Natural Language Processing with Deep Learning

CS224N/Ling284

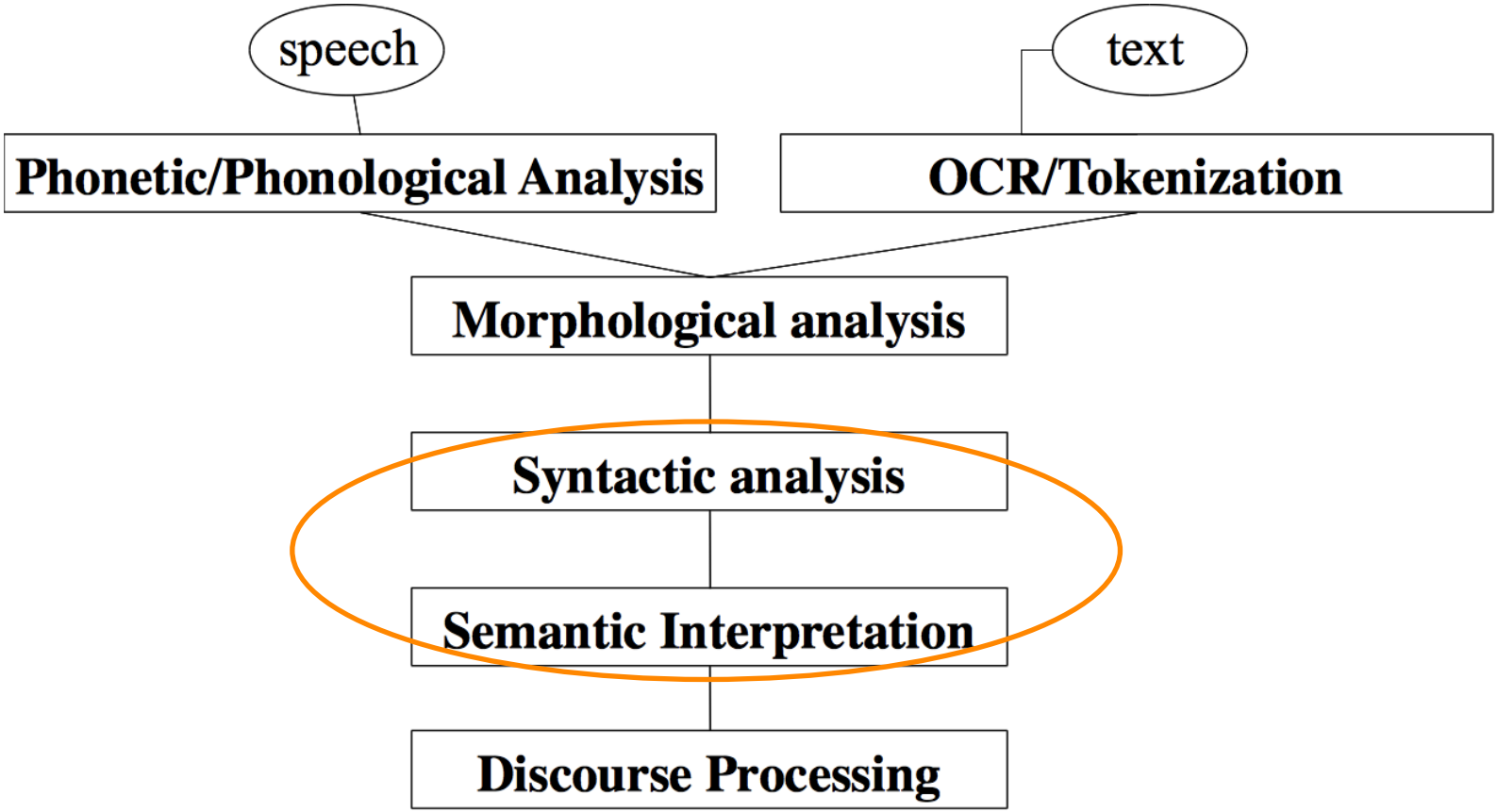
Richard Socher

Lecture 1: Introduction

1. What is Natural Language Processing (NLP)?

- **Natural language processing** is a field at the intersection of
  - computer science
  - artificial intelligence
  - and linguistics.
- **Goal:** for computers to process or "understand" natural language in order to perform tasks that are useful, e.g.,
  - Performing Tasks, like making appointments, buying things
  - Language translation
  - Question Answering
    - Siri, Google Assistant, Facebook M, Cortana …
- Fully **understanding and representing** the **meaning** of language (or even defining it) is a difficult goal.
- Perfect language understanding is AI-complete

NLP Levels


speech
text
Phonetic/Phonological Analysis
OCR/Tokenization
Morphological analysis
Syntactic analysis
Semantic Interpretation
Discourse Processing

(A tiny sample of) NLP Applications

Applications range from simple to complex:

- Spell checking, keyword search, finding synonyms

- Extracting information from websites such as
  - product price, dates, location, people or company names
- Classifying: reading level of school texts, positive/negative sentiment of longer documents

- Machine translation
- Spoken dialog systems
- Complex question answering

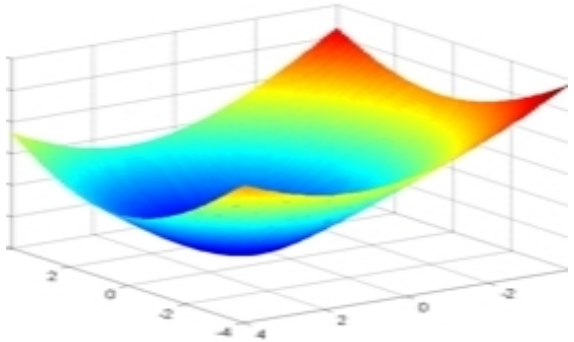
2. What's Deep Learning (DL)?

- **Deep learning** is a subfield of **machine learning**

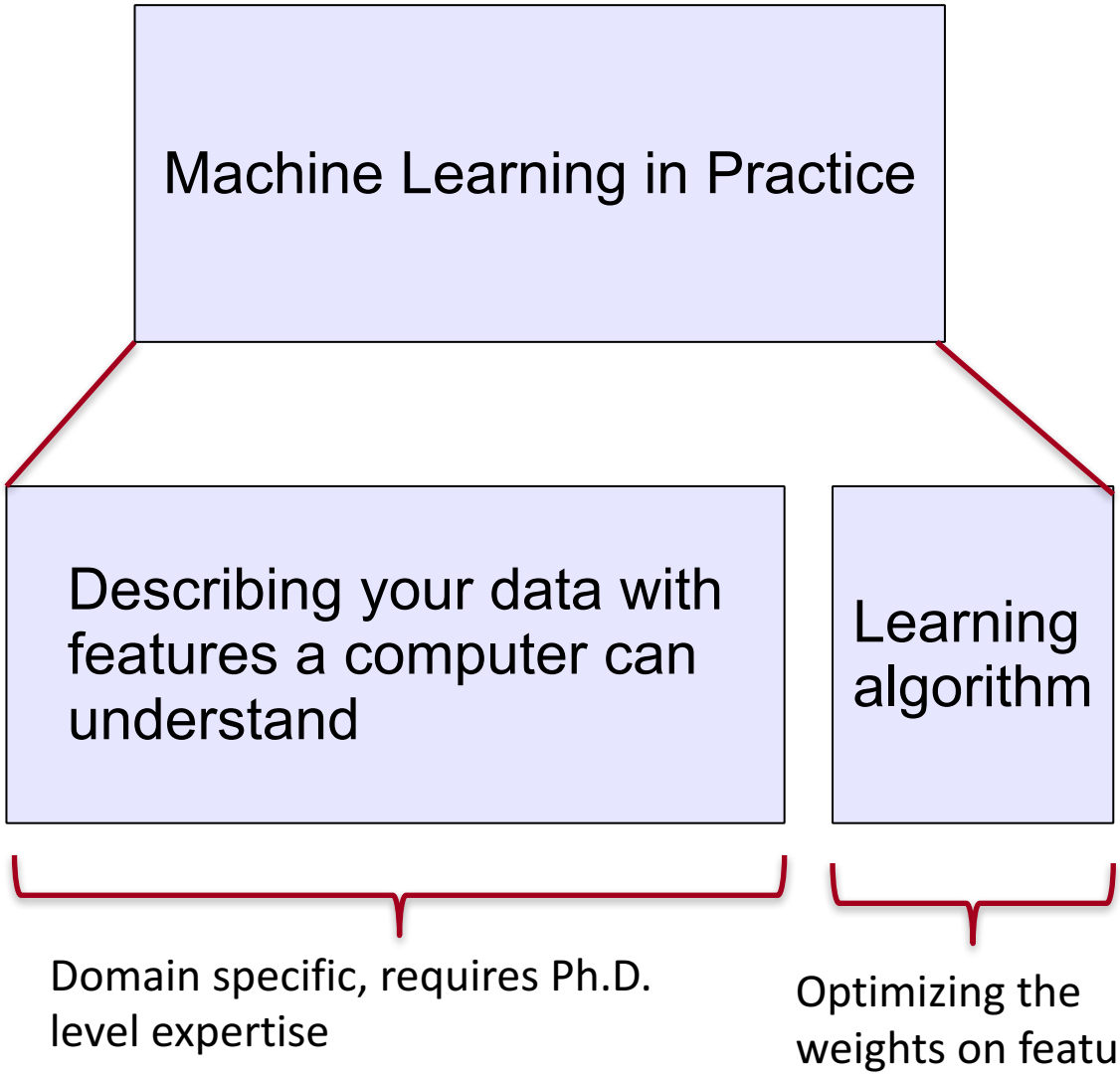
- Most machine learning methods work well because of **human-designed representations** and **input features**
  - For example: features for finding named entities like locations or organization names (Finkel et al., 2010):

