A practical introduction to distributional semantics
PART I: Co-occurrence matrix models

Marco Baroni
Center for Mind/Brain Sciences
University of Trento

Symposium on Semantic Text Processing
Bar Ilan University November 2014
Acknowledging.

Georgiana Dinu

COMPOSES:
COMPositional Operations in SEmantic Space
The vastness of word meaning
The distributional hypothesis
Harris, Charles and Miller, Firth, Wittgenstein? ... 

The meaning of a word is (can be approximated by, learned from) the set of contexts in which it occurs in texts

We found a little, hairy wampimuk sleeping behind the tree

See also MacDonald & Ramscar CogSci 2001
Distributional semantic models in a nutshell

“Co-occurrence matrix” models, see Yoav’s part for neural models

- Represent words through vectors recording their co-occurrence counts with context elements in a corpus

- (Optionally) apply a re-weighting scheme to the resulting co-occurrence matrix

- (Optionally) apply dimensionality reduction techniques to the co-occurrence matrix

- Measure geometric distance of word vectors in “distributional space” as proxy to semantic similarity/relatedness
he curtains open and the moon shining in on the barely
ars and the cold, close moon. And neither of the w
rough the night with the moon shining so brightly, it
made in the light of the moon. It all boils down, wr
surely under a crescent moon, thrilled by ice-white
sun, the seasons of the moon? Home, alone, Jay pla
m is dazzling snow, the moon has risen full and cold
un and the temple of the moon, driving out of the hug
in the dark and now the moon rises, full and amber a
bird on the shape of the moon over the trees in front
But I could n’t see the moon or the stars, only the
rning, with a sliver of moon hanging among the stars
they love the sun, the moon and the stars. None of
the light of an enormous moon. The plash of flowing w
man’s first step on the moon; various exhibits, aer
the inevitable piece of the moon rock. Housing The Airsh
oud obscured part of the moon. The Allied guns behind
Extracting co-occurrence counts
Variations in context features

```
<table>
<thead>
<tr>
<th>Doc1</th>
<th>Doc2</th>
<th>Doc3</th>
</tr>
</thead>
<tbody>
<tr>
<td>stars</td>
<td>38</td>
<td>45</td>
</tr>
</tbody>
</table>

The nearest • to Earth stories of • and their stars

```

```
see bright shiny stars
dobj
mod
mod
1
dobj
←−−see mod−−→bright mod−−→shiny
```

```
<table>
<thead>
<tr>
<th>stars</th>
<th>38</th>
<th>45</th>
<th>44</th>
</tr>
</thead>
</table>
```
Extracting co-occurrence counts

Variations in the definition of co-occurrence

E.g.: Co-occurrence with words, window of size 2, scaling by distance to target:

... two [intensely bright stars in the] night sky ...

<table>
<thead>
<tr>
<th>intensely</th>
<th>bright</th>
<th>in</th>
<th>the</th>
</tr>
</thead>
<tbody>
<tr>
<td>stars</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Same corpus (BNC), different window sizes

Nearest neighbours of **dog**

**2-word window**
- cat
- horse
- fox
- pet
- rabbit
- pig
- animal
- mongrel
- sheep
- pigeon

**30-word window**
- kennel
- puppy
- pet
- bitch
- terrier
- rottweiler
- canine
- cat
- to bark
- Alsatian
From co-occurrences to vectors

<table>
<thead>
<tr>
<th></th>
<th>bright</th>
<th>in</th>
<th>sky</th>
</tr>
</thead>
<tbody>
<tr>
<td>stars</td>
<td>8</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>sun</td>
<td>10</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>dog</td>
<td>2</td>
<td>20</td>
<td>1</td>
</tr>
</tbody>
</table>
Weighting

Re-weight the counts using corpus-level statistics to reflect co-occurrence *significance*

**Positive Pointwise Mutual Information (PPMI)**

$$\text{PPMI}(\text{target}, \text{ctxt}) = \max(0, \log \frac{P(\text{target}, \text{ctxt})}{P(\text{target})P(\text{ctxt})})$$
Weighting

Adjusting raw co-occurrence counts:

<table>
<thead>
<tr>
<th></th>
<th>bright</th>
<th>in</th>
</tr>
</thead>
<tbody>
<tr>
<td>stars</td>
<td>385</td>
<td>10788</td>
</tr>
<tr>
<td></td>
<td>43.6</td>
<td>5.3</td>
</tr>
</tbody>
</table>

