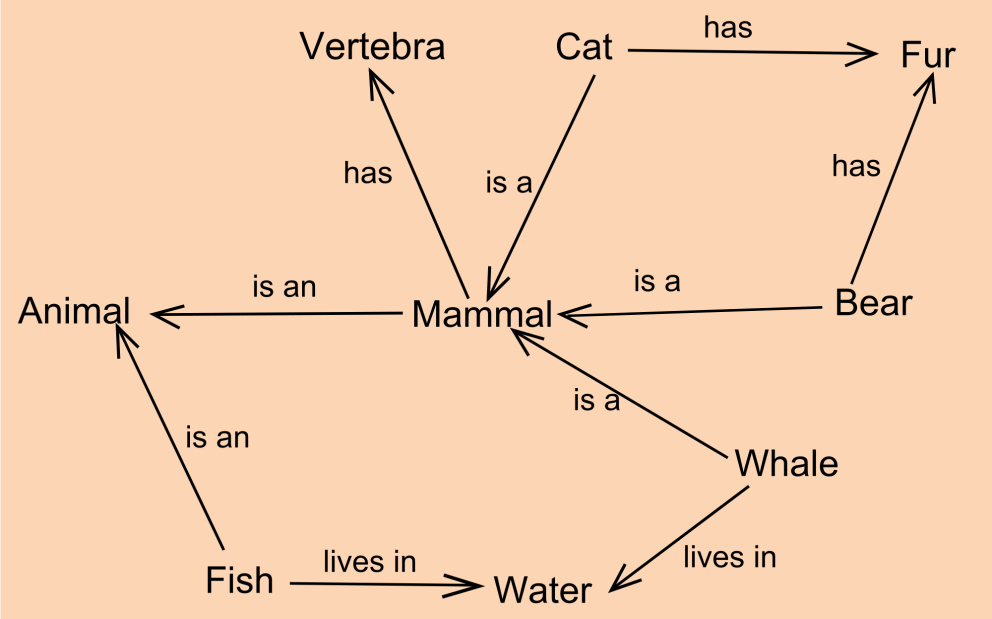
ML4NLP

Introduction to Lexical Semantics and Distributed Representations



Dan Goldwasser

Purdue University

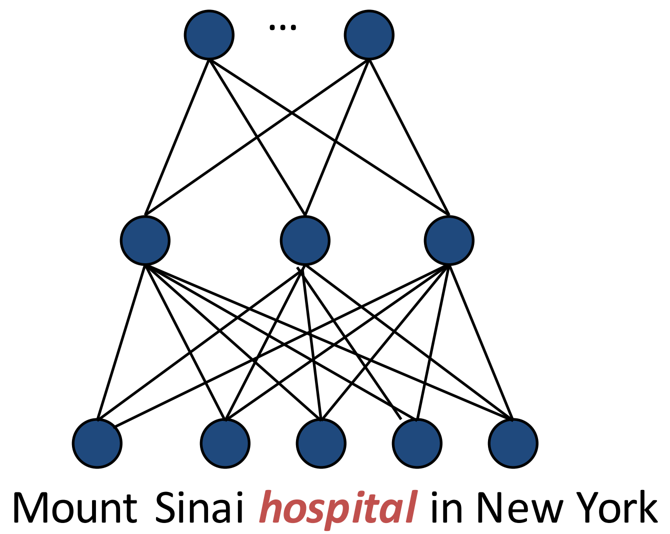
dgoldwas@purdue.edu

Learning Hidden Layer Representation

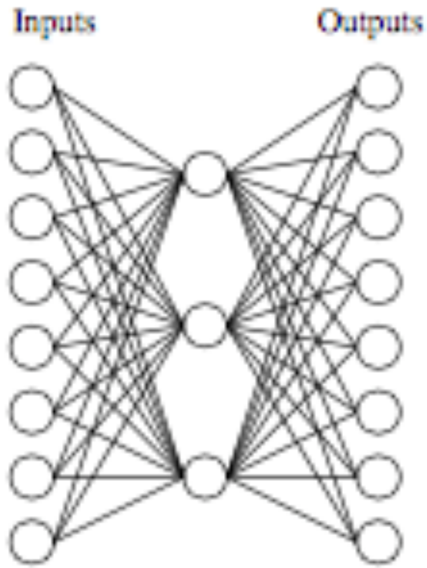
- **NN can be seen as a way to learn a feature representation**
  - Weight-tuning sets weights that define hidden units representation most effective at minimizing the error
- Backpropagation can define new hidden layer features that are not explicit in the input representation, but which capture properties of the input instances that are most relevant to learning the target function.
- *Trained hidden units can be seen as newly constructed features that re-represent the examples so that they are linearly separable*

DL as Representation Learning

How did we account of representation learning so far?

