Graph-Theoretic Approaches to Minimally-Supervised Natural Language Learning

Mamoru Komachi
Tokyo Metropolitan University
komachi@tmu.ac.jp

Sep 9, 2013
Web & AI Seminar
National Institute of Informatics
Corpus-based extraction of semantic knowledge

Input
Instance
Singapore

(Extracted from corpus)
Pattern
___ visa
Singapore visa
Singapore map

Output
New instance
Hong Kong
Australia
China
Egypt

Alternate step by step
Semantic drift is the central problem of bootstrapping.

Generic patterns
Patterns co-occurring with many irrelevant instances

Input
Instance:
Singapore

(Extracted from corpus)
Pattern:
visa ___ is

Output
New instance:
Australia
card
messages
words

Semantic category changed!

Errors propagate to successive iteration
Two major problems solved by this work

• Why semantic drift occurs?

• Is there any way to prevent semantic drift?
Answers to the problems of semantic drift


2. Solve semantic drift using “relatedness” measure (regularized Laplacian) instead of “importance” measure (HITS authority) used in link analysis community
Espresso Algorithm [Pantel and Pennacchiotti, 2006]

- Repeat
  - Pattern extraction
  - Pattern ranking
  - Pattern selection
  - Instance extraction
  - Instance ranking
  - Instance selection
- Until a stopping criterion is met
Pattern/instance ranking in Espresso

Score for pattern $p$

$$r_\pi(p) = \frac{\sum_{i \in I} \left( \frac{pmi(i, p)}{\max_{p'mi} r_i(i)} \right) \cdot r_i(i)}{|I|}$$

Score for instance $i$

$$r_i(i) = \frac{\sum_{p \in P} \left( \frac{pmi(i, p)}{\max_{p'mi} r_\pi(p)} \right) \cdot r_\pi(p)}{|P|}$$

$pmi(i, p) = \log \frac{|x, p, y|}{|x, *, y||*, p, *|}$

$p$: pattern

$i$: instance

$P$: set of patterns

$I$: set of instances

$pmi$: pointwise mutual information

$max pmi$: max of $pmi$ in all the patterns and instances
Espresso uses pattern-instance matrix $M$ for ranking patterns and instances

$|P| \times |I|$-dimensional matrix holding the (normalized) pointwise mutual information (pmi) between patterns and instances

$\begin{bmatrix}
1 & 2 & \ldots & i & \ldots & |I| \\
1 & \vdots & & \vdots & & \vdots \\
|P| & p & & & & \\
\end{bmatrix}$

$[M]_{p,i} = \frac{\text{pmi}(p,i)}{\max_{p,i} \text{pmi}(p,i)}$
Pattern/instance ranking in Espresso

\[ p = \text{pattern score vector} \]
\[ i = \text{instance score vector} \]
\[ M = \text{pattern-instance matrix} \]

Reliable instances are supported by reliable patterns, and vice versa.

\[
p \leftarrow \frac{1}{|I|} M i \quad \text{... pattern ranking}
\]

\[
i \leftarrow \frac{1}{|P|} M^T p \quad \text{... instance ranking}
\]

\[ |P| = \text{number of patterns} \]
\[ |I| = \text{number of instances} \]

Normalization factors to keep score vectors not too large.
Three simplifications to reduce Espresso to HITS

- Repeat
  - Pattern extraction
  - Pattern ranking
  - Pattern selection
  - Instance extraction
  - Instance ranking
  - Instance selection
- Until a stopping criterion is met

For graph-theoretic analysis, we will introduce 3 simplifications to Espresso
Keep pattern-instance matrix constant in the main loop

- Compute the pattern-instance matrix
- Repeat
  - Pattern extraction
  - Pattern ranking
  - Pattern selection
  - Instance extraction
  - Instance ranking
  - Instance selection
- Until a stopping criterion is met

Simplification 1
Remove pattern-instance extraction steps
Instead, pre-compute all patterns and instances once in the beginning of the algorithm
Remove pattern-instance selection heuristics

- Compute the pattern-instance matrix
- Repeat
  - Pattern ranking
  - Pattern selection
  - Instance ranking
  - Instance selection
- Until a stopping criterion is met

Simplification 2
Remove pattern-instance selection steps which retain only highest scoring $k$ patterns / $m$ instances for the next iteration i.e., reset the scores of other items to 0

Instead, retain scores of all patterns and instances
Remove early stopping heuristics

- Compute the pattern-instance matrix
- Repeat
  - Pattern ranking
  - Instance ranking

