Weakly-supervised Relation Extraction by Pattern-enhanced Embedding Learning

1. Introduction

1.1 关系抽出的分类

在这里首先作者对关系抽出做了个分类. 这也是这篇论文除了本身提案外很有价值的一点, 给像我一样圈外人一个概述, 知道了这个方向下更详细的分类.

1.1.1 Sentence-level relation extraction

从单个句子中抽出给定实体. 在这个研究里面,

介绍了三个研究, 只有一个是最新研究, 其他都是十年前的. 这个研究是非常有名的.

Cotype: Joint extraction of typed entities and relations with knowledge bases.

1.1.2 Corpus-level relation extraction

从语料库的多个句子中抽出固定实体的关系. 可以在语料库等级上使用weakly-supervised methods.

这个里面就有之前介绍过的2015年的PCNN文章.

大概思想就是, 根据几个已经确定的实例, 将其作为种子从语料库中找到更过实例, 这个过程中引入了冗余信息. 这是没有办法的事情.

这个研究的定位在于 Corpus-level relation extraction.

- **基于目标任务的分类**

这样被抽出的新的实例集和之前的实例集就可以被用于很多 downstream 任务.

比如说,

- Kowledge base completion

  Relation extraction with matrix factorization and universal schema(2013)

  Representing text for joint embedding of text and knowledge base(2015)

  Knowledge graph completion via complex tensor factorization(2017)

- Corpus-level relation extraction

  Neural relation extraction with selective attention over instances.(2016)

  Distant supervision for relation extraction via piecewise convolutional neural networks(2015)

  (这两篇都介绍过, 看来之前介绍的三篇论文都属于这个范围, 这个和上面的有什么区别吗?)

- hypernym discovery(上位语挖掘)

  Improving hypernymy detection with an integrated path-based and distributional method(2016)

- Synonym discovery(同位语挖掘)

  Automatic synonym discovery with knowledge bases.(2017)

Kowledge base completion 和 Corpus-level relation extraction到底什么区别?

- **基于方法的再分类**

从模型的方法去看, 这个类别下的方法可以分为两类.

- Pattern-based approaches

  通过分析包含两个实体的句子来预测这两个实体的关系.

  一些比较老的研究使用模式匹配来做预测, 但是这种方法对于词汇很敏感, 使得特征比较稀疏, 不利于抽出. 在此之上有了基于神经网路的利用向量的近似度来替代编码文本模式的方法. 但是这种方法也需要大量的标注文本.

- Distributional approaches

  这种方法中, 对于两个实体关系的预测并不是直接通过句子来实现的, 也就是说直接输入并不是句子, 而是实体的向量. 这个过程中, 要使用语料库去编码每个实体的词向量. 接下来通过神经网络去预测实体词向量其之间的关系. 但是这种方法同样需要大量的语料.

  > 虽说,编码实体向量也需要句子的信息, 但并不是直接使用句子去预测关系.

  基本方法有有监督学习和无监督学习两种方式:

  - 无监督学习:

    就是只利用语料库而不利用知识图谱的方法.

    Line: Large-scale information network embedding.

    Glove: Global vectors for word representation.

    > 居然还有 Glvoe, 看来是单纯的利用词向量去预测的方法.
    >
    > 在这里说不定可以用上我说的方法.

  - 有监督学习:

    结合语料库和知识图谱来进行编码. 然后联合训练实体词向量和关系预测模型.

    Knowledge graph and text jointly embedding(2014)

    Rc-net: A general framework for incorporating knowledge into word representations.(2015)

1.2 模型的结合

1.2.1 出发点

上面的 Distributional approaches 和 Pattern-based approaches 的出发点不同, 因此可以利用两个模型的互补来提升效果.

现在已经有一些类似的想法, 但是他们都只是 joint 训练两种模型而已, 只是同时用到了两种方法, 而没有使这两种方法进行互补. 而这就是这篇文章的出发点.

本文所提方案的模型是: REPEL (Relation Extraction with Pattern-enhanced Embedding Learning)

从名字来看, 基于分布的方法在这个方法中占主要地位.

