# CS290D – Advanced Data Mining

Instructor: Xifeng Yan
Computer Science
University of California at Santa Barbara

# Knowledge Bases

Instructor: Yang Li
Computer Science
University of California at Santa Barbara

Slides adjusted from: http://resources.mpi-inf.mpg.de/yago-naga/vldb2014-tutorial/

# Probase: builds a taxonomy from the Web [Wu et al.2012]

- Extract with Hearst Patterns and isA Patterns

NP such as NP, NP, ..., and|or NP

such NP as NP,* or|and NP

NP, NP*, or other NP

NP, NP*, and other NP

NP, including NP,* or | and NP

NP, especially NP,* or|and NP

NP is a/an/the NP

# Probase: builds a taxonomy from the Web [Wu et al.2012]

- Extraction is not trivial

… **animals** other than **cats** such as **dogs** …

# Probase: builds a taxonomy from the Web [Wu et al.2012]

- Extraction is not trivial

… **household pets** other than **animals** such as **reptiles**, aquarium fish …

# Probase: builds a taxonomy from the Web [Wu et al.2012]

- Iterative extraction & leveraging existing knowledge

Syntactic patterns

Knowledge

Let $X_s = \{x_1, \cdots, x_m\}$. We compute likelihood $p(x_k|Y_s)$ for $x_k \in X_s$. Without loss of generality, we assume $x_1$ and $x_2$ have the *largest* likelihoods, and $p(x_1|Y_s) \geq p(x_2|Y_s)$. We compute the ratio of likelihood $r(x_1, x_2)$ as follows and then we pick $x_1$ if the ratio is above a threshold:

$$r(x_1, x_2) = \frac{p(x_1|Y_s)}{p(x_2|Y_s)} = \frac{p(Y_s|x_1)p(x_1)}{p(Y_s|x_2)p(x_2)} = \frac{p(x_1)\prod_{i=1}^{n} p(y_i|x_1)}{p(x_2)\prod_{i=1}^{n} p(y_i|x_2)}$$

# Probase: builds a taxonomy from the Web [Wu et al.2012]

- Handling different senses of the same label (e.g. plants)

**Figure 1: From a single sentence to a local taxonomy**

**Figure 2: Horizontal merge**

# Probase: builds a taxonomy from the Web [Wu et al.2012]

- Handling different senses of the same label (e.g. plants)

(a) Single sense alignment

(b) Multiple sense alignment

**Figure 3: Vertical merge**

# Probase: builds a taxonomy from the Web [Wu et al.2012]

- Probabilistic taxonomy
  - Plausibility

$$\mathcal{P}(x,y) = 1 - p(\bar{E}) = 1 - p(\bigwedge_{i=1}^{n} \bar{s_i}) = 1 - \prod_{i=1}^{n}(1-p_i).$$

  - Typicality

$$\mathcal{T}(i|x) = \frac{n(x,i)\cdot\mathcal{P}(x,i)}{\sum_{i'\in I_x} n(x,i')\cdot\mathcal{P}(x,i')}$$

**bird**

"robin" is a more *typical* bird than a "penguin"

$$p(robin|bird) > p(penguin|bird)$$

# Probase covers many long tail concepts

## Frequency distribution of the 2.7 million concepts

# Probase is a high-quality Web-scale taxonomy

**Scale comparison of several open domain taxonomies**

| ***name*** | ***# of concepts*** | ***# of isA pairs*** |
|---|---|---|
| Freebase | 1,450 | 24,483,434 |
| WordNet | 25,229 | 283,070 |
| WikiTaxonomy | 111,654 | 105,418 |
| YAGO | 352,297 | 8,277,227 |
| DBPedia | 259 | 1,900,000 |
| ResearchCyc | ≈ 120,000 | < 5,000,000 |
| KnowItAll | N/A | < 54,753 |
| TextRunner | N/A | < 11,000,000 |
| OMCS | 173,398 | 1,030,619 |
| NELL | 123 | < 242,453 |
| **Probase** | **2,653,872** | **20,757,545** |

**Overall precision: 92.8%**

# Impact of iterative extraction

# Automatic Knowledge Harvesting

- Taxonomic Knowledge:
  - Entities and Classes
- Factual Knowledge:
  - Relations between Entities
- Emerging Knowledge:
  - New Entities & Relations
- Common-sense Knowledge

# Supervised Relation Extraction

- Pros
  - High quality supervision
- Cons
  - Very expensive to generate supervision
  - Not easy to add more relations
  - Cannot generalize to text from different domains