Feature	NER
Current Word	✓
Previous Word	✓
Next Word	✓
Current Word Character n-gram	all
Current POS Tag	✓
Surrounding POS Tag Sequence	✓
Current Word Shape	✓
Surrounding Word Shape Sequence	✓
Presence of Word in Left Window	size 4
Presence of Word in Right Window	size 4

- Machine learning becomes just optimizing weights to best make a final prediction

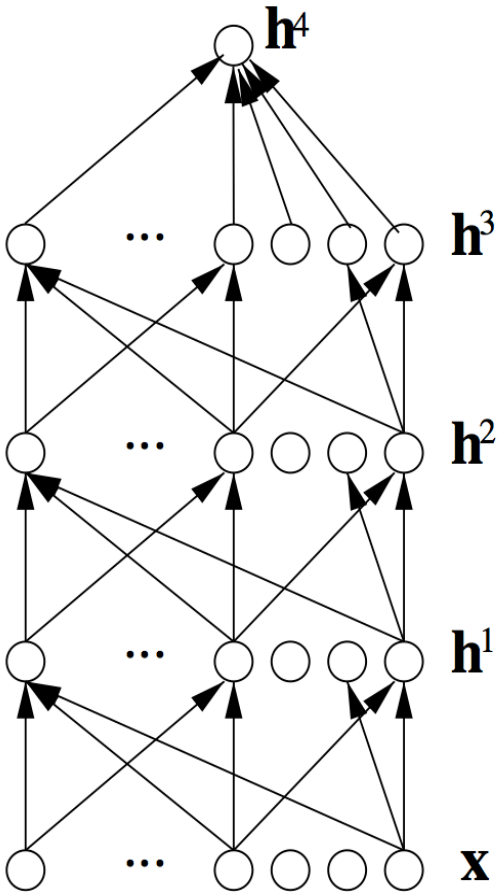


Machine Learning vs. Deep Learning



What's Deep Learning (DL)?

- In contrast to standard machine learning,
- Representation learning attempts to automatically learn good features or representations
- Deep learning algorithms attempt to learn (multiple levels of) representations (here: h^1,h^2,h^3) and an output (h^4)
- From "raw" inputs **x** (e.g. sound, pixels, characters, or words)



Reasons for Exploring Deep Learning

- Manually designed features are often over-specified, incomplete and take a long time to design and validate

- **Learned Features** are easy to adapt, fast to learn

- Deep learning provides a very flexible, (almost?) universal, learnable framework for **representing** world, visual and linguistic information.

- Deep learning can learn **unsupervised** (from raw text) and **supervised** (with specific labels like positive/negative)

Reasons for Exploring Deep Learning

- In ~2010 **deep** learning techniques started outperforming other machine learning techniques. Why this decade?

- Large amounts of training data favor deep learning
- Faster machines and multicore CPU/GPUs favor Deep Learning
- New models, algorithms, ideas
  - Better, more flexible learning of intermediate representations
  - Effective end-to-end joint system learning
  - Effective learning methods for using contexts and transferring between tasks
  - Better regularization and optimization methods

→ **Improved performance** (first in speech and vision, then NLP)

4. Why is NLP hard?

- Complexity in representing, learning and using linguistic/situational/contextual/world/visual knowledge
- But interpretation depends on these

- Human languages are ambiguous (unlike programming and other formal languages)

- E.g. “I made her duck.”

Why NLP is difficult: Real newspaper headlines/tweets

1. The Pope's baby steps on gays
2. Boy paralyzed after tumor fights back to gain black belt
3. Enraged cow injures farmer with axe
4. Juvenile Court to Try Shooting Defendant

5. Deep NLP = Deep Learning + NLP

Combine ideas and goals of NLP with using representation learning and deep learning methods to solve them

Several big improvements in recent years in NLP

- **Linguistic levels**: (speech), words, syntax, semantics
- **Intermediate tasks/tools:** parts-of-speech, entities, parsing
- **Full applications**: sentiment analysis, question answering, dialogue agents, machine translation

Word meaning as a neural word vector – visualization

expect =

$$\begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$

Word similarities

Nearest words to frog:

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

litoria

leptodactylidae

rana

eleutherodactylus

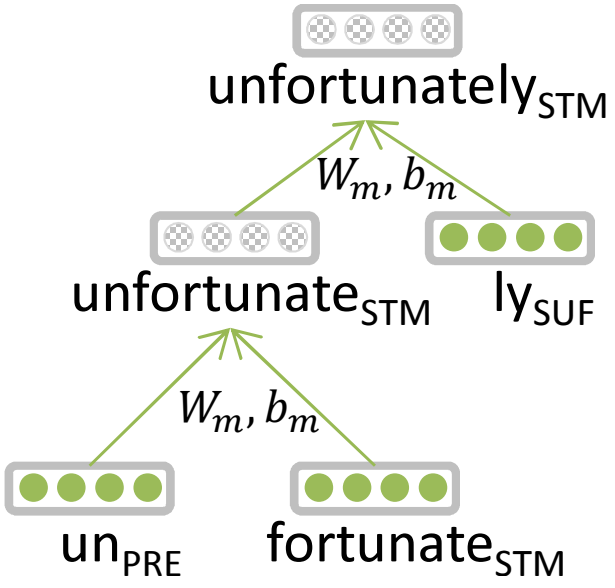
http://nlp.stanford.edu/projects/glove/

Representations of NLP Levels: Morphology

- Traditional: Words are made of morphemes

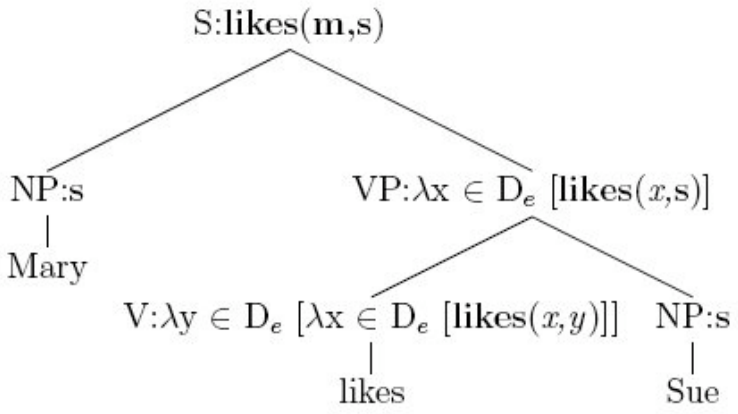
prefix	stem	suffix
un	interest	ed

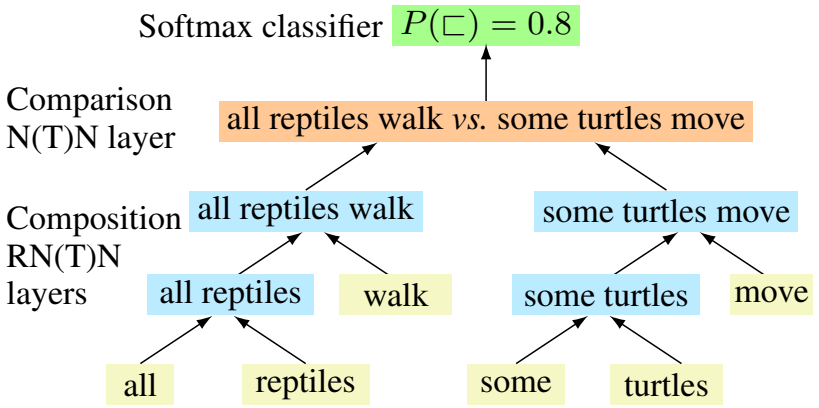
- DL:
  - every morpheme is a vector
  - a neural network combines two vectors into one vector
  - Luong et al. 2013



