NUS National University of Singapore | School of Computing

Embeddings

CS4248 Natural Language Processing

Week 06

Liangming PAN, Reza QORIB and Min-Yen KAN

6

Recap of Week 05

Generative vs. Discriminative Classifiers

Classification with Logistic Regression
and a Runthrough

Cross Entropy

Stochastic Gradient Descent

LR as a Probabilistic ML Classifier

Regularization

XOR

Neural Networks

Week 06 Agenda

One-hot Representation

Co-occurrence Vectors

Word Embeddings

Word2Vec: CBOW and Skip-gram

Properties of Embeddings

One-hot Representation

NUS National University of Singapore | School of Computing

What are various ways to represent the meaning of a word?

Representing words as discrete symbols

In traditional NLP, we regard words as discrete symbols:
hotel, *conference*, *b&b* – a localist representation

Words are represented by one-hot vectors:

Means one '1' and rest '0's

b&b = [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]

hotel = [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]

Where the vector dimension = $|V|$ # of words in the vocabulary
(20,000 to 50,000 dictionary lemmas, or 500K inflected tokens)

Problem with words as discrete symbols

Example: in Web search, if the user searches for *singapore hotel*, we would like to match documents containing *singapore b&b*.

b&b = [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]

hotel = [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]

But we can see that the two vectors are orthogonal. Oh no!

There is no natural notion of similarity for one-hot vectors!

Distributional Hypothesis

"The meaning of a word is its use in the language"

[Wittgenstein, *Philosophical Investigations*, n.d.]

"You shall know a word by the company it keeps"

[Firth, 1957]

"If A and B have almost identical environments, we say that they are synonyms"

[Harris, 1954]

…by the company it keeps…

When a word w appears in a text, its context is the set of words that appear nearby (within a fixed-size window).

Use the many contexts of w to build up representation for w.

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

These context words will represent banking

Works for OOV too

Remember the *gompies*?

All <u>gompies</u> are biff and luff voomly.

M'moon is a cramy <u>gompy</u>, she is the biffiest and luffs voomly

Co-occurrence Vectors

NUS National University of Singapore | School of Computing

Back to the term–document matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Term–Document Matrix

	As You Like It	*Twelfth Night*	*Julius Caesar*	*Henry V*
battle	1	0	7	17
soldier	2	80	62	89
fool	36	58	1	4
clown	20	15	2	3

How about a term–context matrix?

Term–Context Matrix (Word–Word Matrix)

	knife	dog	sword	love	like
knife	0	1	6	5	5
dog	1	0	5	5	5
sword	6	5	0	5	5
love	5	5	5	0	5
like	5	5	5	5	2

Word Embeddings

NUS National University of Singapore | School of Computing

Sparse versus Dense Vectors

PPMI vectors are

- Long (length |V|=20,000 to 50,000)
- Sparse (most elements are zero)

Alternative: learn vectors which are

- Short (length 200-1000)
- Dense (most elements are non-zero)

Why Dense Vectors?

Short vectors may be easier to use as features (less weights to tune; avoid overfitting)

Dense vectors may generalize better than storing explicit counts

They may do better at capturing synonymy

- *car* and *automobile* are synonyms but are represented as distinct dimensions; This fails to capture similarity between a word with *car* as a neighbor and a word with *automobile* as a neighbor.

In practice, they work better

Intuition of Word Embeddings

- Hypothesis: Words that are semantically similar often have same surrounding words. (Distributional Hypothesis)
- Goal: We want words that are semantically similar to have close word vectors.

