CS 6840: Natural Language Processing

Word Embeddings

Razvan C. Bunescu

School of Electrical Engineering and Computer Science

bunescu@ohio.edu

One-Hot Vector Representations

- Sparse vector representation:
  - V is the vocabulary
  - Each word *w* is mapped to a unique id(*w*) between 1 and |V|.
    - i.e. the position of the word in the vocabulary.
  - Represent a word w using a "one-hot" vector **w** of length |V|:
    - **w**[i] = 1, if i = id(w).
    - **w**[i] = 0, otherwise
- Example:
  - Suppose id(ocean) = 2 and id(water) = 4. Then:
    - **w**(ocean) = [0, 1, 0, 0, 0, ..., 0]
    - **w**(water) = [0, 0, 0, 1, 0, ..., 0]

Sparse Representations of Words are Problematic for Machine Learning in NLP

1. Document classification:
   - Bag-of-words representation: each document is the sum of the vectors of all the words in the document, normalized to unit length.
   - Suppose we use *softmax regression* to classify into classes in C.
     - A parameter is needed for each (word, class) pair:
       - => |V| × |C| parameters => 100K × 10 => **1M parameters**.
       - The number of labeled documents needed to train these many parameters may be unfeasible to obtain.
     - If voleyball does not appear in the training documents, but is mentioned in the test document, it will be completely ignored:
       - Even though voleyball is semantically close to basketball, which appeared many times in training documents from the *Sports* category.

Sparse Representations of Words are Problematic for Machine Learning in NLP

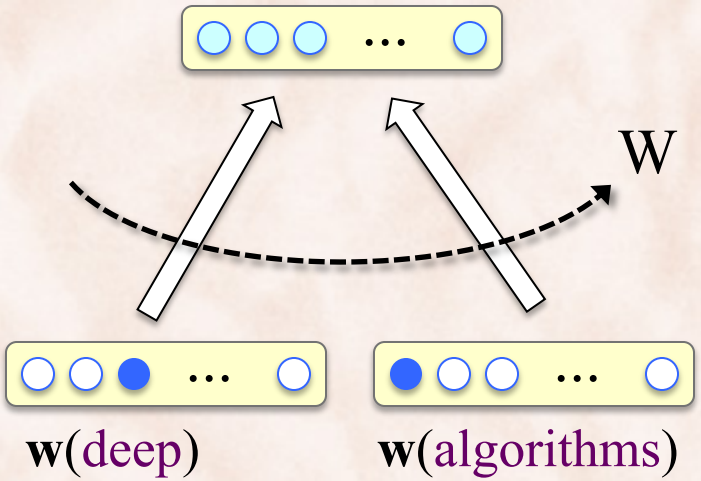
2. (Neural) Language Modeling:
   - **Predict the next word in a sequence**:
     - AI systems use deep ..... (dish? learning? about?, ...)
     - Need to compute P(w | w_{-1}, w_{-2}, ...):
       - want P(learning | deep, use) > P(about | deep, use).
   - **Predict the most likely word in a context**:
     - AI systems use deep ..... algorithms. (dish? learning? about?, ...)
     - Need to compute P(w | w_{-1}, w_{-2}, ..., w_1, w_2, ...).
   - Language modelling is useful for many tasks in NLP:
     - spell checking.
     - machine translation.
     - speech recognition.

(Neural) Language Modeling with Sparse Word Representations

2. (N)LM with (Naive) Softmax Regression:

AI systems use deep ..... algorithms

P(*w* | deep, algorithms) <== for each word *w* in V



– Need |W| = 2×|V|×|V| parameters!

Sparse vs. Dense Representations of Words

- Sparse representations:
  - Each word *w* is a sparse vector $\mathbf{w} \in \{0,1\}^{|V|}$ or $\mathbb{R}^{|V|}$.
    - Using words as features leads to *large number of parameters*!
    - sim(ocean, water) = 0 => *no meaning* => *low generalization*!
- Dense representations:
  - Each word *w* is a dense vector $\mathbf{w} \in \mathbb{R}^k$, where $k \ll |V|$.
  - Can use unsupervised learning:
    - Use Harris' **Distributional Hypothesis** [Word, 1954]
      - words that appear in the same contexts tend to have similar meanings.
    - sim(ocean, water) > sim(ocean, forest) > 0