# Semi-Supervised Relation Extraction

- Pros
  - Do not require manually labeling
- Cons
  - Sensitive to original set of seeds
  - Semantic drift at each iteration
  - Hard to measure confidences of patterns/extractions

# Distant-Supervised Relation Extraction

- Pros
  - Can scale to the web, as no supervision required
  - Generalizes to text from different domains
  - Generates a lot more supervision than patterns
- Cons
  - False positive/negative training data (open research problem!)
  - Poor coverage for tail relations

# Open Relation Extraction

- Pros
  - Require no assumptions about domain knowledge
  - Require no prior information of relation types
  - Extract a large number of relations with high coverage
- Cons
  - Result is noisy
  - Extracted relations are not canonicalized
  - Far from high quality

# More Relation Extraction

- Joint relation extraction + other NLP tasks
  - Named Entity Tagging [Yao et al., 10]
  - Entity Linking [Chen et al., 14]
  - Coreference Resolution [Singh et al., 13]

# Joint Inference of Entities, Relations & Coreference

[Singh et al., 2013]

- Entities tagging, relation extraction, coreference resolution can mutually facilitate each other.
- Joint inference is effective to avoid cascading of errors.

Increase accuracy by 1%~2% for all tasks

# Relation Taxonomy [N. Nakashole et al.: EMNLP2012, VLDB2012]

- A large resource of relational patterns that are arranged in a semantically meaningful taxonomy.

# Relation Taxonomy: Motivation

- Understanding semantic equivalence of phrases
  - “<person> is born in <place>” = “<person>’s birthplace is <place>”
- Mapping phrases into semantic relations
  - "A maintains good friendship with B” => “A isFriendOf B”
- Supporting inferences
  - "A isCapticalOf B” implies “A isCityOf B”,
  - which further implies “A isLocatedIn B”

# Relation Taxonomy: Goal

- Systematically harvest textual patterns from text corpora
- Group synonymous patterns into pattern synsets
- Order the synsets into a subsumption hierarchy
- Fully automatic, without manual efforts

# Relation Taxonomy: Pattern Extraction

**Step1**

- Prepare entity name and semantic type dict
- Match noun phrases against the dict

**Step2**

- If phrases matches more than one entities, disambiguate using context-similarity

**Step3**

- Obtain textual relation patterns by traversing dependency tree to get the shortest path connecting two entities

- "Winehouse effortlessly performed her song Rehab"
- →"[Amy Winehouse] effortlessly performed [Rehab(song)]"

# Relation Taxonomy: SOL Pattern

- SOL: Syntactic-Ontologic-Lexical Patterns
  - A sequence of words, POS-tags, * and ontological types
- Example: <person>'s [adj] voice * <song>
  - Matches “Amy’s soft voice in ‘Rehab’”
  - Matches “Presley’s solid voice in his song ‘All shook up’”
  - Type signature: <person> X <song>
  - Support set: {(Amy, Rehab), (Presley, All Shook Up)}
- To generate SOL pattern from textual pattern:
  - Replace entities with their types
  - Decompose pattern into n-grams
  - Infrequent n-grams are replaced by wildcards

# Relation Taxonomy: SOL Pattern

- Pattern B is “syntactically more general” than Pattern A
  - Every string matches A also matches B
- Pattern B is “semantically more general” than Pattern A
  - A’s support set $\subseteq$ B’s support set
- Synonymous: Pattern A $\subseteq_{sem}$ Pattern B $\wedge$ B $\subseteq_{sem}$ A
  - A set of synonymous patterns form a pattern synset

# Relation Taxonomy: Syntactic Pattern Generalization

- How to generalize?
  - Replace words by POS-tags
  - Replace word sequence with “*”
  - Generalize the types in the pattern
- Whether to generalize?
  - First, generate all possible generalizations
  - Second, test if a generalization subsumes multiple patterns with disjoint support sets: if so, abandon it.