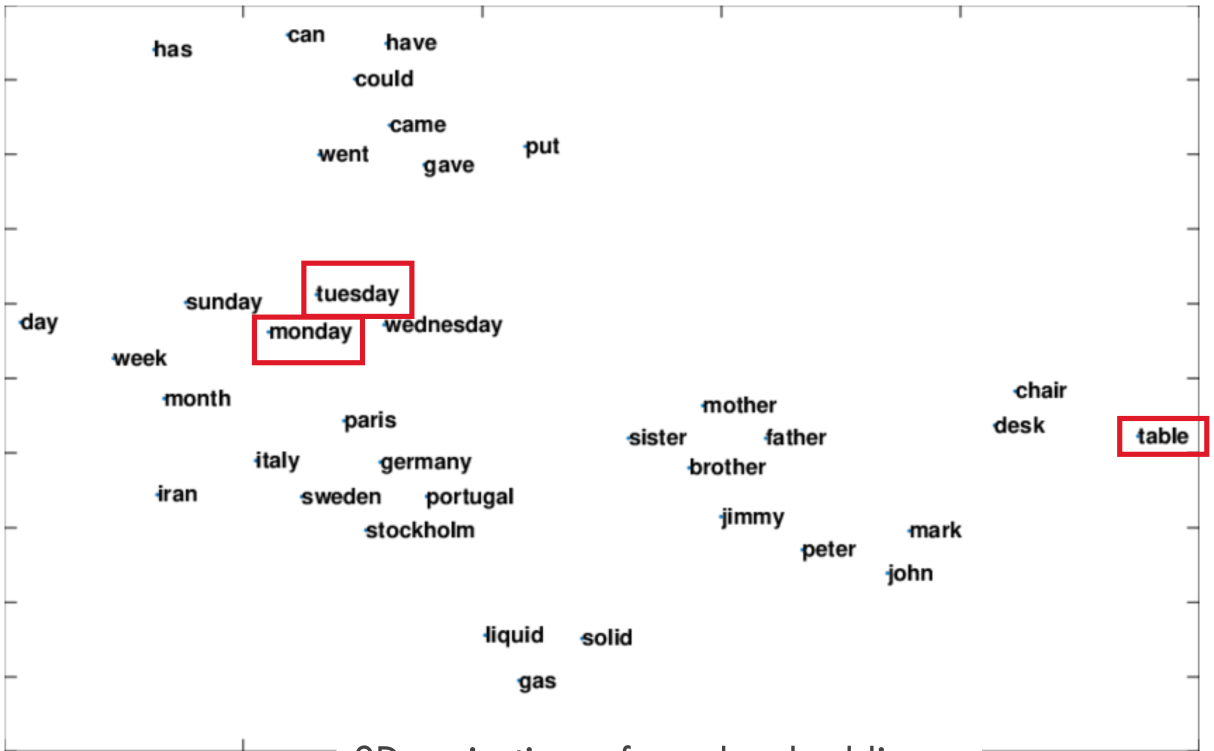
- Intermediate Goal: We need to make the words that have the same surrounding words to have close word vectors.

$$tuesday = \begin{bmatrix} 0.75 \\ -0.18 \\ 0.34 \\ \vdots \\ 0.25 \end{bmatrix}$$

$$monday = \begin{bmatrix} \\ \vdots \\ \\ \end{bmatrix}$$

$$table = \begin{bmatrix} \\ \vdots \\ \\ \end{bmatrix}$$

NUS National University of Singapore | School of Computing



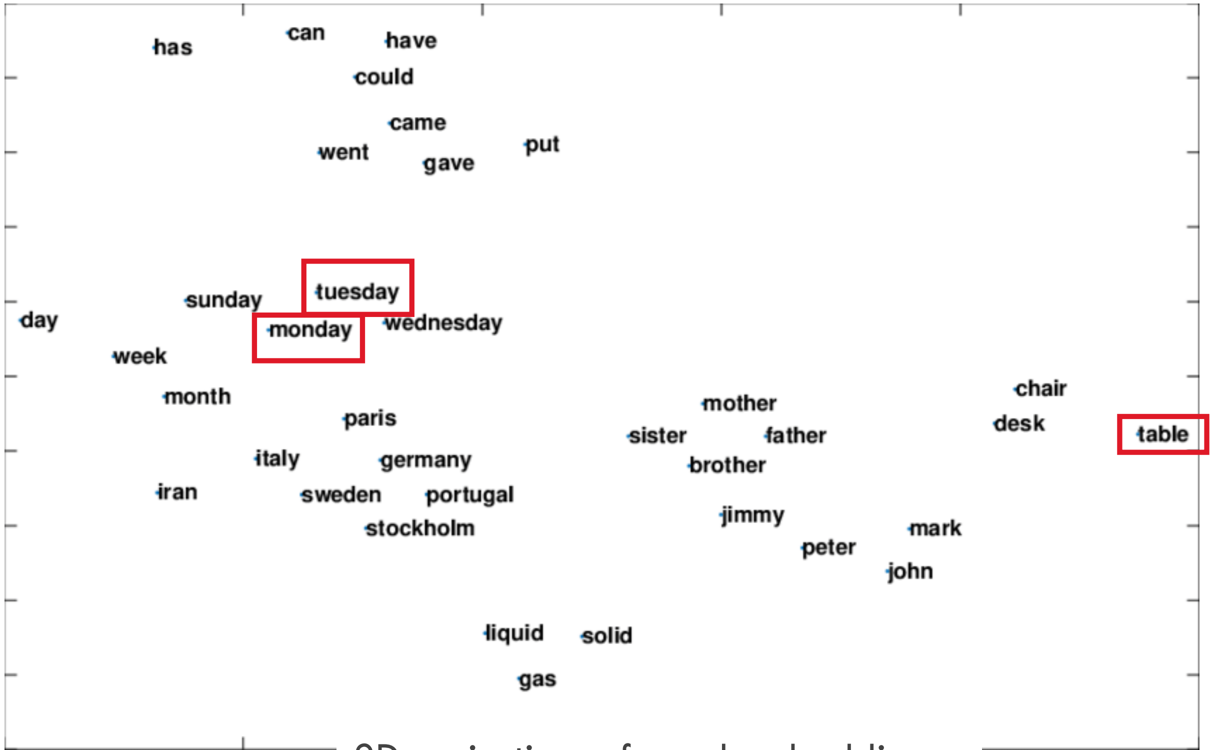
2D projections of word embeddings

Diagram Credits: https://www.researchgate.net/publication/327074728_Principal_Word_Vectors

$$tuesday = \begin{bmatrix} 0.75 \\ -0.18 \\ 0.34 \\ \vdots \\ 0.25 \end{bmatrix}$$

$$monday = \begin{bmatrix} 0.78 \\ -0.2 \\ 0.34 \\ \vdots \\ 0.21 \end{bmatrix}$$

$$table = \begin{bmatrix} -0.3 \\ 0.12 \\ 0.51 \\ \vdots \\ 0.02 \end{bmatrix}$$



2D projections of word embeddings

Diagram Credits: https://www.researchgate.net/publication/327074728_Principal_Word_Vectors

