# Unsupervised Template Mining for Semantic Category Understanding

**Lei Shi**[1,2], Shuming Shi[3], Chin-Yew Lin[3], Yi-Dong Shen[1], Yong Rui[3]

[1]Institute of Software, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]Microsoft Research

EMNLP 2014, Doha, Qatar

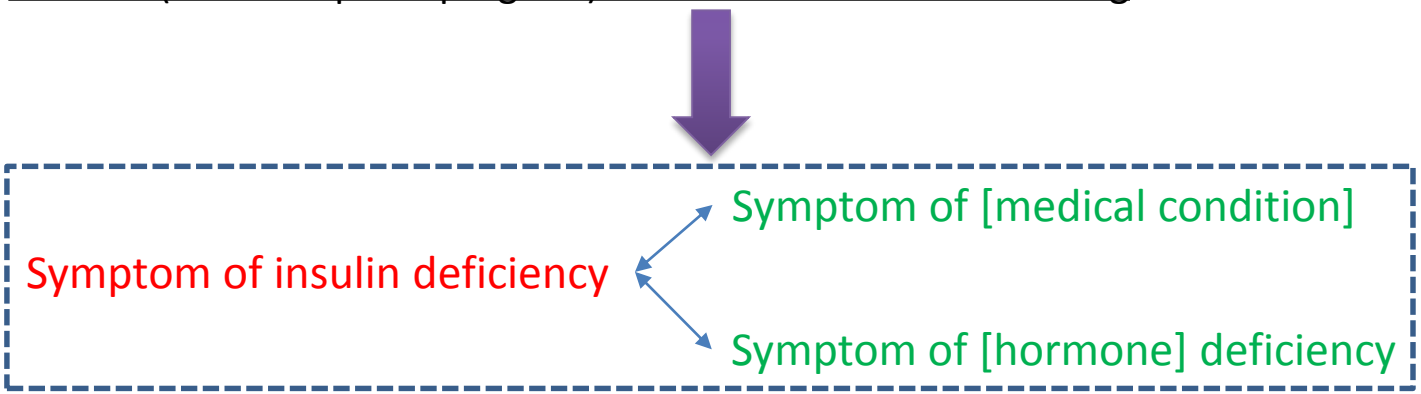
# Semantic category names*

- A plain string which can describe a set of items sharing common semantic properties
  - {Carnival, Christmas,…} → national holiday of Brazil
  - {Nocturia, weight loss,…} → symptom of insulin deficiency
- Manually edited
  - Existing knowledge bases, like Wikipedia
- Automatic extraction
  - Hypernymy (is-A) relation extraction techniques

*The term Category name and category used interchangeably in this slide.

# Understand category names

- Category names are in plain text
- Internal structures of category names
  - A set of category names : {CEO of General Motors, CEO of Yahoo, …}
  - A template : CEO of [company]
- Potential applications
  - Additional features (web search and question answering)
  - Cleaning of noisy category names collection (***promising results in our experiments!***)
  - Possible (for a computer program) to infer the semantic meaning