1.2.2 具体实现

1.2.2.1 Notation

先对一些重要的概念进行定义:

- **实体向量对**, (e_h, e_t) .
- **Instance** : 已知relation 为 r 的 实体向量对 : (e_h^r, e_t^r) .也就是说, 关系+实体向量对=instance.
- **Instances** : 即关系为 r 所有 instances : $\{(e_h^{r(k)}, e_t^{r(k)}, r)\}_{k=1}^{N_r}$.

  其中, N_r 是语料库中关系为 r 的instance的数量.
- **Seed instances** : 即训练集中所有的 **Instances**.

1.2.2.2 模型概览

这个方法包含两个module, 一个是 基于pattern的, 一个基于分布的.如下图所示:

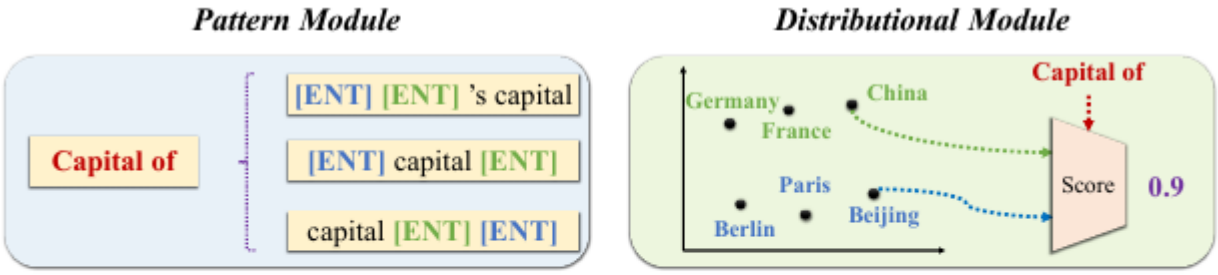

Figure 2: Illustration of the modules. The pattern module aims to learn reliable textual patterns for each relation. The distributional module tries to learn entity representations and a score function to estimate the quality of each instance.

- **pattern**

  基于pattern的模型的目标是 **找到 set of reliable textual patterns**.

  前向预测 : 接受的是一个 **Relation** 和 **corpus**, 输出是 **instances**

  反向训练 : 训练数据是一些 **seed instances**

- **distribution**

  基于分布的模型的目标是 **训练一个基于实体词向量的分类器**

  前向预测 : 接受的是一个 **Instance** , 输出是这个 **Instance** 的可信度

  反向训练 : 训练数据是 **seed Instances** .

这里使用了一个叫做 co-training.

- **generator**

  设置 pattern module为 generator. generator 的功能是, 输入值为语料库, 根据模型中发现的可信赖的 pattern 来发觉候选的 instance.

- **discriminator**

  设置 distributional module为 discriminator, 其功能是, 去评价一个实例(instance)是否可靠.

1.2.2.3 具体训练过程:

具体分为两个子进程, 操作如下:

- discriminator 对一些 **instance** 进行评价, 这个 instance 是由 generator 挑选出的. 然后, 评价的结果作为信号传给 generator, 作为其进行训练的 **反向训练数据** 进行训练..
- genertor 提供一些高可信度 **instance** , 这些 instance 用来训练 discriminator.

虽然上面说的是子进程, 但是两者并不是同时运行, 而是交替迭代运行的.

1.2.3 实验部分

这个模型在

- knowledge base completion with text corpora 和
- corpus-level relation extraction.

两个任务上进行了实验, 他说很棒.

2. Problem definition

- **实体识别**

  这个部分都是使用的现成工具, 分为了实体识别(NER)和指代消解两步.

- **找到关系实例**

  也就是找到含有两个实体的句子.

- **Problem Definition**

  根据已经有的信息去从语料库中抽出更多的, 确定关系标签的实体对, 如下:

> *Definition 2.1.* **(Problem Definition)** ***Given*** *a text corpus D and some target relations R, where each target relation r is characterized by a few seed instances $\{(e_h^{r(k)}, e_t^{r(k)}, r)\}_{k=1}^{N_r}$ or in other words a few seed entity pairs $\{(e_h^{r(k)}, e_t^{r(k)})\}_{k=1}^{N_r}$, the weakly-supervised relation extraction task* ***aims to*** *extract more instances $\{(e_h^{(i)}, e_t^{(i)}, r^{(i)})\}_{i=1}^{M}$ from the corpus. In other words, we aim at discovering more entity pairs $\{(e_h^{r(i)}, e_t^{r(i)})\}_{i=1}^{M_r}$ under each target relation $r \in R$.*

3. THE REPEL FRAMEWORK

3.1 框架的概览

这个在1.2 中说过了.