Counts

PPMI

Other weighting schemes:

- TF-IDF
- Local Mutual Information
- Dice

Dimensionality reduction

- Vector spaces often range from tens of thousands to millions of context dimensions
- Some of the methods to reduce dimensionality:
  - Select context features based on various relevance criteria
  - Random indexing
  - Following claimed to also have a beneficial *smoothing* effect:
    - Singular Value Decomposition
    - Non-negative matrix factorization
    - Probabilistic Latent Semantic Analysis
    - Latent Dirichlet Allocation
The SVD factorization

\[ M \approx U S V \]

contexts

words

Image courtesy of Yoav
Dimensionality reduction as “smoothing”
From geometry to similarity in meaning

Vectors

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>stars</td>
<td>2.5</td>
</tr>
<tr>
<td>sun</td>
<td>2.9</td>
</tr>
</tbody>
</table>

Cosine similarity

$$\cos(x, y) = \frac{\langle x, y \rangle}{\|x\|\|y\|}$$

$$= \frac{\sum_{i=1}^{n} x_i \times y_i}{\sqrt{\sum_{i=1}^{n} x^2} \times \sqrt{\sum_{i=1}^{n} y^2}}$$

Other similarity measures: Euclidean Distance, Dice, Jaccard, Lin...
<table>
<thead>
<tr>
<th>rhino</th>
<th>fall</th>
<th>good</th>
<th>sing</th>
</tr>
</thead>
<tbody>
<tr>
<td>woodpecker</td>
<td>rise</td>
<td>bad</td>
<td>dance</td>
</tr>
<tr>
<td>rhinoceros</td>
<td>increase</td>
<td>excellent</td>
<td>whistle</td>
</tr>
<tr>
<td>swan</td>
<td>fluctuation</td>
<td>superb</td>
<td>mime</td>
</tr>
<tr>
<td>whale</td>
<td>drop</td>
<td>poor</td>
<td>shout</td>
</tr>
<tr>
<td>ivory</td>
<td>decrease</td>
<td>improved</td>
<td>sound</td>
</tr>
<tr>
<td>plover</td>
<td>reduction</td>
<td>perfect</td>
<td>listen</td>
</tr>
<tr>
<td>elephant</td>
<td>logarithm</td>
<td>clever</td>
<td>recite</td>
</tr>
<tr>
<td>bear</td>
<td>decline</td>
<td>terrific</td>
<td>play</td>
</tr>
<tr>
<td>satin</td>
<td>cut</td>
<td>lucky</td>
<td>hear</td>
</tr>
<tr>
<td>sweatshirt</td>
<td>hike</td>
<td>smashing</td>
<td>hiss</td>
</tr>
</tbody>
</table>
Benchmarks
Similarity/relatedness

E.g: Rubenstein and Goodenough, WordSim-353, MEN, SimLex-99...

MEN

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>chapel</td>
<td>church</td>
<td>0.45</td>
</tr>
<tr>
<td>eat</td>
<td>strawberry</td>
<td>0.33</td>
</tr>
<tr>
<td>jump</td>
<td>salad</td>
<td>0.06</td>
</tr>
<tr>
<td>bikini</td>
<td>pizza</td>
<td>0.01</td>
</tr>
</tbody>
</table>

How: Measure correlation of model cosines with human similarity/relatedness judgments

Top MEN Spearman correlation for co-occurrence matrix models (Baroni et al. ACL 2014): 0.72
Benchmarks

Categorization

E.g: Almuhareb/Poesio, ESSLLI 2008 Shared Task, Battig set

ESSLLI

<table>
<thead>
<tr>
<th>VEHICLE</th>
<th>MAMMAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>helicopter</td>
<td>dog</td>
</tr>
<tr>
<td>motorcycle</td>
<td>elephant</td>
</tr>
<tr>
<td>car</td>
<td>cat</td>
</tr>
</tbody>
</table>

**How:** Feed model-produced similarity matrix to clustering algorithm, look at overlap between clusters and gold categories

Top ESSLLI cluster purity for co-occurrence matrix models (Baroni et al. ACL 2014): 0.84
Benchmarks
Selectional preferences

E.g: Ulrike Padó, Ken McRae et al.’s data sets

Padó

<table>
<thead>
<tr>
<th>eat</th>
<th>villager</th>
<th>obj</th>
<th>1.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>eat</td>
<td>pizza</td>
<td>obj</td>
<td>6.8</td>
</tr>
<tr>
<td>eat</td>
<td>pizza</td>
<td>subj</td>
<td>1.1</td>
</tr>
</tbody>
</table>

