Demystifying Neural Word Embeddings

Yoav Goldberg
yoav.goldberg@gmail.com

September 2015
Language
Language

People use language to communicate
Language

People use language to communicate

Language is Everywhere
Language

People use language to communicate

Language is Everywhere

- Newspapers
Language

People use language to communicate

Language is Everywhere

- Newspapers
- Scientific articles
Language

People use language to communicate

Language is Everywhere

- Newspapers
- Scientific articles
- Medicine (patient records)
Language

People use language to communicate

Language is Everywhere

- Newspapers
- Scientific articles
- Medicine (patient records)
- Patents
Language

People use language to communicate

Language is Everywhere

- Newspapers
- Scientific articles
- Medicine (patient records)
- Patents
- Law
Language

People use language to communicate

Language is Everywhere

- Newspapers
- Scientific articles
- Medicine (patient records)
- Patents
- Law
- Product reviews
Language

People use language to communicate

Language is Everywhere

- Newspapers
- Scientific articles
- Medicine (patient records)
- Patents
- Law
- Product reviews
- Blogs
- Facebook, Twitter
- ...
A lot of text.

Need to understand what’s being said.

this is where we come in.
What does it mean to understand?

I focus on the building blocks...
NLP

What does it mean to understand?
NLP

What does it mean to **understand**?

I focus on the building blocks
Understanding the Structure

The soup, which I expected to be good, was bad.
The soup, which I expected to be good, was bad.
Understanding the Structure

This is called **Syntactic Parsing**.
The soup, which I expected to be good, was bad.
Understanding the Structure

The soup, which I expected to be good, was bad.
Understanding the Structure

The gromp, which I furpled to be drogby, was spujky
The gromp, which I furpled to be drogby, was spujky
Greedy parsing to Stanford Dependencies

The gromp, which I furpled to be drogby, was spujky.

- R-/ROOT  The  gromp  ,  which  I  furpled  to  be  drogby  ,  was  spujky  .
Can understand **structure** without understanding **words**.

But the words are also important.
I almost gave you a talk about parsing.

Today we will focus on the words.
Understanding the Words

soup was bad

chowder was nasty
pudding was terrible
hamburger was lousy
service was poor
atmosphere was shoddy
hammer was heavy

To the computer, each word is just a symbol, so these are all the same.
But to us, some are more similar than others.
We'd like a word representation that can capture that.
Understanding the Words

soup was bad
soup was awful
Understanding the Words

soup was bad
soup was awful
soup was lousy
Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

▶ To the computer, each word is just a symbol, so these are all the same.
▶ But to us, some are more similar than others.
▶ We’d like a word representation that can capture that.
Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

chowder was nasty
Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

chowder was nasty
pudding was terrible
Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

chowder was nasty
pudding was terrible
cake was bad
Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

chowder was nasty
pudding was terrible
cake was bad
hamburger was lousy
Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

chowder was nasty
pudding was terrible
cake was bad
hamburger was lousy

service was poor
Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

chowder was nasty
pudding was terrible
cake was bad
hamburger was lousy

service was poor
atmosphere was shoddy
Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

chowder was nasty
pudding was terrible
cake was bad
hamburger was lousy

service was poor
atmosphere was shoddy
hammer was heavy
Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

chowder was nasty
pudding was terrible
cake was bad
hamburger was lousy

service was poor
atmosphere was shoddy
hammer was heavy

- To the computer, each word is just a symbol, so these are all the same.
- But to us, some are more similar than others.
- We’d like a word representation that can capture that.
Representing Words

Use a dictionary?
Representing Words

Use a dictionary?

Doesn’t scale.
Dr. Baroni saw a hairy little wampinuck sleeping behind a tree
Dr. Baroni saw a hairy little wampinuck sleeping behind a tree

The Distributional Hypothesis – Haris, 1954
Words in similar contexts tend to have similar meanings

Firth, 1957
“you should know a word by the company it keeps”
The curtains open and the moon shining in on the barely ars and the cold, close moon." And neither of the wrough 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 plam 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 splash of flowing wman 's first step on the moon; various exhibits, aer the inevitable piece of moon rock. H Housing The Airsh oud obscured part of the moon. The Allied guns behind
Words as Vectors

- Represent each word as a sparse, high dimensional vector of the words that co-occur with it.
  
moon = (the:324, shining:4, cold:1, brightly:2, stars:12, elephant:0, ...)

- Words are similar if their vectors are similar.

- We measure similarity using geometric measures, for example cosine distance.

