Word representations

Édouard Grave, Armand Joulin

Facebook AI Research
egrave@fb.com

Introduction

- Traditional way to represent words as **atomic symbols** with a unique integer is associated with each word:

$$\{1\text{=movie, }2\text{=hotel, }3\text{=apple, }4\text{=movies, }5\text{=art}\}$$

- Equivalent to represent words as **1-hot vectors**:

$$\begin{aligned}
\text{movie} &= [1, 0, 0, 0, 0] \\
\text{hotel} &= [0, 1, 0, 0, 0] \\
\ldots & \\
\text{art} &= [0, 0, 0, 0, 1]
\end{aligned}$$

Introduction

- Implicit assumption: word vectors are an orthonormal basis
  - orthogonal ($x^T y = 0$)
  - normalized ($x^T x = 1$)

- Problem: Not very informative:
  - Weird to consider "movie" and "movies" as independent entities
  - Or to consider all words equidistant:

$$\|\texttt{dog} - \texttt{cat}\| = \|\texttt{dog} - \texttt{moon}\|$$

Introduction

- Reminder:
  - Word types are element of the vocabulary
  - Word tokens are instances of word types in text
- Here, we want representations for word types

Feature based representation

- Solution: represent words with hand crafted features and relations
- Example of potential features:
  - Morphology: prefix, suffix, stem...
  - Grammar: part of speech, gender, number,...
  - Shape: capitalization, digit, hyphen
- Example of potential relations:
  - synonyms,
  - hypernyms,
  - antonyms...

Limitations of feature based representation

- Requires (a lot of) human annotations
- Subjectivity of the annotators
- does not adapt to new words (languages are not stationary!):

  Mocktail, Guac, Fave, Biohacking

  were added to Merriam-Webster in 2018
- Existing online taxonomy like WordNet are not always very precise:
  - "Good" synonyms: skillful, practiced, proficient, adept

Distributional hypothesis

"You shall know a word by the company it keeps" Firth (1957)

- Meaning of a word: set of contexts in which it occurs in texts

Example: What is the meaning of "**bardiwac**"?

- He handed her her glass of bardiwac.

Source: Distributional Semantic Models, Stefan Evert, 2015.

Example: What is the meaning of "**bardiwac**"?

- He handed her her glass of bardiwac.
- Beef dishes are made to complement the bardiwacs.

Source: Distributional Semantic Models, Stefan Evert, 2015.

Example: What is the meaning of "**bardiwac**"?

- He handed her her glass of bardiwac.
- Beef dishes are made to complement the bardiwacs.
- Nigel staggered to his feet, face flushed from too much bardiwac.

Source: Distributional Semantic Models, Stefan Evert, 2015.

Example: What is the meaning of "**bardiwac**"?

- He handed her her glass of bardiwac.
- Beef dishes are made to complement the bardiwacs.
- Nigel staggered to his feet, face flushed from too much bardiwac.
- Malbec, one of the lesser-known bardiwac grapes, responds well to Australia's sunshine.

Source: Distributional Semantic Models, Stefan Evert, 2015.

Example: What is the meaning of "**bardiwac**"?

- He handed her her glass of bardiwac.
- Beef dishes are made to complement the bardiwacs.
- Nigel staggered to his feet, face flushed from too much bardiwac.
- Malbec, one of the lesser-known bardiwac grapes, responds well to Australia's sunshine.
- I dined off bread and cheese and this excellent bardiwac.

Source: Distributional Semantic Models, Stefan Evert, 2015.

Example: What is the meaning of "**bardiwac**"?

- He handed her her glass of bardiwac.
- Beef dishes are made to complement the bardiwacs.
- Nigel staggered to his feet, face flushed from too much bardiwac.
- Malbec, one of the lesser-known bardiwac grapes, responds well to Australia's sunshine.
- I dined off bread and cheese and this excellent bardiwac.
- The drinks were delicious: blood-red bardiwac as well as light, sweet Rhenish.

Source: Distributional Semantic Models, Stefan Evert, 2015.

Example: What is the meaning of "**bardiwac**"?

- He handed her her glass of bardiwac.
- Beef dishes are made to complement the bardiwacs.
- Nigel staggered to his feet, face flushed from too much bardiwac.
- Malbec, one of the lesser-known bardiwac grapes, responds well to Australia's sunshine.
- I dined off bread and cheese and this excellent bardiwac.
- The drinks were delicious: blood-red bardiwac as well as light, sweet Rhenish.

$\rightarrow$ bardiwac is a heavy red alcoholic beverage made from grapes

---

Source: Distributional Semantic Models, Stefan Evert, 2015.

Distributional word representation in a nutshell

- Define what is the context of a word
- Count how many times each target word occurs in this context
- Build vectors out of (a function of) these context occurrence counts

Source: Foundations of Distributional Semantic Models, Stefan Evert and Alessandro Lenci, 2009.

Distributional word representation in a nutshell

- Define what is the context of a word
- Count how many times each target word occurs in this context
- Build vectors out of (a function of) these context occurrence counts

Caveat:

- Similar vectors represent words with similar distributions in contexts

Source: Foundations of Distributional Semantic Models, Stefan Evert and Alessandro Lenci, 2009.

Distributional word representation in a nutshell

- Define what is the context of a word
- Count how many times each target word occurs in this context
- Build vectors out of (a function of) these context occurrence counts

Caveat:

- Similar vectors represent words with similar distributions in contexts
- Distributional hypothesis: bridging assumption from distributional representation to semantic representation

Source: Foundations of Distributional Semantic Models, Stefan Evert and Alessandro Lenci, 2009.

Distributional word representation in a nutshell

- Define what is the context of a word
- Count how many times each target word occurs in a certain context
- Build vectors out of (a function of) these context occurrence counts

Source: Foundations of Distributional Semantic Models, Stefan Evert and Alessandro Lenci, 2009.

What is the “context”?

The silhouette of the sun beyond a wide-open bay on the lake; the sun still glitters although evening has arrived in Kuhmo. It’s midsummer; the living room has its instruments and other objects in each of its corners

Source: Distributional Semantics, Marco Baroni and Gemma Boleda. 2017.

