Distributed word representations: Basic reweighting

Christopher Potts

Stanford Linguistics

CS224u: Natural language understanding
Goals of reweighting

- Amplify the important, the trustworthy, the unusual; deemphasize the mundane and the quirky.
- Absent a defined objective function, this will remain fuzzy.
- The intuition behind moving away from raw counts is that frequency is a poor proxy for the above values.
- So we should ask of each weighting scheme: How does it compare to the raw count values?
- What overall distribution of values does it deliver?
- We hope to do no feature selection based on counts, stopword dictionaries, etc. Rather, we want our methods to reveal what’s important without these ad hoc interventions.
Normalization

**L2 norming (repeated from earlier)**

Given a vector $u$ of dimension $n$, the L2-length of $u$ is

$$||u||_2 = \sqrt{\sum_{i=1}^{n} u_i^2}$$

and the length normalization of $u$ is

$$\left[ \frac{u_1}{||u||_2}, \frac{u_2}{||u||_2}, \cdots, \frac{u_n}{||u||_2} \right]$$

**Probability distribution**

Given a vector $u$ of dimension $n$ containing all positive values, let

$$\text{sum}(u) = \sum_{i=1}^{n} u_i$$

and then the probability distribution of $u$ is

$$\left[ \frac{u_1}{\text{sum}(u)}, \frac{u_2}{\text{sum}(u)}, \cdots, \frac{u_n}{\text{sum}(u)} \right]$$
Observed/Expected

\[
\text{rowsum}(X, i) = \sum_{j=1}^{n} X_{ij} \quad \text{colsum}(X, j) = \sum_{i=1}^{m} X_{ij} \quad \text{sum}(X) = \sum_{i=1}^{m} \sum_{j=1}^{n} X_{ij}
\]

\[
\text{expected}(X, i, j) = \frac{\text{rowsum}(X, i) \cdot \text{colsum}(X, j)}{\text{sum}(X)}
\]

\[
\text{oe}(X, i, j) = \frac{X_{ij}}{\text{expected}(X, i, j)}
\]
### Observed/Expected

**Rowsum** \( \text{rowsum}(X, i) = \sum_{j=1}^{n} X_{ij} \)

**Colsum** \( \text{colsum}(X, j) = \sum_{i=1}^{m} X_{ij} \)

**Sum** \( \text{sum}(X) = \sum_{i=1}^{m} \sum_{j=1}^{n} X_{ij} \)

\[ \text{expected}(X, i, j) = \frac{\text{rowsum}(X, i) \cdot \text{colsum}(X, j)}{\text{sum}(X)} \]

\[ \text{oe}(X, i, j) = \frac{X_{ij}}{\text{expected}(X, i, j)} \]

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>rowsum</th>
<th>oe</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>34</td>
<td>11</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>47</td>
<td>7</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>colsum</td>
<td>81</td>
<td>18</td>
<td>99</td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{array}{cc|c}
    & a & b \\
\hline
   x & 34 & 11 \quad 45 \quad \frac{45}{81} \quad \frac{45}{18} \\
   y & 47 &  7 \quad 54 \quad \frac{54}{81} \quad  \frac{7}{99} \\
\end{array}
\]
Observed/Expected

\[
\text{rowsum}(X, i) = \sum_{j=1}^{n} X_{ij} \quad \text{colsum}(X, j) = \sum_{i=1}^{m} X_{ij} \quad \text{sum}(X) = \sum_{i=1}^{m} \sum_{j=1}^{n} X_{ij}
\]

\[
\text{expected}(X, i, j) = \frac{\text{rowsum}(X, i) \cdot \text{colsum}(X, j)}{\text{sum}(X)}
\]

\[
\text{oe}(X, i, j) = \frac{X_{ij}}{\text{expected}(X, i, j)}
\]

**Observed**

<table>
<thead>
<tr>
<th></th>
<th>tabs</th>
<th>reading</th>
<th>birds</th>
</tr>
</thead>
<tbody>
<tr>
<td>keep</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>enjoy</td>
<td>1</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

*keep and tabs co-occur more than expected given their frequencies, enjoy and tabs less than expected.*

**Expected**

<table>
<thead>
<tr>
<th></th>
<th>tabs</th>
<th>reading</th>
<th>birds</th>
</tr>
</thead>
<tbody>
<tr>
<td>keep</td>
<td>\frac{60\cdot21}{101}</td>
<td>\frac{60\cdot40}{101}</td>
<td>\frac{60\cdot40}{101}</td>
</tr>
<tr>
<td>enjoy</td>
<td>\frac{41\cdot21}{101}</td>
<td>\frac{41\cdot40}{101}</td>
<td>\frac{41\cdot40}{101}</td>
</tr>
</tbody>
</table>

\[
= \frac{12.48}{16.24} \quad \frac{23.76}{16.24} \quad \frac{23.76}{16.24}
\]
Pointwise Mutual Information (PMI)

PMI is observed/expected in log-space (with $\log_e(0) = 0$):

$$pmi(X, i, j) = \log_e \left( \frac{X_{ij}}{\text{expected}(X, i, j)} \right) = \log_e \left( \frac{P(X_{ij})}{P(X_{i\ast}) \cdot P(X_{\ast j})} \right)$$

<table>
<thead>
<tr>
<th></th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
<th>$P(w, d)$</th>
<th>$P(w)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0.11</td>
<td>0.11</td>
</tr>
<tr>
<td>$B$</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0.11</td>
<td>0.00</td>
</tr>
<tr>
<td>$C$</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0.11</td>
<td>0.00</td>
</tr>
<tr>
<td>$D$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