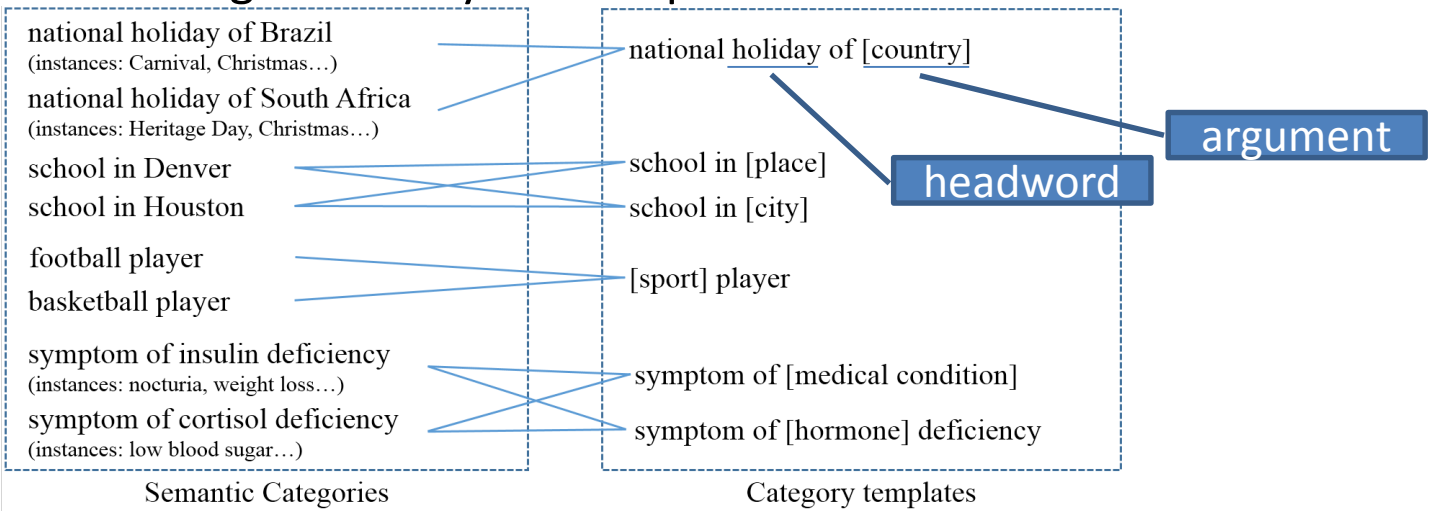
How to get these
*templates automatically from*
*a large collection of category names*?

# Outline

- The problem
- Approach
- Experiments
- Related work
- Conclusion

# Problem definition

- Input : a large collection of category names
  - Perform hyernymy extraction on 3 billion English pages
  - 40 million terms, **74 million hypernyms** and 321 million edges (term→hypernym)
  - All the multi-word hypernyms are used as the category name collection
- Output : a list of templates
  - Template: Multi-word string with one headword and several arguments
  - A score indicating how likely the template is valid



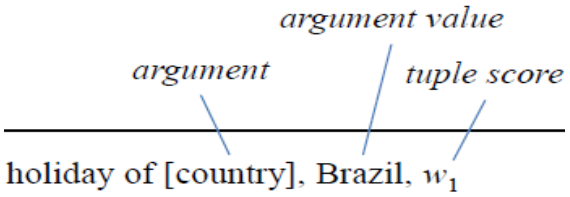
# Problem analysis

- A straightforward way to get templates
  - Divide & Replace (we have a term→hypernym map)
    - Divide : CEO of Delphinus → CEO + of + Delphinus
    - Replace : CEO of [company] (v) CEO of [constellation] (×)
- Main Challenge
  - Ambiguity: many segments have multiple meanings
  - CEO of [constellation] (a wrong template!)