How (Erk et al. CL 2010): 1) Create “prototype” argument vector by averaging vectors of nouns typically occurring as argument fillers (e.g., frequent objects of to eat); 2) measure cosine of target noun with prototype (e.g., cosine of villager vector with eat-object prototype vector); 3) correlate with human scores

Top Padó Spearman correlation for co-occurrence matrix models (Baroni et al. ACL 2014): 0.41
## Selectional preferences

Examples from Baroni/Lenci implementation

To kill... 

<table>
<thead>
<tr>
<th>object</th>
<th>cosine</th>
<th>with</th>
<th>cosine</th>
</tr>
</thead>
<tbody>
<tr>
<td>kangaroo</td>
<td>0.51</td>
<td>hammer</td>
<td>0.26</td>
</tr>
<tr>
<td>person</td>
<td>0.45</td>
<td>stone</td>
<td>0.25</td>
</tr>
<tr>
<td>robot</td>
<td>0.15</td>
<td>brick</td>
<td>0.18</td>
</tr>
<tr>
<td>hate</td>
<td>0.11</td>
<td>smile</td>
<td>0.15</td>
</tr>
<tr>
<td>flower</td>
<td>0.11</td>
<td>flower</td>
<td>0.12</td>
</tr>
<tr>
<td>stone</td>
<td>0.05</td>
<td>antibiotic</td>
<td>0.12</td>
</tr>
<tr>
<td>fun</td>
<td>0.05</td>
<td>person</td>
<td>0.12</td>
</tr>
<tr>
<td>book</td>
<td>0.04</td>
<td>heroin</td>
<td>0.12</td>
</tr>
<tr>
<td>conversation</td>
<td>0.03</td>
<td>kindness</td>
<td>0.07</td>
</tr>
<tr>
<td>sympathy</td>
<td>0.01</td>
<td>graduation</td>
<td>0.04</td>
</tr>
</tbody>
</table>
Benchmarks

Analogy

Method and data sets from Mikolov and collaborators

<table>
<thead>
<tr>
<th>syntactic analogy</th>
<th>semantic analogy</th>
</tr>
</thead>
<tbody>
<tr>
<td>work speak</td>
<td>brother grandson</td>
</tr>
<tr>
<td>works speaks</td>
<td>sister granddaughter</td>
</tr>
</tbody>
</table>

speaks $\approx$ works $-$ work $+$ speak

How: Response counts as hit only if nearest neighbour (in large vocabulary) of vector obtained with subtraction and addition operations above is the intended one

Top accuracy for co-occurrence matrix models (Baroni et al. ACL 2014): 0.49
Distributional semantics: A general-purpose representation of lexical meaning
Baroni and Lenci 2010

- Similarity (*cord-string* vs. *cord-smile*)
- Synonymy (*zenith-pinnacle*)
- Concept categorization (*car* ISA *vehicle*; *banana* ISA *fruit*)
- Selectional preferences (*eat topinambur* vs. *eat sympathy*)
- Analogy (*mason* is to *stone* like *carpenter* is to *wood*)
- Relation classification (*exam-anxiety* are in CAUSE-EFFECT relation)
- Qualia (TELIC ROLE of *novel* is *to entertain*)
- Salient properties (*car-wheels*, *dog-barking*)
- Argument alternations (*John broke the vase* - *the vase broke*, *John minces the meat* - *the meat minced*)
Practical recommendations
Mostly from Baroni et al. ACL 2014, see more evaluation work in reading list below

- Narrow context windows are best (1, 2 words left and right)
- Full matrix better than dimensionality reduction
- PPMI weighting best
- Dimensionality reduction with SVD better than with NMF
An example application
Bilingual lexicon/phrase table induction from monolingual resources

Saluja et al. (ACL 2014) obtain significant improvements in English-Urdu and English-Arabic BLEU scores using phrase tables enlarged with pairs induced by exploiting distributional similarity structure in source and target languages.

