Distributional Semantics

Class-based N-gram Language Models and Word Embeddings

Chase Geigle
2017-09-28
What is a word?

“dog”
What is a word?

“dog”    “canine”
<table>
<thead>
<tr>
<th></th>
<th>“dog”</th>
<th>“canine”</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sparsity!
What is a word?

<table>
<thead>
<tr>
<th>“dog”</th>
<th>“canine”</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>399,999</td>
</tr>
</tbody>
</table>
What is a word?

“dog”

3

0

0

0

0

“canine”

399,999

Sparsity!
**What is a word?**

<table>
<thead>
<tr>
<th>“dog”</th>
<th>“canine”</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>399,999</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
What is a word?

“dog”

3

\[
\begin{pmatrix}
0 \\
0 \\
0 \\
\vdots \\
0
\end{pmatrix}
\]

“canine”

399,999

\[
\begin{pmatrix}
0 \\
0 \\
0 \\
\vdots \\
1
\end{pmatrix}
\]

Sparsity!
In the real world, **sparsity** hurts us:

- Low vocabulary coverage in training $\rightarrow$ high OOV rate in applications (poor generalization)
- “one-hot” representations $\rightarrow$ huge parameter counts
- Information about one word *completely unutilized* for another!

Thus, a good representation must:

- reduce parameter space,
- improve generalization, and
- somehow “transfer” or “share” knowledge between words
The Distributional Hypothesis

You shall know a word by the company it keeps.
(J. R. Firth, 1957)

Words with **high similarity** occur in the **same contexts** as one another.
The Distributional Hypothesis

You shall know a word by the company it keeps.
(J. R. Firth, 1957)

Words with high similarity occur in the same contexts as one another.

A bit of foreshadowing:

- A word ought to be able to predict its context (word2vec Skip-Gram)
- A context ought to be able to predict its missing word (word2vec CBOW)
Brown Clustering
A language model is a probability distribution over word sequences.

- **Unigram language model** \( p(w|\theta) \):
  \[
  p(W|\theta) = N \prod_{i=1}^{N} p(w_i|\theta)
  \]

- **Bigram language model** \( p(w_i|w_{i-1}, \theta) \):
  \[
  p(W|\theta) = N \prod_{i=1}^{N} p(w_i|w_{i-1}, \theta)
  \]

- **n-gram language model**:
  \[
  p(w_n|w_{n-1}, \theta) \cdot p(W|\theta) = N \prod_{i=1}^{N} p(w_i|w_{i-n+1}, \theta)
  \]
A language model is a \textit{probability distribution} over
A language model is a probability distribution over word sequences.
A language model is a **probability distribution** over word sequences.

- Unigram language model $p(w \mid \theta)$:

\[
p(W \mid \theta) = \prod_{i=1}^{N} p(w_i \mid \theta)
\]
A language model is a **probability distribution** over **word sequences**.

- **Unigram language model** $p(w_i | \theta)$:
  \[
p(W | \theta) = \prod_{i=1}^{N} p(w_i | \theta)
  \]

- **Bigram language model** $p(w_i | w_{i-1}, \theta)$:
  \[
p(W | \theta) = \prod_{i=1}^{N} p(w_i | w_{i-1}, \theta)
  \]
A language model is a **probability distribution over word sequences**.

- **Unigram language model** $p(w \mid \theta)$:
  \[
p(W \mid \theta) = \prod_{i=1}^{N} p(w_i \mid \theta)
  \]

- **Bigram language model** $p(w_i \mid w_{i-1}, \theta)$:
  \[
p(W \mid \theta) = \prod_{i=1}^{N} p(w_i \mid w_{i-1}, \theta)
  \]

- **n-gram language model**: $p(w_n \mid w_{1}^{n-1}, \theta)$
  \[
p(W \mid \theta) = \prod_{i=1}^{N} p(w_i \mid w_{i-n+1}^{i-1}, \theta)
  \]
A Class-Based Language Model

Main Idea: cluster words into a fixed number of clusters $C$ and use their cluster assignments as their identity instead (reducing sparsity)

If $\pi : V \rightarrow C$ is a mapping function from a word type to a cluster (or class), we want to find

$$\pi^* = \arg \max \ p(W \mid \pi)$$

where

$$p(W \mid \pi) = \prod_{i=1}^{N} p(c_{i} \mid c_{i-1})p(w_{i} \mid c_{i})$$

with $c_{i} = \pi(w_{i})$.

