CoType: Joint Extraction of Typed Entities and Relations with Knowledge Bases

提示 : 这篇文章与其看做是一个关系抽出任务, 不如看作是很多任务的结合体, 这篇论文是真的复杂, 工作量感觉是一般论文的几倍. 但是不知道效果怎么样, 如果只想看思想, 可以忽略各小节中的细节.

这篇文章中出现的数学公式和符号由于太多没有进行转换, 有兴趣的可以赋值下面链接中的内容到自己的电脑上,用markdown编辑软件查看.

1. Introduction

1.1 出发点

传统方法 : 大量的人工标注语料库 和 incremental pipeline.

> incremental pipeline : 指的是引入很多工具, 比如说实体识别, 句法标注等等. 这些含有误差的工具管道式连接在一起, 误差也逐渐放大.

这里提出的方案是 : 使用DS获取的数据联合的抽出entities and relations, 这里就避免了实体识别工具等等带来的误差累加. 也就是说, 相比于其他的模型(例如PCNN), 这里的优点在于一起抽出实体和关系, 利用了他们之间的约束.

这个任务的名字叫做: joint extraction of typed entities and relations with distant supervision.

> 那如果这样说, 现在的实体识别岂不是都不能用在下流任务中了? 都联合抽出就好了, 要什么有误差的工具? 这个地方有点无法理解. 不知道是不是多次一举.

1.2 domain-independent

(个人猜想)文中一直强调这是一个 domain-independent 的方法, 应该是说这个模型适用于各种各样的情况, 不会局限于某种特殊的领域的语料.

1.3 CoType 算法步骤

提前提醒, 这个模型的结构特别复杂, 一开始看这里根本就不看不懂, 不要灰心, 接着看就好了.

- 使用一个数据驱动文本分割算法去抽取出 entity mentions. 这里的数据驱动文本分割算法是指一类算法, 比如形态素分析算法也属于这一类算法.
- 利用将文本中的检测出来的实体, 再加上知识图谱的信息, 外部知识库的信息, 将一个句子可能呈现的实体和关系的组合状态列举出来.(详见2.2)
- 联合地对 entity mentions , relation mentions, text features, type labels 进行了向量化, 即赋予了其向量. 并且将其分别投射进了两个低维空间, 分别是 entity mention的相关空间和relation mentions的相关空间. (详见2.3)
- 然后用这些向量去估计 test (unlinkable) mention 的 type,关于unlinkable的说明在3.2.2中解释. 这一步就是说, 利用在语料库中训练得到的两个低维空间, 可以对一个新的句子进行分析, 得到他的 Entity type 和 Relation type. 详见2.1节.

1.4 学习向量的方法

公式化一个优化问题, 这个问题的输入为语料库和KB. 求解优化问题后得到上面的向量.

具体的话,

1) 采用了一个部分标注的损失函数(partial-label loss function)迎来应对有误差的标注数据. (顾名思义, 部分标注就是从标注数据中挑出一个使用来进行优化.)

2) 并且, 使用了一个 object "translation" function 来使得两个低位空间中的向量进行 cross-constraint(相互约束), 进而得到更好的效果. 这里的相互约束的 motivation 是, 在上面的两个低维空间中, Entity type 和Relation type 分属于两个空间, 但是两者之间也是应该存在一定的约束性的, 因此需要两个空间进行交互.具体见下面的分析.

1.5 一些其他的补充

额外知识, 不看也行

- **Weak supervision** : 指的是利用一部分高质量的标注数据作为引子, 在此基础上训练出一定质量的模型后, 引进一些预测效果很高的数据到数据集中, 扩充训练集数量.
- **Distant supervision** :

之前我的理解 : 将KB中拥有关系的entity作为索引, 在语料库中搜索. 将同时出现两个拥有entity的句子抽出, 假设这个句子含有这两个entity的关系信息. 将这样的句子加上关系标签, 加入训练集.

但是在这个任务中, 情况有所改变, 像Obama这样的 entity mentions 在KB中可能对应着多重Entity type, 因此需要:

- 先识别出语料库中的 entity mentions.
- 将可能的 Entity type 映射给句子中间两个 entity mentions.
- 找到这些查看这两个entity mentions可能的 Entity type的组合.

2. 模型框架和问题设定

2.1 概念和Notation

1) 概念的定义

Sentence instance (s) : 指数据中某个含有两个entity mentions的句子.

- **Entity 相关**

  m : *Entity mentions* ,具体指的是数据集中的代表实体的**字符串**, 是一种**具体的数据**."奥巴马"这三个字就是一个 Entity mentions

  e : *Entity instance* ,指的是, 知识库中的实体, 是一个概念 "奥巴马"这三个字表达的**概念**是一个实体,.

  y : *Entity type* , 这里对应的是KB(knowledge base)中的**概念**, 是**抽象的符号**. 例如, "奥巴马"可能对应的是KB中,人的概念. 也可能是总统的概念. 比如说, "奥巴马是一个person" 这句话中, "奥巴马"代表的就是一个entity instance, person就是一个Entity type.

- **Relation相关**

  $z = (m_1, m_2, s)$: *Relation mention* 其中 s 是句子, 代表的是数据集中的代表关系的**字符串**, 这里就比较特殊一点, 因为使用字符串去表示关系, 而关系是用句子表达的, 那么这里的 relation mention 就是包含两个entity mentions的句子.

$r(e_1, e_2, \ldots, e_n)$: *Relation instance* : 注意这里是有具体的e作为参数的, e不同代表不一样的Relation.

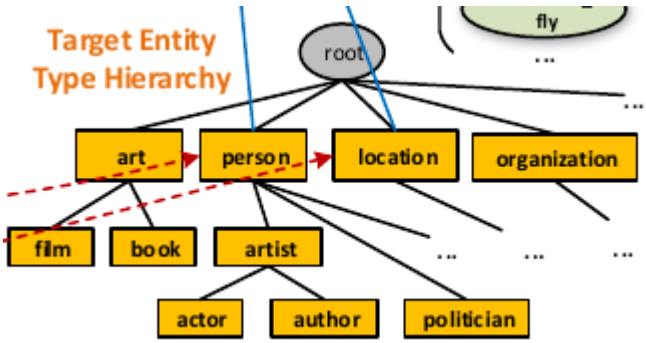
r : *Relation type* , 也是KB中的**概念**, 相当于谓词的感觉, 例如, "出生于"就可以是一种关系. 和参数无关.

