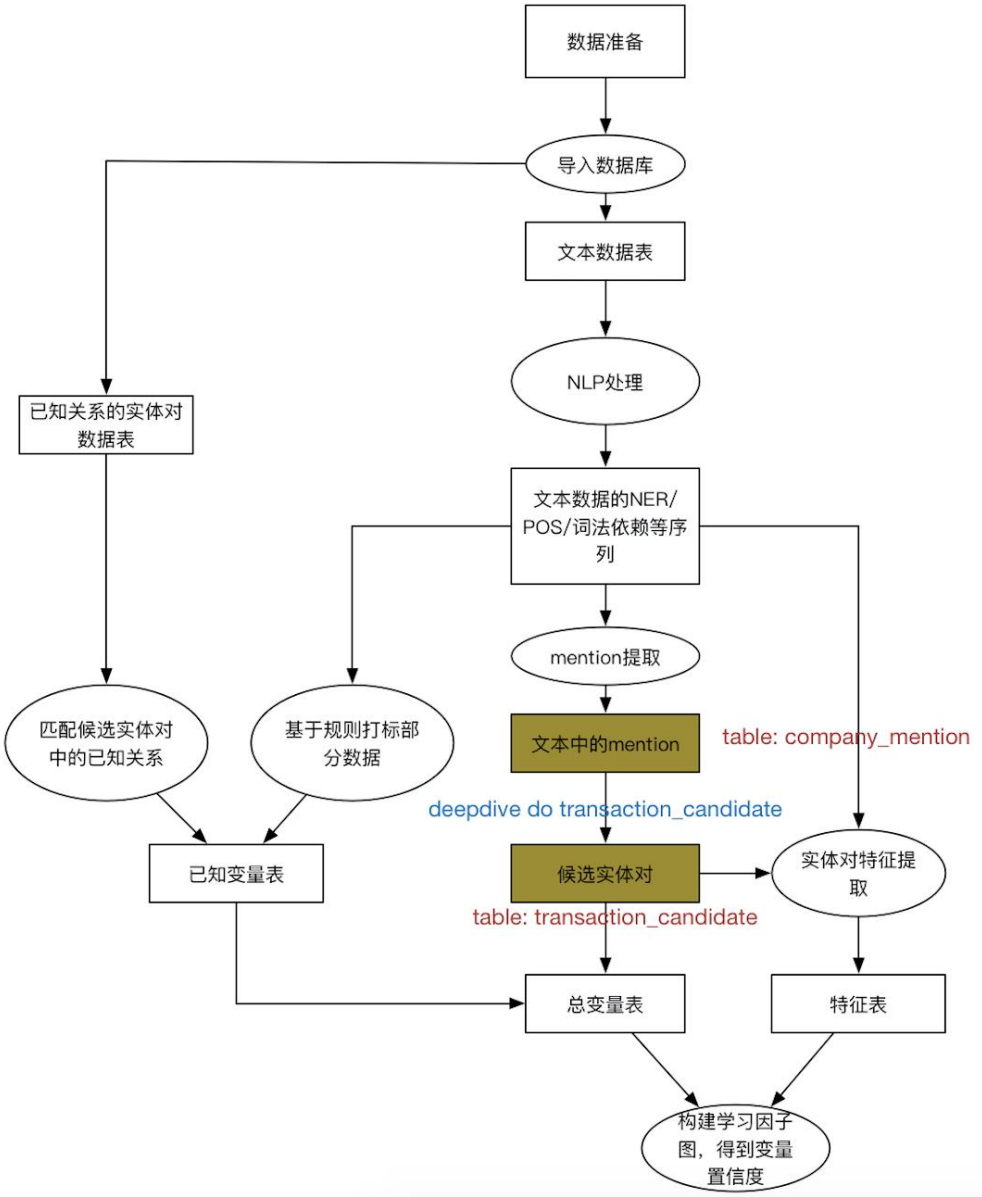
```
transaction_candidate(p1, p1_name, p2, p2_name) :-
    company_mention(p1, p1_name, same_doc, same_sentence,
p1_begin, _),
    company_mention(p2, p2_name, same_doc, same_sentence,
p2_begin, _),
    p1_name != p2_name,
    p1_begin != p2_begin.
```

- 编译并执行$ deepdive do transaction_candidate，生成候选实体对表

p1_id	p1_name	p2_id	p2_name
186_0_12_12	新研究院	186_0_2_2	银基烯碳新材料集团股份有限公司
942_0_10_10	浙江省铁路投资集团有限公司	942_0_8_8	浙江江山化工股份有限公司
3800_0_9_9	佳都新太科技股份有限公司	3800_0_19_19	广州新科佳都科技有限公司
965_1_27_28	液压集团	965_1_32_37	北京英纳超导技术有限公司
4558_1_49_55	德昌厚地稀土矿业有限公司	4558_1_79_82	成都市中级人民法院
9_0_7_7	深圳美丽生态股份有限公司	9_0_42_42	江苏八达园林有限责任公司
2049_0_28_28	控股子公司	2049_0_5_5	江西省富煌钢构有限公司
779_1_21_26	天宏瑞科硅材料有限责任公司	779_1_62_65	南京宝色股份公司
1974_0_5_5	尹洪卫	1974_0_3_3	岭南园林股份有限公司
2285_0_7_7	为沈阳三聚凯特催化剂有限公司	2285_0_45_45	中行

(10 rows)

工作流


数据准备
导入数据库
文本数据表
NLP处理
已知关系的实体对数据表
文本数据的NER/POS/词法依赖等序列
mention提取
匹配候选实体对中的已知关系
基于规则打标部分数据
文本中的mention
table: company_mention
deepdive do transaction_candidate
已知变量表
候选实体对
实体对特征提取
table: transaction_candidate
总变量表
特征表
构建学习因子图，得到变量置信度

特征抽取

- 抽取候选实体对的文本特征
- 在app.ddlog中定义特征表
- 定义特征抽取的函数extract_transaction_features
  - 输入为sentence的NLP结果，输出NLP组合的各种特征
  - 函数调用extract_transaction_features.py，脚本调用自带的ddlib库生成NLP的一系列窗口特征组合

```
@extraction
transaction_feature(
    p1_id   text,
    p2_id   text,
    feature text
).
```

```
function extract_transaction_features over (
        p1_id          text,  p2_id          text,
        p1_begin_index int,   p1_end_index   int,
        p2_begin_index int,   p2_end_index   int,
        doc_id         text,  sent_index     int,
        tokens         text[],lemmas         text[],
        pos_tags       text[],ner_tags    text[],
        dep_types      text[],dep_tokens     int[]
) returns rows like transaction_feature
implementation "udf/extract_transaction_features.py" handles
tsv lines.
```

特征抽取

- extract_transaction_features.py
  - 调用ddlib库，得到各种POS/NER/词序列的窗口特征，此处可以自定义特征
  - 每个实体对都要生成大量特征，这一步耗时比较久。

```
# Create a DDLIB sentence object, which is just a list of DDLIB Word objects
sent = []
for i,t in enumerate(tokens):
    sent.append(ddlib.Word(
        begin_char_offset=None,
        end_char_offset=None,
        word=t,
        lemma=lemmas[i],
        pos=pos_tags[i],
        ner=ner_tags[i],
        dep_par=dep_parents[i] - 1,  # Note that as stored from CoreNLP 0 is ROOT, but for DDLIB -
        dep_label=dep_types[i]))

# Create DDLIB Spans for the two person mentions
p1_span = ddlib.Span(begin_word_id=p1_begin_index, length=(p1_end_index-p1_begin_index+1))
p2_span = ddlib.Span(begin_word_id=p2_begin_index, length=(p2_end_index-p2_begin_index+1))

# Generate the generic features using DDLIB
for feature in ddlib.get_generic_features_relation(sent, p1_span, p2_span):
    yield [p1_id, p2_id, feature]
```

实体特征

关系特征

特征抽取

- extract_transaction_features.py
  - Ddlib自带的一些特征如下：

特征	数据库中的存储名
实体对间指定window的词语序列	NGRAM_[WINDOW]
实体对间完整的词语序列	WORD_SEQ_[ALL]
实体对间完整的NER序列	NER_SEQ_[ALL]
实体对间完整的POS序列	POS_SEQ_[ALL]
实体对左侧指定window的分离词语序列	W_LEFT_[WINDOW]
实体对右侧指定window的分离词语序列	W_RIGHT_[WINDOW]
实体对左右两侧window的混合词语序列	W_WORD_L_WINDOW_R_WINDOW[ALL]
是否存在词典中的特征词	IN_DICT_[WORD]
两个实体的语法依赖词语序列	BETW_L_[WORD]
两个实体的语法依赖POS序列	BETW_D_[WORD]

特征抽取

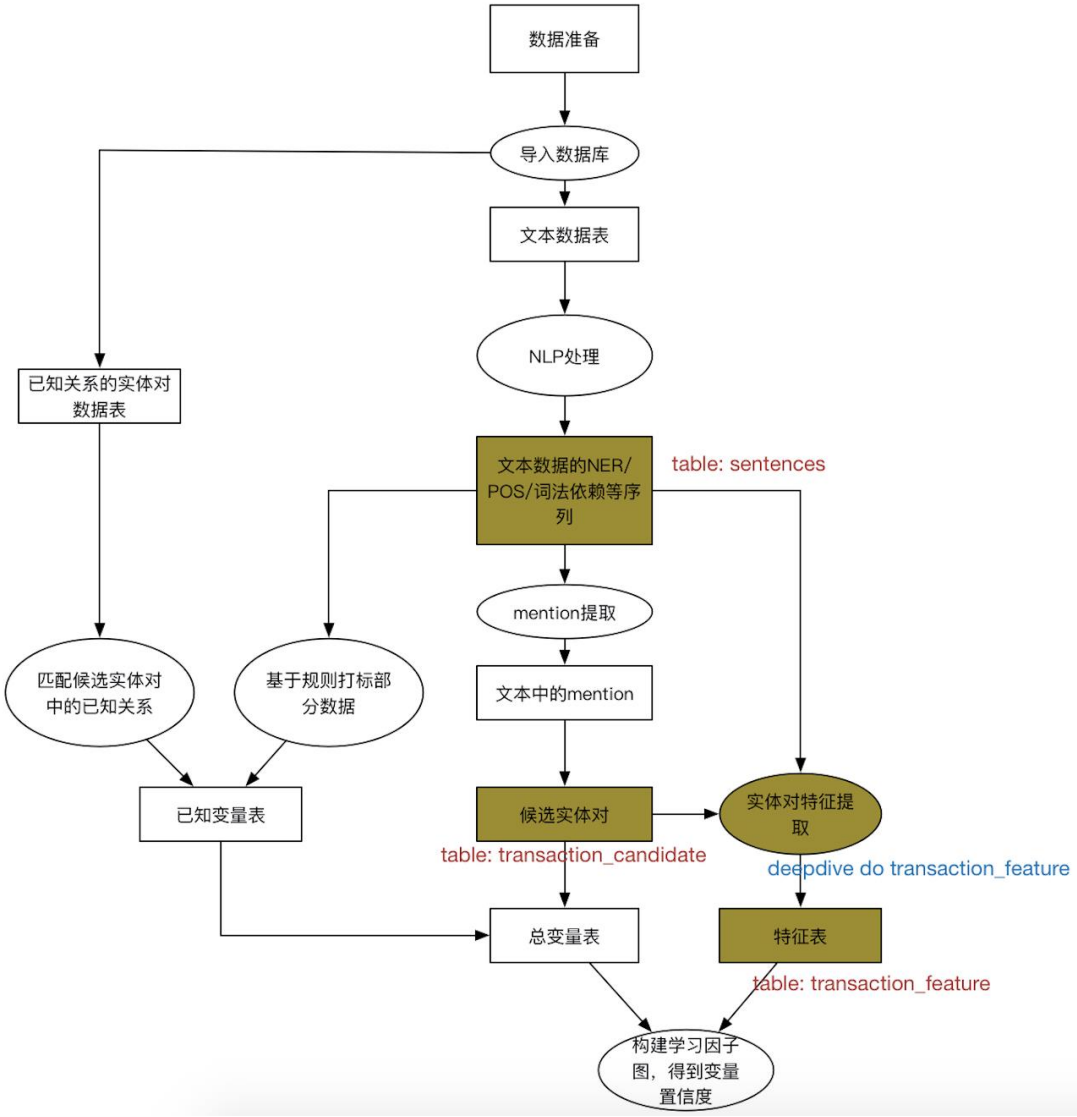
- Join sentences表和mention表作为输入，输出到transaction_feature

```
transaction_feature += extract_transaction_features(
    p1_id, p2_id, p1_begin_index, p1_end_index,
p2_begin_index, p2_end_index,
    doc_id, sent_index, tokens, lemmas, pos_tags, ner_tags,
dep_types, dep_tokens
) :-
company_mention(p1_id, _, doc_id, sent_index, p1_begin_index,
p1_end_index),
company_mention(p2_id, _, doc_id, sent_index, p2_begin_index,
p2_end_index),
sentences(doc_id, sent_index, _, tokens, lemmas, pos_tags,
ner_tags, _, dep_types, dep_tokens).
```

- 编译并执行$ deepdive do transaction_feature，生成特征表
(下图是部分窗口词特征)

```
         p1_id          |          p2_id          |          feature
------------------------+-------------------------+---------------------------
 1201746717_11_987_988 | 1201746717_11_1166_1167 | NGRAM_2_[公司 50]
 1201746717_11_987_988 | 1201746717_11_1166_1167 | NGRAM_3_[公司 50 徐洪林]
 1201746717_11_987_988 | 1201746717_11_1166_1167 | NGRAM_1_[50]
 1201746717_11_987_988 | 1201746717_11_1166_1167 | NGRAM_2_[50 徐洪林]
 1201746717_11_987_988 | 1201746717_11_1166_1167 | NGRAM_3_[50 徐洪林 现代]
 1201746717_11_987_988 | 1201746717_11_1166_1167 | NGRAM_1_[徐洪林]
 1201746717_11_987_988 | 1201746717_11_1166_1167 | NGRAM_2_[徐洪林 现代]
 1201746717_11_987_988 | 1201746717_11_1166_1167 | NGRAM_3_[徐洪林 现代 农业]
 1201746717_11_987_988 | 1201746717_11_1166_1167 | NGRAM_1_[现代]
 1201746717_11_987_988 | 1201746717_11_1166_1167 | NGRAM_2_[现代 农业]
(10 rows)
```

工作流


数据准备
导入数据库
文本数据表
NLP处理
已知关系的实体对数据表
文本数据的NER/POS/词法依赖等序列
table: sentences
mention提取
匹配候选实体对中的已知关系
基于规则打标部分数据
文本中的mention
已知变量表
候选实体对
实体对特征提取
table: transaction_candidate
deepdive do transaction_feature
总变量表
特征表
table: transaction_feature
构建学习因子图，得到变量置信度

样本打标

- 从候选实体对中标出部分正负例
  - 利用已知的实体对和候选实体对关联
  - 利用规则打部分正负标签
- 在app.ddlog中定义标签表

```
@extraction
transaction_label(
    p1_id   text,
    p2_id   text,
    label   int,
    rule_id text
).
```

- 导入所有的候选实体对，初始标签均为0

```
transaction_label(p1,p2, 0, NULL) :- transaction_candidate(p1,
_, p2, _).
```

- 将db数据与候选实体对关联，关联到的权重标注为+3，规则标记为从知识库得到

```
transaction_label(p1,p2, 3, "pos_from_dbdata") :-
    transaction_candidate(p1, p1_name, p2, p2_name),
pos_transaction(n1, n2),
    [ lower(n1) = lower(p1_name), lower(n2) = lower(p2_name) ;
      lower(n2) = lower(p1_name), lower(n1) = lower(p2_name)
].
```

样本打标

- 通过规则再标注一部分实体，输入候选实体对的关联文本，进行打标

```
function supervise over (
        p1_id text, p1_begin int, p1_end int,
        p2_id text, p2_begin int, p2_end int,
        doc_id text, sentence_index int, sentence_text  text,
        tokens text[], lemmas  text[], pos_tags  text[],
        ner_tags text[], dep_types text[], dep_tokens int[]
    ) returns rows like transaction_label
    implementation "udf/supervise_transaction.py" handles tsv lines.
```

- 将规则抽取的标签也加入到transaction_label中

```
transaction_label += supervise(
    p1_id, p1_begin, p1_end,
    p2_id, p2_begin, p2_end,
    doc_id, sentence_index, sentence_text,
    tokens, lemmas, pos_tags, ner_tags, dep_types, dep_token_indexes
) :-
    transaction_candidate(p1_id, _, p2_id, _),
    company_mention(p1_id, p1_text, doc_id, sentence_index, p1_begin,
p1_end),
    company_mention(p2_id, p2_text,      _,             _, p2_begin,
p2_end),
    sentences( doc_id, sentence_index, sentence_text,
tokens, lemmas, pos_tags, ner_tags, _, dep_types,   dep_token_indexes).
```

样本打标

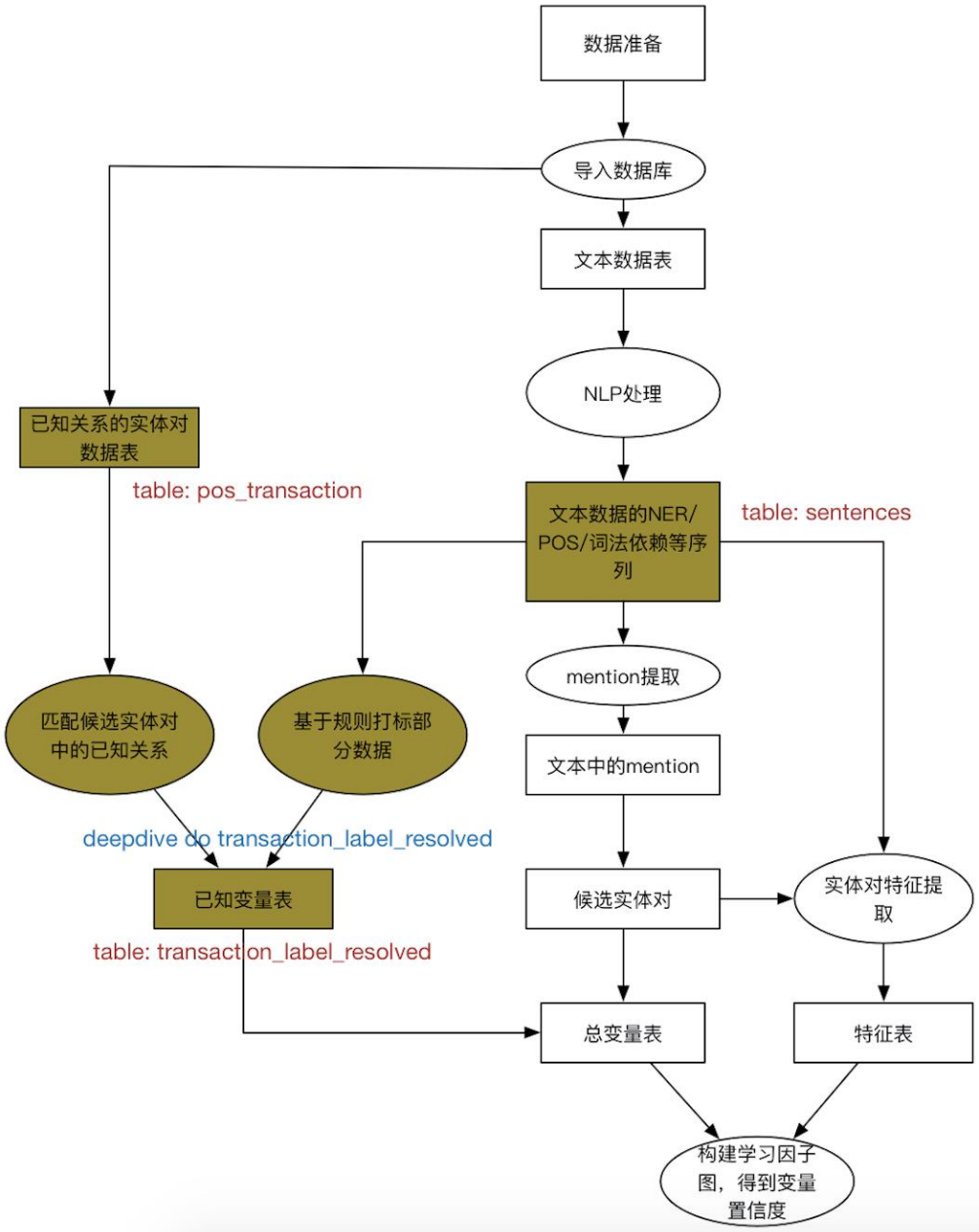
- 规则在supervise_transaction.py中定义：

```
if len(intermediate_lemmas) > MAX_DIST:
    yield transaction._replace(label=-1, type='neg:far_apart')
if 'ORG' in intermediate_ner_tags:
    yield transaction._replace(label=-1, type='neg:third_company_between')
if len(TRANSLATION_COM.intersection(intermediate_lemmas)) > 0:
    yield transaction._replace(label=1, type='pos:A持股B')
if len(TRANSLATION.intersection(intermediate_lemmas)) > 0 and len(STOCK.intersection(tail_lemmas)
    yield transaction._replace(label=1, type='pos:A购买B股权')
if len(TRANSLATION_AFTER.intersection(intermediate_lemmas)) > 0:
    yield transaction._replace(label=1, type='pos:A增资B')
if len(OTHERS_COM.intersection(intermediate_lemmas)) > 0 and len(OTHERS_AFTER.intersection(tail_l
    yield transaction._replace(label=-1, type='neg:A购买B的产品')
if len(CONF.intersection(intermediate_lemmas)) > 0 and len(OTHERS.intersection(tail_lemmas)) > 0:
    yield transaction._replace(label=-1, type='neg:A向B提供担保')
if ("与" in intermediate_lemmas) and ("签署股权转让协议" in tail_lemmas):
    yield transaction._replace(label=1, type='pos:A和B签署股权转让协议')
if len(COMMAS.intersection(intermediate_lemmas)) > 0:
    yield transaction._replace(label=-1, type='neg:中间有特殊符号')
```