# Approach

# Intuitive approach

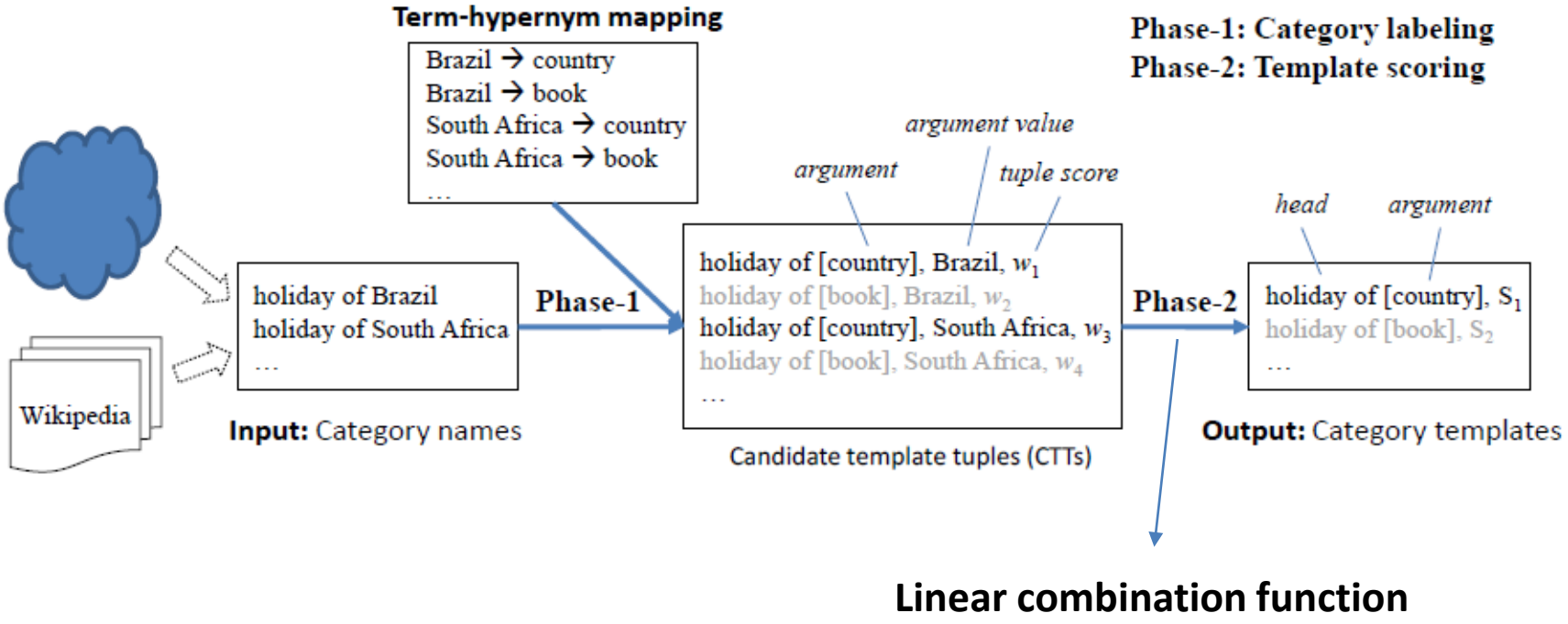
- Category labeling
  - Category segmentation
    - Divide each category into multiple segments
    - Each segment is one word or phrase in an entity dictionary
    - e.g. holiday of South Africa (holiday + of + South Africa)
  - Segment to hypernym
    - We use a term→hypyrnym mapping from a dump of Freebase
    - Hint: no weight in the mapping
  - Candidate Template Tuple (CTT) generation
    - $U_1$: (holiday of [country], Brazil, $w_1$)
    - $U_2$: (holiday of [book], Brazil, $w_2$)
- Template scoring
  - Merge all the CTTs for each template
  - e.g. holiday of [country]
    - $U_1$: (holiday of [country], South Africa, $w_1$)
    - $U_2$: (holiday of [country], Brazil, $w_2$)
    - $U_3$: (holiday of [country], Germany, $w_3$)
    - …
  - $\vec{U} = \{U_1, U_2, U_3, \dots\}$



# Intuitive approach (cont.)

- Scoring function (a TF-IDF style function)
  - $F(\vec{U}) = \sum_{i=1}^{n} w_i \cdot IDF(h)$ (linear combination function)
- h : the argument type (like, [country] in holiday of [country])
  - $IDF_1(h) = log \frac{1+N}{1+DF(h)}$
    - N is the total number of terms in term→hypernym mapping
    - DF(h) is the number of terms belong to hypernym h
  - $IDF_2(h) = \frac{1}{sqrt(DF(h))}$
- Estimation of tuple score $w_i$
  - $w_i = 1$
  - No weight information in the term→hypeynym mapping of Freebase

# Intuitive approach (cont.)



# Approach: Enhancing Template Scoring

- Enhancing tuple scoring
  - Leveraging statistical information from large corpus to estimate tuple score $w_i$
- Enhancing tuple combination function
  - Limitations of linear combination function
  - Nonlinear functions
- Refinement with term similarity and terms clusters
  - Building term clusters
  - Refining template score

# Enhancing tuple scoring

- Intuition
  - $U_1$: (holiday of [country], South Africa, $w_1$)
  - $U_2$: (holiday of [book], South Africa, $w_2$)
  - "South Africa" is more likely to be a country than a book , $w_1 > w_2$
- The idea : performing statistics in a large corpus
  - Get the popularity F of (term, hypernym) by referring to a corpus
  - $w_i = \log\big(1 + F(v,h)\big)$
    - v indicates the argument value and h indicates the argument type
  - $w_i = \frac{F(v,h)}{\gamma + \sum_{h_j \in H} F(v,h_j)}$
    - v indicates the argument value; h and $h_j$ indicates the argument type