现在来公式化:

$$\max_{P,D} O = \max_{P,D} \{O_p + O_d + \lambda O_i\}.$$

其中, D 代表的是 discriminator 中的参数, 包括 实体词向量和 损失函数.

P 代表的是pattern 的参数, 包括数个可靠的模式.

O_d : discrimintor 的目标函数, 即接受由pattern module 生成的 「指定关系(label)的实体对(input)」 来进行监督学习.

O_p : pattern 的目标函数, 即接受由discrimintor提供的 「实体对(input)关于其关系为R的可信度值」, 进行关于自身有关pattern的参数训练.

O_i : 是两者的交互的目标函数.

下面开始讲解具体的模型, 不过为了简单化, 这里只考虑一种 relation 的情况.

3.2 Pattern Module

这里使用了两种pattern的策略, 一种是基于路径的, 一种是基于属性的.

基于路径的说的是基于依存句法关系和词汇的结合组成的模式.

基于属性的说的是基于实体附近的词汇.

这里暂时不需要深究, 只需要知道, 模型中提供了大量可供选择的pattern.

- **模式正确率**

我们假设某一个 pattern 为 π, 那么根据这个 π, 我们通过语料库可以抽取出多个 **instances**. 记录这些 **instances** 为 $G(\pi)$, 而在训练数据集中, 我们知道关系为 π 的 **实体向量对** 组成的 **instances** 为 S_{pair}. 那么, 这个模式下的预测正确率就是 $R(\pi)$:

$$R(\pi) = \frac{|G(\pi) \cap S_{pair}|}{|G(\pi)|}$$

- **模型目标函数**

  超参数K是指, 使用模式正确率最高的K个模式作为这个模型使用的模式, 设这K个正确率最高的模式的集合为 P. 那么, 模型正确率, 也就是模型的目标函数就是:

$$O_p = \sum_{\pi \in P} R(\pi),$$

- **利用pattern模型预测的方法**

  虽然感觉有点奇怪,这里根据K个模式结构来取模型的结果的方式是取并集:

$$G(P) = \cup_{\pi \in P} G(\pi)$$

3.3 Distributional Module

这里的用的不是接受两个输入然后用神经网络去预测的方法. 是一个较为复杂的方法, 先给出最后的目标函数, 其中系数是为了维持两个模型的平衡需要手动调:

$$O_d = O_{text} + \eta O_{seed},$$

一旦模型收敛, 就可以利用这个目标函数去判断一个 **instance** 的可信度.

- **上下文目标函数** O_{text}

  这里的出发点是, 相近的词汇具有相似的向量. 也就是用这个来修改各种词向量.

这里的采用的并不是简单的共现的方法, 而是用的二分网络. entities和普通词汇, 两部分之间可以有链接, 各部分不可以有连接.

> 感觉这样会少很多信息, 和词向量差远了

O_{text} 的计算方法是:

$$P(w|e) = \frac{\exp(\mathbf{x}_e \cdot \mathbf{c}_w)}{Z}, O_{text} = \sum_{w,e} n_{w,e} \log P(w|e),$$

其中, x_e 是实体向量. c_w 是词向量.

- **relation目标函数 O_{seed}**

  我们知道, 分布模块的输入是 **seed instances**, 也就是所有真实的数据对.

  其评价一个instance的可信度的函数是 :

  $$L_D(f|r) = 1 - ||x_{e_h} + y_r - x_{e_t}||_2^2.$$

  用的是欧式距离, 越正确, 值越高, 最高为1(完全可信).

  那么,最后的目标函数就是 seed instance 中的所有的可信度的和, 应该最大化.

  $$O_{seed} = \sum_{f \in S_{pair}} \sum_{f'=(e'_h, e'_t)} \min\{1, L_D(f|r) - L_D(f'|r)\}.$$

3.4 Modeling the Module Interaction

到目前为止, 两个模型的训练还都是靠 **Seed instances**. 接下来就要co-training 两者. 这个是两者结合的目标函数.

$$O_i = E_{f \in G(P)}[L_D(f|r)]$$

很有意思的地方是, 就这一个式子就包含了 $generator \rightarrow discrimintor$ 和 $discrimintor \rightarrow generator$ 两个部分.