- 最后，在多条规则和知识库标记的结果中为每对实体做vote，执行deepdive do transaction_label_resolved生成最终标签。

```
transaction_label_resolved(p1_id, p2_id, SUM(vote)) :-
transaction_label(p1_id, p2_id, vote, rule_id).
```

工作流



因子图构建

- 定义最终存储的表格，「?」表示此表是用户模式下需要推导label的最终表。

```
@extraction
has_transaction?(
    p1_id text,
    p2_id text
).
```

- 定义一系列推导关系，构建因子图

```
@weight(f)
has_transaction(p1_id, p2_id) :-
    transaction_candidate(p1_id, _, p2_id, _),
    transaction_feature(p1_id, p2_id, f).
@weight(3.0)
has_transaction(p1_id, p2_id) => has_transaction(p2_id, p1_id) :-
transaction_candidate(p1_id, _, p2_id, _).
```

- 根据打标的结果，灌入已知的变量

```
has_transaction(p1_id, p2_id) = if l > 0 then TRUE
                    else if l < 0 then FALSE
                    else NULL end :-
transaction_label_resolved(p1_id, p2_id, l).
```

因子图定义

R_1: has_transaction

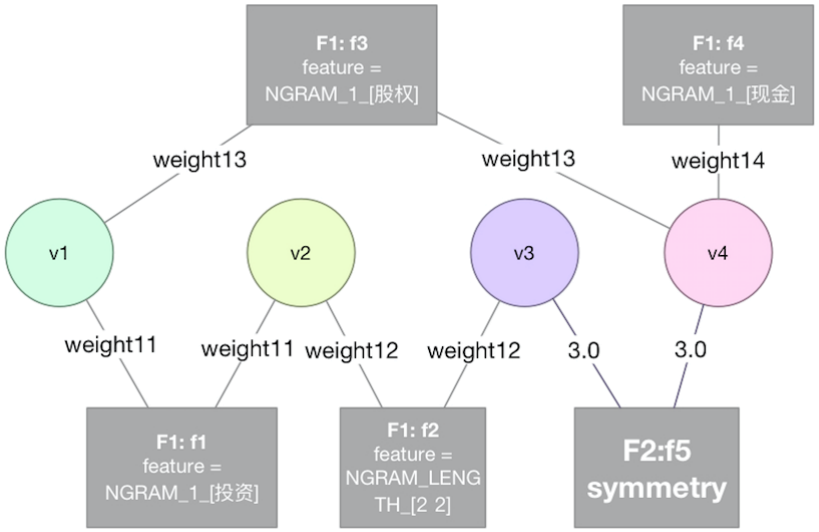
vid	P1_id	P2_id	label
v_1	1263_0_9_9	1263_0_7_7	NULL
v_2	3316_0_5_6	3316_0_10_11	-1
v_3	3315_0_20_21	3315_0_17_18	1
v_4	3315_0_17_18	3315_0_20_21	NULL

transaction_feature

P1_id	P2_id	feature
1263_0_9_9	1263_0_7_7	NGRAM_1_[投资]
1263_0_9_9	1263_0_7_7	NGRAM_1_[股权]
3316_0_5_6	3316_0_10_11	NGRAM_1_[投资]
3316_0_5_6	3316_0_10_11	NGRAM_LENGTH_[2 2]
3315_0_20_21	3315_0_17_18	NGRAM_LENGTH_[2 2]
3315_0_17_18	3315_0_20_21	NGRAM_1_[股权]
3315_0_17_18	3315_0_20_21	NGRAM_1_[现金]

- 用户模式 ($R_i \in R$)
  - 用户定义的变量元组$v_i \in V$
  - 变量映射到二值$\sigma(v) \in \{0, 1\}$
  - Possible world ($I_\sigma : V \rightarrow B$)
    - v_1->0, v_2->1, v_3->1, v_4->1 (a)
    - v_1->1, v_2->0, v_3->1, v_4->1 (b)
  - 先验变量：$P \subseteq V$，$N \subseteq V$
    - 已知取值固定的
    - $v_3 \in P$, $v_2 \in N$
  - τ：所有possible world的集合
    - τ_e：和先验变量取值一致，如(b)

因子图定义



F_1： feature

fid	vars	weight
f1	(v1, v2)	func(N_GRAM_1['投资'])
f2	(v2, v3)	func(N_GRAM_LENGTH_[2 2])
f3	(v1,v4)	func(N_GRAM_1['股权'])
f4	v4	func(N_GRAM_1['现金'])

- 定义I_σ的总的聚合概率

- 对于给定的 fid $\in F_j$和possible world I_σ
  - 该fid在当前possible world上的聚合值

$$g_j(\text{fid}, I_\sigma) = weight_{\text{fid}} * aggr(vars_{\text{fid}})$$

- 定义I_σ的总的聚合概率

$$Z(I_\sigma) = \exp\{\sum_{F_j \in F} \sum_{fid \in F_j} g_j(fid, I_\sigma)\}$$

- 变量的边缘概率

$$\Pr[v_i] = \frac{\sum_{I \in \mathcal{I}_e^+} Z(I)}{\sum_{I \in \mathcal{I}_e^+} Z(I) + \sum_{I \in \mathcal{I}_e^-} Z(I)}$$

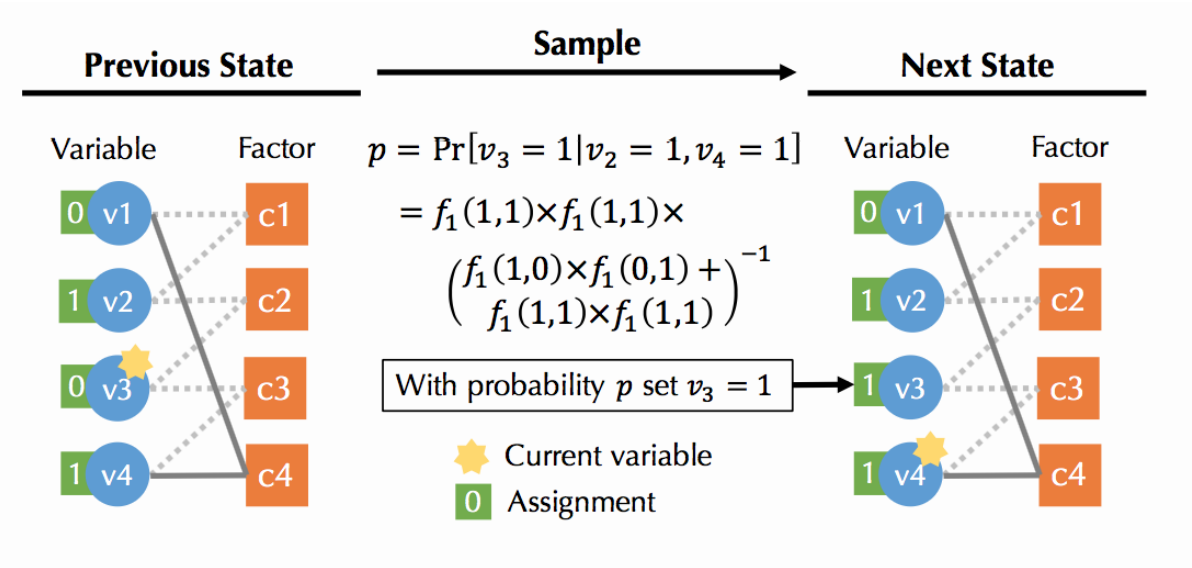
F_2： symmetry

fid	vars	weight
f5	(v3, v4)	3.0

```
@weight(3.0)
has_transaction(p1_id, p2_id) => has_transaction(p2_id, p1_id) :-
transaction_candidate(p1_id, _, p2_id, _).
```

吉布斯采样

- 采样得到符合内在条件概率的possible world集合
  - 先随机一个possible world I_0
  - 根据每个变量的相关变量，依次更新每个变量v的边缘概率
  - 得到新的possible world I_1，再循环
- 不共享factor的变量独立sample



权重学习

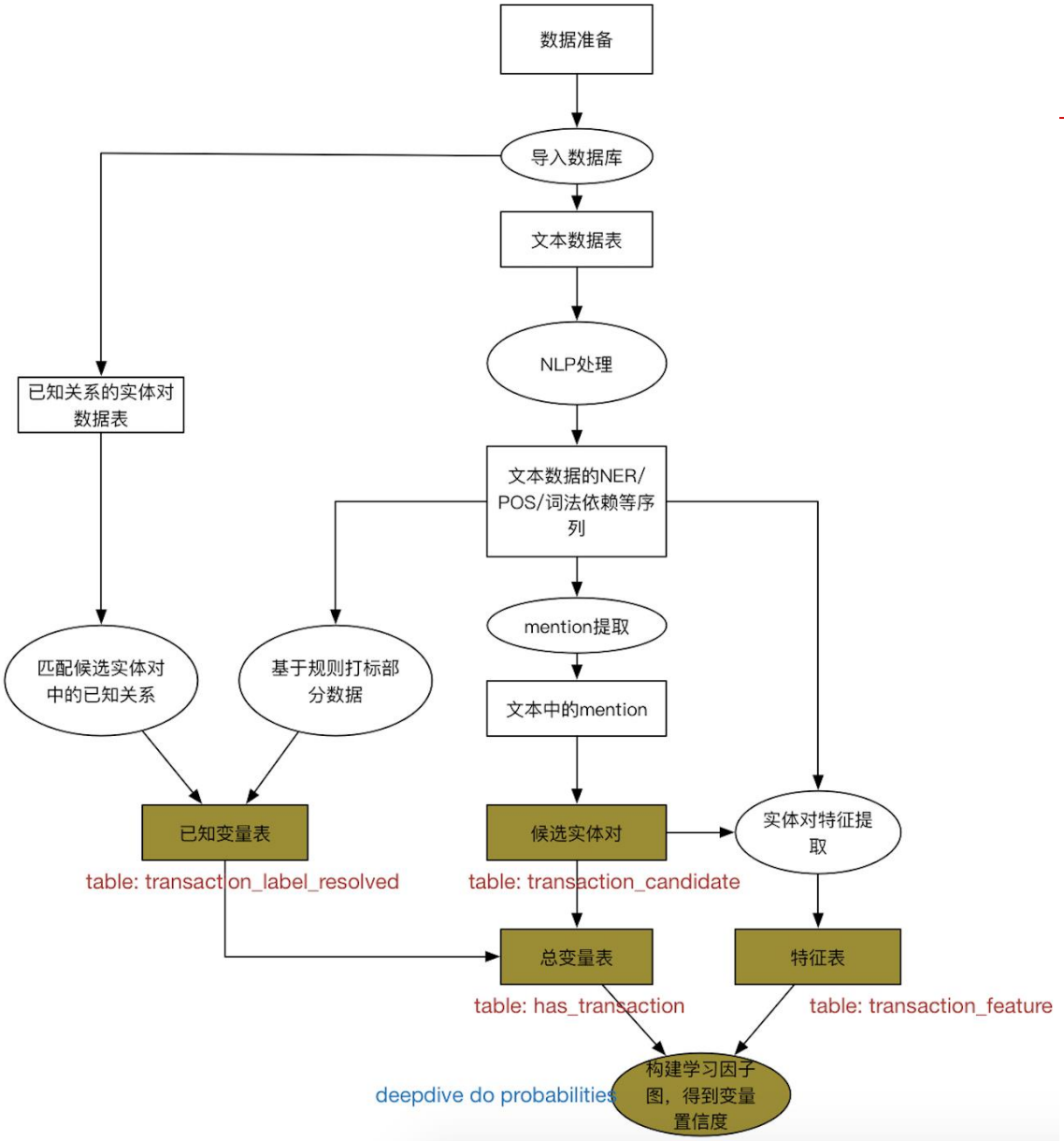
- 最大化和先验变量取值一致的possible world的数目

$$\arg\max_{w_j} \frac{\sum_{I \in \mathcal{I}_e} Z(I)}{\sum_{I \in \mathcal{I}} Z(I)}$$

- 在采样得到的样本上随机梯度下降
- deepdive针对硬件做了优化，支持分布式、增量式训练

- 执行deepdive do probablities，开始训练。

工作流


数据准备
导入数据库
文本数据表
NLP处理
已知关系的实体对数据表
文本数据的NER/POS/词法依赖等序列
mention提取
匹配候选实体对中的已知关系
基于规则打标部分数据
文本中的mention
已知变量表
候选实体对
实体对特征提取
table: transaction_label_resolved
table: transaction_candidate
总变量表
特征表
table: has_transaction
table: transaction_feature
deepdive do probabilities
构建学习因子图，得到变量置信度

数据表

```
                                List of relations
 Schema |                                Name                                |
--------+--------------------------------------------------------------------+
 public | articles                                                           |
 public | company_mention                                                    |
 public | dd_factors_inf_imply_has_transaction_has_transaction               |
 public | dd_factors_inf_istrue_has_transaction                              |
 public | dd_graph_variables_holdout                                         |
 public | dd_graph_variables_observation                                     |
 public | dd_graph_weights                                                   |
 public | dd_inference_result_variables                                      |
 public | dd_inference_result_variables_mapped_weights                       |
 public | dd_inference_result_weights                                        |
 public | dd_inference_result_weights_mapping                                |
 public | dd_weights_inf_imply_has_transaction_has_transaction               |
 public | dd_weights_inf_istrue_has_transaction                              |
 public | has_transaction                                                    |
 public | has_transaction_label_inference                                    |
 public | num_company                                                        |
 public | pos_transaction                                                    |
 public | sentences                                                          |
 public | transaction_candidate                                              |
 public | transaction_feature                                                |
 public | transaction_label                                                  |
 public | transaction_label__0                                               |
 public | transaction_label_resolved                                         |
(23 rows)
```

系统底层表

置信度结果表

其他配置文件

- db.url

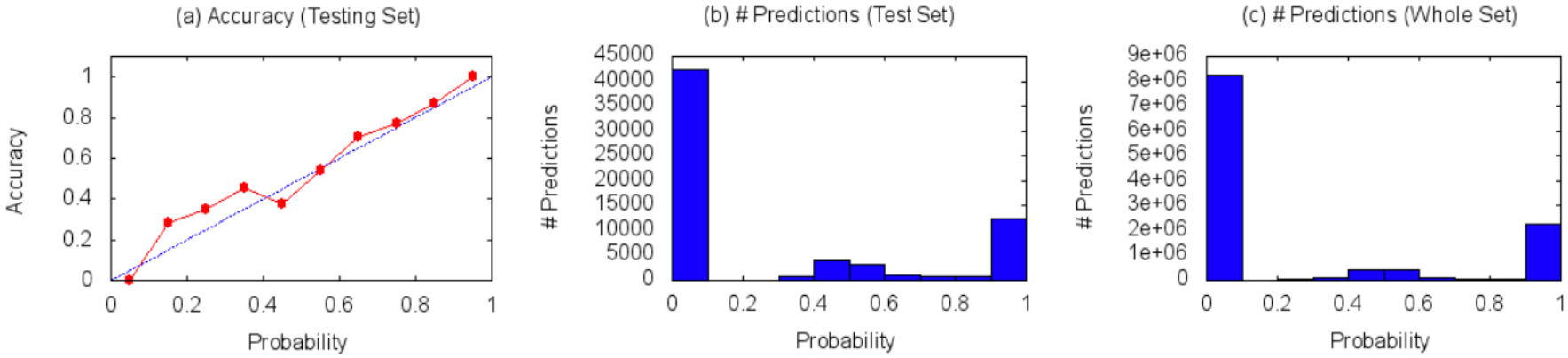
```
postgresql://wgy@localhost:5432/transaction_example
```

- deepdive.conf

```
deepdive.calibration.holdout_fraction:0.25
deepdive.sampler.sampler_args: "-l 1000 -s 1 -i 1000 --alpha 0.01 --sample_evidence"
```

迭代调试

- $ deepdive do calibration_plots
- 系统自动在run/model/calibration_plots下生成此图



(a) 横轴表示模型输出的分数，纵轴是该分数段的正例率。越趋近蓝色标准线，模型效果越好。

(b) 测试集上置信度的分布，越靠近横轴两端，说明模型区分度越好。

(c) 全集上置信度的分布。

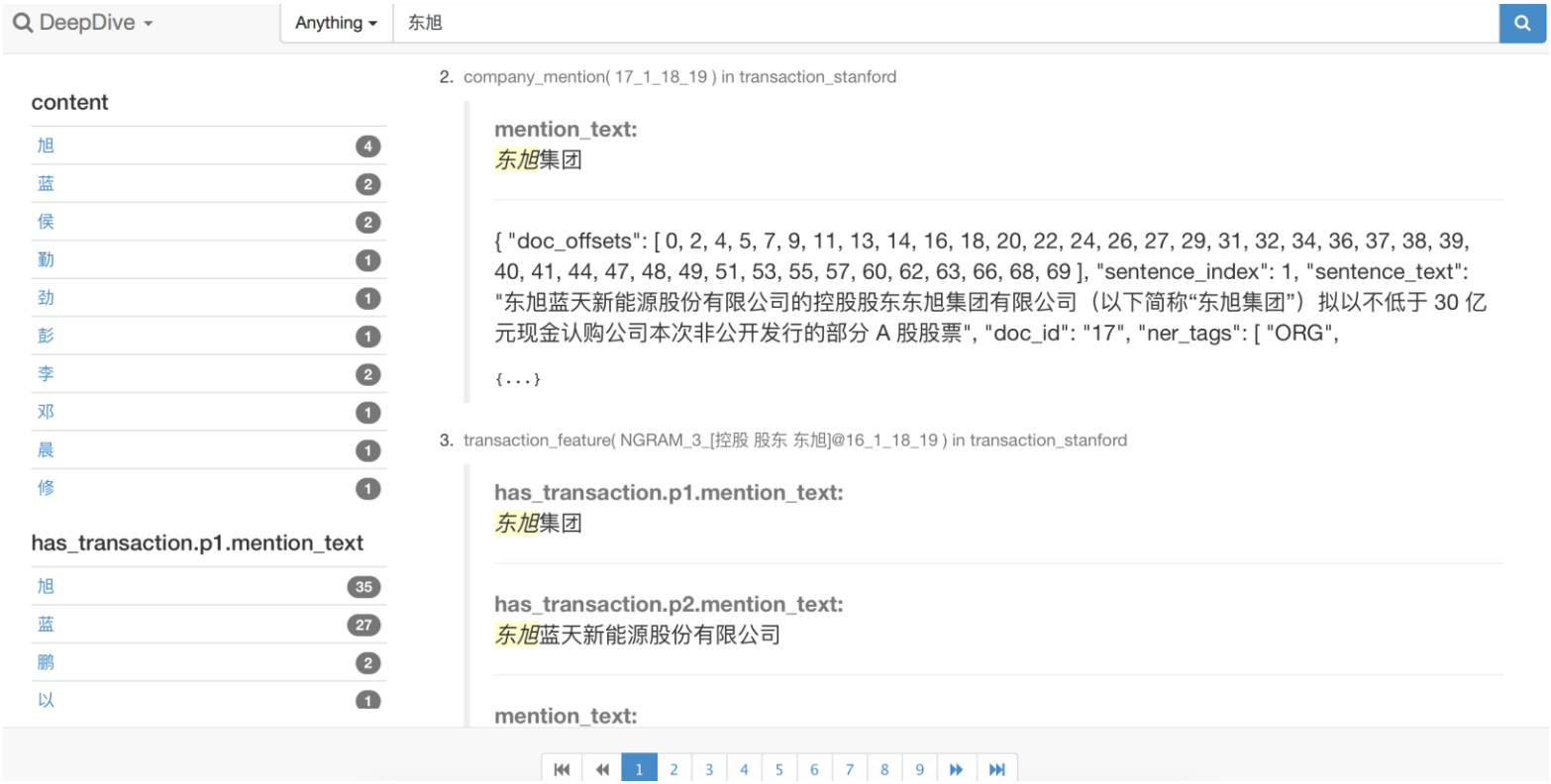
迭代调试

- $ mindbender tagger labeling/*/mindtagger.conf
  - Mindtagger是deepdive提供的一套可视化工具，可参考 http://deepdive.stanford.edu/labeling 搭建环境
  - 执行完毕后，会开启web服务，默认通过http://localhost:8000访问