---

Finding the best partition

\[ \pi^* = \arg \max_{\pi} P(W \mid \pi) = \arg \max_{\pi} \log P(W \mid \pi) \]

One can derive \(^2\) \(L(\pi)\) to be

\[
L(\pi) = \sum_{w} n_w \log n_w + \sum_{c_i, c_j} n_{c_i, c_j} \log \frac{n_{c_i, c_j}}{n_{c_i} \cdot n_{c_j}}
\]

(nearly) unigram entropy
(fixed w.r.t. \(\pi\))

(nearly) mutual information
(varies with \(\pi\))

---

Finding the best partition

\[ L(\pi) = \sum_{w} n_w \log n_w + \sum_{c_i, c_j} n_{c_i, c_j} \log \frac{n_{c_i, c_j}}{n_{c_i} \cdot n_{c_j}} \]

- (nearly) unigram entropy (fixed w.r.t. \( \pi \))
- (nearly) mutual information (varies with \( \pi \))

What does maximizing this mean?

Recall

\[ MI(c, c') = \sum_{c_i, c_j} p(c_i, c_j) \log \frac{p(c_i, c_j)}{p(c_i)p(c_j)} \]

which can be shown to be

\[ MI(c, c') = \frac{1}{N} \sum_{c_i, c_j} n_{c_i, c_j} \log \frac{n_{c_i, c_j}}{n_{c_i} \cdot n_{c_j}} + \log N \]

(constant)

and thus **maximizing MI of adjacent classes** selects the best \( \pi \).
Finding the best partition (cont’d)

\[ \pi^* = \arg \max_{\pi} \left( \sum_w n_w \log n_w \right) + \sum_{c_i, c_j} n_{c_i, c_j} \log \frac{n_{c_i, c_j}}{n_{c_i} \cdot n_{c_j}} \]

(nearly) unigram entropy
(fixed w.r.t. \( \pi \))

(nearly) mutual information
(varies with \( \pi \))

Direct maximization is **intractable**! Thus, agglomerative (bottom-up) clustering is used as a greedy heuristic.

The best merge is determined by the lowest loss in average mutual information.
Agglomerative Clustering

$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5 \quad w_6 \quad w_7 \quad w_8 \quad w_9 \quad w_{10} \quad w_{11}$
Agglomerative Clustering

\[ w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5 \quad w_6 \quad w_7 \quad w_8 \quad w_9 \quad w_{10} \quad w_{11} \]
Agglomerative Clustering
Agglomerative Clustering
Agglomerative Clustering
Agglomerative Clustering
Agglomerative Clustering
Agglomerative Clustering
Agglomerative Clustering
Agglomerative Clustering
Agglomerative Clustering
Do they work? (Yes.)

Named entity recognition:

- Scott Miller, Jethran Guinness, and Alex Zamanian. “Name Tagging with Word Clusters and Discriminative Training.”. In: *HLT-NAACL*. 2004, pp. 337–342

Dependency parsing:


Constituency parsing:

Vector Spaces for Word Representation
Brown clusters are nice, but limiting:

• Arbitrary cut-off point → two words may be assigned different classes arbitrarily because of our chosen cutoff

• Definition of “similarity” is limited to only bigrams of classes\(^3\)

• Completely misses lots of **regularity** that’s present in language

\(^3\)There are adaptations of Brown clustering that move past this, but what we’ll see today is still even better.
What do I mean by “regularity”? 

woman is to sister as man is to
What do I mean by “regularity”? 

woman is to sister as man is to brother
What do I mean by “regularity”?

woman is to sister as man is to brother

summer is to rain as winter is to...
What do I mean by “regularity”?  

woman is to sister as man is to brother  
summer is to rain as winter is to snow
What do I mean by “regularity”?

woman is to sister as man is to brother
summer is to rain as winter is to snow
man is to king as woman is to
What do I mean by “regularity”?

woman is to sister as man is to brother
summer is to rain as winter is to snow
man is to king as woman is to queen
What do I mean by “regularity”?

woman is to sister as man is to brother
summer is to rain as winter is to snow
man is to king as woman is to queen
fell is to fallen as ate is to
What do I mean by “regularity”?

woman is to sister as man is to brother
summer is to rain as winter is to snow
man is to king as woman is to queen
fell is to fallen as ate is to eaten

The differences between each pair of words are similar. Can our word representations capture this? (demo)
What do I mean by “regularity”?

