GloVe & FastText

Nora Graichen, Insa Kröger & Aqsa Nazir, Egla Hajdini

Saarland University
Seminar: Embeddings for NLP and IR
Lecturer: Cristina España i Bonet

May 29th, 2019

GloVe

Global Vectors for Word Representation

J. Pennington, R. Socher and C. Manning

Saarland University
Seminar: Embeddings for NLP and IR
Lecturer: Cristina España i Bonet

Nora Graichen and Insa Kröger
May 29th, 2019

GloVe, is a new global log-bilinear regression model for the unsupervised learning of word representations that outperforms other models on word analogy, word similarity, and named entity recognition tasks.

— GloVe: Global Vectors for Word Representation, 2014.

Table of contents

Methods to produce linear directions of meaning

Matrix factorization	Shallow-window methods

Methods to produce linear directions of meaning

Matrix factorization	Shallow-window methods
LSA, HAL (Lund & Burgess, 1996) **COALS method** (Rohde et al., 2006) **Hellinger PCA (HPCA)** (Lebret and Collobert, 2014)	**skip-gram and continuous bag-of-words (CBOW)** (Mikolov et al. , 2013) **vLBL and ivLBL** (Mnih & Kavukcuoglu, 2013) **NNLM** (Bengio et al., 2003) **HLBL** (Collobert & Weston, 2008)

Methods to produce linear directions of meaning

Matrix factorization	Shallow-window methods
LSA, HAL (Lund & Burgess, 1996) COALS method (Rohde et al., 2006) Hellinger PCA (HPCA) (Lebret and Collobert, 2014)	skip-gram and continuous bag-of-words (CBOW) (Mikolov et al. , 2013) vLBL and ivLBL (Mnih & Kavukcuoglu, 2013) NNLM (Bengio et al., 2003) HLBL (Collobert & Weston, 2008)
efficient usage of statistics, co-occurrence counts	
fast training	

Methods to produce linear directions of meaning

Matrix factorization	Shallow-window methods
LSA, HAL (Lund & Burgess, 1996) COALS method (Rohde et al., 2006) Hellinger PCA (HPCA) (Lebret and Collobert, 2014)	skip-gram and continuous bag-of-words (CBOW) (Mikolov et al. , 2013) vLBL and ivLBL (Mnih & Kavukcuoglu, 2013) NNLM (Bengio et al., 2003) HLBL (Collobert & Weston, 2008)
efficient usage of statistics, co-occurrence counts	
fast training	
captures word similarity primarily	
large counts (frequent/function words) → disproportionate importance	

Methods to produce linear directions of meaning

Matrix factorization	Shallow-window methods
LSA, HAL (Lund & Burgess, 1996) COALS method (Rohde et al., 2006) Hellinger PCA (HPCA) (Lebret and Collobert, 2014)	skip-gram and continuous bag-of-words (CBOW) (Mikolov et al. , 2013) vLBL and ivLBL (Mnih & Kavukcuoglu, 2013) NNLM (Bengio et al., 2003) HLBL (Collobert & Weston, 2008)
efficient usage of statistics, co-occurrence counts	inefficient usage of statistics (repetition)
fast training	scales with corpus size
captures word similarity primarily	
large counts (frequent/function words) → disproportionate importance	

Methods to produce linear directions of meaning

Matrix factorization	Shallow-window methods
LSA, HAL (Lund & Burgess, 1996) COALS method (Rohde et al., 2006) Hellinger PCA (HPCA) (Lebret and Collobert, 2014)	skip-gram and continuous bag-of-words (CBOW) (Mikolov et al. , 2013) vLBL and ivLBL (Mnih & Kavukcuoglu, 2013) NNLM (Bengio et al., 2003) HLBL (Collobert & Weston, 2008)
efficient usage of statistics, co-occurrence counts	inefficient usage of statistics (repetition)
fast training	scales with corpus size
captures word similarity primarily	captures complex patterns of natural language
large counts (frequent/function words) → disproportionate importance	performance word analogy, semantic and syntactic tasks

Table of contents

GloVe Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014

- *Global Vectors* (GloVe)

The statistics of word occurrences in a corpus is the primary source of information available to all unsupervised methods for learning word representations.

GloVe - Introduction

- *Global Vectors* (GloVe)

Methods to produce linear directions of meaning

Matrix factorization	Shallow-window methods
LSA, HAL (Lund & Burgess, 1996) COALS method (Rohde et al., 2006) Hellinger PCA (HPCA) (Lebret and Collobert, 2014)	skip-gram and continuous bag-of-words (CBOW) (Mikolov et al. , 2013) vLBL and ivLBL (Mnih & Kavukcuoglu, 2013) NNLM (Bengio et al., 2003) HLBL (Collobert & Weston, 2008)
efficient usage of statistics, co-occurrence counts	inefficient usage of statistics (repetition)
fast training	scales with corpus size
captures word similarity primarily	captures complex patterns of natural language
large counts (frequent/function words) → disproportionate importance	performance word analogy, semantic and syntactic tasks

GloVe - Introduction

- *Global Vectors* (GloVe)

- fast training
- scalable to huge corpora
  → captures rare words

- small corpus, small vectors: good performance
  (efficient statistic usage!)

Methods to produce linear directions of meaning

Matrix factorization	Shallow-window methods
LSA, HAL (Lund & Burgess, 1996) COALS method (Rohde et al., 2006) Hellinger PCA (HPCA) (Lebret and Collobert, 2014)	skip-gram and continuous bag-of-words (CBOW) (Mikolov et al. , 2013) vLBL and ivLBL (Mnih & Kavukcuoglu, 2013) NNLM (Bengio et al., 2003) HLBL (Collobert & Weston, 2008)
efficient usage of statistics, co-occurrence counts	inefficient usage of statistics (repetition)
fast training	scales with corpus size
captures word similarity primarily	captures complex patterns of natural language
large counts (frequent/function words) → disproportionate importance	performance word analogy, semantic and syntactic tasks

Does it work?

O. frog

nearest neighbors

Does it work?

O. frog

nearest neighbors

Does it work? - GloVe results

O. frog

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

Does it work? - GloVe results

O. frog

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus




Does it work? - Yes!

O. frog

1. frogs
2. toad
3. **litoria**
4. **leptodactylidae**
5. **rana**
6. lizard
7. **eleutherodactylus**

3. litoria

4. leptodactylidae

5. rana

7. eleutherodactylus

Table of contents

How GloVe finds meaning in statistics

ratios of word-word co-occurrence probabilities encode meaning:

How GloVe finds meaning in statistics

ratios of word-word co-occurrence probabilities encode meaning:

Probability and Ratio	***k = solid***	***k = gas***	***k = water***	***k = fashion***
P(k|ice)	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
P(k|steam)	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
P(k|ice)/P(k|steam)	8.9	8.5×10^{-2}	1.36	0.96

How GloVe finds meaning in statistics

ratios of word-word co-occurrence probabilities encode meaning:

Probability and Ratio	***k = solid***	***k = gas***	***k = water***	***k = fashion***
P(k|ice)	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
P(k|steam)	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
P(k|ice)/P(k|steam)	8.9	8.5×10^{-2}	1.36	0.96

no colour indicates relative probability and ratio close to 1

lighter colour indicates relative low probability and ratio

darker colour indicates relative high probability and ratio

How GloVe finds meaning in statistics

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

How GloVe finds meaning in statistics

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

word pair

How GloVe finds meaning in statistics

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

word pair

counts from word-occurrence matrix

How GloVe finds meaning in statistics

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

dot product of target words

How GloVe finds meaning in statistics

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

dot product of target words
plus additional bias

How GloVe finds meaning in statistics

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

least squares problem

How GloVe finds meaning in statistics

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

least squares problem

minimizing distance between inner product and log count of word pair

How GloVe finds meaning in statistics

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

weighting function

How GloVe finds meaning in statistics

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

weighting function

no overweighting of rare or very frequent occurrences → avoids noise

How GloVe finds meaning in statistics

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

weighting function

no overweighting of rare or very frequent occurrences → avoids noise

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} .$$

How GloVe finds meaning in statistics

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

weighting function

no overweighting of rare or very frequent oc

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} .$$

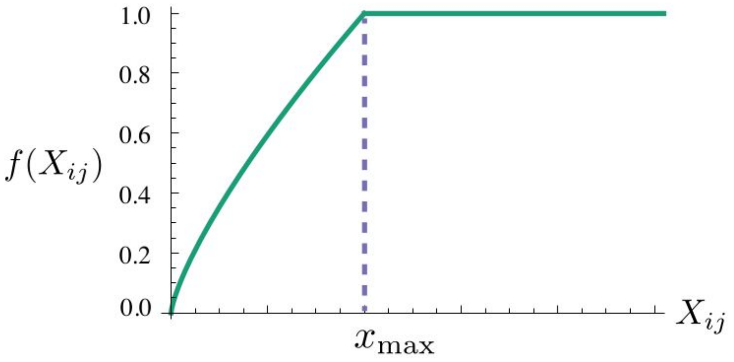


Figure 1: Weighting function f with $\alpha = 3/4$.