- 可以通过特征的weight，分析置信度的误差，调整先验数据等

迭代调试

- $ mindbender search update && mindbender search gui
  - 直接执行，默认通过http://localhost:8000访问



- 可以通过可视搜索功能，方便定位要找的实体/特征等

总结

- 模块化、便于更改替换
- NLP影响较大，可以考虑尝试其他端到端模型
- 便于分析和迭代开发

下载链接：http://www.openkg.cn/tool/cn-deepdive

开放域关系抽取

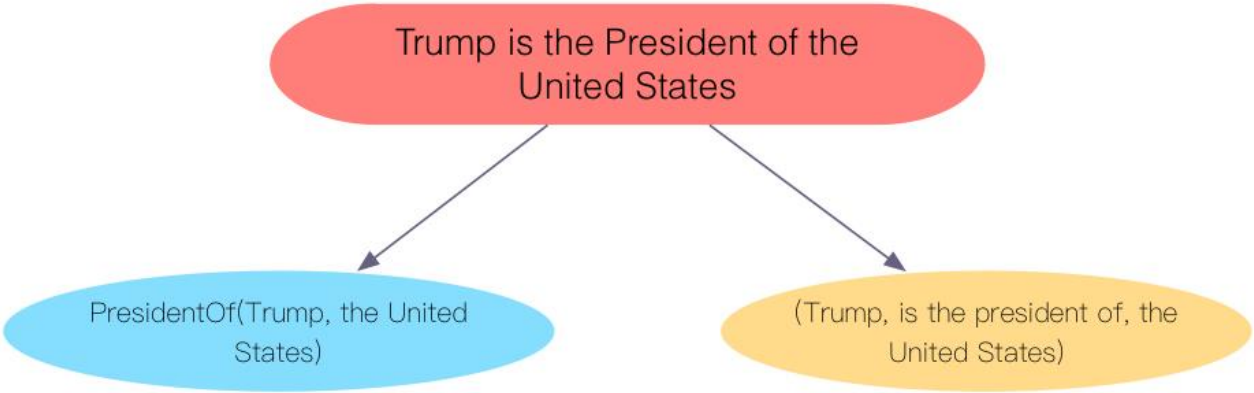
IE的发展趋势

传统IE

- 针对特定领域
- 事先预定义关系类型
- 依赖于领域专业知识
- 规模小但精度高

开放域 IE

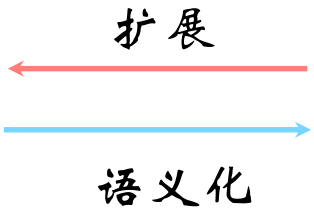
- 通用模型
- 不需要预定义关系
- 依赖于句法特征
- 全网规模但精度低



主要系统

传统 IE

- FREEBASE
- DBpedia
- YAGO
- Deepdive



Open IE

- TextRunner
- WOE
- REVERB
- OLLIE
- ClauseIE

传统IE和OpenIE互相补充

可以按当前知识库的规范数据，链接更多网络数据

OpenIE得到的三元组可以用扩充知识库

第一代OpenIE系统

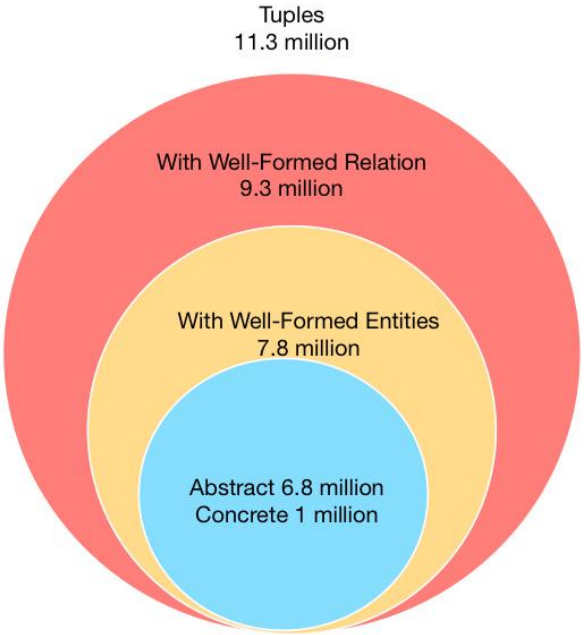
TextRunner

- 抽取特征

  NER, POS, Dependency Parsing

- 学习模型

  Naive Bayes and CRF



scale of TextRunner

WOE

把核心语法路径也作为一个关系

Albert Einstein ——> was awarded <—— the Nobel Prize

nsubjpass　　dobj

(Albert Einstein, was awarded, the Nobel Prize)

第一代OpenIE系统

面临的挑战

- 关系不一致、不准确

E.g. Peter thought that John began his career as a scientist

True: (John, began, his career as a scientist)

False: (Peter, began, his career as a scientist)

- 提取的关系不包含有效信息

E.g. Al-Qaeda claimed responsibility for the 9/11 attacks

True: (AI-Qaeda, claimed responsibility, for the 9/11 attacks)

False: (Al-Qaeda, claimed, responsibility)

第二代OpenIE系统

更深入研究了句子的语法特性

Reverb

http://www.openkg.cn/tool/reverb

- 基于动词的关系抽取

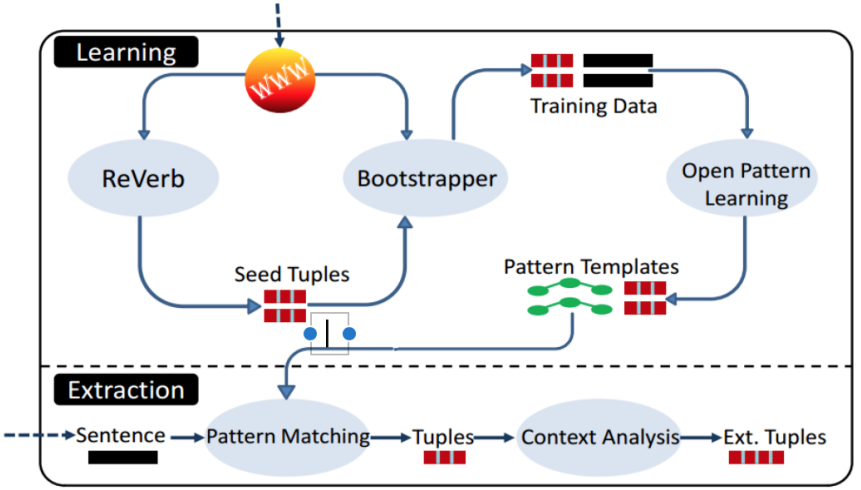
  围绕动词词组抽取以下关系

  V | VP | VW* P

  V = verb particle? adv?

  W = (noun | adj | adv | pron | det)

  P = (prep | particle | inf. marker)



OLLIE 流程图

OLLIE

http://www.openkg.cn/tool/ollie

- 增加抽取了名词和形容词中包含的语义信息

  E.g. Microsoft co-founder Bill Gates spoke at…

  OLLIE 可以抽取(Bill Gates, be co-founder of, Microsoft)， Reverb不可以

- 把Reverb中抽取的关系用作种子，学习更多的模板

第二代OpenIE系统

ClauseIE

基于子句的抽取

将句子拆分成各个从句，定义从句类型

用语法规则和句法依赖判断从句类型 (Decision Tree)

抽取从句集合 → 识别从句类型 → 抽取关系

Table 1: Patterns and clause types (based on [15]).

	Pattern	Clause type	Example	Derived clauses
			Basic patterns	
S_1:	SV_i	SV	AE died.	(AE, died)
S_2:	SV_eA	SVA	AE remained in Princeton.	(AE, remained, in Princeton)
S_3:	SV_cC	SVC	AE is smart.	(AE, is, smart)
S_4:	$SV_{mt}O$	SVO	AE has won the Nobel Prize.	(AE, has won, the Nobel Prize)
S_5:	$SV_{dt}O_iO$	SVOO	RSAS gave AE the Nobel Prize.	(RSAS, gave, AE, the Nobel Prize)
S_6:	$SV_{ct}OA$	SVOA	The doorman showed AE to his office.	(The doorman, showed, AE, to his office)
S_7:	$SV_{ct}OC$	SVOC	AE declared the meeting open.	(AE, declared, the meeting, open)

S: Subject, V: Verb, C: Complement, O: Direct object, O_i: Indirect object, A: Adverbial, V_i: Intransitive verb, V_c: Copular verb, V_c: Extended-copular verb, V_{mt}: Monotransitive verb, V_{dt}: Ditransitive verb, V_{ct}: Complex-transitive verb

更多进展

模型

- 联合训练

  训练一个统一模型，同时抽取实体和关系

- 模板匹配 + 深度学习
- 矩阵因式分解等所有好用的分类器

源数据

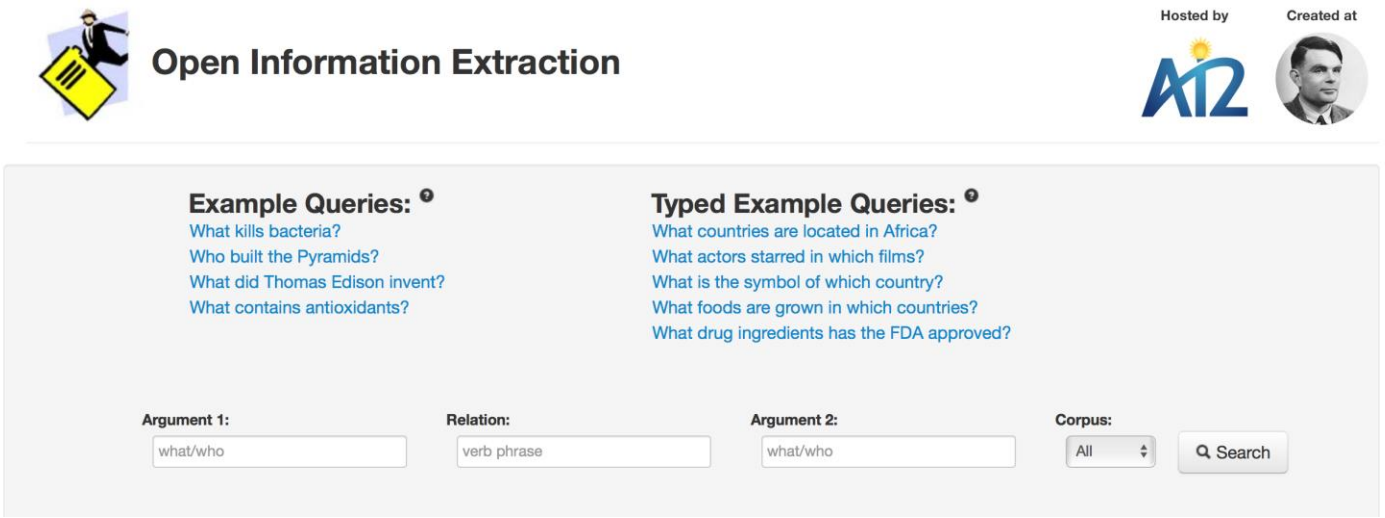
- 结构化的知识库

  可以依赖知识库进行更好的链接和特征抽取

OpenIE的应用

直接回答问题

可以回答不同用户提出的不同领域中形如 (A1, ?, A2) 的问题



用作其他NLP任务的特征

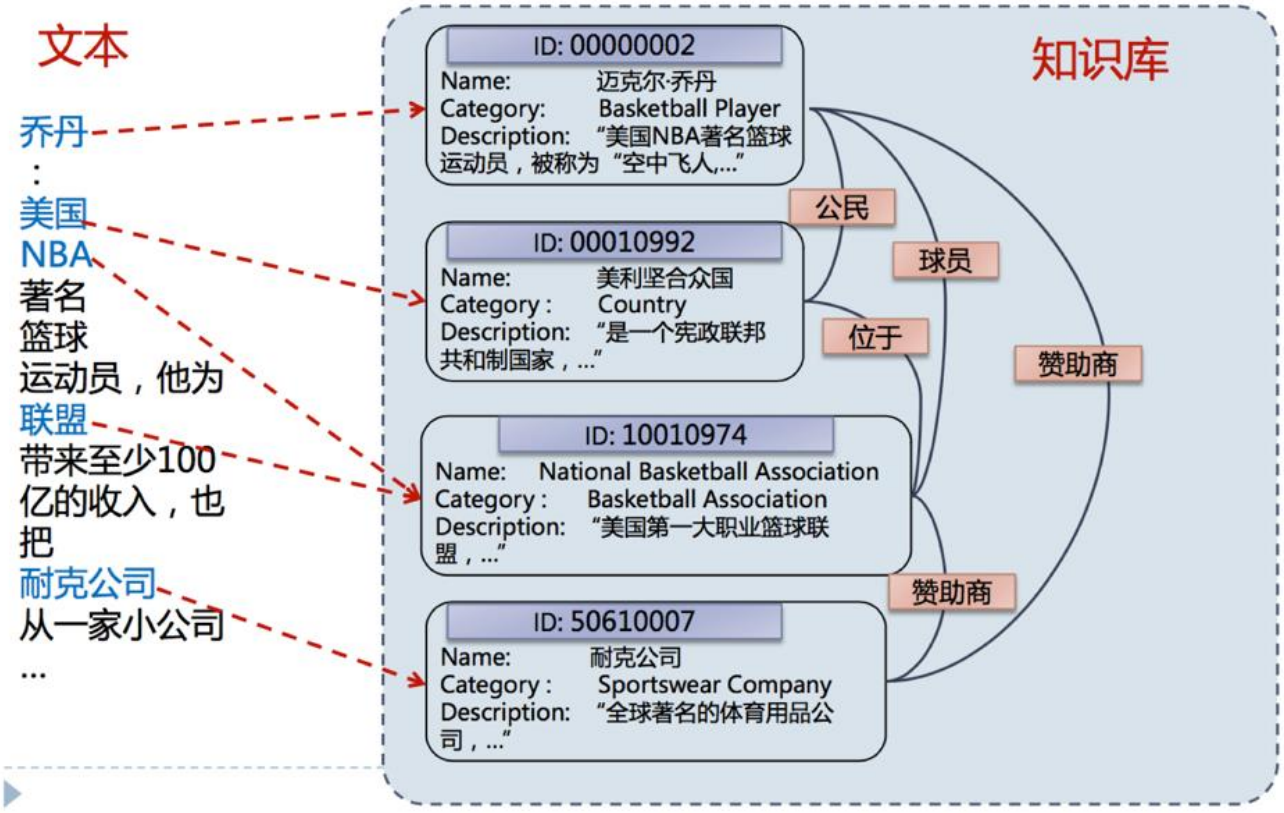
- 文本理解
- 相似度比较
- …

第二部分:知识挖掘

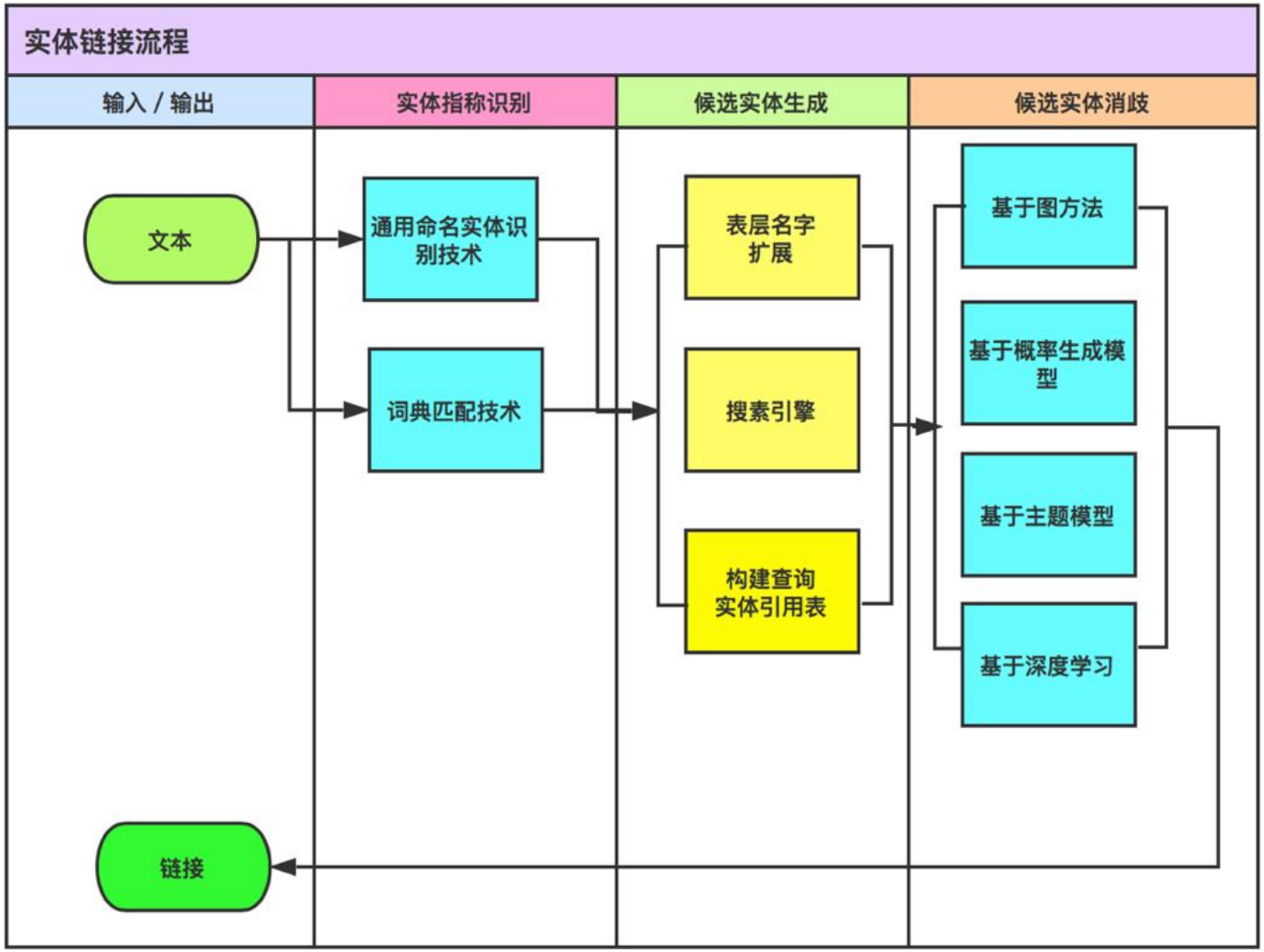
实体消歧与链接

实体链接简介

- 给定一篇文本中的实体指称(mention),确定这些指称在给定知识库中的目标实体 (entity)



实体链接基本流程


实体链接流程
输入 / 输出
实体指称识别
候选实体生成
候选实体消歧
文本
通用命名实体识别技术
词典匹配技术
表层名字扩展
搜素引擎
构建查询实体引用表
基于图方法
基于概率生成模型
基于主题模型
基于深度学习
链接

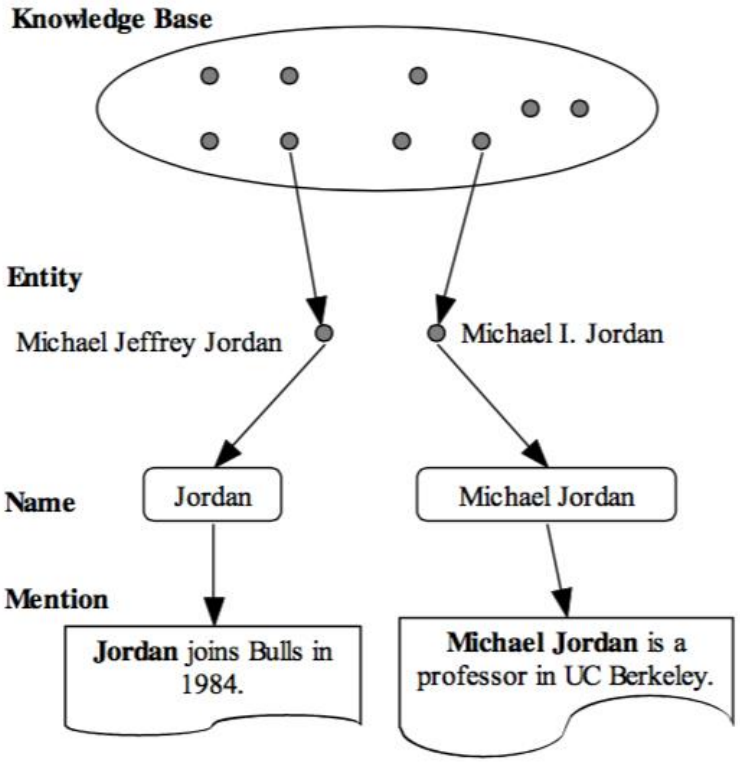
示例一

百科型知识库，适用于长、短文本场景

The entity-mention model:
a generative probabilistic model

P(m,e)=P(s,c,e)= P(e) P(sle) P(cle)