What is the “context”?

The whole document

The silhouette of the sun beyond a wide-open bay on the lake; the sun still glitters although evening has arrived in Kuhmo. It’s midsummer; the living room has its instruments and other objects in each of its corners

Source: Distributional Semantics, Marco Baroni and Gemma Boleda. 2017.

What is the “context”?

A window of surrounding words

The silhouette of the sun beyond a wide-open bay on the lake; the sun still glitters although evening has arrived in Kuhmo. It’s midsummer; the living room has its instruments and other objects in each of its corners

Source: Distributional Semantics, Marco Baroni and Gemma Boleda. 2017.

What is the “context”?

A window of surrounding words after preprocessing

The silhouette of the sun beyond a wide-open bay on the lake; the sun still glitters although evening has arrived in Kuhmo. It's midsummer; the living room has its instruments and other objects in each of its corners

Source: Distributional Semantics, Marco Baroni and Gemma Boleda. 2017.

Distributional word representation in a nutshell

- Define what is the context of a word
- Count how many times each target word occurs in a certain context
- Build vectors out of (a function of) these context occurrence counts

Source: Foundations of Distributional Semantic Models, Stefan Evert and Alessandro Lenci, 2009.

Collecting context counts for target word **dog**

The dog barked in the park.
The owner of the dog put him
on the leash since he barked.

barked	++
park	+
owner	+
leash	+

Source: Foundations of Distributional Semantic Models, Stefan Evert and Alessandro Lenci, 2009.

Collecting context counts for target word **dog**

The dog barked in the park.
The owner of the dog put him
on the leash since he barked.

barked	++
park	+
owner	+
leash	+

Source: Foundations of Distributional Semantic Models, Stefan Evert and Alessandro Lenci, 2009.

Collecting context counts for target word **dog**

The dog barked in the park.
The owner of the dog put him
on the leash since he barked.

barked	++
park	+
owner	+
leash	+

Source: Foundations of Distributional Semantic Models, Stefan Evert and Alessandro Lenci, 2009.

Collecting context counts for target word **dog**

The dog barked in the park.
The owner of the dog put him
on the leash since he barked.

barked	++
park	+
owner	+
leash	+

Source: Foundations of Distributional Semantic Models, Stefan Evert and Alessandro Lenci, 2009.

Collecting context counts for target word **dog**

The dog barked in the park.
The owner of the dog put him
on the leash since he barked.

barked	++
park	+
owner	+
leash	+

Source: Foundations of Distributional Semantic Models, Stefan Evert and Alessandro Lenci, 2009.

Collecting context counts for target word **dog**

The dog barked in the park.
The owner of the dog put him
on the leash since he barked.

barked	++
park	+
owner	+
leash	+

Source: Foundations of Distributional Semantic Models, Stefan Evert and Alessandro Lenci, 2009.

The co-occurrence matrix

	dog	barked	leash	walk	run	owner	pet	...
dog	0	2	3	5	2	5	3	...
barked	2	0	1	0	0	2	1	...
leash	3	1	0	1	0	2	0	...
...	...	...	...	...	...	...	...	...
cat	2	0	0	3	3	2	3	...
lion	1	0	0	3	2	0	1	...
light	0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...

Source: Foundations of Distributional Semantic Models, Stefan Evert and Alessandro Lenci, 2009.

Distributional word representation in a nutshell

- Define what is the context of a word
- Count how many times each target word occurs in a certain context
- Build vectors out of (a function of) these context occurrence counts

Source: Foundations of Distributional Semantic Models, Stefan Evert and Alessandro Lenci, 2009.

Word vectors from context occurence counts

- Goal: Build word vectors from occurence count with their context
- We focus on context as a fixed size window around the word
- Distance between vectors should reflect "similarity" between words
- We use the cosine similarity:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

Word vectors from context occurence counts

	⋯	leash	walk	run	owner	pet	barked	the
dog	⋯	3	5	2	5	3	2	8
cat	⋯	0	3	3	2	3	0	9
lion	⋯	0	3	2	0	1	0	6
light	⋯	0	0	0	0	0	0	5
bark	⋯	1	0	0	2	1	0	0
car	⋯	0	0	1	3	0	0	3

- Naive approach: takes the row of the co-occurence matrix $\mathbf{M}$

Word vectors from context occurence counts

	⋯	leash	walk	run	owner	pet	barked	the
dog	⋯	3	5	2	5	3	2	8
cat	⋯	0	3	3	2	3	0	9
lion	⋯	0	3	2	0	1	0	6
light	⋯	0	0	0	0	0	0	5
bark	⋯	1	0	0	2	1	0	0
car	⋯	0	0	1	3	0	0	3

- Naive approach: takes the row of the co-occurence matrix $\mathbf{M}$

Word vectors from context occurence counts

	⋯	leash	walk	run	owner	pet	barked	the
dog	⋯	3	5	2	5	3	2	8
cat	⋯	0	3	3	2	3	0	9
lion	⋯	0	3	2	0	1	0	6
light	⋯	0	0	0	0	0	0	5
bark	⋯	1	0	0	2	1	0	0
car	⋯	0	0	1	3	0	0	3

- Naive approach: takes the row of the co-occurence matrix $\mathbf{M}$

Word vectors from context occurence counts

	⋯	leash	walk	run	owner	pet	barked	the
dog	⋯	3	5	2	5	3	2	8
cat	⋯	0	3	3	2	3	0	9
lion	⋯	0	3	2	0	1	0	6
light	⋯	0	0	0	0	0	0	5
bark	⋯	1	0	0	2	1	0	0
car	⋯	0	0	1	3	0	0	3

- Naive approach: takes the row of the co-occurence matrix $\mathbf{M}$

Word vectors from context occurence counts

Problems with using the co-occurence matrix **M** directly:

- Co-occurence matrix norm is proportional to corpus size
- Entries associated with frequent words dominate the matrix
- Sensitive to small changes in counts of rare words

Pointwise Mutual Information Matrix

- An alternative *context weighting* is the Mutual Information (MI):

$$MI(i,j) = \log p(i,j) - \log p(i) - \log p(j)$$

- In our case $p(i,j) = \mathbf{M}_{i,j}/n$ and $p(i) = \sum_{ij} \mathbf{M}_{ij}/n$
- The resulting matrix is called the Pointwise Mutual Information (PMI) matrix.