- But more intuitively, words are similar if they share many similar contexts.
Weighting

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

Positive Pointwise Mutual Information (PPMI)

$$PPMI(target, ctxt) = \max(0, \log \frac{P(target, ctxt)}{P(target)P(ctxt)})$$
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>stars</td>
<td>43.6</td>
<td>5.3</td>
</tr>
</tbody>
</table>

← Counts

← PPMI

Other weighting schemes:

- TF-IDF
- Local Mutual Information
- Dice

We can arrange the words in a huge, sparse matrix, where each row is a word, and each column is a context.
Words as Vectors

We often apply SVD or similar technique of dimensionality reduction.
Words as Vectors – It works

Nearest neighbours to dog

- cat
- horse
- fox
- pet
- rabbit
- pig
- animal
- mongrel
- sheep
- pigeon
Words as Vectors – It works
Nearest neighbours to dog

2-word window

- cat
- horse
- fox
- pet
- rabbit
- pig
- animal
- mongrel
- sheep
- pigeon
Words as Vectors – It works

Nearest neighbours to 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
- Alastian
word2vec

Tool for computing continuous distributed representations of words.

Project Information
- Starred by 694 users
- Project feeds
- Code license: Apache License 2.0
- Labels: NeuralNetwork, MachineLearning, NaturalLanguageProcessing, WordVectors, Google
- Members: tmiko...@gmail.com, 6 contributors

Links

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
MLMU.cz - Radim Řehůřek - Word2vec & friends (7.1.2015 ... 
www.youtube.com/watch?v=wTq3P2UnTfQ
Jan 14, 2015 - Uploaded by Marek Modrý
I'll go over a particular model published by Google, called
word2vec, its optimizations, applications and ...

Word2Vec convergence on Vimeo
https://vimeo.com/1124698934
Nov 18, 2014
This is "Word2Vec convergence" by MaciejLyst on Vimeo, the home
for high quality videos and the people who ...

Statistical Semantic入門～分布仮説からword2vecまで #1 ...
www.ustream.tv/recorded/43497190
Statistical Semantic入門～分布仮説からword2vecまで #1. February 5, 2014 at 7:16pm ...

Statistical Semantic入門～分布仮説からword2vecまで #2, PFI ...
www.ustream.tv/recorded/43497424
Feb 5, 2014
非常に説明がわかりやすくて!「データマイニング入門」と「マーケティングとスタートアップの話」を見ましたが、どちらも非常にお得に ...

GigaOM Show: Samsung watch secrets spilled! B&N's Nook ...
word2vecによる自然言語処理

著者: 西尾幸和
発行日: 2016年12月
ISBN: 4-87311-663-9
フォーマット:ePub mobi

Ebook Storeで電子版を購入:
価格: ¥1,312

内容

Tomas Mikolovらによって提案されたニューラルネットワーク (CBOW, Skip-gram) のオープンソース実装
word2vecについて、基本的な使い方を体験し、さらにその仕組みを学ぶ書籍です。
基本的な使い方から、自分の好きなコーパスの作り方、登場の背景、仕組み、さらには運用例や弱点についてもコンパクトな構成で解説できます。付録にはword2vecの出力結果を主成分分析を使って可視化する方法について解説しています。
なお本書はEbook版のみの発売となります。

著者の西尾先生による本書の解説[リンク]
From Distributional to Distributed Semantics

The new kid on the block

- “Distributed” word representations.
  - Feed text into neural-net. Get back “word embeddings”.
  - Each word is represented as a low-dimensional vector.
  - Vectors capture “semantics”.
- word2vec (Mikolov et al)
word2vec

feed in text

Text

WIKIPEDIA

wait a few hours

d

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

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
word2vec

Other appealing properties
Mikolov et al. (2013a,b,c)

\[ b \cdot a \cdot a^* \cdot b^* \]

king − man + woman = queen
Mikolov et al. (2013a,b,c)

\[ b \quad a \quad a^* \quad b^* \]

Tokyo \(-\) Japan \(+\) France \(=\) Paris
Mikolov et al. (2013a,b,c)

\[ b \quad a \quad a^* \quad b^* \]

best − good + strong = strongest
Mikolov et al. (2013a,b,c)

\[ b - a + a^* + b^* = \text{strongest} \]

vectors in \( \mathbb{R}^n \)
Seems magical.
Seems magical.

“Neural computation, just like in the brain!”
Seems magical.

“Neural computation, just like in the brain!”