Making Word Embeddings

- Use neutral network to train a self-supervised task.
- Use the weight matrix as the word vector representation.
- Common techniques for embedding words: CBOW, Skip-Gram, and GLoVe.
- Do not care about the actual model, we only take the embedding weight.

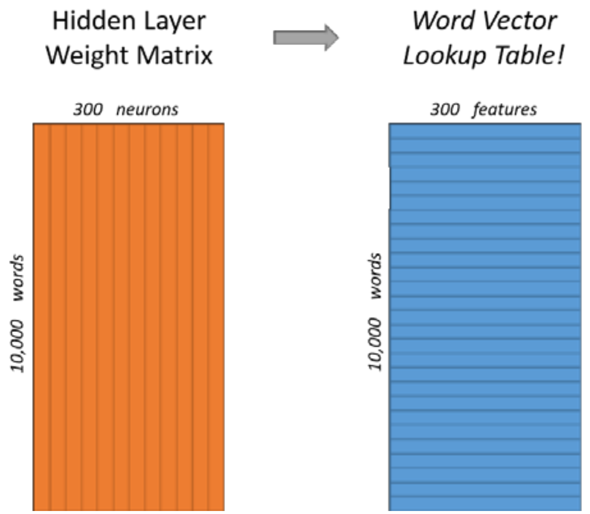


Diagram Credits: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Word2Vec: CBOW & Skip-Gram

Intuition

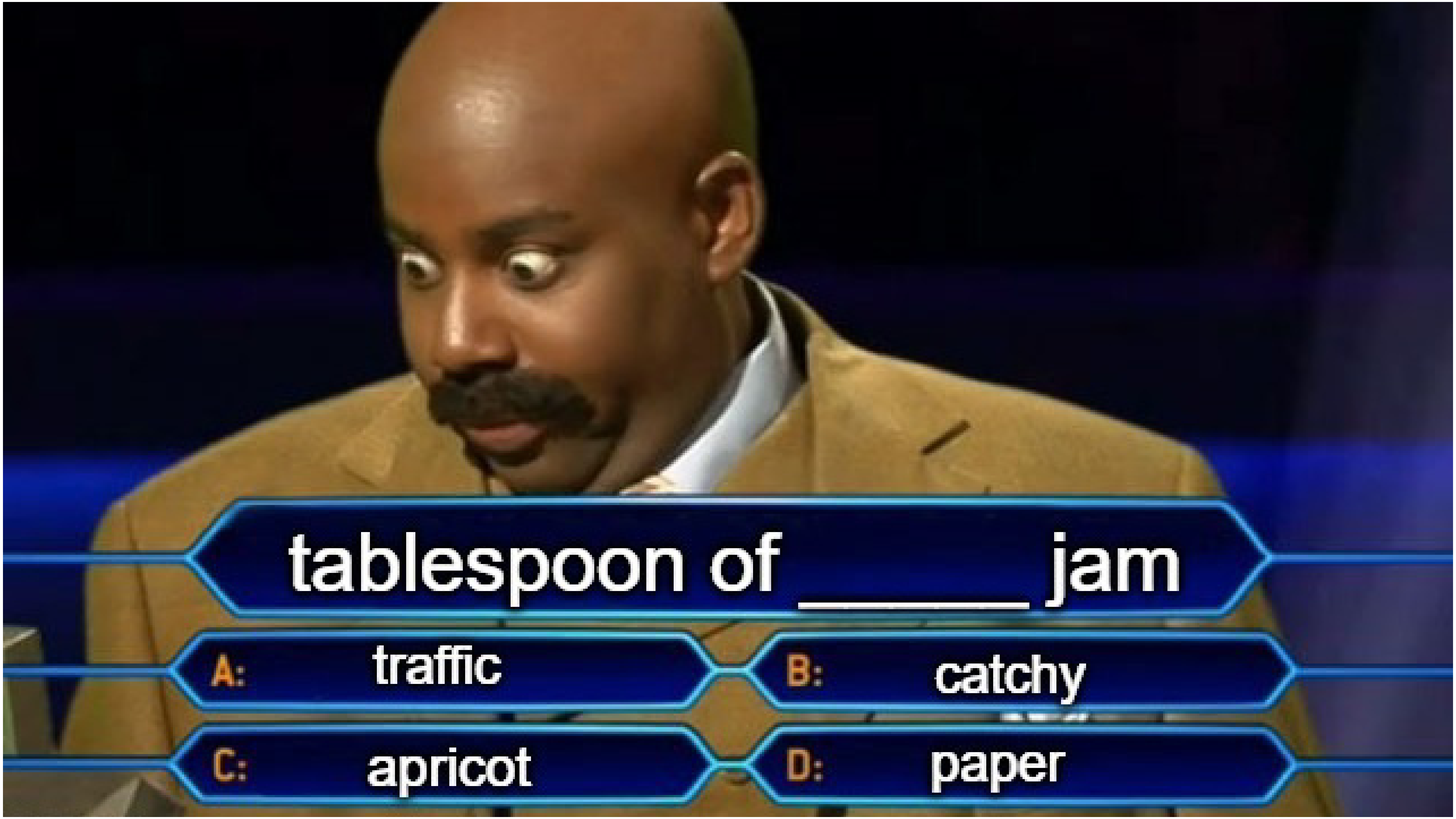
Words that are semantically similar often have same surrounding words. (Distributional Hypothesis)