Pointwise Mutual Information Matrix

- Co-occurence matrix norm is proportional to corpus size

Pointwise Mutual Information Matrix

- Co-occurence matrix norm is proportional to corpus size

$\rightarrow$ divide entries by it:

$$\mathbf{P}_{ij} = \frac{1}{n}\mathbf{M}_{ij}$$

Pointwise Mutual Information Matrix

- Co-occurence matrix norm is proportional to corpus size

$\rightarrow$ divide entries by it:

$$\mathbf{P}_{ij} = \frac{1}{n}\mathbf{M}_{ij}$$

- Entries associated with frequent words dominate the matrix

Pointwise Mutual Information Matrix

- Co-occurence matrix norm is proportional to corpus size

$\rightarrow$ divide entries by it:

$$\mathbf{P}_{ij} = \frac{1}{n}\mathbf{M}_{ij}$$

- Entries associated with frequent words dominate the matrix

$\rightarrow$ Normalized vector by word counts:

$$\mathbf{Q}_{ij} = \frac{\mathbf{P}_{ij}}{\mathbf{P}_j \mathbf{P}_i} \quad \text{where} \quad \mathbf{P}_i = \sum_j \mathbf{P}_{ij} = \sum_j \mathbf{P}_{ji}$$

Pointwise Mutual Information Matrix

- Co-occurence matrix norm is proportional to corpus size

$\rightarrow$ divide entries by it:

$$\mathbf{P}_{ij} = \frac{1}{n}\mathbf{M}_{ij}$$

- Entries associated with frequent words dominate the matrix

$\rightarrow$ Normalized vector by word counts:

$$\mathbf{Q}_{ij} = \frac{\mathbf{P}_{ij}}{\mathbf{P}_j \mathbf{P}_i} \quad \text{where} \quad \mathbf{P}_i = \sum_j \mathbf{P}_{ij} = \sum_j \mathbf{P}_{ji}$$

- Sensitive to small changes in counts of rare words

Pointwise Mutual Information Matrix

- Co-occurence matrix norm is proportional to corpus size

$\rightarrow$ divide entries by it:

$$\mathbf{P}_{ij} = \frac{1}{n}\mathbf{M}_{ij}$$

- Entries associated with frequent words dominate the matrix

$\rightarrow$ Normalized vector by word counts:

$$\mathbf{Q}_{ij} = \frac{\mathbf{P}_{ij}}{\mathbf{P}_j \mathbf{P}_i} \quad \text{where} \quad \mathbf{P}_i = \sum_j \mathbf{P}_{ij} = \sum_j \mathbf{P}_{ji}$$

- Sensitive to small changes in counts of rare words

$\rightarrow$ take the log to smooth high frequencies:

$$\mathbf{R}_{ij} = \log \mathbf{Q}_{ij}$$

This is the PMI matrix!

Limitations of PMI

- Word pairs with $p(a,b) < p(a)p(b)$ lead to instability in MI

Example

- with context size = 3: $p(''a'',''the'') \ll p(''a'')p(''the'')$
- $p(''a'') = 0.1$, $p(''the'') = 0.2$
- $p(''a'',''the'') = 10^{-5} \rightarrow MI(''a'',''the'') = -7.6$
- $p(''a'',''the'') = 10^{-9} \rightarrow MI(''a'',''the'') = -16.8$

- Small error in estimation of rare events are blown out by log
- Impacts the similarities between words
- An alternative is the Positive PMI (Bullinaria and Levy, 2007):

$$PPMI(x,y) = \max(PMI(x,y), 0)$$

Dimensionality reduction

- The word vectors are the rows of the PMI matrix
- The size of word vector is the size of the vocabulary
- Problems:
  - Requires lot of memory: needs to store in sparse matrix all non-zero co-occurence.
  - large dimensional vectors are hard to handle (e.g. in a text classifier)
  - cannot compare word vectors estimated on 2 different corpora unless they have exactly the same vocabulary!
- Solution: build vectors with fixed predefined size from the PMI matrix

Dimensionality reduction

- PMI does not differentiate between words and context: symmetric matrix
- However PMI matrix $\mathbf{M}$ is not positive definite
- We build a similarity matrix between words as: $\mathbf{S} = \mathbf{M}\mathbf{M}^T$
- $\mathbf{S}$ is a symmetric positive definite matrix that measure similarity between words based on PMI
- **Goal** Find a $n \times d$ dimensional matrix $\mathbf{X}_d$ such that:
$$\mathbf{X}_d = \mathrm{argmin}_{\mathbf{Y}} \|\mathbf{S} - \mathbf{Y}\mathbf{Y}^T\|_2^2$$
- $\mathbf{X}_d$'s row are word vectors that explain most of the variance of $\mathbf{S}$, and thus $\mathbf{M}$
- Solution: truncated Singular Value Decomposition (SVD)

Truncated Singular Value Decomposition (SVD)

- The SVD of a matrix $\mathbf{A}$ is:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$$

where Σ is a diagonal matrix with the singular values, and $\mathbf{U}$ and $\mathbf{V}$ are orthonormal basis.

- The truncated SVD is:

$$\mathbf{A}_d = \mathbf{U}_d\Sigma_d\mathbf{V}_d^T$$

Σ_d is the diagonal matrix formed with the d largest singular value.

$\mathbf{U}_d$ is the matrix formed by the d columns of $\mathbf{U}$ corresponding to the d largest singular value.