How does this actually work?
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 $C$. 

$$|V_w|$$

$w$ 

$$|V_c|$$

$C$
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:

\[ \text{A springer is [ a cow or heifer close to calving ].} \]

\[
\begin{align*}
&c_1 \quad c_2 \quad c_3 \quad w \quad c_4 \quad c_5 \quad c_6 \\
\end{align*}
\]

» 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 $WC^T$
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?
Reinterpretation

Does this remind you of something?
Very similar to SVD over distributional representation:
What is SGNS learning?

• A $V_W \times V_C$ matrix

• Each cell describes the relation between a specific word-context pair

\[
\vec{w} \cdot \vec{c} = ?
\]

“Neural Word Embeddings as Implicit Matrix Factorization”
Levy & Goldberg, NIPS 2014
What is SGNS learning?

• We prove that for large enough $d$ and enough iterations

"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014
What is SGNS learning?

• We prove that for large enough $d$ and enough iterations
• We get the word-context PMI matrix

\[ M^{PMI} = V_d W \]
What is SGNS learning?

• We prove that for large enough $d$ and enough iterations
• We get the word-context PMI matrix, shifted by a global constant

$$\text{Opt}(\vec{w} \cdot \vec{c}) = \text{PMI}(w, c) - \log k$$

“Neural Word Embeddings as Implicit Matrix Factorization”
Levy & Goldberg, NIPS 2014
What is SGNS learning?

• SGNS is doing something very similar to the older approaches

• SGNS is factorizing the traditional word-context PMI matrix

• So does SVD!

• Do they capture the same similarity function?
# SGNS vs SVD

<table>
<thead>
<tr>
<th>Target Word</th>
<th>SGNS</th>
<th>SVD</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>dog, rabbit, cats, poodle, pig</td>
<td>dog, rabbit, pet, monkey, pig</td>
</tr>
</tbody>
</table>
# SGNS vs SVD

<table>
<thead>
<tr>
<th>Target Word</th>
<th>SGNS</th>
<th>SVD</th>
</tr>
</thead>
<tbody>
<tr>
<td>wine</td>
<td>wines, grape, grapes, winemaking, tasting</td>
<td>wines, grape, grapes, varietal, vintages</td>
</tr>
<tr>
<td>Target Word</td>
<td>SGNS</td>
<td>SVD</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
<td>--------------</td>
</tr>
<tr>
<td>November</td>
<td>October</td>
<td>October</td>
</tr>
<tr>
<td></td>
<td>December</td>
<td>December</td>
</tr>
<tr>
<td></td>
<td>April</td>
<td>April</td>
</tr>
<tr>
<td></td>
<td>January</td>
<td>June</td>
</tr>
<tr>
<td></td>
<td>July</td>
<td>March</td>
</tr>
</tbody>
</table>
But *word2vec* is still better, isn’t it?

- Plenty of evidence that *word2vec* outperforms traditional methods
  - In particular: “Don’t count, predict!” (Baroni et al., 2014)

- How does this fit with our story?
The Big Impact of “Small” Hyperparameters
Hyperparameters

• *word2vec* is more than just an algorithm...

• Introduces many **engineering tweaks** and **hyperparameter settings**
  - May seem minor, but **make a big difference** in practice
  - Their impact is often more significant than the embedding algorithm’s

• These modifications can be ported to distributional methods!

Levy, Goldberg, Dagan (In submission)
Hyperparameters

• Preprocessing
• Association Metric
• Postprocessing
Hyperparameters

• Preprocessing
• Association Metric
• Postprocessing
Association Metric Hyperparameters

- Since SGNS and PMI are strongly related, we can import 2 of SGNS’s hyperparameters to traditional PMI:

  1. Shifted PMI
  2. **Negative Sampling Smoothing**

- Both stem from the negative sampling procedure

Levy, Goldberg, Dagan (In submission)
Negative Sampling Smoothing

• Recall that SGNS picks \( w' \sim P \) to form negative \((w', c)\) examples

• Our analysis assumes \( P \) is the unigram distribution

\[
P (w) = \frac{\#w}{\sum_{w' \in V_w} \#w'}
\]
Negative Sampling Smoothing

• Recall that SGNS picks \( w' \sim P \) to form negative \((w', c)\) examples

• Our analysis assumes \( P \) is the unigram distribution

• In practice, it’s a **smoothed** unigram distribution

\[
P^{0.75}(w) = \frac{(\#w)^{0.75}}{\sum_{w' \in V_W} (#w')^{0.75}}
\]