其中 $generator \rightarrow discrimintor$ 的部分比较好理解, 就是在 $f \in G(P)$ 的部分, 告诉我们 discrimintor的训练数据来自generator. 但是同时, 如果 P 的质量如果不好, 这个式子的值也不会高. 而 P 是由 seed instances 训练得到的.

这个 seed instances 就由原始的数据集和discrimintor预测的高分数据产生 (这部分没有在式子中体现出来, 但是在实际操作过程中实际存在).

4. THE JOINT OPTIMIZATION PROBLEM

具体的优化部分等到有必要复现的时候再看, 但是说实话, 个人总感觉这个方法听起来很帅, 实现起来可能有些坑, 毕竟基于很多人工特征.

5. Experiment

5.3. Evaluation Setup.

(1) Knowledge Base Completion.

这个就是将知识图谱中的头实体或者尾实体去掉之后, 利用语料库进行预测.

(2) Corpus-level Relation Extraction

这个是根据句子来预测关系.例如之前的三篇文章.

感觉两个任务在本质上并没有分别.

看结果:

Algorithm	Wiki + Freebase						NYT + Freebase					
	P@50	R@50	F1@50	P@100	R@100	F1@100	P@50	R@50	F1@50	P@100	R@100	F1@100
Snowball [1]	58.00	22.14	32.05	65.00	49.62	56.28	20.00	4.50	7.35	21.00	9.46	13.04
PATTY [23]	60.00	22.90	33.15	61.00	46.56	52.81	28.00	6.31	10.30	20.00	9.01	12.42
CNN-ATT [16]	26.00	9.92	14.36	22.00	16.79	19.05	24.00	5.41	8.83	29.00	13.06	18.01
PCNN-ATT [16]	58.00	22.14	32.05	36.00	27.48	31.17	46.00	10.36	16.91	26.00	11.71	16.15
PathCNN [45]	36.00	13.74	19.89	38.00	29.01	32.90	42.00	9.46	15.44	26.00	11.71	16.15
LexNET [29, 30]	74.00	28.24	40.88	61.00	46.56	52.81	32.00	7.21	11.77	26.00	11.71	16.15
REPEL-D	14.00	5.34	7.73	17.00	12.98	14.72	6.00	1.35	2.20	7.00	3.15	4.34
REPEL-P	64.00	24.43	35.36	70.00	53.44	60.61	32.00	7.21	11.77	33.00	14.86	20.49
REPEL	**78.00**	**29.77**	**43.09**	**76.00**	**58.02**	**65.80**	**48.00**	**10.81**	**17.65**	**43.00**	**19.37**	**26.71**

这个是这篇论文的实验对照.

而在PCNN-ATT的模型中, 实验对照如下:

Test Settings	One				Two				All			
P@N(%)	100	200	300	Mean	100	200	300	Mean	100	200	300	Mean
CNN+ONE	68.3	60.7	53.8	60.9	70.3	62.7	55.8	62.9	67.3	64.7	58.1	63.4
+AVE	75.2	67.2	58.8	67.1	68.3	63.2	60.5	64.0	64.4	60.2	60.1	60.4
+ATT	**76.2**	65.2	**60.8**	67.4	76.2	65.7	62.1	68.0	**76.2**	68.6	59.8	68.2
PCNN+ONE	73.3	64.8	56.8	65.0	70.3	67.2	63.1	66.9	72.3	69.7	64.1	68.7
+AVE	71.3	63.7	57.8	64.3	73.3	65.2	62.1	66.9	73.3	66.7	62.8	67.6
+ATT	73.3	**69.2**	**60.8**	**67.8**	**77.2**	**71.6**	**66.1**	**71.6**	**76.2**	**73.1**	**67.4**	**72.2**

这篇论文用的是NYT数据集. 上面的 PCNN-ATT 应该对应的是下面的 PCNN+ATT(ALL)

但是……为什么结果差这么多, 差一点也就罢了….