# Enhancing tuple combination function

- Definitions of some events
  - $T$ : Template T is a valid template;
  - $\bar{T}$ : T is an invalid template;
  - $E_i$ : The observation of tuple $U_i$;
- Posterior odds of event T, Given $U_1$ and $U_2$
  - Assume $E_1$ and $E_2$ are conditionally independent given $T$ or $\bar{T}$
  - $\frac{P(T|E_1,E_2)}{P(\bar{T}|E_1,E_2)} = \frac{P(T|E_1)\cdot P(\bar{T})}{P(\bar{T}|E_1)\cdot P(T)} \cdot \frac{P(T|E_2)\cdot P(\bar{T})}{P(\bar{T}|E_2)\cdot P(T)} \cdot \frac{P(T)}{P(\bar{T})}$
  - Define $G(T|E) = log\frac{P(T|E)}{P(\bar{T}|E)} - log\frac{P(T)}{P(\bar{T})}$
  - $G(T|E_1, E_2) = G(T|E_1) + G(T|E_2)$

# Enhancing tuple combination function (cont.)

- Easy to get
  - $G(T|E_1, \ldots, E_n) = \sum_{i=1}^{n} G(T|E_i)$
- Connection with $F(\vec{U}) = \sum_{i=1}^{n} w_i \cdot IDF(h)$
  - Assume $G(T|E_i) = w_i \cdot IDF(h)$
  - These two equations are in the same form!
  - Assumption: tuples are conditional independent (may not hold true in reality)
- Nonlinear functions
  - In the task of hypernymy relation extraction (Zhang et al., 2011)
  - p-Norm
    - $F(\vec{U}) = \sqrt[p]{\sum_{i=1}^{n} {w_i}^p} \cdot IDF(h) \quad (p > 1)$ (empirically setting as 2)

# Enhancing tuple combination function (cont.) : an example

- Two Templates
  - City of [country], $\left|\overrightarrow{U_A}\right| = 200$, average score for each tuple: 1.0
  - City of [book], $\left|\overrightarrow{U_B}\right| = 1000$, average score for each tuple: 0.2
- Linear functions
  - $F\left(\overrightarrow{U_A}\right) = 200 * 1.0 = 200$
  - $F\left(\overrightarrow{U_B}\right) = 1000 * 0.2 = 200$
- Nonlinear functions
  - $F\left(\overrightarrow{U_A}\right) = 14.1$
  - $F\left(\overrightarrow{U_B}\right) = 6.32$
- The score given by the nonlinear functions is more reasonable!

# Refinement with term clusters

- Intuition
  - {“city in Brazil”, “city in South Africa”, “city in China”, “city in Japan”}
  - {Brazil, South Africa, China, Japan} very similar!
  - City in [country] is more likely to be a good template
- Building term clusters
  - Term peer similarity
    - “dog” and “cat”
    - Kozareva et al., 2008; Shi et al., 2010; Agirre et al., 2009
  - Clustering
    - Choose top-30 neighbors for each term
    - Run hierarchical clustering algorithm
    - Merge highly duplicated clusters
  - Assigning top hypernyms

# Refinement with term clusters (cont.)

- Template score refinement
  - Template T with argument type **h** and supporting tuples $\vec{U} = (U_1, U_2, \dots, U_n)$ $V = (V_1, V_2, \dots V_n)$ is the corresponding argument values.
  - Observation
    - Compute the intersection of V and every term cluster
    - Good template : at least one cluster which has hypernym h and contains many elements in V
    - Bad template : only contains a few elements in V
  - Calculating supporting scores
    - $S(C,T) = k(C,V) \cdot w(C,h)$
      - C is a term cluster
  - Calculating the final template score
    - $S(T) = F(\vec{U}) \cdot S(C^*, T)$
    - $C^*$ has the maximum supporting score for T