Dimensionality reduction

- Since $\mathbf{S}$ is definite positive, $\forall i,\ \lambda_i(\mathbf{S}) \geq 0$

- Apply SVD to $\mathbf{S}$, the matrix of word vectors is:

$$\mathbf{X}_d = \mathbf{U}_d(\Sigma_d)^{1/2}$$

- Each row of $\mathbf{X}_d$ is a word vector

- $\mathbf{S}$ and $\mathbf{M}$ gives same matrix $\mathbf{U}_d$ and $\mathbf{V}_d$, and $(\lambda_i(\mathbf{S}))_i = (\lambda_i(\mathbf{M}))_i^2$

Different examples of distributional word representation

We have seen one instance of word vector, but we can vary many parameters:

Linguistic parameters

pre-processing and linguistic annotation - raw text, stemming, POS tagging and lemmatisation, (dependency) parsing, semantically relevant patterns

choice of context - document, sentence, window, dependency relations, etc.

Mathematical parameters

matrix column and row entries - words, document id

context weighting (w) - log-frequency, association scores, entropy, etc.

measuring similarity (s) - cosine similarity, Euclidean, Manhattan, Minkowski (p-norm)

dimensionality reduction (r) - feature selection, SVD projection (PCA), random indexing

---

Source: Foundations of Distributional Semantic Models. Stefan Evert and Alessandro Lenci, 2009.

Different examples of distributional word representation

Latent Semantic Analysis (Landauer and Dumais, 1997)

context documents
matrix word$\times$ document id
w log term frequency and term entropy in the corpus
s cosine
r SVD

Hyperspace Analogue to Language (Lund and Burgess, 1996)

context triangular window-based with position as context-typing function
matrix word$\times$ word
w frequency
s Minkowski metric
r dimensions with the highest variance

Source: Foundations of Distributional Semantic Models. Stefan Evert and Alessandro Lenci, 2009.

Distributional word representation: in a nutshell

- Define what is the context of a word

- Count how many times each target word occurs in this context

$\rightarrow$ co-occurence matrix $\mathbf{M}$

- Build vectors out of (a function of) these context occurrence counts

$\rightarrow$ Similarity matrix $\mathbf{S} = \phi(\mathbf{M})$ (e.g., PMI)

- Reduce dimensionality with SVD

$\rightarrow$ matrix of word representation: $\mathbf{X} = \text{argmin}_{\mathbf{Y} \in \mathbb{R}^{n \times d}} \|\mathbf{S} - \mathbf{Y}\mathbf{Y}^T\|_2^2$

Limitations of this approach

- Building the co-occurence matrix: $O(V^2)$ in memory (e.g. on Common Crawl: $V = 2\text{M}$)
- Complexity of truncated SVD: $O(d^2 V)$
- Inefficient to build a large matrix and reduce it later: Can we do both simultaneously?

Continuous word representations

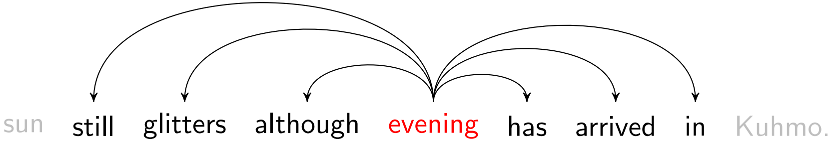
Learning distributed word representation

- Directly learning low dimensional vectors
- Moving from count based statistics to machine learning
- **Key idea 1 (Collobert and Weston, 2008)**
  learning distributed word vectors as a discriminative problem
- **Key idea 2 (Mikolov et al., 2013a)**
  efficient online training to scale to large dataset
- State-of-the-art model: `word2vec` by Mikolov et al. (2013a)

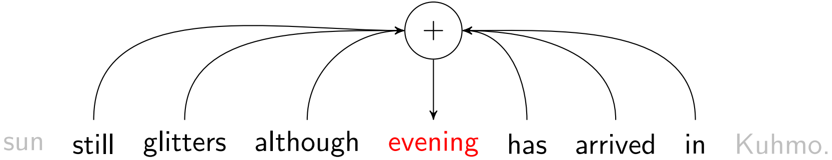
Word2vec: the skipgram and cbow models

- `word2vec`: context is a fixed size window around the word

- **Skipgram** predict context from the word



- **Continuous Bag of Word (Cbow)** predict word from the context



Word2vec: word vectors as a discriminative problem

- Given a vocabulary of V words and a dataset of N tokens:

$$(w_1, ..., w_N) \in \{1, ..., V\}^N$$

- Each word i in the vocabulary is associated with a word vector $\mathbf{x}_i \in \mathbb{R}^d$ and a context vector $\mathbf{y}_i \in \mathbb{R}^d$, with $d << V$

- Denote by $\mathbf{X}$ the matrix with the i-th row equal to $\mathbf{x}_i$ (same for $\mathbf{Y}$)

Skipgram as a discriminative problem

- **Skipgram** predicts each word c in context C_n of n-th token
- Discriminate between correct word in the context against the rest of the vocabulary
- Frame as a minimization problem:

$$\min_{\mathbf{X}\in\mathbb{R}^{V\times d},\ \mathbf{Y}\in\mathbb{R}^{V\times d}} \quad \frac{1}{N}\sum_{n=0}^{N}\left[\frac{1}{|C_n|}\sum_{c\in C_n}\ell(\mathbf{x}_{w_n},\mathbf{y}_c)\right]$$

where:

$$\ell(\mathbf{x},\mathbf{y}) = -\mathbf{x}^T\mathbf{y} + \log\left(\sum_{k=1}^{V}\exp(\mathbf{y}_k^T\mathbf{x})\right)$$

is the negative log-softmax function

Cbow as a discriminative problem

- **Cbow** predicts the word associated with the n-th token based on its context C_n
- The context is represented as a Bag-of-Word (BoW)
- Discriminate between the correct word and the rest of the vocabulary
- Frame as a minimization problem:

$$\min_{\mathbf{X}\in\mathbb{R}^{V\times d},\ \mathbf{Y}\in\mathbb{R}^{V\times d}} \quad \frac{1}{N}\sum_{n=0}^{N} \ell\Big(\underbrace{\frac{1}{|C_n|}\sum_{c\in C_n}\mathbf{x}_c}_{\text{Context BoW}}, \mathbf{y}_{w_n}\Big)$$

where ℓ is the negative log-softmax

Optimization of word2vec

Gradient descent

$$\mathbf{X}_{t+1} \leftarrow \mathbf{X}_t - \alpha_t \frac{1}{N} \sum_{n=1}^{N} \sum_{c \in C_n} \nabla_{\mathbf{X}} \ell(\mathbf{x}_{w_n}, \mathbf{y}_c)$$