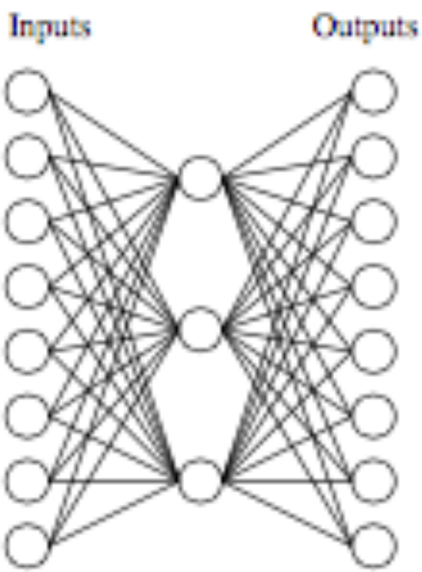


Auto-associative Network



Input		Output
10000000	→	10000000
01000000	→	01000000
00100000	→	00100000
00010000	→	00010000
00001000	→	00001000
00000100	→	00000100
00000010	→	00000010
00000001	→	00000001

Auto-associative Network

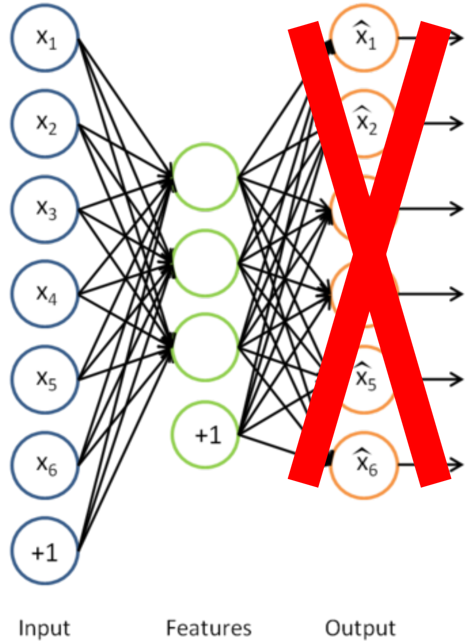


Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.01	.11	.88	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.22	.99	.99	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

Sparse Auto-Encoder

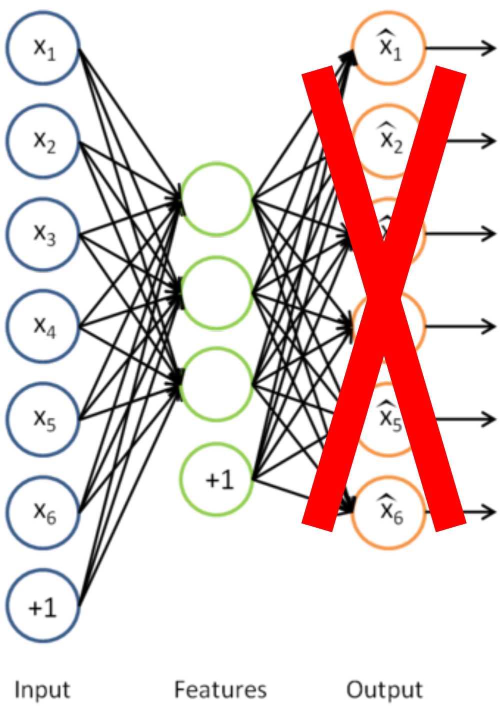
Goal: perfect reconstruction of the input vector x, by the output x’

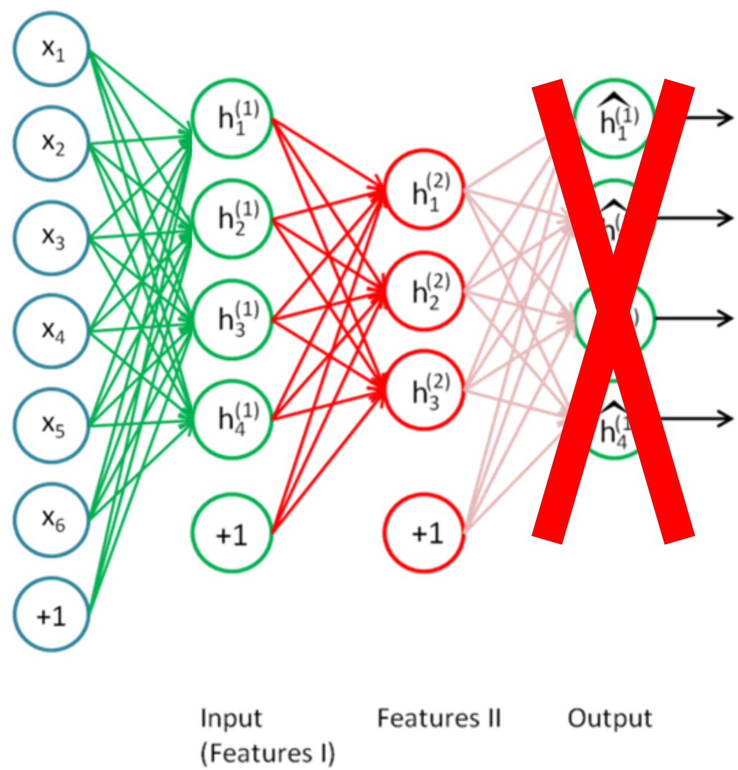
- **Simple approach**:
  - Minimize the error function l(h(x),x)
  - After optimization:
    - Drop the reconstruction layer



Stacking Auto Encoder

- Add a new layer, and a reconstruction layer for it.
- Repeat.

