Building Re-usable Dictionary Repositories for Real-world Text Mining

JongHeum Yeon

Intelligent Data Systems Laboratory

2011-01-28

S. Godbole$^1$, I. Bhattacharya$^2$, A. Gupta$^1$, A. Verma$^1$

$^1$IBM Research - India

$^2$Indian Institute of Science
Motivation I

- Dictionary-based text mining
  - Information extraction, Entity annotation, Classification, Link analysis tasks.

- Statistical and model-based techniques thrive academic research in text mining and NLP

- Real-world industrial products and services prefer tuned linguistic rule-sets and pre-packaged dictionary-based components
  - The variance in the skill levels of end users
Motivation II

- Constructing dictionaries (e.g., WordNet)
  - Experience
  - A lot of trial and error
- How tooling and automation can help text mining practitioners construct
- Use and evolve their ‘private Wordnets’, in the form of dictionary repositories.
  - Dictionary construction
  - Dictionary adaptation across tasks
Challenges: Building dictionaries

- Verification is an easier text mining task than specification for users.
- How can we best assist the practitioner in constructing these dictionaries?
- Posing dictionary construction task as a ranking problem
  - To enable the user to provide feedback on a ranking with multiple seeds, how should the ranking be ‘explained’ to the user?
  - Usually several concept dictionaries need to be constructed for a dataset
  - Are there benefits to constructing them simultaneously, rather than sequentially?
Challenges: Re-using dictionaries

- Allowing practitioners and domain experts to re-use their own or others’ effort in past tasks and engagements can significantly speed up text mining.
  - How can a dictionary ontology be adapted for a new domain?
  - What are the best interaction models for dictionary adaptation?
Building Dictionaries

- Dictionary $D = \text{Dict}(C, X)$
  - A set of words that refer to or describe a semantic concept $C$ in a document collection $X$

- Dictionary construction problem: Given a semantic concept $C$ and a document collection $X$ using a vocabulary $V$, return a ranking over words $w \in V$ such that words that refer to concept $C$ appear higher in the ranking than words that do not refer to $C$.

- The general assumption is that ‘critiquing is easier than constructing’
Single Seed Word I

- the user provides a single word $w^s$ as an illustrative example of words corresponding to a concept $C$
  - e.g. Car Parts: engine
  - $w^s$ seed word for $C$
- Dictionary construction task: to rank words in $V$ according to ‘semantic similarity’ to the seed word $w^s$
- Basic assumption: words that are used in similar local contexts over all documents in the collection are similar in meaning for that collection
- Local context around a word $w$ occurring in a document by considering a context window of length $l$ centered around $w$ considering words that appear within the context window
- Weight vector $WV(w)$ over words in vocabulary $V$
Single Seed Word II

- the weight $WV(w, w')$ for any word $w'$ captures the number of times (TF-IDF) $w'$ has appeared in context windows around $w$ over all documents in the collection.

- $sim(w_1, w_2)$: cosine similarity

```
Algo BuildDict(Word \(w^s\), Int k, Double t)
1. Initialize candidate set $CS$ to empty set
2. For all words in $WV(w^s)$
3.   If $WV(w^s, w') >$ some threshold $t'$
4.     Add $w'$ to $CS$
6. For each word $w'$ in $CS$
7.   Compute similarity $sim(w^s, w')$ with $w^s$
8.   Reject $w'$ if similarity below threshold $t$
9. Sort remaining words by similarity and return top $k$
```

**Figure 1:** Dictionary building algorithm with one Seed Word
Set of Seed Words I

- In general the user may provide not one but a set of keywords for specifying a concept \( C \)
  
  - e.g., Car Parts: engine, tires, brakes, gear

```
Algorithm BuildDict(Seed Set S, Int k, Double t, Operator +)

1. Initialize candidate set CS to empty set
2. For each word \( w_i^s \in S \)
3.   For all words in \( WV(w_i^s) \)
4.     If \( WV(w_i^s, w') > \text{some threshold } t' \)
5.     Add \( w' \) to \( CS \)
6. For each word \( w' \) in \( CS \)
7.   For each word \( w_i^s \) in \( S \)
8.     Compute similarity \( \text{sim}(w_i^s, w') \)
9. Aggregate similarity for \( w' \) using operator +
10. Reject \( w' \) if similarity below threshold \( t \)
11. Sort remaining words by similarity and return top \( k \)
```