Figure credit: Mikolov et al 2013
The infinity of sentence meaning

What you've got means such a lot to me.
I know a mouse, and he hasn't got a house. Who put all those things in your hair?
Doctor Robert, you're a new and better man. There's one for you. Nineteen for me.
Ring my friend, I said you call Doctor Robert. Because I'm the taxman. Yeah, I'm the taxman. Lying there and staring at the ceiling. If you don't want to pay some more. No fair. You can't hear me but I can you.
He'll be found when you're around. But now he's resigned to his fate. Leave me where I find him.
Oh no. Here we go! Ever so high. He had a big adventure. Amidst the grass and fresh air at last. Here a man, there a man, lots of gingerbread men.
So we sailed up to the sun till we found the sea of green.
There's people standing round Who screw you in the ground.

I was a boy when everything was right. Everything was right I said.
I knew I'd never care But to love her is to need her everywhere.
The sky of blue and sea of green in our yellow submarine.
Waits at the window, wearing the face that she keeps in a jar by the door.

Eleanor Rigby died in the church and was buried along with her name.

It's like the sea of green. A sky of blue and sea of green in our yellow submarine.

No need to make plans. I'm planning to go and make them....

I'm planning to go and make them....

The infinity of sentence meaning
Compositionality
The meaning of an utterance is a function of the meaning of its parts and their composition rules (Frege 1892)
Compositional distributional semantics: What for?

Word meaning in context (Mitchell and Lapata ACL 2008)

Paraphrase detection (Blacoe and Lapata EMNLP 2012)

the cucumber is rotten

the cucumber is old

the cucumber is ancient

"cookie dwarfs hop under the crimson planet"

"gingerbread gnomes dance under the red moon"

"red gnomes love gingerbread cookies"

"students eat cup noodles"
Compositional distributional semantics: How?

From:
Simple functions

\[ \text{very} \quad \rightarrow \quad + \quad \rightarrow \quad \text{good} \quad \rightarrow \quad \text{movie} \]

\[ \text{very good movie} \]

Mitchell and Lapata
ACL 2008

To:
Complex composition operations

Socher at al. EMNLP 2013
Some references

- **Classics:**
  - Schütze’s 1997 CSLI book
  - Landauer and Dumais PsychRev 1997

- **Overviews:**
  - Turney and Pantel JAIR 2010
  - Erk LLC 2012
  - Baroni LLC 2013
  - Clark to appear in Handbook of Contemporary Semantics

- **Evaluation:**
  - Sahlgren’s 2006 thesis
  - Bullinaria and Levy BRM 2007, 2012
  - Baroni, Dinu and Kruszewski ACL 2014
  - Kiela and Clark CVSC 2014
Fun with distributional semantics!

http://clic.cimec.unitn.it/infomap-query/
Making Sense of Distributed (Neural) Semantics

Yoav Goldberg
yoav.goldberg@gmail.com

Nov 2014
The new kid on the block

- Deep learning / neural networks
- “Distributed” word representations
  - Feed text into neural-net. Get back “word embeddings”.
  - Each word is represented as a low-dimensional vector.
  - Vectors capture “semantics”
- \texttt{word2vec} (Mikolov et al)
From Distributional to Distributed Semantics

This part of the talk

- word2vec as a black box
- a peek inside the black box
- relation between word-embeddings and the distributional representation
- tailoring word embeddings to your needs using word2vecf
Introduction

This tool provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. These representations can be subsequently used in many natural language processing applications and for further research.

Quick start

- Download the code: svn checkout http://word2vec.googlecode.com/svn/trunk/
- Run 'make' to compile word2vec tool
- Run the demo scripts: /demo-word.sh and /demo-phrases.sh
- For questions about the toolkit, see http://groups.google.com/group/word2vec-toolkit

How does it work

The word2vec tool takes a text corpus as input and produces the word vectors as output. It first
word2vec

feed in text

Text

WIKIPEDIA

wait a few hours

\[
\begin{align*}
dog &= (0.12, -0.32, 0.92, 0.43, -0.3 \ldots) \\
cat &= (0.15, -0.29, 0.90, 0.39, -0.32 \ldots) \\
chair &= (0.8, 0.9, -0.76, 0.29, 0.52 \ldots)
\end{align*}
\]

get a $|V| \times d$ matrix $W$ where each row is a vector for a word
word2vec

- **dog**
  - cat, dogs, dachshund, rabbit, puppy, poodle, rottweiler, mixed-breed, doberman, pig

- **sheep**
  - cattle, goats, cows, chickens, sheeps, hogs, donkeys, herds, shorthorn, livestock

- **november**
  - october, december, april, june, february, july, september, january, august, march

- **jerusalem**
  - tiberias, jaffa, haifa, israel, palestine, nablus, damascus katamon, ramla, safed

