# 实体链接技术总结

作者：纽约的自行车

日期：2021 年 12 月 9 号

**摘　　要**

本文首先介绍了实体链接的基本概念、研究内容和常用方法，然后详细介绍了两种近年来的在候选实体排序上的 SOTA 方法，最后介绍了两种端到端的方法。

## 1　实体链接系统

### 1.1　基本概念

实体链接（entity link，EL）是将实体提及（entity mention）链接到知识库中对应实体，也称为实体消歧。所谓实体提及就是自然文本中表达实体的文本片段 [1]，在实体链接任务中，实体提及是事先在文本中标注好的。一般有知识库的地方就离不开 EL，以下是 EL 的几个应用：

1. 基于知识库的问答系统：问题中的实体提及需要链接到知识库中的实体；
2. 语义理解：可应用于舆情分析、内容推荐、阅读增强；
3. 信息检索：基于语义实体的搜索引擎；
4. 知识库创建或合并：扩充知识库，更新实体和关系。

### 1.2　研究内容

一个实体链接系统包括如下研究内容 [1]：

1. 候选实体生成。针对每一个提及，识别该提及在知识图谱中可能指向的候选目标实体。一个提及可能对应多个实体，例如“苹果”可能指向的目标实体包括苹果 (水果)，苹果公司，苹果 (电影)，苹果 (银行)，…；
2. 候选实体排序。基于提及的上下文等信息对目标实体进行排序。例如，系统需要根据“苹果”的上下文词语发布会，编程语言，开发者，…识别出该段文本中“苹果”指的是苹果公司，而不是苹果 (水果) 或者苹果 (电影)；
3. 不可链接提及预测。考虑到知识的规模和更新速度，知识库往往不能覆盖所有真实世界实体。为了解决上述问题，需要识别出知识库尚未包含其目标实体的提及，并将这些提及按其指向的真实世界实体进行聚类。例如，由于现有知识库没有包含上文中提及“Swift”指向的目标实体 Swift（编程语言），实体链接系统需要将“Swift”的目标实体设置为空实体“NIL”，表示该提及在知识库中没有链接对象。

研究内容 1 和 2 解决实体多样性问题，即同一提及可对应不同实体，是实体链接问题最大的挑战。针对多样性问题，研究内容 1 采用候选实体生成（Candidate Entity Generation, CEG）方

法，从提及出发，找到知识图谱中所有可能的实体，组成候选实体集。研究内容 2 采用实体消歧（Entity Disambiguation, ED）方法，从候选实体集中，选择最可能的实体作为预测实体。

## 1.3 候选实体生成

候选实体生成是确定文本中的实体提及可能指向的实体集合。生成实体提及的候选实体有三种方法 [1]：

1. 词典匹配。词典匹配是候选实体生成最主流的方法。字典是一个 <key,value> 集合，key 表示一个提及，对应的 value 表示该提及所有指向的实体。例如，key= “苹果”，value=[ “苹果公司”，“水果苹果”，“苹果电影” ]。在使用时，将实体提及与词典中的 key 匹配，返回匹配成功的 key 对应的 value 集合;
2. 表层名字扩展。某些实体提及是缩略词或全名的一部分，因此可通过表层名字扩展技术，从实体提及出现的相关文档中识别可能的扩展变体（如全名），利用这些扩展变体从词典中生成候选实体集合;
3. 基于搜索引擎的方法。将实体提及和上下文提交至搜索引擎，根据检索结果生成候选实体。

### 1.3.1 使用维基百科构建词典

用什么方法构建词典，取决于 EL 的使用场景。如果是某个具体的行业领域，就需要通过一些启发式的方法、用户日志、网页爬取，甚至人工标注的方法来构建词典。如果做百科问答或是通用文本的阅读增强，一般使用百度百科或维基百科构建。在百度百科中输入词条，搜索结果中会显示与该词条相关的词条，或者同义词，使用同义词链接即可构建字典。Wei Shen [1] 描述了如何使用维基百科的词条页面、重定向页面、消歧页面、词条正文超链接构建词典。

维基百科每个页面称为词条页面，词条页面描述的对象通常被当作知识库中的实体，词条页面的标题作为 key，页面描述的实体词作为 value。例如，词条 “Microsoft’’ 页面标题是 Microsoft，因此得到 {Microsoft: Microsoft}。

