持续学习与开放域知识库问答

Never-Ending Learning for Open-Domain Question Answering over Knowledge Bases

胡伟龙
huweilong@whu.edu.cn
2018 年 10 月 25 日

Contents

INTRODUCTION

前言

- 知识库问答的一个核心挑战是将自然语言问题翻译为 SPARQL 查询语言
- 目前的方法主要分为线下训练阶段和线上部署阶段

目前存在的两个问题

1. 线下训练阶段需要大量的标注数据集
2. 往往对未见过领域的问题回答失败

» 本文提出了 NEQA，一个用于 KB-QA 的持续学习框架。

动机

目前的 KB-QA 系统:

- 常用方法：通过语义分析，将用户问题翻译为 SPARQL 查询语句，然后在知识库中执行，获取答案
- 两个阶段：1）线下自动学习或手动创建模型；2）线上部署回答用户问题

三个主要缺点:

1. 需要大量标注数据，且涵盖各种用户问题的语法结构和词汇
2. 在部署之后，无法再进行学习
3. 受限于训练阶段的语料，对于未见过领域的问题回答失败

本文贡献

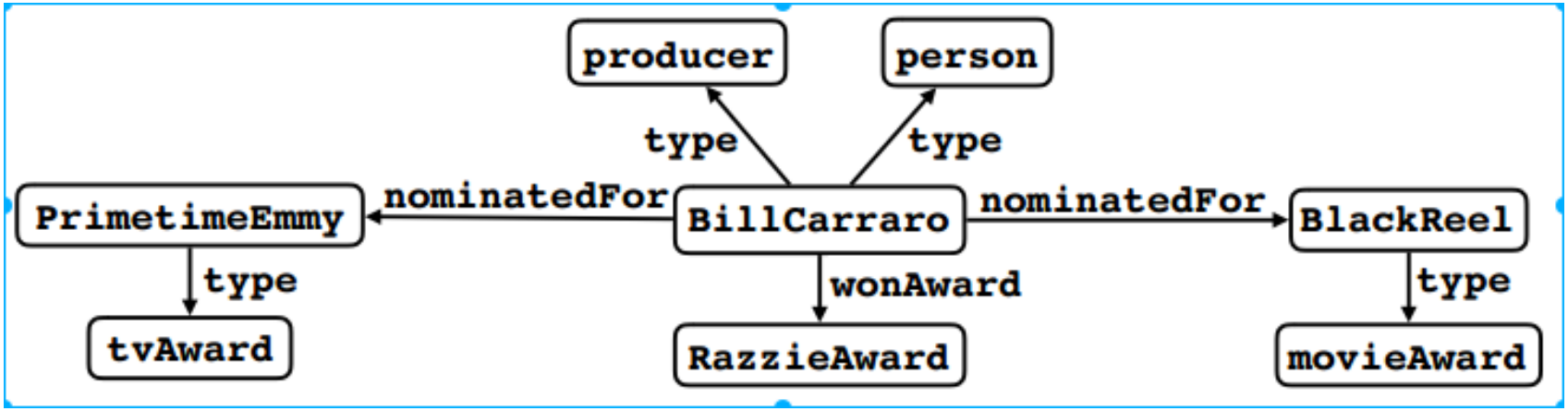
- 提出了新的 KB-QA 系统，能够从小部分训练数据开始，支持在回答问题过程中持续学习
- 提出了基于相似度函数的回答机制，能够回答未见过的问题(新的语法结构)，从而扩展了问题覆盖率
- 用户反馈模块明确对非专家用户询问满意答案，从而允许持续学习
- 两份数据上的实验表明该持续进化方法的有效性，和回答未见过领域问题的能力