Representations of NLP Levels: Semantics

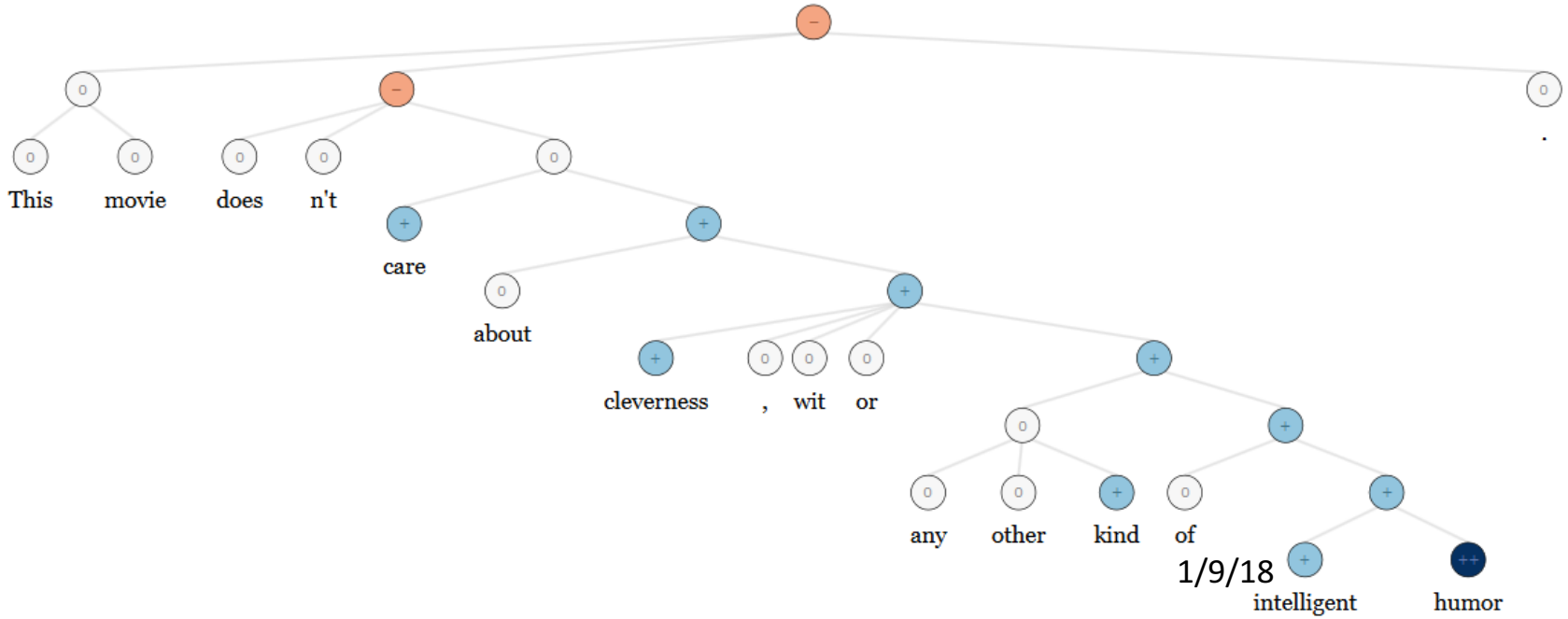
- Traditional: Lambda calculus
  - Carefully engineered functions
  - Take as inputs specific other functions
  - No notion of similarity or fuzziness of language
- DL:
  - Every word and every phrase and every logical expression is a vector
  - a neural network combines two vectors into one vector
  - Bowman et al. 2014





NLP Applications: Sentiment Analysis

- Traditional: Treat sentence as a bag-of-words (ignore word order); consult a curated list of "positive" and "negative" words to determine sentiment of sentence. Need hand-designed features to capture negation! --> Ain't gonna capture everything 😖
- Same deep learning model that could be used for morphology, syntax and logical semantics → RecursiveNN (aka TreeRNNs)



Question Answering

- Traditional: A lot of feature engineering to capture world and other knowledge, e.g., regular expressions, Berant et al. (2014)

Is main verb trigger?

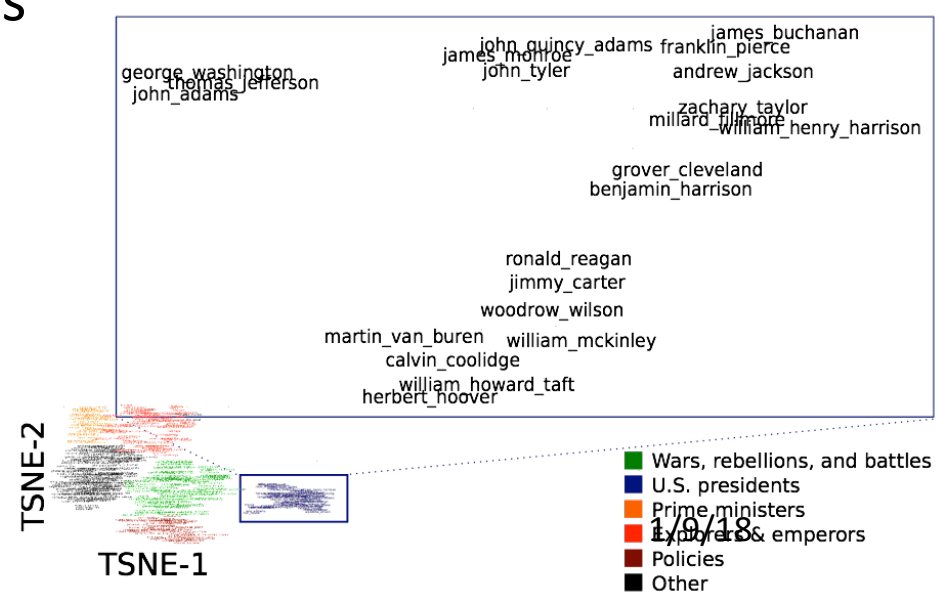
Yes

Condition	Regular Exp.
Wh- word subjective?	AGENT
Wh- word object?	THEME

No

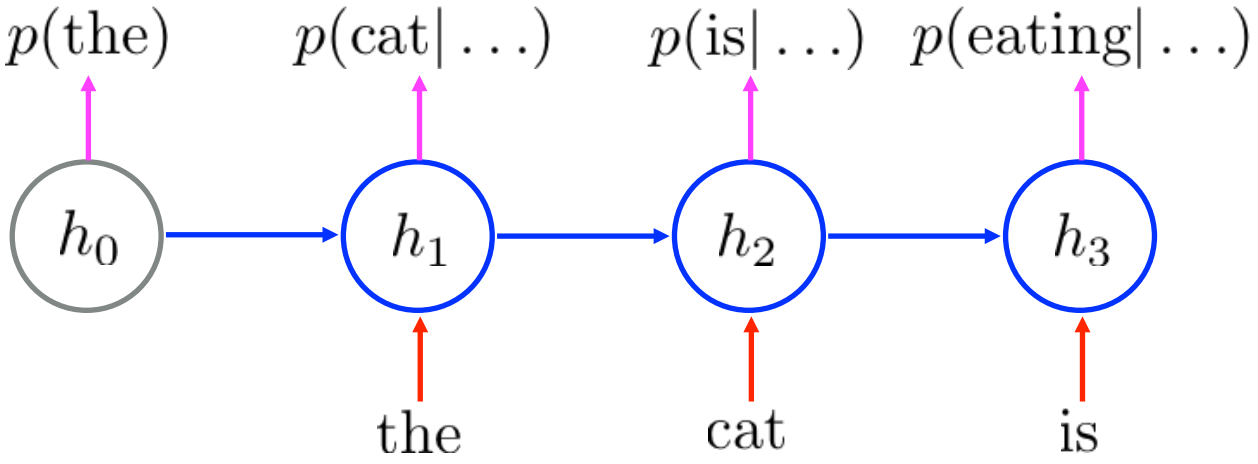
Condition	Regular Exp.
default	(ENABLE|SUPER)$^+$
DIRECT	(ENABLE|SUPER)
PREVENT	(ENABLE|SUPER)* PREVENT(ENABLE|SUPER)*

- DL: Again, a deep learning architecture can be used!
- Facts are stored in vectors