Using Context to Build Word Representations

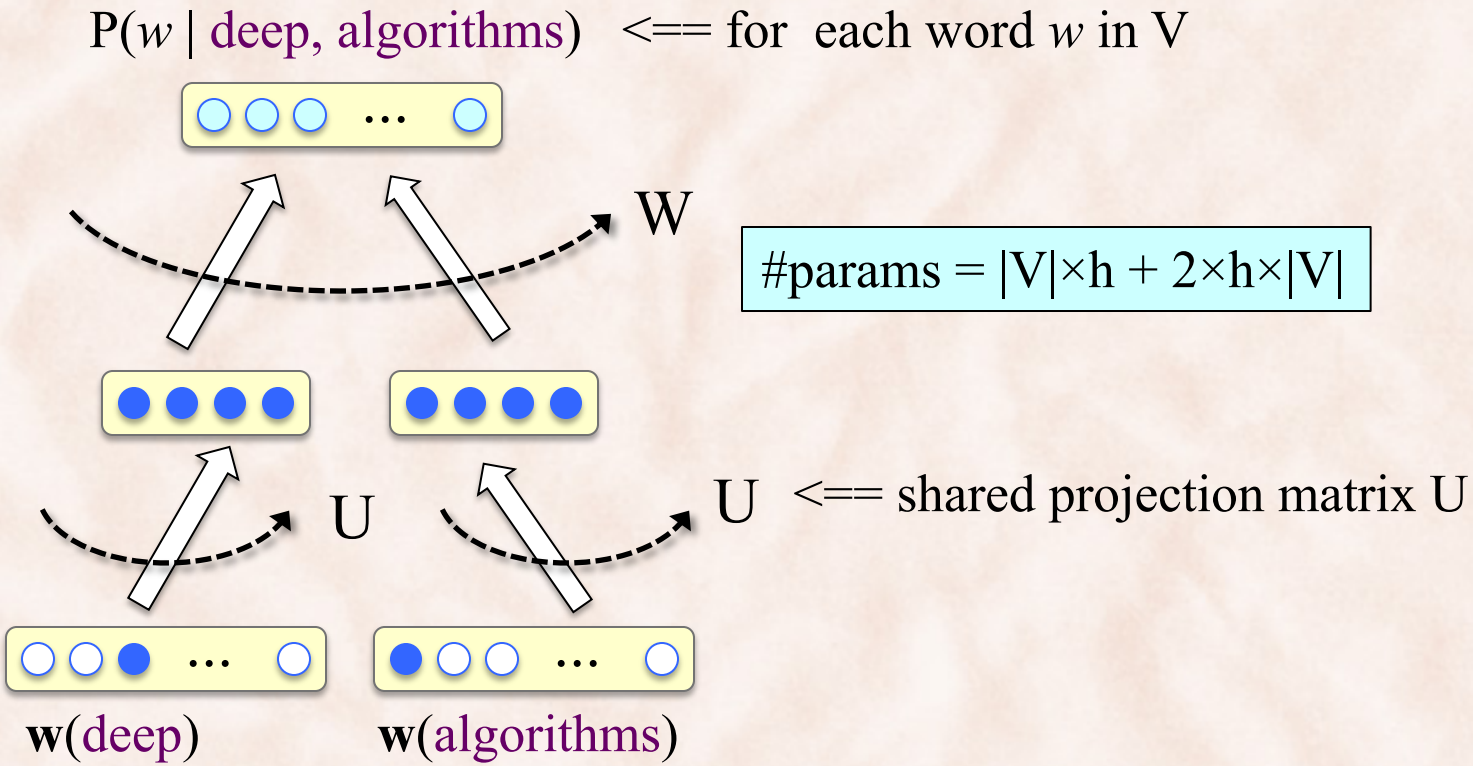
- **Distributional Semantics** idea:
  - *The meaning of a word is determined by the words that appear nearby, i.e. its context.*
    - "You shall know a word by the company it keeps" (Firth 1957).
    - One of the most successful ideas of modern statistical NLP!
  - Given an *occurrence* of word *w*, its *context* = the set of words that appear within a fixed-size window to the left and to the right.
    - Use all the contexts of word *w* to build its representation:

Enculturation is the process by which people learn values and behaviors that are ...
Reading directions helps a player learn the patterns that solve the Rubik's ...
... some people may be motivated to learn how to play a real instrument ...

5 left context words *5 right context words*

(Neural) Language Modeling with Dense Word Representations

- Softmax on top of a hidden layer of size h per word:



Neural Language Modeling with Dense Word Representations

- Neural Language Modeling:
  - Associate each word *w* with its distributed representation U**w**.
    - **w** is the *sparse (one-hot) representation* of word *w*.
    - U**w** is the *dense representation* of word *w*:
      - i.e. word representation, i.e. word embedding, i.e. distributed representation.
    - U is the projection or embedding matrix:
      - its columns are the word embeddings.
  - Simultaneously learn the word embeddings (U) and the softmax parameters (W).
  - After training on large text corpus, throw away W, keep only U:
    - Can be *tied* for even better performance [Press & Wolf, ACL'17].

Neural Languange Models for Learning Word Embeddings

- Softmax training of NLMs is expensive:
  - Maximum Likelihood => minimize cross-entropy.
    - Need to compute the normalization constant for each training example i.e. for each word instance in the corpus:

$$E_D(W) = -\ln \prod_{t=1}^{T} p(w_t \mid \mathrm{h}_t) = -\sum_{t=1}^{T} \ln \frac{\exp(W[w_t :] \times \mathrm{h}_t)}{Z(w_t)}$$

$$Z(w_t) = \sum_{v \in V} \exp(W[v :] \times \mathrm{h}_t)$$

  - Use Pairwise Ranking approach instead.