SETUP

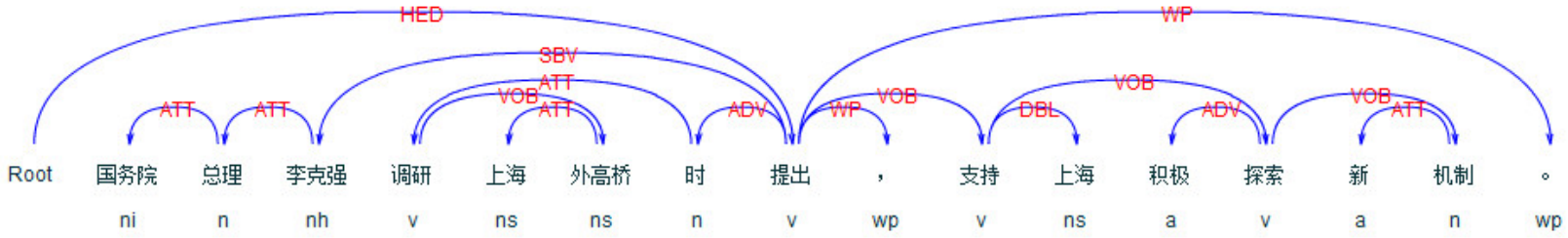
基本概念

知识库 (Knowledge Base)： 由图形式表示的事实 (Fact) 集合

- 节点 (node)
  - 实体 $e \in E$, 比如 BillCarraro
  - 类或者类型 $c \in C$, 比如 MovieAward
  - 字面量、常量 $h \in H$, 比如 date
- 边 (predict) $p \in P$, 比如 nominatedFor



句法分析 (Dependency Parse)：



句子的核心谓词为“提出”，主语是“李克强”，提出的宾语是“支持上海…”，“调研…时”是“提出”的 (时间) 状语，“李克强”的修饰语是“国务院总理”，“支持”的宾语是“探索 新机制”。有了上面的句法分析结果，我们就可以比较容易的看到，“提出者”是“李克强”，而不是“上海”或“外高桥”，即使它们都是名词，而且距离“提出”更近。