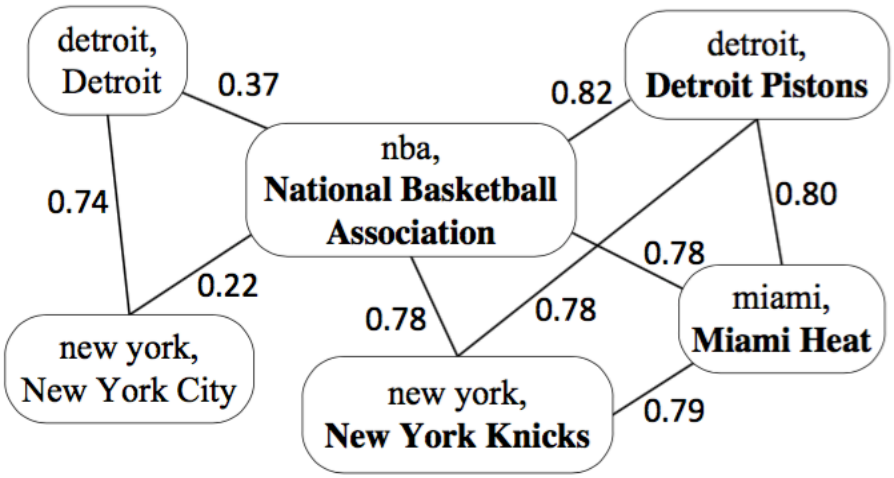
$$e = \arg\max_{e} \frac{P(m,e)}{P(m)} = \arg\max_{e} P(e)P(s\,|\,e)P(c\,|\,e)$$



示例二

百科型知识库，适用于长文本场景

构建实体关联图



实体关联图由3个部分组成：

(1) 每个顶点Vi=<mi, ei>

由mention-entity对构成

(2) 每个顶点得分：代表实体指称mi的目标实体为ei概率可能性大小

(3) 每条边的权重：代表语义关系计算值，表明顶点Vi和Vj的关联程度

示例二(续)

顶点的得分初始化方法：

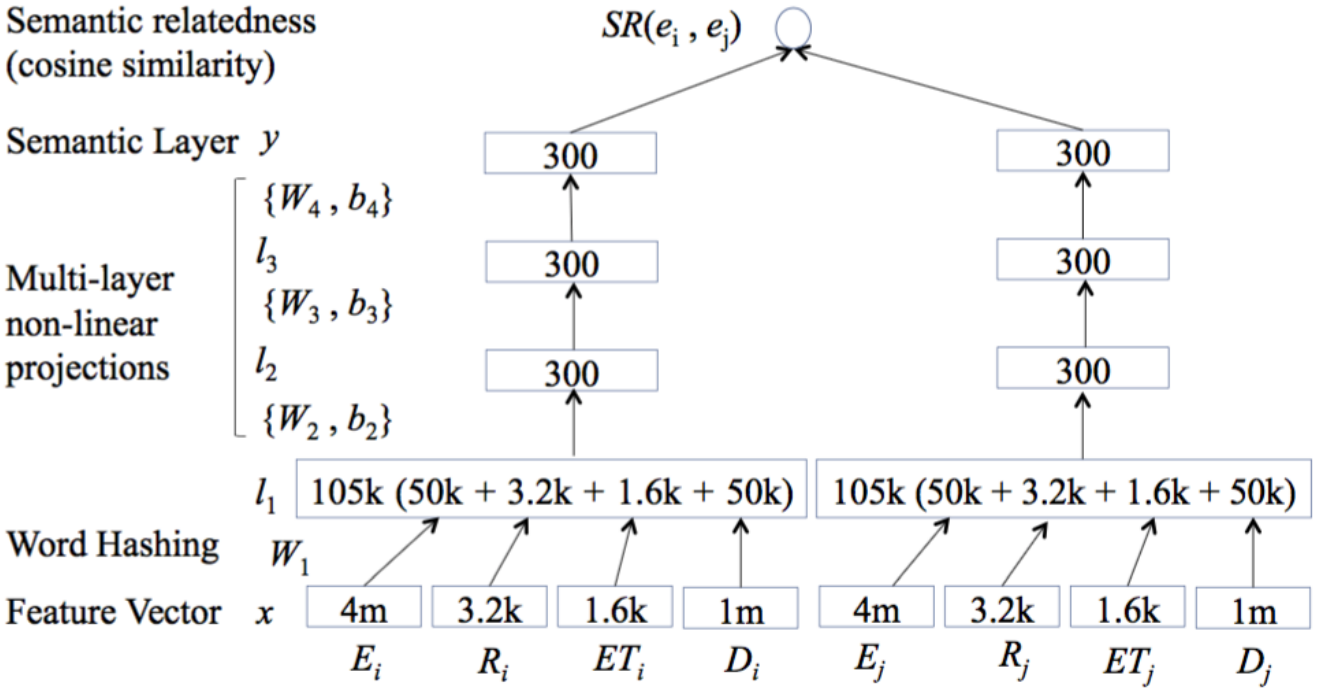
(1) 若顶点V实体不存在歧义，则顶点得分设置为1；

(2) 若顶点中mention和entity满足p(e|m)>=0.95，则顶点得分也设置为1。

(3) 其余顶点的得分设置为p(e|m)。

示例二（续）

边权初始化方法：深度语义关系模型

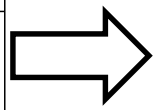


示例二（续）

基于图的标签传播算法

(1) 构造相似矩阵
(2) 迭代传播直到收敛算法结束。

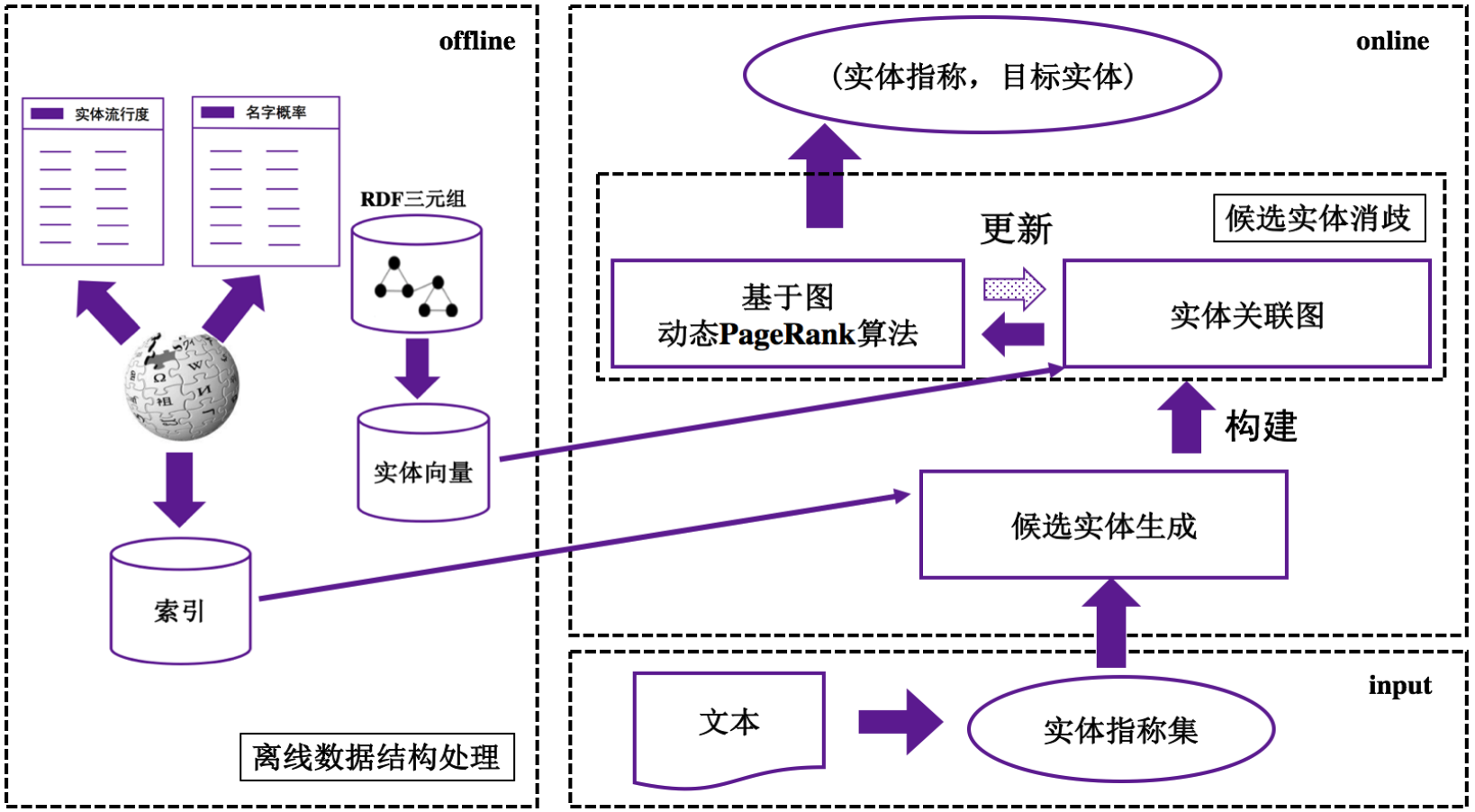
Detroit　Detroit	unlabeled
Detroit_Pistons Detroit	unlabeled
National_Basketball_Association nba	**labeled**
New_York_City new york	unlabeled
New_York_Knicks new york	unlabeled
Miami_Heat miami	**labeled**



Detroit　Detroit	unlabeled
Detroit_Pistons Detroit	**labeled**
National_Basketball_Association nba	**labeled**
New_York_City new york	unlabeled
New_York_Knicks new york	**labeled**
Miami_Heat miami	**labeled**

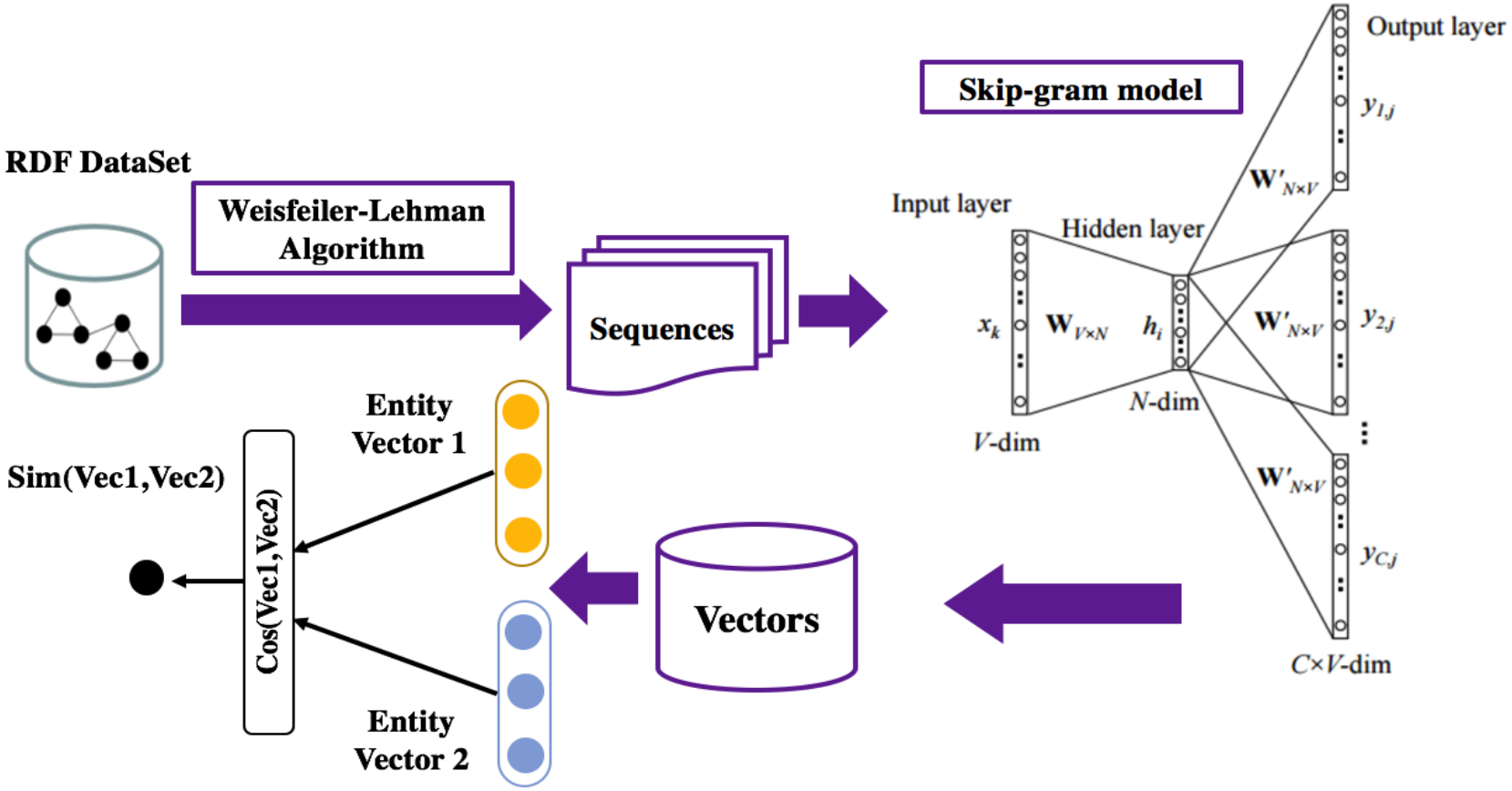
示例三

百科型知识库，适用于长文本场景



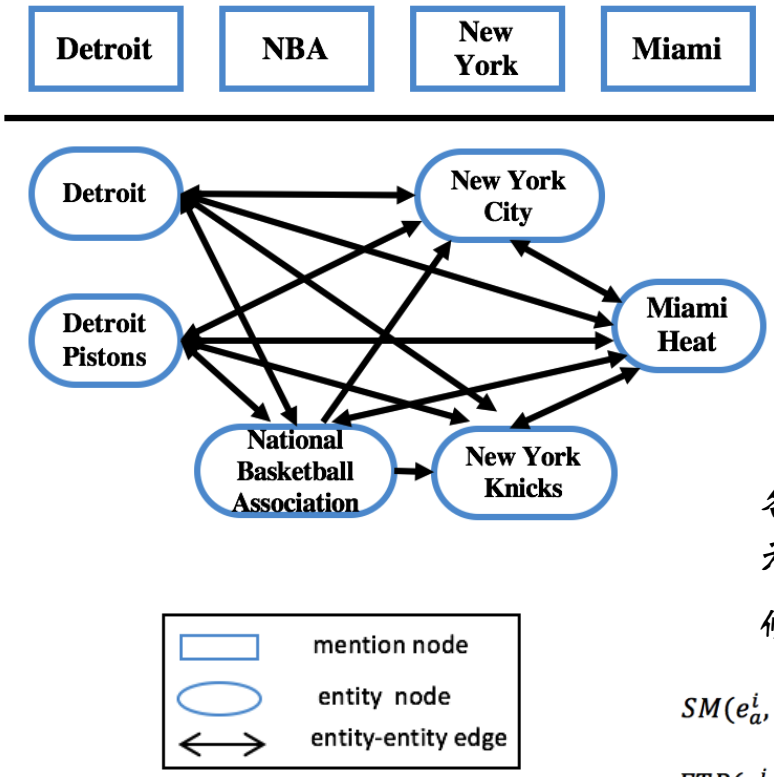
示例三（续）

候选实体间语义相似度计算



示例三（续）

构建实体关联图



实体关联图由四个部分组成：

(1) 实体指称节点

(2) 候选实体节点

(3) 候选实体节点顶点值：代表该候选实体是实体指称的目标实体概率大小

(4) 候选实体节点边权值：代表两个候选实体间的转化概率大小

各候选实体节点顶点值：初始化为均等，之后每轮更新为上一轮的PageRank得分

候选实体节点边权值计算公式如下：

$$SM(e_a^i, e_b^j) = \cos\left(v(e_a^i), v(e_b^j)\right)$$ → 两个候选实体间的相似度大小

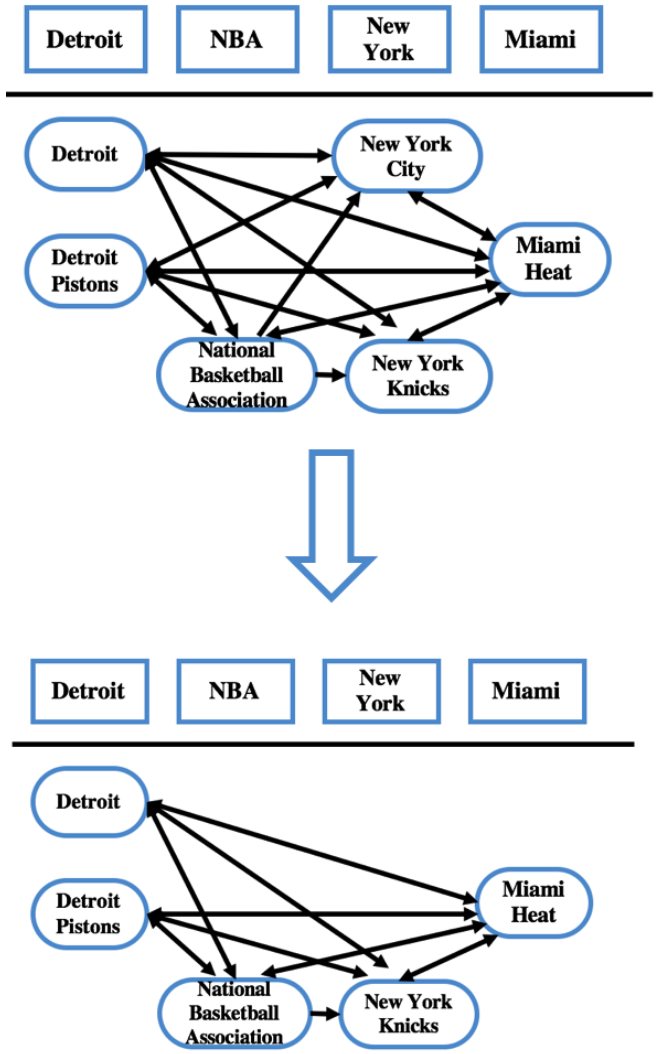
$$ETP(e_a^i, e_b^j) = \frac{SM(e_a^i, e_b^j)}{\sum_{k \epsilon (V \setminus V_i)} SM(e_a^i, k)}$$ → 两个候选实体间的转化概率

示例三（续）

更新实体关联图

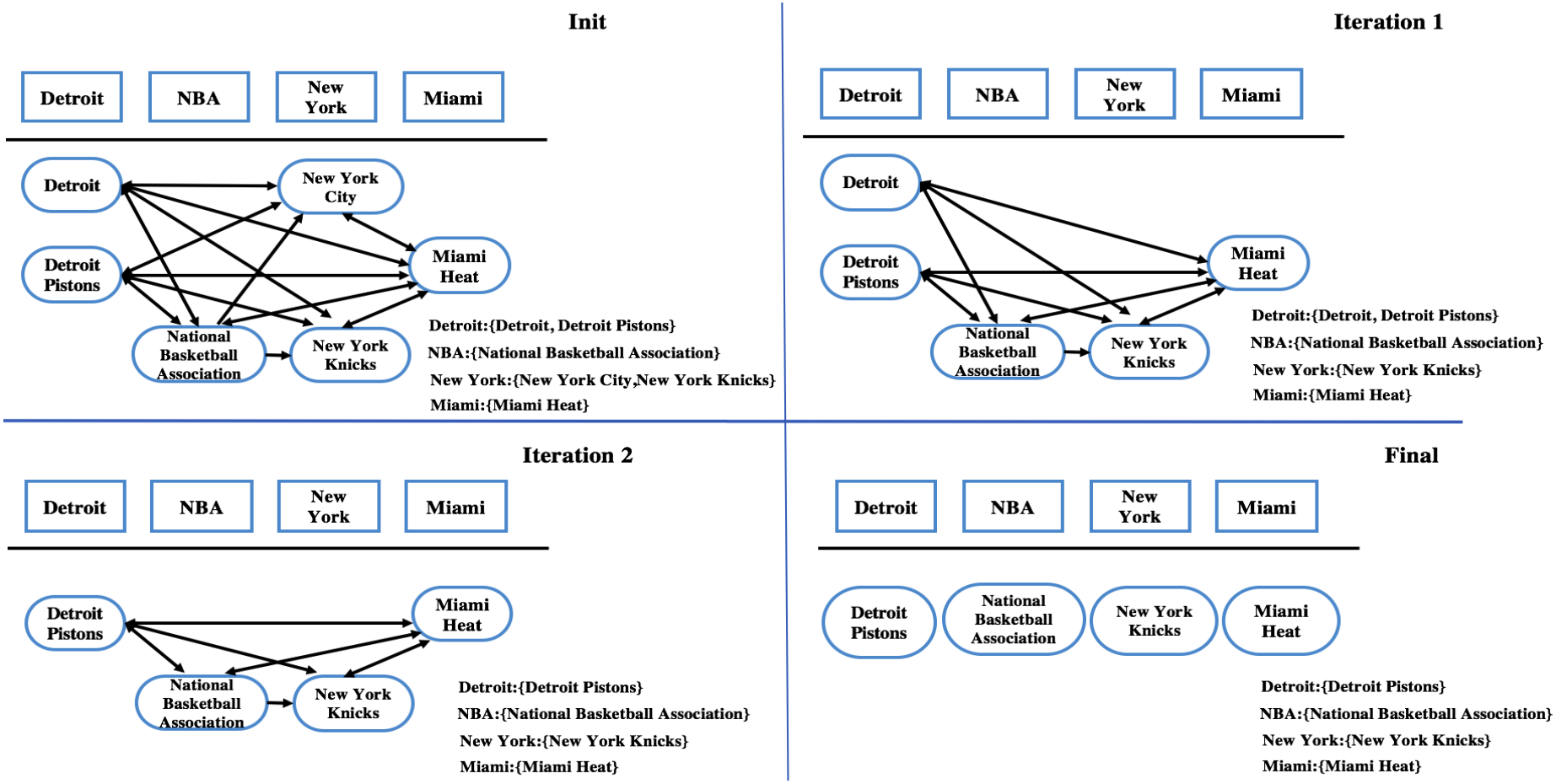
节点代表实体	节点 PageRank 得分
Detroit	0.44314869
Detroit Pistons	0.77259475
Nation Basketball Association	0.85422741
New York City	0.36443149
New York Knicks	**0.78134111**
Miami Heat	0.78425656