- woman is to sister as man is to brother
- summer is to rain as winter is to snow
- man is to king as woman is to queen
- fell is to fallen as ate is to eaten
- running is to ran as crying is to
What do I mean by “regularity”?

woman   is to    sister    as    man    is to    brother
summer   is to    rain     as    winter  is to    snow
man      is to    king     as    woman  is to    queen
fell     is to    fallen   as    ate    is to    eaten
running is to    ran      as    crying is to    cried
What do I mean by “regularity”?

woman is to sister as man is to brother
summer is to rain as winter is to snow
man is to king as woman is to queen
fell is to fallen as ate is to eaten
running is to ran as crying is to cried

The differences between each pair of words are similar.

Can our word representations capture this? (demo)
Neural Word Embeddings
Associate a low-dimensional, dense vector \( \vec{w} \) with each word \( w \in V \) so that similar words (in a distributional sense) share a similar vector representation.
To solve analogy problems of the form “$w_a$ is to $w_b$ as $w_c$ is to what?”, we can simply compute a query vector

$$q = w_b - w_a + w_c$$

and find the most similar word vector $v \in W$ to $q$. If we normalize $q$ to unit-length

$$\hat{q} = \frac{q}{||q||}$$

and assume each vector in $W$ is also unit-length, this reduces to computing

$$\arg \max_{v \in W} v \cdot \hat{q}$$

and returning the associated word $v$. 
The CBOW and Skip-Gram Models\textsuperscript{4} (word2vec)

\textsuperscript{4}Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: ICLR Workshop (2013).
Skip-Gram, Mathematically

Training corpus \( w_1, w_2, \ldots, w_N \) (with \( N \) typically in the billions) from a fixed vocabulary \( V \).

Goal is to maximize the average log probability:

\[
\frac{1}{N} \sum_{i=1}^{N} \sum_{-L \leq k \leq L; k \neq 0} \log p(w_{i+k} \mid w_i)
\]

Associate with each \( w \in V \) an "input vector" \( w \in \mathbb{R}^d \) and an "output vector" \( \tilde{w} \in \mathbb{R}^d \). Model context probabilities as

\[
p(c \mid w) = \frac{\exp(w \cdot \tilde{c})}{\sum_{c' \in V} \exp(w \cdot \tilde{c}')}
\]

The problem? \( V \) is huge! \( \nabla \log p(c \mid w) \) takes time \( O(|V|) \) to compute!
Given a pair \((w, c)\), can we determine if this came from our corpus or not? Model probabilistically as

\[
p(D = 1 \mid w, c) = \sigma(w \cdot \tilde{c}) = \frac{1}{1 + \exp(-w \cdot \tilde{c})}.
\]

Goal: maximize \(p(D = 1 \mid w, c)\) for pairs \((w, c)\) that \textbf{occur in the data}.

Also maximize \(p(D = 0 \mid w, c_N)\) for \((w, c_N)\) pairs where \(c_N\) is \textbf{sampled randomly} from the empirical unigram distribution.

---

Locally,

\[ \ell = \log \sigma(w \cdot \tilde{c}) + k \cdot \mathbb{E}_{c_N \sim p_D(c_N)} [\log \sigma(-w \cdot c_N)] \]

and thus globally

\[ \mathcal{L} = \sum_{w \in V} \sum_{c \in V} (n_{w,c}) \left( \log \sigma(w \cdot \tilde{c}) + k \cdot \mathbb{E}_{c_N \sim p_n(c_N)} [\log \sigma(-w \cdot c_N)] \right) \]

- \( k \): number of negative samples to take (hyperparameter)
- \( n_{w,c} \): number of times \((w, c)\) was seen in the data
- \( \mathbb{E}_{c_N \sim p_n(c_N)} \) indicates an expectation taken with respect to the noise distribution \( P_n(c_N) \).
1. What is the noise distribution?

\[ P_n(c_N) = \frac{(n_{c_N}/N)^{3/4}}{Z} \]

(which is the empirical unigram distribution raised to the 3/4 power).

2. Frequent words can dominate the loss. Throw away word \( w_i \) in the training data according to

\[ P(w_i) = 1 - \sqrt{\frac{t}{n_{w_i}}} \]

where \( t \) is some threshold (like \( 10^{-5} \)).