重定向页面指向维基百科中存在的页面。重定向页面的标题作为 key，其所指向页面的词条作为 value。例如，词条为 “Microsoft Corporation’’ 的页面重定向为 “Microsoft’’ 页面，所以得到 {Microsoft:Microsoft Corporation}。

多个实体存在同名情况，这些实体会在一个消歧页面说明。消歧页面标题可作为 key，页面中列出的词条实体作为 value。例如，Michael Jordan 的消歧页面[1]，在这个页面中列出了 13 个名字同为 Michael Jordan 的人物，将 Michael Jordan 作为 key，其他人物的超链接标题作为 value。

词条页面的第一段中可能会有加粗词，这些词通常是页面标题的全称或简写。这些加粗字体可作为 value，页面标题作为 key。

维基百科页面中的超链接是以 [[实体 | 实体提及]] 的格式标记的，因此处理所有的超链接可以提取实体和实体提及的对应关系。将超链接中的文本作为 key，超链接指向的页面中的标题作为 value。例如在词条 “Hewlett-Packard’’ 页面中，锚文本 “Bill Hewlett’’ 存在超链接，超链

[1]https://en.wikipedia.org/wiki/Michael_Jordan_(disambiguation

接指向的页面标题为 “William Reddington Hewlett’’，因此得到 {Bill Hewlett: William Reddington Hewlett}。

使用上述四种方法得到的词典格式是实体：实体提及，根据这些词典可得到实体提及：实体格式的词典，用于生成候选实体。

### 1.3.2 表层名字扩展

针对实体提及为缩写或者是实体子字符串的情况，可使用名字扩展方法，这些方法可分为启发式方法和监督学习方法。

启发式方法是寻找一些规则。在一些文本中，缩写词后面可能紧跟一个括号，括号中给出了缩写的全称，例如 UIUC(University of Illinois at Urbana-Champaign)，或者是缩写在括号中 Hewlett-Packard(HP)，这种情况可根据规则提取全称。

还可以使用 NER 工具，当实体提及存在于 NER 识别的命名实体中时，可将这个命名实体作为该实体提及的全称。

### 1.3.3 基于搜索引擎

Zhang Wei [2] 借助 Wikipedia 的检索和拼写检查功能生成候选实体。检索功能可以返回和实体提及相关的实体页面，在 Wikipedia 中搜索一个关键词可以返回相关的实体页面。拼写检查功能可以纠正拼写错误的实体提及，将纠正后的词作为候选实体。在 Wikipedia 中 “did you mean’’ 信息表示拼写纠正后的搜索关键词。Zhang Wei [2] 使用的算法如图 1所示，当词典法返回的候选实体集合 Ref(s) 为空时，则使用上述算法，相似度计算使用最长公共子串长度。

**Algorithm 1** Candidate Set Generation

**Input:** mention *s;*

1: **if** $Ref_E(s)$ is empty
2: *s'* ← Wikipedia“did you mean”Suggestion
3: **If** s' is not NULL
4: s ← *s'*
5: **else**
6: EntityPageList ← WikipediaSearchEngine(s)
7: EntityPage←FirstPage of EntityPageList
8: Sim=Similarity(s,EntityPage.title)
9: **if** Sim > Threshold
10: *s* ← *EntityPage.title*
11: **end if**
12: **end if**
13: **end if**

**Output:** $Ref_E(s)$;

**图 1:** 算法伪代码

Xianpei Han [3] 使用 Google API 搜索实体提及和上下文，将结果中返回的 Wikipedia 链接页

面的标题作为候选实体。例如搜索 “Macau’s fledgling airline, Air Macau’’，返回的 Wikipedia 链接如图 2所示，将 “Air Macau’’ 和 “AirAsia’’ 加入到候选实体集合中。


**图 2:** Google 搜索结果

## 1.4 候选实体排序

候选实体排序（候选实体消歧）是对候选实体集进行排序，选取得分最高的候选实体作为实体提及指向的实体。候选实体消歧方法主要包括基于图 [4]、基于主题模型 [5] 和基于深度学习方法 [1, 6-7]，这些都是监督学习方法。

