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

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

and then the probability distribution of u is

$$\left[\frac{u_1}{\textbf{sum}(u)}, \frac{u_2}{\textbf{sum}(u)}, \cdots, \frac{u_n}{\textbf{sum}(u)}\right]$$

Observed/Expected

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

$$\mathbf{expected}(X, i, j) = \frac{\mathbf{rowsum}(X, i) \cdot \mathbf{colsum}(X, j)}{\mathbf{sum}(X)}$$

$$\mathbf{oe}(X, i, j) = \frac{X_{ij}}{\mathbf{expected}(X, i, j)}$$

Observed/Expected

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

$$\textbf{expected}(X,i,j) = \frac{\textbf{rowsum}(X,i) \cdot \textbf{colsum}(X,j)}{\textbf{sum}(X)}$$

$$\textbf{oe}(X,i,j) = \frac{X_{ij}}{\textbf{expected}(X,i,j)}$$

	a	b	**rowsum**
x	34	11	45
y	47	7	54
colsum	81	18	**99**

$\overset{\textbf{oe}}{\Longrightarrow}$

	a	b
x	$\frac{34}{\frac{45 \cdot 81}{99}}$	$\frac{11}{\frac{45 \cdot 18}{99}}$
y	$\frac{47}{\frac{54 \cdot 81}{99}}$	$\frac{7}{\frac{54 \cdot 18}{99}}$

Observed/Expected

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

$$\textbf{expected}(X, i, j) = \frac{\textbf{rowsum}(X, i) \cdot \textbf{colsum}(X, j)}{\textbf{sum}(X)}$$

$$\textbf{oe}(X, i, j) = \frac{X_{ij}}{\textbf{expected}(X, i, j)}$$

Observed

	tabs	reading	birds
keep	20	20	20
enjoy	1	20	20

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

Expected

	tabs	reading	birds
keep	$\frac{60 \cdot 21}{101}$	$\frac{60 \cdot 40}{101}$	$\frac{60 \cdot 40}{101}$
enjoy	$\frac{41 \cdot 21}{101}$	$\frac{41 \cdot 40}{101}$	$\frac{41 \cdot 40}{101}$

$=$

	tabs	reading	birds
keep	12.48	23.76	23.76
enjoy	8.5	16.24	16.24

Pointwise Mutual Information (PMI)

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

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

	d_1	d_2	d_3	d_4
A	10	10	10	10
B	10	10	10	0
C	10	10	0	0
D	0	0	0	1

$\Rightarrow$

	$P(w, d)$				$P(w)$
A	0.11	0.11	0.11	0.11	0.44
B	0.11	0.11	0.11	0.00	0.33
C	0.11	0.11	0.00	0.00	0.22
D	0.00	0.00	0.00	0.01	0.01
$P(d)$	0.33	0.33	0.22	0.12	

PMI

$\Downarrow$

	d_1	d_2	d_3	d_4
A	−0.28	−0.28	0.13	0.73
B	0.01	0.01	0.42	0.00
C	0.42	0.42	0.00	0.00
D	0.00	0.00	0.00	2.11

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 ⇒ large PMI
- Smaller than expected count ⇒ small PMI
- 0 count ⇒ 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):
$$\frac{x_{ij}}{\textbf{colsum}(X, j)}$$
  - Inverse document frequency (IDF):
$$\log_e\left(\frac{|D|}{\left|\{d \in D : w \in d\}\right|}\right) \qquad \log_e(0) = 0$$
  - TF-IDF: TF · IDF
- Pairwise distance matrices:

	d_x	d_y
A	2	4
B	10	15
C	14	10

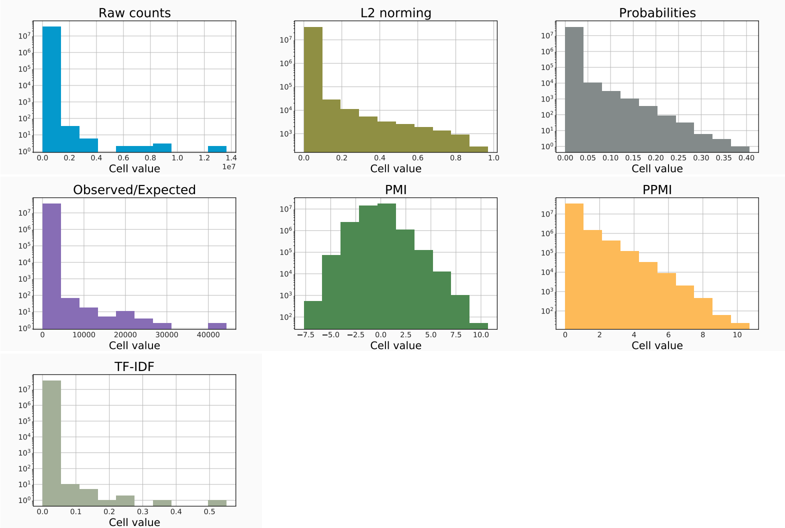
$\overset{\text{cosine}}{\Rightarrow}$

	A	B	C
A	0	0.008	0.116
B	0.008	0	0.065
C	0.116	0.065	0

High-level effects

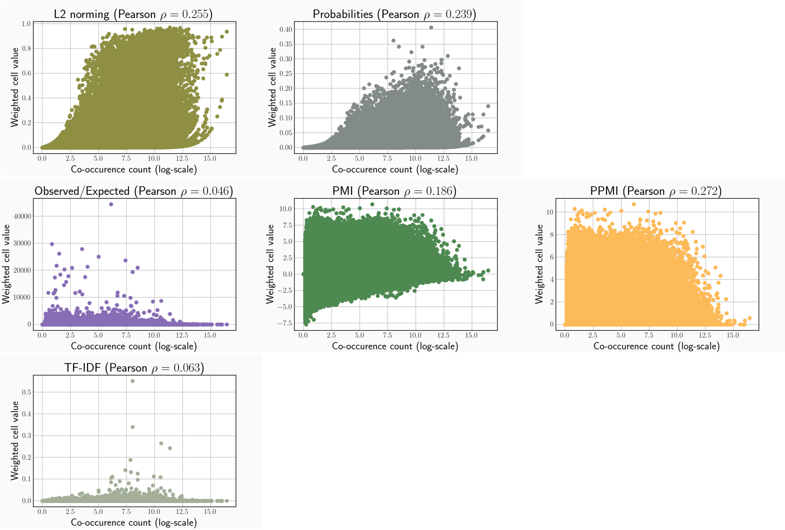
- Amplify the important, the trustworthy, the unusual; deemphasize the mundane and the quirky.
- Absent a defined objective function, this will remain fuzzy.
- So we should ask of each weighting scheme: How does it compare to the raw count values?
- What overall distribution of values does it deliver?
- 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 noise.
- TF-IDF severely punishes words that appear in many documents – it behaves oddly for dense matrices, which can include word × word matrices.

Code snippets

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

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

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

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

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

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

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

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

Code snippets

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

[9]: bad              0.000000
     unfortunately    0.116183
     memorable        0.120179
     …              0.122024
     obviously        0.123120
     dtype: float64

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

[10]: bad         0.000000
      terrible    0.471554
      horrible    0.516562
      awful       0.571104
      poor        0.599081
      dtype: float64
```

References I

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*.