Neural Languange Models for Learning Word Embeddings

- Pairwise Ranking approach:
  - Train such that $p(w_t \mid \mathbf{h}_t) > p(w \mid \mathbf{h}_t)$ i.e. $W[w_t:] \times \mathbf{h}_t > W[w:] \times \mathbf{h}_t$
    - w is sampled at random from V.
    - give a higher score to the actual word w_t than to random words.

$$W, U = \text{argmin} \sum_{t=1}^{T} \sum_{w \in V} \max\{0, 1 - W[w_t:] \times \mathbf{h}_t + W[w:] \times \mathbf{h}_t\}$$

$$\text{minimize} \quad J(U,W) = \sum_{t=1}^{T} \sum_{w \in V} \xi_{t,w} + \text{R}(U) + \text{R}(W)$$

$$\text{subject to:} \quad W[w_t:] \times \mathbf{h}_t - W[w:] \times \mathbf{h}_t \geq 1 - \xi_{t,w}$$

Neural Languange Models for Learning Word Embeddings

- Pairwise Ranking approach [Collobert et al., JMLR'11]:
  - Train using SGD on the ranking criterion.
  - Sample "negative" words from V for each w_t.

- Evaluation of learned embeddings:
  - Word similarity questions:
    - given seed word w, find word(s) v with most similar embedding:

$$\arg\max_{v \in V} \cos(U\mathbf{w}, U\mathbf{v})$$

  - Analogy questions [Mikolov et al., NIPS'13].

Evaluation of Word Embeddings

[Collobert et al., JMLR'11]

COLLOBERT, WESTON, BOTTOU, KARLEN, KAVUKCUOGLU AND KUKSA

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
454	1973	6909	11724	29869	87025
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

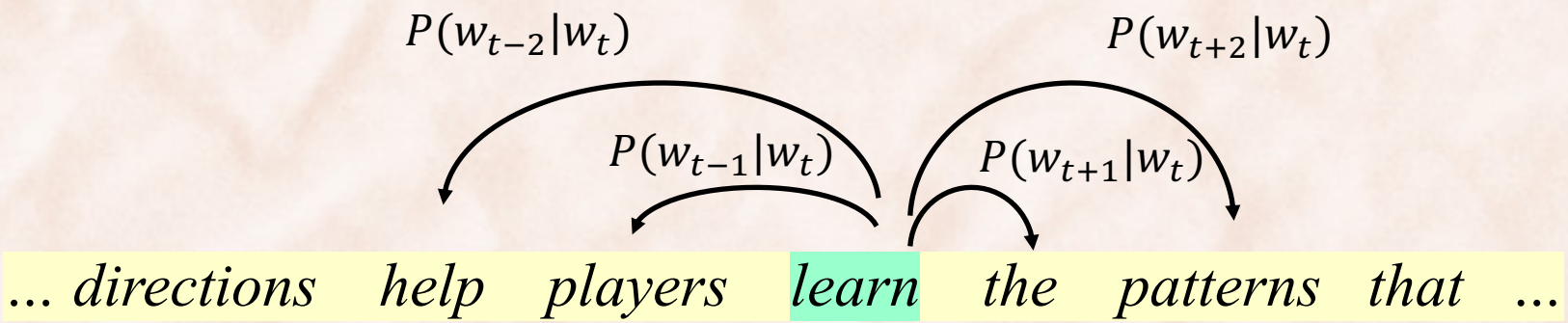
Table 7: Word embeddings in the word lookup table of the language model neural network LM1 trained with a dictionary of size 100,000. For each column the queried word is followed by its index in the dictionary (higher means more rare) and its 10 nearest neighbors (using

Word2Vec Framework

[Mikolov et al., NIPS'13]

- Assign 2 vector representations v_w and u_w to each word w in a given vocabulary V:
  - Initialize vector representations with random values.
- Go through each position t in the text:
  - Word w_t is considered the *center* word c.
  - Words in the context are considered *outside* words o.
- Use the dot-product between the word vectors for v_c and u_o to calculate the probability of:
  - o given c (Skip-gram model).
  - c given o (Cbow model).
- Repeatedly adjust the word vectors to increase probabilities.

The Skip-gram Model: Naïve Implementation with Softmax



- To calculate $P(w_{t+k}|w_t)$ use softmax with 2 vectors / word:
  - v_w if w is the center (c) word w_t, i.e. the *parameters*.
  - u_w if w is a context (o) word w_{t+k}, i.e. the *features*.

$$p(o|c) = \frac{\exp(v_c^T u_o)}{\sum_{w \in V} \exp(v_c^T u_w)}$$

The Skip-gram Model

- Maximize likelihood over all positions $t = 1..T$ in the text:

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{k\in C_t} \log p(w_{t+k}|w_t)$$

where context $C_t = \{k| -m \leq k \leq m, k \neq 0\}$.

- $\theta \in R^{2d|V|}$ is the vector of all parameters:

$$\theta = [v_{\text{aardvark}} \mid \ldots \mid v_{\text{zoo}} \mid u_{\text{aardvark}} \mid \ldots \mid u_{\text{zoo}}]$$

The Skip-gram Model: Gradient Computation

- Training the center and outside embeddings:
  - Batch GD on $J(\theta)$ will be very slow!
  - Instead, run SGD on $J(\theta, t)$, i.e. update after each position t.
    1. Compute the gradient of $J(\theta, t)$ w.r.t. the center word params v_c.
    2. Compute the gradient of $J(\theta, t)$ w.r.t. the context words features u_o.

$$J(\theta, t) = \sum_{k \in C_t} \log p(w_{t+k} | w_t)$$

$$p(o|c) = \frac{\exp(v_c^T u_o)}{\sum_{w \in V} \exp(v_c^T u_w)}$$

  - Need to create & use hash from words to their word vectors.