→ Requires a pass over dataset for one gradient: $O(N)$

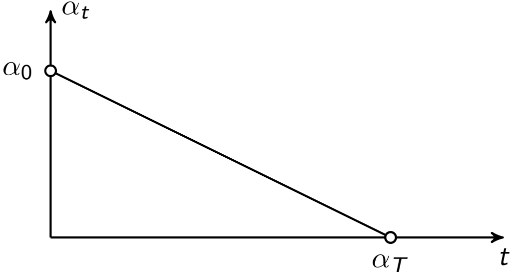
Stochastic gradient descent with predefined sequential scheduler

- loop over the N tokens in dataset, take gradient step at each token
- Repeat process for E epoch. Total number of iteration $T = NE$
- t-th update:

$$\mathbf{X}_{t+1} \leftarrow \mathbf{X}_t - \alpha_t \nabla_{\mathbf{X}} \sum_{c \in C_n} \ell(\mathbf{x}_{w_n}, \mathbf{y}_c)$$

with $n = t/N$

Optimization of word2vec



Learning rate scheduler $(\alpha_t)_t$

- set α_0 and number of iteration T:

$$\alpha_t = \left(1 - \frac{t}{T}\right) \alpha_0$$

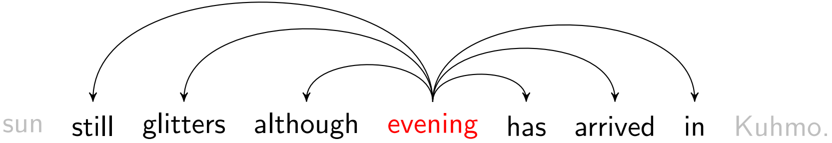
Optimization of word2vec

Hogwild parallelizes this process over P processes:

- split dataset in P subsets.
- Read P subsets in parallel.
- Share parameters between processes
- Each process compute a gradient per token and update shared parameters.

$\rightarrow$ Update parameters sequentially and in parallel.

Word2vec: efficient distributed training



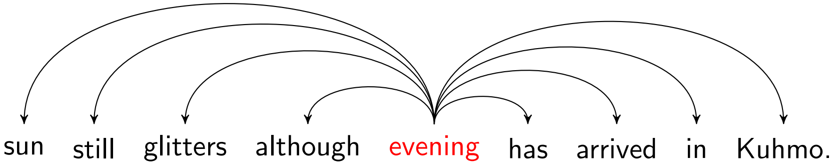
- Instead of fixing the window size $|C_n|$, sample it
- Uniform sample w in $\{1, \ldots, w_{max}\}$ and $C_n = 2w$

Word2vec: efficient distributed training

sun still glitters although evening has arrived in Kuhmo.

- Instead of fixing the window size $|C_n|$, sample it
- Uniform sample w in $\{1, \ldots, w_{max}\}$ and $C_n = 2w$

Word2vec: efficient distributed training



- Instead of fixing the window size $|C_n|$, sample it
- Uniform sample w in $\{1, \ldots, w_{max}\}$ and $C_n = 2w$

Word2vec: efficient distributed training

- Computing softmax over the whole vocabulary is slow $O(V)$

$\rightarrow$ Replace it by negative sampling

- **Negative sampling** (Skipgram) sample $K << V$ words N_n that does not appear in the context of $\mathbf{x}_n$ and replace softmax by sum of 1-versus-all losses:

$$\ell(\mathbf{x}_{w_n}, \mathbf{y}_c) \leftarrow \sigma(\mathbf{x}_{w_n}, \mathbf{y}_c) + \frac{1}{K} \sum_{k \in N_n} \sigma(-\mathbf{x}_{w_n}, \mathbf{y}_k)$$

where $\sigma(\mathbf{x}, \mathbf{y}) = \log(1 + \exp(-\mathbf{x}^T \mathbf{y}))$ is the negative log-sigmoid function

- Important to sample negatives based on word frequency to match dataset distribution:

$$p_{\text{negative}}(w) \propto \text{freq}^{0.75}(w)$$

- Same for cbow

Word2vec: efficient distributed training

- Word frequency in corpora follows a Zipf distribution

- **Zipf distribution** ranked by frequency, each word is x times less frequent than previous one.

  Example: proba(the) = 0.1, proba(a)=0.05, proba(is)=0.025....

→ a subset of vocabulary ($\approx$ 2k words) covers > 80% of dataset

→ 80% of training spent on learning 2k word vectors out of 2M

- discard words during training based on frequency ($t \in [10^{-5}, 10^{-3}]$):

$$p(\texttt{discard} \mid w) = \max\left(0,\ \ 1 - \sqrt{\frac{t}{\text{freq}(w)}}\right)$$

Example of nearest neighbors

- Trained on 1B tokens from Wikipedia, dimension 300

moon	score
mars	0.615
moons	0.611
lunar	0.602
sun	0.602
venus	0.583

talking	score
discussing	0.663
telling	0.657
joking	0.632
thinking	0.627
talked	0.624

blue	score
red	0.704
yellow	0.677
purple	0.676
green	0.655
pink	0.612

Word vector analogies

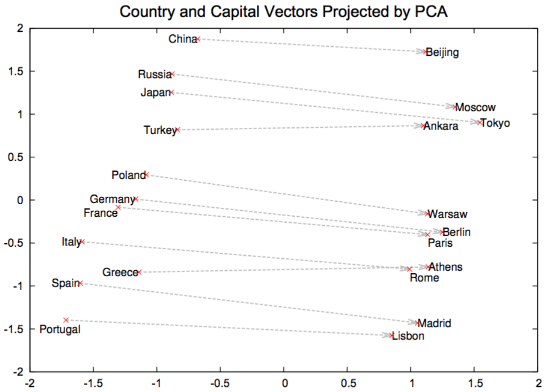


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

Credit: Mikolov et al. (2013)