选择本轮最高得分的未消歧实体 New York Knicks作为实体指称New York的最佳实体，删除其他候选实体 New York City及相关的边，更新图中的边权值。



示例三（续）

实体消歧整体过程示例



参考文献

[1] Shen W, Wang J, Han J. Entity linking with a knowledge base: Issues, techniques, and solutions[J]. IEEE Transactions on Knowledge and Data Engineering, 2015, 27(2): 443-460.
[2] Xue C, Wang H, Jin B, et al. Effective Chinese Organization Name Linking to a List-Like Knowledge Base[J]. Communications in Computer & Information Science, 480:97-110 (2014)
[3] Han X, Sun L. A generative entity-mention model for linking entities with knowledge base[C]. In: Proc. of ACL, pp. 945-954 (2011)
[4] Huang H, Heck L, Ji H. Leveraging deep neural networks and knowledge graphs for entity disambiguation[J]. Computer Science (2015)
[5] Huang H, Cao Y, Huang X, et al. Collective Tweet Wikification based on Semi-supervised Graph Regularization[C]. In: Proc. of ACL, pp. 380-390 (2014)
[6] Cucerzan S. Large-Scale Named Entity Disambiguation Based on Wikipedia Data[C]//EMNLP-CoNLL. 2007, 7: 708-716.
[11] He Z, Liu S, Li M, et al. Learning Entity Representation for Entity Disambiguation[C]//ACL (2). 2013: 30-34.
[12] Huang H, Heck L, Ji H. Leveraging deep neural networks and knowledge graphs for entity disambiguation[J]. arXiv preprint arXiv:1504.07678, 2015.
[13] Zhang W, Sim Y C, Su J, et al. Entity Linking with Effective Acronym Expansion, Instance Selection, and Topic Modeling

参考文献

[14] Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation[J]. Journal of machine Learning research, 2003, 3(Jan): 993-1022.
[15] Zheng Z, Li F, Huang M, et al. Learning to link entities with knowledge base[C]//Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 2010: 483-491.
[16] Ramos J. Using tf-idf to determine word relevance in document queries[C]//Proceedings of the first instructional conference on machine learning. 2003.
[17] Alhelbawy A, Gaizauskas R. Named entity disambiguation using hmms[C]//Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013 IEEE/WIC/ACM International Joint Conferences on. IEEE, 2013, 3: 159-162.
[18] Eddy S R. Hidden markov models[J]. Current opinion in structural biology, 1996, 6(3): 361-365.
[19] Han X, Sun L, Zhao J. Collective entity linking in web text: a graph-based method[C]//Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. ACM, 2011: 765-774.
[20] Alhelbawy A, Gaizauskas R J. Graph Ranking for Collective Named Entity Disambiguation[C]//ACL (2). 2014: 75-80.
[21] Hoffart J, Yosef M A, Bordino I, et al. Robust disambiguation of named entities in text[C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2011: 782-792.

总结

- 知识库的变更：从百科知识库发展到特定领域知识库
- 实体链接的载体：从长文本到短文本，甚至到列表和表格数据
- 候选实体生成追求同义词、简称、各种缩写等的准备和高效从Mention到实体候选的查找
- 实体消歧则考虑相似度计算的细化和聚合，以及基于图计算协同消歧

知识规则挖掘

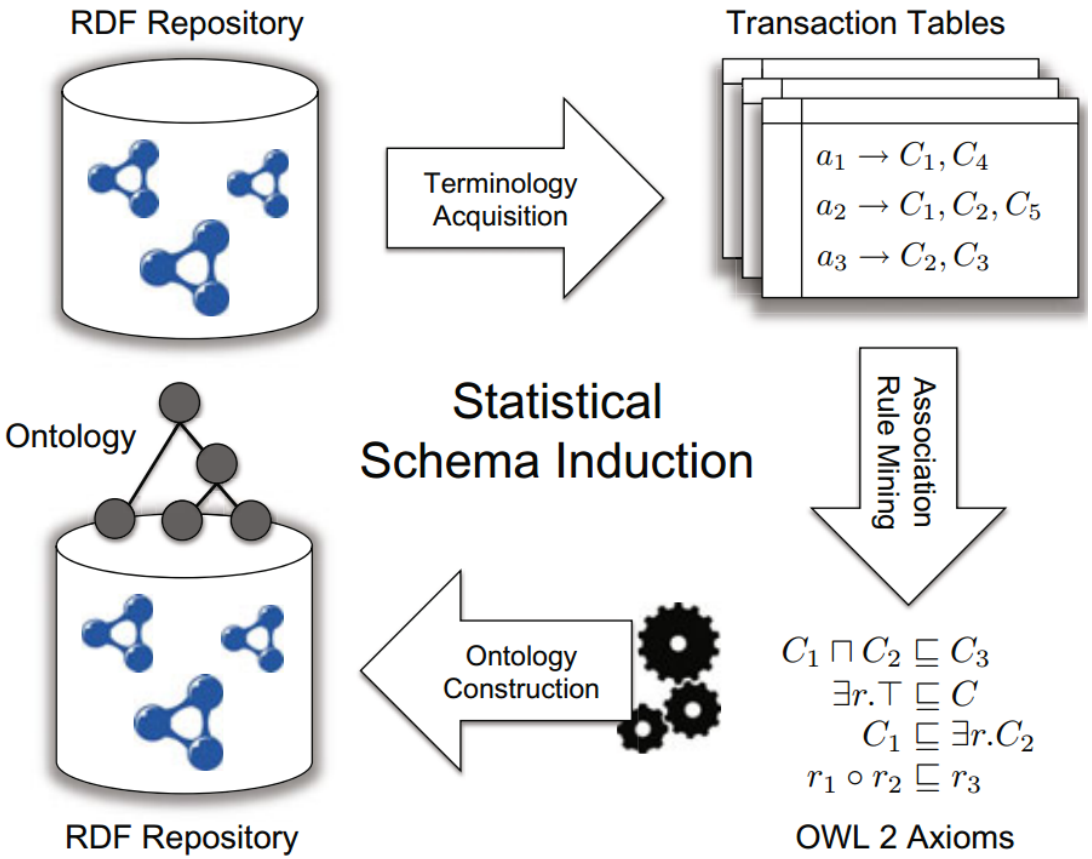
Statistical Schema Induction – 主要方法

- 基于归纳逻辑编程 (Inductive Logic Programming, ILP) 的方法
  - 使用精化算子 (refinement operators)
- 基于统计关系学习 (Statistical Relational Learning, SRL) 的方法
  - 主要对贝叶斯网络进行扩展
- 基于关联规则挖掘 (Association Rule Mining，ARM)的方法

① 构建事务表
② 挖掘规则
③ 将规则转换为OWL公理
④ 构建本体

Statistical Schema Induction – 关联规则挖掘 (ARM)

- OWL2公理可被转换为关联规则



示例

公理 (Axiom)	规则 (Rules)
$C \sqsubseteq D$	$\{C\} \Rightarrow \{D\}$

规则 $\{C\} \Rightarrow \{D\}$ 意味着：概念C的实例同时也属于概念D

规则的置信度 (confidence)越高，$C \sqsubseteq D$越可能成立

关联规则挖掘（续）

事务表 (Transaction Table)

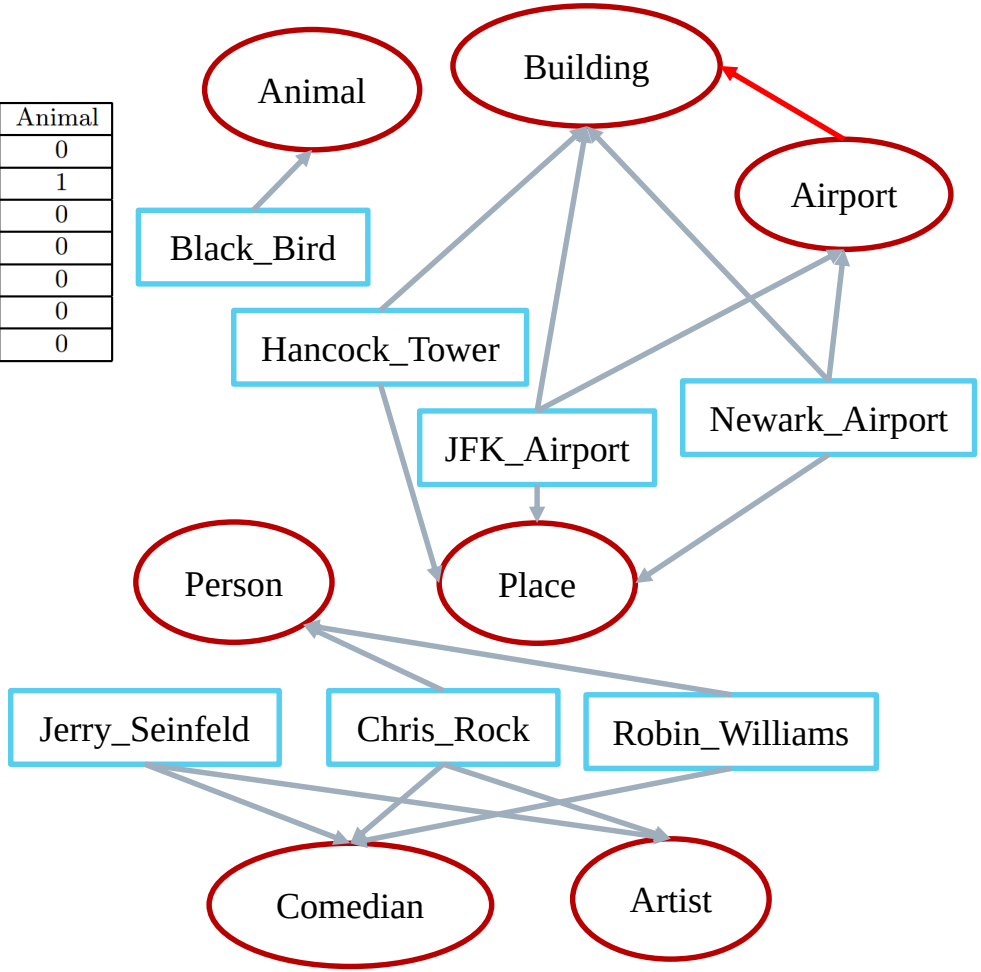
IRI	Comedian	Artist	Person	Airport	Building	Place	Animal
Jerry_Seinfeld	1	1	0	0	0	0	0
Black_Bird	0	0	0	0	0	0	1
Chris_Rock	1	1	1	0	0	0	0
Robin_Williams	1	0	1	0	0	0	0
JFK_Airport	0	0	0	1	1	1	0
Hancock_Tower	0	0	0	0	1	1	0
Newark_Airport	0	0	0	1	1	1	0

$$support(Airport, Building) = 2$$

$$support(Airport) = 2$$

$$confidence(Airport \Longrightarrow Building) = 1$$

$$Airport \sqsubseteq Building$$



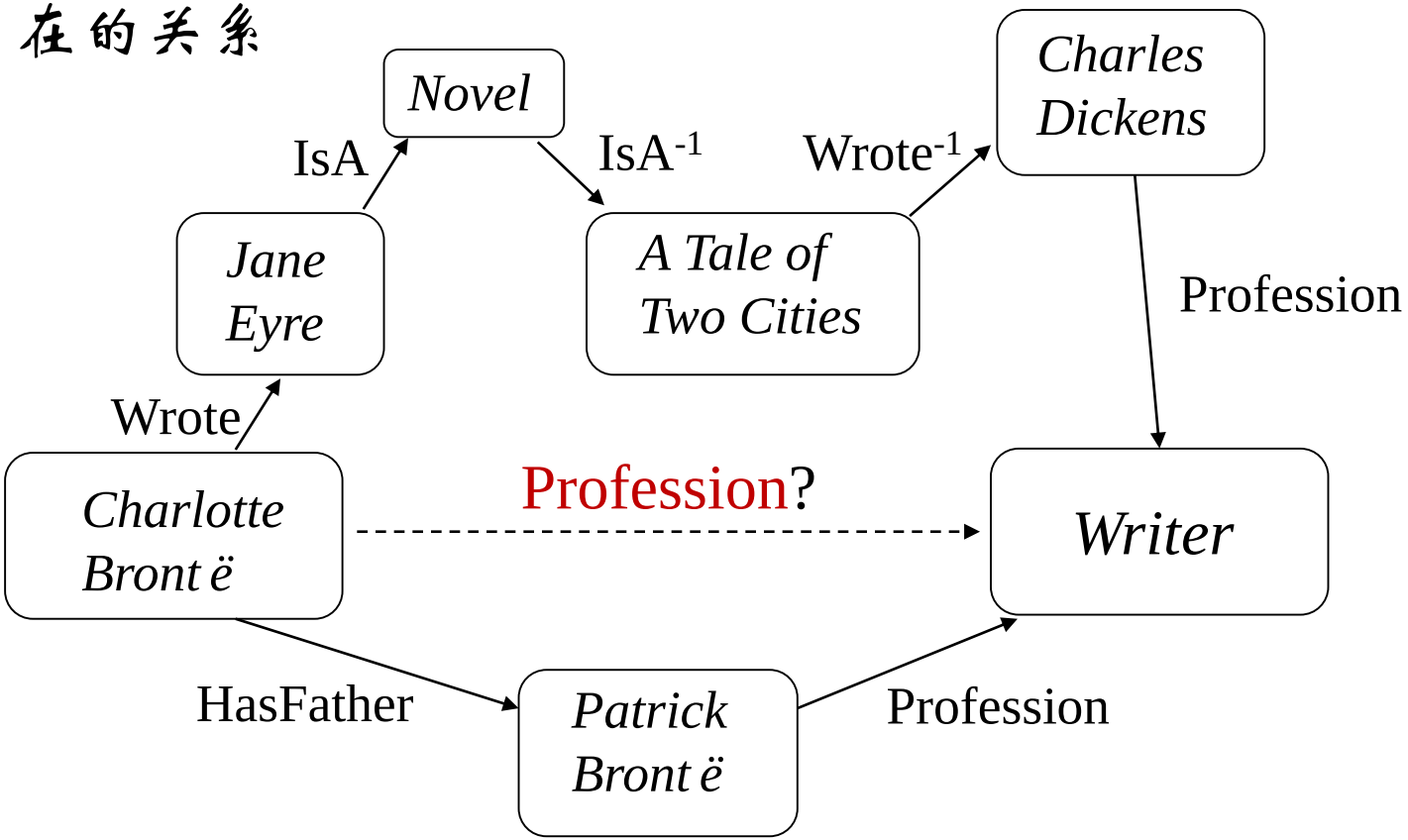
Statistical Schema Induction – 统计关系学习 (SRL)

- 输入
  - 实体集合$\{e_i\}$
  - 关系集合$\{r_k\}$
  - 已知三元组集合$\{(e_i, r_k, e_j)\}$
- 目标
  - 根据已知三元组对未知三元组成立的可能性进行预测
  - $\mathrm{P}\left((e_i, r_k, e_j) = 1\right)$
  - 可以应用于知识图谱补全

统计关系学习 – 基于图的方法

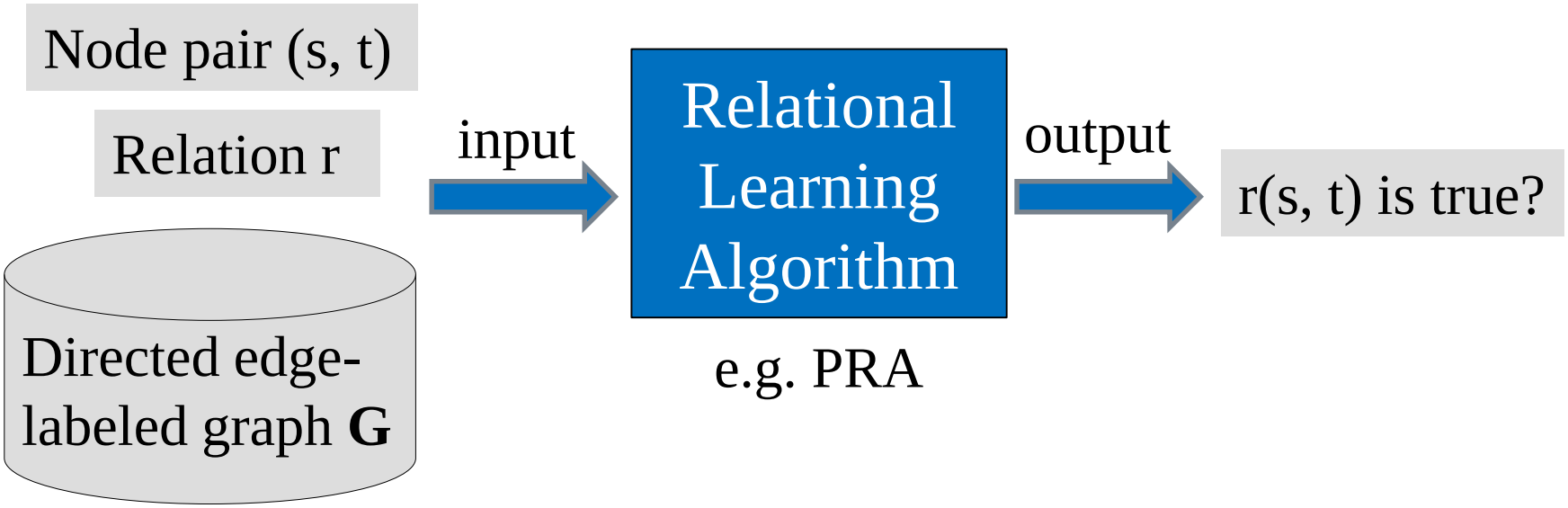
基本思想

- 将连接两个实体的路径作为特征来预测其间可能存在的关系

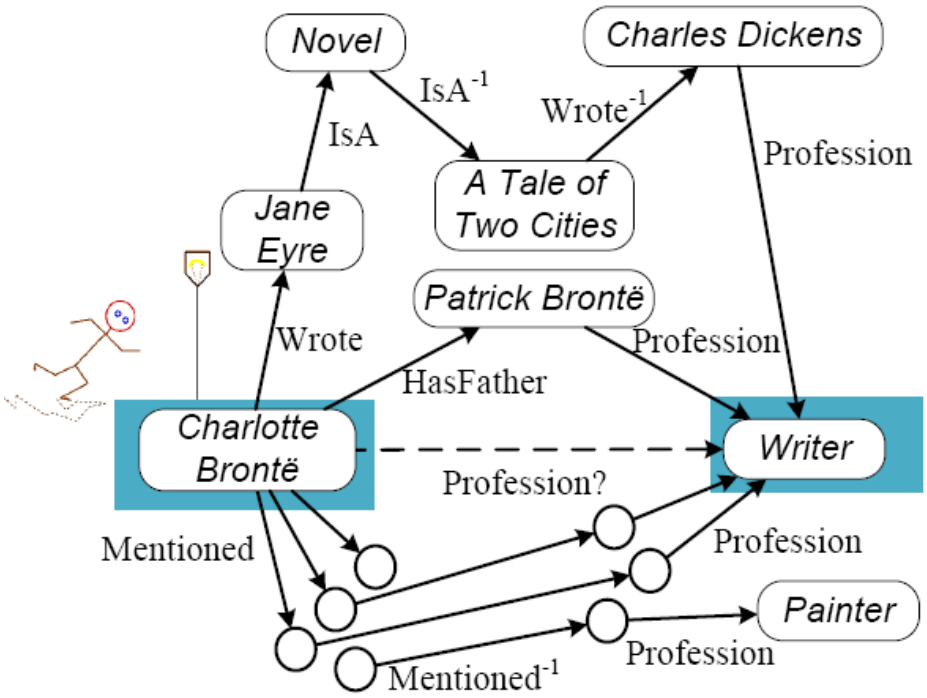


路径排序算法 – Path Ranking Algorithm (PRA)

- 通用关系学习框架 (generic relational learning framework)



Path Ranking Algorithm (续)



G=(N,E, R)