Continuous Bag of Words (CBOW):

- Train a model to predict a word from the surrounding words.

Skip-Gram:

- Train a model to predict the surrounding words from a word.


tablespoon of ______ jam
A: traffic
B: catchy
C: apricot
D: paper

Predicting a word from context

at

apricot

glass

bottle

cars

juice

honey

strawberry

table

bowl

tablespoon of ______ jam

Predicting a word from context

at

apricot

glass

cars

bottle

juice

honey

strawberry

table

bowl

tablespoon of ______ jam

Predicting a word from context

at

apricot

glass

cars

bottle

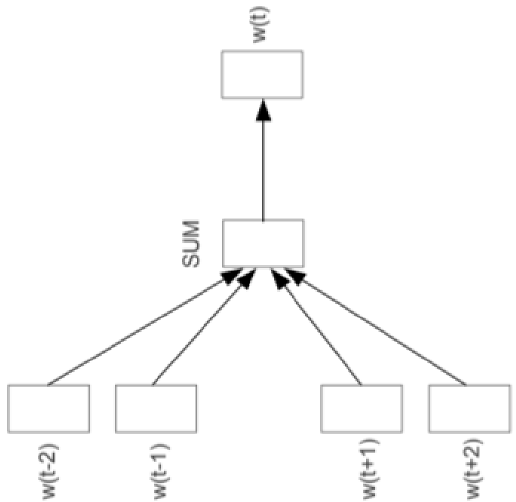
juice

honey

strawberry

table

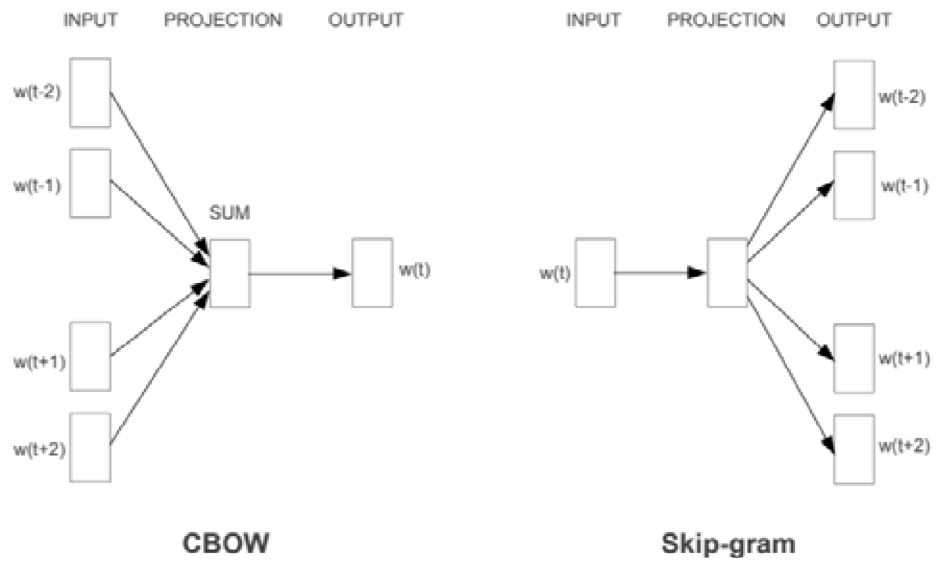
bowl



tablespoon of ______ jam

CBOW and Skip-Gram

- CBOW → given a context, predict the word.
- Skip-gram → given a word, predict the context.