- **teva**
  - pfizer, schering-plough, novartis, astrazeneca, glaxosmithkline, sanofi-aventis, mylan, sanofi, genzyme, pharmacia
Word Similarity

- Similarity is calculated using cosine similarity:

\[
\text{sim}(\vec{dog}, \vec{cat}) = \frac{\vec{dog} \cdot \vec{cat}}{||\vec{dog}|| \cdot ||\vec{cat}||}
\]

- For normalized vectors ($||x|| = 1$), this is equivalent to a dot product:

\[
\text{sim}(\vec{dog}, \vec{cat}) = \vec{dog} \cdot \vec{cat}
\]

- Normalize the vectors when loading them.
Working with Dense Vectors

Finding the most similar words to $\vec{dog}$

- Compute the similarity from word $\vec{v}$ to all other words.

- This is a $|V|$-sized vector of similarities.

- Take the indices of the $k$-highest values.

- FAST! for 180k words, $d=300$: $\sim 30$ ms
Working with Dense Vectors

Finding the most similar words to $\vec{dog}$

- Compute the similarity from word $\vec{v}$ to all other words.
- This is a **single matrix-vector product**: $W \cdot \vec{v}^T$

![Diagram showing matrix-vector product and similarity calculation]

$$W \in |V| \times d \quad \vec{v}^T \in d \times 1 \quad \text{similarities} \in 1 \times |V|$$
Working with Dense Vectors

Finding the most similar words to $\vec{dog}$

- Compute the similarity from word $\vec{v}$ to all other words.
- This is a **single matrix-vector product**: $W \cdot \vec{v}^T$

\[
\begin{array}{c}
\text{cat} & \text{chair} & \text{june} & \text{sun} & \text{bark} & \ldots & \ldots & \text{eat} \\
\end{array}
\begin{array}{c}
\text{dog} \\
\end{array}
\]

\[
W \cdot \vec{v}^T = \begin{pmatrix}
0.9 & -0.3 & -0.1 & -0.9 & 0.3 & \ldots & \ldots & 0.2 \\

dog & cat & june & sun & bark & \ldots & \ldots & eat
\end{pmatrix}
\]

- Result is a $|V|$ sized vector of similarities.
- Take the indices of the $k$-highest values.
Working with Dense Vectors

Finding the most similar words to $\vec{dog}$

- Compute the similarity from word $\vec{v}$ to all other words.
- This is a **single matrix-vector product**: $W \cdot \vec{v}^T$

\[
\begin{align*}
|V| & \quad \begin{bmatrix}
\text{cat} & \text{chair} & \text{june} & \text{sun} & \text{bark} & \ldots & \ldots & \text{eat} \\
\end{bmatrix} \\
W & \quad \begin{bmatrix}
0.9 & -0.3 & -0.1 & -0.9 & 0.3 & \ldots & \ldots & 0.2 \\
\end{bmatrix} \\
\vec{v}^T & \quad \begin{bmatrix}
\text{dog} & \text{cat} & \text{chair} & \text{june} & \text{sun} & \text{bark} & \ldots & \ldots & \text{eat} \\
\end{bmatrix} \\
\end{align*}
\]

- Result is a $|V|$ sized vector of similarities.
- Take the indices of the $k$-highest values.
- **FAST!** for 180k words, $d=300$: $\sim 30$ms
Most Similar Words, in python+numpy code

```
W, words = load_and_normalize_vectors("vecs.txt")
# W and words are numpy arrays.
w2i = {w:i for i,w in enumerate(words)}

dog = W[w2i[\'dog\']]  # get the dog vector
sims = W.dot(dog)  # compute similarities

most_similar_ids = sims.argsort()[-1:-10:-1]
sim_words = words[most_similar_ids]
```
Similarity to a group of words

- “Find me words most similar to cat, dog and cow”.
- Calculate the pairwise similarities and sum them:
  \[ W \cdot \vec{cat} + W \cdot \vec{dog} + W \cdot \vec{cow} \]
- Now find the indices of the highest values as before.
Working with Dense Vectors

Similarity to a group of words

- “Find me words most similar to cat, dog and cow”.
- Calculate the pairwise similarities and sum them:
  \[ W \cdot \vec{cat} + W \cdot \vec{dog} + W \cdot \vec{cow} \]
- Now find the indices of the highest values as before.
- Matrix-vector products are wasteful. **Better option:**
  \[ W \cdot (\vec{cat} + \vec{dog} + \vec{cow}) \]
Working with dense word vectors can be very efficient.
Working with dense word vectors can be very efficient.