- **Target Types相关**

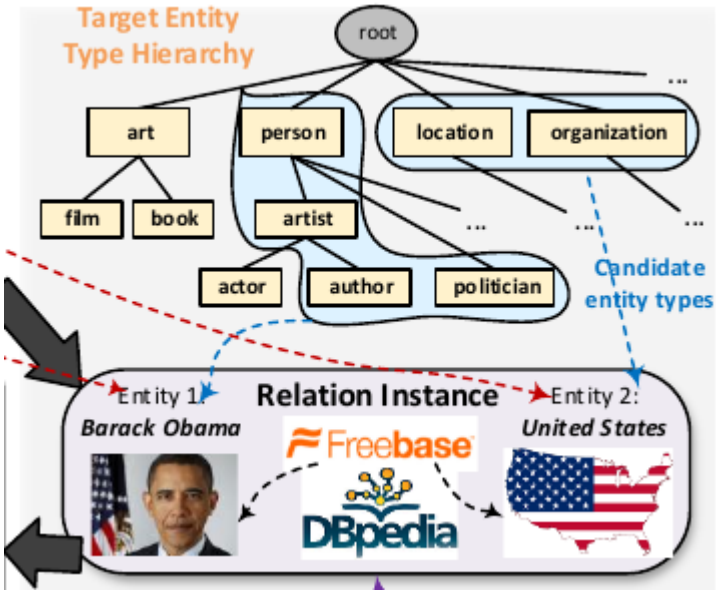
  y : *Entity type*, 上面讲过了.

  r : *Relation type* , 上面讲过了.

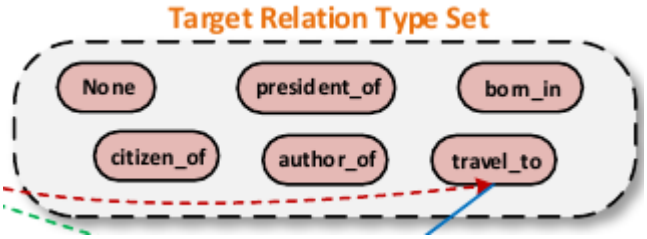
  $\mathcal{Y}$: Entity type 的**外部类型知识库**, 原名叫做**Target Types hierarchy Tree**, 这个知识库里的基本单位是 Entity type, 并且是有上下义的树状结构.如下图所示:



我们最后要预测的就是一个 Entity Mentions 在 其中的路径, 即这个Entity Mentions的真实 Entity Type. 一个Entity Mentions可能有多个Entity Type. 如下图的蓝色区域所示:



$\mathcal{R}$: Relation type 的**外部关系类型库**. 如下图:



- **知识图谱相关**

  Ψ : 知识图谱

  $\mathcal{E}_\Psi$: *Set of entities instances*, 知识图谱中所有 entity instances 的集合.

  $\mathcal{I}_\Psi = \{r(e_1, e_2\} \subset \mathcal{R}_\Psi \times \mathcal{E}_\Psi \times \mathcal{E}_\Psi$: Set of Relation instances, 知识图谱中所有 relation Instance的集合.

  $\mathcal{T}_\Psi = \{(e, y)\} \subset \mathcal{E}_\Psi \times \mathcal{Y}_\Psi$: *Set of entity-type facts*, 知识图谱中所有的实体和这个实体的类型组成的对的集合. 例如, (奥巴马,person), (奥巴马,president),(奥巴马,father) 都应该是其中的成员.

  $\mathcal{Y}_\Psi$: 知识图谱中的实体类型库, $\mathcal{Y} \in \mathcal{Y}_\Psi$

  $\mathcal{R}_\Psi$: 知识图谱中的关系类型库, $\mathcal{R} \in \mathcal{R}_\Psi$

- **语料库相关**

  $\mathcal{D}$: POS-tagged 语料库

  $\mathcal{M} = m_i{}_{i=1}^{N}$: *Set of entity mentions* , 代表从语料库 $\mathcal{D}$ 抽出的所有的 entity mentions 的集合.

  如图所示: 并且是有上下义的树状结构.如下图所示:

Candidate Generation & Distant Supervision

ID	Sentence
S1	***US*** president ***Barack Obama*** visit China today.
S2	A clip of ***Obama*** reading from his book "***Dreams of My Father***" has been shared out of context.
S3	***Barack Obama*** is the 44th and current President of the ***United States***
S4	President Clinton and ***Obama*** attended the funeral of former Israeli Prime Minister, and were scheduled to fly back to the ***US*** together.
...	...

2) 概念之间的对应关系

对于两个Entity mentions而言, 比如说 "奥巴马" 和 "美国".

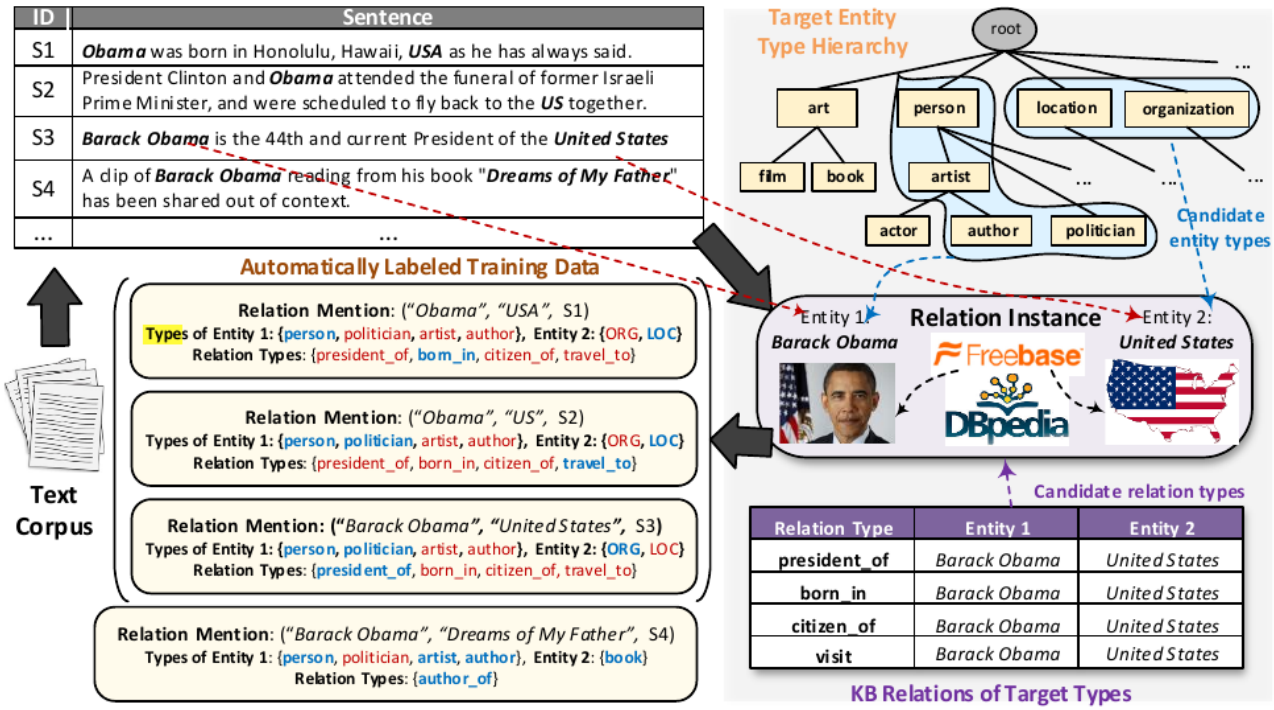
在不同的Sentence instance中, 其Entity type 可能是不同的, 在有的句子中, 奥巴马可能是作为总统出现的, 而在有的句子中, 奥巴马可能是作为父亲出现的.