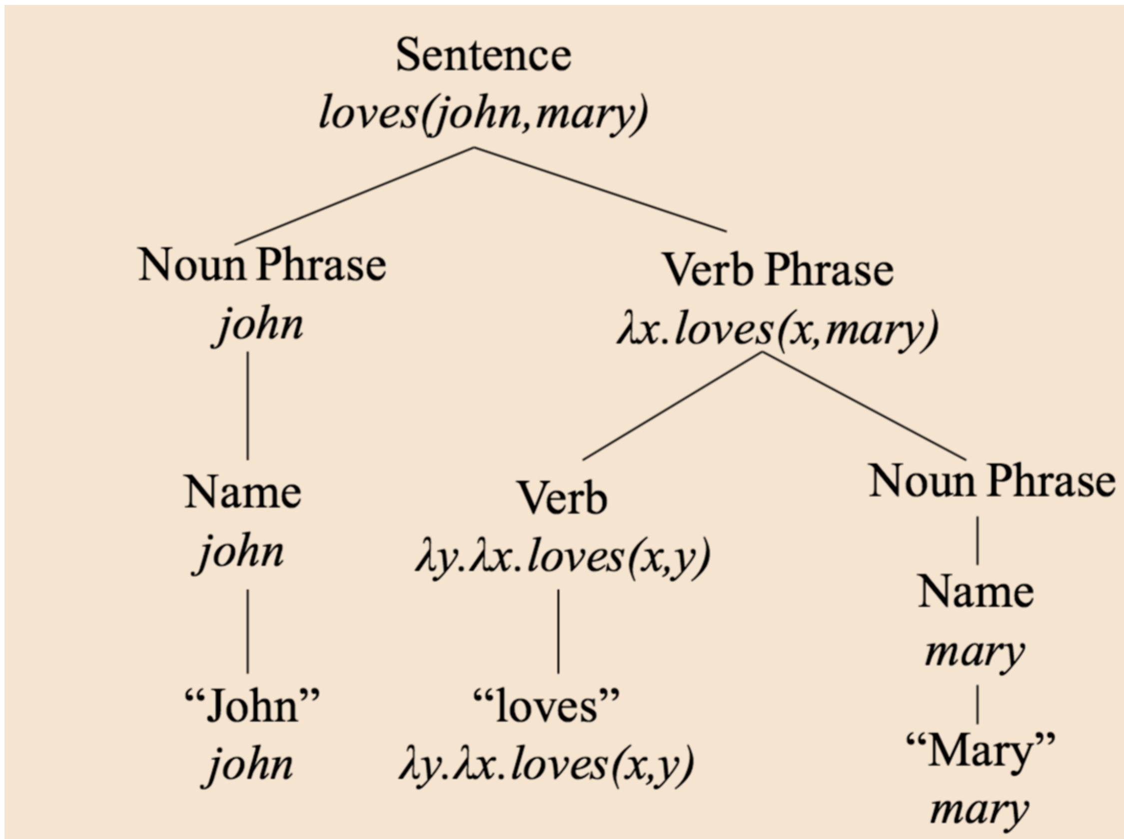




From Auto-Encoders to Word Embeddings

- So far the representations that we learned were **compressed** representation of the original inputs.
  - **Why is that better?**
- Ideally we would like to "inject" some meaning into these representations.
  - **What could be a simple requirement for this representation?**

Compositional Semantics



The Secret Lives of Words!

- First some definitions –
- **Word form** : inflected word appearing in text
- **Lemma**: stem of the word

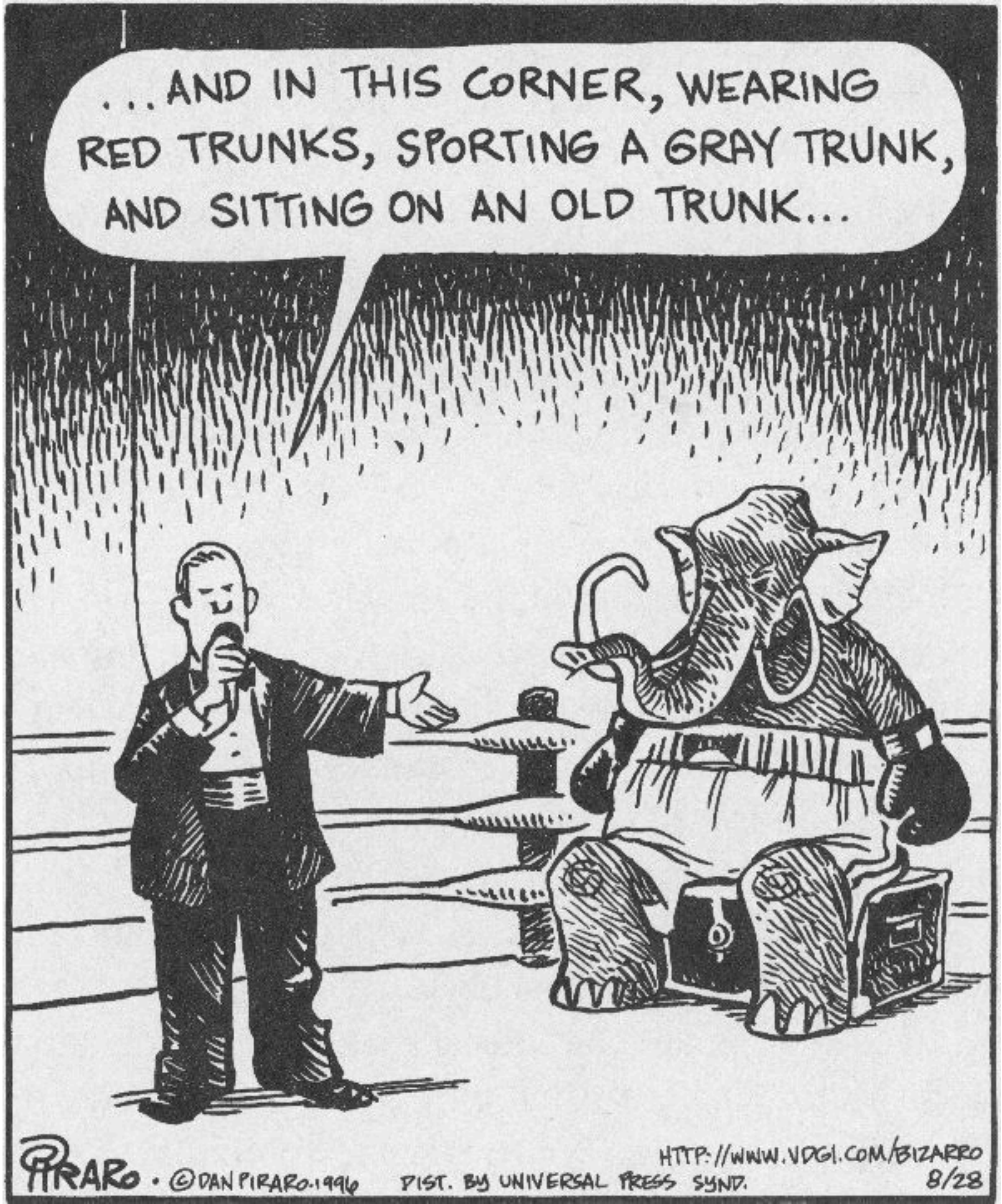
- *Several word forms will have the same lemma*
  - *Banking, Banked, Banks*