基于图的方法将所有的实体构建一个图，具体来说将实体提及和候选实体作为图的节点构建无向图，边有两类：实体提及—候选实体、候选实体 A—候选实体 B。每个边拥有权重，权重使用前面介绍的特征做组合或者是使用单一特征，利用相似度计算方法得到该值。最后利用 PageRank 算法即可求得重要性的排序，最终得到候选实体。

基于主题模型的观点是一个文本中出现的实体应该与文本表述的主题相关，因此可以根据候选实体与文本中所有实体的主题相关性对候选实体排序。

近年来很多深度学习方法应用在候选实体排序中。深度学习方法可以很好的建模实体提及的上下文。在长文本中实体提及可以获得丰富的上下文，而在问答系统中问题的长度通常很短，导致上下文不够丰富，短文本的候选实体排序难度更大。本文介绍近年来几种应用在问答系统中的候选实体排序方法。

### 1.4.1 深度学习方法

Daniil Sorokin [6] 使用不同细粒度的实体提及向量和候选实体向量对候选实体排序。具体地，使用 DCNN 网络将问题编码成向量 $o_s$，使用另一个 DCNN 网络将实体提及编码成向量 $o_n$ 以及将候选实体编码成向量 $o_l$。用 TransE 方法训练 wikidata 知识库，得到知识库中每个节点和关系的向量表示。由此得到候选实体的另一个向量表示 $o_d$，同时还得到与候选实体连接的关系向量表示 $o_r$ 和与候选实体连接的客体向量表示 $o_e$。此外，直接使用词向量得到关系的向量表示 $o_{rl}$。这些向量最终拼接得到向量 $o_c$ 和 $o_t$。

$$o_c = [o_d; o_l; o_{rl}; o_r; o_e]$$

$$o_t = [o_s; o_n]$$

$o_c$ 和 $o_t$ 分别经过一层全连接网络和激活函数，得到概率 $p_c$ 和 $p_n$。$p_n$ 表示实体提及识别正确的得分，只有当大于 0.5 时才会返回它的最佳候选实体。$p_c$ 表示候选实体得分。消融实验

说明 TransE 训练的节点和关系表示对模型影响最大。

Wu Ledell [7] 提出一种两阶段的短文本实体链接方法，是当时的最优方法，并可用于零样本场景。论文提出的方法适用于基于知识库的问答系统，将问题中的实体提及链接到知识库中，前提是问题中的实体提及已知。论文提出的方法示意图如图 3所示。

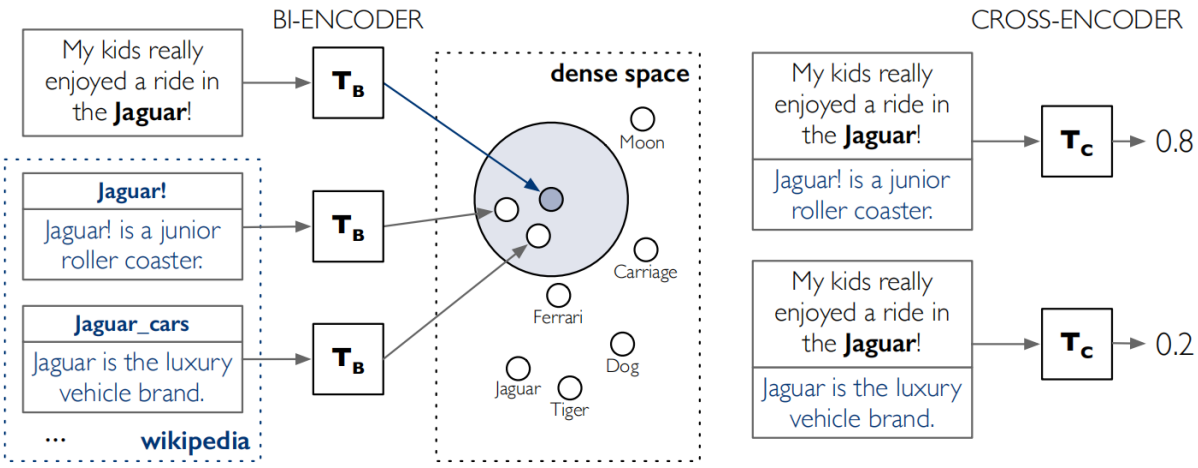


**图 3:** 模型框架

在第一阶段，使用 bi-encoder 分别对实体提及的短文本和候选实体描述文本编码，得到实体提及向量 $m$ 和候选实体向量 $e$。然后计算 $m$ 与每个候选实体向量 $e_i$ 的空间距离，使用最近邻搜索算法返回 K 个最近的候选实体。在第二阶段，使用 cross-encoder 计算每对 $(m, e_i)$ 匹配的概率，选择概率最大的 $e_i$ 作为 $m$ 的链接实体。