Dialogue agents / Response Generation

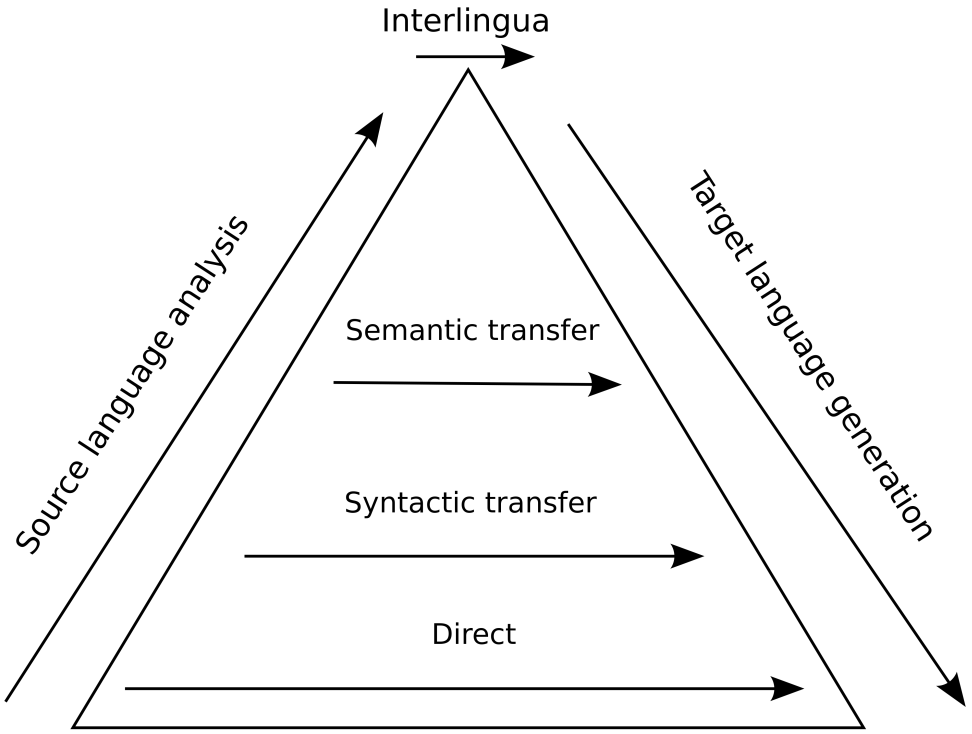
- A simple, successful example is the auto-replies available in the Google Inbox app
- An application of the powerful, general technique of **Neural Language Models**, which are an instance of Recurrent Neural Networks



Machine Translation

- Many levels of translation have been tried in the past:

- Traditional MT systems are very large complex systems

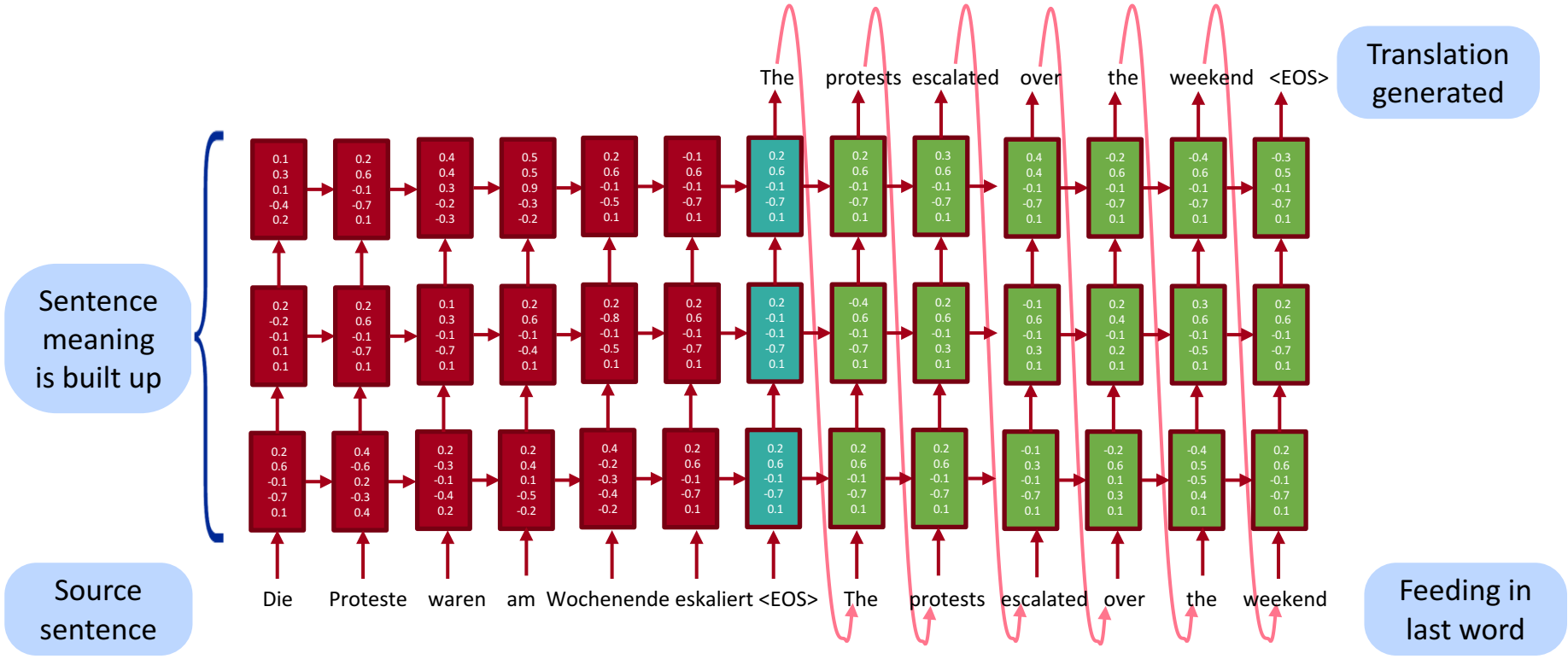


- What do you think is the interlingua for the DL approach to translation?

Neural Machine Translation

Source sentence is mapped to **vector**, then output sentence generated

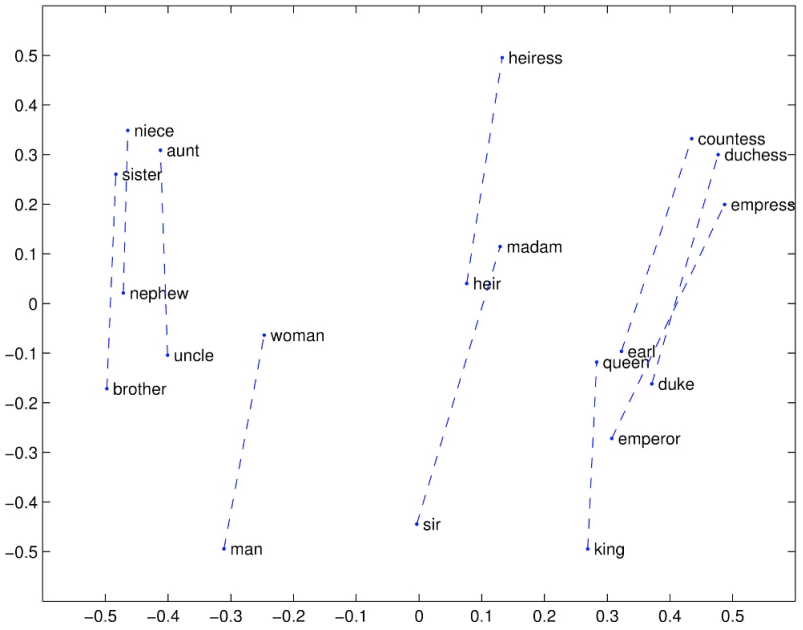
[Sutskever et al. 2014, Bahdanau et al. 2014, Luong and Manning 2016]



Now live for some languages in Google Translate (etc.), with big error reductions!

Conclusion: Representation for all levels? Vectors

We will study in the next lecture how we can learn vector representations for words and what they actually **represent.**



Next week: how neural networks work and how they can use these vectors for all NLP levels and many different applications

Representing words as discrete symbols

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

Means one 1, the rest 0s

Words can be represented by one-hot vectors:

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

Vector dimension = number of words in vocabulary (e.g., 500,000)

Problem with words as discrete symbols

Example: in web search, if user searches for "Seattle motel", we would like to match documents containing "Seattle hotel".

But:

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

These two vectors are orthogonal.