Predicting surrounding words

NUS National University of Singapore | School of Computing

at

of

glass

cars

wall

juice

tablespoon

jam

table

apron

_____________ ____ apricot _____

Photo by [Philippe Gonyea](#) on [Unsplash](#)

Predicting surrounding words

at

of

glass

cars

wall

juice

tablespoon

jam

table

apron

_____________ ____ apricot _____

Photo by Philippe Gonyea on Unsplash

Predicting surrounding words

at

of

glass

cars

wall

juice

tablespoon

jam

table

apron

_____________ ___ apricot _____

Photo by Philippe Gonyea on Unsplash

Predicting surrounding words

glass

at

of

cars

wall

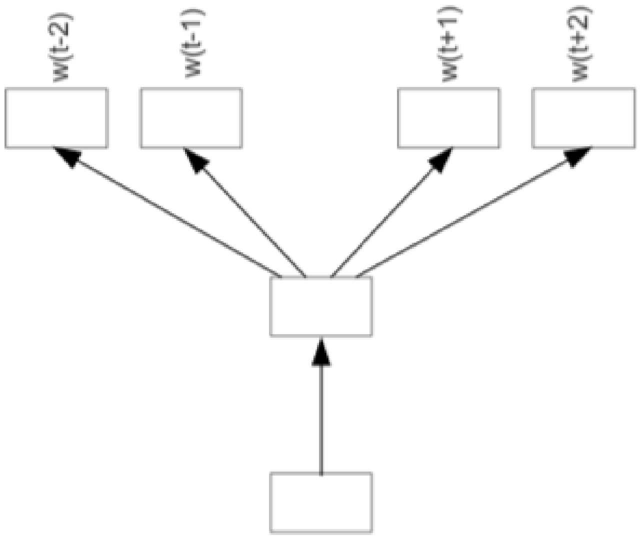
juice

tablespoon

jam

table

apron



____________ ____ apricot _____

Photo by Philippe Gonyea on Unsplash

Training Objective

Optimize the weight so that word and its context have close vector representation

$$\mathcal{L}(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\neq 0}} \log P\left(w_{t+j}\middle|w\right)$$

Training Objective

Optimize the weight so that word and its context have close vector representation

$$\mathcal{L}(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\neq 0}} \log P\left(w_{t+j}\middle|w\right)$$

$$P\left(w_{t+j}\middle|w_t\right) = \frac{\exp\left(u_{w_{t+j}} \cdot v_{w_t}\right)}{\sum_{w'} \exp\left(u_{w'} \cdot v_{w_t}\right)}$$

Training Objective

Optimize the weight so that word and its context have close vector representation

$$\mathcal{L}(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m\\ j\neq 0}} \log P\left(w_{t+j}\middle|w\right)$$

more similar = higher dot product
= larger probability

$$P\left(w_{t+j}\middle|w_t\right) = \frac{\exp\left(u_{w_{t+j}} \cdot v_{w_t}\right)}{\sum_{w'} \exp\left(u_{w'} \cdot v_{w_t}\right)}$$

Training Objective

Optimize the weight so that word and its context have close vector representation

$$\mathcal{L}(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m\\ j\neq 0}} \log P\left(w_{t+j}\middle|w\right)$$

more similar = higher dot product
= larger probability

$$P\left(w_{t+j}\middle|w_t\right) = \frac{\exp\left(u_{w_{t+j}} \cdot v_{w_t}\right)}{\sum_{w'} \exp\left(u_{w'} \cdot v_{w_t}\right)}$$

Normalize over entire vocabulary to give probability distribution

Skip-Gram with Negative Sampling (SGNS)

For each positive sample, create k negative samples from random word, c_{neg}

example with $k = 2$

positive examples +

w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

w	c_{neg}	w	c_{neg}
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

Slide Credits: Dan Jurafsky (Stanford)

SGNS Objective

$$P\left(w_{t+j} \middle| w_t\right) = \frac{\exp\left(u_{w_{t+j}} \cdot v_{w_t}\right)}{\sum_{w'} \exp\left(u_{w'} \cdot v_{w_t}\right)}$$

$$\mathcal{L}(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \log p\left(w_{t+j} \middle| w\right)$$

SGNS Objective

$$\cancel{P(w_{t+j}|w_t) = \frac{\exp\left(u_{w_{t+j}} \cdot v_{w_t}\right)}{\sum_{w'} \exp\left(u_{w'} \cdot v_{w_t}\right)}} \Rightarrow P(+|c_{pos}, w)$$

$$\cancel{\mathcal{L}(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m \\ j\ne 0}} \log P\left(w_{t+j}|w\right)} \Rightarrow \text{Contrastive Loss}$$

SGNS Objective

$$P(+|c_{pos}, w) = \frac{1}{1+\exp(-c_{pos} \circ w)}$$

$$c_{pos} = \{w_{t-j}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+j}\}$$

context or surrounding words

SGNS Objective

$$P(+|c_{pos}, w) = \sigma(c_{pos} \circ w) = \frac{1}{1+\exp(-c_{pos} \circ w)}$$

$$c_{pos} = \{w_{t-j}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+j}\}$$

context / surrounding words

$$\mathcal{L}(\theta) = -\log\left[P(+|c_{pos}, w) \prod_{i=1}^{k} P(-|c_{neg}, w)\right]$$

SGNS Objective

$$P\left(+\middle|c_{pos}, w\right) = \sigma(c_{pos} \circ w) = \frac{1}{1+\exp(-c_{pos} \circ w)}$$