bi-encoder 和 cross-encoder 均使用 BERT 模型。实体提及和它的上下文输入到 bi-encoder 的格式为：

$$[CLS]\ ctxt_l\ [M_s]\ mention\ [M_e]\ ctxt_r\ [SEP]$$

$ctxt_l$ 表示 mention 的上文，mention 表示实体提及，$ctxt_r$ 表示 mention 的下文。候选实体输入到 bi-encoder 的格式为：

$$[CLS]\ title\ [ENT]\ description\ [SEP]$$

title 表示候选实体，description 表示实体描述文本。选取 BERT 的 [CLS] 位置输出作为各自的向量表示 $y_m, y_e$。候选实体 $e_i$ 与实体提及的距离为 $s(m_i, e_i) = y_m \cdot y_{e_i}$，训练时选取一个 batch 内的其他候选实体作为当前实体提及的负样本，损失计算如下式：

$$L(m_i, e_i) = -s(m_i, e_i) + log \sum_{j=1}^{B} exp\left(s(m_i, e_j)\right)$$

检索距离最近的前 K 个候选实体，与实体提及组成一对，这样就有 K 对实例作为第二阶段的训练样本。

在第二阶段，使用 cross-encoder 编码实体提及和候选实体。模型的输入是：

$$[CLS]\ ctxt_l\ [M_s]\ mention\ [M_e]\ ctxt_r\ [SEP]\ title\ [ENT]\ description\ [SEP]$$

选择 [CLS] 位置的输出向量 $y_{m,e}$，输入到 softmax 分类，$s_cross(m, e) = y_{m,e}W$，判断候选实体 $e$ 是否是实体提及 $m$ 的链接实体。

Wu Ledell [7] 提出的方法没有候选实体生成过程，直接将知识库中所有实体看作是候选实体集合。当知识库中实体数量很大时，搜索与实体提及最近邻的 K 个候选实体效率慢。Daniil Sorokin [6] 使用了知识库结构，即与候选实体连接的关系和客体，而 Wu Ledell [7] 只使用了实体描述文本。Wu Ledell [7] 的方法在 TACKBP-2010 数据集上取得 SOTA。

### 1.5 端到端方法

端到端方法联合了实体提及识别、候选实体生成和候选实体排序过程。

Chen Haotian [8] 提出一种端到端的实体链接模型，联合学习实体提及识别和实体消歧任务。实体提及识别看作是序列标注任务，实体消歧是对候选实体排序。论文事先使用预训练模型对实体的 Wikipedia 描述文本训练得到该实体的向量表示。模型结构如图 4所示。

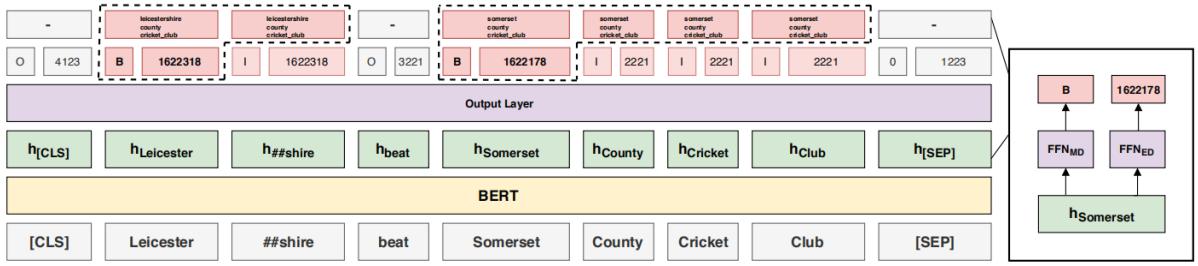


**图 4:** 模型框架

右边方框是整体结构，在 BERT 基础上有两个全连接网络 $FFN_{md}$ 和 $FFN_{ed}$，分别用于序列标注和候选实体排序。在 $FFN_{md}$ 中，计算过程如下：