而不同的Sentence instance, 这两个 Entity mentions之间的关系也可能是不同的, 比如下图的S1和S3就分别代表了 "Born_in" 和 "President_of".

因此在一个Sentence instance中, 一个Entity mention对应着一个可能的Entity type的集合. 而这个Sentence instance对应了一个可能的Relation type的集合.

3) 具体的例子

Type 在数据集中并没有其位置, 因为它是一种用来组织认知的概念; 而mention就是单纯的指概念(Type) 在实际数据中的instanse. 有点像类和实例的关系.具体见下图:



2.2 从语料库到知识图谱的映射

映射是指从语料库到知识图谱的映射.经过了两步.

2.2.1 从mentions到instance

由于语料库非常巨大, 知识图谱无法涵盖所, 因此并不是所有在语料库中出现的 entity mention 都可以 映射到 entity instance $\mathcal{E}_{\Psi}$. 即 $m \rightarrow e$

L : 可以映射过去的 entity mentions 的 index.

$\mathcal{M}_L$: *linkable entity mentions* , 可以映射过去的 entity mentions. 一般, $\mathcal{M}_L$ 只占不到全部的一半.

unlinkable entity mentions : 不可以映射过去的 entity mentions

而 relation instance的映射也是伴随着这个发生的 z:

z_i : z中的 m_1, m_2 都可以映射到 e_1, e_2 的情况下的 $z = (m_1, m_2, s)$.

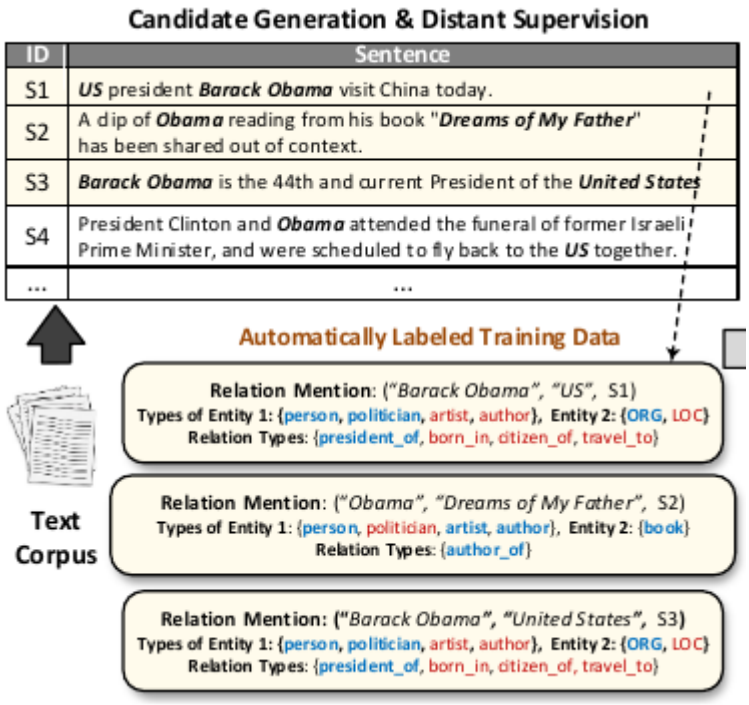
$\mathcal{R}_i$: 映射到的 e_1, e_2 之间可能有的关系. $\mathcal{R}_i = \{r | r(e_1, e_2) \in \mathcal{R}_\Psi\}$

2.2.2 从entity instance 到 Entity type

$\mathcal{Y}_{i,x}$: $\mathcal{Y}_{i,x} = \{y | (e_x, y) \in \mathcal{Y}_\Psi\}$, entity instance e_x 可能的 Entity type 的集合.

2.2.3 信息汇总

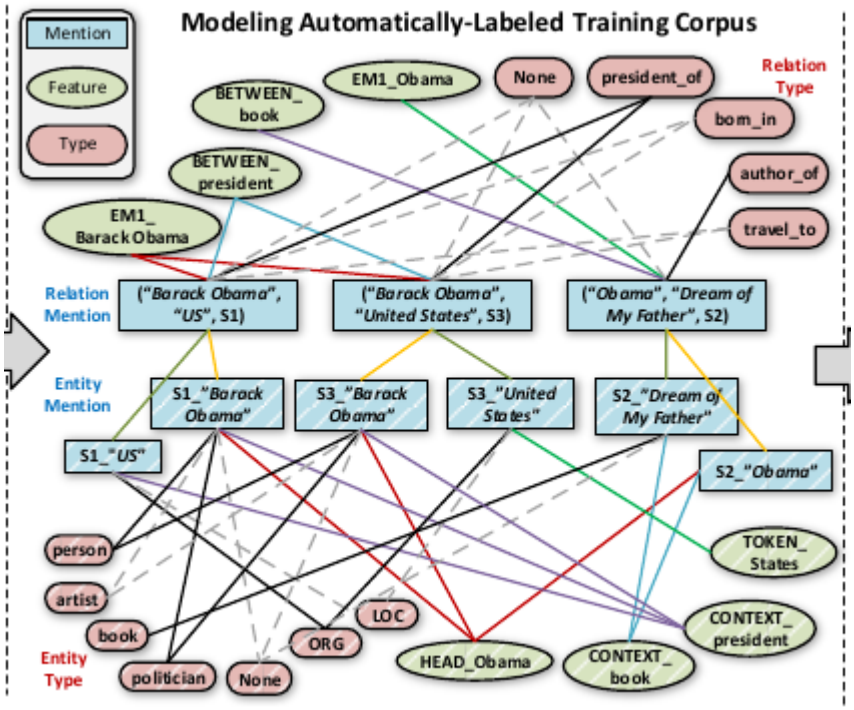
到这里,图示如下:



- 整体的一些符号

  $\mathcal{Z}_L = z_i{}_{i=1}^{N_L}$: 代表的是, 可以成功从entity mentions到entity instance 再到Entity type的 relation mentions 的集合.