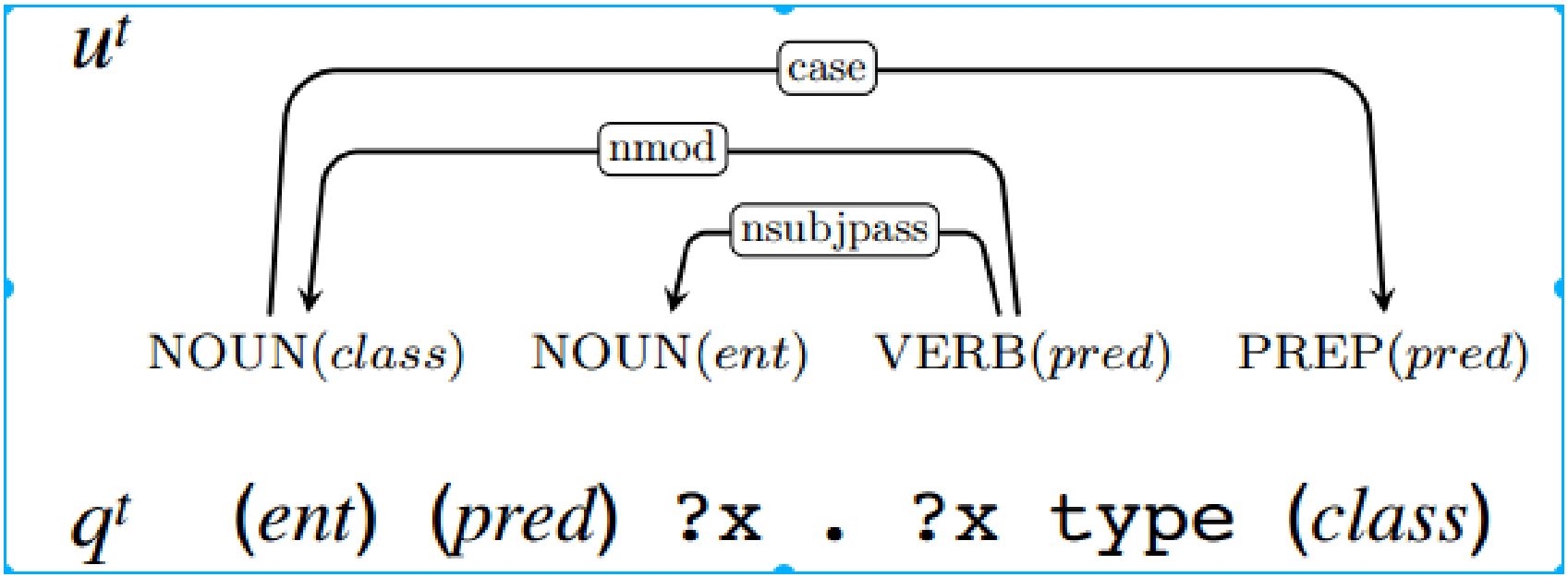
基本概念

问题、查询与答案 (Question,Query and Answer):

- Question
  - Which film award was Bill Carraro nominated for?
- Query
  - SPARQL triple pattern: (e.g. ?x type movieAward)
  - query: (e.g. BillCarraro nominatedFor ?x. ?x type MovieAward)
  - ?x 被指定为投影变量 (projection variable)
- Answer
  - 一个或多个知识库中的实体
  - 通过将 query 中的变量映射到知识库得到，e.g. BlackReel

问题和查询模板

模板(Template) 负责将以自然语言方式描述的问题中的语法结构映射为 SPARQL 的语义谓词参数 (predicate-argument) 结构



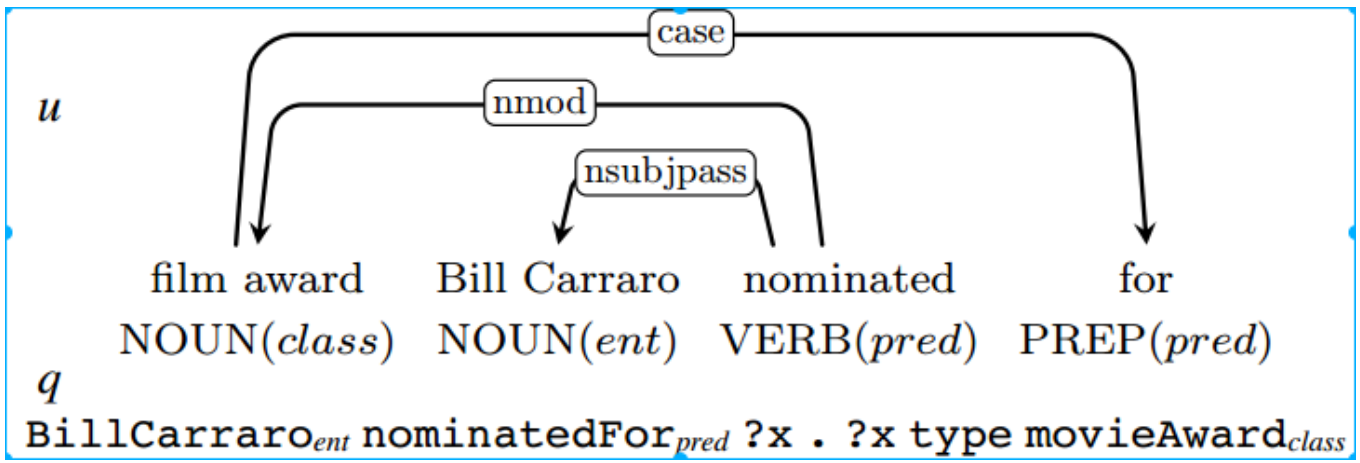
问题模板 (u^t) 和查询模板 (q^t) 的对齐是通过共享 ent,pred,class 完成的