$$m_{md} = W_{md}h + b_{md}$$

$$p_{md} = softmax\left(m_{md}\right)$$

$$\hat{y}_{md} = argmax\left\{p_{md}(i)\right\}$$

$W_{md}$ 表示 $FFN_{md}$ 网络参数，$h$ 表示 BERT 每个位置的输出向量，$\hat{y}_{md}$ 是预测标签。在 $FFN_{ed}$ 中，计算过程如下：

$$m_{ed} = tanh\left(W_{ed}h + b_{ed}\right)$$

$$p_{ed} = s\left(m_{ed}, E\right)$$

$$\hat{y}_{ed} = argmax_j\left\{p_{ed}\left(j\right)\right\}$$

$W_{ed}$ 是 $FFN_{ed}$ 网络参数，$h$ 表示 BERT 每个位置的输出向量，$E$ 是候选实体向量矩阵，$\hat{y}_{ed}$ 是排序得分最高的候选实体。模型损失函数是两者加权和。

$$J\left(\theta\right) = \lambda L_{md}\left(\theta\right) + \left(1 - \lambda\right) L_{ed}\left(\theta\right)$$

本文没有候选实体生成过程，直接使用现有的 {实体提及：候选实体} 字典。在实验中也测试了将知识库中所有实体看作候选实体的效果。在 AIDA/CoNLL 数据集上测试结果如图 5：

本文提出的方法取得了 SOTA。最后两行结果说明候选实体生成过程对模型性能影响很大，目前的方法还无法有效的从知识库所有实体中选择最佳链接实体，实体生成过程能有效的缩小候选实体范围，提升模型性能。

| | AIDA/testa F1 (val) | | AIDA/testb F1 (test) | |
|---|---|---|---|---|
| | Macro | Micro | Macro | Micro |
| Martins et al. (2019) | 82.8 | 85.2 | 81.2 | 81.9 |
| Kolitsas et al. (2018) | 86.6 | 89.4 | 82.6 | 82.4 |
| Cao et al. (2018) | 77.0 | 79.0 | 80.0 | 80.0 |
| Nguyen et al. (2016) | - | - | - | 78.7 |
| Broscheit (2019) | - | 76.5 | - | 67.8 |
| Poerner et al. (2019) | 89.1 | 90.8 | 84.2 | 85.0 |
| Fine-tuned BERT with candidate sets | **92.6**±0.2 | **93.6**±0.2 | **87.5**±0.3 | **87.7**±0.3 |
| Fine-tuned BERT without candidate sets | 82.6±0.2 | 83.5±0.2 | 70.7±0.3 | 69.4±0.3 |

**图 5:** 实验结果

Belinda Z. Li [9] 提出的方法将实体提及识别、候选实体生成和候选实体排序合并成一个端到端的过程。模型结构如图 6所示，模型由 Question Encoder 和 Entity Encoder 组成，两者均是 BERT 模型。Question Encoder 对问题编码得到问题中每个字符的向量表示，选取问题中每个片段 [i, j] 作为候选实体提及 $m_{ij}$，将该片段内的字符向量取平均作为候选实体提及的向量表示 $m$。Entity Encoder 对 Wikipedia 中的实体编码得到实体向量表示 $e$，输入是实体字符和实体的介绍文本。