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

Solution:

- Could try to rely on WordNet's list of synonyms to get similarity?
  - But it is well-known to fail badly: incompleteness, etc.
- **Instead: learn to encode similarity in the vectors themselves**

Representing words by their context

- Distributional semantics: **A word's meaning is given by the words that frequently appear close-by**
  - *"You shall know a word by the company it keeps"* (J. R. Firth 1957: 11)
  - One of the most successful ideas of modern statistical NLP!
- 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 a representation of *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***

3. But why not capture co-occurrence counts directly?

With a co-occurrence matrix *X*

- 2 options: windows vs. full document
- Window: Similar to word2vec, use window around each word → captures both syntactic (POS) and semantic information
- Word-document co-occurrence matrix will give general topics (all sports terms will have similar entries) leading to "Latent Semantic Analysis"

Example: Window based co-occurrence matrix

- Window length 1 (more common: 5–10)
- Symmetric (irrelevant whether left or right context)
- Example corpus:
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.

Window based co-occurrence matrix

- Example corpus:
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Problems with simple co-occurrence vectors

Increase in size with vocabulary

Very high dimensional: requires a lot of storage

Subsequent classification models have sparsity issues

→ Models are less robust

Solution: Low dimensional vectors

- Idea: store “most” of the important information in a fixed, small number of dimensions: a dense vector

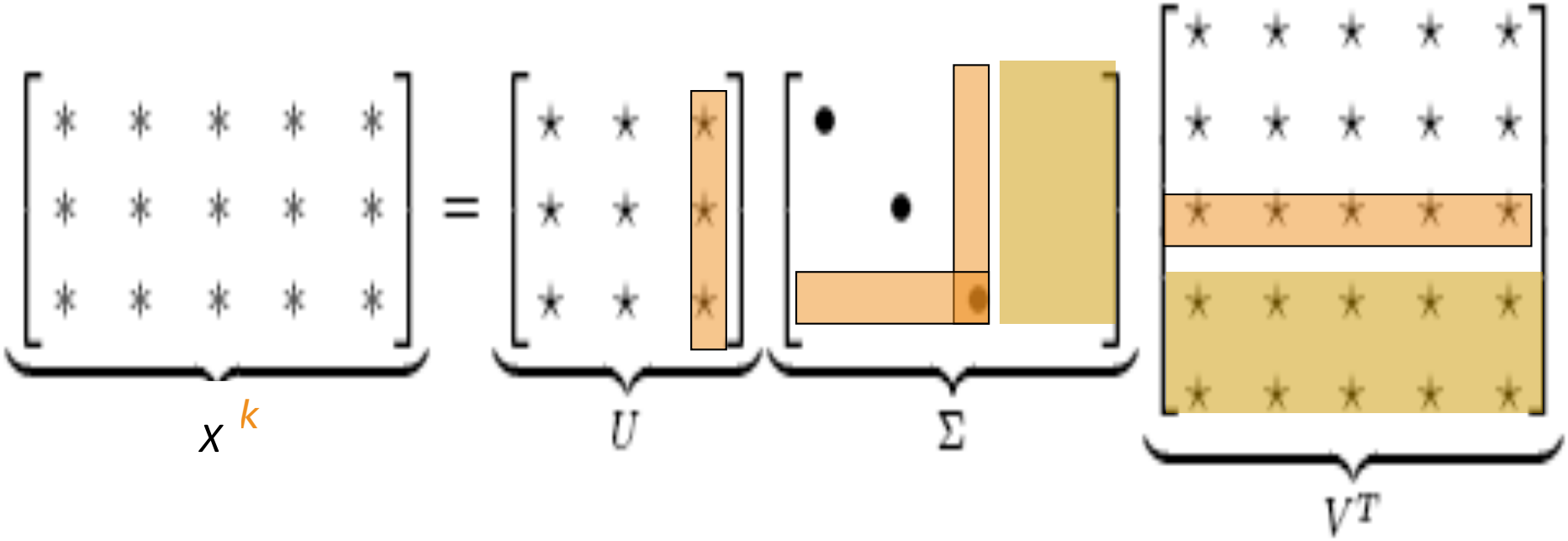
- Usually 25–1000 dimensions, similar to word2vec

- How to reduce the dimensionality?

Method 1: Dimensionality Reduction on X (HW1)

Singular Value Decomposition of co-occurrence matrix *X*

Factorizes *X* into $U\Sigma V^T$, where *U* and *V* are orthonormal



Retain only *k* singular values, in order to generalize.

$\hat{X}$ is the best rank *k* approximation to *X* , in terms of least squares.

Classic linear algebra result. Expensive to compute for large matrices.

Simple SVD word vectors in Python

Corpus:
I like deep learning. I like NLP. I enjoy flying.

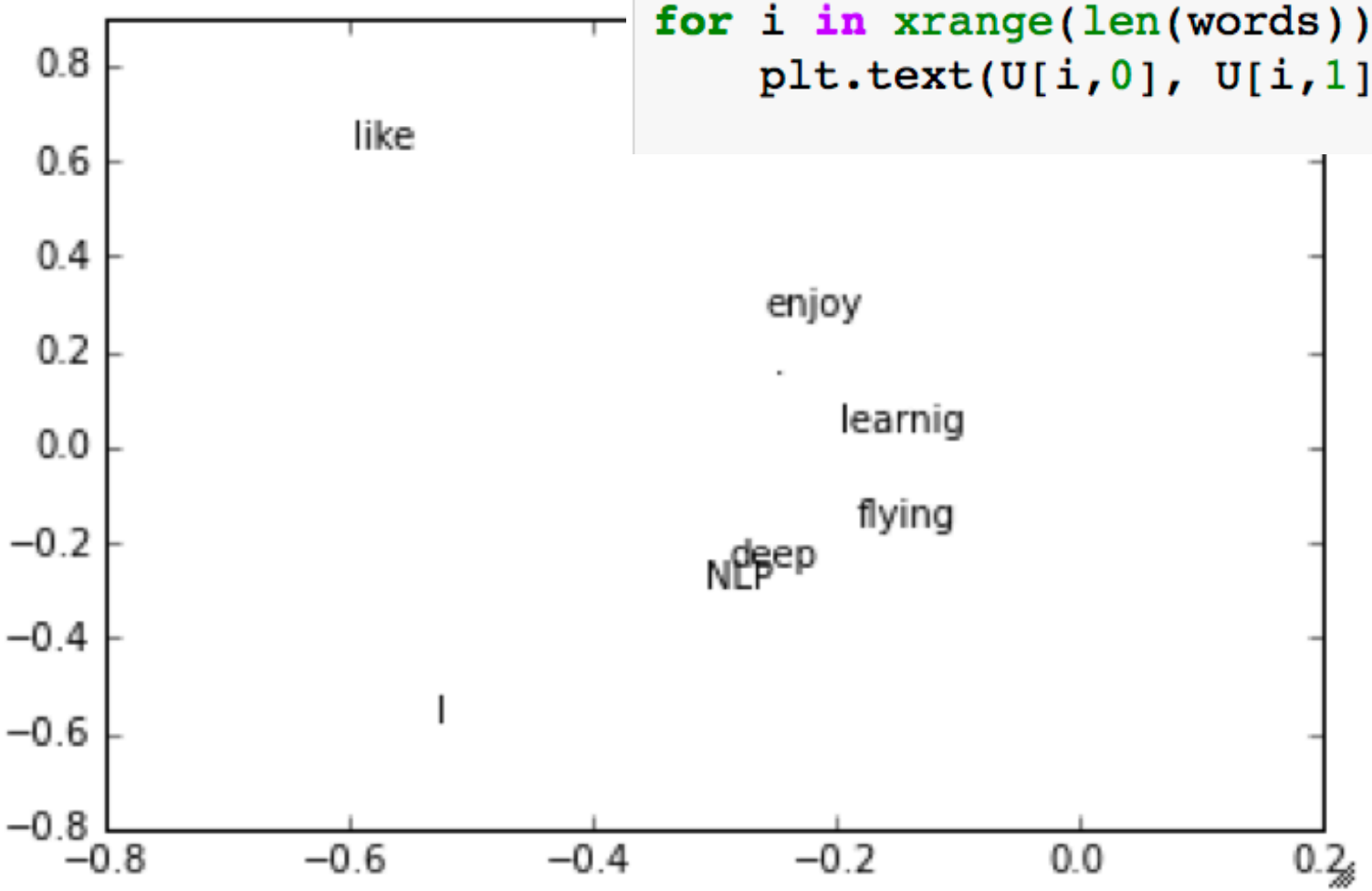
```
import numpy as np
la = np.linalg
words = ["I", "like", "enjoy",
         "deep","learnig","NLP","flying","."]
X = np.array([[0,2,1,0,0,0,0,0],
              [2,0,0,1,0,1,0,0],
              [1,0,0,0,0,0,1,0],
              [0,1,0,0,1,0,0,0],
              [0,0,0,1,0,0,0,1],
              [0,1,0,0,0,0,0,1],
              [0,0,1,0,0,0,0,1],
              [0,0,0,0,1,1,1,0]])

U, s, Vh = la.svd(X, full_matrices=False)
```