- N: nodes (instances or concepts)
- E: edges
- R: edge types

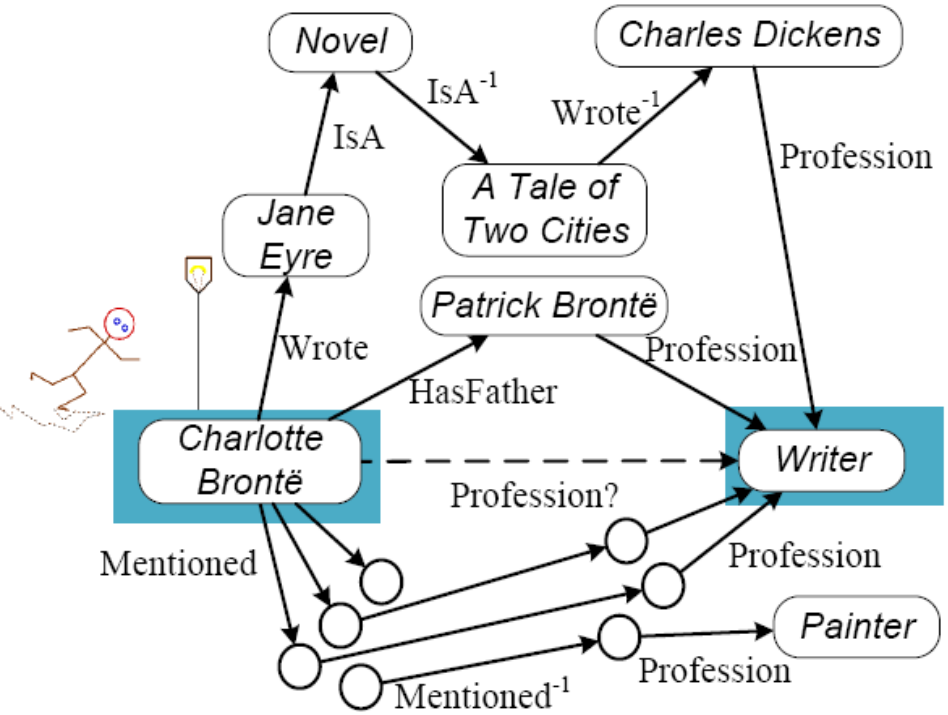
Note: r^{-1}: reverse of edge type r

Path type π: $<r_1,r_2,\ldots,r_n>$

e.g. <HasFather, Profession>

Lao et. al. EMNLP 2011

Path Ranking Algorithm (续)



Profession(Charlotte Bonte, Writer)?

$$score(s,t) = \sum_{\pi \in Q} P(s \rightarrow t; \pi)\theta_\pi$$

Q: all path types starting from s and ending with t (with length of n)

θ_π: weights obtained by training

Path Ranking Algorithm (续)

- Probability of a path type

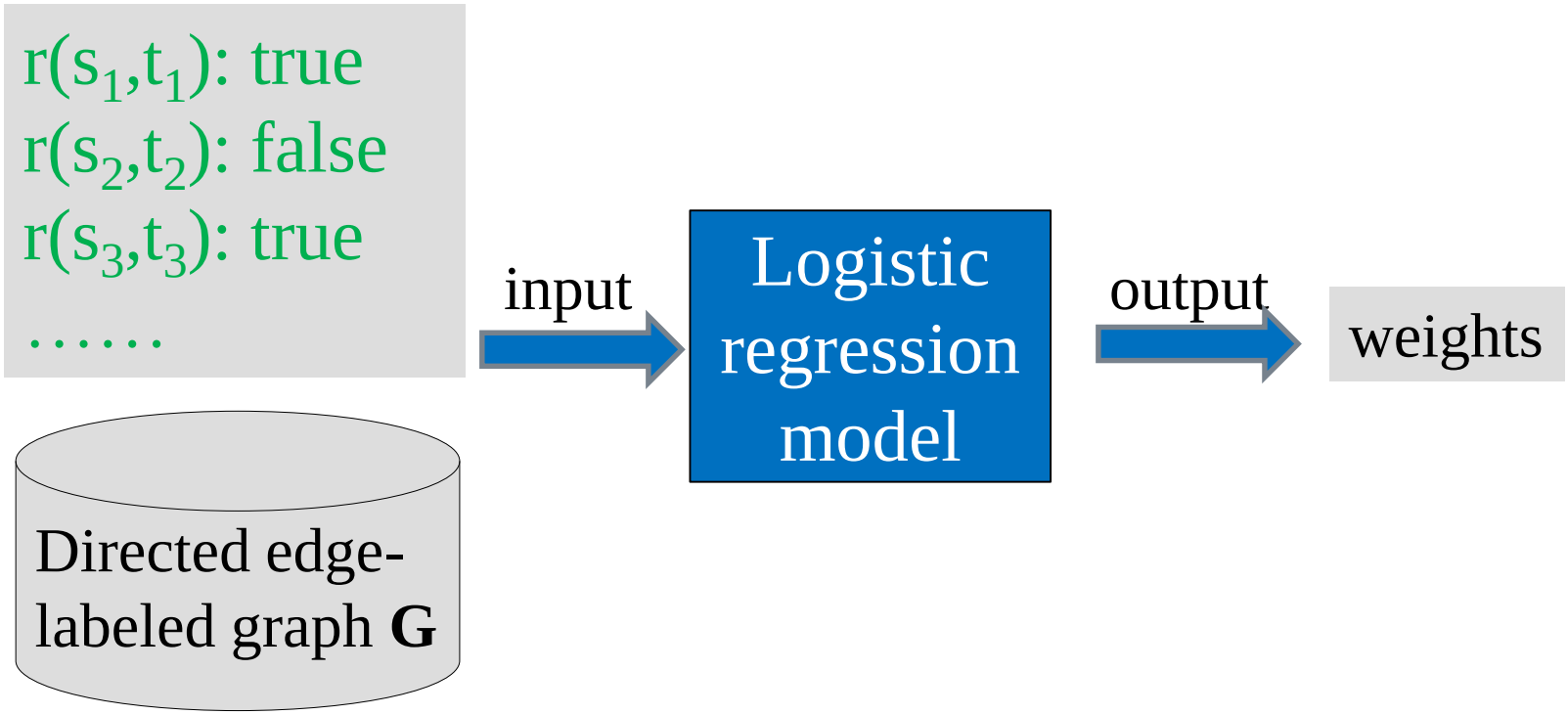
$$P(s \to t; \pi) = \sum P(s \to z; \pi') P(z \to t; r)$$

P:
the probability of reaching target node t starting from source node s and following path π

Use dynamic programming procedure

Path Ranking Algorithm (续)

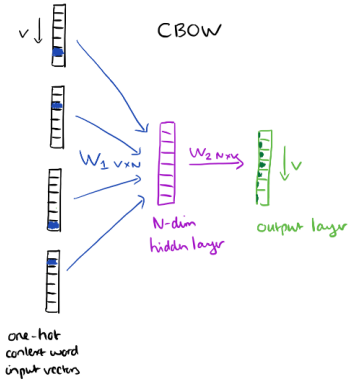
- Weight training (offline)

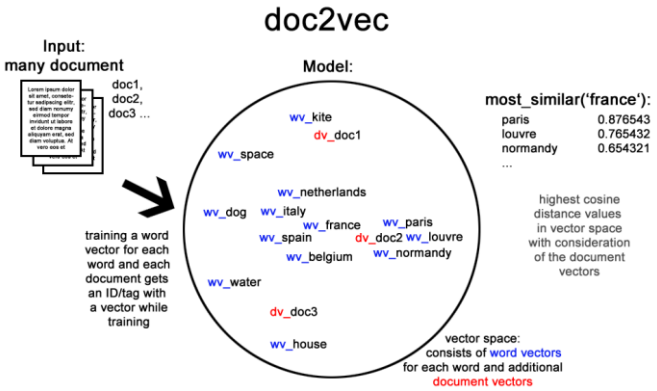


知识图谱表示学习

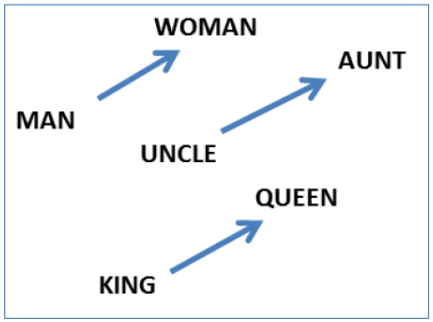
表示学习的意义

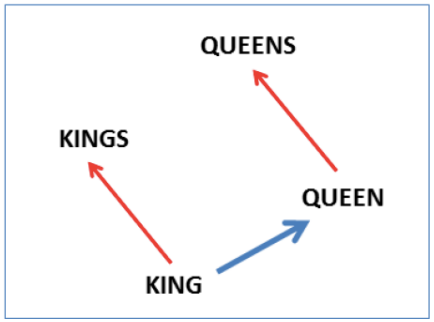
自然语言中的表示学习





建立统一的语义空间，语义可计算





知识图谱表示学习

实体预测和推理

卧虎藏龙

观影人群

知识图谱表示学习

实体预测和推理

卧虎藏龙

观影人群

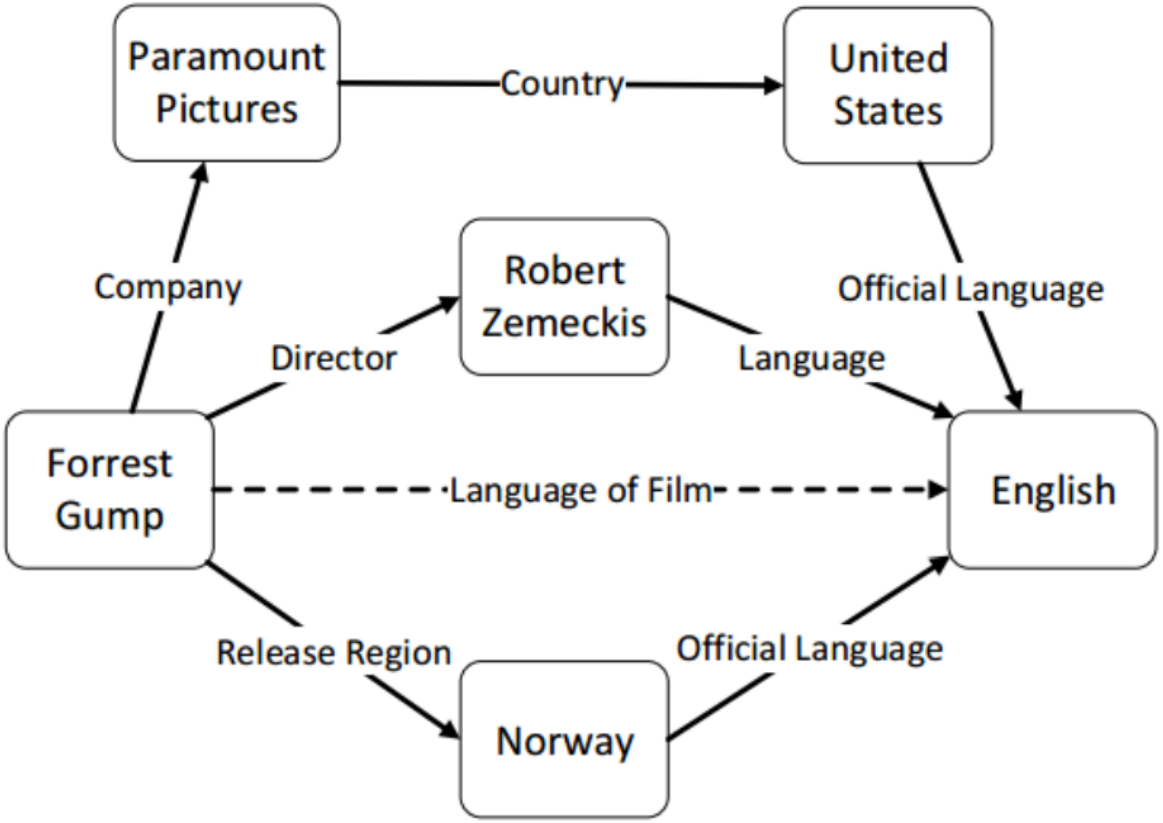
章子怡的粉丝

武侠片的粉丝

周润发的粉丝

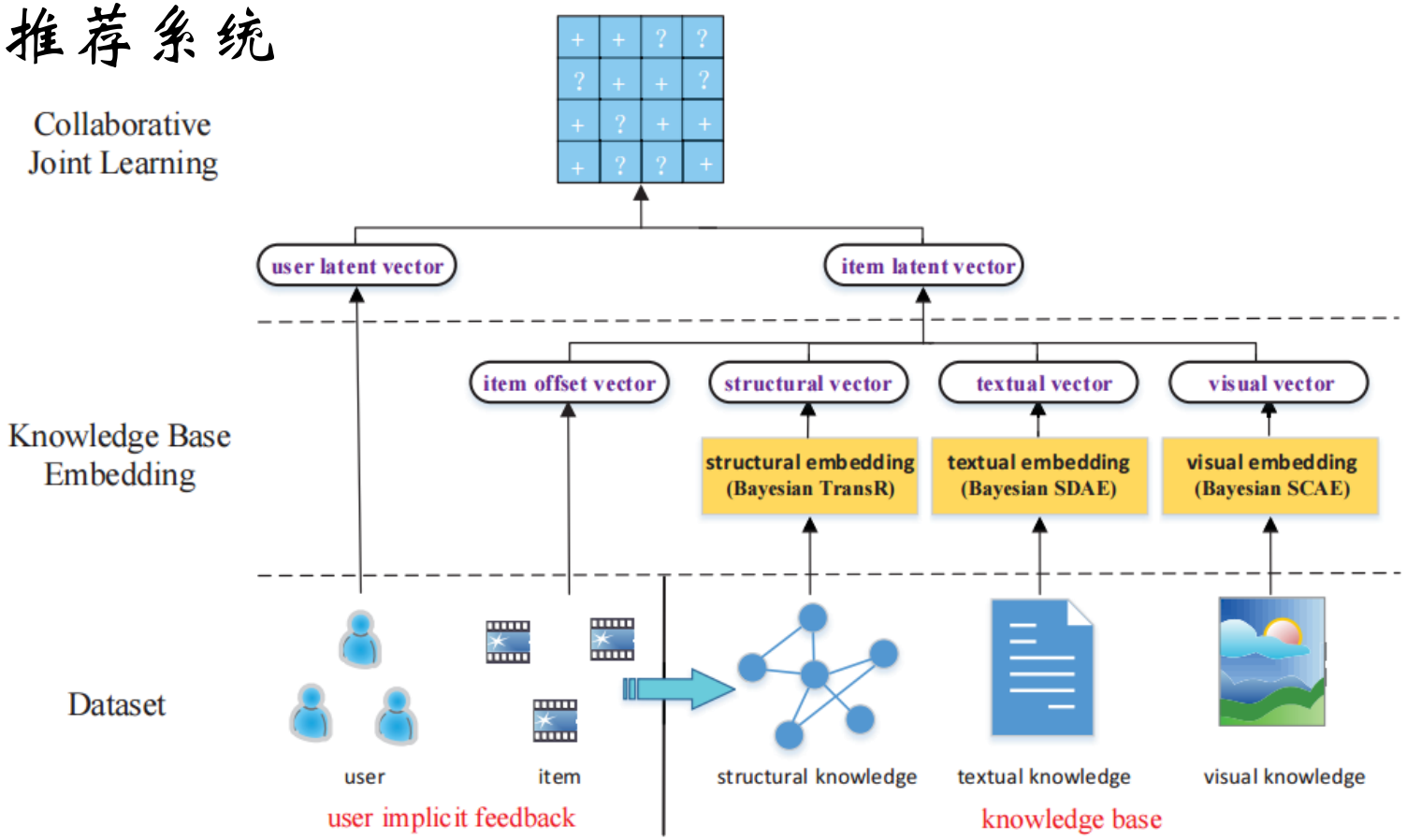
知识图谱表示学习

关系推理



知识图谱表示学习

推荐系统



Fuzheng Zhang, et al 2016

知识图谱表示学习 -- TransE

将三元组<h, r, t>看成h通过r翻译到t的过程

Beijing-China = Pairs-France = Capital-of

- 优化目标
  - 势能函数

$$f(h,r,t)=\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|_2$$

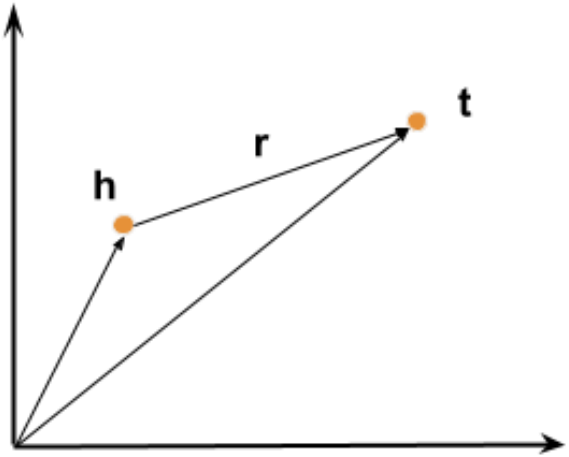
f(Beijing, Capital-of, China) < f(Shanghai, Capital-of, China)

  - 目标函数

$$min \sum_{(h,r,t)\in\Delta}\sum_{(h',r,t')\in\Delta'}[\gamma+f(h,r,t)-f(h',r,t')]_+$$