The Skip-gram Model: From Softmax to Binary Logistic Regression

- Partition function in Softmax is too computationally expensive:

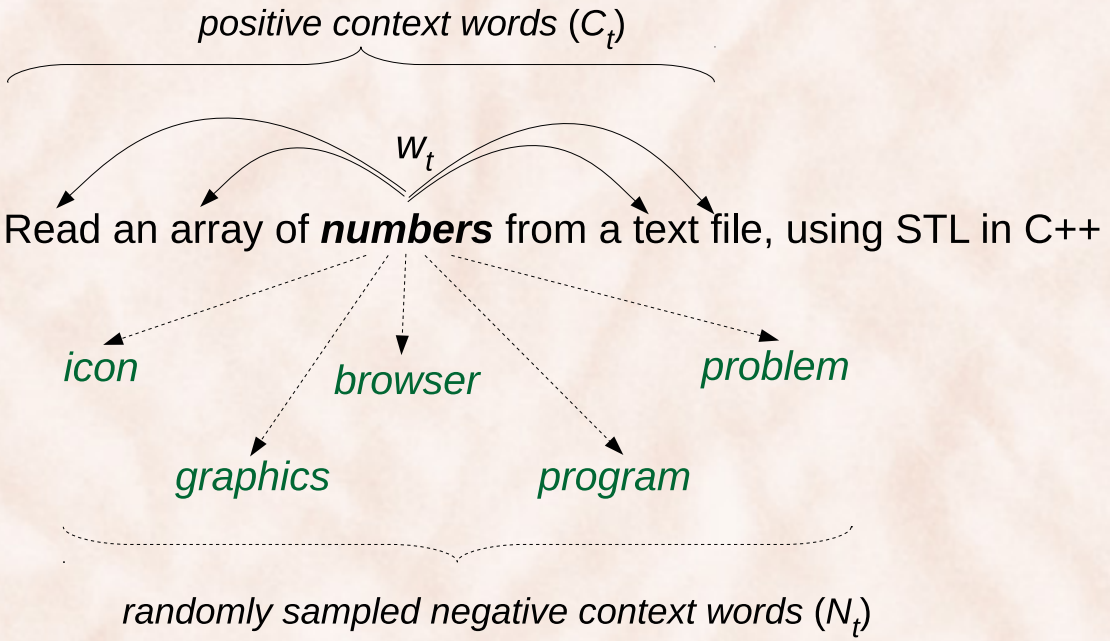
$$p(o|c) = \frac{\exp(v_c^T u_o)}{\sum_{w \in V} \exp(v_c^T u_w)}$$

- Instead, word2vec trains Logistic Regression to classify:
  - *Positive*: The true (c, o) pairs.
  - *Negative*: Noisy (c, r) pairs, where r is a word sampled at random.
    - Called **Negative Sampling (NS)**:
      - A special case of **Noise Contrastive Estimation (NCE)**:
        - » NS good for learning embeddings, but not for LMs.

        Chris Dyer – Notes on NCE and NS
    - See also Ruder – Approximating the Sofmax.

The word2vec Skip-gram Model: Logistic Regression with Negative Sampling

- Learn vector representations of words in order to predict surrounding words:
  - Logistic regression with negative sampling and subsampling of frequent words.



The word2vec Skip-gram Model

[Mikolov et al., NIPS'13]

- Common words provide less information than rare words:
  - co-occurrence (*France*, *Paris*) more important than (*France*, *the*).
    - try to counter imbalance between rare and frequent words.

1. Negative sampling:
   - Estimate unigram distribution U(*w*) of words from the training corpus.
   - Sample negatives according to distribution $P_n(w) = U(w)^{3/4} / Z$.

2. Subsampling of frequent words:
   - Compute discounting distribution $P_d(w) = 1 - \text{sqrt}(10^{-5} / f(w))$.
   - Discard (positive) examples $c = w_t$ according to $P_d(w_t)$.

The word2vec Skip-gram Model

[Mikolov et al., NIPS'13]

- Use vector representations both as features and parameters:
  - Parameters: vector $\mathbf{v}_c$ of center word c.
  - Features: vector $\mathbf{u}_o$ of word o to be predicted *in context* C.

$$p(o \in C|c) = \sigma(v_c^T u_o) = (1 + exp(-v_c^T u_o))^{-1}$$

  - Features: vector $\mathbf{u}_k$ of word k to be predicted *not in context* C.

$$p(k \notin C|c) = 1 - \sigma(u_k^T v_c) = \sigma(-u_k^T v_c)$$

- For each position $t = 1$ .. T in a sequence of T words:
  - For each *not discarded positive* (c, o) pair, *sample* K *negative* words:
    - Use SGD to minimize negative log-likelihood objective:

$$J(o, \mathbf{v}_c, U) = -\log \sigma(v_c^T u_o) - \sum_{k=1}^{K} \log \sigma(-v_c^T u_k)$$

The Skip-gram Model

[Mikolov et al., NIPS'13]