Simple SVD word vectors in Python

Corpus: I like deep learning. I like NLP. I enjoy flying.
Printing first two columns of U corresponding to the 2 biggest singular values

```
for i in xrange(len(words)):
    plt.text(U[i,0], U[i,1], words[i])
```

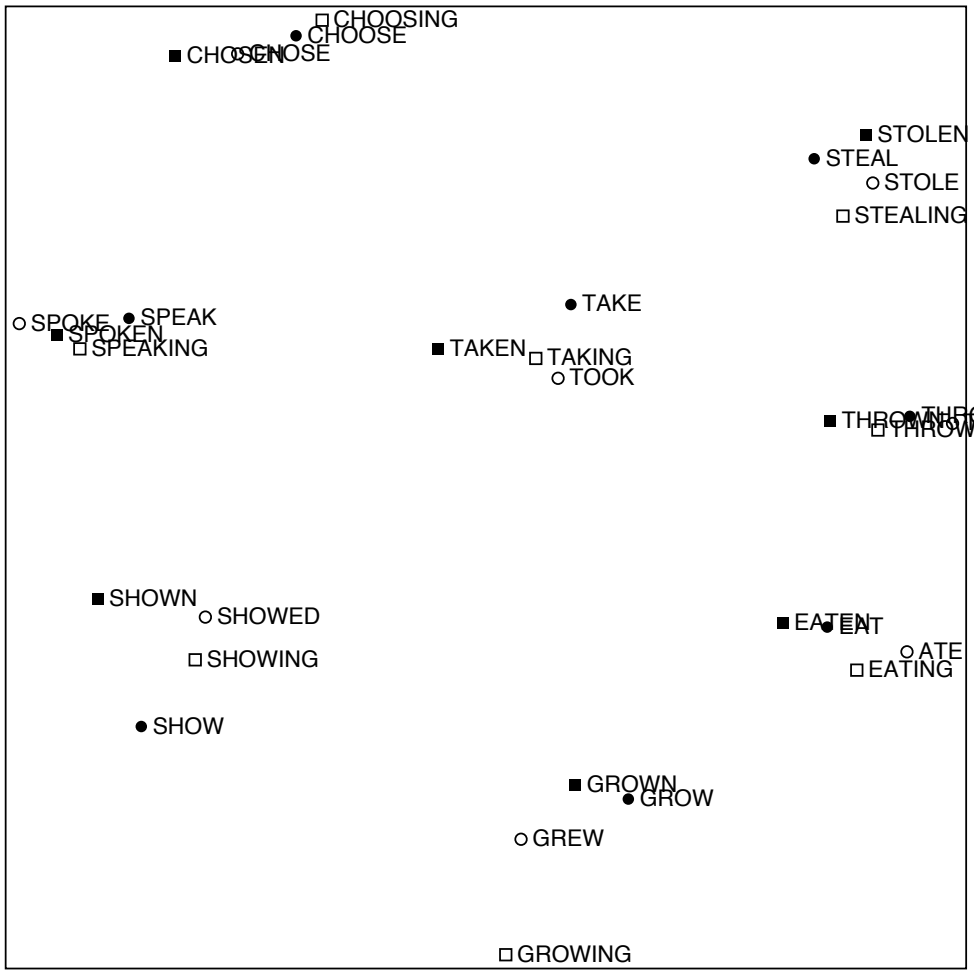


Hacks to X (several used in Rohde et al. 2005)

Scaling the counts in the cells can help ***a lot***

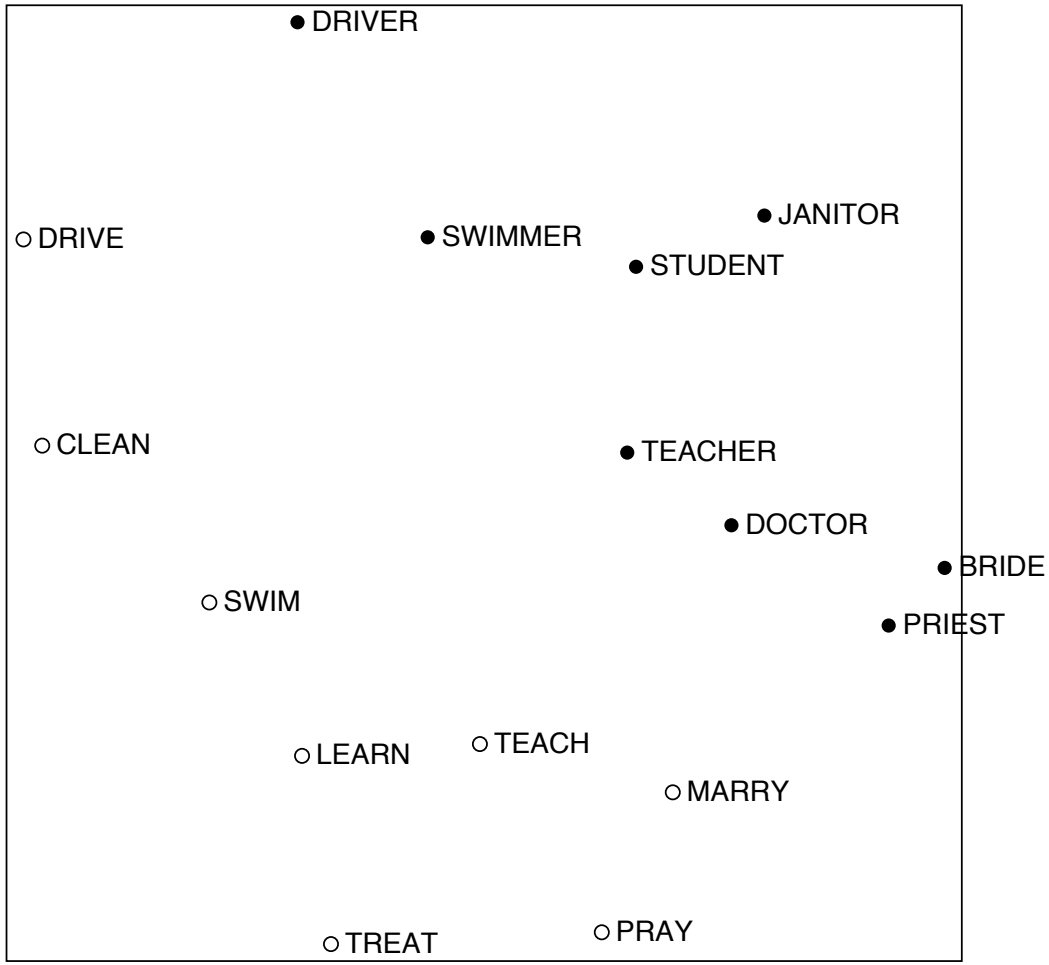
- Problem: function words (*the, he, has*) are too frequent → syntax has too much impact. Some fixes:
  - min(X,t), with t ≈ 100
  - Ignore them all
- Ramped windows that count closer words more
- Use Pearson correlations instead of counts, then set negative values to 0
- Etc.

Interesting syntactic patterns emerge in the vectors



COALS model from
An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence
Rohde et al. ms., 2005

Interesting semantic patterns emerge in the vectors



COALS model from
An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence
Rohde et al. ms., 2005

Count based vs. direct prediction

- LSA, HAL (Lund & Burgess),
- COALS, Hellinger-PCA (Rohde et al, Lebret & Collobert)

- Fast training
- Efficient usage of statistics
- Primarily used to capture word similarity
- Disproportionate importance given to large counts

- Skip-gram/CBOW (Mikolov et al)
- NNLM, HLBL, RNN (Bengio et al; Collobert & Weston; Huang et al; Mnih & Hinton)

- Scales with corpus size
- Inefficient usage of statistics
- Generate improved performance on other tasks
- Can capture complex patterns beyond word similarity

Encoding meaning in vector differences

[Pennington, Socher, and Manning, EMNLP 2014]

Crucial insight: Ratios of co-occurrence probabilities can encode meaning components

	x = solid	x = gas	x = water	x = random
$P(x\|\text{ice})$	large	small	large	small
$P(x\|\text{steam})$	small	large	large	small
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	large	small	~1	~1

Encoding meaning in vector differences

[Pennington, Socher, and Manning, EMNLP 2014]

Crucial insight: Ratios of co-occurrence probabilities can encode meaning components

	x = solid	x = gas	x = water	x = fashion
$P(x\|\text{ice})$	1.9 x 10^{-4}	6.6 x 10^{-5}	3.0 x 10^{-3}	1.7 x 10^{-5}
$P(x\|\text{steam})$	2.2 x 10^{-5}	7.8 x 10^{-4}	2.2 x 10^{-3}	1.8 x 10^{-5}
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	8.9	8.5 x 10^{-2}	1.36	0.96

Encoding meaning in vector differences

Q: How can we capture ratios of co-occurrence probabilities as linear meaning components in a word vector space?