Evaluation

Analogies as intrinsic evaluation of word representation

- Word vector analogies:

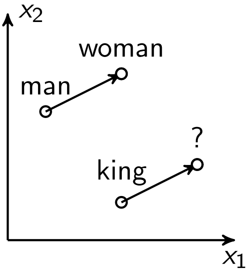
$$\text{king} - \text{man} + \text{woman} = \ ?$$

- Frame as a retrieval problem:

→ Normalize word embeddings $\mathbf{x}_i \leftarrow \mathbf{x}_i / \|\mathbf{x}_i\|$

→ Find the closest vectors w.r.t. l_2 distance:

$$\mathbf{x}_d = \text{argmax}_i (\mathbf{x}_c + \mathbf{x}_b - \mathbf{x}_a)^\top \mathbf{x}_i$$



Analogies as intrinsic evaluation of word representation

- Semantic analogies:
  - `capital-common-countries`:
    Athens : Greece :: Helsinki : Finland
  - `currency`:
    Japan : yen :: Sweden : krona
  - `family`:
    father : mother :: uncle : aunt

- Syntactic analogies:
  - `gram2-opposite`:
    logical : illogical :: clear : unclear
  - `gram3-comparative`:
    strong : stronger :: good : better
  - `gram5-present-participle`:
    think : thinking :: listen : listening

Analogies as intrinsic evaluation of word representation

	PPMI	PPMI+SVD	Skipgram
Analogies	.552	.554	**.694**

Figure: Accuracy on the analogy dataset of Mikolov et al. (2013b)

Source: Levy et al. (2015).

Impact of dimension

	100	200	300	400
Semantic	73.7	80.8	82.2	82.6
Syntactic	69.6	74.4	75.0	74.8
Total	71.2	76.9	77.8	77.9

Figure: Accuracy on the analogy dataset of Mikolov et al. (2013b)

- Take home message:

  dimension 300 is good enough for most applications

Extensions

Extensions: GloVe (Pennington et al., 2014)

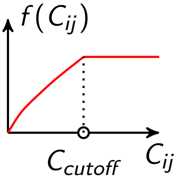
- **GloVe (Global Vector)** is a word2vec model trained with a different loss:

$$\min_{X,Y,b} \sum_{i,\ j \in V} f(C_{ij}) \left(\mathbf{x}_i^T \mathbf{y}_j + b_i + b_j - \log C_{ij} \right)^2$$

- $(b_i)_{i \in V}$ are scalars to learn
- C_{ij} co-occurence counts of words i and j in same context
- f reweighting function:

$$f(C) = \min\left(1, (C/C_{cutoff})^{3/4}\right)$$

similar to discount factor of word2vec



English vectors available at `nlp.stanford.edu/projects/glove/`

Extensions: fastText (Bojanowski et al., 2017)

- Represent a word as bag of character n-grams:

  $\textit{skiing} = \{$ `^skiing$, ^ski, skii, kiin, iing ing$` $\}$
- $\mathcal{G}_w$ is the set of n-grams appearing in word w.

  $$s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{g}^\top \mathbf{c}.$$

  (It includes the word w in the set of n-grams)
- **Advantage 1** Get word vectors for out-of-vocabulary words using subwords!
- **Advantage 2** Generalize well to text with typos or agglutinative languages

Pre-trained vectors in 90 languages available at `www.fasttext.cc`

Technical details of fastText

- n-grams between 3 and 6 characters
- Hashing to map n-grams to integers in 1 to K
- Same training / sampling procedure as in word2vec

$\rightarrow$ Less than $2\times$ slower than word2vec skipgram!

Pre-trained vectors in 90 languages available at `www.fasttext.cc`

Experiments – word analogy (A is to B as C is to ?)

- All models trained on Wikipedia:

		sg	cbow	ours
Cs	Semantic	25.7	**27.6**	**27.5**
	Syntactic	52.8	55.0	**77.8**
De	Semantic	66.5	**66.8**	62.3
	Syntactic	44.5	45.0	**56.4**
En	Semantic	**78.5**	78.2	77.8
	Syntactic	70.1	69.9	**74.9**
It	Semantic	52.3	**54.7**	52.3
	Syntactic	51.5	51.8	**62.7**

Table: Accuracy of our model and baselines on word analogy tasks for Czech, German, English and Italian. We report results for semantic and syntactic analogies separately.

Further Extensions

- **Position vectors** multiply input word vectors of cbow by position vectors (Mnih and Kavukcuoglu, 2013):

$$\mathbf{h}_C = \frac{1}{|P|} \sum_{p \in P} \mathbf{d}_p \odot \mathbf{x}_{n+c}$$

  $\mathbf{d}_p$ =learnable position vectors. $\odot$ = pointwise multiplication.

- Reminder, regular cbow:

$$\mathbf{h}_C = \frac{1}{|P|} \sum_{p \in P} \mathbf{x}_{n+c}$$

Further Extensions

- **Phrase vectors** pre-processing of dataset to convert with probability, bigrams with high MI into token (Mikolov et al., 2013b):

  $$\text{New York} \rightarrow \text{New\_York}$$

  Repeat process:

  $$\text{New York University} \rightarrow \text{New\_York\_University}$$

- Score to merge two tokens:

  $$\text{score}(w_i,\ w_j) = \frac{\text{count}(w_i\ w_j) - \delta}{\text{count}(w_i) \times \text{count}(w_j)}$$

  where δ is a discount factor to prevent phrases of infrequent words

→ These extensions are in new fastText vectors (Mikolov et al., 2017)

Models in 150+ languages from Grave et al. (2018) available at `www.fasttext.cc`

Evaluation of these extensions

	Semantic	Syntactic	Total
cbow	79	73	76
cbow + phrases	82	78	80
cbow + phrases + position	87	82	85

Models trained on Common Crawl (Mikolov et al., 2017)

Impact of training data

- Wikipedia: high quality but small
  - 28 languages with more than 100M tokens
  - Hindi: only 39M tokens
- Crawl: noisy but larger and more domains
- Preprocessing: language id / deduplication / tokenization

language	wiki	crawl
German	1.3B	65B
French	1.1B	68B
Japanese	1.0B	92B
Russian	0.8B	102B
Spanish	0.8B	72B

language	wiki	crawl
Italian	0.7B	36B
Polish	0.4B	21B
Portuguese	0.4B	35B
Chinese	0.4B	30B
Czech	0.2B	13B

Table: Dataset sizes (number of tokens) for Wikipedia and Crawl.