- *Do all of these have the same* ***meaning****?*

The Secret Lives of Words!

- **Lemmas can mean different things –**
  – John waited by the river bank.
  – John waited by the River bank.

- The word “bank” has different **senses**
- A sense is a discrete representation of the words meaning.
- **Homonymy:** words that share a form but have unrelated meanings


...AND IN THIS CORNER, WEARING RED TRUNKS, SPORTING A GRAY TRUNK, AND SITTING ON AN OLD TRUNK...
PIRARO • ©DAN PIRARO-1996 DIST. BY UNIVERSAL PRESS SYND.
HTTP://WWW.VDGI.COM/BIZARRO
8/28

Ok, so what?

the spirit is willing but the flesh is weak

Russian

English

The Vodka is good, but the meat is rotten.

Disclaimer: "MT Myth", but still a nice example..

The Secret Lives of Words!

- "The bank is the oldest building in Lafayette. It opened in 1852"
- "The bank refused John's loan"

- Polysemy: word that has several **related** meanings
- Happens systematically:
  - Building-organization, Food-animal, author-book

The Secret Lives of Words!

- "I sat of the sofa, it was big"
- "I sat on the couch, it was large"

- Words that have similar meaning are **synonyms**
- **There are often small differences:**
  - "Garbage can" vs. "Rubbish bin"
  - "Water" vs. H2O
  - "Big" vs. "large" (My *big/large* brother)

The Secret Lives of Words!

- Words with opposite meanings are **antonyms**.
  - Short – Long
  - Big – Small
- Words are **hyponyms** if one word is a subclass of the other.
  - *Car is a **hyponym** of vehicle.*
- The other direction is called a **hypernym**
  - *Vehicle is a **hypernym** of a car.*

The Secret Lives of Words!

- Hyponyms define a IS-A hierarchy
  - A cat is-a mammal is-an animal.
  - Hyponyms are transitive:

  If cat is a mammal **AND** mammals are animals

  Then a car is-an animal.
- A very useful resource: WordNet
  - A comprehensive hierarchy of concepts
  - *New York is-a city*

https://wordnet.princeton.edu/

WordNet

- Lexical database organized hierarchically
- Defines the possible senses of each word
- WordNet provides a ***SynSet*** for each word

Noun

- S: (n) **bank** (sloping land (especially the slope beside a body of water)) *"they pulled the canoe up on the bank"; "he sat on the bank of the river and watched the currents"*
- S: (n) depository financial institution, **bank**, banking concern, banking company (a financial institution that accepts deposits and channels the money into lending activities) *"he cashed a check at the bank"; "that bank holds the mortgage on my home"*
- S: (n) **bank** (a long ridge or pile) *"a huge bank of earth"*

WordNet Noun Relations

Relation	Also called	Definition	Example
Hypernym	Superordinate	From concepts to superordinates	$\textit{breakfast}^1 \rightarrow \textit{meal}^1$
Hyponym	Subordinate	From concepts to subtypes	$\textit{meal}^1 \rightarrow \textit{lunch}^1$
Member Meronym	Has-Member	From groups to their members	$\textit{faculty}^2 \rightarrow \textit{professor}^1$
Has-Instance		From concepts to instances of the concept	$\textit{composer}^1 \rightarrow \textit{Bach}^1$
Instance		From instances to their concepts	$\textit{Austen}^1 \rightarrow \textit{author}^1$
Member Holonym	Member-Of	From members to their groups	$\textit{copilot}^1 \rightarrow \textit{crew}^1$
Part Meronym	Has-Part	From wholes to parts	$\textit{table}^2 \rightarrow \textit{leg}^3$
Part Holonym	Part-Of	From parts to wholes	$\textit{course}^7 \rightarrow \textit{meal}^1$
Antonym		Opposites	$\textit{leader}^1 \rightarrow \textit{follower}^1$

Lexical Semantics

- It's convenient to think about WordNet as "ground truth"
- We can define Lexical semantics tasks, with respect to WordNet:
- Given a sentence, can you:
  - **Determine the right sense of each word?**
  - **Answer questions?**
    - *Identify synonyms, or other relations*

Word Similarity

- It's often more realistic to discuss word **similarity** instead of synonyms.
  - **Synonym**: binary relationship
  - **Similarity**: "soft" assignment
    - Sim(w1,w2) ~ 1 if words are synonyms
    - Sim(w1,w2) > 0, if words are related
- For example – Information retrieval engines need to identify similarity between content and query terms.