But where do these vectors come from?
How does word2vec work?

word2vec implements several different algorithms:

**Two training methods**
- Negative Sampling
- Hierarchical Softmax

**Two context representations**
- Continuous Bag of Words (CBOW)
- Skip-grams
How does word2vec work?

word2vec implements several different algorithms:

**Two training methods**

- **Negative Sampling**
- **Hierarchical Softmax**

**Two context representations**

- **Continuous Bag of Words (CBOW)**
- **Skip-grams**

We’ll focus on skip-grams with negative sampling

intuitions apply for other models as well
How does word2vec work?

- Represent each word as a $d$ dimensional vector.
- Represent each context as a $d$ dimensional vector.
- Initialize all vectors to random weights.
- Arrange vectors in two matrices, $W$ and $C$. 

$$|V_w|$$ words

$$|V_c|$$ contexts

$W$ $C$
How does word2vec work?

While more text:

- Extract a word window:
  A springer is [ a cow or heifer close to calving ].
  \[ c_1 \quad c_2 \quad c_3 \quad w \quad c_4 \quad c_5 \quad c_6 \]

- \( w \) is the focus word vector (row in \( W \)).
- \( c_i \) are the context word vectors (rows in \( C \)).
How does word2vec work?
While more text:

▶ Extract a word window:

A springer is [a cow or heifer close to calving].

\[ c_1 \quad c_2 \quad c_3 \quad w \quad c_4 \quad c_5 \quad c_6 \]

▶ Try setting the vector values such that:

\[ \sigma(w \cdot c_1) + \sigma(w \cdot c_2) + \sigma(w \cdot c_3) + \sigma(w \cdot c_4) + \sigma(w \cdot c_5) + \sigma(w \cdot c_6) \]

is high
How does word2vec work?

While more text:

- Extract a word window:
  A springer is [ a cow or heifer close to calving ].
  \[ c_1 \quad c_2 \quad c_3 \quad w \quad c_4 \quad c_5 \quad c_6 \]

- Try setting the vector values such that:
  \[ \sigma(w \cdot c_1) + \sigma(w \cdot c_2) + \sigma(w \cdot c_3) + \sigma(w \cdot c_4) + \sigma(w \cdot c_5) + \sigma(w \cdot c_6) \]
  is high

- Create a corrupt example by choosing a random word \( w' \)
  [ a cow or comet close to calving ]
  \[ c_1 \quad c_2 \quad c_3 \quad w' \quad c_4 \quad c_5 \quad c_6 \]

- Try setting the vector values such that:
  \[ \sigma(w' \cdot c_1) + \sigma(w' \cdot c_2) + \sigma(w' \cdot c_3) + \sigma(w' \cdot c_4) + \sigma(w' \cdot c_5) + \sigma(w' \cdot c_6) \]
  is low
How does word2vec work?

The training procedure results in:
- $w \cdot c$ for **good** word-context pairs is **high**
- $w \cdot c$ for **bad** word-context pairs is **low**
- $w \cdot c$ for **ok-ish** word-context pairs is **neither high nor low**

As a result:
- Words that share many contexts get close to each other.
- Contexts that share many words get close to each other.

At the end, word2vec throws away $C$ and returns $W$. 
Reinterpretation

Imagine we didn’t throw away $C$. Consider the product $WC^\top$

▶ Each row corresponds to a word.
▶ Each column corresponds to a context.
▶ Each cell corresponds to $w \cdot c$, an association measure between a word and a context.
Reinterpretation

Imagine we didn’t throw away $C$. Consider the product $WC^T$

The result is a matrix $M$ in which:

- Each row corresponds to a word.
- Each column corresponds to a context.
- Each cell correspond to $w \cdot c$, an association measure between a word and a context.
Reinterpretation

Does this remind you of something?

Very similar to SVD over distributional representation:
Reinterpretation

Does this remind you of something?
Very similar to SVD over distributional representation:
Relation between SVD and word2vec

SVD

- Begin with a word-context matrix.
- Approximate it with a product of low rank (thin) matrices.
- Use thin matrix as word representation.

word2vec (skip-grams, negative sampling)

- Learn thin word and context matrices.
- These matrices can be thought of as approximating an implicit word-context matrix.
  - In Levy and Goldberg (NIPS 2014) we show that this implicit matrix is related to the well-known PPMI matrix.
Relation between SVD and word2vec

word2vec is a dimensionality reduction technique over an (implicit) word-context matrix.