$\mathcal{D}_L$: $\mathcal{D}_L = \{(z_i, \mathcal{R}_i, \mathcal{Y}_{i,1}, \mathcal{Y}_{i,2}\}$ 在整个映射中的所有信息的集合, 用于后续的使用方便.这里的这个信息量是非常大的, 第一个是可以成功映射的 relation mentions的集合, 第二个是成功映射后两个实体之间可能有的关系, 后两个是两个entity instance可能属于的Entity type.图示如下:



2.3 结合两者信息映射文本到特征空间

1) 名词解释

文中说道, jointly embeds **entity mentions, relation mentions, text features and type labels** into two low-dimensional spaces. 这里具体的意思如下:

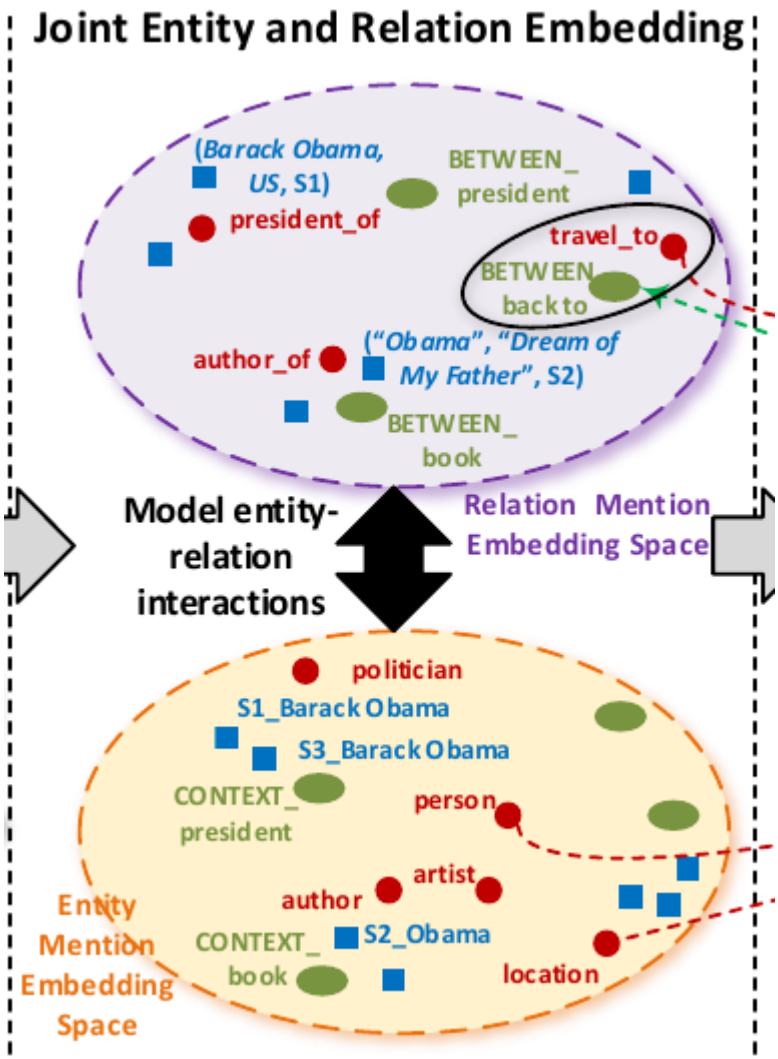
entity mentions, relation mentions : 这个上面已经解释过了就不重复了.

text feature : 是指在两个entity mentions附近的词汇.

type label : 是指Type of XX, 这里的XX是指在不同的空间内是不同的. 例如, 在Relation mention Embedding Space中, type label就是指的 Relation type; 反之亦此.

2) 对应关系

具体见下图:



从这个里面可以看出, 在每个空间内各有三种类型的数据.

Relation mention Embedding Space : 有

- Relation type
- Text feature
- Relation mention embedding

Entity mention Embedding Space : 有

- Entity type
- Text feature
- Entity mention embedding

3) 映射的规则

很简单的思想, 语义相近的向量离得近.

2.4 问题设定

上面我们介绍了 $\mathcal{Z}_L = z_i{}_{i=1}^{N_L}$, 是可以成功将自身的实体与知识图谱知识库结合的 relation mentions 的集合(其实就是句子了, 因为relation mentions 包含句子, $z = (m_1, m_2, s)$)

但是还有其他的部分, 我们把语料库 $\mathcal{D}$ 中所有的 $\mathcal{Z}$ 分为三部分:

- linkable relation mentions : $\mathcal{Z}_L$
- unlinkable relation mentions
- false relation mention (i.e., no target relation expressed between).

其中把后两者的集合称为 : $\mathcal{Z}_U$ (不清楚2和3的区别)

我们把 $\mathcal{Z}_U$ 当做unlabeled relation mentions, 这个就是我们的test集.

任务就是

- 预测 $\mathcal{Z}_U$ 中的 Relation type
- $\mathcal{Z}_U$ 中所有 z 中的 entity mentions 的 Entity type.

Non-goals

这里作者提出了两个本文不会涉及到内容:

- 外部entity linking system的错误, 错误的识别映射之后也是错误的.但是这里不着手去解决.
- 假设 外部实体知识库, 即有上下义的知识库是给出的, 不需要自己去设计.

3. CoType Framework

3.1 问题与模型

有两个基本问题:

- **Type prediction task** : 从entity mentions 和 relation mentions 到 Entity Type 和Relation Type 的映射是有噪音的, 我们需要从中选择正确的一组.

- **Relational learning** task: Entity Mentions 和 entity 相关句子描述上(即文本特征)有相关关系.

这两个问题的解决方案为:

- **Type prediction task** : weakly-supervised learning, 找到 mention 和 tyoe 之间的映射关系和映射条件.
- **Relational learning task** : 找到 Relation Mentions 和 **Entity Mentions Arguement** 之间的关系. 也就是根据句子的内容去预测这个句子的 Relation Type.

整个模型的框架是:

1. 语料库 $\mathcal{D}$ 中 $\rightarrow$ Entity Mentions $\mathcal{M}$.

   > 主要内容见3.2

2. Entity Mentions $\mathcal{M} \rightarrow$ **Relation Mentions** $\mathcal{Z} +$ 每个z中的**text feature** $+$ **Entity Mentions Argument**

   > 主要内容见3.2

   再结合知识图谱的信息利用远程监督去生成 $\mathcal{D}_L$, $\mathcal{D}_L = \{(z_i, \mathcal{R}_i, \mathcal{Y}_{i,1}, \mathcal{Y}_{i,2}\}$ 见2.2.3节

   > 1. 其中, $\mathcal{R}_i$ 是 $\mathcal{R}_i = \{r | r(e_1, e_2) \in \mathcal{R}_\Psi\}$, 讲的是将 m 映射到 e 之后的, 组成的对. 这里首相需要将**m 映射到 e**, 这里只需要进行字符匹配即可. 其中, e 是 Entity Instance.
   > 2. $\mathcal{Y}_{i,x} = \{y | (e_x, y) \in \mathcal{Y}_\Psi\}$ 是, 实体与实体类型的可能的组合. 这里就需要用到知识图谱中关于类型的信息, 即实体类型库 $\mathcal{Y}_\Psi$.