Both are essentially unexplained by Mikolov et al.
What does this actually do?

It turns out this is nothing that new! Levy and Goldberg\(^6\) show that SGNS is **implicitly factorizing** the matrix

\[
M_{i,j} = \mathbf{w}_i \cdot \tilde{c}_j = \text{PMI}(w_i, c_j) - \log k = \log \frac{p(w_i, c_j)}{p(w_i)p(c_j)} - \log k
\]

using an objective that weighs deviations in more frequent \((w, c)\) pairs more strongly than less frequent ones.

Thus...

---

Matrix Factorization Methods for Word Embeddings
Why not just SVD?

Since SGNS is just factorizing a shifted version of the PMI matrix, why not just perform a rank-\(k\) SVD on the PMI matrix directly?

MP2!

With a little TLC, this method can actually work and does surprisingly well.
How can we capture words related to **ice**, but not to **steam**?

<table>
<thead>
<tr>
<th>Prob. or Ratio</th>
<th>( w_k = \text{solid} )</th>
<th>( w_k = \text{gas} )</th>
<th>( w_k = \text{water} )</th>
<th>( w_k = \text{fashion} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(w_k</td>
<td>\text{ice}) )</td>
<td>( 1.9 \times 10^{-4} )</td>
<td>( 6.6 \times 10^{-5} )</td>
<td>( 3.0 \times 10^{-3} )</td>
</tr>
<tr>
<td>( P(w_k</td>
<td>\text{steam}) )</td>
<td>( 2.2 \times 10^{-5} )</td>
<td>( 7.8 \times 10^{-4} )</td>
<td>( 2.2 \times 10^{-3} )</td>
</tr>
<tr>
<td>( \frac{P(w_k</td>
<td>\text{ice})}{P(w_k</td>
<td>\text{steam})} )</td>
<td>( 8.9 )</td>
<td>( 8.5 \times 10^{-2} )</td>
</tr>
</tbody>
</table>

Probability ratios are most informative:

- **solid** is related to **ice** but not **steam**
- **gas** is related to **steam** but not **ice**
- **water** and **fashion** do not discriminate between **ice** or **steam** (ratios close to 1)

---

We would like for the vectors $\mathbf{w}_i$, $\mathbf{w}_j$, and $\mathbf{w}_k$ to be able to capture the information present in the probability ratio. If we set

$$\mathbf{w}_i^\top \mathbf{w}_k = \log P(w_k | w_i)$$

then we can easily see that

$$(\mathbf{w}_i - \mathbf{w}_j)^\top \mathbf{w}_k = \log P(w_k | w_i) - \log P(w_k | w_j) = \log \frac{P(w_k | w_i)}{P(w_k | w_j)},$$

ensuring that the information present in the co-occurrence probability ratio is expressed in the vector space.
GloVe Objective

\[
  \mathbf{w}_i^T \tilde{\mathbf{w}}_j + b_i + \tilde{b}_j = \log X_{ij}
\]

Again: two sets of vectors: “target” vectors \( \mathbf{w} \) and “context” vectors \( \tilde{\mathbf{w}} \).

\( X_{ij} = n_{w_i,w_j} \) is the number of times \( w_j \) appears in the context of \( w_i \).

**Problem:** all co-occurrences are weighted equally

\[
  J = \sum_{i,j=1}^{V} f(X_{ij})(\mathbf{w}_i^T \tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - \log X_{ij})^2
\]

**Final objective** is just weighted least-squares. \( f(X_{ij}) \) serves as a “dampener”, lessening the weight of the rare co-occurrences.

\[
  f(x) = \begin{cases} 
    \left( \frac{x}{x_{\max}} \right)^{\alpha} & \text{if } x < x_{\max} \\
    1 & \text{otherwise}
  \end{cases}
\]

where \( \alpha \)'s default is (you guessed it) 3/4, and \( x_{\max} \)'s default is 100.
Relation to word2vec

GloVe Objective:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

word2vec (Skip-Gram) Objective (after rewriting):

$$J = -\sum_{i=1}^{V} X_i \sum_{j=1}^{V} P(w_j | w_i) \log Q(w_j | w_i)$$

where $X_i = \sum_k X_{ik}$ and $P$ and $Q$ are the empirical co-occurrence and model co-occurrence distributions, respectively. **Weighted cross-entropy error!**