Table of contents

Training

- matrix of co-occurrence counts
- large corpora: computationally expensive

Training

- matrix of co-occurrence counts
- large corpora: computationally expensive
- collecting statistics in single pass through corpus
- training on non-zero entries

Training

- matrix of co-occurrence counts
- large corpora: computationally expensive
- collecting statistics in single pass through corpus
- training on non-zero entries
- AdaGrad
    - stochastic gradient descent with per-feature adaptive learning rate
    - learning rate: 0.05

Table of contents

GloVe - Model Analysis

- **Vector Length and Context Size:**
  → around 300 dimension
  → 8 around each center word, not asymmetric!
- **Corpus Size:**
  → overall: bigger is better

- **Run Time:**
  → depends on various factors (↑),
  populating matrix X in 85 minutes: 2.1GHz Intel Xeon E5-2658 machine
  10 word symmetric context window
  400’000 word vocabulary
  6 billion token corpus

Compared to other Models - skip-gram, ivLBL

for model comparison:
rewritten skip-gram model equation

$$J = -\sum_{i=1}^{V} X_i \sum_{j=1}^{V} P_{ij} \log Q_{ij} = \sum_{i=1}^{V} X_i H(P_i, Q_i)$$

maximizes log probability of context windows

Compared to other Models - skip-gram, ivLBL

GloVe model similar to a “global skip-gram” model

$$J = -\sum_{i=1}^{V} X_i \sum_{j=1}^{V} P_{ij} \log Q_{ij} = \sum_{i=1}^{V} X_i \boxed{H(P_i, Q_i)}$$

cross entropy
similarity to least squares problem

Compared to other Models - skip-gram, ivLBL

GloVe model similar to a “global skip-gram” model

$$J = -\sum_{i=1}^{V} X_i \sum_{j=1}^{V} P_{ij} \log Q_{ij} = \sum_{i=1}^{V} X_i \boxed{H(P_i, Q_i)}$$

cross entropy
similarity to least squares problem
drawback → overweighting of unlikely events

Word2Vec - Negative Sampling

each training sample updates only a small percentage of the model’s weights

Word2Vec - Negative Sampling

each training sample updates only a small percentage of the model's weights

$\longrightarrow$ reduces training time

$\longrightarrow$ improves quality word vectors

Word2Vec - Negative Sampling

each training sample updates only a small percentage of the model's weights

Example:

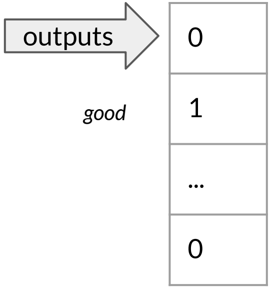
X_{ij} = *("dog", "good")*

Word2Vec - Negative Sampling

each training sample updates only a small percentage of the model's weights

Example:

X_{ij} = ("dog", "good")

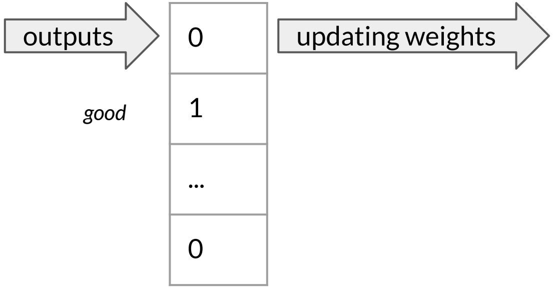


Word2Vec - Negative Sampling

each training sample updates only a small percentage of the model's weights

Example:

X_{ij} = (*"dog", "good"*)



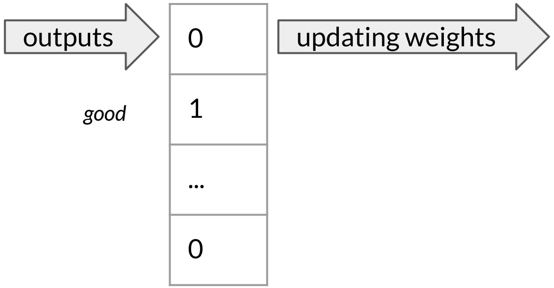
negative sample size: 6

Word2Vec - Negative Sampling

each training sample updates only a small percentage of the model's weights

Example:

X_{ij} = (*"dog", "good"*)



negative sample size: 6

→ update weights on output layer for target

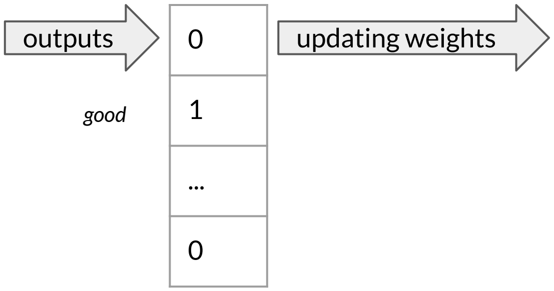
+ 6 additional words = 0

Word2Vec - Negative Sampling

each training sample updates only a small percentage of the model's weights

Example:

X_{ij} = (*"dog", "good"*)



negative sample size: 6

→ update weights on output layer for target
+ 6 additional words = 0

weight matrix of 300 × ~~10.000~~ 6
→ 1800 weight updates instead of 3M!

Word2Vec - Negative Sampling

- each training sample updates only a small percentage of the model’s weights

  → reduces training time
  → improves quality word vectors

- good performance with 10 negative samples

Word2Vec - Negative Sampling

- each training sample updates only a small percentage of the model's weights

  → reduces training time
  → improves quality word vectors

- good performance with 10 negative samples

makes comparison with GloVe difficult

Table of contents

General Evaluation Methods for NLP

- intrinsic
  - evaluation on intermediate subtask
  - fast to compute
  - questionable: correlation to real tasks

- extrinsic
  - eval on real task
  - time consuming
  - difficult to see subsystem interaction

Evaluation

methods: intrinsic or extrinsic

word analogy, semantic and syntactic tasks - intrinsic

word similarity - intrinsic

named entity recognition - extrinsic

Evaluation: Word Vector Analogies

- new evaluation task, came out with word2vec paper
- How well does the word vector captures intuitive semantic and syntactic analogy patterns?
- works with cosine distance after addition

a:b :: c:?

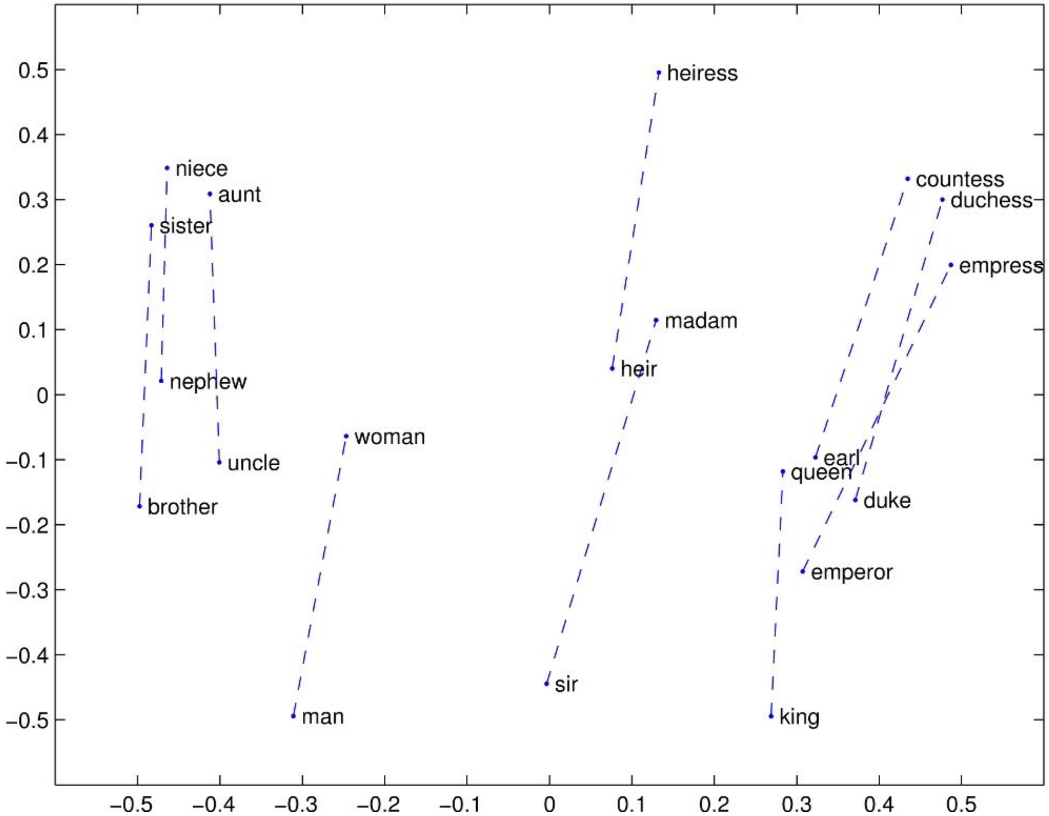
man:woman :: king:?

$$d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$

Evaluation: Word Vector Analogies

a:b :: c:?

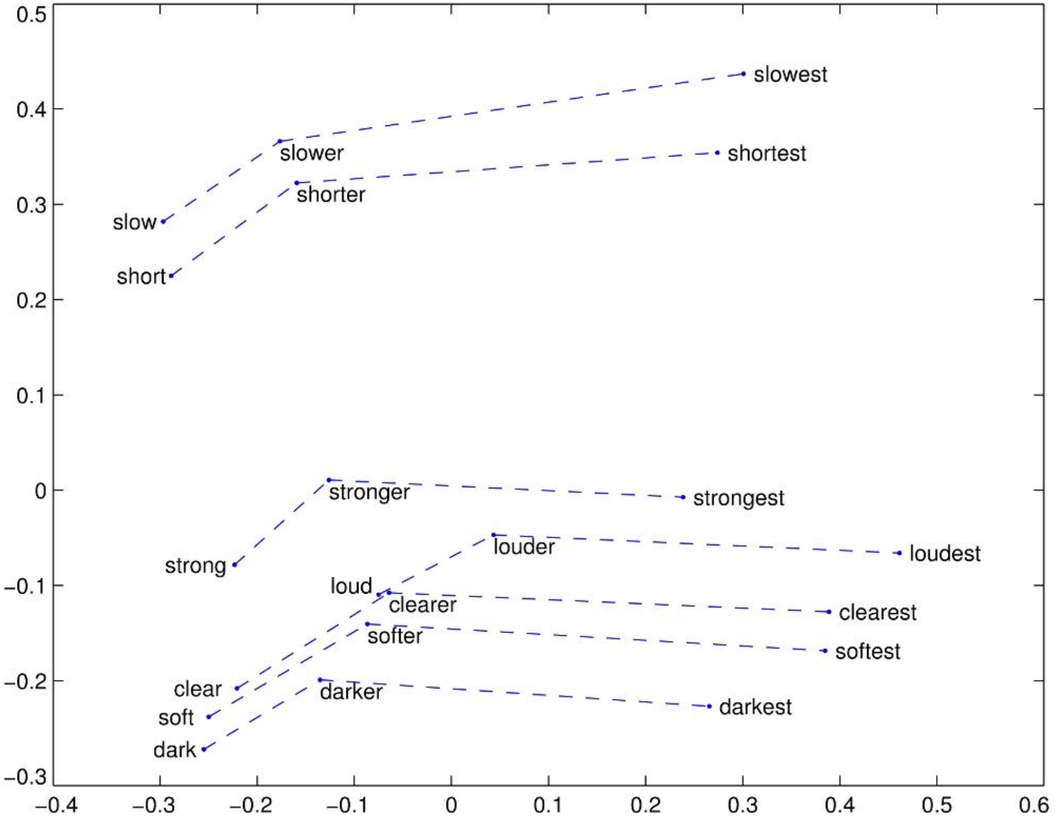
man:woman :: king:?