3. 结合上面的信息, 映射文本到特征空间, 这里做的就是2.3介绍的工作. 具体使用的输入是什么见3.3.

   . 2中得到的 $\mathcal{D}_L$ $+$ 3中得到的各种embedding $\rightarrow$ 对$\mathcal{Z}_U$所有Relation Mentions的类型预测 r^* (根据外部关系类型库) $+$ $\mathcal{Z}_U$所有Entity Mentions的类型预测(即找到每个Entity Mentions所属的外部类型知识库的路径 $\mathcal{Y}^*$)

3.2 Candidate Generation

重复一下问题:

1. 语料库 $\mathcal{D}$ 中 $\rightarrow$ Entity Mentions $\mathcal{M}$ 和
   . Entity Mentions $\mathcal{M}$ $\rightarrow$ **Relation Mentions** $\mathcal{Z}$ $+$ 每个z中的 **text feature** $+$ **Entity Mentions Argument**

将这两个问题分为三个:

1. Entity Mention Detection
2. Relation Mention Generation
3. Text Feature Extraction

3.2.1 Entity Mention Detection

这里可以看做是一个 NER.

本文拒绝使用单纯结合distant supervision的序列标注模型(因为distant supervision的数据很少) , 而设计了一个 Distantly-supervised text segmentation algorithm for domain-agnostic entity detection. 也就是

适用于任何领域文本的远距离监督文本分割算法去解决实体检测任务.(名字好长)

segment quality : 评价一个分割的词汇段是否像是一个实体的指标

主要思想是 : 评估$(segment\ quality)$ = $\mathcal{F}$(评估$(phrase\ quality)$, 评估$(POS\ pattern\ quality)$)

这个方法中使用了三种信息:

1. 语料库的信息
2. 句子层级的词汇信息
3. 语法的限制(比如说POS-tagging)

流程如下:

1. 从POS付き的语料库中挖掘出常出现的**词汇组合模式**和**POS组合模式**
2. 利用**从语料库中提取的特征**和**从句子中提取的特征**去训练**两个随机森林**, 分别计算出 评估$(phrase\ quality)$ 和 评估$(POS\ pattern\ quality)$

   举个例子, 下面就是 评估$(POS\ pattern\ quality)$ 大概感觉:

	POS Tag Pattern	Example
Good (high score)	*NNP NNP*	San Francisco/Barack Obama/United States
	NN NN	comedy drama/car accident/club captain
	CD NN	seven network/seven dwarfs/2001 census
	JJ NN	crude oil/nucletic acid/baptist church
Bad (low score)	*DT JJ NND*	a few miles/the early stages/the late 1980s
	CD CD NN IN	2 : 0 victory over/1 : 0 win over
	NN IN NNP NNP	rating on rotten tomatoes
	VVD RB IN	worked together on/spent much of

3. 计算并最优化 评估$(segment\ quality)$

   计算:

   首先这个评估是基于整个语料库 $\mathcal{D}$ 的, 设其中的句子(或者文档)为 d, 设其分割形式为 S_d, 那么我们要最大化两种的联合概率, 即 $\sum_d^D log\, p(S_d, d)$.其公式为:

$$\sum_d^{\mathcal{D}} \log p(S_d, d) = \sum_d^{\mathcal{D}} \sum_{t=1}^{|d|} \log p\left(b_{t+1}^{(d)}, c^{(d)} \mid b_t^{(d)}\right)$$

   为了方便说明, 假设一个句子为 **ZHAO is 21 years old**

   上面的式子右边出现的概率函数中, $c^{(d)}$ 是指一个分段片段, 那么 **ZHAO** 就是一个片段. b_t 为文本中的词汇的index. 那么, $c^{(d)}$ 就是在句子中, 起始索引为 $b_t^{(d)}$, 结束于 $b_{t+1}^{(d)}$ 的一个片段.

   右式的具体公式为:

$$p\left(b_{t+1}, c \mid b_t\right) = p\left(b_{t+1} - b_t\right) \cdot p\left(c \mid b_{t+1} - b_t\right) \cdot Q(c)$$

   这个式子中, $Q(c)$ 利用到了 2. 中计算的
   评估$(phrase\ quality)$, 评估$(POS\ pattern\ quality)$.

   优化:

   使用Viterbi算法去最大化上面的概率.

4. 一开始使用的是从KB中distant supervision的可靠数据去进行上面的的计算, 通过2和3后, 对新的数据进行预测, 把高概率的数据加入训练数据集后, 重复上面的运算.

下面是结果:

Dataset	NYT	Wiki-KBP	BioInfer
FIGER segmenter [26]	0.751	0.814	0.652
Our Approach	**0.837**	**0.833**	**0.785**

Table 2: Comparison of F1 scores on entity mention detection.

3.2.2 Relation Mention Generation

这个部分对应的是2.2节.

1) 生成 z

这里的思想比较简单, 从3.2.1中生成的 Entity Mentions 中, 对照KB中的词汇. 生成 Relation Mentions : $z = (m_1, m_2, s)$

但是, 这里由于关系是有方向的, 即 KB 是有向图, 所以一个句子可以构建两个 Relation Mentions. $z = (m_a, m_b, s)$ 和 $z = (m_b, m_a, s)$. 对语料库中的每一个句子进行这里的处理, 得到了所有的 z 的集合 $\mathcal{Z}$

2) 按照是否可映射分割 $\mathcal{Z}$

这里就可以将 $\mathcal{Z}$ 分为三部分:

1. $\mathcal{Z}_L$ *: linkable Relation Mentions* , 这种Relation Mentions满足两个条件 :
   - 这个里面的 m_a, m_b 是可以映射到KB中的node(e_a, e_b)的, 即 其中的Entity Mentions都属于 $\mathcal{M}_L$ (定义看上面). 即, $m_a, m_b \in \mathcal{M}_L$
   - 并且他们之间有连接. 也就是映射到的两个node之间是有edge的.

   这部分数据是 **positive samples**, 可以用来建模 **Type label** 的关系.
2. *unlinkable entity mentions* , 这种Relation Mentions满足两个条件 :
   - 这个里面的 m_a, m_b 是可以映射到KB中的node(e_a, e_b)的. 即, $m_a, m_b \in \mathcal{M}_L$
   - 但是在KB中这两个node (e_a, e_b) 之间没有edge.