- Evaluation of learned embeddings:
  - Word similarity questions:
    - given seed word *w*, find word(s) *v* with most similar embedding:

$$v = \text{argmax}_{v \in V} \frac{\mathbf{w}^T \mathbf{v}}{||\mathbf{v}||}$$

  - Analogy questions:
    - find word *x* such that: *w* (e.g. *Paris*) is to *v* (e.g. *France*) what *x* is to *u* (e.g. *Germany*).
    - want **w** − **v** to be similar to **x** − **u** <=> **w** − **v** + **u** similar to **x**.

$$x = \text{argmax}_{x \in V} \frac{(\mathbf{w} - \mathbf{v} + \mathbf{u})^T \mathbf{x}}{||\mathbf{x}||}$$

The Skip-gram Model

[Mikolov et al., ICLR'13]

Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

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

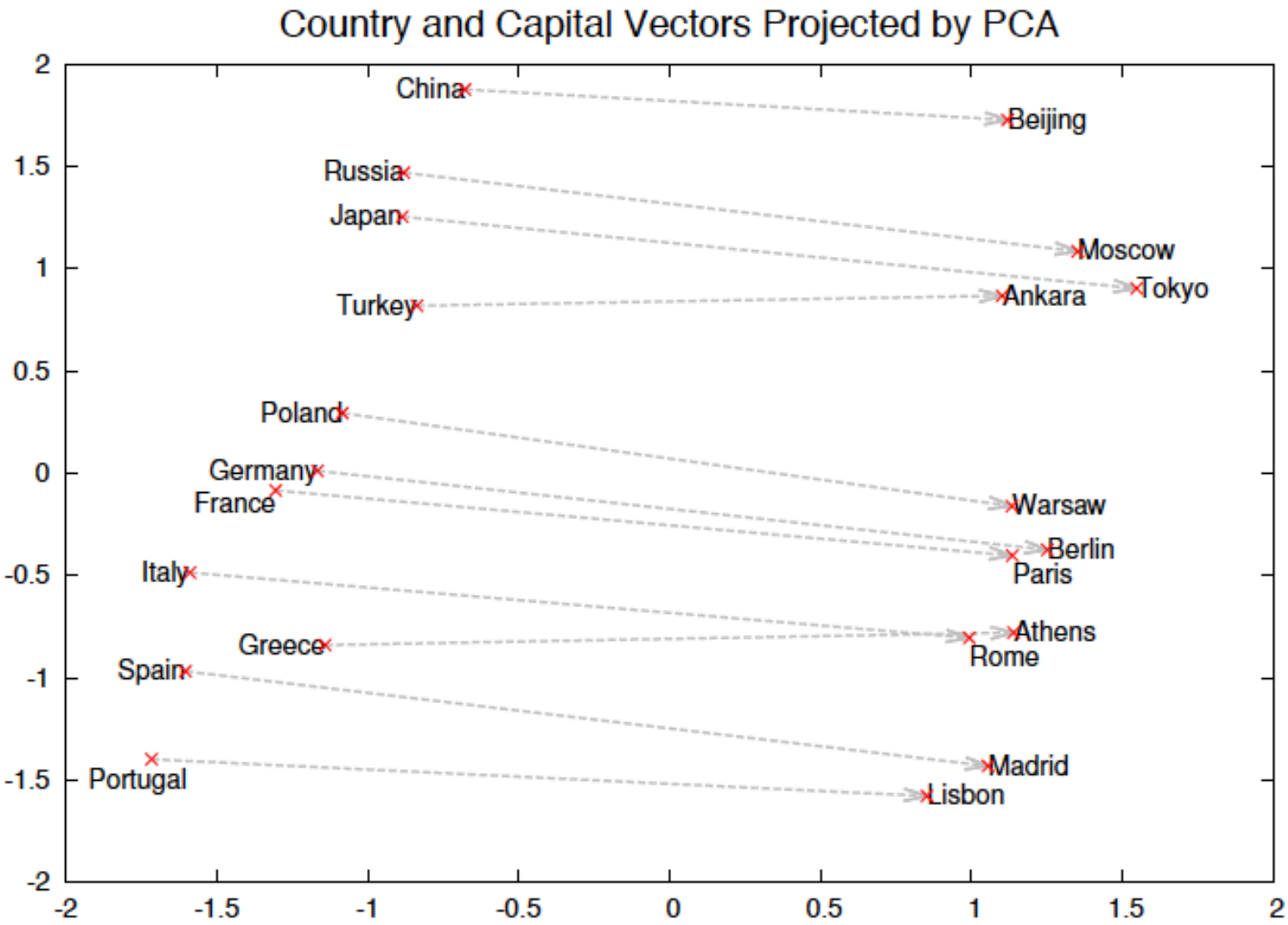


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

GloVe

https://nlp.stanford.edu/projects/glove [Pennington et al., EMNLP'14]

- **Insight**: Ratios of co-occurrence probabilities can encode meaning components.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

  - Co-occurrences with *solid* and *gas* are important for the meaning of target words *ice* and *steam*:
    - Large values of the ratio correlate with properties specific to *ice*.
    - Small values of the ratio correlate with properties specific to *steam*.
  - Co-occurrences with *water* and *fashion* are not important:
    - Values close to 1.