“<person> is father of <person>”
“<person> is mother of <person>”

“<person> is * of <person>”

# Relation Taxonomy: Semantic Pattern Generalization

- Soft Set Inclusion
  - Set A can be a subset of set B to some degree
  - Can be measured by confidence:
  - Degree($A \subseteq B$) = $|A \cap B|/|A|$

- Used to arrange pattern synsets into a hierarchy

# Relation Taxonomy: Subsumption mining

- Naïve solution is to compare $n^2$ support sets : too slow!
- Instead, store support sets in an extended prefix tree
  - Compact representation of support sets
  - Patterns with entity-pairs in common share are a prefix

A, B, C, … entity pairs

p1, p2, … patterns

**Support sets:**

P1: {A, B, C}

P2: {A, B, C, D, E}

P3: {F, G, H }

P4: {I, J, F, G, K}

# Relation Taxonomy: Subsumption mining

- Leverage tree structure as follows:
  - Traverse tree bottom up
  - Start at last node of a pattern P, find all patterns with a non-zero support set intersection with pattern P
  - Complexity: O(# patterns)

$p1 \Rightarrow p2$

A, B, C, … entity pairs

p1, p2, … patterns

**Support sets:**

P1: {A, B, C}

P2: {A, B, C, D, E}

P3: {F, G, H }

P4: {I, J, F, G, K}

# Relation Taxonomy: DAG Construction

- The subsumption graph may contain cycles:
  - To get a taxonomy, need to form a DAG
- Goal: to discard min number of edges whose removal results in DAG
  - NP-hard
- A greedy algorithm
  - Construct DAG bottom up
  - Add highest weight edge first
  - Repeatedly add edges if they don't intro cycles
  - End up with the PATTY Taxonomy

# Relation Taxonomy: Subsumption Precision

| Corpus/Types | Number of Edges | Random |
|---|---|---|
| NYT/YAGO2 | 12,000 | 68% |
| NYT/Freebase | 80000 | 41% |
| WKP/YAGO2 | 8,000 | 75% |
| WKP/Freebase | 20000 | 62% |

Example subsumptions from Wikipedia

<person> winner of <award> ⇒ <person> nominated for <award>

<person> 's widow <person> ⇒ <person> 's wife <person>

# Automatic Knowledge Harvesting

- Taxonomic Knowledge:
  - Entities and Classes
- Factual Knowledge:
  - Relations between Entities
- Emerging Knowledge:
  - New Entities & Relations
- Common-sense Knowledge

# Emerging Entities [Hoffart et. al, WWW2014]

- Emerging entities are usually not covered by KB
  - E.g. "**Prism** program was revealed by the whistleblower **Snowden**."
- Identifying emerging entities using existing KB and Web text
  - KBs have lexicon of (name, entity) pairs
  - Key idea: profile emerging entities from the Web
  - Assumption: for one name, one emerging entity

| | New Entity | Existing Entity |
|---|---|---|
| New Name | assumption | ✖ |
| Existing Name | ? | disambiguation |

# Emerging Entities [Hoffart et. al, WWW2014]

# Emerging Entities [Hoffart et. al, WWW2014]

**Name Keyphrases**
*extracted from any document mentioning "PRISM"*

**Entity Keyphrases**
*extracted from annotations of entities referred to by "PRISM"*

keyphrases defined by POS pattern filters

# Emerging Entities [Hoffart et. al, WWW2014]

# Emerging Relations: OpenIE [Fader et al. 2011, Lin et al. 2012, Mausam et al. 2012]

- Extracting phrases as relations
  - *Madame Bruni in her happy marriage with the French president …*
  - *The first lady had a passionate affair with Stones singer Mick …*
  - *Natalie was honored by the Oscar …*
- Grouping phrases with relation taxonomy (e.g. Patty)
  - *{cover songs, interpretation of, singing of, voice in, …}* ⇔ *SingerCoversSong*
  - *{classic piece of, 's old song, written by, composition of, …}* ⇔ *MusicianCreatesSong*
- Grouping phrases with paraphrase DBs

# Automatic Knowledge Harvesting

- Taxonomic Knowledge:
  - Entities and Classes
- Factual Knowledge:
  - Relations between Entities
- Emerging Knowledge:
  - New Entities & Relations
- Common-sense Knowledge

# Common-sense Knowledge

- About general concepts instead of specific entities

Apples are green, red, round, juicy, …
but not fast, funny, verbose, …

Snakes can crawl, doze, bite, hiss, …
but not run, fly, laugh, write, …

Pots and pans are in the kitchen, cupboard, …
but not in in the bedroom, in your pocket, in the sky, …

- (Relatively) rarely mentioned in text (since they are obvious!)
- Critical for reasonings/inferences

# Crowdsourcing for Common-sense Knowledge

ConceptNet 5: 3.9M concepts & 12.5M edges

http://conceptnet5.media.mit.edu/

# Harvesting Common-sense Knowledge from Web

[Tandon et al. AAAI 2011]