Evaluation: Word Vector Analogies

a:b :: c:?

man:woman :: king:?



Evaluation: Word Vector Analogies

Results on the word analogy task, given as percent accuracy. Underlined scores are best within groups of similarly-sized models; bold scores are best overall.

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	67.4	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	**81.9**	**69.3**	**75.0**

a:b :: c:?

man:woman :: king:?

$$d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$

Evaluation: Word Vector Analogies

a:b :: c:?

man:woman :: king:?

$$d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$

dimension

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	67.4	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	**81.9**	**69.3**	**75.0**

Evaluation: Word Vector Analogies

a:b :: c:?

man:woman :: king:?

$$d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$

data size

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	<u>67.5</u>	<u>54.3</u>	<u>60.3</u>
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	<u>64.8</u>	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	<u>80.8</u>	61.5	<u>70.3</u>
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	<u>67.4</u>	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	<u>77.4</u>	67.0	<u>71.7</u>
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	**<u>81.9</u>**	**<u>69.3</u>**	**<u>75.0</u>**

Evaluation: Word Vector Analogies

dimension

data size

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	67.4	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	**81.9**	**69.3**	**75.0**

a:b :: c:?

man:woman :: king:?

$$d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$

Evaluation: Word Vector Analogies

data size

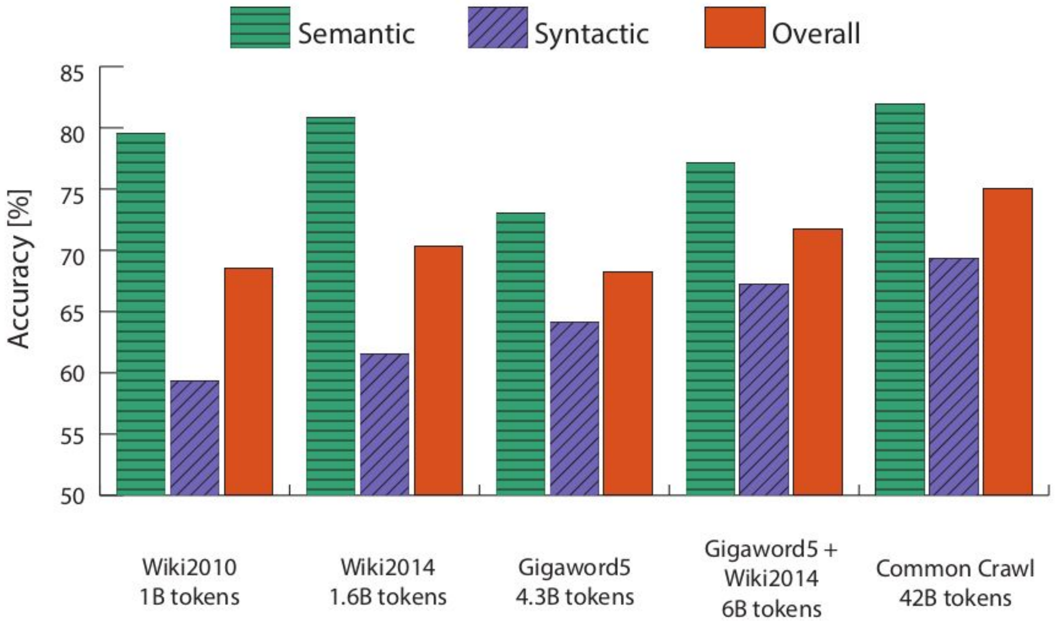


Figure 3: Accuracy on the analogy task for 300-dimensional vectors trained on different corpora.

Evaluation: Word Similarity

- WordSim-353 (Finkelstein et al., 2001):
  word vector distances and their correlation with human judgments

Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10.00
book	paper	7.46
computer		internet 7.58
plane	car	5.77
professor		doctor 6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

```
w1 = "sweden"
model_gigaword.wv.most_similar(positive=w1,topn=10)
```

Evaluation: Word Similarity

- WordSim-353 (Finkelstein et al., 2001):
  word vector distances and their correlation with human judgments

Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10.00
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

closest words to **sweden**:

```
[('denmark', 0.8624402284622192),
 ('norway', 0.8073249459266663),
 ('finland', 0.7906495332717896),
 ('netherlands', 0.7468465566635132),
 ('austria', 0.7466837167739868),
 ('switzerland', 0.7233394384384155),
 ('germany', 0.7173627018928528),
 ('swedish', 0.7107290029525757),
 ('belgium', 0.7081865072250366),
 ('hungary', 0.6932627558708191)]
```

Evaluation: Word Similarity

- WordSim-353 (Finkelstein et al., 2001):
  word vector distances and their correlation with human judgments

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	72.7	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	58.1	37.2
GloVe	6B	65.8	72.7	77.8	53.9	38.1
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	**75.9**	**83.6**	**82.9**	**59.6**	**47.8**

Spearman rank correlation on word similarity tasks with 300-dimensional vectors

Evaluation: Word Similarity

- WordSim-353 (Finkelstein et al., 2001):
  word vector distances and their correlation with human judgments

data size

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	72.7	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	58.1	37.2
GloVe	6B	65.8	72.7	77.8	53.9	38.1
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	**75.9**	**83.6**	**82.9**	**59.6**	**47.8**

Spearman rank correlation on word similarity tasks with 300-dimensional vectors

Evaluation

methods: intrinsic or extrinsic

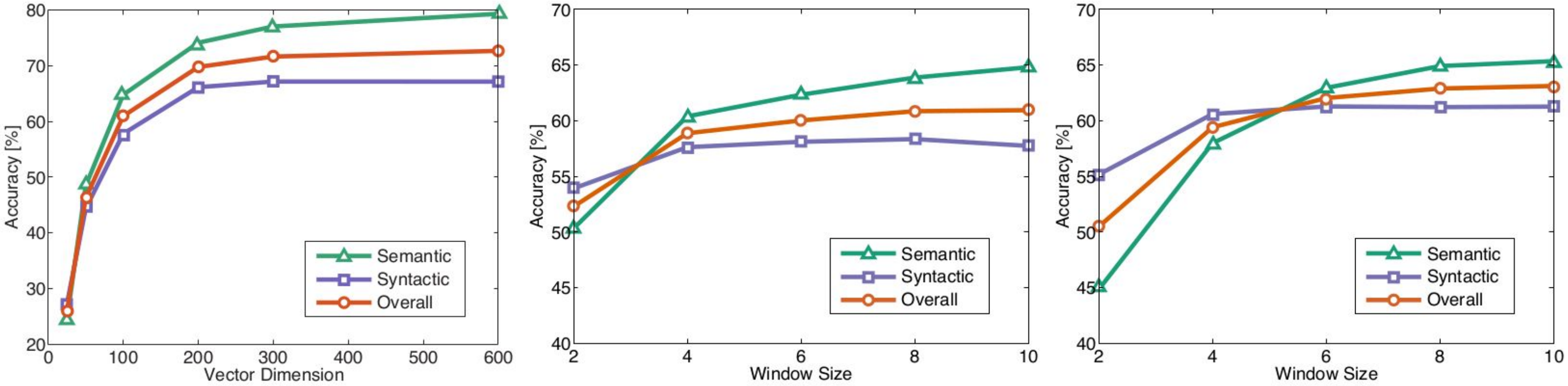
word analogy, semantic and syntactic tasks - intrinsic

word similarity - intrinsic

named entity recognition - extrinsic

Hyperparameters (for your embedding?)

- best dimension: 300
- Window size of 8 around each center word
- don't use asymmetric context



(a) Symmetric context (b) Symmetric context (c) Asymmetric context

Figure 2: Accuracy on the analogy task as function of vector size and window size/type. All models are trained on the 6 billion token corpus. In (a), the window size is 10. In (b) and (c), the vector size is 100.

Evaluation

methods: intrinsic or extrinsic

word analogy, semantic and syntactic tasks - intrinsic

word similarity - intrinsic

named entity recognition - extrinsic

Evaluation: Named Entity Recognition

- CoNLL-2003, doc - collection of Reuters news articles,
  4 entity types: person, location, organization, and miscellaneous

Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	**88.7**	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	**93.2**	88.3	**82.9**	**82.2**

F1 score on NER task with 50d vectors

Evaluation: Named Entity Recognition

- CoNLL-2003, doc - collection of Reuters news articles,
  4 entity types: person, location, organization, and miscellaneous

Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	**88.7**	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	**93.2**	88.3	**82.9**	**82.2**

F1 score on NER task with 50d vectors

Evaluation: Named Entity Recognition

- CoNLL-2003, doc - collection of Reuters news articles,
  4 entity types: person, location, organization, and miscellaneous

Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	**88.7**	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	**93.2**	88.3	**82.9**	**82.2**

F1 score on NER task with 50d vectors

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3

Results on the word analogy task

Table of contents

Conclusion

- count-based method over whole corpus

  $\rightarrow$ takes advantage of global information

Conclusion

- count-based method over whole corpus

  $\rightarrow$ takes advantage of global information
- considers and reduces noise from high frequency terms

Conclusion

- count-based method over whole corpus

  $\rightarrow$ takes advantage of global information
- considers and reduces noise from high frequency terms
- **but** fair comparison to word2vec is difficult

Conclusion

- count-based method over whole corpus

  $\rightarrow$ takes advantage of global information
- considers and reduces noise from high frequency terms
- **but** fair comparison to word2vec is difficult
- offers pre-trained word embeddings $\rightarrow$ reduces training

Conclusion

- count-based method over whole corpus

  → takes advantage of global information
- considers and reduces noise from high frequency terms
- **but** fair comparison to word2vec is difficult
- offers pre-trained word embeddings → reduces training
- shows **significant better accuracy** to baseline models

Questions?