# Experiments

# Experimental Setup

- Data source
  - A large corpus containing 3 billion English web pages
  - Extract 74 million category names
- Datasets
  - Subsets
    - Choose 20 diverse headwords from 100 random sampled headwords
    - 20 subsets : each set contains all the categories having the same headword
    - E.g., "*symptom* of insulin deficiency" and "depression *symptom*" are in the same set
  - Fullset
    - All the 74 million category names
- Labeling
  - Good (1), fair (0.5) and bad (0)
- Metric
  - precision

# Experimental Setup

- Comparing methods
  - Base : the intuitive methods
  - LW and LP: with a reasonable estimation of tuple score
  - NLW and NLP : using the nonlinear functions
  - LW+C, LP+C, NLW+C and NLP+C : refinement with term cluster
  - SC (Cheung and Li, 2012)

$w_i = \log\big(1 + F(v,h)\big)$ : LW, NLW, LW+C, NLW+C

$w_i = \frac{F(v,h)}{\gamma + \sum_{h_j \in H} F(v,h_j)}$ : LP, NLP, LP+C, NLP+C

# Template Quality Comparison

| Method | P@10 | P@20 | P@30 |
|---|---|---|---|
| Base (baseline-1) | 0.359 | 0.361 | 0.358 |
| SC (Cheung and Li, 2012) | 0.382 | 0.366 | 0.371 |
| LW (baseline-2) | 0.633 | 0.582 | 0.559 |
| NLW | 0.711 | 0.671 | 0.638 |
| LW+C | 0.813 | 0.786 | 0.754 |
| NLW+C | 0.854 | 0.833 | 0.808 |

- Base → LW : the edge weight can boost the performance
- LW → NLW : the effectiveness of nonlinear functions
- LW→LW+C and NLW→NLW+C : the effectiveness of term similarity
- The combination of the three techniques lead to the best performance

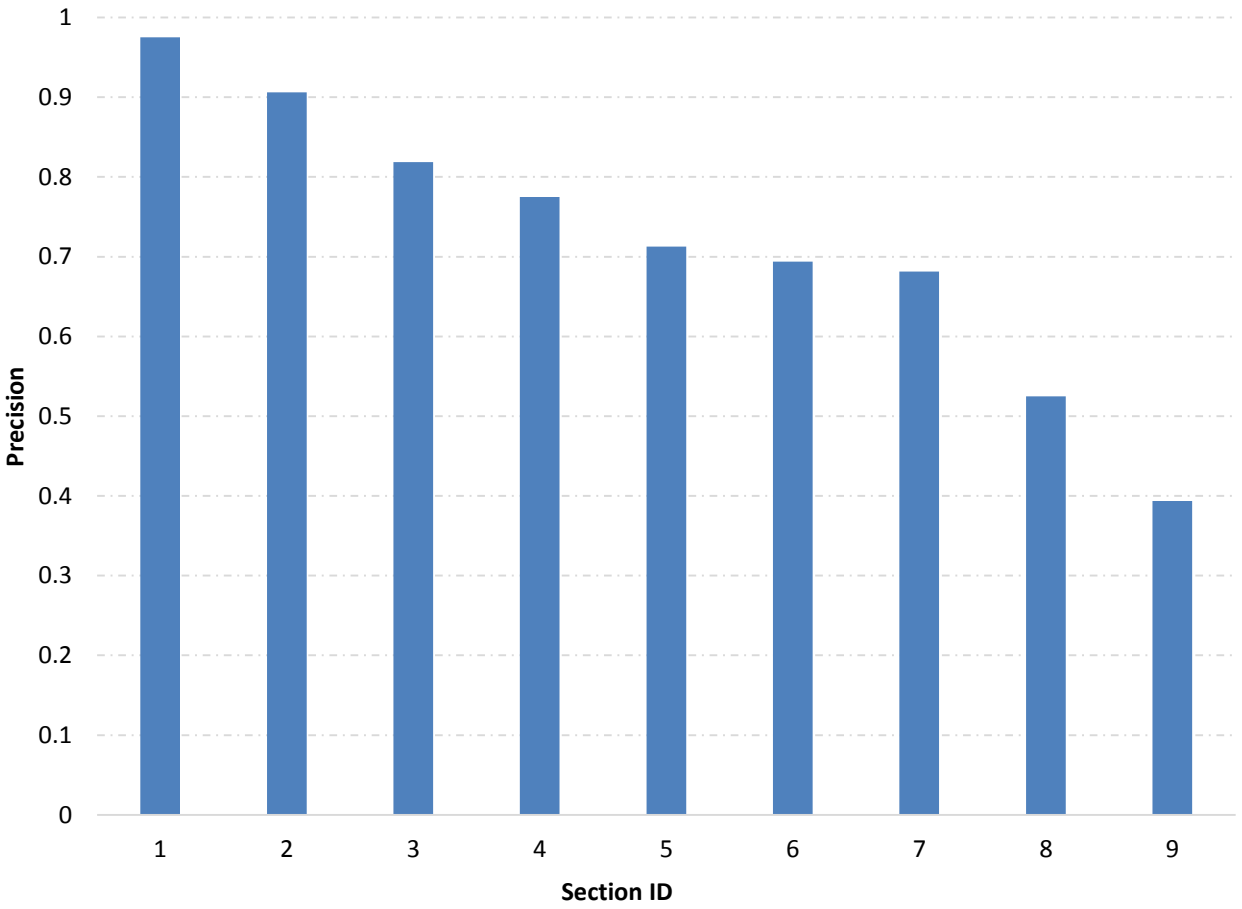
# Template Quality Comparison (cont.)