A: Log-bilinear model: $w_i \cdot w_j = \log P(i|j)$

with vector differences $w_x \cdot (w_a - w_b) = \log \frac{P(x|a)}{P(x|b)}$

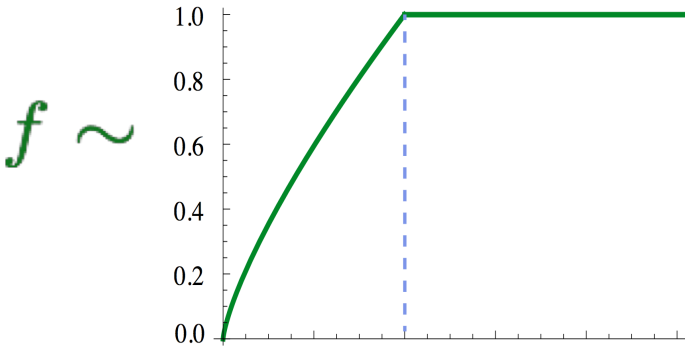
Combining the best of both worlds

GloVe [Pennington et al., EMNLP 2014]

$$w_i \cdot w_j = \log P(i|j)$$

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

- Fast training
- Scalable to huge corpora
- Good performance even with small corpus and small vectors



GloVe results

Nearest words to frog:

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

litoria

leptodactylidae

rana

eleutherodactylus

How to evaluate word vectors?

- Related to general evaluation in NLP: Intrinsic vs extrinsic
- Intrinsic:
  - Evaluation on a specific/intermediate subtask
  - Fast to compute
  - Helps to understand that system
  - Not clear if really helpful unless correlation to real task is established
- Extrinsic:
  - Evaluation on a real task
  - Can take a long time to compute accuracy
  - Unclear if the subsystem is the problem or its interaction or other subsystems
  - If replacing exactly one subsystem with another improves accuracy → Winning!

Intrinsic word vector evaluation

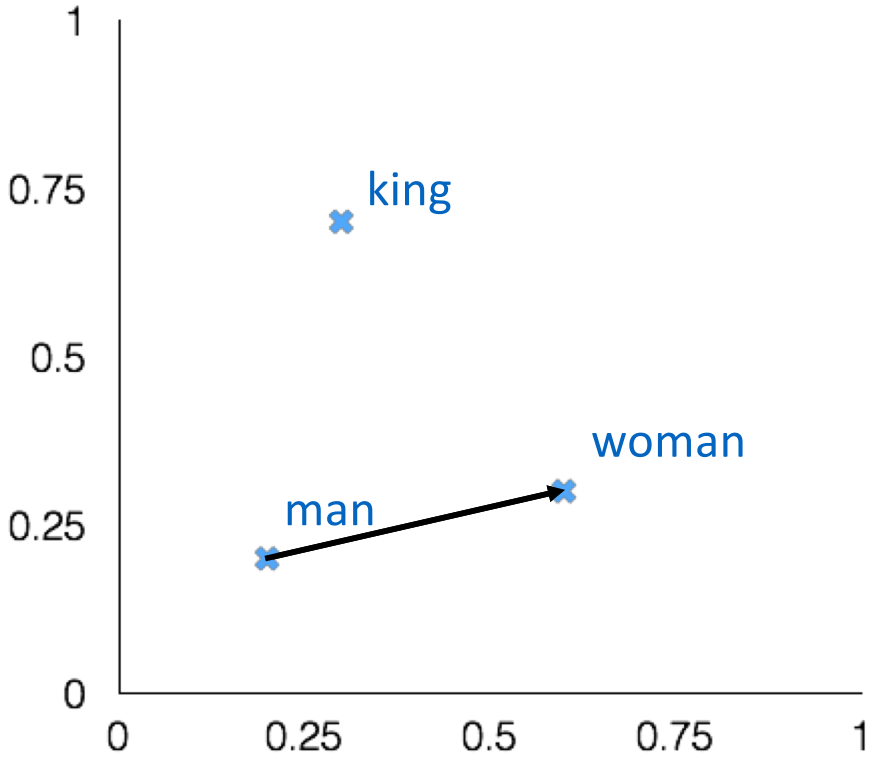
- Word Vector Analogies

a:b :: c:?

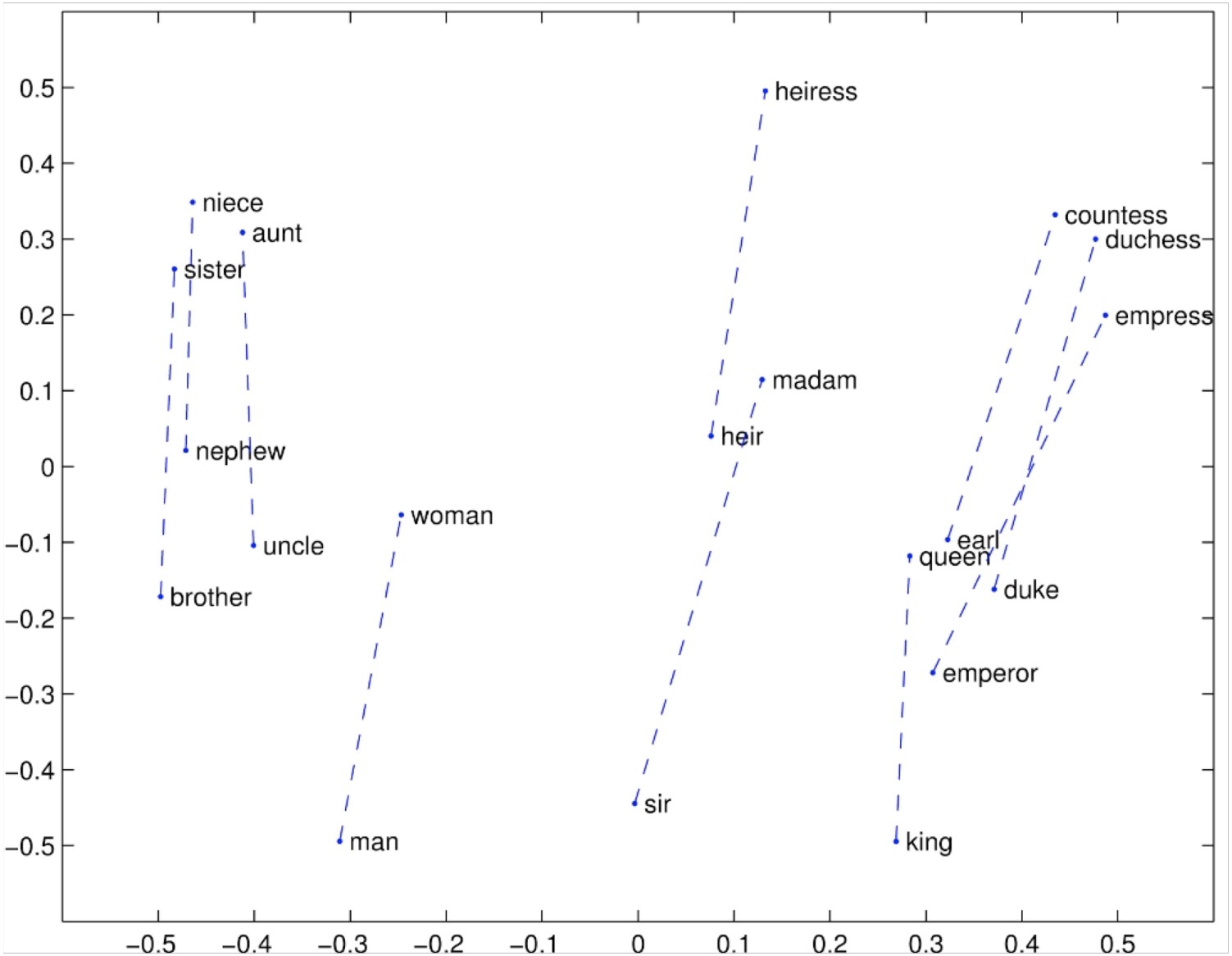
man:woman :: king:?