   这部分数据可以用来建模 **None relation label** 的关系.
3. *false relation mention* : 即, $m_a \notin \mathcal{M}_L \; || \; m_b \notin \mathcal{M}_L$

   这部分数据可以用来建模 **None entity label** 的关系.

后两部分作为 **negative examples** 存在.

3) 构建 $\mathcal{D}_L$

先回顾 $\mathcal{D}_L$ 的内容 : $\mathcal{D}_L : \mathcal{D}_L = \{(z_i, \mathcal{R}_i, \mathcal{Y}_{i,1}, \mathcal{Y}_{i,2}\}$. 这个四元组中, 后面的三个都可以结合KB(只是图谱)的信息和 z_i 的信息来实现生成. 具体的看上面的定义.

关于 $\mathcal{D}_L$ 使用的 z_i 的范围的定义, 这个地方和上面的地方有一些出入 :

- 这里说的是不论 z_i 中的 m_a, m_b 是否可以被映射到KB中, 这里的都可以将其作为 $\mathcal{D}_L$ 的一部分.
- 而上面说的是 只有可以被映射过去的部分才可以, 也就是说这里的 negative example 不可以.

3.2.2.3 Text Feature Extraction

这里要做的事情很容易明白, 如下图所示:

需要注意这个里面用了词性信息

Feature	Description	Example
Entity mention (EM) head	Syntactic head token of each entity mention	"*HEAD_EM1_Obama*"
Entity Mention Token	Tokens in each entity mention	"*TKN_EM1_Barack*"
Tokens between two EMs	Each token between two EMs	"*was*", "*elected*", "*President*", "*of*", "*the*"
Part-of-speech (POS) tag	POS tags of tokens between two EMs	"*VBD*", "*VBN*", "*NNP*", "*IN*", "*DT*"
Collocations	Bigrams in left/right 3-word window of each EM	"*Honolulu native*", "*native Barack*", ...
Entity mention order	Whether EM 1 is before EM 2	"*EM1_BEFORE_EM2*"
Entity mention distance	Number of tokens between the two EMs	"*EM_DISTANCE_5*"
Entity mention context	Unigrams before and after each EM	"*native*", "*was*", "*the*", "*in*"
Special pattern	Occurrence of pattern "em1_in_em2"	"*PATTERN_NULL*"
Brown cluster (learned on $\mathcal{D}$)	Brown cluster ID for each token	"*8_1101111*", "*12_111011111111*"

Table 3: Text features for relation mentions used in this work [17, 43] (excluding dependency parse-based features and entity type features). ("*Barack Obama*", "*United States*") is used as an example relation mention from the sentence "*Honolulu native* ***Barack Obama*** *was elected President of the* ***United States*** *on March 20 in 2008.*".

然而, 符号难以理解.

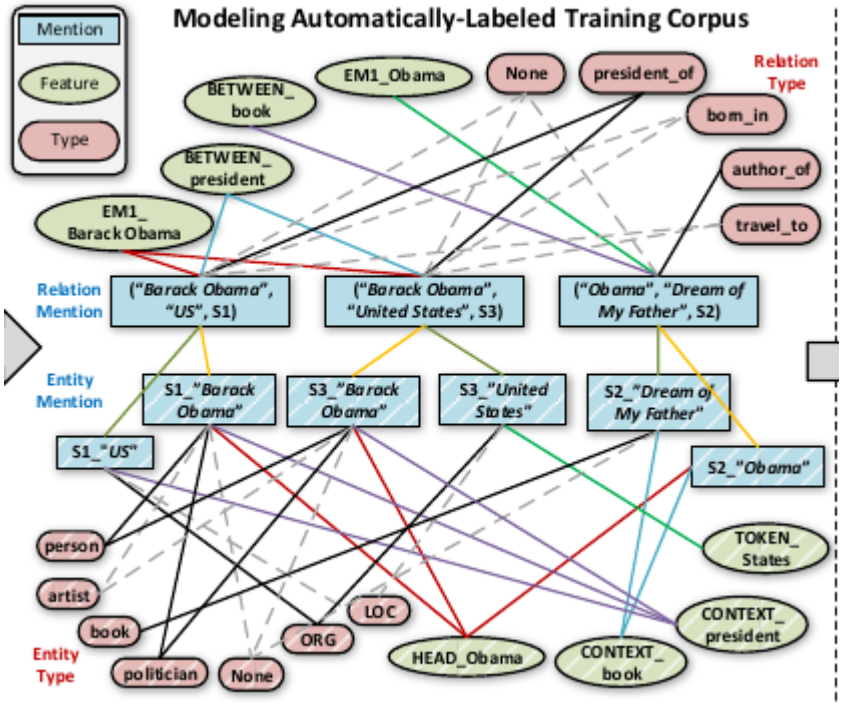
这里对于一个句子有两种特征,

- $\mathcal{F}_z$ 这个句子的专有特征
- $\mathcal{F}_m$ 这个句子中的Entity Mentions 的全局特征. 也就是说对于拥有相同的Entity Mentions的不同句子, 这个部分是一样的.

那么一个句子的特征就是 $(\mathcal{F}_z, \mathcal{F}_m)$.

3.2.2.4 图示

一个 $\mathcal{D}_L$ 的图示:



3.3 Joint Entity and Relation Embedding

这个部分对应的是第2.3节.

首先这个投射的方法很简单, 就是让同一个句子中的 Entity Mentions, Relation Mentions, Relation Type, Entity Type, Text features 的向量在空间上也距离的近.

但是面临着两个问题:

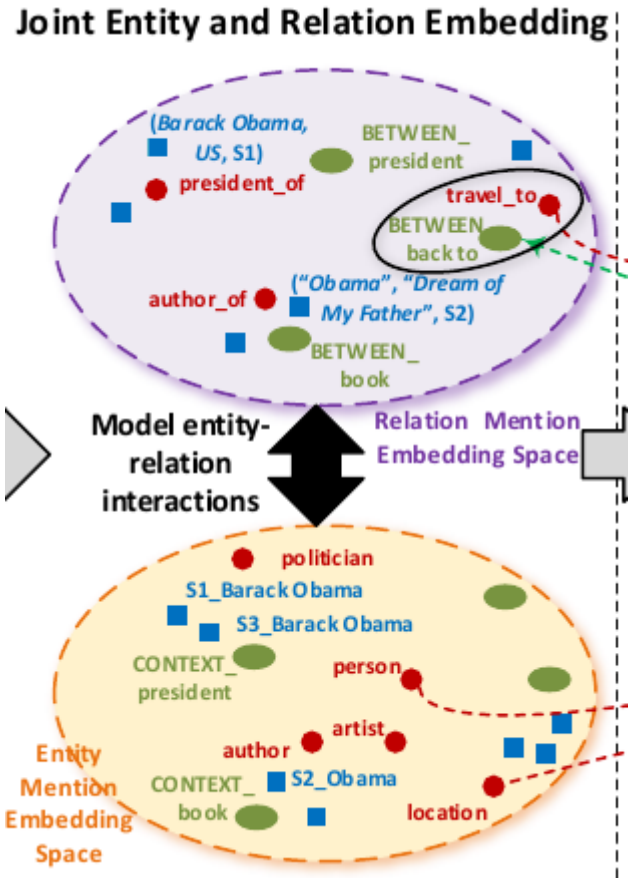
1. 如上面的图所示, 对于一个句子而言, 其对应的 $\mathcal{R}_i$ 有可能含有多种 r, 这样就会出现很多错误的 Relation Type, 会影响整体的向量分配.
2. 一个特征向量空间无法捕捉到 Entity Type 和 Relation Type 的不同. 原因如下:

> 因为我们进行编码的依据是co-occurrence次数多的词汇可以拥有相近的向量, 但是这对于 Relation Mentions 的向量和 Entity Mentions的向量来说并不是好事, 因为, 即使他们两个拥有很高的 co-occurrence次数, 他们之间也没有相关性. 只是单纯的 Relation Mentions 包括 Entity Mentions而已. 使用这样的方法会导致两者向量相近, 这样的后果是:
>
> Entity Mentions 的向量和 Entity Type 的向量相近
>
> Relation Mentions 的向量和 Relation Type 的向量相近

> 这样的话, Entity Type 的向量就会和 Relation Type 的向量相近, 而两者是没有必然联系的.

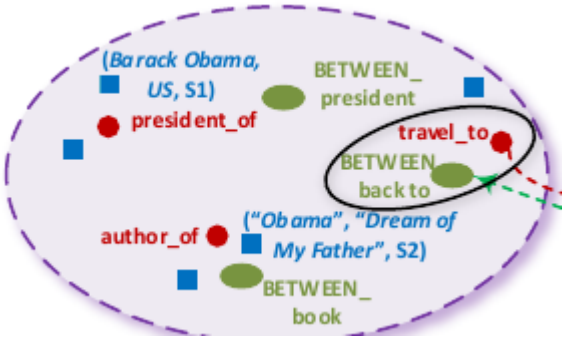
因此这里使用了两个空间, 而不是使用一个空间.

我们的理想向量空间的状态如下, 注意两个空间使用的text feature也是不同的:



下面分别针对两个空间的embedding来进行说明.

3.3.1 Modeling Types of Relation Mentions



这个里面我们要编码的部分有三个, Relation Mentions, Relation Type 和 体现Relation的text features. 这里对三者的向量进行约束的方法是, 两两分开进行约束.

3.3.1.1 Relation Mentions 和 Text features 的约束

$$\mathcal{L}_{ZF} = -\sum_{z_i \in \mathcal{Z}_L} \sum_{f_j \in \mathcal{F}_z} w_{ij} \cdot \log p(f_j|z_i),$$

z_i 是 Relation Mentions, f_i 是 Text features.

w_{ij} 是 Relation Mentions 和 Text features co-occurrence 的频度.

这个式子其实比较好理解的, 最小化负的共现概率就是最大化共现概率.

其中重要的是 $p(f_i|z_i)$ 的计算方法, 这里就用到了两者(Entity Mentions和Text features)的向量.如下:

$$p(f_j|z_i) = \exp(\mathbf{z}_i^T \mathbf{c}_j) / \sum_{f' \in \mathcal{F}_z} \exp(\mathbf{z}_i^T \mathbf{c}_{j'})$$

注意这里的分母是对所有的特征进行了计算, 然后为了简便计算用了负采样:

$$\log \sigma(\mathbf{z}_i^T \mathbf{c}_j) + \sum_{v=1}^{V} \mathbb{E}_{f_{j'} \sim P_n(f)} \left[\log \sigma(-\mathbf{z}_i^T \mathbf{c}_{j'})\right]$$

最大化这个函数之后, 就可以使得拥有相同特征的Entity Mentions具有相近的向量, 也可以使得语义相近的Entity Mentions和Text features拥有相近的向量.

3.3.1.2 Relation Mentions 和 Relation Type 的约束

我们知道, 在 $\mathcal{D}_L$ 中的 $\mathcal{R}_i$ 其实是包含了Entity Mentions对应的Entity Instance之间的所有可能的Relation Type. 即: $\mathcal{R}_i = \{r|r(e_1, e_2) \in \mathcal{R}_\Psi\}$

但是我们可以用其他完全无关的 r 来和这里的r进行对比, 再利用大量的数据, 就可以获得好的效果.

公式为:

$$\ell_i = \max\left\{0, 1 - \left[\max_{r \in \mathcal{R}_i} \phi(z_i, r) - \max_{r' \in \overline{\mathcal{R}}_i} \phi(z_i, r')\right]\right\}.$$

其中, 第二个max就是从 非$\mathcal{R}_i$ 中进行的采样, 就是利用了负采样中的绝对不可能性, 再加上大数据成功学得向量. 其中, $\phi(z_i, r_k) = \mathrm{z}_i^T \mathrm{r}_k$.

并且要注意, 这里是对每个句子进行的计算, 因此我们还需要累加所有 $\mathcal{M}_L$ 中的句子.

3.3.1.3 两者的结合

$$O_Z = \mathcal{L}_{ZF} + \sum_{i=1}^{N_L} \ell_i + \frac{\lambda}{2} \sum_{i=1}^{N_L} \|\mathbf{z}_i\|_2^2 + \frac{\lambda}{2} \sum_{k=1}^{K_r} \|\mathbf{r}_k\|_2^2,$$

后面的是正则项, 但是奇怪为什么没有Text features的正则约束.