References

Pennington, J.; Socher, R. & Manning, C. D.
GloVe: Global Vectors for Word Representation
Empirical Methods in Natural Language Processing (EMNLP), **2014**, 1532-1543

References - Images

- https://nlp.stanford.edu/projects/glove/
- https://www.petmd.com/reptile/care/evr_rp_frog-care-101-what-you-need-know-you-get-frog
- https://www.youtube.com/watch?v=ASn7ExxLZws
- https://skymind.ai/wiki/word2vec#glove

FastText

Enriching Word Vectors with Subword Information

P. Bojanowski, E. Grave, A. Joulin and T. Mikolov

Saarland University
Seminar: Embeddings for NLP and IR
Lecturer: Cristina España i Bonet

Aqsa Nazir and Egla Hajdini
May 29th, 2019

Table of contents

Introduction

Introduction

- Continuous word representations, trained on large corpus are useful for many NLP tasks.

Introduction

- Continuous word representations, trained on large corpus are useful for many NLP tasks.

- Models that learn such representations ignores the morphology of the words by assigning a distinct vector to each word.

Introduction

- Continuous word representations, trained on large corpus are useful for many NLP tasks.

- Models that learn such representations ignores the morphology of the words by assigning a distinct vector to each word.

- We present a new approach based on the skipgram model.

Introduction

- In neural network community, Collobert and Weston (2008) proposed to learn word embedding using feed forward neural network by predicting a word based on the two words on the left and two on the right.

Introduction

- In neural network community, Collobert and Weston (2008) proposed to learn word embedding using feed forward neural network by predicting a word based on the two words on the left and two on the right.

- More recently, Mikolov et al. (2013b) proposed simple log-bilinear models to learn continuous representations of words on very large corpora efficiently.

Introduction

Example:

- In French or Spanish, most verbs have more than 40 different inflected forms,

- While the Finnish language has 15 cases for nouns.

Introduction

Example:

- In French or Spanish, most verbs have more than 40 different inflected forms,

- While the Finnish language has 15 cases for nouns.

- These languages contain many word forms that occur rarely (or not at all) in the training corpus, making it difficult to learn good word representations.

Introduction

Here we propose:

- to learn representations for character n-grams,

Introduction

Here we propose:

- to learn representations for character n-grams,
- to represent words as the sum of the n-gram vectors.

Introduction

Here we propose:

- to learn representations for character n-grams,
- to represent words as the sum of the n-gram vectors.
- We introduce an extension of the continuous skipgram model (Mikolov et al., 2013b), which takes into account subword information.

Table of contents

Model

- We present our model to learn word representations while taking into account morphology.

Model

- We present our model to learn word representations while taking into account morphology.

- We model morphology by considering subword units, and representing words by a sum of its character n-grams.

Model

- We present our model to learn word representations while taking into account morphology.

- We model morphology by considering subword units, and representing words by a sum of its character n-grams.

- We will begin by presenting the general framework that we use to train word vectors,

Model

- We present our model to learn word representations while taking into account morphology.
- We model morphology by considering subword units, and representing words by a sum of its character n-grams.
- We will begin by presenting the general framework that we use to train word vectors,
- then present our subword model and eventually describe how we handle the dictionary of character n-grams.

Model: Skipgram Model

Model: Skipgram Model

- Given a word vocabulary of size W , where a word is identified by its index $w \in \{1, ..., W\}$,

Model: Skipgram Model

- Given a word vocabulary of size W , where a word is identified by its index $w \in \{1, ..., W\}$,

- the goal is to learn a vectorial representation for each word *w*.

Model: Skipgram Model

Formally,

- given a large training corpus represented as a sequence of words $w_1, \ldots, w_T$,
- the objective of the skipgram model is to maximize the following log-likelihood:

$$\sum_{t=1}^{T} \sum_{c \in C_t} \log p\left(w_c | w_t\right)$$

Model: Skipgram Model

Formally,

- given a large training corpus represented as a sequence of words $w_1, \ldots, w_T$,
- the objective of the skipgram model is to maximize the following log-likelihood:

$$\sum_{t=1}^{T} \sum_{c \in C_t} \log p\left(w_c | w_t\right)$$

- where the context C_t is the set of indices of words surrounding word w_t.

Model: Skipgram Model

Formally,

- given a large training corpus represented as a sequence of words $w_1, ..., w_T$,
- the objective of the skipgram model is to maximize the following log-likelihood:

$$\sum_{t=1}^{T} \sum_{c \in C_t} \log p\,(w_c | w_t)$$

- where the context C_t is the set of indices of words surrounding word w_t.
- The probability of observing a context word w_c given w_t will be parameterized using the aforementioned word vectors.

Model: Skipgram Model

- For now, let us consider that we are given a scoring function *s* which maps pairs of (word, context) to scores in **R**.

Model: Skipgram Model

- For now, let us consider that we are given a scoring function *s* which maps pairs of (word, context) to scores in **R**.

- One possible choice to define the probability of a context word is the softmax:

$$p(w_c|w_t) = \frac{e^{s(w_t,w_c)}}{\sum_{j=1}^{W} e^{s(w_t,j)}}$$

Model: Skipgram Model

- For the word at position *t* we consider all context words as positive examples and sample negatives at random from the dictionary.

Model: Skipgram Model

- For the word at position *t* we consider all context words as positive examples and sample negatives at random from the dictionary.
- For a chosen context position *c*, using the binary logistic loss, we obtain the following negative log-likelihood:

$$\log\left(1 + e^{-s(w_t, w_c)}\right) + \sum_{n \in N_{t,c}} \log\left(1 + e^{s(w_t, n)}\right)$$

Model: Skipgram Model

- For the word at position *t* we consider all context words as positive examples and sample negatives at random from the dictionary.
- For a chosen context position *c*, using the binary logistic loss, we obtain the following negative log-likelihood:

$$\log\left(1 + e^{-s(w_t, w_c)}\right) + \sum_{n \in N_{t,c}} \log\left(1 + e^{s(w_t, n)}\right)$$

- Where $N_{t,c}$ is a set of negative examples sampled from the vocabulary.

Model: Skipgram Model

- By denoting the logistic loss function *l*: $x \rightarrow \log(1+e^{-x})$, we can rewrite the objective as:

$$\sum_{t=1}^{T}\left[\sum_{c\in C_t} l(s(w_t, w_c)) + \sum_{n\in N_{t,c}} l(-s(w_t, n))\right]$$

Model: Skipgram Model

- A natural parameterization for the scoring function s between a word w_t and a context word w_c is to use word vectors.

Model: Skipgram Model

- A natural parameterization for the scoring function s between a word w_t and a context word w_c is to use word vectors.

- Let us define for each word w in the vocabulary two vectors u_w and v_w in R^d.

Model: Skipgram Model

- A natural parameterization for the scoring function s between a word w_t and a context word w_c is to use word vectors.

- Let us define for each word w in the vocabulary two vectors u_w and v_w in R^d .

- In particular, we have vectors $\boldsymbol{u}_{\boldsymbol{wt}}$ and $\boldsymbol{v}_{\boldsymbol{wt}}$, corresponding, respectively, to words w_t and w_c .

Model: Skipgram Model

- A natural parameterization for the scoring function *s* between a word w_t and a context word w_c is to use word vectors.

- Let us define for each word *w* in the vocabulary two vectors u_w and v_w in R^d.

- In particular, we have vectors $\boldsymbol{u}_{wt}$ and $\boldsymbol{v}_{wt}$, corresponding, respectively, to words w_t and w_c.

- Then the score can be computed as the scalar product between word and context vectors as:

$$s(w_t, w_c) = \boldsymbol{u}^T{}_{wt}\, \boldsymbol{v}_{wc}$$

Table of contents

Model: Subword Model

Model: Subword Model

- Each word w is represented as a bag of character *n-gram*.

Model: Subword Model

- Each word *w* is represented as a bag of character *n-gram*.

- We add special boundary symbols < and > at the beginning and end of words, allowing to distinguish prefixes and suffixes from other character sequences.

Model: Subword Model

- Each word *w* is represented as a bag of character *n-gram*.
- We add special boundary symbols < and > at the beginning and end of words, allowing to distinguish prefixes and suffixes from other character sequences.
- We also include the word *w* itself in the set of its *n-grams*, to learn a representation for each word (in addition to character *n-grams*)

Model: Subword Model

For Example:

- Taking the word *where* and *n = 3*,

- it will be represented by the character *n-grams*:

  <wh, whe, her, ere, re>

- and the special sequence

  <where>.

Model: Subword Model

- Suppose that given a dictionary of *n-grams* of size G.

Model: Subword Model

- Suppose that given a dictionary of *n-grams* of size G.

- Given a word w, let us denote by:
  $G_w \subset \{1, \ldots, G\}$ the set of *n-grams* appearing in w.

Model: Subword Model

- Suppose that given a dictionary of *n-grams* of size G.

- Given a word w, let us denote by:
  $G_w \subset \{1, \ldots, G\}$ the set of *n-grams* appearing in w.

- We associate a vector representation $\boldsymbol{z}_g$ to each *n-gram* g.

Model: Subword Model

- Suppose that given a dictionary of *n-grams* of size G.

- Given a word w , let us denote by:
  $G_w \subset \{1, \ldots, G\}$ the set of *n-grams* appearing in w.

- We associate a vector representation $\boldsymbol{z}_g$ to each *n-gram* g.

- We represent a word by the sum of the vector representations of its *n-grams*.

Model: Subword Model

- We thus obtain the scoring function:

$$s(w, c) = \sum_{g \in G_w} \boldsymbol{z}^T{}_g \, \boldsymbol{v}_c$$

Model: Subword Model

- We thus obtain the scoring function:

$$s(w, c) = \sum_{g \in G_w} \mathbf{z}^T{}_g \, \mathbf{v}_c$$

- This simple model allows sharing the representations across words, thus allowing to learn reliable representation for rare words.