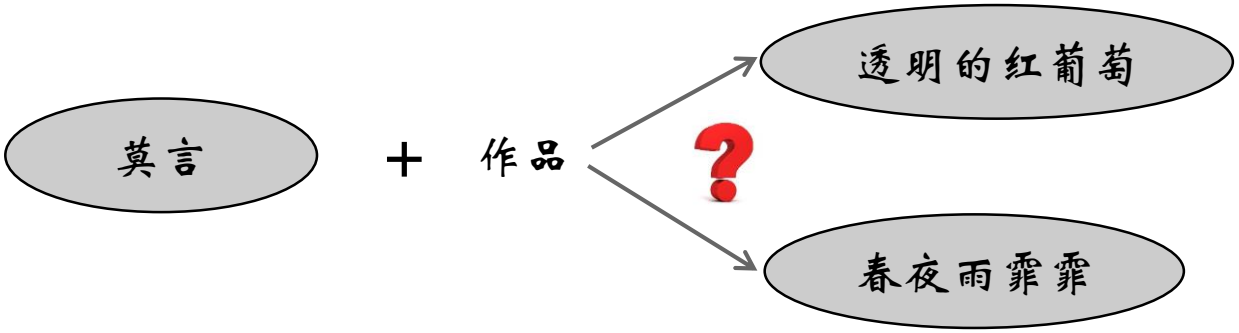
其中 $[x]_+=\max(0,x)$

Bordes, et al 2013

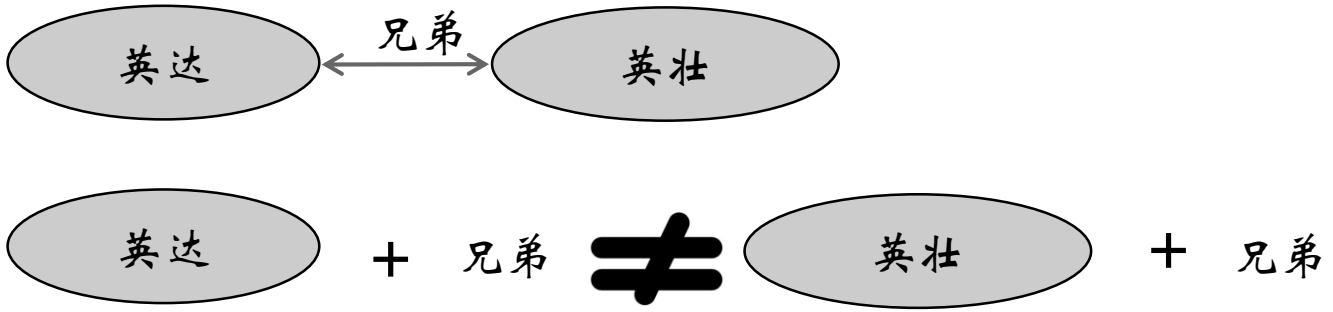


知识图谱表示学习 -- TransE

无法处理一对多、多对一和多对多问题

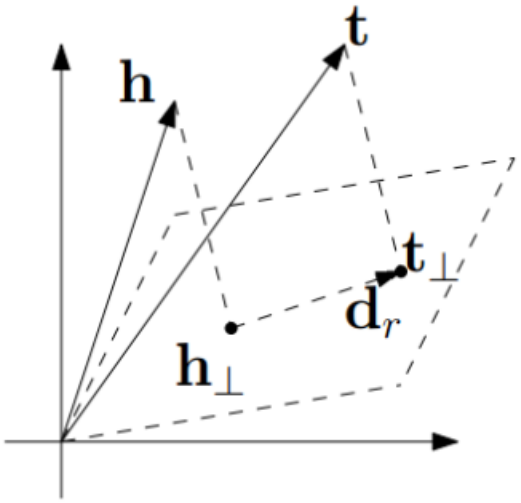


关系的性质

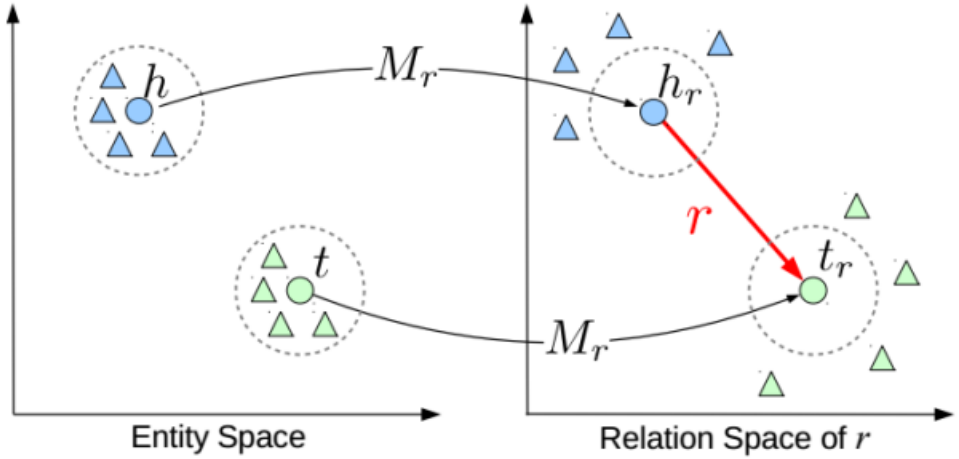


克服TransE的缺陷

实体语义空间投影



TransH

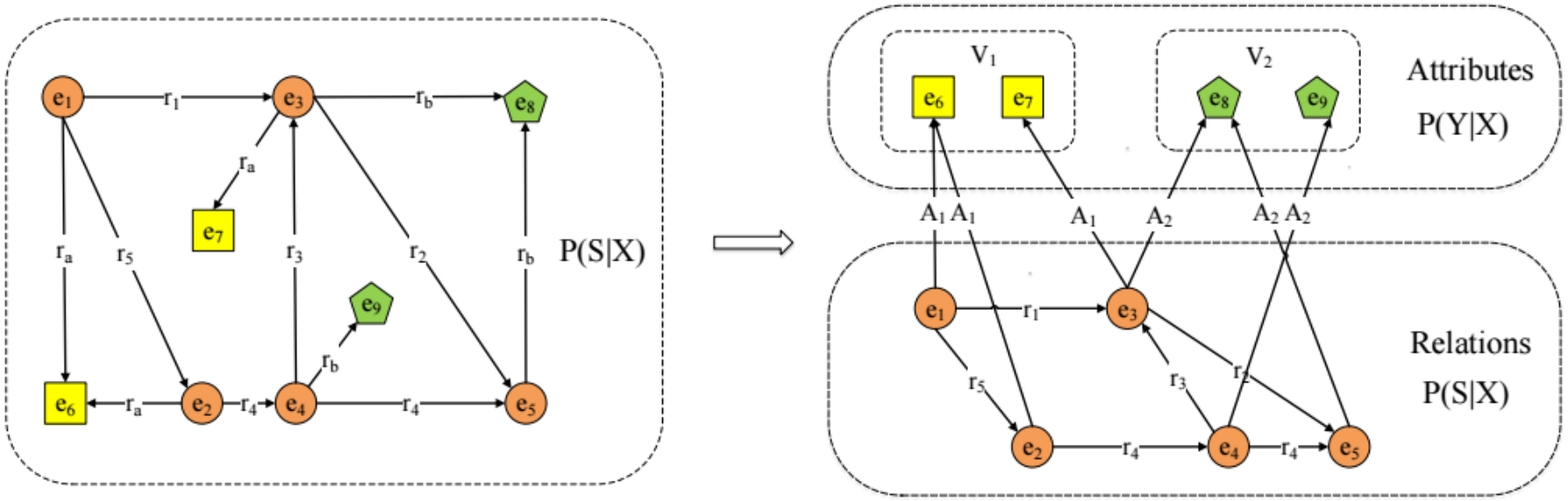


TransR

Wang, et al 2014
Lin, et al 2015

属性如何表示

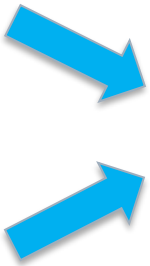
分而治之



Lin Y, et al 2016

PRA vs. TransE

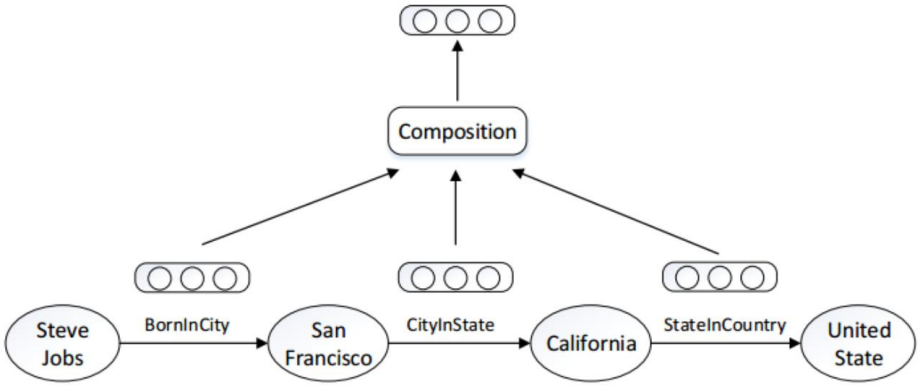
- PRA
  - 可解释性强
  - 能够从数据中挖掘出推理规则
  - 难以处理稀疏关系
  - 路径特征提取效率不高
- TransE
  - 能够表示数据中蕴含的潜在特征
  - 参数较少，计算效率较高
  - 模型简单，难以处理多对一、一对多、多对多的复杂关系
  - 可解释性不强



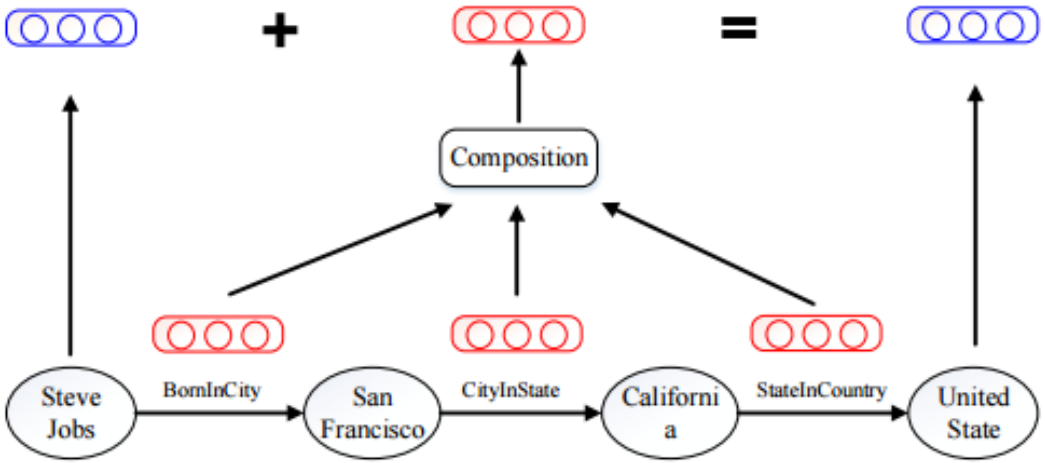
两类方法之间存在互补性

路径的表示学习

TransE孤立地学习每个事实三元组，关系之间存在复杂关系，涉及关系推理

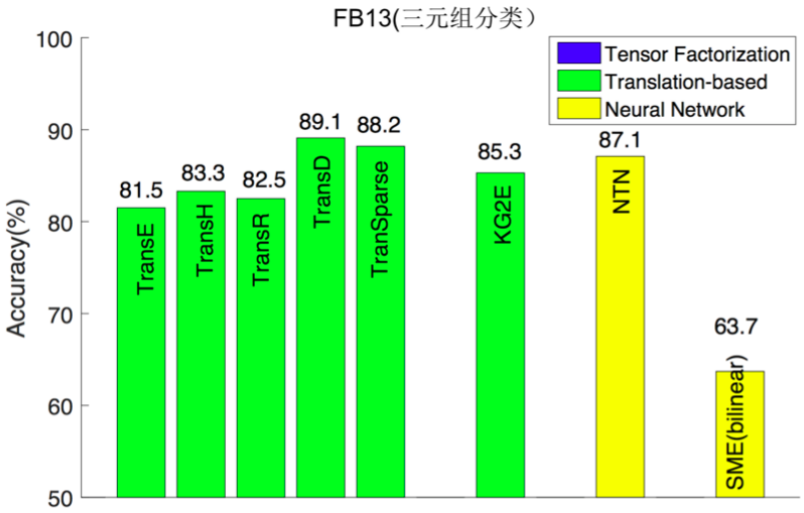


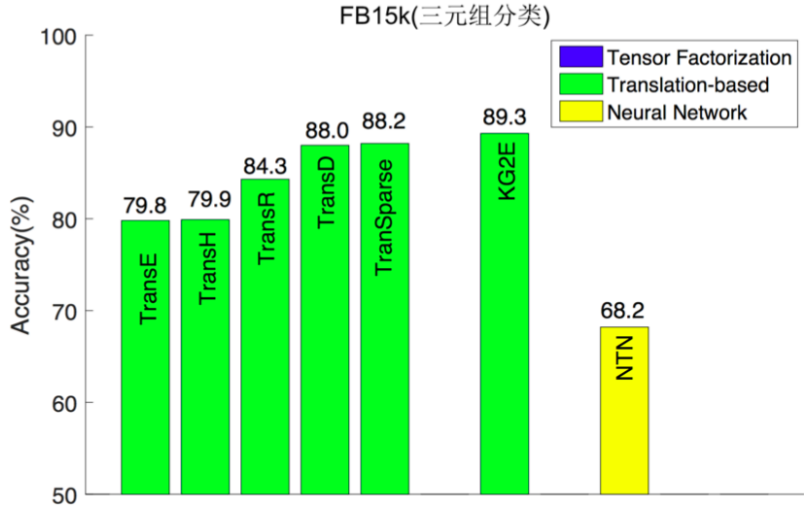
	TransE	PTransE
KB	$h \xrightarrow{r} t$	$h \xrightarrow{r_1} e_1 \xrightarrow{r_2} t$
Triples	(h, r, t)	(h, r_1, e_1) (e_1, r_2, t) $(h, r_1 \circ r_2, t)$
Objectives	$\mathbf{h} + \mathbf{r} = \mathbf{t}$	$\mathbf{h} + \mathbf{r}_1 = \mathbf{e}_1$ $\mathbf{e}_1 + \mathbf{r}_2 = \mathbf{t}$ $\mathbf{h} + (\mathbf{r}_1 \circ \mathbf{r}_2) = \mathbf{t}$

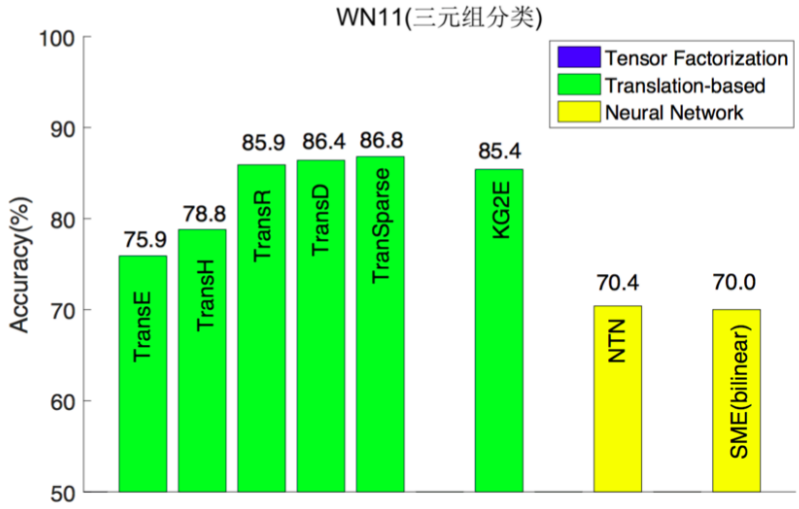


Gardne, et al 2013

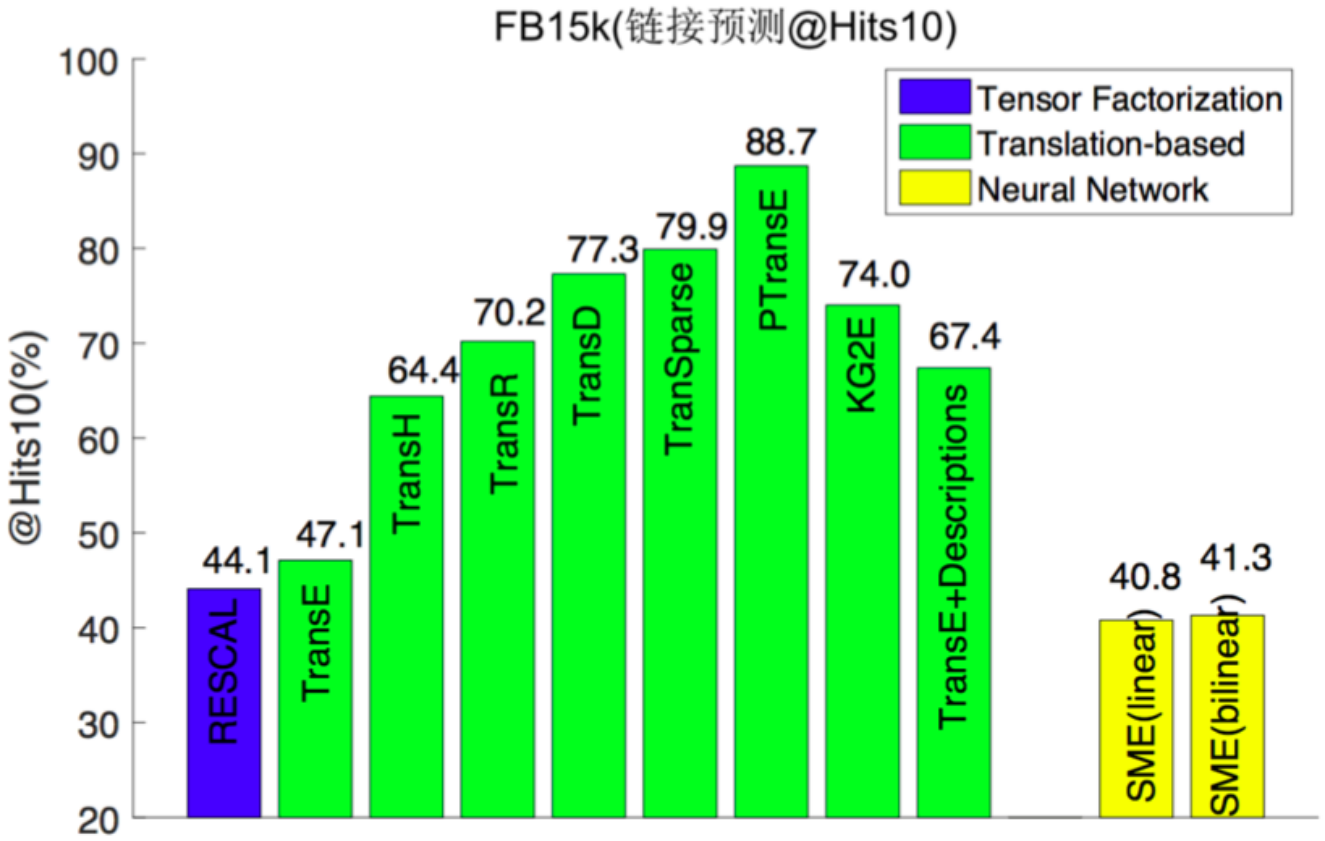
性能比较 – 三元组分类


FB13(三元组分类)
Tensor Factorization
Translation-based
Neural Network
Accuracy(%)
100
90
80
70
60
50
81.5
83.3
82.5
89.1
88.2
85.3
87.1
63.7
TransE
TransH
TransR
TransD
TranSparse
KG2E
NTN
SME(bilinear)

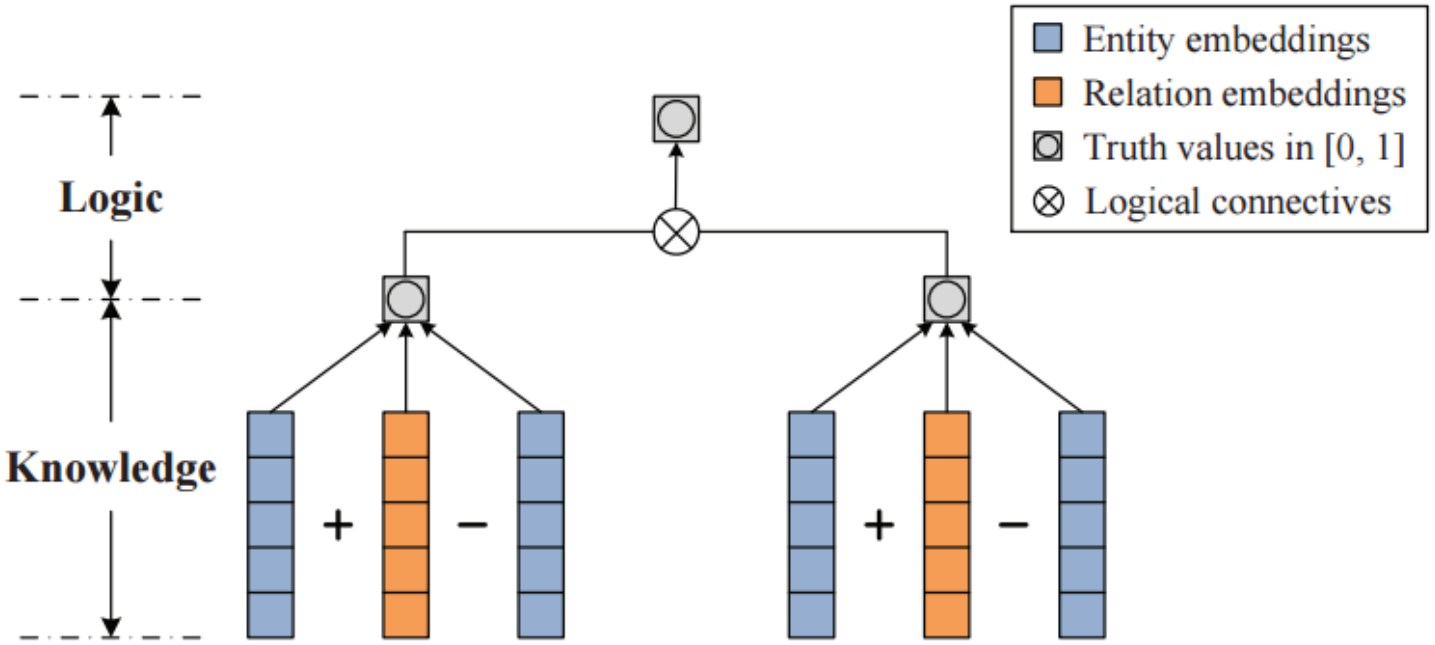

FB15k(三元组分类)
Tensor Factorization
Translation-based
Neural Network
Accuracy(%)
100
90
80
70
60
50
79.8
79.9
84.3
88.0
88.2
89.3
68.2
TransE
TransH
TransR
TransD
TranSparse
KG2E
NTN


WN11(三元组分类)
Tensor Factorization
Translation-based
Neural Network
Accuracy(%)
100
90
80
70
60
50
75.9
78.8
85.9
86.4
86.8
85.4
70.4
70.0
TransE
TransH
TransR
TransD
TranSparse
KG2E
NTN
SME(bilinear)

性能比较 – 链接预测


FB15k(链接预测@Hits10)
@Hits10(%)
100
90
80
70
60
50
40
30
20
Tensor Factorization
Translation-based
Neural Network
44.1
RESCAL
47.1
TransE
64.4
TransH
70.2
TransR
77.3
TransD
79.9
TranSparse
88.7
PTransE
74.0
KG2E
67.4
TransE+Descriptions
40.8
SME(linear)
41.3
SME(bilinear)