**Figure 2: Dictionary building algorithm with a Seed Set**
Positive and Negative Seed Sets I

- User provides two seed sets, $P^s$ and $N^s$
  - Contact Center Agent
  - Positive: agent, rep, representative
  - Negative: manager, mgr, supervisor
- $+\{w^p_i \in P_s\} \text{AND NOT}\{w^n_i \in N_s\}$
  - $+$ may be $\text{AND}$ or $\text{OR}$
- Algorithm in Figure 2 and $\text{BuildDict}(S^p, S^n, k, t, +)$.
- The only difference appears in Step 9, where we compute the aggregate similarity by taking into account the negative seed words
Interactive Supervision

• the user starts off with a small set of words, inspects the results, selects and rejects words from the returned ranking, and iterates until he is satisfied

Algo InteractiveBuildDict(Seed Set S)

1. Initialize $S^p$ to $S$ and $S^n$ to empty set
2. Initialize length $k$, thresh $t$, operator +
3. Invoke BuildDict($S^p, S^n, k, t, +$) of Sec. 4.3 for ranking $R$
4. While user is not satisfied with $R$
5. Refine $S^p$ and $S^n$ from feedback
6. Get optional feedback on $k$, $t$, +
5. Rebuild $R$ using BuildDict($S^p, S^n, k, t, +$) of Sec. 4.3

Figure 3: The Interactive Dictionary Building Framework
Multi Dictionary Construction

- In the multi-dictionary construction approach, the user provides $n$ seed sets (or pairs of positive and negative seed sets) to the algorithm and gets back $n$ different rankings, one corresponding to each (or each pair) of the seed sets.
- The additional knowledge available to the algorithm is that the same output word should not appear in more than one returned ranking.
Seeding using Context

• For example, when ‘agent’ is provided as the seed word, no similar words can be found for a document collection where ‘agent’ is never used and the appropriate word is ‘representative’ instead.
• In an alternative model, we may imagine the user as directly providing the ‘meaning’ of the words by specifying a context vector.
Re-using Dictionaries

- The dictionary $n.D$ associated with a concept node can directly be used as a positive seed set for the algorithm.
- The ranking that is returned is inspected by the user to create the adapted dictionary $\text{adapt}(n.D)$ for the new document collection.
Experiments I

Datasets and Tasks

- CRM Analytics
  - Several CRM voice of customer (VOC) datasets in the form of customer satisfaction (CSat) survey forms, from electronics (15,000 surveys), telecom (20,000), and automobile (6,000) company helpdesks
  - Find reasons for dissatisfied customers, and subsequently implement operational improvements in call centers
## Experiments II