Simplification 3
No early stopping
i.e., run until convergence

Until score vectors p and i converge
Make Espresso look like HITS

\[ p = \text{pattern score vector} \]
\[ i = \text{instance score vector} \]
\[ M = \text{pattern-instance matrix} \]

\[
\begin{align*}
p & \leftarrow \frac{1}{|I|} M i & \text{... pattern ranking} \\
i & \leftarrow \frac{1}{|P|} M^T p & \text{... instance ranking}
\end{align*}
\]

\[ |P| = \text{number of patterns} \]
\[ |I| = \text{number of instances} \]

normalization factors to keep score vectors not too large
Simplified Espresso

Input

- Initial score vector of seed instances $i = (0,1,0,0)$
- Pattern-instance co-occurrence matrix $M$

Main loop

Repeat

\[ p \leftarrow \frac{1}{|I|} M^T i \]  \hspace{1cm} \text{... pattern ranking}

\[ i \leftarrow \frac{1}{|P|} M p \]  \hspace{1cm} \text{... instance ranking}

Until $i$ and $p$ converge

Output

Instance and pattern score vectors $i$ and $p$
HITS Algorithm [Kleinberg 1999]

**Input**
- Initial hub score vector \( h = (1, 1, 1, 1) \)
- Adjacency matrix \( M \)

**Main loop**
Repeat
\[
\begin{align*}
\mathbf{a} &\leftarrow \alpha \mathbf{M}^T \mathbf{h} \\
\mathbf{h} &\leftarrow \beta \mathbf{M} \mathbf{a}
\end{align*}
\]
Until \( \mathbf{a} \) and \( \mathbf{h} \) converge

**Output**
Hub and authority score vectors \( \mathbf{a} \) and \( \mathbf{h} \)

\( \alpha \): normalization factor
\( \beta \): normalization factor
Simplified Espresso is HITS

Simplified Espresso = HITS in a bipartite graph whose adjacency matrix is $M$

**Problem**
The ranking vector $\mathbf{i}$ tends to the principal eigenvector of $M^T M$ as the iteration proceeds regardless of the seed instances!

→ No matter which seed you start with, the same instance is always ranked topmost

→ Semantic drift (also called topic drift in HITS)
How about Espresso?

Espresso has heuristics not present in Simplified Espresso

- Early stopping
- Pattern and instance selection

Do these heuristics really help reduce semantic drift?
Experiments on semantic drift

Does the heuristics in original Espresso help reduce drift?
Word sense disambiguation task of Senseval-3
English Lexical Sample

Predict the sense of “bank”

… the financial benefits of the bank (finance)'s employee package (cheap mortgages and pensions, etc), bring this up to …

In that same year I was posted to South Shields on the south bank (bank of the river) of the River Tyne and quickly became aware that I had an enormous burden possibly aligned to water a sort of bank(???) by a rushing river.

Training instances are annotated with their sense

Predict the sense of target word in the test set
Senseval-3 word sense disambiguation task

System output = k-nearest neighbor (k=3)
i=(0.9, 0.1, 0.8, 0.5, 0, 0, 0.95, 0.3, 0.2, 0.4) → sense A
Seed instance = the instance to predict its sense
System output = k-nearest neighbor (k=3)

Heuristics of Espresso
• Pattern and instance selection
  • # of patterns to retain $p=20$ (increase $p$ by 1 on each iteration)
  • # of instance to retain $m=100$ (increase $m$ by 100 on each iteration)
• Early stopping
Heuristics in Espresso helps reducing semantic drift (However, early stopping is required for optimal performance)

Output the most frequent sense regardless of input

Semantic drift occurs (always outputs the most frequent sense)
Learning curve of Espresso: per-sense breakdown

- **Most frequent sense**
- 
  # of most frequent sense predictions increases

- **Precision for infrequent senses worsens even with original Espresso**

- **Other senses**
Summary: Espresso and semantic drift

Semantic drift happens because
- Espresso is designed like HITS
- HITS gives the same ranking list regardless of seeds

Some heuristics reduce semantic drift
- Early stopping is crucial for optimal performance

Still, these heuristics require
- many parameters to be calibrated
- but calibration is difficult
Main contributions of this work

1. Suggest a parallel between semantic drift in Espresso-like bootstrapping and topic drift in HITS (Kleinberg, 1999)

2. Solve semantic drift by graph-based approaches used in link analysis community
Q. What caused drift in Espresso?

A. Espresso's resemblance to HITS

HITS is an importance computation method (gives a single ranking list for any seeds)

Why not use a method for another type of link analysis measure - which takes seeds into account? "relatedness" measure (it gives different rankings for different seeds)
The regularized Laplacian

- A relatedness measure
- Has only one parameter

Normalized Graph Laplacian

\[ L = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} \]

Regularized Laplacian matrix

\[ \mathbf{R}_\alpha = \sum_{n=0}^{\infty} \alpha^n (-\mathbf{L})^n = (\mathbf{I} + \alpha \mathbf{L})^{-1} \]

A: similarity matrix of the graph
D: (diagonal) degree matrix

\( \alpha \): parameter
Each column of \( \mathbf{R}_\alpha \) gives the rankings relative to a node
The von Neumann kernel

- A mixture of relatedness and importance measure [Ito+ 08]
- Has only one parameter
- Small $\alpha$: relatedness measure (co-citation matrix)
- Large $\alpha$: importance measure (HITS authority vector)

Von Neumann kernel matrix

$$K_\alpha = A \sum_{n=0}^{\infty} \alpha^n A^n = A(I - \alpha A)^{-1}$$

- $A$: similarity matrix of the graph
- $D$: (diagonal) degree matrix

$\alpha$: parameter
Each column of $K_\alpha$ gives the rankings relative to a node
Condition of diffusion parameter

- For convergence, diffusion parameter $\alpha$ should be in range

$$0 \leq \alpha < \lambda^{-1}$$

where $\lambda$ is the principal eigenvalue of $A$.