$P(d)$  | 0.33  | 0.33  | 0.22  | 0.12  |

PMI

<table>
<thead>
<tr>
<th></th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>-0.28</td>
<td>-0.28</td>
<td>0.13</td>
<td>0.73</td>
</tr>
<tr>
<td>$B$</td>
<td>0.01</td>
<td>0.01</td>
<td>0.42</td>
<td>0.00</td>
</tr>
<tr>
<td>$C$</td>
<td>0.42</td>
<td>0.42</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>$D$</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>2.11</td>
</tr>
</tbody>
</table>
Positive PMI

The issue
PMI is actually undefined when $X_{ij} = 0$. The usual response is the one given above: set PMI to 0 in such cases. However, this is arguably not coherent (Levy and Goldberg 2014):

- Larger than expected count $\Rightarrow$ large PMI
- Smaller than expected count $\Rightarrow$ small PMI
- 0 count $\Rightarrow$ placed right in the middle!?
Other weighting/normalization schemes

- **t-test:** \[ \frac{P(w,d) - P(w)P(d)}{\sqrt{P(w)P(d)}} \]

- **TF-IDF:** For a corpus of documents \( D \):
  - Term frequency (TF):
    \[ \frac{x_{ij}}{\text{colsum}(X, j)} \]
  - Inverse document frequency (IDF):
    \[ \log_e\left(\frac{|D|}{|\{d \in D : w \in d\}|}\right) \]
    \[ \log_e(0) = 0 \]
  - TF-IDF: \( TF \cdot IDF \)

- **Pairwise distance matrices:**

<table>
<thead>
<tr>
<th></th>
<th>( d_x )</th>
<th>( d_y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>C</td>
<td>14</td>
<td>10</td>
</tr>
</tbody>
</table>

\[ \text{cosine} \Rightarrow \]

<table>
<thead>
<tr>
<th></th>
<th>( A )</th>
<th>( B )</th>
<th>( C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0.008</td>
<td>0.116</td>
</tr>
<tr>
<td>B</td>
<td>0.008</td>
<td>0</td>
<td>0.065</td>
</tr>
<tr>
<td>C</td>
<td>0.116</td>
<td>0.065</td>
<td>0</td>
</tr>
</tbody>
</table>
High-level effects

- Amplify the important, the trustworthy, the unusual; deemphasize the mundane and the quirky.
- Absent a defined objective function, this will remain fuzzy.
- So we should ask of each weighting scheme: How does it compare to the raw count values?
- What overall distribution of values does it deliver?
- We hope to do no feature selection based on counts, stopword dictionaries, etc. Rather, we want our methods to reveal what’s important without these ad hoc interventions.
Weighting scheme cell-value distributions

Uses the giga5 matrix loaded earlier. Others look similar.
Weighting scheme relationships to counts

Uses the giga5 matrix loaded earlier. Others look similar.
Relationships and generalizations

- The theme running through nearly all these schemes is that we want to weight a cell value $X_{ij}$ relative to the value we expect given $X_{i*}$ and $X_{*j}$.

- The magnitude of counts can be important; [1, 10] and [1000, 10000] might represent very different situations; creating probability distributions or length normalizing will obscure this.

- PMI and its variants will amplify the values of counts that are tiny relative to their rows and columns. Unfortunately, with language data, these might be noise.

- TF-IDF severely punishes words that appear in many documents – it behaves oddly for dense matrices, which can include word $\times$ word matrices.
Code snippets

[1]:
```python
import os
import pandas as pd
import vsm
```

[2]:
```python
DATA_HOME = os.path.join('data', 'vsmdata')
```

[3]:
```python
yelp5 = pd.read_csv(
    os.path.join(DATA_HOME, 'yelp_window5-scaled.csv.gz'), index_col=0)
```

[4]:
```python
yelp_oe = vsm.observed_over_expected(yelp5)
```

[5]:
```python
yelp_norm = yelp5.apply(vsm.length_norm, axis=1)
```

[6]:
```python
yelp5_ppmi = vsm.pmi(yelp5)
```

[7]:
```python
yelp5_pmi = vsm.pmi(yelp5, positive=False)
```

[8]:
```python
yelp5_tfidf = vsm.tfidf(yelp5)
```
Code snippets

[9]: `vsm.neighbors('bad', yelp5).head()

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bad</td>
<td>0.000000</td>
</tr>
<tr>
<td>unfortunately</td>
<td>0.116183</td>
</tr>
<tr>
<td>memorable</td>
<td>0.120179</td>
</tr>
<tr>
<td>...</td>
<td>0.122024</td>
</tr>
<tr>
<td>obviously</td>
<td>0.123120</td>
</tr>
<tr>
<td>dtype: float64</td>
<td></td>
</tr>
</tbody>
</table>
```

[10]: `vsm.neighbors('bad', yelp5_ppmi).head()

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bad</td>
<td>0.000000</td>
</tr>
<tr>
<td>terrible</td>
<td>0.471554</td>
</tr>
<tr>
<td>horrible</td>
<td>0.516562</td>
</tr>
<tr>
<td>awful</td>
<td>0.571104</td>
</tr>
<tr>
<td>poor</td>
<td>0.599081</td>
</tr>
<tr>
<td>dtype: float64</td>
<td></td>
</tr>
</tbody>
</table>
```