问题和查询模板 生成模板

对于训练对 (u, A_u)，生成**问题查询对**(u, q). 例如：

u=Which film awards was Bill Carraro nominated for?

q=BillCarraro moninated ?x . ?x type MovieAward



- 使用Stanford Dependency Parser进行依存解析 u 中依存树节点对齐到 q 的语义项
- 带权词典 L 用于链接 u 中的词组和 q 中的候选语义项，对齐问题建模为整数线性规划问题

问题和查询模板 使用模板

使用模板的流程:

1. 新问题 u_{new} 到来
2. 通过问题的依存句法匹配问题模板
3. 使用对齐信息和词典实例化对应的查询模板
4. 对生成的多个查询使用 LTR(learning-to-rank) 排序
5. 将 top-ranked 查询的答案输出

问题 → 问题模板 → 查询模板 → 查询 → 查询排序 → 答案

谓词和类词典

谓词词典 L_P 和类词典 L_C：

- ClueWeb09-FACC1，利用 Freebase 标注的 500M 个网页
- L 中每个实体根据频率分配权重

定理（L_P 构造）

if $(e_1\ r\ e_2)$, assume exists $(e_1\ p\ e_2)$, then

add $r \mapsto p$ to L_P

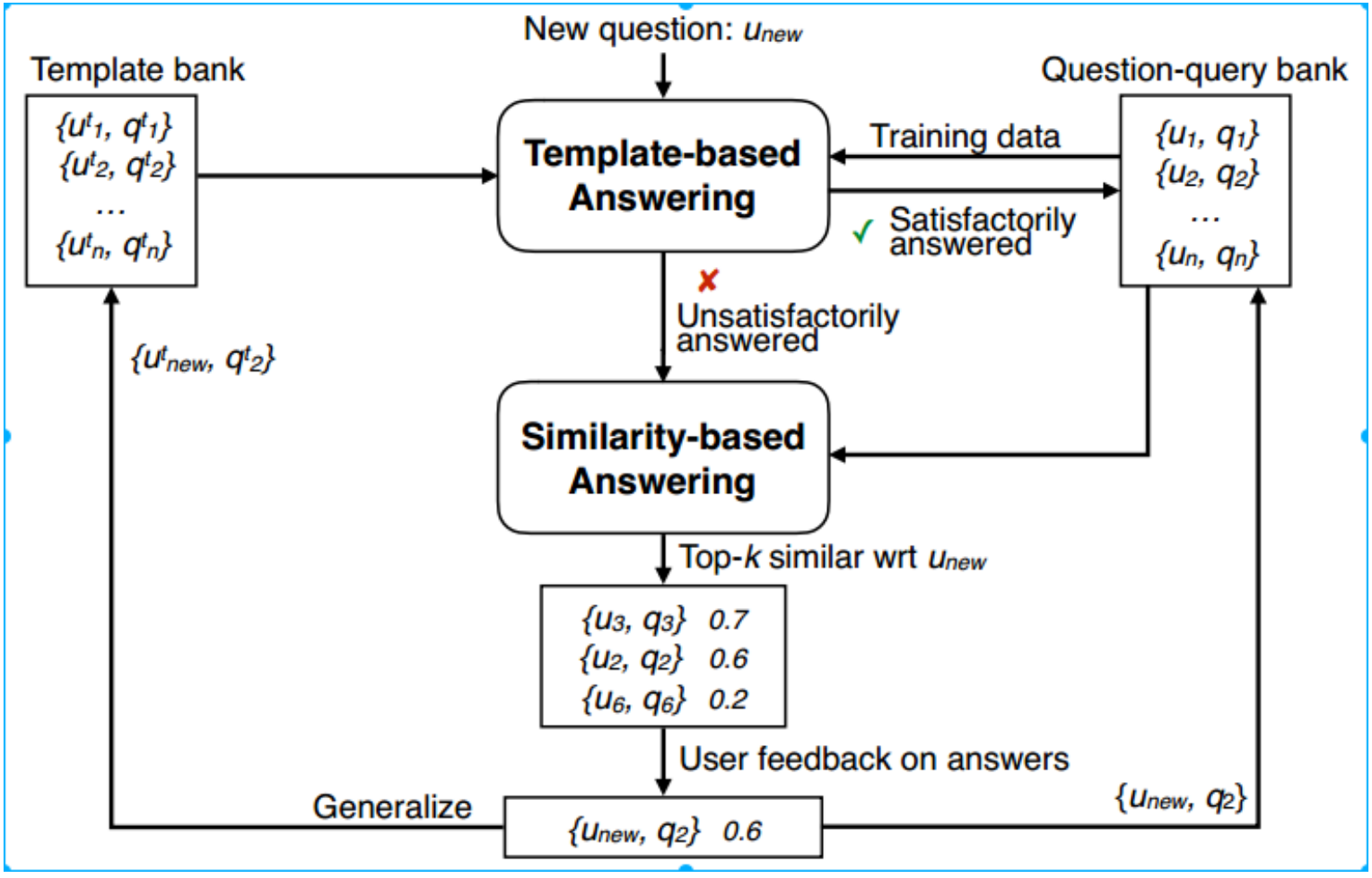
定理（L_C 构造）

if exists 'e and other np' then

each c $s.t.$ $np \mapsto c$ is added to L_C such that $(e\ type\ c) \in$ KB

NEQA FRAMEWORK

框架概览

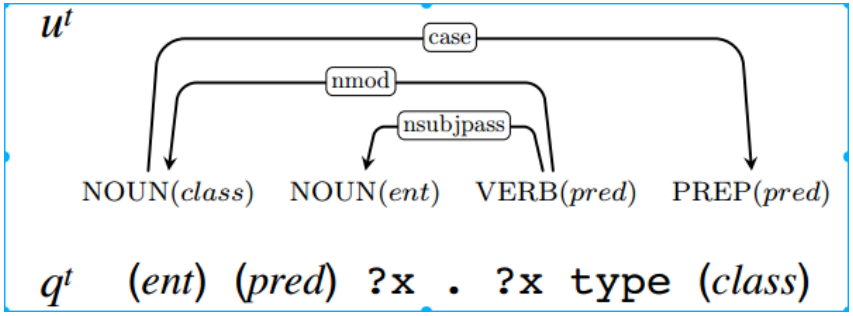