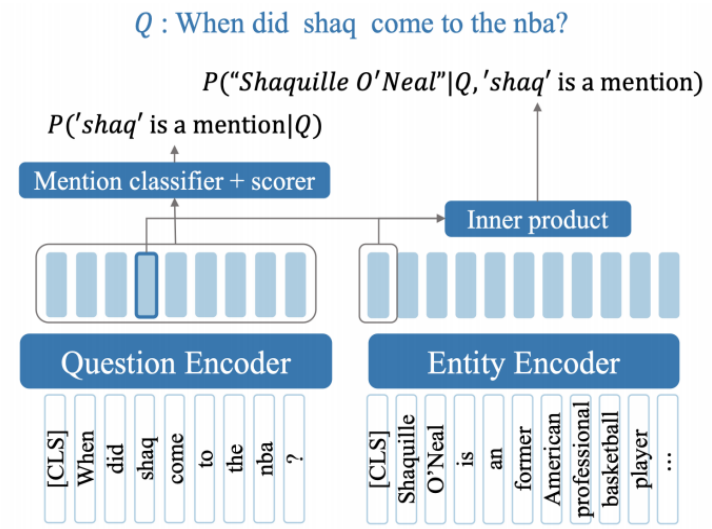


**图 6:** 模型结构

模型中没有候选实体排序过程，作者直接将 Wikipedia 中所有实体作为候选实体，因此只剩下实体提及识别和候选实体排序。实体提及识别对应图中的 Mention classify+scorer 模块，该模块计算在给定问题条件下片段 [i, j] 是实体提及的概率。候选实体排序对应图中 Inner product 模块，该模块计算在给定问题和片段 [i, j] 的条件下候选实体是待链接实体的概率。

给定输入问题 $q = q_1, \ldots, q_n$，首先使用 Question Encoder 获得问题中每个 token 的向量表示。

$$[q_1 \ldots q_n]^T = BERT([CLS]\ q_1 \ldots q_n\ [SEP]) \in \Re^{nxh}$$

将问题中每个片段 [i, j] 作为候选实体提及。为此，需要判断每个 token 是否是实体提及开

始、结束和中间部分，以此来计算片段 [i, j] 是实体提及的概率。

$$s_{start}(i) = w_{start}^T q_i$$
$$s_{end}(j) = w_{end}^T q_j$$
$$s_{mention}(t) = w_{mention}^T q_t$$
$$p([i,j]) = \sigma(s_{start}(i) + s_{end}(j) + \sum_{t=i}^{j} s_{mention}(t))$$

得到 p([i, j]) 即可计算实体提及识别的损失，

$$L_{MD} = -\frac{1}{N} \sum_{1 \le i \le j \le min(i+L-1,n)} (y_{[i,j]} log p([i,j]) + (1 - y_{[i,j]}) log(1 - p([i,j])))$$

然后计算每个候选实体的向量表示，候选实体来自 Wikipedia 中每个实体。本模型同样是没有候选实体生成过程的。

$$x_e = BERT_{CLS}([CLS]\ t(e_i)[ENT]\ d(e_i)\ [SEP]) \in \Re^h$$

$T(e_i)$ 表示候选实体 token，$d(e_i)$ 表示候选实体在 Wikipedia 中的摘要文本。候选实体提及的向量表示 $y_{i,j}$ 是将该片段内的字符向量取平均。得到 $y_{i,j}$ 和 $x_e$ 后即可计算候选实体 $e$ 是片段 [i, j] 的链接实体的概率。

$$s(e, [i,j]) = x_e^T y_{i,j}$$
$$p(e|[i,j]) = \frac{exp(s(e,[i,j]))}{\sum_{e' \in \varepsilon} exp(s(e',[i,j]))}$$

然后计算候选实体排序损失 $L_{MD} = -log p(e_g|[i,j])$。最终模型的损失由 $L_{MD}$ 和 $L_{ED}$ 组成。MD 表示 mention detection，ED 表示 entity Disambiguation。

| Training Data | Model | WebQSP$_{EL}$ | | | | GraphQ$_{EL}$ (zero-shot) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **Prec** | **Recall** | **F1** | **#Q/s** | **Prec** | **Recall** | **F1** | **#Q/s** |
| WebQSP$_{EL}$ | VCG† | 82.4 | 68.3 | 74.7 | 0.45 | 54.1 | 30.6 | 39.0 | 0.26 |
| | ELQ | 90.0 | 85.0 | 87.4 | 1.56 | 60.1 | 57.2 | 58.6 | 1.57 |
| Wikipedia | TAGME | 53.1 | 27.3 | 36.1 | **2.39** | 49.6 | 36.5 | 42.0 | **3.16** |
| | BLINK | 82.2 | 79.4 | 80.8 | 0.80 | 65.3 | 61.2 | 63.2 | 0.78 |
| | ELQ | 86.1 | 81.8 | 83.9 | 1.56 | 69.8 | **69.8** | 69.8 | 1.57 |
| Wikipedia + WebQSP$_{EL}$ | ELQ | **91.0** | **87.0** | **89.0** | 1.56 | **74.7** | 66.4 | **70.3** | 1.57 |