- Use ConceptNet data as seeds to harvest commonsense facts from Google N-Grams corpus
- Iterative pattern expansion

- Avoids semantic drift by using large amount of seeds
- Constrain patterns by Part Of Speech tags
  - X<noun> is very Y<adjective>

# Harvesting Common-sense Knowledge from Web

[Tandon et al. AAAI 2011]

| Relation | Precision (%) | #Facts Extracted |
|---|---|---|
| CapableOf | 77 | **907,173** |
| Causes | 88 | **3,218,388** |
| Desires | 58 | **4,386,685** |
| HasPrerequisite | 82 | **5,336,630** |
| HasProperty | 62 | **2,976,028** |
| IsA | 62 | **11,694,235** |
| LocatedNear | 71 | **13,930,656** |
| PartOf | 71 | **11,175,349** |
| SimilarSize | 74 | **8,640,737** |
| … | … | **…** |

More than 200 million facts extracted
Extension of ConceptNet by orders of magnitude

# Common-sense Rules Mining on KBs [Galarraga et al. WWW'13]

AMIE inferred commonsense rules from YAGO, such as

$$\boldsymbol{marriedTo(x,y) \wedge livesIn(x,z) \Rightarrow livesIn(y,z)}$$

$$\boldsymbol{bornIn(x,y) \wedge locatedIn(y,z) \Rightarrow citizenOf(x,z)}$$

$$\boldsymbol{hasWonPrize(x,LeibnizPreis) \Rightarrow livesIn(x,Germany)}$$

http://www.mpi-inf.mpg.de/departments/ontologies/projects/amie/

$$married(x,y) \wedge hasChild(x,z) \Rightarrow hasChild(y,z)$$

# Common-sense Rules Mining on KBs [Galarraga et al. WWW'13]

hasChild(y, x), marriedTo(y, z) => hasChild(z, x)

# Common-sense Rules Mining on KBs [Galarraga et al. WWW'13]

hasChild(y, x), marriedTo(y, z) => hasChild(z, x)

Add dangling atom ($O_D$)

# Common-sense Rules Mining on KBs [Galarraga et al. WWW'13]

hasChild(y, x), marriedTo(y, z) => hasChild(z, x)

Add dangling atom ($O_D$)

Add closing atom ($O_C$)

# Common-sense Rules Mining on KBs [Galarraga et al. WWW'13]

hasChild(y, x), marriedTo(y, z) => hasChild(z, x)

Add dangling atom ($O_D$)

Add closing atom ($O_C$)

# Common-sense Rules Mining on KBs [Galarraga et al. WWW'13]

| Dataset | Facts | Runtime | Rules |
|---|---|---|---|
| YAGO2 | 1M | 3.62min | 138 |
| YAGO2 (const) | 1M | 17.76min | 18K |
| Dbpedia (2 atoms) | 6.7M | 2.89min | 6.9K |

$$\text{marriedTo}(x,y) \wedge \text{hasChild}(y,x) \Rightarrow \text{hasChild}(z,x)$$
$$\text{hasAdvisor}(x, y) \wedge \text{graduatedFrom}(x, z) \Rightarrow \text{worksAt}(y, z)$$
$$\text{wasBornIn}(x, y) \wedge \text{isLocatedIn}(y, z) \Rightarrow \text{isCitizenOf}(x, z)$$

# Common-sense Question Answering

1 Which example describes an organism taking in nutrients?

A a dog burying a bone
B a girl eating an apple
C an insect crawling on a leaf
D a boy planting tomatoes in a garden

$H_B$ A girl eating an apple is an example of an organism taking in nutrients?

Question Interpreter

eat(girl,apple) => take-in(organism,nutrient) ? **Answer is B**

Reasoning

take-in(organism,nutrient)

Paraphrase

*An animal is an organism* | *get ≈ take in*

get(animal,nutrient)

Rule extracted from text

*"Animals must eat to get nutrients"*
*eat(animal,X) -EFFECT→ get(animal,nutrient)*

eat(animal,X)

Taxonomic

*A girl is a person is an animal*

eat(girl,apple)

Question dataset: http://allenai.org/content/data/Arisciencexams.txt

# Knowledge Fusion

- Knowledge Fusion is the process of identifying and resolving noisy or conflict information in a KB
  - Input:
    - A knowledge base K containing noisy/conflict information.
  - Output:
    - A knowledge base K' with noisy/conflict info removed
- Knowledge Fusion is a critical component in assuring knowledge base's high quality