| Method | P@10 | P@20 | P@30 |
|---|---|---|---|
| Base (baseline-1) | 0.359 | 0.361 | 0.358 |
| SC (Cheung and Li, 2012) | 0.382 | 0.366 | 0.371 |
| LP (baseline-2) | 0.771 | 0.734 | 0.707 |
| NLP | 0.818 | 0.791 | 0.765 |
| LP+C | 0.818 | 0.788 | 0.778 |
| NLP+C | 0.868 | 0.839 | 0.788 |

- Base → LP : the edge weight can boost the performance
- LP → NLP : the effectiveness of nonlinear functions
- LP→LP+C and NLP→NLP+C : the effectiveness of term similarity
- The combination of the three techniques lead to the best performance

# Experimental results on Full-set

| Section ID | Range |
|---|---|
| 1 | [1~100] |
| 2 | (100~1,000] |
| 3 | (1,000~10,000] |
| 4 | (10,000~100,000] |
| 5 | (100,000~120,000] |
| 6 | (120,000~140,000] |
| 7 | (140,000~160,000] |
| 8 | (160,000~180,000] |
| 9 | (180,000~200,000] |




Performance of NLP+C method in the full-set

# Cleaning of Noisy Category Name Collection

- Category name collection is noisy
  - Automatically constructed from the web
- Basic idea
  - If a category name can match a template, it is more likely to be correct.
  - $S_{new}(H) = \log(1 + S(H)) \cdot S(T^*)$
    - $S(H)$ is the existing category score
    - $S(T^*)$ is the score of template $T^*$, $T^*$ is the best template for the category
    - Re-ranked the category names list based on the new score
  - The precision increases from 0.81 to 0.89

# Related work

- Hypernym relation extraction
  - Category names as plain text
    - Hearst (1992); Pantel and Ravichandran (2004); Van Durme and Pasca (2008); Zhang et al. (2011)
- Query understanding
  - Query tagging
    - Li et al. (2009); Reisinger and Pasca (2011)
  - Query template construction
    - Agarwal et al. (2010); Szpektor et al. (2011);Pandey and Punera (2012); Cheugn and Li (2012)
- Category name exploration
  - Third (2012); Fernandez-Breis et al. (2010); Martinez et al. (2012)

# Summary

- Mining templates to understand category names
  - Edge weight (term→hypernym)
  - Nonlinear scoring function
  - Term similarity and term clusters
- Contributions
  - First work of template generation specifically for category names in unsupervised manner
  - Extract semantic knowledge and statistical information from a web corpus for improving template generation
  - Study the characteristics of scoring function and demonstrate the effectiveness of nonlinear functions
- Future work
  - Supporting multi-argument templates
  - Applying our approach to general short text template mining

Thanks for your attention!
Questions?