» 每个 batch 之后，NEQA 重新训练 LTR 排序模块以改善系统性能

通过模板问答

匹配：配对 u_{new} 和模板 $\{u_t\}$, 通过依存句法的 egde labels 和 POS Tag

"which president was lincoln succeeded by?" →



用户反馈：验证 $top-k$ 个 query 产生的答案

更新：生成新的问题查询对

↓

给出的答案集中至少有一个被用户采纳

通过相似函数问答

EXAMPLE

u_{new} = "what are the film award nominations that bill carraro received?"

↓

$u_{similar}$ = "which film awards was bill carraro nominated for?"

↓

q^* = "BillCarraro nominatedFor ?x . ?x type movieAward"

- 一旦匹配成功，将会产生新的 question-query 对(u_{new}, q^*)，然后产生新的模板对(u^t, q^t)
- 通过获取越来越多的模板，系统能够处理的问题 (对应于不同的语法结构) 越来越多

相似度函数 Similarity Function

无监督相似度函数，包含两个组件：

1. 基于语言模型的问题似然

$$score_{LM}(u_{new}, u_i) = \prod_{w \in u_{new}} [(1-\lambda) \cdot p_{ml}(w|u_i) + \lambda \cdot p_{ml}(w|C)]$$

2. 基于词嵌入的相似度

$$score_{w2v}(u_{new}, u_i) = \frac{1}{|\varphi|} \sum_{(w_j, w_k) \in \varphi} cos(w2v(w_j), w2v(w_k))$$