# Fact Finder Model [X. Yin et al. KDD'07, TKDE'08]

- Models the following information:
  - Trustworthiness of information provider/source w: t(w)
  - Confidence of fact f being true: s(f)

- Iteratively compute t(w) and s(f)
  - Initially, it has little information about the web sites and the facts
  - Each iteration, improves its knowledge about trustworthiness and confidence

$$s(f) = 1 - \prod_{w \in W(f)} (1 - t(w)) \qquad t(w) = \frac{\sum_{f \in F(w)} s(f)}{|F(w)|}$$

- Stops when the computation reaches a stable state.

# Source Copying Model [X. Dong et al. VLDB'09]

- Models one more information:
  - Interdependency between sources

- Iterative process:
  - Dependence Detection
    - Two sources share a lot of claims ⇨ more likely to be dependent
    - Accuracy of S1∩S2 highly different from accuracy of S1 ⇨ S1 copy from S2
  - Fact Confidence Computation
    - Ignore the sources which are "copy cats"
  - Source Trustworthiness Computation
    - Ignore the credits from "copying"

# Latent Truth Model [B. Zhao *et al*. VLDB'12]

- Models sources' aggressiveness/conservativeness:
  - Sensitivity or Recall of source s is the probability of true facts being claimed as true. And 1-sensitivity is known as the false negative rate.

    Sensitivity = $\frac{TP_s}{TP_s+FN_s}$
  - Specificity of source s is the probability of false facts being claimed as false. And 1-specificity is known as the false positive rate.

    Specificity = $\frac{TN_s}{FP_s+TN_s}$
  - High Sensitivity ⇔ Aggressive
  - High Specificity ⇔ Conservative

**Confusion matrix of source $s$**

| | $t = True$ | $t = False$ |
|---|---|---|
| $o = True$ | True Positives ($TP_s$) | False Positives ($FP_s$) |
| $o = False$ | False Negatives ($FN_s$) | True Negatives ($TN_s$) |

# Latent Truth Model [B. Zhao *et al*. VLDB'12]

Graphical Representation

- For each source **k**:
  - Generate false positive rate $\phi_k^0 \sim Beta(\alpha_{0,1}, \alpha_{0,0})$
  - Generate sensitivity/recall $\phi_k^1 \sim Beta(\alpha_{1,1}, \alpha_{1,0})$
- For each fact **f**:
  - Generate its prior truth probability $\theta_f \sim Beta(\beta_1, \beta_0)$
  - Generate its truth label $t_f \sim Bernoulli(\theta_f)$
- For each claim **c** of fact **f**, generate observation **o** of **c**
  - If **f** is false, use false positive rate of **s** $o_c \sim Bernoulli(\phi_{s_c}^0)$
  - If **f** is true, use sensitivity of **s** $o_c \sim Bernoulli(\phi_{s_c}^1)$
- MAP inference: find truth assignment that maximizes posterior probabilities $\hat{\boldsymbol{t}}_{MAP} = \arg\max_{\boldsymbol{t}} \int\int\int p(\boldsymbol{o}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\phi}^0, \boldsymbol{\phi}^1) \mathrm{d}\boldsymbol{\theta} \mathrm{d}\boldsymbol{\phi}^0 \mathrm{d}\boldsymbol{\phi}^1$

# Google Knowledge Vault [Dong *et al.* KDD'14]

# Fact extraction from the Web

# Fact extraction from NL text

- First identify named entities (entity linking)
- Then classify verb phrase as one of 2000 relations

- The result is a probabilistic triple:
  - Pr(<subject, rel, object>=1 | text)
- Classifier trained using distant supervision

# Fact extraction from DOM Trees

- First identify named entities on page
- Then classify X-path connecting each entity pair as one of 2000 relations

# Fact extraction from Tables

## Caldecott Medal

From Wikipedia, the free encyclopedia

| | Illustrator | Title |
|---|---|---|
| 1999 | Mary Azarian | *Snowflake Bentley* |
| 1954 | Ludwig Bemelmans | *Madeline's Rescue* |
| 1983 | Marcia Brown | *Shadow* |
| 1962 | Marcia Brown | *Once a Mouse* |
| 1955 | Marcia Brown | *Cinderella, or the Little Glass Slipper* |
| 1943 | Virginia Lee Burton | *The Little House* |
| 1980 | Barbara Cooney | *Ox-Cart Man* |
| 1959 | Barbara Cooney | *Chanticleer and the Fox* |