<table>
<thead>
<tr>
<th>Concept</th>
<th>Iter#</th>
<th>OR/AND</th>
<th>Synonym list with feedback</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>Recall&lt;sub&gt;max&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pleasant agent</td>
<td>1</td>
<td>OR</td>
<td>polite&lt;sup&gt;+&lt;/sup&gt;, helpful&lt;sup&gt;+&lt;/sup&gt;, friendly&lt;sup&gt;+&lt;/sup&gt;, professional&lt;sup&gt;+&lt;/sup&gt;, courteous&lt;sup&gt;+&lt;/sup&gt;, nice&lt;sup&gt;+&lt;/sup&gt;, pleasant&lt;sup&gt;+&lt;/sup&gt;, discouraged&lt;sup&gt;-&lt;/sup&gt;, aspects&lt;sup&gt;-&lt;/sup&gt; polite&lt;sup&gt;+&lt;/sup&gt;, helpful&lt;sup&gt;+&lt;/sup&gt;, friendly&lt;sup&gt;+&lt;/sup&gt;, professional&lt;sup&gt;+&lt;/sup&gt;, courteous&lt;sup&gt;+&lt;/sup&gt;, nice&lt;sup&gt;+&lt;/sup&gt;, pleasant&lt;sup&gt;+&lt;/sup&gt;, very&lt;sup&gt;+&lt;/sup&gt;, good&lt;sup&gt;+&lt;/sup&gt;</td>
<td>91.4%</td>
<td>43.2%</td>
<td>58.6%</td>
<td>57.6%</td>
</tr>
<tr>
<td>(seed: kind)</td>
<td>2</td>
<td>OR</td>
<td></td>
<td>80.1%</td>
<td>51%</td>
<td>62.3%</td>
<td></td>
</tr>
<tr>
<td>Timeliness</td>
<td>1</td>
<td>OR</td>
<td>forever&lt;sup&gt;+&lt;/sup&gt;, hung&lt;sup&gt;+&lt;/sup&gt;, minute&lt;sup&gt;+&lt;/sup&gt;, while&lt;sup&gt;+&lt;/sup&gt;, years&lt;sup&gt;+&lt;/sup&gt;, finished&lt;sup&gt;-&lt;/sup&gt;, 20&lt;sup&gt;-&lt;/sup&gt;, 15&lt;sup&gt;-&lt;/sup&gt;, 16&lt;sup&gt;-&lt;/sup&gt; forever&lt;sup&gt;+&lt;/sup&gt;, hung&lt;sup&gt;+&lt;/sup&gt;, minute&lt;sup&gt;+&lt;/sup&gt;, while&lt;sup&gt;+&lt;/sup&gt;, years&lt;sup&gt;+&lt;/sup&gt;, atleast&lt;sup&gt;+&lt;/sup&gt;, specifically&lt;sup&gt;-&lt;/sup&gt;, clock&lt;sup&gt;-&lt;/sup&gt;, wait&lt;sup&gt;-&lt;/sup&gt;</td>
<td>57.1%</td>
<td>6.3%</td>
<td>11.3%</td>
<td>33.8%</td>
</tr>
<tr>
<td>(seed: minutes)</td>
<td>2</td>
<td>OR</td>
<td></td>
<td>59.5%</td>
<td>16.6%</td>
<td>25.9%</td>
<td></td>
</tr>
<tr>
<td>Car parts</td>
<td>1</td>
<td>AND</td>
<td>opener&lt;sup&gt;+&lt;/sup&gt;, locks&lt;sup&gt;+&lt;/sup&gt;, mat&lt;sup&gt;+&lt;/sup&gt;, latch&lt;sup&gt;+&lt;/sup&gt;, trunk&lt;sup&gt;+&lt;/sup&gt;, automatic&lt;sup&gt;+&lt;/sup&gt;, garage&lt;sup&gt;-&lt;/sup&gt;, 60&lt;sup&gt;-&lt;/sup&gt;, open opener&lt;sup&gt;+&lt;/sup&gt;, locks&lt;sup&gt;+&lt;/sup&gt;, mat&lt;sup&gt;+&lt;/sup&gt;, latch&lt;sup&gt;+&lt;/sup&gt;, trunk&lt;sup&gt;+&lt;/sup&gt;, automatic&lt;sup&gt;+&lt;/sup&gt;, side&lt;sup&gt;+&lt;/sup&gt;, home&lt;sup&gt;-&lt;/sup&gt;, motor&lt;sup&gt;-&lt;/sup&gt;</td>
<td>43.7%</td>
<td>1.6%</td>
<td>3.1%</td>
<td></td>
</tr>
<tr>
<td>(seed: door)</td>
<td>2</td>
<td>AND</td>
<td></td>
<td>63.4%</td>
<td>5.2%</td>
<td>9.6%</td>
<td>40.2%</td>
</tr>
</tbody>
</table>

Table 1: Utility of interactive dictionary construction
## Experiments III

<table>
<thead>
<tr>
<th>Dictionary</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electronics ‘Pleasant agent’ → Automobile ‘Pleasant agent’</td>
<td></td>
<td></td>
</tr>
<tr>
<td>kind, thorough, courteous, helpful, polite</td>
<td>91.9%</td>
<td>29.3%</td>
</tr>
<tr>
<td>kind, thorough, courteous, helpful, polite, friendly, nice, professional, knowledgeable, pleasant</td>
<td>87.7%</td>
<td>46.6%</td>
</tr>
<tr>
<td>Telecom ‘Timeliness’ → Automobile ‘Timeliness’</td>
<td></td>
<td></td>
</tr>
<tr>
<td>minutes, hours, wait, while, mins,</td>
<td>60.1%</td>
<td>15.2%</td>
</tr>
<tr>
<td>minutes, hours, wait, while, mins, forever, hung, putting, hold, research</td>
<td>49.6%</td>
<td>24.2%</td>
</tr>
<tr>
<td>Fictional ‘Car parts’ → Automobile ‘Car parts’</td>
<td></td>
<td></td>
</tr>
<tr>
<td>door, window, steering, tire</td>
<td>90.6%</td>
<td>13.3%</td>
</tr>
<tr>
<td>door, window, steering, tire, motor, garage, informed, open, passenger, flat</td>
<td>75.7%</td>
<td>16.1%</td>
</tr>
</tbody>
</table>

**Table 2: Adapting dictionaries**
Conclusion

- Presented interactive, corpus-aware dictionary construction, adaptation, and re-use techniques
- Showed these techniques work well practically on real-world datasets
- Proposed an evolving repository of dictionaries with associated tooling that can significantly help reduce the time and effort practitioners spend on new dictionary-based text mining tasks