• This little change makes a big difference

Levy, Goldberg, Dagan (In submission)
Negative Sampling Smoothing

• This smoothing has an analogue in PMI

• Replace $P(w)$ with $P^{0.75}(w)$:

$$PMI^{0.75}(w, c) = \log \frac{P(w, c)}{P^{0.75}(w)P(c)}$$

• Yields a **dramatic** improvement with **every method** on **every task**

Levy, Goldberg, Dagan (In submission)
Experiments & Results

• We compared several methods, while controlling for hyperparameters
  • PPMI, SVD(PPMI), SGNS, GloVe

• Methods perform on-par in most tasks
  • Slight advantage to SVD in word similarity
  • SGNS is better at syntactic analogies
  • SGNS is robust in general

• **Negative sampling smoothing** accounts for much of the differences observed in “Don’t count, predict!”
Other Hyperparameters

• There are many other hyperparameters that can be investigated

• Perhaps the most interesting one is the type of context
What’s in a Context?
What’s in a Context?

• Importing ideas from embeddings improves distributional methods

• Can distributional ideas also improve embeddings?

• **Idea:** change SGNS’s default BoW contexts into dependency contexts

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014
Example

Australian scientist discovers star with telescope

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014
Australian scientist discovers star with telescope
Bag of Words (BoW) Context

Australian scientist *discovers* star with telescope

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014
Bag of Words (BoW) Context

Australian scientist *discovers* star with *telescope*

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014
Australian scientist discovers star with telescope

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014
Syntactic Dependency Context

Australian scientist discovers star with telescope

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014
Syntactic Dependency Context

Australian scientist discovers star through telescope

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014
Syntactic Dependency Context

Australian scientist discovers star with telescope.

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014
## Embedding Similarity with Different Contexts

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

**Related to Harry Potter**

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014
### 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</td>
<td>Pauling</td>
</tr>
<tr>
<td></td>
<td>non-deterministic</td>
<td>Hotelling</td>
</tr>
<tr>
<td></td>
<td>computability</td>
<td>Heting</td>
</tr>
<tr>
<td></td>
<td>deterministic</td>
<td>Lessing</td>
</tr>
<tr>
<td></td>
<td>finite-state</td>
<td>Hamming</td>
</tr>
</tbody>
</table>

**Related to computability**

**Scientists**

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014
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

Gerunds

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014
What is the effect of different context types?

- Thoroughly studied in distributional methods
  - Lin (1998), Padó and Lapata (2007), and many others...

**General Conclusion:**
- Bag-of-words contexts induce *topical* similarities
- Dependency contexts induce *functional* similarities
  - Share the same semantic type
  - Cohyponyms

- Holds for *embeddings* as well

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014
Peeking into Skip-Gram’s Black Box

• In explicit representations, we can look at the features and analyze

• But embeddings are a black box!

• Dimensions are latent and don’t necessarily have any meaning

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014
Peeking into Skip-Gram’s Black Box

• Skip-Gram allows a peek...

• Contexts are embedded in the same space!

• Given a word $w$, find the contexts $c$ it “activates” most:

$$\arg\max_c (\vec{w} \cdot \vec{c})$$

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014
## Associated Contexts

<table>
<thead>
<tr>
<th>Target Word</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hogwarts</td>
<td>students/prep_at(^{-1})</td>
</tr>
<tr>
<td></td>
<td>educated/prep_at(^{-1})</td>
</tr>
<tr>
<td></td>
<td>student/prep_at(^{-1})</td>
</tr>
<tr>
<td></td>
<td>stay/prep_at(^{-1})</td>
</tr>
<tr>
<td></td>
<td>learned/prep_at(^{-1})</td>
</tr>
</tbody>
</table>

“Dependency-Based Word Embeddings”  
Levy & Goldberg, ACL 2014
## Associated Contexts

<table>
<thead>
<tr>
<th>Target Word</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turing</td>
<td>machine/nn⁻¹</td>
</tr>
<tr>
<td></td>
<td>test/nn⁻¹</td>
</tr>
<tr>
<td></td>
<td>theorem/poss⁻¹</td>
</tr>
<tr>
<td></td>
<td>machines/nn⁻¹</td>
</tr>
<tr>
<td></td>
<td>tests/nn⁻¹</td>
</tr>
</tbody>
</table>

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014
## Associated Contexts

<table>
<thead>
<tr>
<th>Target Word</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>dancing</td>
<td>dancing/conj</td>
</tr>
<tr>
<td></td>
<td>dancing/conj⁻¹</td>
</tr>
<tr>
<td></td>
<td>singing/conj⁻¹</td>
</tr>
<tr>
<td></td>
<td>singing/conj</td>
</tr>
<tr>
<td></td>
<td>ballroom/nn</td>
</tr>
</tbody>
</table>

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014
Analyzing Embeddings

• We show a way to linguistic analyze embeddings

• Together with the ability to engineer contexts...