最终的相似度：

$$score_{sim}(u_{new}, u_i) = \alpha \cdot score_{LM}(u_{new}, u_i) + (1-\alpha) score_{w2v}(u_{new}, u_i)$$

用户反馈的利用

两种情况：

1. 使用模板回答:
   用户反馈用于评价问题和答案的关联度。通过评估答案质量，Question-query bank将会拓展
2. 使用相似函数回答:
   同时扩展Template bank和Question-query bank

新的问题：

获得用户反馈是否可行？如何量化用户反馈？

EXPERIMENTS

实验设置 Benchmarks

- WebQuestions(WQ)
  - 通过 Google suggest API 和众包创建
  - 3778 训练数据 + 2032 测试数据，问题答案对
- ComplexQuestions(CQ)
  - 集中于更复杂的多限制条件的问题
  - 1300 训练数据 + 800 测试数据

属性	WQ	CQ
train set 大小	300	105
dev set 大小	300	300
获得的初始模板	223	85

Train 集用于初始化：

- question-query 和 template 库
- Learning-to-rank(LTR) 模型
- 相似度函数的语言模型

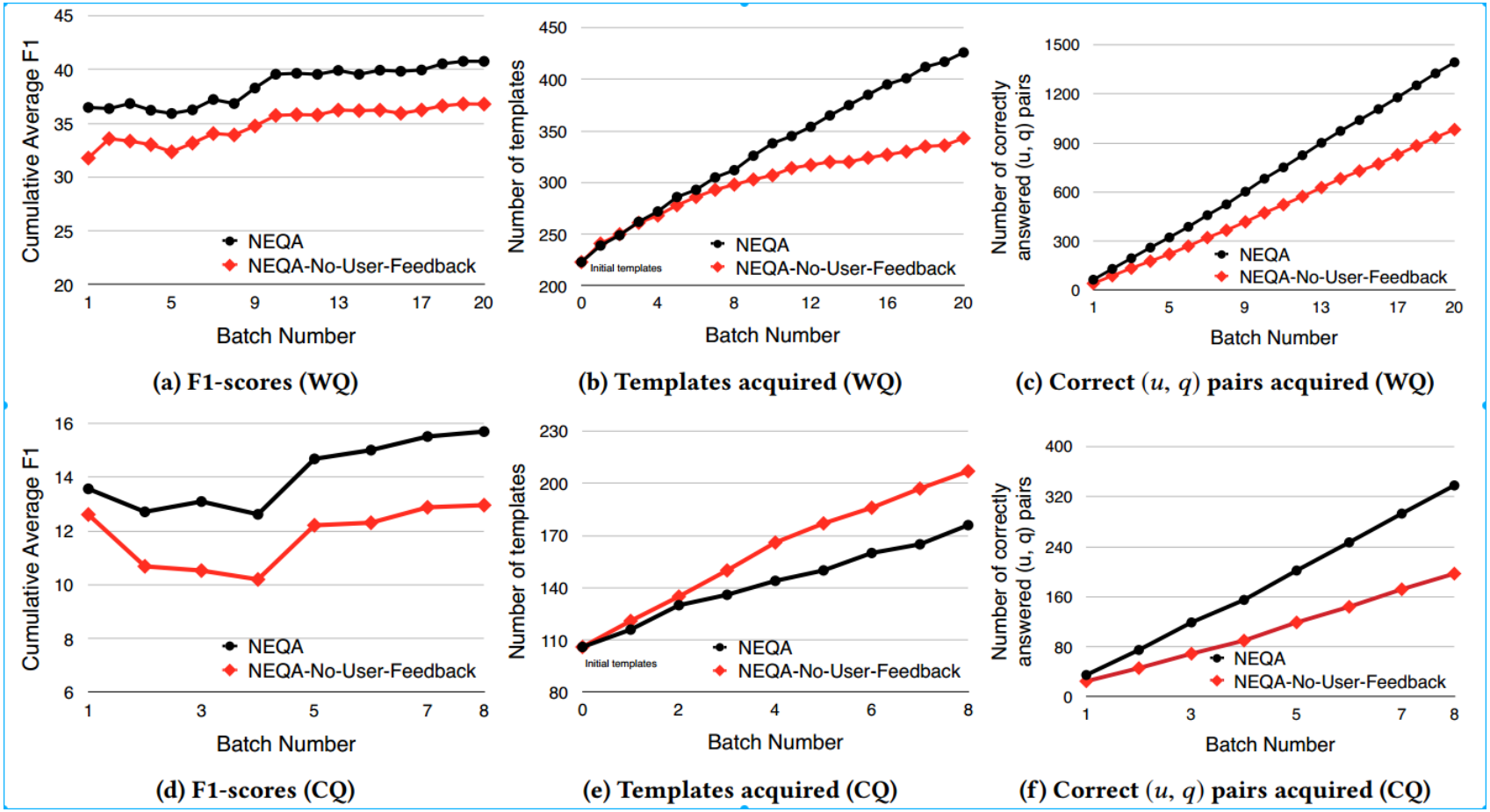
Dev 集用于调整 λ 和 α 参数

实验设置 User Feedback

用户反馈的影响：

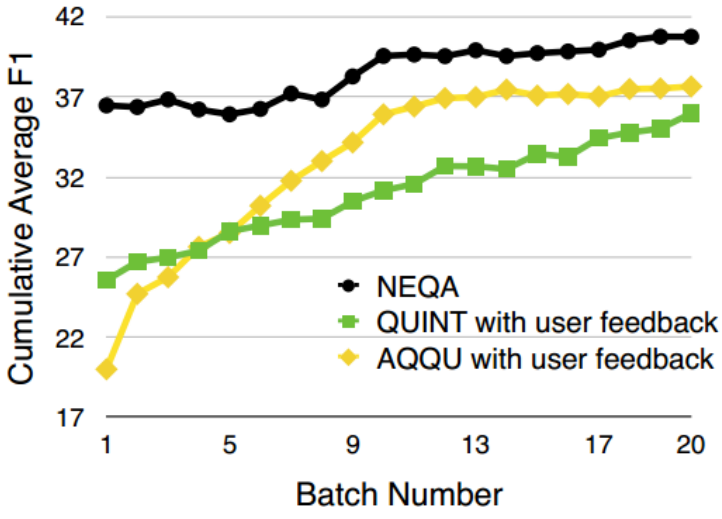
- NEQA (with user feedback)
  - 利用用户反馈从 $top-k$ 中挑选。如果没有，则利用相似度函数
  - 实验中使用答案标签模拟用户反馈
  - $k=5$，用户选择的答案数目
- NEQA-No-User-Feedback
  - 排名最前的答案作为正确答案
  - 使用模板的答案列表为空时，使用相似度函数

实验结果

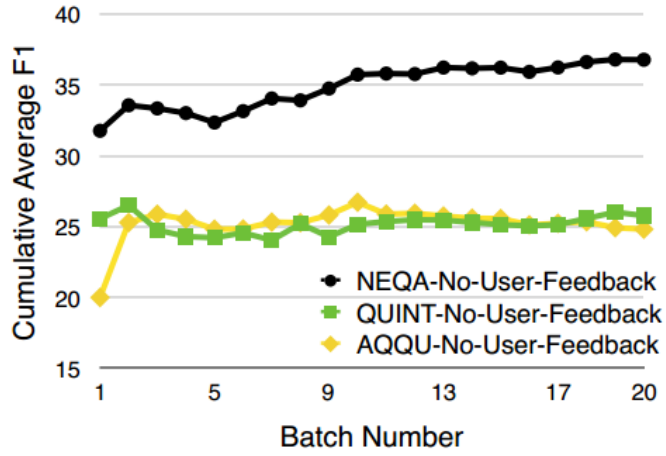


- Answering performance over time
- Augmentation of bank

实验结果



(a) Harnessing user feedback.