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

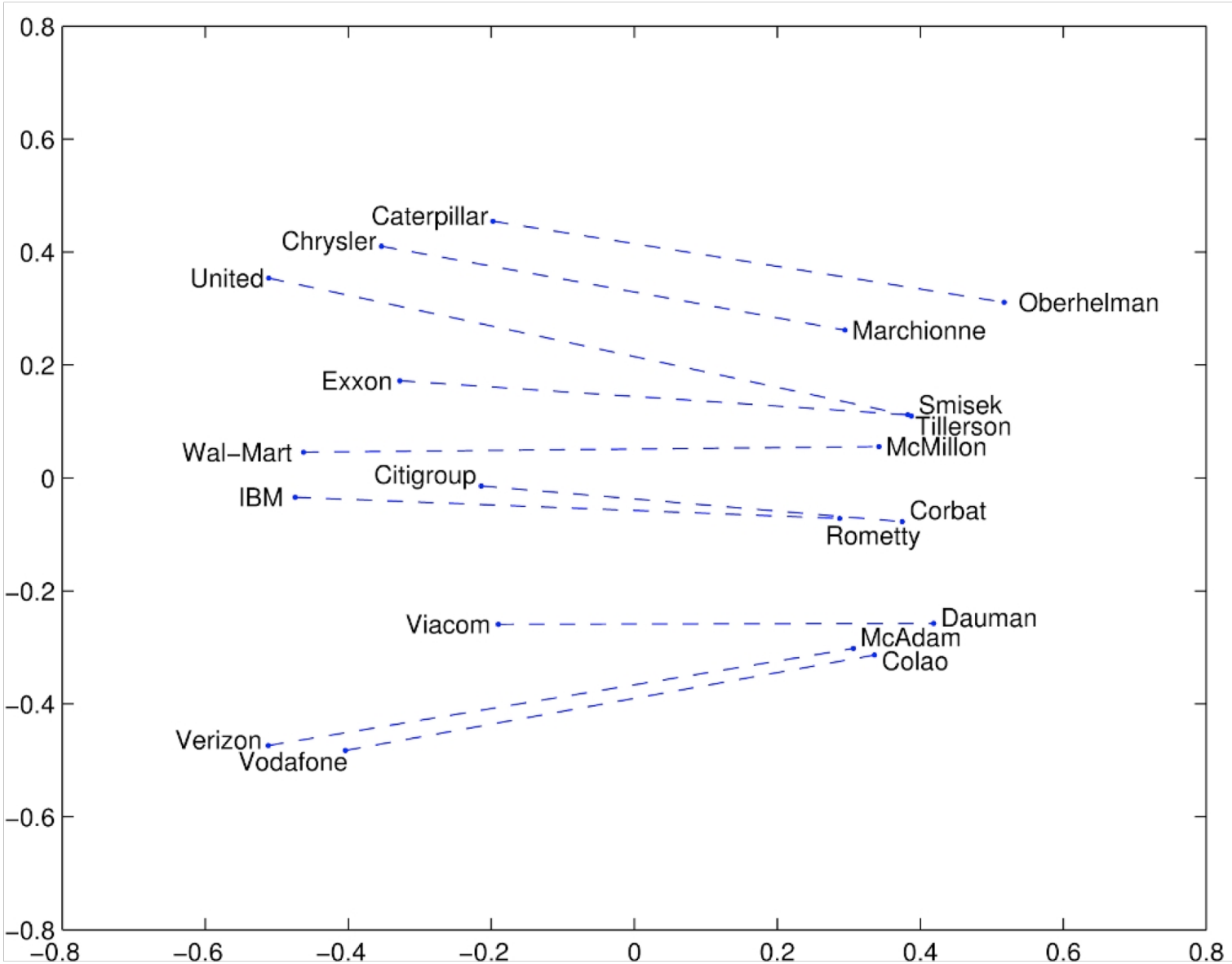
- Evaluate word vectors by how well their cosine distance after addition captures intuitive semantic and syntactic analogy questions
- Discarding the input words from the search!
- Problem: What if the information is there but not linear?



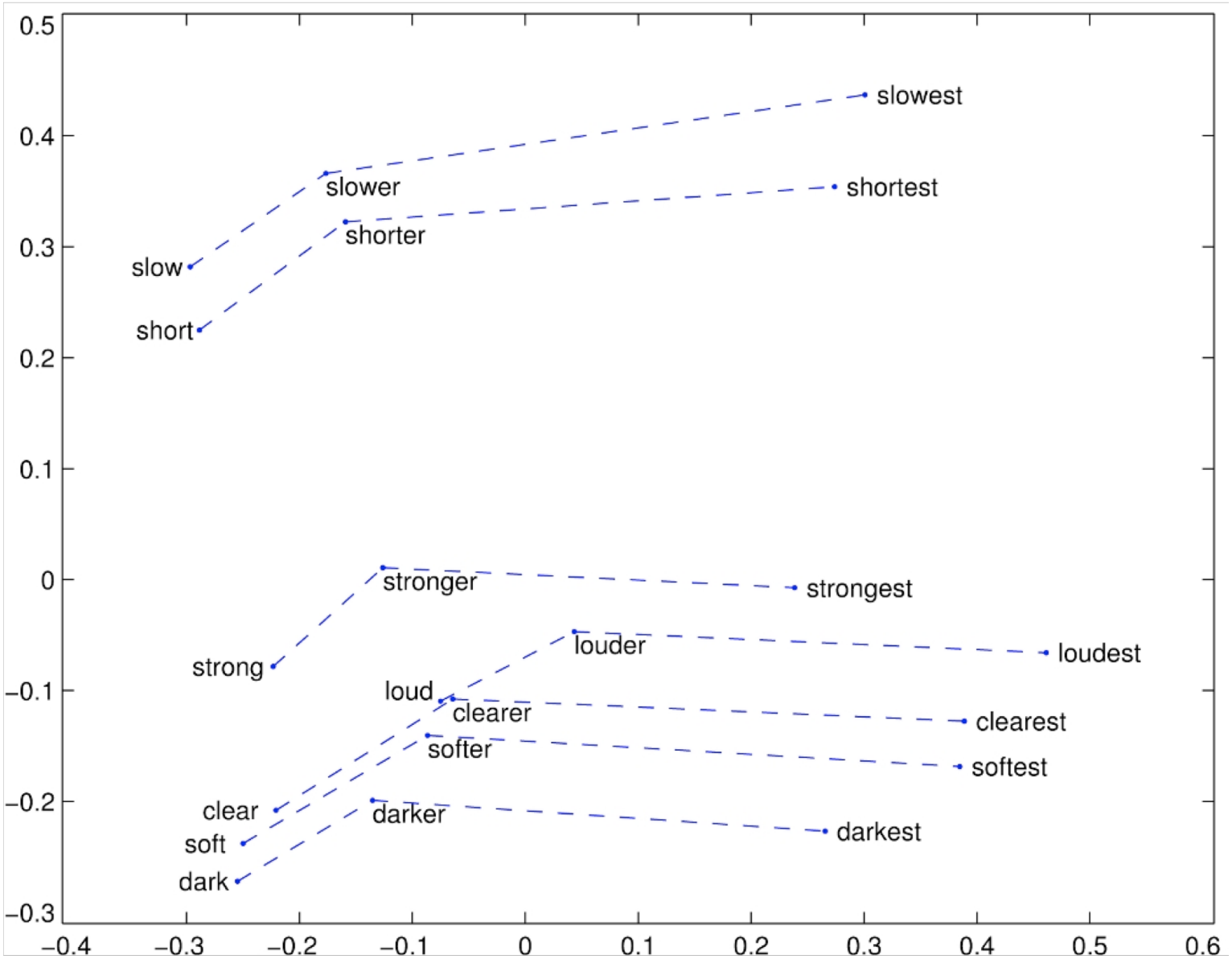
Glove Visualizations



Glove Visualizations: Company - CEO



Glove Visualizations: Superlatives



Details of intrinsic word vector evaluation

- Word Vector Analogies: Syntactic and **Semantic** examples from http://code.google.com/p/word2vec/source/browse/trunk/questions-words.txt

: city-in-state
Chicago Illinois Houston Texas
Chicago Illinois Philadelphia Pennsylvania
Chicago Illinois Phoenix Arizona
Chicago Illinois Dallas Texas
Chicago Illinois Jacksonville Florida
Chicago Illinois Indianapolis Indiana
Chicago Illinois Austin Texas
Chicago Illinois Detroit Michigan
Chicago Illinois Memphis Tennessee
Chicago Illinois Boston Massachusetts

problem: different cities may have same name

Details of intrinsic word vector evaluation

- Word Vector Analogies: **Syntactic** and Semantic examples from

: gram4-superlative
bad worst big biggest
bad worst bright brightest
bad worst cold coldest
bad worst cool coolest
bad worst dark darkest
bad worst easy easiest
bad worst fast fastest
bad worst good best
bad worst great greatest