$c_{pos} = \{w_{t-j}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+j}\}$

context / surrounding words

$$\mathcal{L}(\theta) = -\log\left[P\left(+\middle|c_{pos}, w\right)\prod_{i=1}^{k} P\left(-\middle|c_{neg}, w\right)\right]$$

Remember, $P\left(-\middle|c_{neg}, w\right) = 1 - P\left(+\middle|c_{neg}, w\right)$

$$= -\left[\log P\left(+\middle|c_{pos}, w\right) + \sum_{i=1}^{k} \log\left(1 - P\left(+\middle|c_{neg}, w\right)\right)\right]$$

SGNS Objective

NUS National University of Singapore | School of Computing

$$P(+|c_{pos}, w) = \sigma(c_{pos} \circ w) = \frac{1}{1+\exp(-c_{pos} \circ w)}$$

$c_{pos} = \{w_{t-j}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+j}\}$

context / surrounding words

$$\mathcal{L}(\theta) = -\log\left[P(+|c_{pos}, w) \prod_{i=1}^{k} P(-|c_{neg}, w)\right]$$

Remember, $P(-|c_{neg}, w) = 1 - P(+|c_{neg}, w)$

$$= -\left[\log P(+|c_{pos}, w) + \sum_{i=1}^{k} \log\left(1 - P(+|c_{neg}, w)\right)\right]$$

$$= -\left[\log \sigma(c_{pos} \circ w) + \sum_{i=1}^{k} \log \sigma(-c_{neg} \circ w)\right]$$

Negative Sampling

Negative samples chosen according to their (α-)weighted **unigram frequency**

- Common practice $\alpha < 0.75$
- $\alpha < 1$ gives more chance to rare words

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_{w'} count(w')^\alpha}$$

NUS National University of Singapore | School of Computing

Negative Sampling

Word	Unigram frequency
the	0.99
durian	0.01

→

Weighted $\alpha = 0.75$
0.97
0.03

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_{w'} count(w')^\alpha}$$

$$P_\alpha(a) = \frac{.99^{.75}}{.99^{.75} + .01^{.75}} = .97$$

$$P_\alpha(b) = \frac{.01^{.75}}{.99^{.75} + .01^{.75}} = .03$$

Slide Credits: Dan Jurafsky (Stanford)

Model training

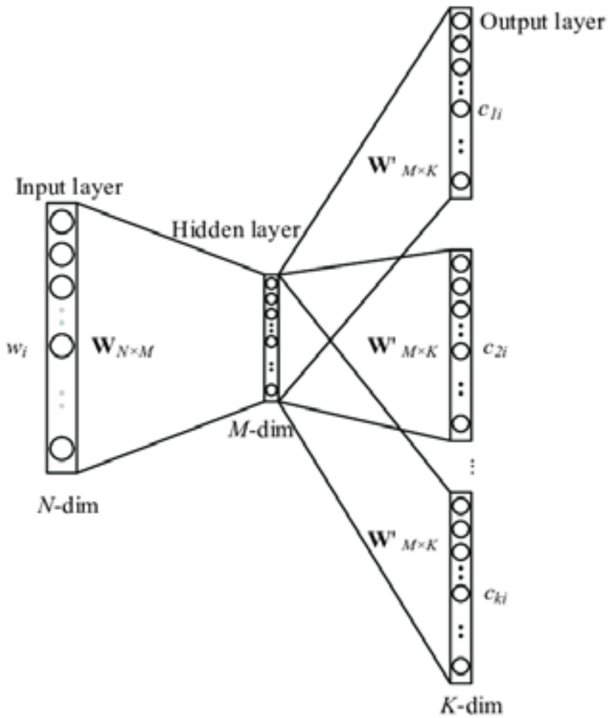
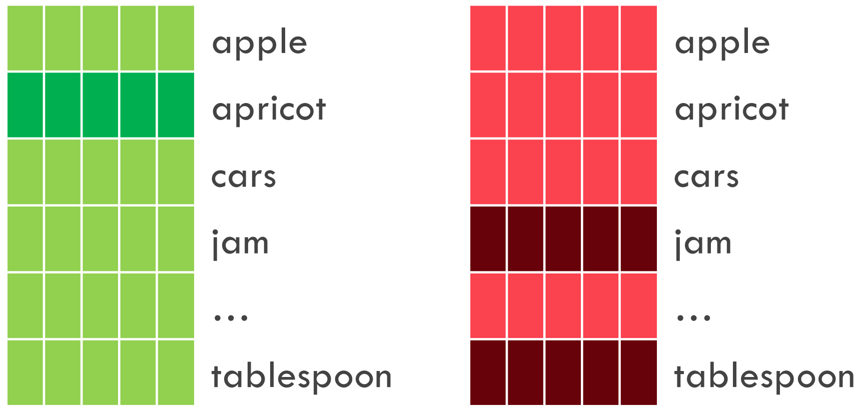