Model: Subword Model

- In order to bound the memory requirements of our model, we use a hashing function that maps *n-grams* to integers in *1* to *K*.

Model: Subword Model

- In order to bound the memory requirements of our model, we use a hashing function that maps *n-grams* to integers in *1* to *K*.

- We hash character sequences using the Fowler-Noll-Vo hashing function (specifically the FNV-1a variant).

Model: Subword Model

- In order to bound the memory requirements of our model, we use a hashing function that maps *n-grams* to integers in *1* to *K*.

- We hash character sequences using the Fowler-Noll-Vo hashing function (specifically the FNV-1a variant).

- We set *K = 2.106*

Model: Subword Model

- In order to bound the memory requirements of our model, we use a hashing function that maps *n-grams* to integers in *1* to *K*.

- We hash character sequences using the Fowler-Noll-Vo hashing function (specifically the FNV-1a variant).

- We set *K = 2.106* below.

- Ultimately, a word is represented by its index in the word dictionary and the set of hashed *n-grams* it contains.

Table of contents

Experimental Setup: Optimization

Experimental Setup: Optimization

- We solve our optimization problem by performing stochastic gradient descent on the negative log likelihood presented before.

Experimental Setup: Optimization

- We solve our optimization problem by performing stochastic gradient descent on the negative log likelihood presented before.

- As in the baseline skipgram model, we use a linear decay of the step size.

Experimental Setup: Optimization

- We solve our optimization problem by performing stochastic gradient descent on the negative log likelihood presented before.

- As in the baseline skipgram model, we use a linear decay of the step size.

- Given a training set containing *T* words and a number of passes over the data equal to *P* , the step size at time *t* is equal to

$$\gamma_0 \left(1 - \frac{t}{TP}\right)$$

- where γ_0 is a fixed parameter.

Experimental Setup: Implementation Details

Experimental Setup: Implementation Details

- For both our model and the baseline experiments, we use the following parameters:

Experimental Setup: Implementation Details

- For both our model and the baseline experiments, we use the following parameters:
    - the word vectors have dimension 300.

Experimental Setup: Implementation Details

- For both our model and the baseline experiments, we use the following parameters:
  - the word vectors have dimension 300.
  - For each positive example, we sample 5 negatives at random.

Experimental Setup: Implementation Details

- For both our model and the baseline experiments, we use the following parameters:
  - the word vectors have dimension 300.
  - For each positive example, we sample 5 negatives at random.
  - We use a context window of size c, and uniformly sample the size *c* between 1 and 5.

Experimental Setup: Implementation Details

- For both our model and the baseline experiments, we use the following parameters:
  - the word vectors have dimension 300.
  - For each positive example, we sample 5 negatives at random.
  - We use a context window of size c, and uniformly sample the size *c* between 1 and 5.
- In order to subsample the most frequent words, we use a rejection threshold of 10^{-4} .

Experimental Setup: Implementation Details

- When building the word dictionary, we keep the words that appear at least 5 times in the training set.

Experimental Setup: Implementation Details

- When building the word dictionary, we keep the words that appear at least 5 times in the training set.

- The step size γ_0 is set to 0.025 for the skipgram baseline and to 0.05 for both our model and the cbow baseline.

Experimental Setup: Datasets

Experimental Setup: Datasets

- We downloaded Wikipedia dumps in nine languages:
  Arabic, Czech, German, English,
  Spanish, French, Italian, Romanian and Russian.

Experimental Setup: Datasets

- We downloaded Wikipedia dumps in nine languages:
  Arabic, Czech, German, English,
  Spanish, French, Italian, Romanian and Russian.

- We normalize the raw Wikipedia data using Matt Mahoney's pre-processing perl script.

Table of contents

Human similarity judgement

- Evaluate quality of representations on the task of word similarity
- Spearman's rank correlation (between human judgement and cosine similarity between vector representations)

Human similarity judgement

- Evaluate quality of representations on the task of word similarity
- Spearman's rank correlation (between human judgement and cosine similarity between vector representations)

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

Human similarity judgement

- Evaluate quality of representations on the task of word similarity
- Spearman's rank correlation (between human judgement and cosine similarity between vector representations)

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

- Method is compared with **cbow** and **skipgram** baselines
- (sisg-) OOV words as null vectors
- (sisg) OOV words as sum of the n-grams vectors

Human similarity judgement

- Advantage of using subword information in the form of character n-grams

		sg	cbow	sisg-	sisg
AR	WS353	51	52	54	**55**
DE	GUR350	61	62	64	**70**
	GUR65	78	78	**81**	**81**
	ZG222	35	38	41	**44**
EN	RW	43	43	46	**47**
	WS353	72	**73**	71	71
ES	WS353	57	58	58	**59**
FR	RG65	70	69	**75**	**75**
RO	WS353	48	52	51	**54**
RU	HJ	59	60	60	**66**

Table1: Correlation between human judgement and similarity scores on word similarity datasets

Human similarity judgement

- Advantage of using subword information in the form of character n-grams
- Effect of character n-grams is more important for (AR, DE, RU)

		sg	cbow	sisg-	sisg
AR	WS353	51	52	54	**55**
DE	GUR350	61	62	64	**70**
	GUR65	78	78	**81**	**81**
	ZG222	35	38	41	**44**
EN	RW	43	43	46	**47**
	WS353	72	**73**	71	71
ES	WS353	57	58	58	**59**
FR	RG65	70	69	**75**	**75**
RO	WS353	48	52	51	**54**
RU	HJ	59	60	60	**66**

Table1: Correlation between human judgement and similarity scores on word similarity datasets

Human similarity judgement

- Advantage of using subword information in the form of character n-grams
- Effect of character n-grams is more important for (AR, DE, RU)
- Outperforms the baselines in datasets composed of rare words

		sg	cbow	sisg-	sisg
AR	WS353	51	52	54	**55**
DE	GUR350	61	62	64	**70**
	GUR65	78	78	**81**	**81**
	ZG222	35	38	41	**44**
EN	RW	43	43	46	**47**
	WS353	72	**73**	71	71
ES	WS353	57	58	58	**59**
FR	RG65	70	69	**75**	**75**
RO	WS353	48	52	51	**54**
RU	HJ	59	60	60	**66**

Table1: Correlation between human judgement and similarity scores on word similarity datasets

Table of contents

Word analogy task

- Word analogy questions: A is to B as C is to **D**
- Four different languages/datasets

Word analogy task

- Word analogy questions: A is to B as C is to **D**
- Four different languages/datasets
- Two tasks:
  - Semantic: meaning of words, phrases
  - Syntactic: grammatical structure

Word analogy task

- Word analogy questions: A is to B as C is to **D**
- Four different languages/datasets
- Two tasks:
  - Semantic: “boy is to girl as man is to woman”
  - Syntactic: “kind is to kindness as slow is to slowness”

Word analogy task

- Word analogy questions: A is to B as C is to **D**
- Four different languages/datasets
- Two tasks:
    - Semantic: “boy is to girls as man is to woman”
    - Syntactic: “kind is to kindness as slow is to slowness”
- Questions that contain OOV words are excluded

Word analogy task

- Morphological information significantly improves the syntactic tasks
- It doesn’t help for semantic questions

		sg	cbow	sisg
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 2: Accuracy on word analogy tasks for Czech, German, English and Italian

Word analogy task

- Morphological information significantly improves the syntactic tasks
- It doesn't help for semantic questions
- When the size of the n-grams is chosen optimally, the semantic analogies degrade less

		sg	cbow	sisg
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 2: Accuracy on word analogy tasks for Czech, German, English and Italian

Word analogy task

- Morphological information significantly improves the syntactic tasks
- It doesn't help for semantic questions
- When the size of the n-grams is chosen optimally, the semantic analogies degrade less
- Improvement is more important for morphologically rich languages, such as Czech and German

		sg	cbow	sisg
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 2: Accuracy on word analogy tasks for Czech, German, English and Italian

Word analogy task

- Morphological information significantly improves the syntactic tasks
- It doesn't help for semantic questions
- When the size of the n-grams is chosen optimally, the semantic analogies degrade less
- Improvement is more important for **morphologically rich languages**, such as Czech and German

		sg	cbow	sisg
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 2: Accuracy on word analogy tasks for Czech, German, English and Italian

A language that encodes a lot of information through morphology

Table of contents

Comparison with morphological representations

- Comparison with previous work on word vector incorporating subword information on word similarity task
    - RNN of Long et al. (2013)
    - Morpheme cbow Qiu et el.(2014)
    - The morphological transformation of Soricut and Och (2015)
- log-bilinear language model introduced by Botha and Blunsom (2014)
- The model is trained on the same data to make results comparable

Comparison with morphological representations

	DE		EN		ES	FR
	GUR350	ZG222	WS353	RW	WS353	RG65
Luong et al. (2013)	-	-	64	34	-	-
Qiu et al. (2014)	-	-	65	33	-	-
Soricut and Och (2015)	64	22	71	42	47	67
sisg	73	43	73	48	54	69
Botha and Blunsom (2014)	56	25	39	30	28	45
sisg	66	34	54	41	49	52

Table 3: Spearman's rank correlation coefficient between human judgement and model scores for different methods using morphology to learn word representations

- It performs well relative to techniques based on subword information
- It outperforms the Soricut and Och (2015) method, which is based on prefix and suffix analysis

Table of contents

Effect of the size of the training data

- The performance of word vectors on the similarity task as a function of the training data
- The **proposed model** and the **cbow** baseline are trained