Word Similarity

Two broad approaches:

- **Thesaurus-based algorithms**.
  - Assume a comprehensive knowledge base (e.g., wordnet)
  - Do words appear nearby in the hypernym hierarchy? Similar definition?
- **Distributional algorithms.**
  - Assume a large collection of text (***not annotated!***)
  - Do the words appear in similar contexts?

Hypernym path based Similarity

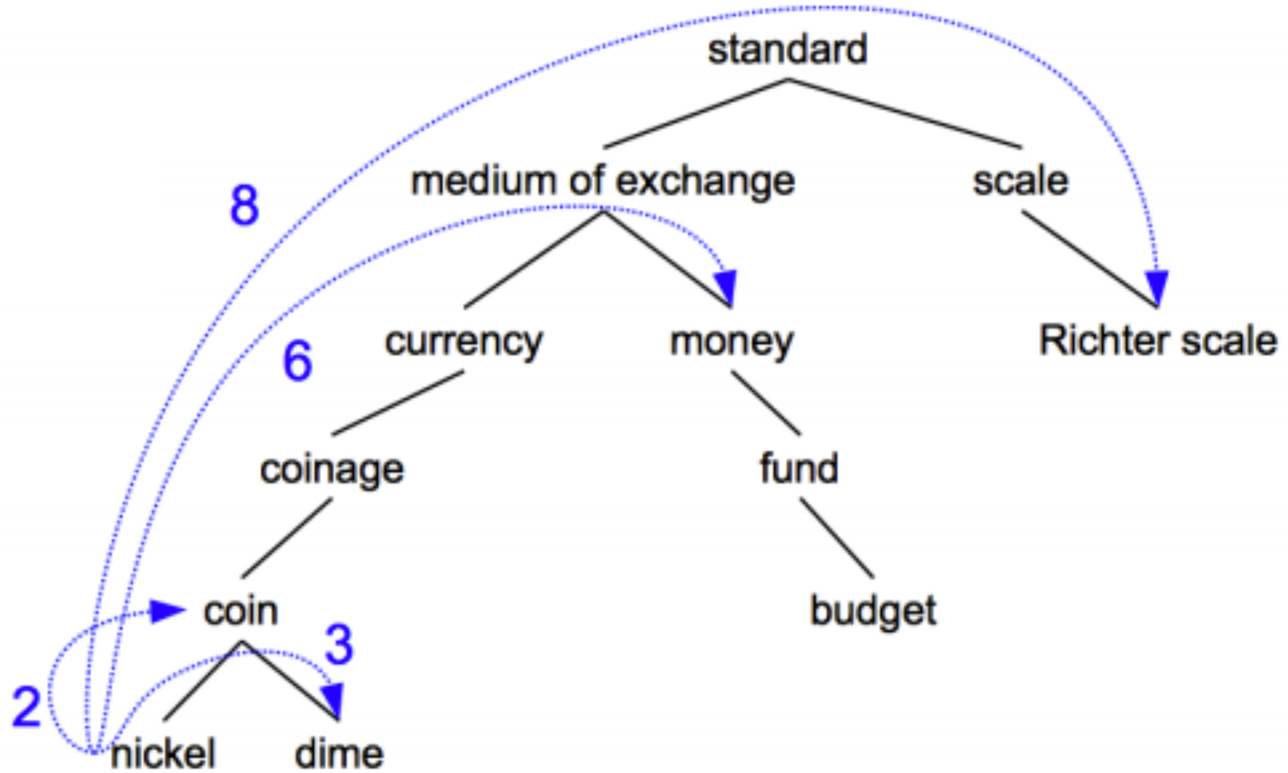


Image taken from D Jurafsky Slides on Lexical similarity

Hypernym path based Similarity

- The simple heuristic assumes uniform cost for each hop.
  - Nodes higher in the hierarchy are more abstract.
  - Ignore content of word definition
- Several works looked into improving it :
  - Resnik'95, Lin'98, Lesk's algorithm.

Hypernym path based Similarity

- **Pros**
  - Simple, exploits existing knowledge
  - Tends to have ***high precision***.
- **Cons**
  - Depends on language specific knowledge
  - Does not evolve with language (***low recall***)

Distributional Similarity

A bottle of tesgüino is on the table
Everybody likes tesgüino
Tesgüino makes you drunk
We make tesgüino out of corn.

Question:
What is tesgüino?

Firth 1957:
"You shall know a word by the company it keeps"

Example from Jurafsky and Martin

Distributional Models

- **Key idea**: *word meaning is defined by it's context.*
- This method, also known as the **Vector Space** model, maintain a vector of context words, for each word.

$$w = (f_1, f_2, f_3, f_4, \ldots f_n)$$

- Given a large corpus, maintain the context words counts for each word.
  - Define context window size.