GloVe

https://nlp.stanford.edu/projects/glove [Pennington et al., EMNLP'14]

- Want to capture ratios of co-occurrence probabilities as linear meaning components in a word vector space:
  - Probabilities are computed as ratios of co-occurrence counts:

$$P(i|k) = \frac{X_{ik}}{X_k}$$ *probability that word i appears in the context of word k*

  - Use a log-bilinear model:

$$w_i^T \tilde{w}_k = \log P(i|k)$$

  - Log-ratios are then expressed in terms of vector differences:

$$w_x^T (\tilde{w}_a - \tilde{w}_b) = \log \frac{P(x|a)}{P(x|b)}$$

GloVe

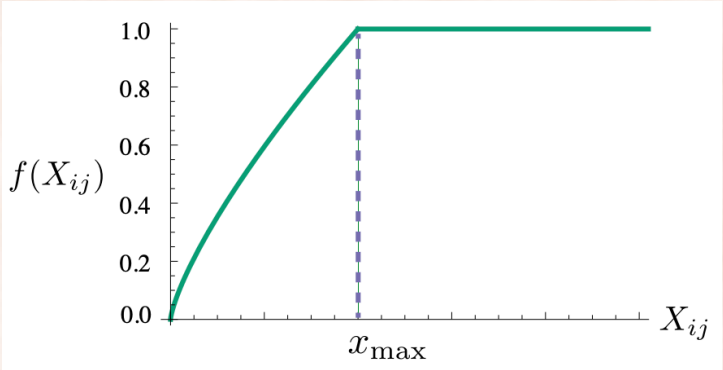
https://nlp.stanford.edu/projects/glove [Pennington et al., EMNLP'14]

- The *exchange symmetry* between a word and its context words can be satisfied by estimating *log counts* as:

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log X_{ik} \qquad \tilde{b}_k = \log X_k$$

- Solve a *weighted* least squares regression model:

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



GloVe

https://nlp.stanford.edu/projects/glove [Pennington et al., EMNLP'14]

- Fast training, scalable to huge corpora:
  - Good performance even with small corpus and small vectors
- Hyper-parameters:
  - Good dimension is ~300.
  - Asymmetric context (only words to the left) not as good.
  - Window size of 8 around each center word is sufficient.
- Some ideas from Glove paper have been shown to improve skip-gram (SG):
  - Use sum of both vectors (context + target).

Visualizations of Semantic Relations

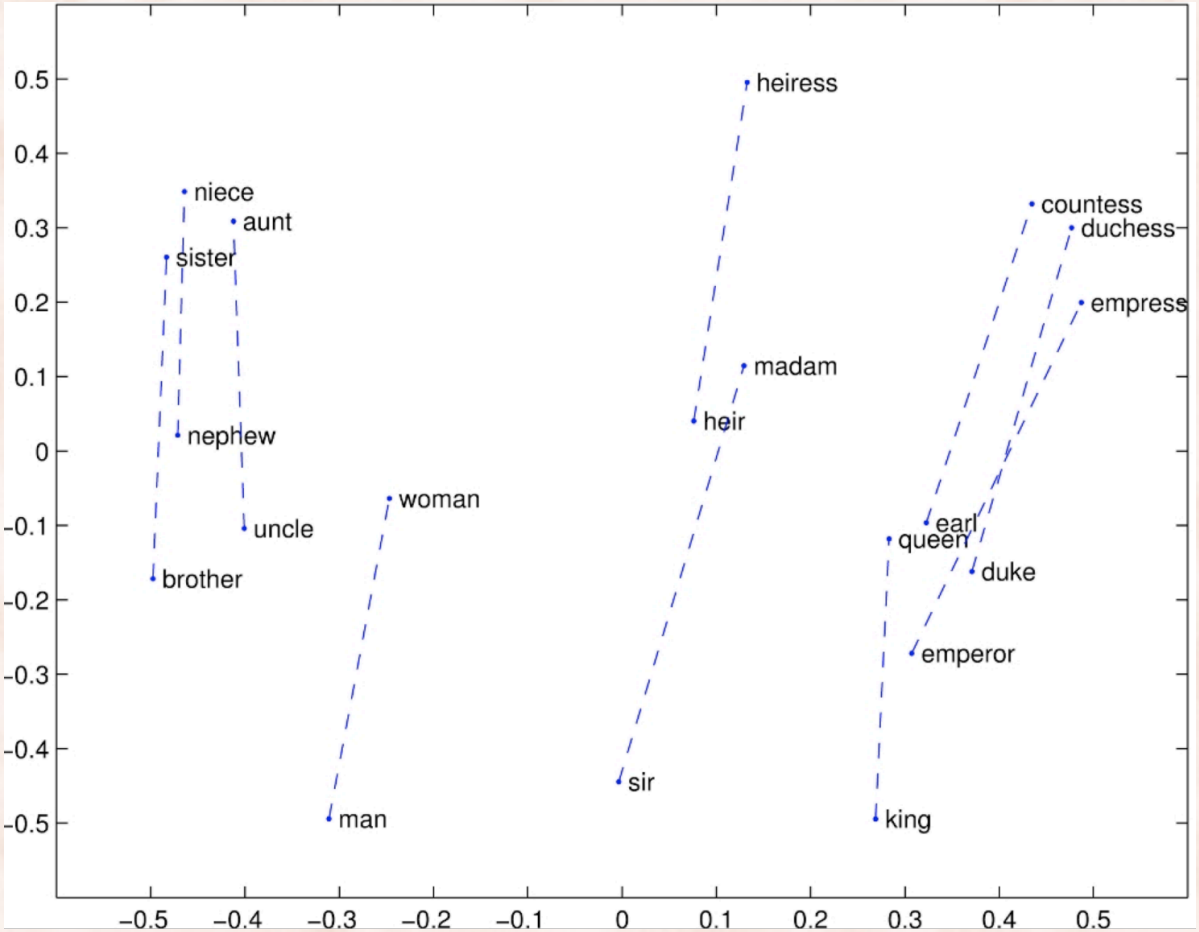


image credit: cs224n at Stanford.