• ...we now have the tools to create task-tailored embeddings!

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014
But there’s still one question left...

• How do you explain this?

\[ \text{king} - \text{man} + \text{woman} = \text{queen} \]
But there’s still one question left…

• How do you explain this?

Tokyo — Japan + France = Paris
But there’s still one question left...

• How do you explain this?

best — good + strong = strongest
Kings, Queens, and Vector Arithmetic
Analogies

*man* is to *woman* as *king* is to ?

- Mikolov et al.: analogies can be recovered by simple vector arithmetic
  
  \[ \text{king} - \text{man} + \text{woman} = \text{queen} \]

- Why does this work?

“Linguistic Regularities...”
Levy & Goldberg, CoNLL 2014
Why does vector arithmetic reveal analogies?

“Linguistic Regularities…”
Levy & Goldberg, CoNLL 2014
Why does vector arithmetic reveal analogies?

• We wish to find the closest $x$ to $king - man + woman$
Why does vector arithmetic reveal analogies?

• We wish to find the closest \( x \) to \( \text{king} - \text{man} + \text{woman} \)

• This is done with cosine similarity:

\[
\arg \max_x \cos(x, \text{king} - \text{man} + \text{woman})
\]

“Linguistic Regularities...”
Levy & Goldberg, CoNLL 2014
Why does vector arithmetic reveal analogies?

• We wish to find the closest $x$ to $king - man + woman$

• This is done with cosine similarity:

$$
\arg \max_x (\cos(x, king - man + woman)) = \\
\arg \max_x (\cos(x, king) - \cos(x, man) + \cos(x, woman))
$$

“Linguistic Regularities...”
Levy & Goldberg, CoNLL 2014
Why does vector arithmetic reveal analogies?

• We wish to find the closest $x$ to $king - man + woman$
• This is done with cosine similarity:

$$\arg \max_x \cos(x, king - man + woman) =$$

$$\arg \max_x (\cos(x, king) - \cos(x, man) + \cos(x, woman))$$

vector arithmetic = similarity arithmetic

“Linguistic Regularities...”
Levy & Goldberg, CoNLL 2014
Why does vector arithmetic reveal analogies?

• We wish to find the closest $x$ to $\textit{king} - \textit{man} + \textit{woman}$
• This is done with cosine similarity:

$$\arg\max_x \cos(x, \textit{king} - \textit{man} + \textit{woman}) = \arg\max_x \cos(x, \textit{king}) - \cos(x, \textit{man}) + \cos(x, \textit{woman})$$

vector arithmetic = similarity arithmetic

“Linguistic Regularities…”
Levy & Goldberg, CoNLL 2014
Why does vector arithmetic reveal analogies?

• We wish to find the closest $x$ to $king - man + woman$

• This is done with cosine similarity:

$$\arg \max_x \cos(x, king - man + woman) = \arg \max_x (\cos(x, king) - \cos(x, man) + \cos(x, woman))$$

vector arithmetic = similarity arithmetic

“Linguistic Regularities…”
Levy & Goldberg, CoNLL 2014
What does each similarity term mean?

- Observe the joint features with explicit representations!

<table>
<thead>
<tr>
<th>$queen \cap king$</th>
<th>$queen \cap woman$</th>
</tr>
</thead>
<tbody>
<tr>
<td>uncrowned</td>
<td>Elizabeth</td>
</tr>
<tr>
<td>majesty</td>
<td>Katherine</td>
</tr>
<tr>
<td>second</td>
<td>impregnate</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Can we do better?

“Linguistic Regularities...”
Levy & Goldberg, CoNLL 2014
Let’s look at some mistakes...
Let’s look at some mistakes...

England − London + Baghdad = ?

“Linguistic Regularities...”
Levy & Goldberg, CoNLL 2014
Let’s look at some mistakes...

England $-$ London $+$ Baghdad $=$ Iraq

“Linguistic Regularities…”
Levy & Goldberg, CoNLL 2014
Let’s look at some mistakes...