Authors show that replacing cross-entropy with least-squares nearly re-derives GloVe:

$$\hat{J} = \sum_{i,j} X_i (w_i^T \tilde{w}_j - \log X_{ij})^2$$
Model Complexity

word2vec: $O(|C|)$, linear in corpus size

GloVe naïve estimate: $O(|V|^2)$, square of vocab size

But it actually only depends on the number of non-zero entries in $X$. If co-occurrences are modeled via a power law, then we have

$$X_{ij} = \frac{k}{(r_{ij})^\alpha}.$$ 

Modeling $|C|$ under this assumption, the authors eventually arrive at

$$|X| = \begin{cases} 
O(|C|) & \text{if } \alpha < 1 \\
O(|C|^{1/\alpha}) & \text{otherwise}
\end{cases}$$

where the corpora studied in the paper were well modeled with $\alpha = 1.25$, leading to $O(|C|^{0.8})$. In practice, it’s faster than word2vec.
Accuracy vs Window Size

Left: Symmetric window, Right: Asymmetric window
Accuracy vs Corpus Choice

- Wiki2010 (1B tokens)
- Wiki2014 (1.6B tokens)
- Gigaword5 (4.3B tokens)
- Gigaword5 + Wiki2014 (6B tokens)
- Common Crawl (42B tokens)

Accuracy [%]

- Semantic
- Syntactic
- Overall
This evaluation is not quite fair!
But wait…

(Referring to word2vec)

...the code is designed only for a single training epoch...
...specifies a learning schedule specific to one pass through
the data, making a modification for multiple passes a
non-trivial task.
But wait...

(Referring to word2vec)

...the code is designed only for a single training epoch...
...specifies a learning schedule specific to one pass through the data, making a modification for multiple passes a non-trivial task.

This is **false**, and a **bad excuse**.

How could you modify word2vec to support multiple “epochs” with zero effort?
...we choose to use the sum \( \mathbf{W} + \tilde{\mathbf{W}} \) as our word vectors.
...we choose to use the sum $\mathbf{W} + \tilde{\mathbf{W}}$ as our word vectors.

word2vec also learns $\tilde{\mathbf{W}}$! Why not do the same for both?
Embedding performance is very much a function of your hyperparameters!
Levy, Goldberg, and Dagan perform a systematic evaluation of word2vec, SVD, and GloVe where hyperparameters are controlled for and optimized across different methods for a variety of tasks.

Interesting results:

*MSR’s analogy dataset is the only case where SGNS and GloVe substantially outperform PPMI and SVD.*

And:

*...SGNS outperforms GloVe in every task.*

---

Both GloVe and word2vec use **context window weighting**:

- **word2vec**: samples a window size $\ell \in [1, L]$ for each token before extracting counts
- **GloVe**: weighs contexts by their **distance from the target word** using the harmonic function $\frac{1}{d}$

What is the impact of using the different context window strategies between methods?

Levy, Goldberg, and Dagan **did not** investigate this, instead using word2vec’s method across all of their evaluations.
Evaluating things properly is non-obvious and often very difficult!
What You Should Know
What You Should Know

• What is the sparsity problem in NLP?
• What is the distributional hypothesis?
• How does Brown clustering work? What is it trying to maximize?
• What is agglomerative clustering?
• What is a word embedding?
• What is the difference between the CBOW and Skip-Gram models?
• What is negative sampling and why is it useful?
• What is the learning objective for SGNS?
• What is the matrix that SGNS is implicitly factorizing?
• What is the learning objective for GloVe?
• What are some examples of hyperparameters for word embedding methods?
word2vec Implementations (non-exhaustive)

1. The **original implementation**:  
   https://code.google.com/archive/p/word2vec/

2. Yoav Goldberg’s word2vecf **modification (multiple epochs, arbitrary context features)**:  
   https://bitbucket.org/yoavgo/word2vecf


4. **Python implementation** in gensim:  
   https://radimrehurek.com/gensim/models/word2vec.html
GloVe Implementations (non-exhaustive)

1. The **original implementation**:  
   http://nlp.stanford.edu/projects/glove/

2. **MeTA**: https://meta-toolkit.org

3. **R implementation** text2vec: http://text2vec.org/
Thanks!