3.3.2 Modeling Types of Entity Mentions

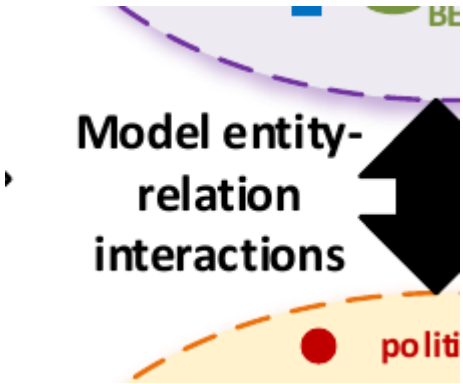
这里不赘述, 用相同的方法去计算:

$$\mathcal{L}_{MF} = - \sum_{m_i \in \mathcal{M}_L} \sum_{f_j \in \mathcal{F}_m} w_{ij} \cdot \log p(f_j | m_i),$$

$$O_M = \mathcal{L}_{MF} + \sum_{i=1}^{N'_L} \ell'_i + \frac{\lambda}{2} \sum_{i=1}^{N'_L} \|\mathbf{m}_i\|_2^2 + \frac{\lambda}{2} \sum_{k=1}^{K_y} \|\mathbf{y}_k\|_2^2.$$

3.3.3 Modeling Entity-Relation Interactions

这一个建模的是 :



上面虽然表达了各种向量的相近关系, 但是忽略了一个很重要的关系, 那就是 **距离函数**, 即:

$$Vector(m1) + Vector(z) = Vector(m2)$$

> HYPOTHESIS 3 (ENTITY-RELATION INTERACTION).
> *For a relation mention $z = (m_1, m_2, s)$, embedding vector of m_1 should be a nearest neighbor of the embedding vector of m_2 plus the embedding vector of relation mention z.*

由此得来损失函数:

$$O_{ZM} = \sum_{z_i \in Z_L} \sum_{v=1}^{V} \max\{0, 1 + \tau(z_i) - \tau(z_v)\},$$

其中,

$$\tau(z) = \|\mathbf{m}_1 + \mathbf{z} - \mathbf{m}_2\|_2^2$$

其中第二个 z_v 是负采样.

3.3.4 联合优化

优化目标是 :

$$\min_{\{\mathbf{z}_i\},\{\mathbf{c}_j\},\{\mathbf{r}_k\},\{\mathbf{m}_i\},\{\mathbf{c}'_j\},\{\mathbf{y}_k\}} \mathcal{O} = \mathcal{O}_M + \mathcal{O}_Z + \mathcal{O}_{ZM}$$

最重要的优化方法在下一节说

3.4 Model Learning and Type Inference

3.4.1 优化方法的简介

这里使用的是, 随机子梯度下降法, 之所以称为子梯度是因为, 这里的参数是所有的向量, 而要采样随机法去进行学习的话, 就每次只能对一部分参数(即向量)进行更新, 因此这里叫做 子梯度下降法.

然后, 文献46保证了这个学习过程会收敛.

3.4.2 Type Inference

对于Relation Type和Entity Type, 采用了相似的策略:

- 先根据 z 的Text features, 确定索引向量, 即:

  对于Entity Mentions :

  $$\mathbf{m} = \sum_{f_j \in \mathcal{F}_m(m)} \mathbf{c}'_j$$

  对于Relation Mentions :

  $$\mathbf{z} = \sum_{f_j \in \mathcal{F}_z(z)} \mathbf{c}_j$$

- 其次通过在外部类型库中利用上面的索引向量作为输入, 利用cosine函数查找最近的值.

  对于Entity Mentions : 使用的外部实体类型知识库, 由于是树状结构所以需要使用递归查找.

  对于Relation Mentions : 遍历查找即可.

4. EXPERIMENTS

文章在三个实验上进行了测试:

- **Entity Recognition and Typing**

  效果如下:

Method	NYT	Wiki-KBP	BioInfer
DS+Perceptron [26]	0.641	0.543	0.470
DS+Kernel [33]	0.632	0.535	0.419
DeepWalk [38]	0.580	0.613	0.408
LINE [50]	0.765	0.617	0.557
DS+Logistic [31]	0.771	0.646	0.543
MultiR [21]	0.693	0.633	0.501
FCM [16]	0.688	0.617	0.467
CoType-RM	0.812	0.634	0.587
CoType-TwoStep	0.829	0.645	0.591
CoType	**0.851**	**0.669**	**0.617**

Table 6: Performance comparison on relation classification accuracy over ground-truth relation mentions on the three datasets.

- **Relation Classification**

  这个任务利用向量空间的向量对一个具体的 Relation Mention 预测其真实关系.

  效果如下:

Method	NYT	Wiki-KBP	BioInfer
DS+Perceptron [26]	0.641	0.543	0.470
DS+Kernel [33]	0.632	0.535	0.419
DeepWalk [38]	0.580	0.613	0.408
LINE [50]	0.765	0.617	0.557
DS+Logistic [31]	0.771	0.646	0.543
MultiR [21]	0.693	0.633	0.501
FCM [16]	0.688	0.617	0.467
CoType-RM	0.812	0.634	0.587
CoType-TwoStep	0.829	0.645	0.591
CoType	**0.851**	**0.669**	**0.617**

Table 6: Performance comparison on relation classification accuracy over ground-truth relation mentions on the three datasets.

- **Relation Extraction**

  其实我有点不明白这里和上面的分类有什么区别, 知道的还望告知一下:

  效果如下:

	NYT [43, 21]				Wiki-KBP [12, 26]				BioInfer [39]			
Method	**Prec**	**Rec**	**F1**	**Time**	**Prec**	**Rec**	**F1**	**Time**	**Prec**	**Rec**	**F1**	**Time**
DS+Perceptron [26]	0.068	**0.641**	0.123	15min	0.233	0.457	0.308	7.7min	0.357	0.279	0.313	3.3min
DS+Kernel [33]	0.095	0.490	0.158	56hr	0.108	0.239	0.149	9.8hr	0.333	0.011	0.021	4.2hr
DS+Logistic [31]	0.258	0.393	0.311	25min	0.296	0.387	0.335	14min	**0.572**	0.255	0.353	7.4min
DeepWalk [38]	0.176	0.224	0.197	1.1hr	0.101	0.296	0.150	27min	0.370	0.058	0.101	8.4min
LINE [50]	0.335	0.329	0.332	2.3min	0.360	0.257	0.299	1.5min	0.360	0.275	0.312	35sec
MultiR [21]	0.338	0.327	0.333	5.8min	0.325	0.278	0.301	4.1min	0.459	0.221	0.298	2.4min
FCM [16]	0.553	0.154	0.240	1.3hr	0.151	**0.500**	0.301	25min	0.535	0.168	0.255	9.7min
DS-Joint [24]	**0.574**	0.256	0.354	22hr	**0.444**	0.043	0.078	54hr	0.102	0.001	0.002	3.4hr
CoType-RM	0.467	0.380	0.419	2.6min	0.342	0.339	0.340	1.5min	0.482	0.406	0.440	42sec
CoType-TwoStep	0.368	0.446	0.404	9.6min	0.347	0.351	0.349	6.1min	0.502	0.405	0.448	3.1min
CoType	0.423	0.511	**0.463**	4.1min	0.348	0.406	**0.369**	2.5min	0.536	**0.424**	**0.474**	78sec

Table 7: Performance comparison on end-to-end relation extraction (at the highest F1 point) on the three datasets.