Regularized Laplacian

- $\alpha = 0$ : identity matrix
- $\alpha \rightarrow \infty$ : uniform distribution

von Neumann kernel

- $\alpha = 0$ : co-citation matrix
- $\alpha \rightarrow \infty$ : HITS authority vector
K-step approximation to speed up the computation

- Proposed kernels require $O(n^3)$ time complexity ($n$: # of nodes) which is intractable for large graphs
- K-step approximation takes only the first $k$ terms:

$$R_{\alpha} = \sum_{n=0}^{\infty} \alpha^n (-L)^n = I + \alpha (-L) + \alpha^2 (-L)^2 + \cdots$$

- K-step approximation = bootstrapping terminated at the $K$-th iteration
- Error is upper bounded by $$\left(V \mid V \mid/k!\right)\left((\alpha \lambda)^{-1} - 1\right)^{-1/2}$$
where $V$ is a volume of the matrix
Memory efficient computation of the regularized Laplacian

- Similarity matrix is large and dense; adjacency matrix is often large but sparse

\[ R_\alpha(k + 1) = \sum_{n=0}^{k+1} \alpha^n (-L)^n \]

\[ = \alpha A R_\alpha(k) + (1 - \alpha) R_\alpha(0) \]

\[ = -\alpha R_\alpha(k) + \alpha D^{-1/2} MM^T D^{-1/2} R_\alpha(k) + (1 - \alpha) R_\alpha(0) \]

- After this factorization, space complexity reduces to O(npk) where \( n \) is the number of nodes, \( p \) is the number of pattern, and \( k \) is the number of steps
Experiments

Properties of the regularized Laplacian and the von Neumann kernel
Label prediction of “bank” (F measure)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Most frequent sense</th>
<th>Other senses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplified Espresso</td>
<td>100.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Espresso (after convergence)</td>
<td>100.0</td>
<td>30.2</td>
</tr>
<tr>
<td>Espresso (optimal stopping)</td>
<td>94.4</td>
<td>67.4</td>
</tr>
<tr>
<td>Regularized Laplacian ($\beta = 10^{-2}$)</td>
<td>92.1</td>
<td>62.8</td>
</tr>
</tbody>
</table>

Espresso suffers from semantic drift (unless stopped at optimal stage)

The regularized Laplacian keeps high recall for infrequent senses
WSD on all nouns in Senseval-3

<table>
<thead>
<tr>
<th>algorithm</th>
<th>F measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most frequent sense (baseline)</td>
<td>54.5</td>
</tr>
<tr>
<td>HyperLex</td>
<td>64.6</td>
</tr>
<tr>
<td>PageRank</td>
<td>64.6</td>
</tr>
<tr>
<td>Simplified Espresso</td>
<td>44.1</td>
</tr>
<tr>
<td>Espresso (after convergence)</td>
<td>46.9</td>
</tr>
<tr>
<td>Espresso (optimal stopping)</td>
<td>66.5</td>
</tr>
<tr>
<td>Regularized Laplacian ($\beta = 10^{-2}$)</td>
<td>67.1</td>
</tr>
</tbody>
</table>

Espresso needs optimal stopping to achieve an equivalent performance.

Outperforms other graph-based methods.
Regularized Laplacian is stable across a parameter$

![](image)

Accuracy

Diffusion factor $\alpha$
von Neumann Kernel tends to HITS authority

Accuracy

Diffusion factor $\alpha$

von Neumann kernel
Simplified Espresso
most frequent sense
Conclusions

- **Semantic drift** in Espresso is a parallel form of topic drift in HITS

- The regularized Laplacian reduces semantic drift in bootstrapping for natural language processing tasks
  - inherently a relatedness measure (↔ importance measure)
Future work

• Investigate if a similar analysis is applicable to a wider class of bootstrapping algorithms (including co-training)

• Investigate the influence of seed selection to bootstrapping algorithms and propose a way to select effective seed instances

• Explore multi-class classification problems in bootstrapping algorithms
References


• Tetsuo Kiso, Masashi Shimbo, Mamoru Komachi and Yuji Matsumoto. HITS-based Seed Selection and Stop List Construction for Bootstrapping. ACL HLT 2011.