Impact of training data

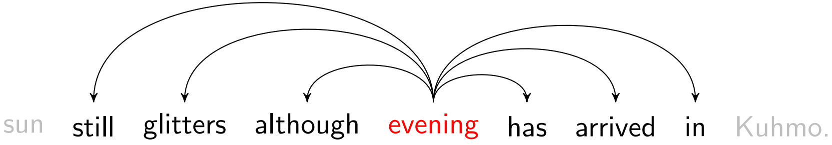
Model	Dataset	**Analogy**	**Similarity (RW)**	**QA**
GloVe	Wiki + news	72	0.38	77.7
GloVe	Crawl	75	0.50	78.8
fastText	Wiki + news	87	0.52	78.9
fastText	Crawl	85	0.58	79.8

Results from Mikolov et al. (2017). **Analogy:** accuracy on the Google analogy dataset. **Similarity (RW):** Spearman rank correlation on the Stanford Rare Word dataset. **QA:** F1 score on the SQuAD question answering dataset. Pre-trained word vectors were used to initialize the lookup table of the RNN DrQA model from Chen et al. (2017).

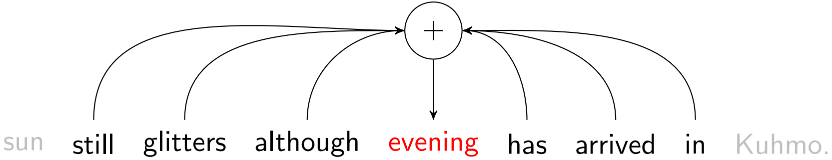
Word vectors recap

Word2vec: the skipgram and cbow models

- `word2vec`: context is a fixed size window around the word
- **Skipgram** predict context from the word



- **Continuous Bag of Word (Cbow)** predict word from the context



Cbow as a discriminative problem

- **Cbow** predicts the n-th token based on its context C_n
- The context is represented as a Bag-of-Word (BoW):

$$\mathbf{h} = \sum_{c \in C_n} \mathbf{x}_c$$

- Given representation of context $\mathbf{h}$, predict w_n with softmax over vocabulary

$$\min_{\mathbf{X} \in \mathbb{R}^{V \times d},\ \mathbf{Y} \in \mathbb{R}^{V \times d}} \quad \frac{1}{N} \sum_{n=0}^{N} \log \left(\frac{\exp(\mathbf{h}^\top \mathbf{y}_{w_n})}{\sum_{k=1}^{V} \exp(\mathbf{h}^\top \mathbf{y}_{w_n})} \right)$$

Skipgram as a discriminative problem

- **Skipgram** predicts each word c in context C_n of n-th token
- Given word vector $\mathbf{x}_{w_n}$, predict c with softmax over vocabulary:

$$\min_{\mathbf{X}\in\mathbb{R}^{V\times d},\ \mathbf{Y}\in\mathbb{R}^{V\times d}} \quad \frac{1}{N}\sum_{n=0}^{N}\left[\frac{1}{|C_n|}\sum_{c\in C_n}\log\left(\frac{exp(\mathbf{x}_{w_n}^{\top}\mathbf{y}_c)}{\sum_{k=1}^{V}\exp(\mathbf{x}_{w_n}^{\top}\mathbf{y}_k)}\right)\right]$$

Skipgram as a discriminative problem

- **Skipgram** predicts each word c in context C_n of n-th token
- Given word vector $\mathbf{x}_{w_n}$, predict c with softmax over vocabulary:

$$\min_{\mathbf{X}\in\mathbb{R}^{V\times d},\ \mathbf{Y}\in\mathbb{R}^{V\times d}} \quad \frac{1}{N}\sum_{n=0}^{N}\left[\frac{1}{|C_n|}\sum_{c\in C_n}\log\left(\frac{exp(\mathbf{x}_{w_n}^{\top}\mathbf{y}_c)}{\sum_{k=1}^{V}\exp(\mathbf{x}_{w_n}^{\top}\mathbf{y}_k)}\right)\right]$$

- Sum over all the tokens of the training data

Skipgram as a discriminative problem

- **Skipgram** predicts each word c in context C_n of n-th token
- Given word vector $\mathbf{x}_{w_n}$, predict c with softmax over vocabulary:

$$\min_{\mathbf{X}\in\mathbb{R}^{V\times d},\ \mathbf{Y}\in\mathbb{R}^{V\times d}} \quad \frac{1}{N}\sum_{n=0}^{N}\left[\frac{1}{|C_n|}\sum_{c\in C_n}\log\left(\frac{exp(\mathbf{x}_{w_n}^\top \mathbf{y}_c)}{\sum_{k=1}^{V}\exp(\mathbf{x}_{w_n}^\top \mathbf{y}_k)}\right)\right]$$

- Sum over all the tokens of the training data
- Sum over all the elements of the context

Skipgram as a discriminative problem

- **Skipgram** predicts each word c in context C_n of n-th token
- Given word vector $\mathbf{x}_{w_n}$, predict c with softmax over vocabulary:

$$\min_{\mathbf{X}\in\mathbb{R}^{V\times d},\ \mathbf{Y}\in\mathbb{R}^{V\times d}} \quad \frac{1}{N}\sum_{n=0}^{N}\left[\frac{1}{|C_n|}\sum_{c\in C_n}\log\left(\frac{exp(\mathbf{x}_{w_n}^{\top}\mathbf{y}_c)}{\sum_{k=1}^{V}\exp(\mathbf{x}_{w_n}^{\top}\mathbf{y}_k)}\right)\right]$$

- Sum over all the tokens of the training data
- Sum over all the elements of the context
- Softmax over the vocabulary