Just like SVD.

With few tricks (Levy, Goldberg and Dagan, in submission) we can get SVD to perform just as well as word2vec.
Relation between SVD and word2vec

word2vec is a dimensionality reduction technique over an (implicit) word-context matrix.

Just like SVD.

With few tricks (Levy, Goldberg and Dagan, in submission) we can get SVD to perform just as well as word2vec.

However, word2vec...

- ...works without building / storing the actual matrix in memory.
- ...is very fast to train, can use multiple threads.
- ...can easily scale to huge data and very large word and context vocabularies.
Beyond word2vec
Beyond word2vec

- word2vec is factorizing a word-context matrix.
- The content of this matrix affects the resulting similarities.
- word2vec allows you to specify a *window size*.
- But what about other types of contexts?
- Example: *dependency contexts* (Levy and Dagan, ACL 2014)
Australian scientist discovers star with telescope
Australian scientist discovers star with telescope

Bag of Words (BoW) Context
Australian scientist discovers star with telescope
Syntactic Dependency Context

Australian *scientist* **discovers** star with telescope
Australian scientist discovers star with telescope

Syntactic Dependency Context

Australian scientist discovers star with telescope
Embedding Similarity with Different Contexts

<table>
<thead>
<tr>
<th>Target Word</th>
<th>Bag of Words (k=5)</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hogwarts</td>
<td>Dumbledore hallows half-blood Malfoy Snape</td>
<td>Sunnydale Collinwood</td>
</tr>
<tr>
<td>(Harry Potter’s school)</td>
<td></td>
<td>Calarts Greendale Millfield</td>
</tr>
</tbody>
</table>

Related to Harry Potter Schools
## Embedding Similarity with Different Contexts

<table>
<thead>
<tr>
<th>Target Word</th>
<th>Bag of Words (k=5)</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turing (computer scientist)</td>
<td>nondeterministic non-deterministic computability deterministic finite-state</td>
<td>Pauling Hotelling Heting Lessing Hamming</td>
</tr>
</tbody>
</table>

**Related to computability**

**Scientists**
# Embedding Similarity with Different Contexts

<table>
<thead>
<tr>
<th>Target Word</th>
<th>Bag of Words (k=5)</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>dancing (dance gerund)</td>
<td>singing, dance, dances, dancers, tap-dancing</td>
<td>singing, rapping, breakdancing, miming, busking</td>
</tr>
</tbody>
</table>

**Related to dance**

**Online Demo!**
Context matters

Choose the correct contexts for your application

- larger window sizes – more topical
- dependency relations – more functional
Context matters

Choose the correct contexts for your application

- larger window sizes – more topical
- dependency relations – more functional
- only noun-adjective relations
- only verb-subject relations
Choose the correct contexts for your application

- larger window sizes – more topical
- dependency relations – more functional
- only noun-adjective relations
- only verb-subject relations
- context: time of the current message
- context: user who wrote the message
Choose the correct contexts for your application

- larger window sizes – more topical
- dependency relations – more functional
- only noun-adjective relations
- only verb-subject relations
- context: time of the current message
- context: user who wrote the message
- …
- the sky is the limit
Software

word2vecf
https://bitbucket.org/yoavgo/word2vecf

- Extension of word2vec.
- Allows saving the context matrix.
- Allows using arbitrary contexts.
  - Input is a (large) file of word context pairs.
hyperwords
https://bitbucket.org/omerlevy/hyperwords/

- Python library for working with either sparse or dense word vectors (similarity, analogies).
- Scripts for creating dense representations using word2vecf or SVD.
- Scripts for creating sparse distributional representations.
Software

dissect
http://clic.cimec.unitn.it/composes/toolkit/

- Given vector representation of words…
- …derive vector representation of phrases/sentences
- Implements various composition methods
Summary

Distributional Semantics

- Words in similar contexts have similar meanings.
- Represent a word by the contexts it appears in.
- But what is a context?

Neural Models (word2vec)

- Represent each word as dense, low-dimensional vector.
- Same intuitions as in distributional vector-space models.
- Efficient to run, scales well, modest memory requirement.
- Dense vectors are convenient to work with.
- Still helpful to think of the context types.

Software

- Build your own word representations.