Effect of the size of the training data

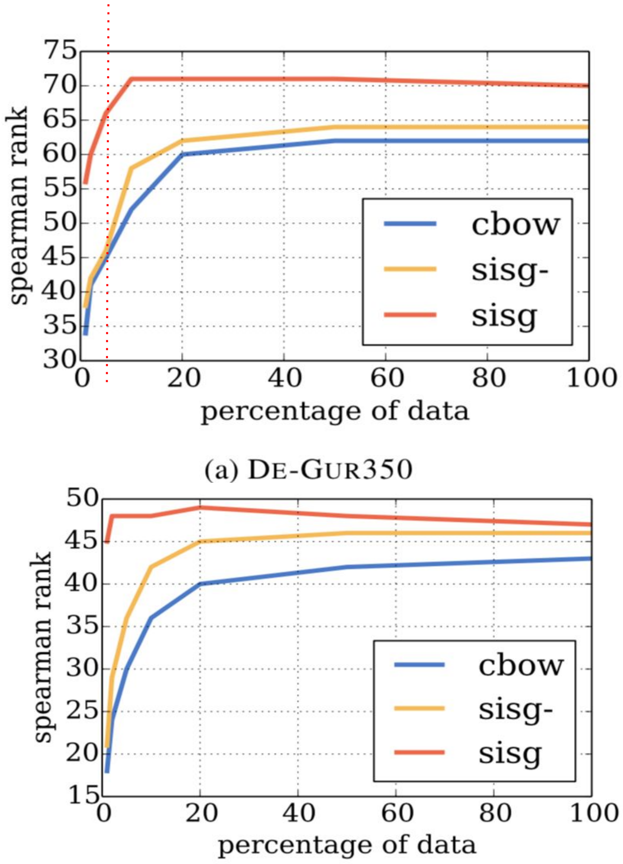
- The performance of word vectors on the similarity task as a function of the training data
- The **proposed model** and the **cbow** baseline are trained
- Portions of Wikipedia of increasing size: 1, 2, 5, 10, 20 and 50 percent of the data

Effect of the size of the training data

- The performance of word vectors on the similarity task as a function of the training data
- The **proposed model** and the **cbow** baseline are trained
- Portions of Wikipedia of increasing size: 1, 2, 5, 10, 20 and 50 percent of the data
- A null vector for OOV words (sisg -)
- A vector by summing the n-gram representations (sisg)

Effect of the size of the training data

- The out-of-vocabulary rate is growing as the dataset shrinks
- The performance of sisg- and cbow necessarily degrades



(a) DE-GUR350

(b) EN-RW

Figure 1: Influence of size of the training data on performance.

Effect of the size of the training data

- The out-of-vocabulary rate is growing as the dataset shrinks
- The performance of sisg- and cbow necessarily degrades
- (sisg) saturates faster

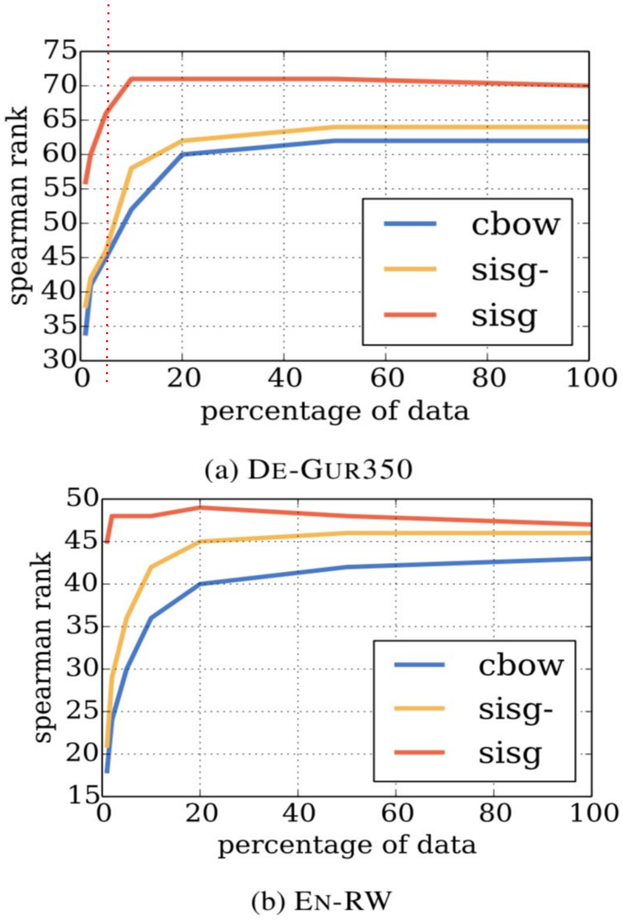
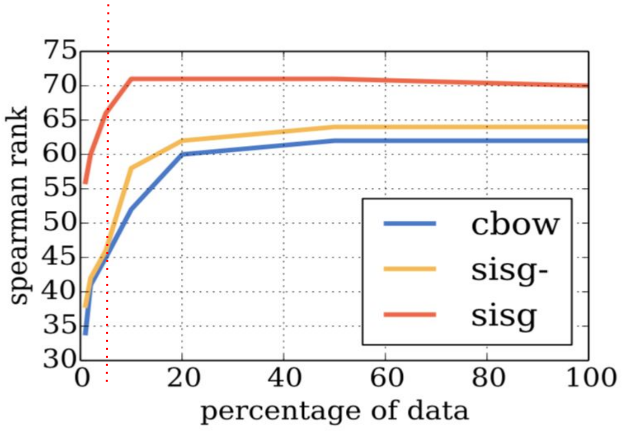


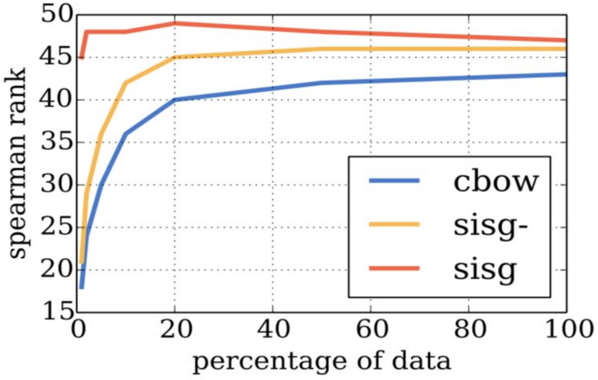
Figure 1: Influence of size of the training data on performance.

Effect of the size of the training data

- The out-of-vocabulary rate is growing as the dataset shrinks
- The performance of sisg- and cbow necessarily degrades.
- (sisg) saturates faster
- Learning from a reduced amount of training data



(a) DE-GUR350



(b) EN-RW

Figure 1: Influence of size of the training data on performance.

Table of contents

Effect of the size of n-grams

- N- grams ranging from 3 to 6 characters
- The optimal choice of length ranges depends on the task and language
- Chosen range provides reasonable amount of subword information
- Important to include long n-grams (i.e German language)
- Larger n-grams help for semantic analogies
- 2-grams are not informative for analogy task

Effect of the size of n-grams

Maximum value of n

Minimum value of n

	2	3	4	5	6
2	57	64	67	69	69
3		65	68	70	70
4			70	70	**71**
5				69	**71**
6					70

(a) DE-GUR350

	2	3	4	5	6
2	59	55	56	59	60
3		60	58	60	62
4			62	62	63
5				64	64
6					**65**

(b) DE Semantic

	2	3	4	5	6
2	45	50	53	54	55
3		51	55	55	**56**
4			54	**56**	**56**
5				**56**	**56**
6					54

(c) DE Syntactic

	2	3	4	5	6
2	41	42	46	47	**48**
3		44	46	**48**	**48**
4			47	**48**	**48**
5				**48**	**48**
6					**48**

(d) EN-RW

	2	3	4	5	6
2	78	76	75	76	76
3		78	77	78	77
4			79	79	79
5				**80**	79
6					**80**

(e) EN Semantic

	2	3	4	5	6
2	70	71	73	74	73
3		72	74	**75**	74
4			74	**75**	**75**
5				74	74
6					72

(f) EN Syntactic

Table 4: Study of the effect of sizes of n-grams considered on performance. Word vectors are computed using character n-grams with *n* in *{i, ….,j}*. Performance for various values of i and j is reported

Table of contents

Language modeling

- Evaluation of word vectors obtained on a language modeling task
- five languages (Cs, De, Es, Fr, Ru)
- RNN with 650 LSTM units
- Dropout regularization (p = 0.5)
- Adagrad algorithm (γ = 0.1)
- Perplexity

$$PP = P(w_1, w_2, \ldots w_n)^{-\frac{1}{N}}$$

Language modeling

- Using word representations trained with subword information outperforms the plain skipgram model
- Improvement is most significant for morphologically rich Slavic languages

	Cs	De	Es	Fr	Ru
Vocab. size	46k	37k	27k	25k	63k
CLBL	465	296	200	225	304
CANLM	371	239	165	184	261
LSTM	366	222	157	173	262
sg	339	216	150	162	237
sisg	**312**	**206**	**145**	**159**	**206**

Table 5: Test perplexity on the language modeling task, for 5 different languages

Table of contents

Nearest Neighbours

- Nearest neighbours according to cosine similarity

query	tiling	tech-rich	english-born	micromanaging	eateries	dendritic
sisg	tile flooring	tech-dominated tech-heavy	british-born polish-born	micromanage micromanaged	restaurants eaterie	dendrite dendrites
sg	bookcases built-ins	technology-heavy .ixic	most-capped ex-scotland	defang internalise	restaurants delis	epithelial p53

Table 6: Nearest neighbors of rare words using proposed representations and skipgram

Nearest Neighbours

- Nearest neighbours according to cosine similarity
- NN are better for complex, technical and infrequent words

query	tiling	tech-rich	english-born	micromanaging	eateries	dendritic
sisg	tile flooring	tech-dominated tech-heavy	british-born polish-born	micromanage micromanaged	restaurants eaterie	dendrite dendrites
sg	bookcases built-ins	technology-heavy .ixic	most-capped ex-scotland	defang internalise	restaurants delis	epithelial p53

Table 6: Nearest neighbors of rare words using proposed representations and skipgram

Table of contents

1. Introduction
2. Model
3. Experimental Setup
4. Results
5. Qualitative analysis
    a. Nearest neighbours
    b. Character n-grams and morphemes
    c. Word similarity for OOV words
6. Conclusion