England − London + Baghdad = Mosul?

“Linguistic Regularities...”
Levy & Goldberg, CoNLL 2014
The Additive Objective

$$\cos(Iraq, England) - \cos(Iraq, London) + \cos(Iraq, Baghdad)$$

$$\cos(Mosul, England) - \cos(Mosul, London) + \cos(Mosul, Baghdad)$$

“Linguistic Regularities...”
Levy & Goldberg, CoNLL 2014
The Additive Objective

\[
\cos(Iraq, \text{England}) - \cos(Iraq, \text{London}) + \cos(Iraq, \text{Baghdad}) = 0.15
\]

\[
\cos(Mosul, \text{England}) - \cos(Mosul, \text{London}) + \cos(Mosul, \text{Baghdad}) = 0.13
\]

“Linguistic Regularities...”
Levy & Goldberg, CoNLL 2014
The Additive Objective

\[
\cos(Iraq, England) - \cos(Iraq, London) + \cos(Iraq, Baghdad)
\]

0.15  

0.13  

\[
\cos(Mosul, England) - \cos(Mosul, London) + \cos(Mosul, Baghdad)
\]

0.13  

0.14  

“Linguistic Regularities...”
Levy & Goldberg, CoNLL 2014
The Additive Objective

\[
\cos(\text{Iraq, England}) - \cos(\text{Iraq, London}) + \cos(\text{Iraq, Baghdad})
\]

\[
0.15 - 0.13 + 0.63 = 0.65
\]

\[
\cos(\text{Mosul, England}) - \cos(\text{Mosul, London}) + \cos(\text{Mosul, Baghdad})
\]

\[
0.13 - 0.14 + 0.75 = 0.74
\]

“Linguistic Regularities...”
Levy & Goldberg, CoNLL 2014
The Additive Objective

\[
\cos(\text{Iraq, England}) - \cos(\text{Iraq, London}) + \cos(\text{Iraq, Baghdad})
\]

0.15 \quad 0.13 \quad 0.63 = 0.65

\[
\cos(\text{Mosul, England}) - \cos(\text{Mosul, London}) + \cos(\text{Mosul, Baghdad})
\]

0.13 \quad 0.14 \quad 0.75 = 0.74

“Linguistic Regularities...”
Levy & Goldberg, CoNLL 2014
The Additive Objective

\[
\cos(Iraq, England) - \cos(Iraq, London) + \cos(Iraq, Baghdad)
\]

\[
0.15 - 0.13 + 0.63 = 0.65
\]

\[
\cos(Mosul, England) - \cos(Mosul, London) + \cos(Mosul, Baghdad)
\]

\[
0.13 - 0.14 + 0.75 = 0.74
\]

• **Problem:** one similarity might dominate the rest

“Linguistic Regularities...”
Levy & Goldberg, CoNLL 2014
How can we do better?

“Linguistic Regularities...”
Levy & Goldberg, CoNLL 2014
How can we do better?

• Instead of **adding** similarities, **multiply** them!

“Linguistic Regularities...”
Levy & Goldberg, CoNLL 2014
How can we do better?

• Instead of adding similarities, multiply them!

\[
\arg\max_x \left( \frac{\cos(x, \text{king}) \cos(x, \text{woman})}{\cos(x, \text{man})} \right)
\]

“Linguistic Regularities...”
Levy & Goldberg, CoNLL 2014
How can we do better?

• Instead of adding similarities, multiply them!

$$\arg \max_x \left( \frac{\cos(x, \text{king}) \cos(x, \text{woman})}{\cos(x, \text{man})} \right)$$

“Linguistic Regularities...”
Levy & Goldberg, CoNLL 2014
Multiplication > Addition

“Linguistic Regularities…”
Levy & Goldberg, CoNLL 2014
Kings, Queens, and Vector Arithmetic

• Why does vector arithmetic reveal analogies?
Because *vector arithmetic* is equivalent to *similarity arithmetic*.

• We can improve analogy recovery with the *multiplicative objective*
Conclusion
To Summarize

- In order to communicate...
- ...we need to understand language.
- The building blocks:
  - Understanding structure.
  - Understanding words.
- Representing words as vectors can go a long way
- ...but shouldn’t be treated as magical.
- By understanding what’s going on behind the scenes, one can improve and control the behavior of the models.
  - better analogical reasoning.
  - syntactic contexts → more functional similarities.
- Understanding language is hard. Still a long way to go.
Thanks — for + listening = )