# Fact extraction from schema.org annotation

- About 20% of webpages have machine-readable annotations of commercial events, products, etc.
- Automatically map to Freebase schema.
- We still need to do entity linking.

```
<script type="application/ld+json">
{"@context" : "http://www.schema.org",
"@type" : "Event",
"startDate" : "2014-07-26",
...}
</script>
```

# Fusing the extractors

- Construct a feature vector ***f(t)*** for each extracted triple t, then apply a binary classifier to compute Pr(t = 1| ***f(t)***)
- ***f(t)*** is composed of two numbers for each extractor:
  - square root of #sources that the extractor extracted this triple from
  - mean score of extractions from this extractor, averaging over sources
- The classifier learns a different weight for each component of this feature vector, and hence can learn the relative reliabilities of each extractor.

# Performance comparison of extractors

# Graph-based Priors

# Path Ranking Algorithm

CityLocatedInCountry(Pittsburgh) = ?

| Feature = Typed Path | Feature Value | Logistic Regresssion Weight |
|---|---|---|
| CityInState, CityInstate$^{-1}$, CityLocatedInCountry | 0.8 | 0.32 |
| AtLocation$^{-1}$, AtLocation, CityLocatedInCountry | 0.6 | 0.20 |
| ... | ... | ... |

CityLocatedInCountry(Pittsburgh) = U.S. p=0.58

# Neural Network Model

- Represent each entity and relation by its own low-dimensional (~60D) embedding vector.

$$\Pr(G(s,p,o)=1)=\sigma\left(\sum_{k=1}^{K} u_{sk} w_{pk} v_{ok}\right) \tag{1}$$

where $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid or logistic function, and $K \sim 60$ is the number of hidden dimensions. Here $\vec{u}_s$, $\vec{w}_p$ and $\vec{v}_o$ are $K$-dimensional vectors, which embed the discrete tokens into a low dimensional "semantic" space.

# Fusing the priors

- Construct a feature vector ***f(t)*** for each extracted triple t, then apply a binary classifier to compute Pr(t = 1| ***f(t)***)

- ***f(t)*** is composed of two numbers for each prior:
  - confidence value
  - indicator value specifying if the prior was able to predict or not

- The classifier learns a different weight for each component of this feature vector, and hence can learn the relative reliabilities of each prior.

# Performance comparison of priors

# Fusing extractions with priors

# Knowledge Vault

| Name | # Entity types | # Entity instances | # Relation types | # Confident facts (relation instances) |
|---|---|---|---|---|
| *Knowledge Vault (KV)* | 1100 | 45M | 4469 | 271M |
| DeepDive [32] | 4 | 2.7M | 34 | 7M[a] |
| NELL [8] | 271 | 5.19M | 306 | 0.435M[b] |
| PROSPERA [30] | 11 | N/A | 14 | 0.1M |
| YAGO2 [19] | 350,000 | 9.8M | 100 | 4M[c] |
| Freebase [4] | 1,500 | 40M | 35,000 | 637M[d] |
| Knowledge Graph (KG) | 1,500 | 570M | 35,000 | 18,000M[e] |

Table 1: Comparison of knowledge bases. KV, DeepDive, NELL, and PROSPERA rely solely on extraction, Freebase and KG rely on human curation and structured sources, and YAGO2 uses both strategies. Confident facts means with a probability of being true at or above 0.9.

# Path Ranking Algorithm

For any relation path $P = R_1 \ldots R_\ell$ and a seed node $s \in domain(P)$, a *path constrained random walk* defines a distribution $h_{s,P}$ recursively as follows. If $P$ is the empty path, then define

$$h_{s,P}(e) = \begin{cases} 1, & \text{if } e = s \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

If $P = R_1 \ldots R_\ell$ is nonempty, then let $P' = R_1 \ldots R_{\ell-1}$, and define

$$h_{s,P}(e) = \sum_{e' \in range(P')} h_{s,P'}(e') \cdot P(e|e'; R_\ell), \quad (5)$$

where $P(e|e'; R_\ell) = \frac{R_\ell(e',e)}{|R_\ell(e',\cdot)|}$ is the probability of reaching node $e$ from node $e'$ with a one step random walk with edge type $R_\ell$. $R(e', e)$ indicates whether there exists an edge with type $R$ that connect $e'$ to $e$.