Diagram Credits: Patent Keyword Extraction Algorithm Based on Distributed Representation for Patent Classification (2018)



word	context	target
apricot	tablespoon	1
apricot	cars	0
apricot	jam	1

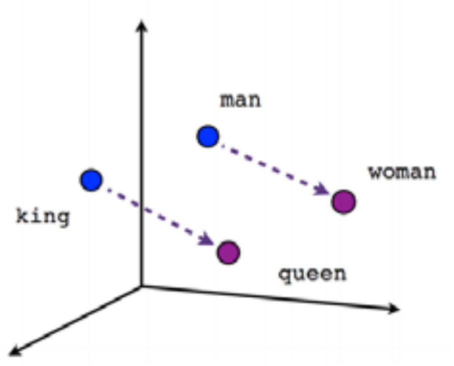
Properties of Embeddings

NUS National University of Singapore | School of Computing

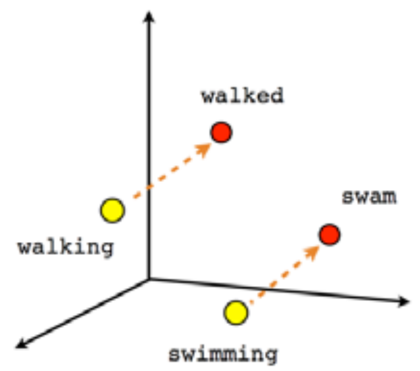
Vector differences yield semantic relationships!

$\vec{v}(\textit{king}) - \vec{v}(\textit{man}) + \vec{v}(\textit{woman}) \approx \vec{v}\ (\textit{queen})$

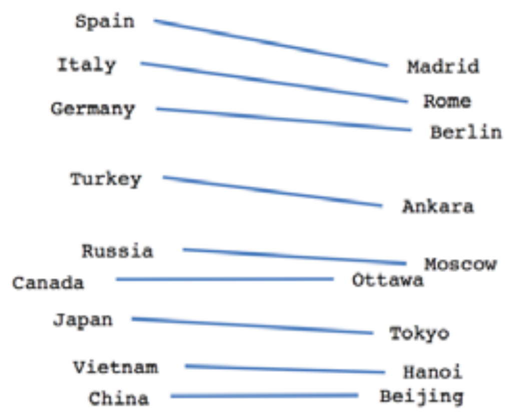
$\vec{v}(\textit{Paris}) - \vec{v}(\textit{France}) + \vec{v}(\textit{Italy}) \approx \vec{v}(\textit{Rome})$



Male-Female



Verb tense



Country-Capital

Modeling semantic similarity in GloVe

Global Vectors (GloVe): represent the probabilities as *ratios* of their co-occurrences.