Distributional Models

$$w = (f_1, f_2, f_3, f_4, \ldots f_n)$$

- Instead of the raw counts, we prefer to have a *normalized score*.
- ***Positive Point-wise Mutual Information***

$$\mathbf{PMI(x,y)} = Log\ \ P(x,y) / P(x)\, P(y)$$

  - ***Intuition****: are words x,y more likely to appear together than independently?*
- ***Positive PMI****: round all negative scores to 0.*

Distributional Similarity

A bottle of tesgüino is on the table
Everybody likes tesgüino
Tesgüino makes you drunk
We make tesgüino out of corn.

Question:

What is tesgüino?

Tesgüino = (Bottle = 123, Table =54, drunk = 141, Corn = 91, …)

Bourbon = (Bottle = 231, Table =41, drunk =231, corn =121, …)

Vodka = (Bottle = 311, Table =82, drunk =321, corn =0, …)

Distributional Similarity

Given the vector based representation of words we can compute their similarity easily -

$$\cos(v,w) = \frac{v \cdot w}{|v|\ |w|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

Using Positive PMI, ensure that Cosine similarity will have non-negative values

A machine learning perspective

- So far we looked at words as **discrete objects**.
  - For example, when building a sentiment classifier, each word was represented as a different coordinate

- "Great" = [0,0,0,0,0,0,0,1,0,0,... 0]
- "Awesome" = [0,0,0,1,0,0,...,0]
  - *This is known as "**one hot**" representation.*

A machine learning perspective

- Using “one-hot” representation, the connections between words are lost.
- We typically designed **complex feature functions** to get over that:
  - Maintain a dictionary of related words according to
    - Meaning (identify synonyms)
    - Word group (slang, function words, positive, negative,..)

A machine learning perspective

- **Can we use vector based methods to represent words other decision tasks?**
  - We can potentially overcome lexical sparseness problems!
  - We will try to answer this question in the coming lectures.

Word Embedding

- Basic idea: represent words in a continuous vector space.
  - Similar idea as using PMI
- **Key difference:**
  - Find low dimensional **dense** representation
  - **Instead of counting co-occurrence, use discriminative learning methods**
    - Predict surrounding words

Word2Vec

" AI fields such as NLP, machine learning , vision, have increased in popularity in recent years"

- For each word, predict other words in window C
- **Training Objective**: maximize the probability of context word, given the current word.

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j<c}\log p(w_{t+j}|w_t)$$

Word2Vec

- We want to evaluate $p(w_o|w_i)$
- For each word maintain **two** vectors
  - **Inside** word and **outside** word (context)
  - V represent inside, V' outside.

$$p(w_o|w_i) = \frac{\exp(v'_{w_o} v_{w_i})}{\sum_{w=1}^{W} \exp(v'_w v_{w_i})}$$

Word2Vec: Update rule

$$\frac{\partial}{\partial v_{w_i}} p(w_o|w_i)$$

-$p(w_o|w_i)$

;

We want to maximize this probability

$$\frac{\partial}{\partial v_{w_i}} \log \; p(w_o|w_i)$$

$$\frac{\partial}{\partial v_{w_i}} \log \left(\frac{\exp(v'_{w_o} \; v_{w_i})}{\sum_{w=1}^{W} \exp(v'_w \; v_{w_i})} \right)$$

$$\frac{\partial}{\partial v_{w_i}} \left(\log \; \exp(v'_{w_o} \; v_{w_i}) - \log \sum_{w=1}^{W} \exp(v'_w \; v_{w_i}) \right)$$

$$\frac{\partial}{\partial v_{w_i}}\left(\cancel{\log}\exp(v'_{w_o}\ v_{w_i}) - \log\sum_{w=1}^{W}\exp(v'_w\ v_{w_i})\right)$$

$$\frac{\partial}{\partial v_{w_i}}(v'_{w_o}\ v_{w_i})$$

Chain rule:

$$\frac{\partial}{\partial v_{w_i}}f(z(v_{wi})) = \frac{\partial f}{\partial z}\cdot\frac{\partial z}{\partial v_{w_i}}$$

$$\frac{\partial \log z}{\partial z} = \frac{1}{z}$$

$$\frac{1}{\sum_{w=1}^{W}\exp(v'_w\ v_{w_i})}$$

$$\frac{\partial}{\partial v_{w_i}}\sum_{w=1}^{W}\exp(v'_w\ v_{w_i})$$

$$\sum_{w=1}^{W}\frac{\partial}{\partial v_{w_i}}\exp(v'_w\ v_{w_i})$$

$$\sum_{w=1}^{W}\exp(v'_w\ v_{w_i})\ \frac{\partial}{\partial v_{w_i}}\ v'_w\ v_{w_i}$$

$$\sum_{w=1}^{W}\exp(v'_w\ v_{w_i})\ \ v'_w$$

Putting it all together ..

$$v'_{w_o} - \sum_{x=1}^{W} \frac{\exp(v'_x \; v_{w_i})}{\sum_{w=1}^{W} \exp(v'_w \; v_{w_i})} v'_x$$

$$v'_{w_o} - \sum_{x=1}^{W} p(x|w_i) \; v'_x$$

Similarly, you have to derive the update rule for the outside vectors..

Gradient Descent for W2V

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j < c} \log p(w_{t+j} | w_t)$$

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J(\theta)$$

You can use stochastic gradient descent to speed up the process.