Visualizations of Syntactic Relations

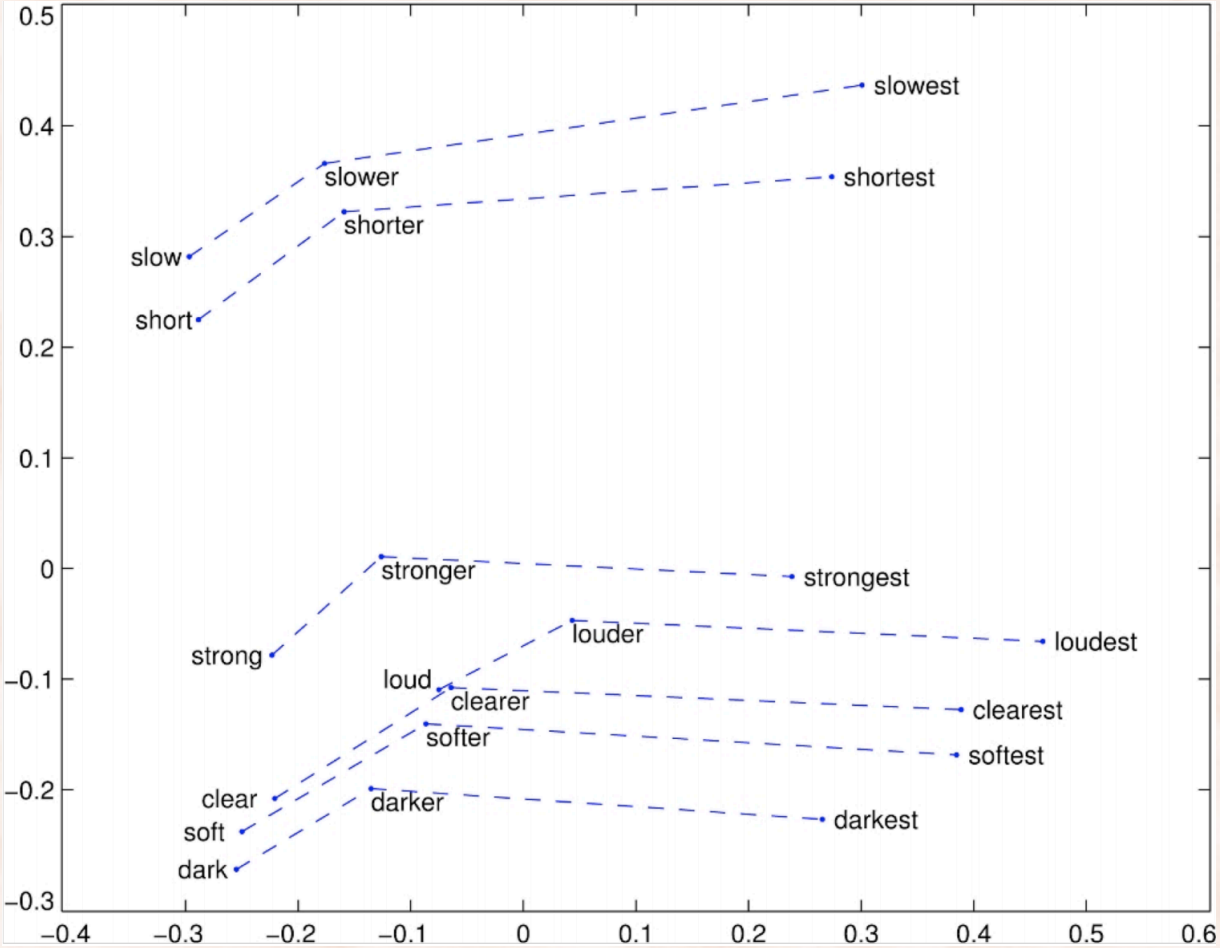


image credit: cs224n at Stanford.

Comparative Evaluation on Syntactic and Semantic Analogy Questions

- Dataset created by [Mikolov et al., ICLR 2013] and available at http://download.tensorflow.org/data/questions-words.txt
- Example of ***semantic*** relation:
  - : family
    - boy girl brother sister
    - boy girl brothers sisters
    - boy girl dad mom
    - boy girl father mother
    - boy girl grandfather grandmother
    - boy girl grandpa grandma
    - boy girl grandson granddaughter
    - boy girl groom bride
    - boy girl he she

Comparative Evaluation on Syntactic and Semantic Analogy Questions

- Dataset created by [Mikolov et al., ICLR 2013] and available at http://download.tensorflow.org/data/questions-words.txt
- Example of ***syntactic*** relation:
  - : gram2-opposite
    - acceptable unacceptable aware unaware
    - acceptable unacceptable certain uncertain
    - acceptable unacceptable clear unclear
    - acceptable unacceptable comfortable uncomfortable
    - acceptable unacceptable competitive uncompetitive
    - acceptable unacceptable consistent inconsistent
    - acceptable unacceptable convincing unconvincing
    - acceptable unacceptable convenient inconvenient
    - acceptable unacceptable decided undecided

Comparative Evaluation on Syntactic and Semantic Analogy Questions

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	67.4	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	**81.9**	**69.3**	**75.0**

Comparative Evaluation on Correlation with Human Judgments of Similarity

- WordSimilarity-353 Test Collection:
  - http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/

Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

image credit: cs224n at Stanford.