Character n-grams and morphemes

- Evaluate whether or not the most important n-grams correspond to morphemes

$$u_w = \sum_{g \in G_w} z_g$$

$$u_{w/g} = \sum_{g' \in G - \{g\}} z'_g$$

- Rank n-grams by increasing order of cosine between **u_w** and **$u_{w/g}$**

Character n-grams and morphemes

- The most important n-grams correspond to valid morphemes
- Separation of compound nouns into morphemes
- N-grams may correspond to affixes in some words

	word	n-grams		
DE	autofahrer	fahr	fahrer	auto
	freundeskreis	kreis	kreis>	<freun
	grundwort	wort	wort>	grund
	sprachschule	schul	hschul	sprach
	tageslicht	licht	gesl	tages
EN	anarchy	chy	<anar	narchy
	monarchy	monarc	chy	<monar
	kindness	ness>	ness	kind
	politeness	polite	ness>	eness>
	unlucky	<un	cky>	nlucky
	lifetime	life	<life	time
	starfish	fish	fish>	star
	submarine	marine	sub	marin
	transform	trans	<trans	form
FR	finirais	ais>	nir	fini
	finissent	ent>	finiss	<finis
	finissions	ions>	finiss	sions>

Table 7: Illustration of most important character n- grams for selected words in three languages

Table of contents

Word similarity for OOV words

- Assess the quality of representation of OOV words
- Which n-grams match best for OOV words
- Select few words from English RW similarity dataset
- Pairs such that one of two words is not in training vocabulary
- Display the cosine similarity between each pair of n-grams that appear in the words
- Models trained on 1% of the Wikipedia data

Word similarity for OOV words

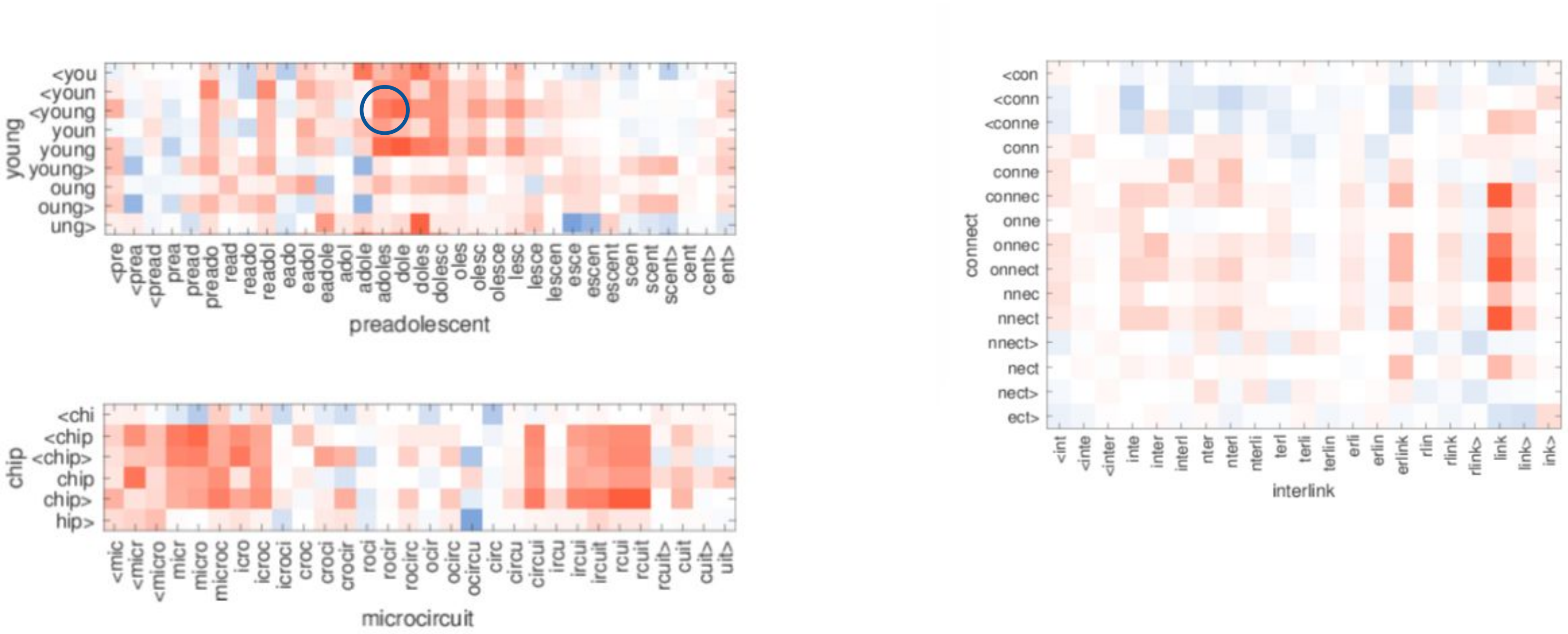


Figure 2: Illustration of the similarity between character n-grams in out-of-vocabulary words.

Word similarity for OOV words

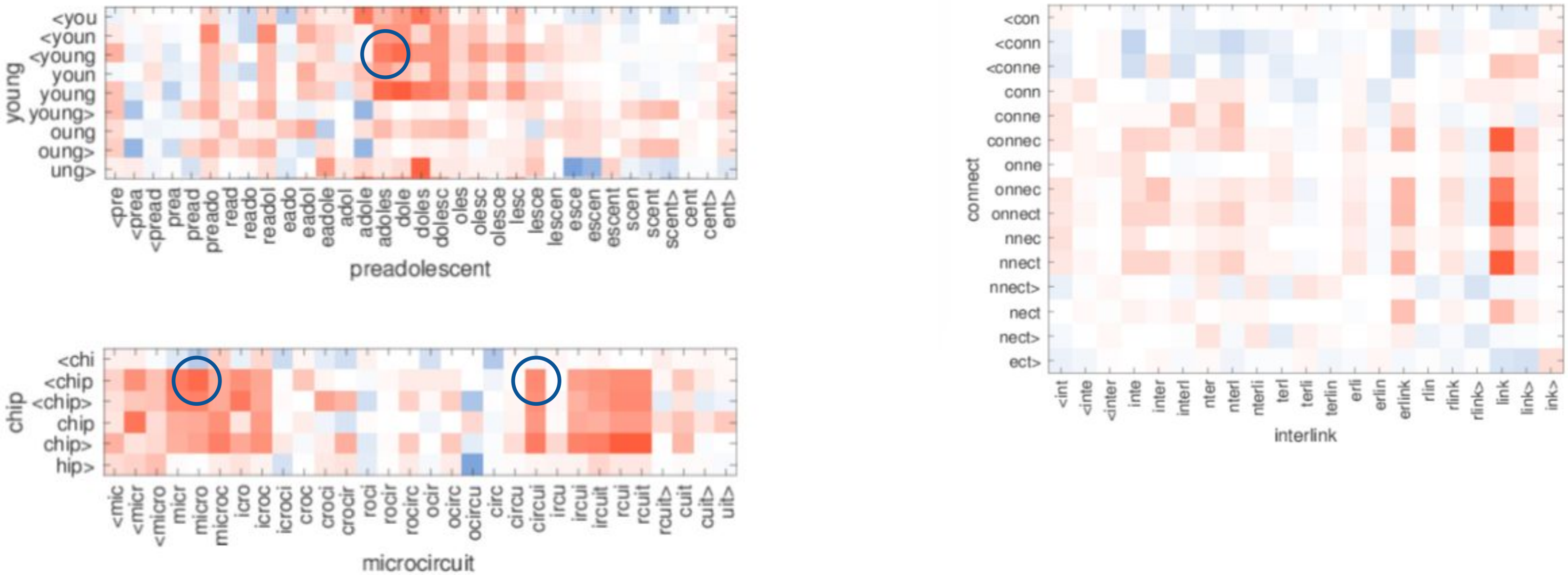


Figure 2: Illustration of the similarity between character n-grams in out-of-vocabulary words.

Word similarity for OOV words

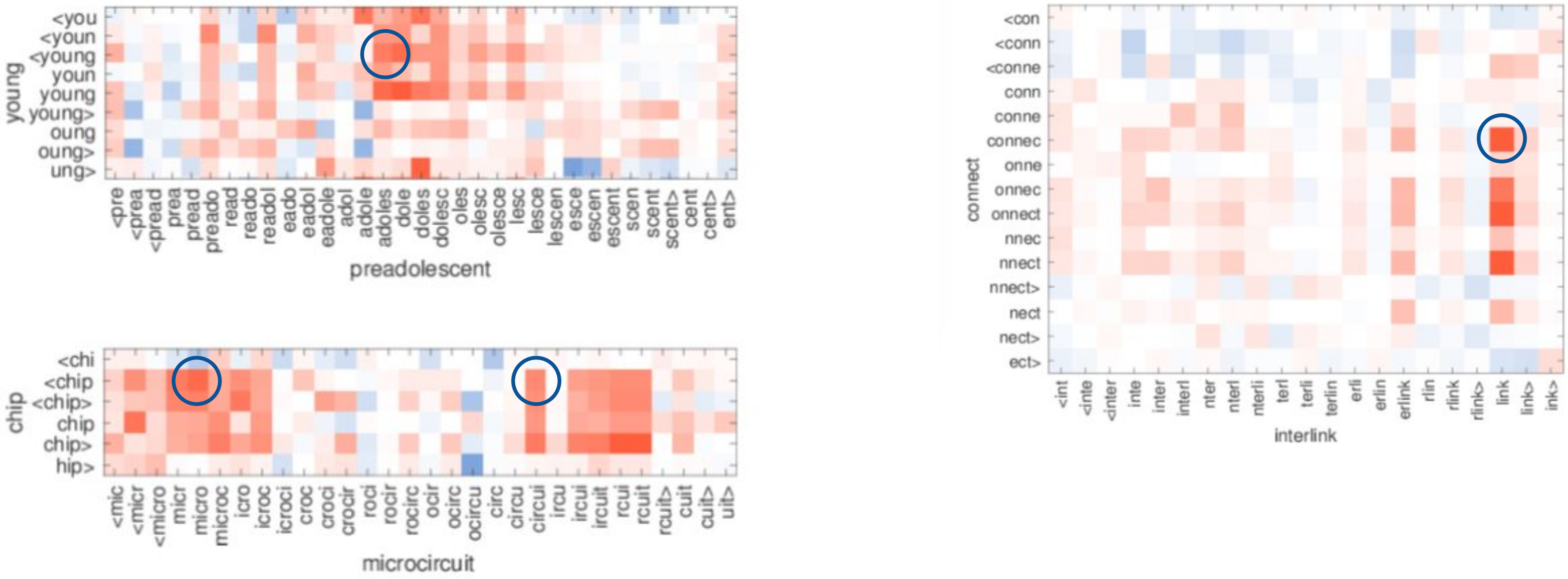


Figure 2: Illustration of the similarity between character n-grams in out-of-vocabulary words.