Efficient Implementation

- For non-trivial vocabulary, the normalization factor is too costly to compute accurately.

$$p(w_o|w_i) = \frac{\exp(v'_{w_o} v_{w_i})}{\sum_{w=1}^{W} \exp(v'_w v_{w_i})}$$

- **Skip-gram with negative sampling**
  - Binary logistic regression for a small subset:
    - True pair, small subset of negative examples.

Skip-gram with Negative Sampling

- New objective function:

$$\log \sigma(v'_{w_o} v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^T v_{w_I}) \right]$$

Maximize the probability of center + context words

Minimize the probability of random words

Note:

- Only pick a **small subset** of negative examples
- samples are drawn from a distribution: $P_n(w)$
- $P_n(w)$ captures unigram statistics, modified to increase the probability of sampling low frequency words.

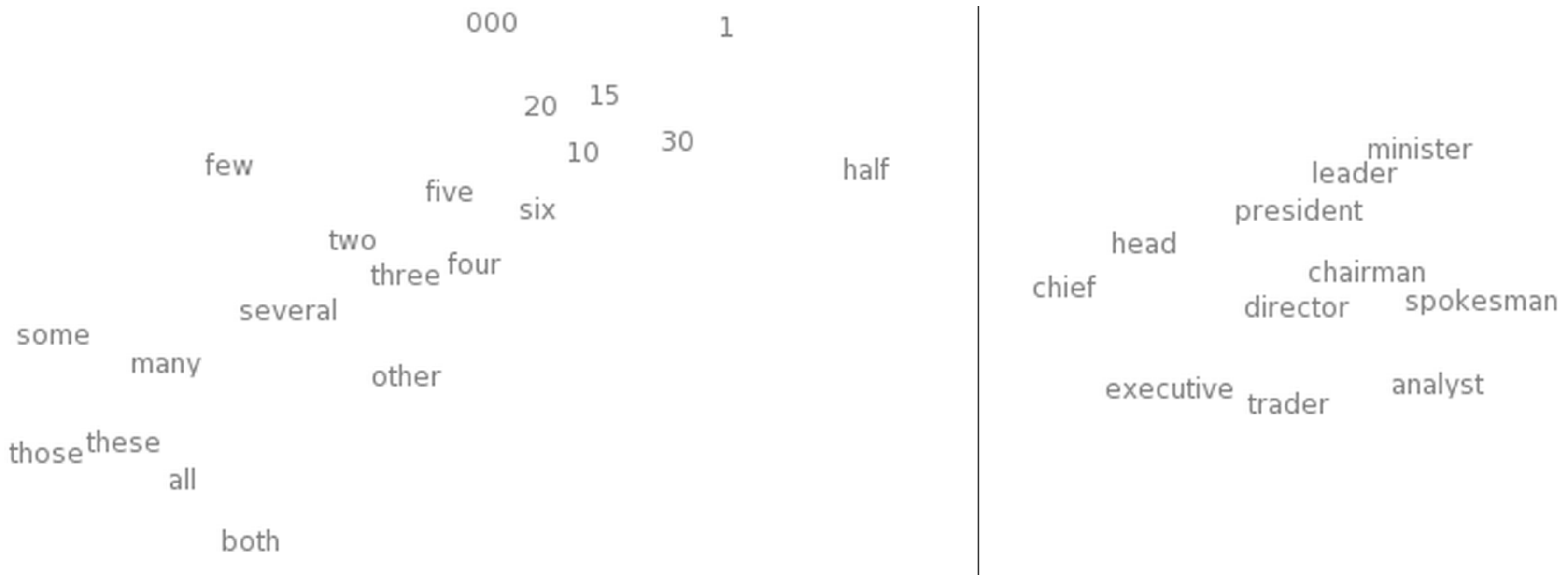
Continuous Bag-of-Words

- Very similar idea:
  - Instead of predicting context words, based on center word,
  - ***Predict center word using context words***
    - Sum up the surrounding words vectors

- Resulting word vectors capture similar information.

Word Embedding

- **Word embedding**: *move to a low dimensional, real valued dense representation of the input*
  - Key idea: similar words should have similar vectors



Turian et-al 2010

Word2Vec

```
Enter word or sentence (EXIT to break): Chinese river

                 Word       Cosine distance
------------------------------------------
        Yangtze_River              0.667376
              Yangtze              0.644091
       Qiantang_River              0.632979
    Yangtze_tributary              0.623527
     Xiangjiang_River              0.615482
        Huangpu_River              0.604726
       Hanjiang_River              0.598110
        Yangtze_river              0.597621
          Hongze_Lake              0.594108
              Yangtse              0.593442
```