加入规则的表示学习

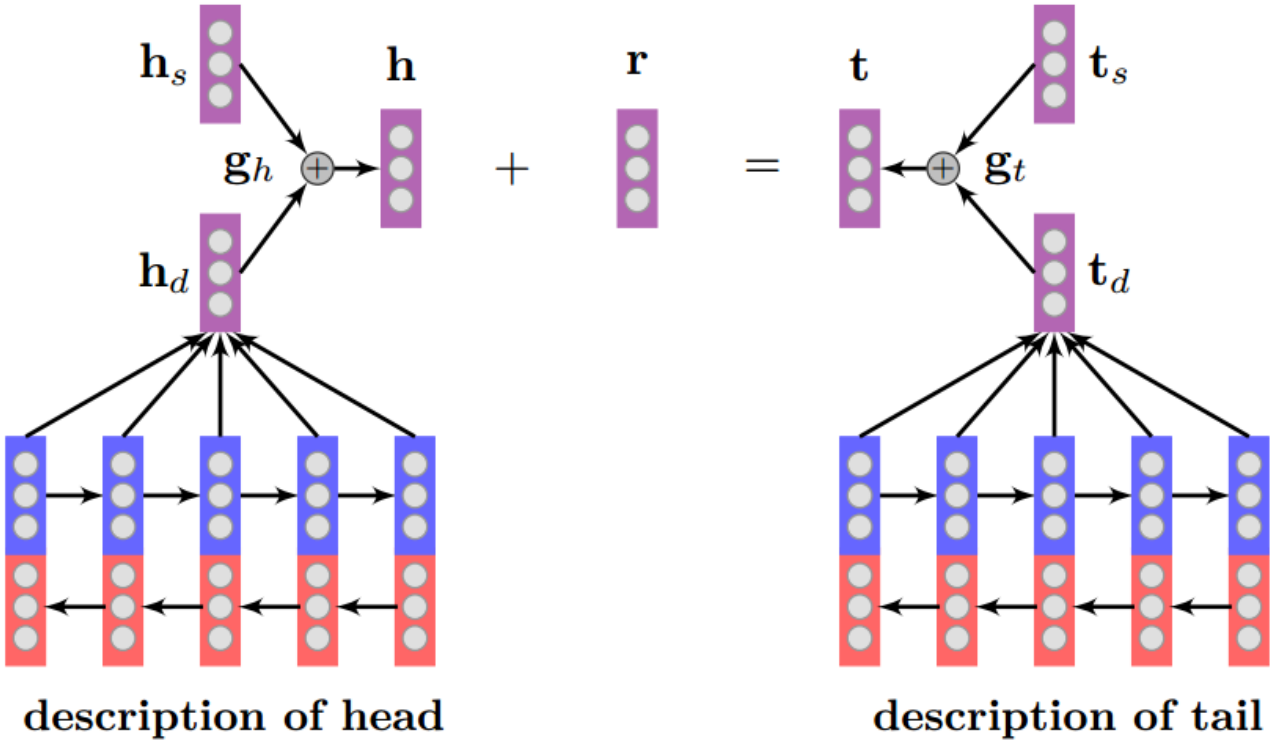


(Paris, Capital-of, France) → (Paris, Located-in, France)

Shu G, et al 2016

多模态的表示学习

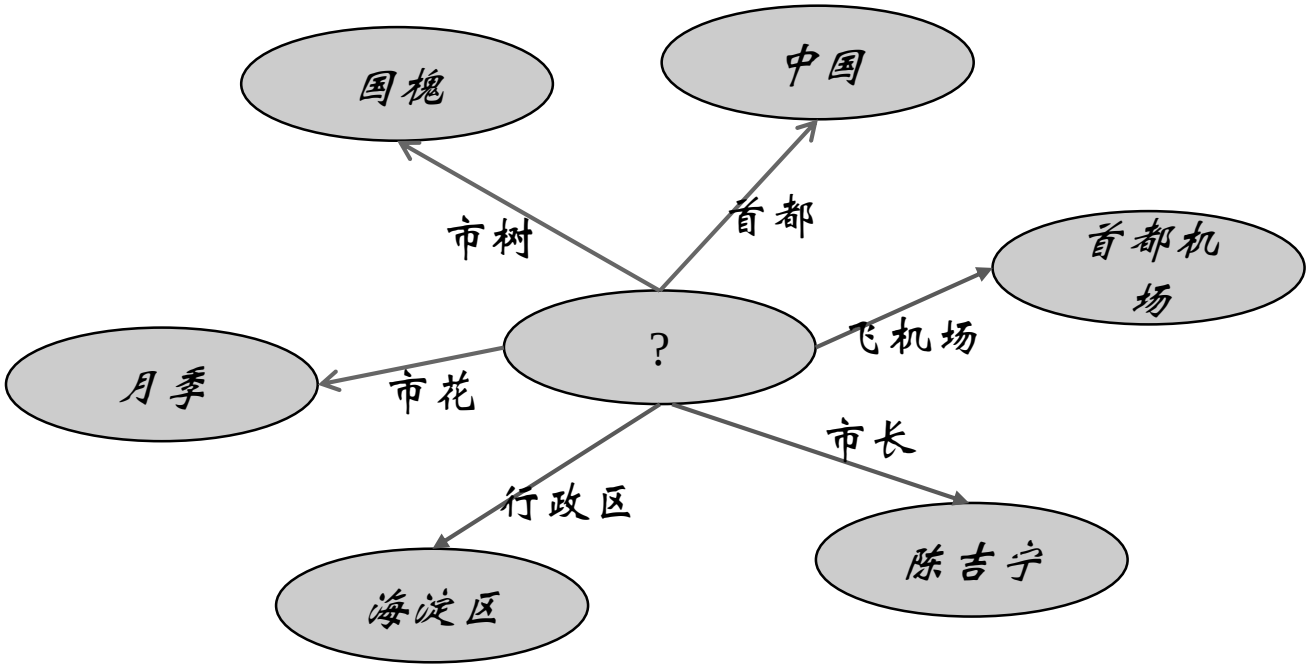
助力Zero-Shot和长尾链接预测



Xu J, et al 2016

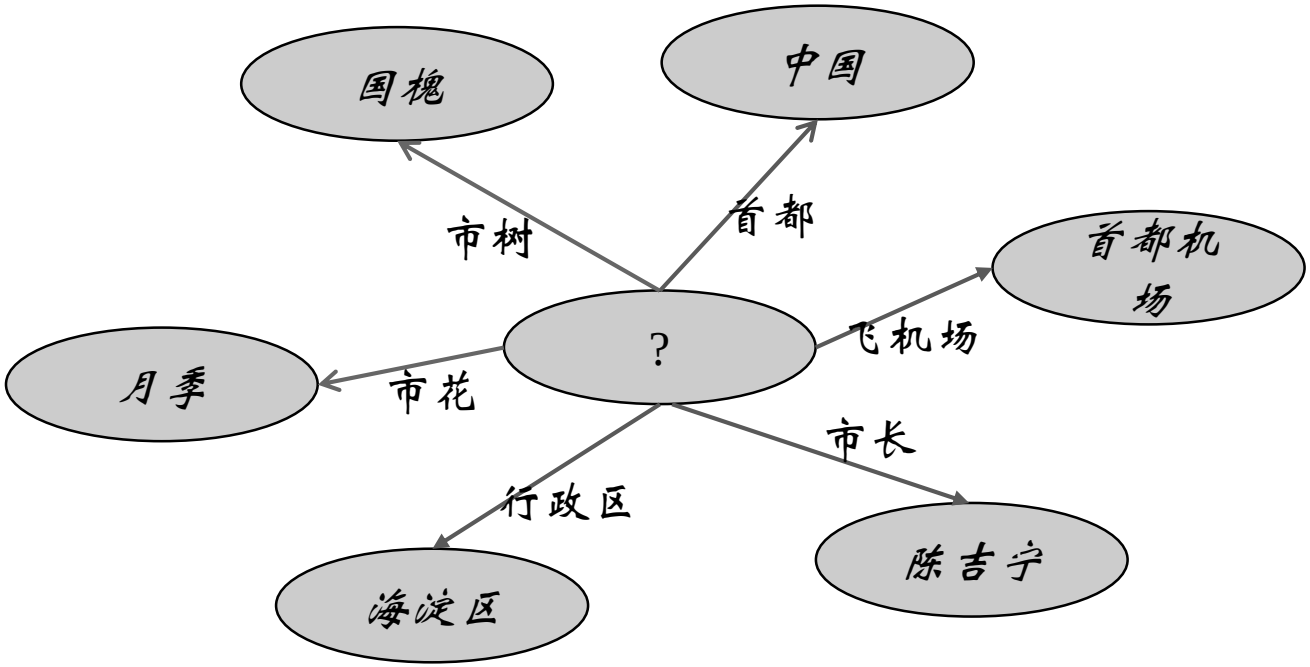
基于知识图谱图结构的表示学习

哪些数据可以用来描述实体



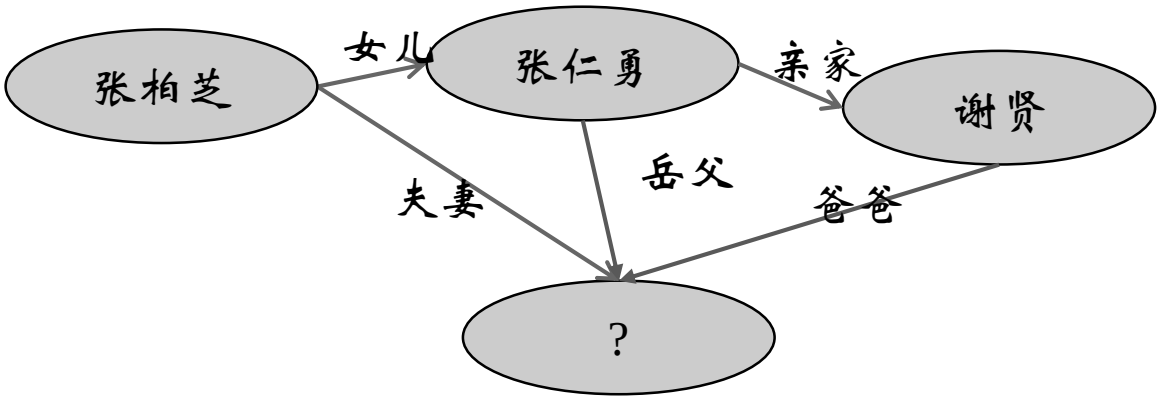
基于知识图谱图结构的表示学习

哪些数据可以用来描述实体



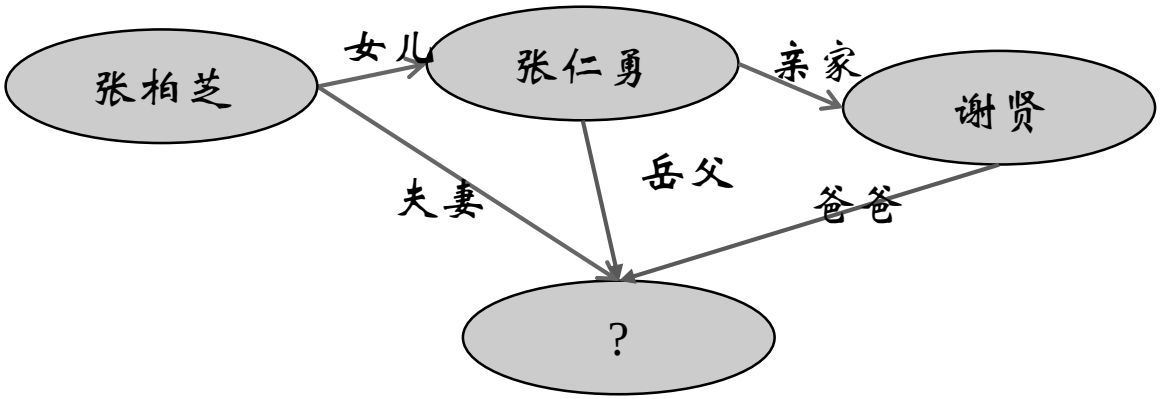
基于知识图谱图结构的表示学习

哪些数据可以用来描述实体



基于知识图谱图结构的表示学习

哪些数据可以用来描述实体

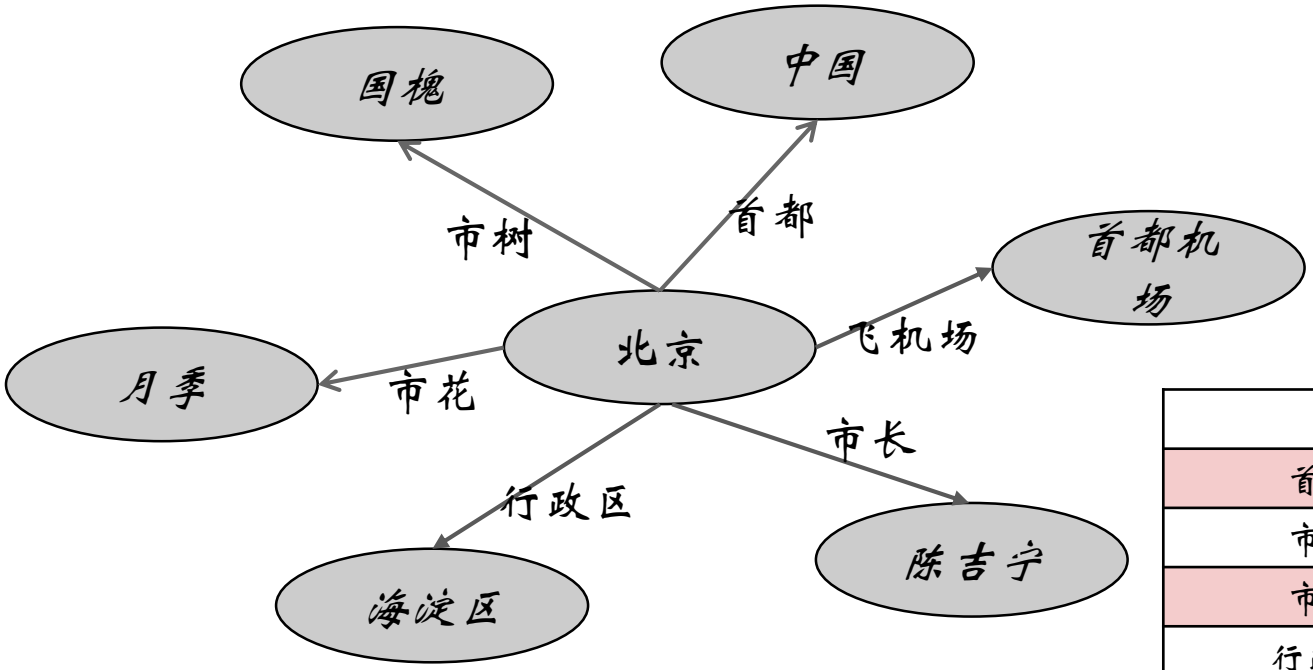


基于知识图谱图结构的表示学习

- 哪些数据可以用来描述实体
  - 实体周围的实体
  - 从一个实体到这个实体的联通路径

基于知识图谱图结构的表示学习

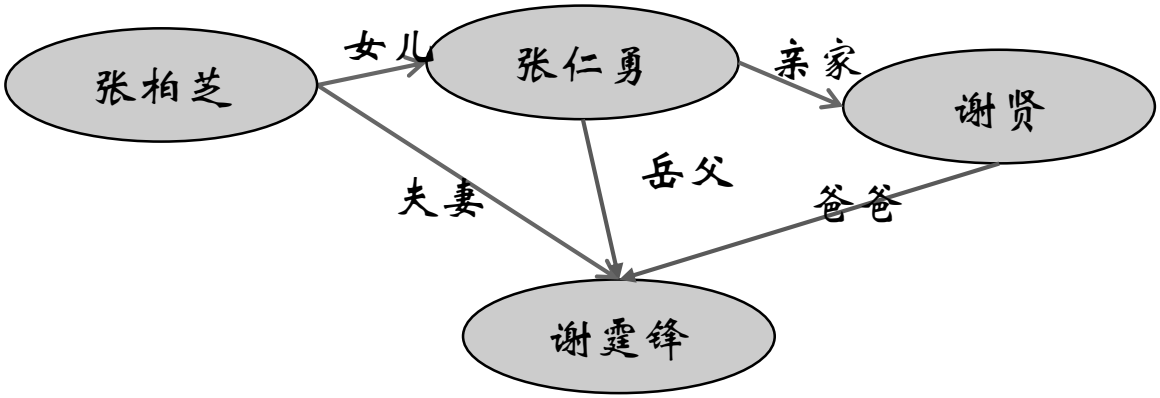
Neighbor Context

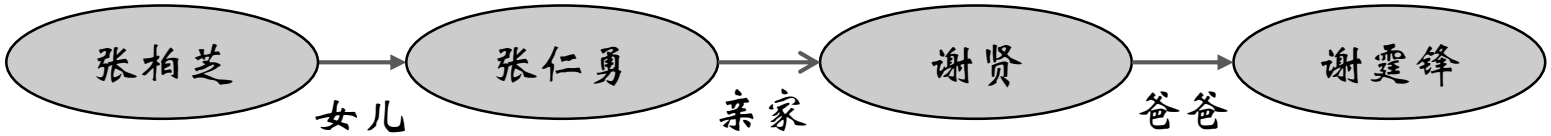


r	t
首都	中国
市树	国槐
市花	月季
行政区	海淀区
市长	陈吉宁
飞机场	首都机场

基于知识图谱图结构的表示学习

Path Context





基于知识图谱图结构的表示学习

Triple Context = Triple + Path Context + Neighbor Context

- 势能函数
  - ✓ 希望三元组在Triple Context概率最大

$$f(h,r,t) = P((h,r,t)|C(h,r,t);\Theta)$$

  - ✓ 假设不同的Context都是相互独立的企且独立用来描述三元组的某一部分

$$\begin{aligned} f(h,r,t) = & P(h|C(h,r,t);\Theta) \\ & \cdot P(t|C(h,r,t),h;\Theta) \\ & \cdot P(r|C(h,r,t),h,t;\Theta) \end{aligned}$$

$$f(h,r,t) \approx P(h|C_N(h);\Theta) \cdot P(t|C_P(h,t),h;\Theta) \cdot P(r|h,t;\Theta)$$

  - ✓ 目标函数

$$P(\mathcal{K}|\Theta) = \prod_{(h,r,t)\in\mathcal{K}} f(h,r,t)$$

基于知识图谱图结构的表示学习

➢ 实验结果

✓ 在一对多、多对多、多对一下均有较好的表现

Task	Predicting head(HITS@10(%))				Predicting tail(HITS@10(%))			
Relation Category	1-To-1	1-To-N	N-To-1	N-To-N	1-To-1	1-To-N	N-To-1	N-To-N
TransE	43.7	65.7	18.2	47.2	43.7	19.7	66.7	50.0
TransH (unif)	66.7	81.7	30.2	57.4	63.7	30.1	83.2	60.8
TransH (bern)	66.8	87.6	28.7	64.5	65.5	39.8	83.3	67.2
TransR (unif)	76.9	77.9	38.1	66.9	76.2	38.4	76.2	69.1
TransR (bern)	78.8	**89.2**	34.1	69.2	79.2	37.4	**90.4**	72.1
CTransR (unif)	78.6	77.8	36.4	68.0	77.4	37.8	78.0	70.3
CTransR (bern)	**81.5**	89.0	34.7	71.2	**80.8**	38.6	90.1	73.8
TCE	71.0	60.3	**83.9**	**81.9**	70.3	**89.9**	76.0	**89.2**

基于知识图谱图结构的表示学习

➢ 实验结果

✓ 如果用TransE训练后的结果作为输入还有提高的空间

Metric	Mean Rank		Hits@10(%)	
	Raw	Filter	Raw	Filter
TransE	243	125	34.9	47.1
TransH (unif)	211	84	42.5	58.5
TransH (bern)	212	87	45.7	64.4
TransR (unif)	226	78	43.8	65.5
TransR (bern)	198	77	48.2	68.7
CTransR (unif)	233	82	44.0	66.3
CTransR (bern)	199	75	48.4	70.2
PTransE	207	58	51.4	**84.6**
GAKE	228	119	44.5	64.8
TCE	**110**	**25**	**55.3**	83.1

总结和挑战

- 融合更多本体特征的知识图谱表示学习算法研发
- 知识图谱表示学习与本体推理之间的等价性分析
- 知识图谱学习与网络表示学习之间的异同
- 神经符号系统

http://www.openkg.cn/tool/openke

参考文献

[1] Fuzheng Zhang , Nicholas Jing Yuan, Defu Lian, Xing Xie, Wei-Ying Ma, Collaborative Knowledge Base Embedding for Recommender Systems, in Proc. of KDD, 2016.

[2] Bordes, Antoine, , Usunier N, Garcia-Duran A, et al. Translating embeddings for modeling multi-relational data. Advances in neural information processing systems, 2013.

[3] Wang, Z., Zhang, J., Feng, J., & Chen, Z. Knowledge Graph Embedding by Translating on Hyperplanes. in Proc. Of AAAI, 2014.

[4] Lin, Y., Liu, Z., Sun, M., Liu, Y., & Zhu, X. (2015, January). Learning Entity and Relation Embeddings for Knowledge Graph Completion. in Proc. Of AAAI, 2015.

[5] Lin Y, Liu Z, Sun M. Knowledge representation learning with entities, attributes and relations. Graph Completion. in Proc. Of IJCAI, 2016.

[6] Gardner, M., Talukdar, P. P., Kisiel, B., & Mitchell, T. Improving Learning and Inference in a Large Knowledge-base using Latent Syntactic Cues. in Proc. Of EMNLP, 2013.

[7] Guo S, Wang Q, Wang L, et al. Jointly Embedding Knowledge Graphs and Logical Rules. in Proc. Of EMNLP, 2016.

[8] Xu, J., Chen, K., Qiu, X., & Huang, X. Knowledge Graph Representation with Jointly Structural and Textual Encoding. in Proc. Of IJCAI, 2016.

谢谢大家！

本课程课件由OpenKG提供支持

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：小象
- 新浪微博：ChinaHadoop