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

$$F(w_i, w_j, w_k) = \frac{P_{ij}}{P_{jk}}$$

Nearest Neighbors

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

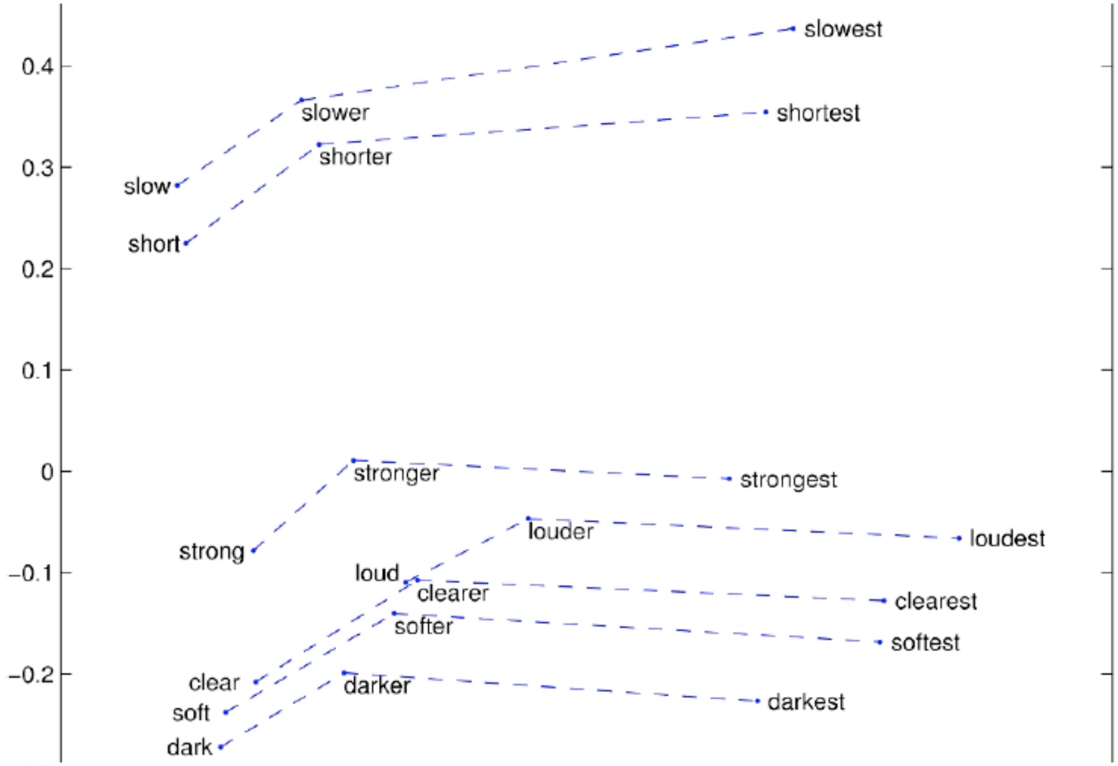
3. litoria

4. leptodactylidae

5. rana

7. eleutherodactylus

Linear substructures



Also works for crosslingual cases

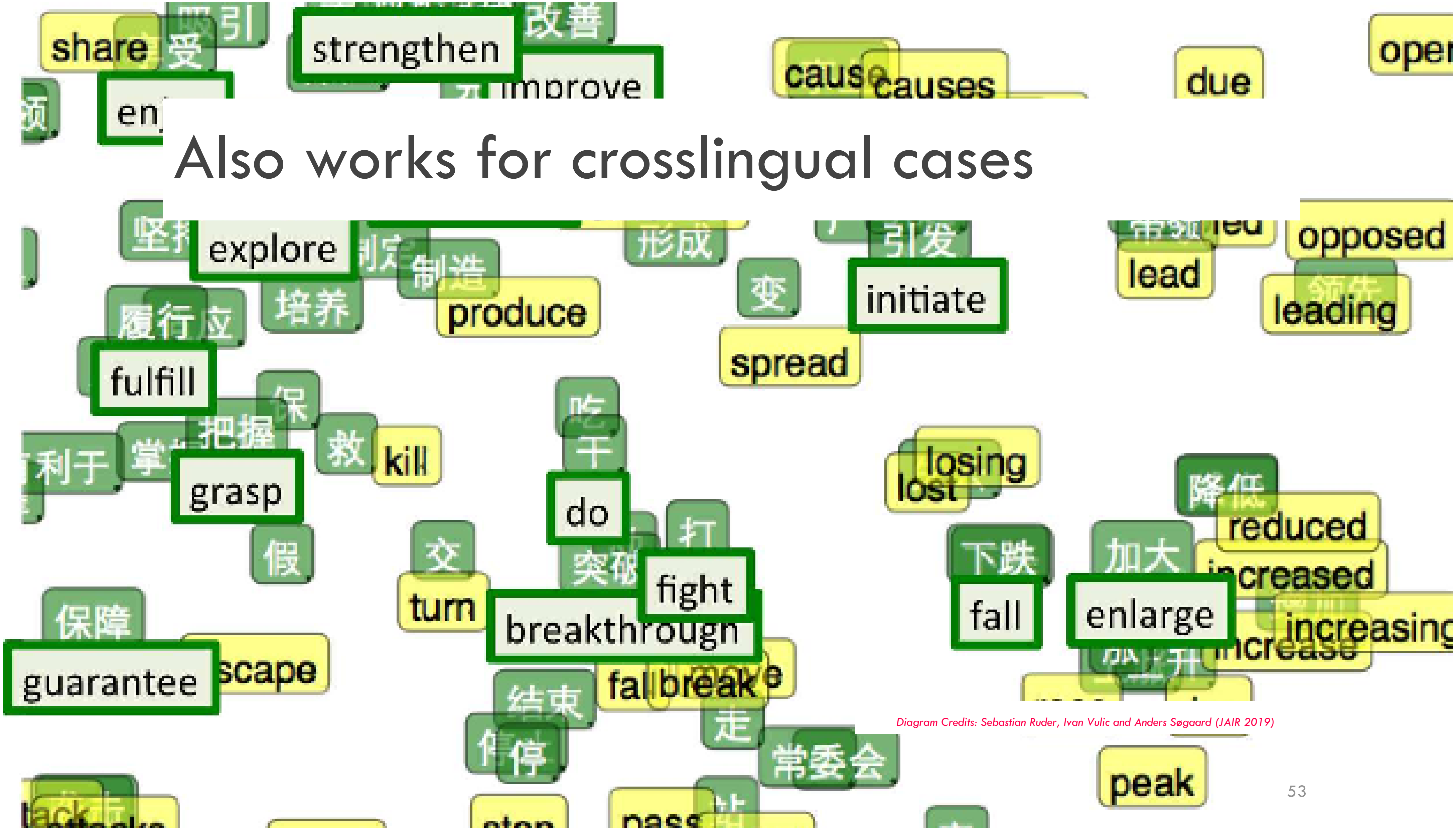


Diagram Credits: Sebastian Ruder, Ivan Vulic and Anders Søgaard (JAIR 2019)

Embeddings Summary

Move from term–document matrix to a term–context matrix

- Solves semantically relatedness

Embed the resultant term–context vectors into a denser space

- The side effect is the objective!
- Solves sparsity problem

Vectorial differences yields semantics relationships

Many extensions, we'll see some later