Bias in word vectors

Background: Implicit Association Test

- Based on four sets of words:

  `Math`: {*math, algebra, geometry, calculus, equations, numbers, ...*}

  `Arts`: {*poetry, art, dance, literature, novel, symphony, drama, ...*}

  `Male`: {*male, man, boy, brother, he, him, his, son*}

  `Female`: {*female, woman, girl, sister, she, her, hers, daughter*}

- Target words: `Math` and `Arts`
- Attributes: `Male` and `Female`
- Objective: determine if math is more associated to male or female

Background: Implicit Association Test

- Based on reaction time to classify word into category:

`Math` `Arts`

geometry

Background: Implicit Association Test

- Based on reaction time to classify word into category:

`Math`	`Arts`
`Female`	`Male`

dance

Background: Implicit Association Test

- Based on reaction time to classify word into category:

`Math`	`Arts`
`Male`	`Female`

dance

Background: Implicit Association Test

- Based on reaction time to classify word into category:

`Math`	`Arts`
`Male`	`Female`
dance	

- Compare reaction time of pairings:

`Math/Male` and `Arts/Female`

v.s.

`Math/Female` and `Arts/Male`

Word embedding association test

Caliskan, Bryson, Narayanan (2017)

- Replicate implicit assocition test with word vectors
- Objective: measure strength of association of four sets of words
- Given a word w and two sets of attribute A and B:

$$s(w, A, B) = \frac{1}{\mathrm{card}(A)} \sum_{a \in A} \cos(w, a) - \frac{1}{\mathrm{card}(B)} \sum_{b \in B} \cos(w, b)$$

  measure association of w to attribute.

- Then, given two sets of word X and Z of equal size

$$s(X, Z, A, B) = \sum_{x \in X} s(x, A, B) - \sum_{z \in Z} s(z, A, B)$$

Experiments

Caliskan, Bryson, Narayanan (2017)

- Replicate humans implicit association test results
- Word vectors trained with `word2vec` on 100B news tokens
- Example of biases:

target words	attributes	p
Flowers vs insects	Pleasant vs unpleasant	10^{-7}
Instruments vs weapons	Pleasant vs unpleasant	10^{-7}
Eur.-Am. vs Afr.-Am. names	Pleasant vs unpleasant	10^{-8}
Male vs female names	Career vs family	10^{-3}
Science vs arts	Male vs female terms	10^{-2}

Experiments

Caliskan, Bryson, Narayanan (2017)

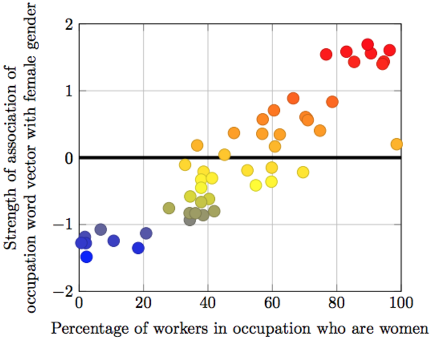


Figure 1: Occupation-gender association. Pearson's correlation coefficient $\rho = 0.90$ with p-value $< 10^{-18}$.

Credit: Caliskan, Bryson, Narayanan (2017)

Bias in word vectors

- Word vectors: capture the biases from the data
- Human biases in data implies human biases in word vectors
- Choice of training data: big impact on biases!
- Careful when using word vectors: biased classifier/system
- But, might be useful to study biases in large corpora!

Limitations

Limitations of word vectors

- **Antonyms:**

small	score
large	0.807
tiny	0.798
smallish	0.730
smalll	0.722
largish	0.693

fast	score
Fast	0.668
super-fast	0.646
slow	0.619
faster	0.603
quick	0.578

bad	score
good	0.751
terrible	0.731
horrible	0.718
lousy	0.708
baaaad	0.702

Limitations of word vectors

- **High similarity:**

Limitations of word vectors

- **High similarity:** plural? synonym? hypernym? co-hyponym?

car	score
cars	0.733
vehicle	0.727
automobile	0.702
Car	0.659
truck	0.647

Limitations of word vectors

- **High similarity:** plural? synonym? hypernym? co-hyponym?

car	score
cars	0.733
vehicle	0.727
automobile	0.702
Car	0.659
truck	0.647

Limitations of word vectors

- **High similarity:** plural? synonym? hypernym? co-hyponym?

car	score
cars	0.733
vehicle	0.727
automobile	0.702
Car	0.659
truck	0.647

Limitations of word vectors

- **High similarity:** plural? synonym? hypernym? co-hyponym?

car	score
cars	0.733
vehicle	0.727
automobile	0.702
Car	0.659
truck	0.647

Limitations of word vectors

Psychological Review
Copyright © 1977 by the American Psychological Association, Inc.
VOLUME 84 NUMBER 4 JULY 1977

Features of Similarity

Amos Tversky
Hebrew University
Jerusalem, Israel

- Tversky (1977): metric assumption of human similarity?
- Human similarity: no symmetry

$$\text{sim}(\texttt{ellipse},\ \texttt{circle}) > \text{sim}(\texttt{circle},\ \texttt{ellipse})$$

- Human similarity: no triangular inequality

$$d(\texttt{ball},\ \texttt{moon}) + d(\texttt{moon},\ \texttt{light}) \leq d(\texttt{ball},\ \texttt{light})$$

References I

Baroni, M. and Lenci, A. (2010). Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4).

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Bullinaria, J. A. and Levy, J. P. (2007). Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods*, 39(3):510–526.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. ICML*.

Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational linguistics*, 19(1):61–74.

References II

Firth, J. R. (1957). *Papers in linguistics, 1934-1951*. Oxford University Press.

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. (2018). Learning word vectors for 157 languages. *arXiv preprint arXiv:1802.06893*.

Landauer, T. K. and Dumais, S. T. (1997). A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

Levy, O., Goldberg, Y., and Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Lund, K. and Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2).

References III

Manning, C. D. and Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2017). Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Adv. NIPS*.

Mnih, A. and Kavukcuoglu, K. (2013). Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273.

References IV

Padó, S. and Lapata, M. (2007). Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.