Word similarity for OOV words

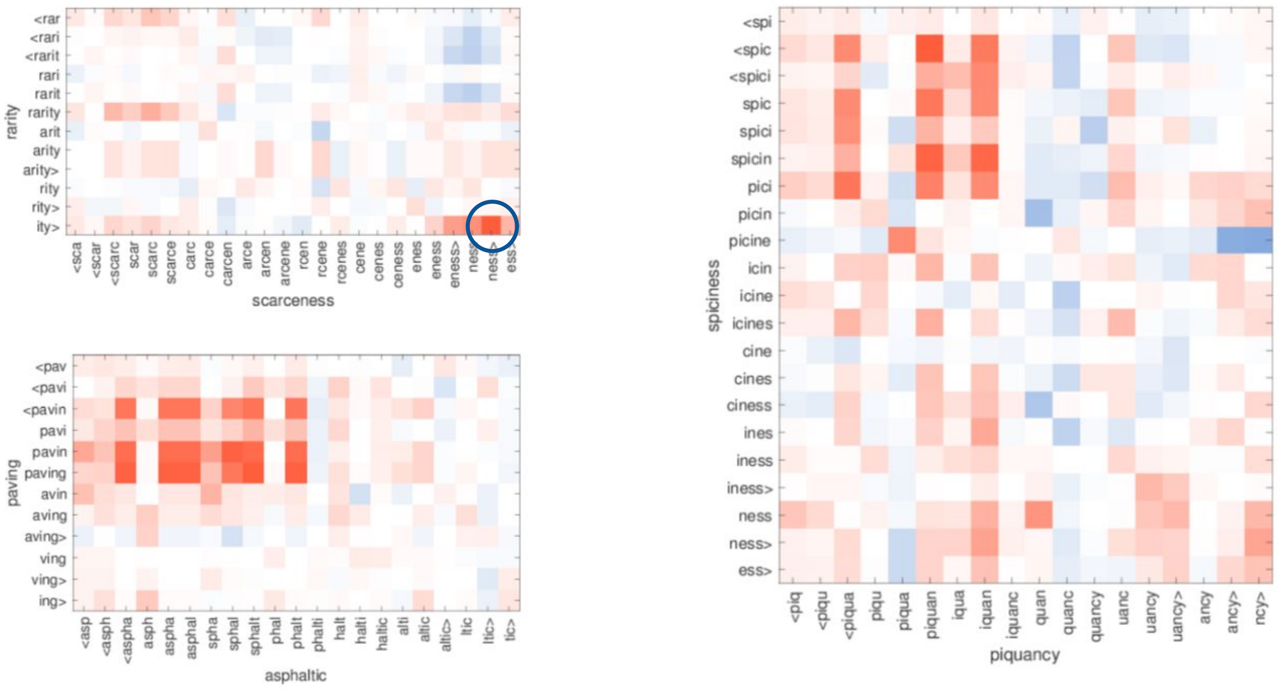


Figure 3: Illustration of the similarity between character n-grams in out-of-vocabulary words.

Word similarity for OOV words

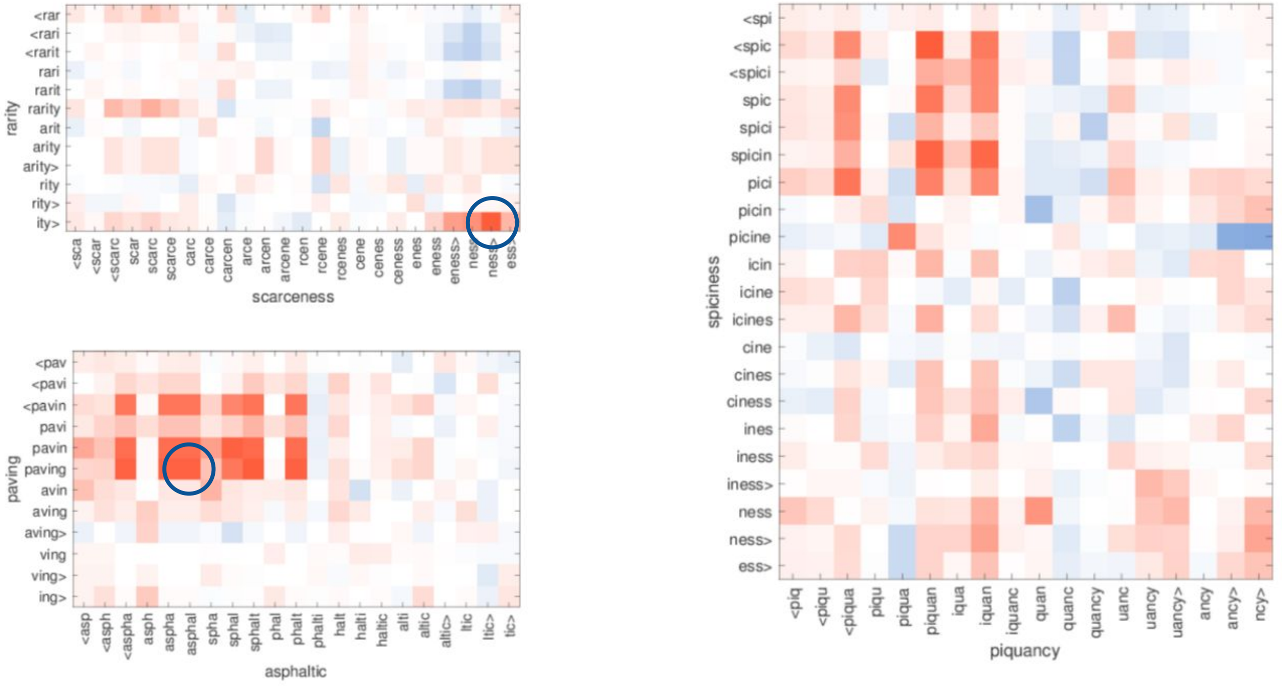


Figure 3: Illustration of the similarity between character n-grams in out-of-vocabulary words.

Word similarity for OOV words

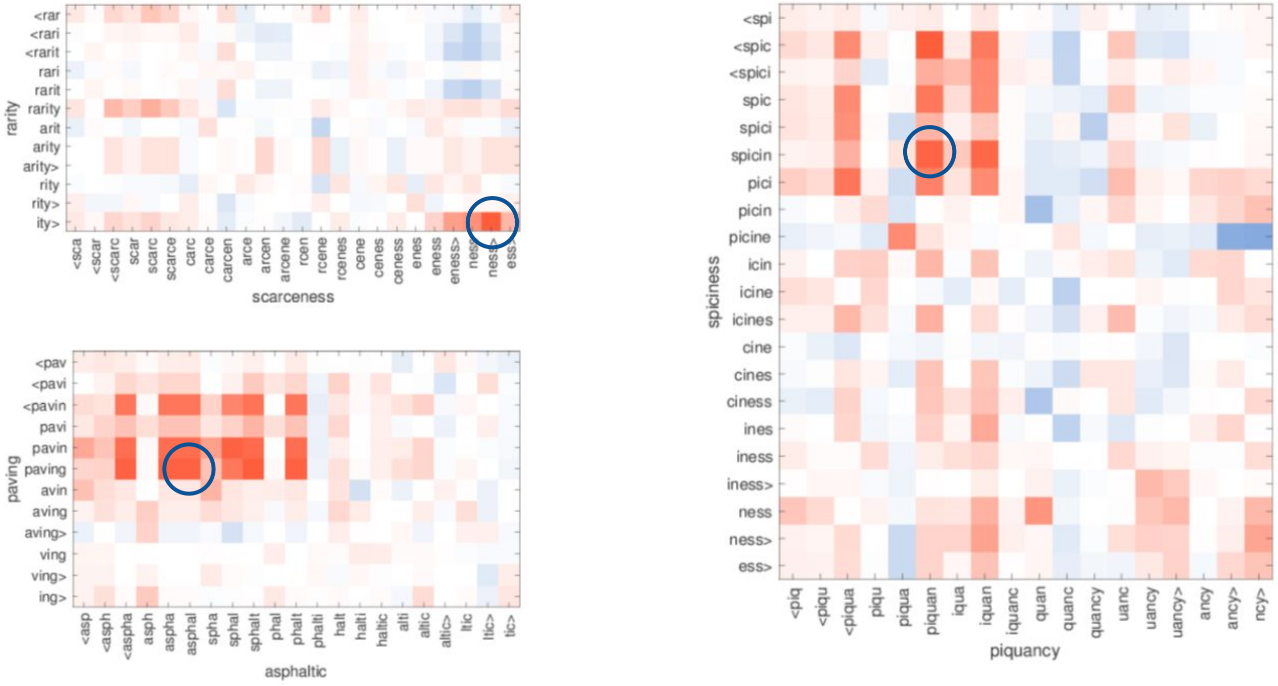


Figure 3: Illustration of the similarity between character n-grams in out-of-vocabulary words.

Table of contents

Conclusions

- A simple method to learn word representations by taking into account subword information

Conclusions

- A simple method to learn word representations by taking into account subword information
- Proposed model trains fast and does not require any preprocessing or super-vision

Conclusions

- A simple method to learn word representations by taking into account subword information
- Proposed model trains fast and does not require any preprocessing or super-vision
- It outperforms baselines that do not take into account subword information

Conclusions

- A simple method to learn word representations by taking into account subword information
- Proposed model trains fast and does not require any preprocessing or supervision
- It outperforms baselines that do not take into account subword information
- As well as methods relying on morphological analysis

Questions?

References

[1] P. Bojanowski*, E. Grave*, A. Joulin, T. Mikolov, *Enriching Word Vectors with Subword Information*