Comparative Evaluation on Correlation with Human Judgments of Similarity

- Spearman rank correlation on word similarity tasks.

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	72.7	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	58.1	37.2
GloVe	6B	65.8	72.7	77.8	53.9	38.1
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	**75.9**	**83.6**	**82.9**	**59.6**	**47.8**
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

* *Some ideas from Glove paper have been shown to improve skip-gram (SG):*

Unkown Words (OOV)

Subword Embeddings

- Word embeddings ignore the internal structure of words:

  skiing = ski + ing

- Limitation for morphologically rich languages:
  - In French or Spanish, most verbs have more than 40 different inflected forms; the Finnish language has 15 cases for nouns.
  - These languages contain many word forms that occur rarely (or not at all) in the training corpus.
    - Makes it difficult to learn good word representations.

- Improve vector representations by using character-level information => **subword embeddings**.

FastText

https://fasttext.cc/

[Bojanowski et al., FAIR'13]

- Learn representation for *character n-grams*, i.e. *subwords*.
- Each word is represented as the sum of its *n*-gram vectors:
  - Use all the *n*-grams for $3 \leq n \leq 6$.

skiing = {<sk, <ski, <skii, <skiin,
ski, skii, skiin, skiing,
kii, kiin, kiing, kiing>,
iin, iing, iing>,
ing, ing>,
ng>,
<skiing>}

  - Short n-grams (n = 4) are good to capture syntactic information.
  - Longer n-grams (n = 6) are good to capture semantic information.

FastText

https://fasttext.cc/

[Bojanowski et al., FAIR'13]

- Use a hashing function that maps *n*-grams to integers in 1 to K = 2.1 million:
  - A word is represented by its index in the word dictionary and the set of hashed *n*-grams it contains.
  - About 1.5x slower to train than word2vec Skip-Gram models.
- For OOV words, word vector = sum of subword vectors.
- Instrinsic evaluations:
  - Correlation with human judegments on word similarity datasets.
  - Word analogy tasks.

FastText Evaluation

https://fasttext.cc/

[Bojanowski et al., FAIR'13]

Trained all models on Wikipedia. Used null vectors for OOV words in word2vec Skip-Gram (sg), CBOS, and Subword Information SG (sisg-).

Semantic similarity

		sg	cbow	sisg-	sisg
AR	WS353	51	52	54	**55**
DE	GUR350	61	62	64	**70**
	GUR65	78	78	**81**	**81**
	ZG222	35	38	41	**44**
EN	RW	43	43	46	**47**
	WS353	72	**73**	71	71
ES	WS353	57	58	58	**59**
FR	RG65	70	69	**75**	**75**
RO	WS353	48	52	51	**54**
RU	HJ	59	60	60	**66**

Word Analogy Tasks

		sg	cbow	sisg
CS	Semantic	25.7	27.6	27.5
	Syntactic	52.8	55.0	77.8
DE	Semantic	66.5	66.8	62.3
	Syntactic	44.5	45.0	56.4
EN	Semantic	78.5	78.2	77.8
	Syntactic	70.1	69.9	74.9
IT	Semantic	52.3	54.7	52.3
	Syntactic	51.5	51.8	62.7

Multiple Word Vectors per Word Sense

- "Improving Word Representations via Global Context and Multiple Word Prototypes" [Huang et al., ACL 2012]
- "Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space" [Neelakantan et al., EMNLP'14].
- "Linear Algebraic Structure of Word Senses, with Applications to Polysemy" [Arora et al., TACL'18].

Word Embeddings: Extensions

- "Dependency-Based Word Embeddings" [Levy & Goldberg, ACL'14]
- Combine with co-occurrence/PPMI based word embeddings:
  - "Symmetric Pattern Based Word Embeddings for Improved Word Similarity Prediction" [Schwartz et al., CoNLL'15].
- Bilingual word embeddings:
  - "Bilingual Word Embeddings for Phrase-Based Machine Translation" [Zou et al., EMNLP'13].
  - "Multilingual Models for Compositional Distributed Semantics" [Hermann & Blunsom, ACL'14].
- Phrase, paragraph, and document embeddings:
  - "Distributed Representations of Sentences and Documents" [Le & Mikolov, ICML'14]
  - "Skip-thought vectors" [Kiros et al., NIPS'15]
  - "A hierarchical neural autoencoder for paragraphs and documents" [Li et al., EMNLP'15]
  - " KATE: K-Competitive Autoencoder for Text" [Chen & Zaki, KDD'17]

Readings

- Chapter 14 in Eisenstein.