Mikolov et-al 2013

Word Representation Arithmetic

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Mikolov et-al 2013

Word Representation Arithmetic

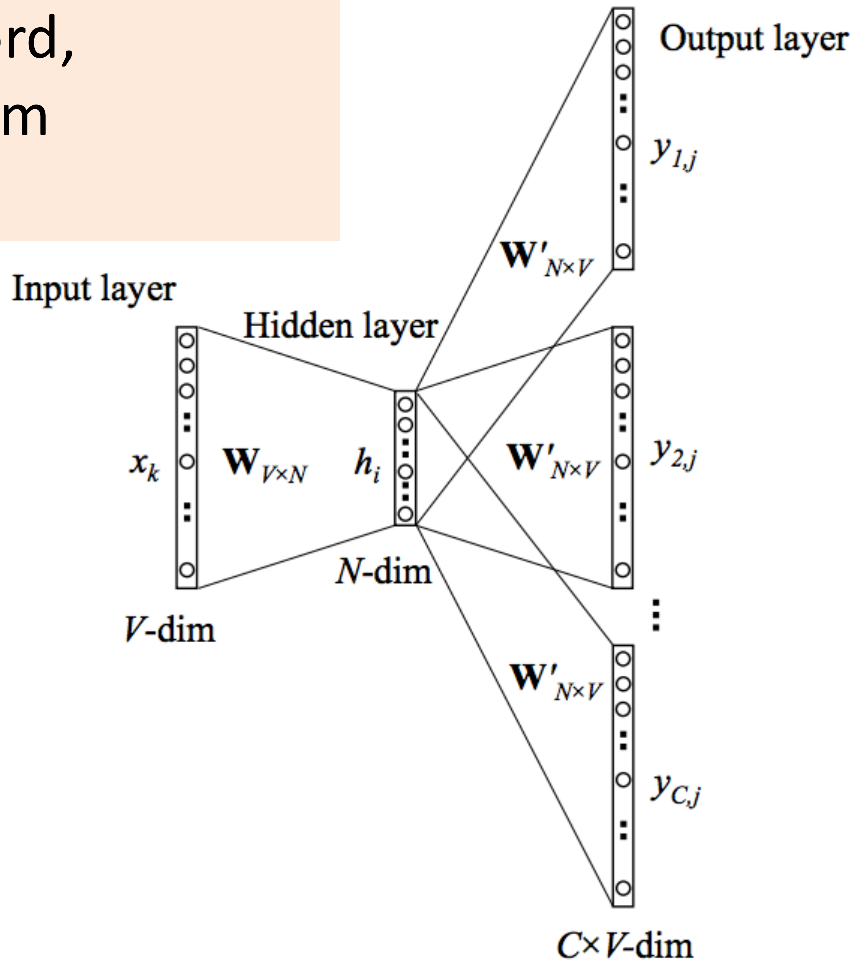
Paris - France + Italy = Rome

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Mikolov et-al 2013

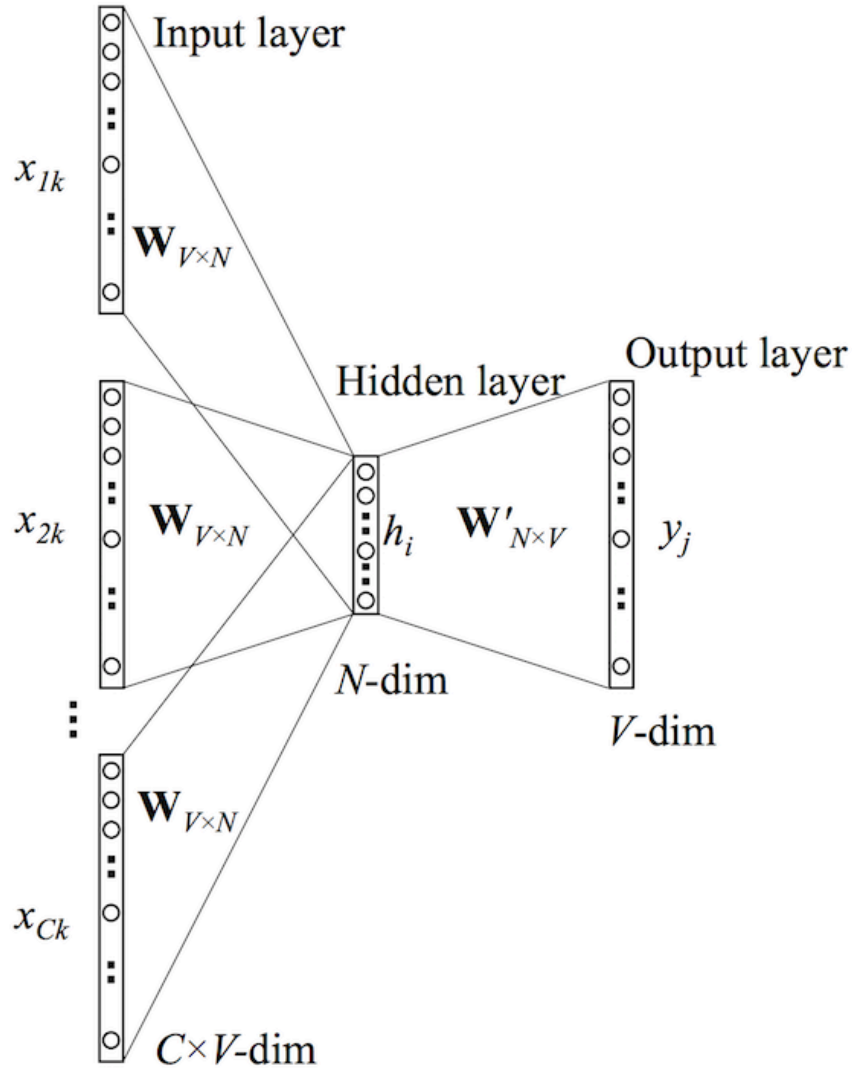
Words2Vec: Skip-gram model

We can tell the W2V story a little differently:
The vectors representing each word, correspond to weights coming from two layers in a NN



Mikolov et-al 2013

Words2Vec: CBOW model



Mikolov et-al 2013