(b) No user feedback. The top-1 query is deemed correct.

Method	Avg. Prec.	Avg. Rec.	Avg. F1
QUINT [1] - No Feedback	25.5	30.2	25.7
QUINT [1] - Feedback	35.2	44.1	35.9
AQQU [6] - No Feedback	24.5	29.6	24.8
AQQU [6] - Feedback	36.3	45.2	37.6
NEQA-No-User-Feedback	36.6	45.4	37.0
NEQA	40.6	49.5	40.8

Method	Avg. Prec.	Avg. Rec.	Avg. F1
Berant et al. [9] (2013)	48.0	41.3	35.7
Yao and Van Durme [60] (2014)	-	-	33.0
Bordes et al. [13] (2014)	-	-	39.2
Bast and Haussmann [6] (2015)	49.8	60.4	49.4
Yih et al. [61] (2015)	52.8	60.7	52.5
Reddy et al. [40] (2016)	-	-	50.3
Savenkov et al. [42] (2016) (w/o text)	49.8	60.4	49.4
Xu et al. [55] (2016) (w/o text)	-	-	47.1
Abujabal et al. [1] (2017)	52.1	60.3	51.0
NEQA	52.1	60.3	51.0

实验分析

模板和相似度函数的影响：

- WQ with user feedback: 1184 + 848
- WQ without user feedback: 1788 + 244

失败情况：

- LTR 未学习到新的模板 (缺少合适的模板，字典不完整等)
- 相似度函数检索语义相似的问题失败，库中没有
- 检索出问题，但未生成模板 (词典或者 NERD 问题)

实验分析

对照实验：

Components	NEQA	NEQA-No-User-Feedback
Both	**40.8**	**37.0**
Only LM	38.3	35.1
Only word2vec	35.0	33.4

案例分析：

"what is the name of the currency used in italy?"
"what is the head judge of the supreme court called?"
"where did the battle of waterloo occur?"

图 1: via templates

Question:	*"what is the currency in [italy?]"*
Most similar:	*"what kind of money is used in [israel]?"*
Question:	*"what films has [scarlett johansson] been in?"*
Most similar:	*"what movies did [zoe saldana] play in?"*
Question:	*"what was [sir isaac newton]'s inventions?"*
Most similar:	*"what inventions did [robert hooke] made?"*

图 2: via similarity function

CONCLUSION

总结

总结:

本文提出了一个持续学习框架 NEQA 用于 KB-QA，结合了语法感知的模板，语义相似函数和非专家用户反馈。NEQA 以少量训练数据开始，利用错误案例改进性能，并通过相似函数学习未见过的语法结构对应的模板。实验表明，1)NEQA 的性能随着时间增加明显改善，2) 比静态方法有效,3) 能做到开放域问答。

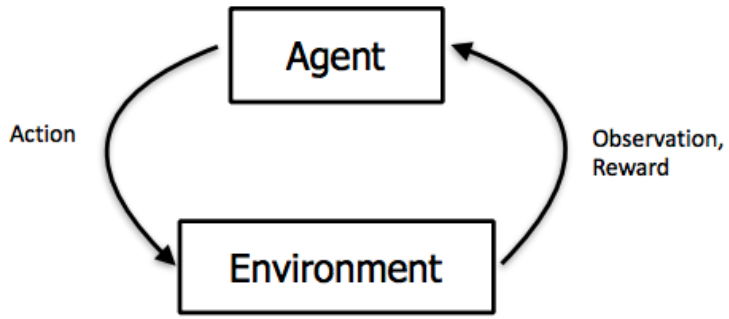
未来工作:

- 改进相似函数，处理更隐式的语义信息
- 避免直接的用户反馈

思考

1. **持续学习：**

与用户交互 → 与环境交互 →



2. **用户反馈：**

主动反馈 → 量化用户行为 →