**图 7:** 实验结果

作者在两个 QA 数据集进行评测：WebQSP 和 GraphQuestions。采用了人工标注的方式标注出 entity mention 以及对应的 golden entity, 该标注数据集也已经给出[2]。测试结果如图 7所示。ELQ 是 Belinda Z. Li [9] 提出的方法，$\#Q/s$ 表示推理速度。在这两个数据集上本模型均取得 SOTA，但是推理速度稍差。在 Wikipedia 数据上训练可以显著提升模型性能。

[2]http://dl.fbaipublicfiles.com/elq/EL4QA_data.tar.gz

## 1.6 总结

早期的实体链接系统一般不包含实体提及识别过程，默认实体提及已在文本中标注。因此实体链接系统只有候选实体生成和候选实体排序过程。候选实体生成最普遍的方法是字典法，一般利用百科网站构建。候选实体排序是实体链接的重点，深度学习方法极大的提升了排序效果。在排序方法中，大多数方法 [7-9] 都使用实体描述文本建模实体向量表示，描述文本多来自 Wikipedia。也有少部分学者 [6] 结合了知识库的 schema 结构建模实体向量表示。因为问题长度一般很短，因此实体提及大多是选取问题的所有片段，从而省略了实体提及识别过程。实体提及向量可直接使用问题的句向量代替，但这种方式难以表达实体提及的含义。而直接使用实体提及的词向量又丢失了上下文信息。

我认为一种比较好的方法是从问题中提取属性名，例如问题 ‘‘广州塔的高度是多少？’’ 中存在属性名 ‘‘高度’’，使用 ‘‘广州塔’’ 和 ‘‘高度’’ 的词向量表示实体提及的向量。此外，还可以利用属性名过滤候选实体，将候选实体中不存在该属性名的排除。也有部分学者利用实体提及的类别过滤候选实体，但是判断实体提及的类别是一个细粒度的分类问题，再加上短文本分类效果通常不好。

近年来流行的是端到端方法，实际上就是简化了实体链接系统的三个过程。在短文本中，端到端方法都是采用文本分段方式自动生成实体提及集合，然后将知识库中所有实体看作候选实体，从而省略候选实体生成过程。因此端到端方法实际上只有候选实体排序过程。Chen Haotian [8] 在论文中的测试结果表明省略候选实体生成过程会导致模型性能显著下降。现阶段来看，端到端方法稍逊于两阶段方法。

# 参考文献


[1] WEI S, JIANYONG W, JIAWEI H. Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions[J]. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINNERING, 2015.

[2] WEI Z, JIAN S, CHEWLIM T, et al. Entity linking leveraging automatically generated annotation[C]//COLING. Beijing, China: [s.n.], 2010.

[3] XIANPEI H, JUN Z. Nlpr kbp in tac 2009 kbp track: A two stage method to entity linking[C]//TAC 2009 Workshop. [S.l.: s.n.], 2009.

[4] HOFFART J, YOSEF M A, BORDINO I, et al. Robust disambiguation of named entities in text[C]//EMNLP. Edinburgh, Scotland, UK.: Association for Computational Linguistics, 2011: 782-792.

[5] HAN X, SUN L. An entity-topic model for entity linking[C]//EMNLP. [S.l.]: Association for Computational Linguistics, 2012: 105-115.

[6] DANIIL S, IRYNA G. Mixing context granularities for improved entity linking on question answering data across entity categories[C]//NAACL. [S.l.: s.n.], 2018.

[7] LEDELL W, FABIO P, MARTIN J, et al. Scalable zero-shot entity linking with dense entity retrieval[C]//EMNLP. [S.l.]: Association for Computational Linguistics, 2020.

[8] HAOTIAN C, ANDREJ Z G, XI L, et al. Contextualized end-to-end neural entity linking[C]//AACL. [S.l.]: Association for Computational Linguistics, 2020.

[9] LI B Z, MIN S, IYER S, et al. Efficient one-pass end-to-end entity linking for questions[C]//EMNLP. [S.l.]: Association for Computational Linguistics, 2020.