CS182/282A: Designing, Visualizing and Understanding Deep Neural Networks

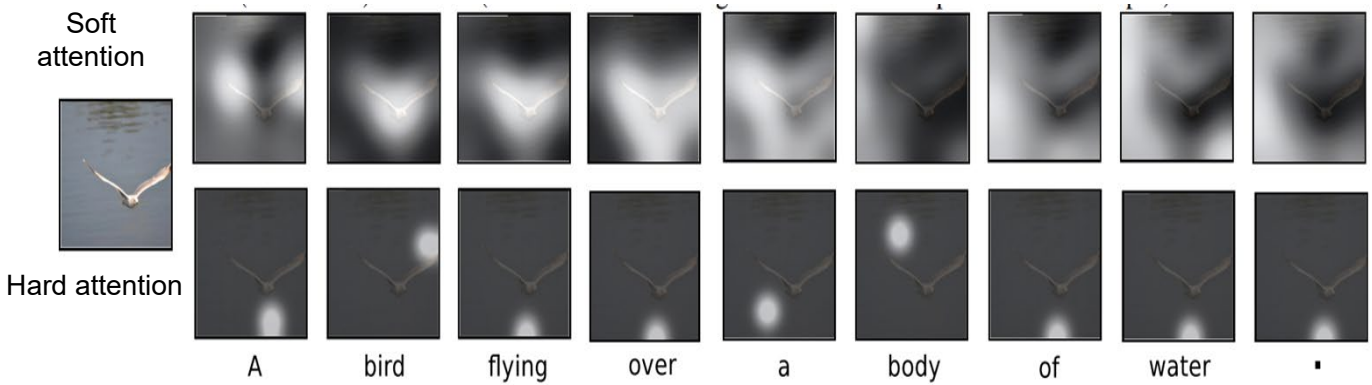
John Canny

Spring 2020

Lecture 12: Text Semantics

Some slide content from Stanford's CS224n, R. Socher

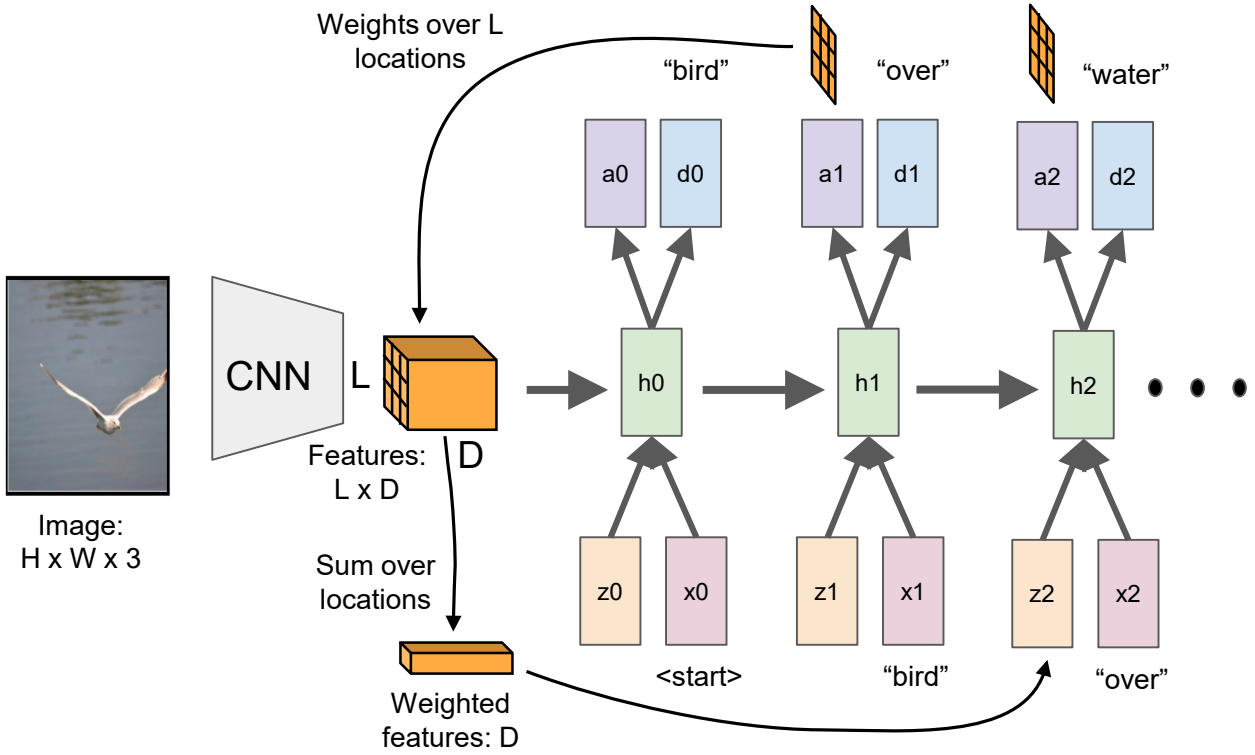
Last Time: Soft vs Hard Attention



Hard attention: Attend to a single input location, can't use gradient descent, Need **reinforcement learning.**

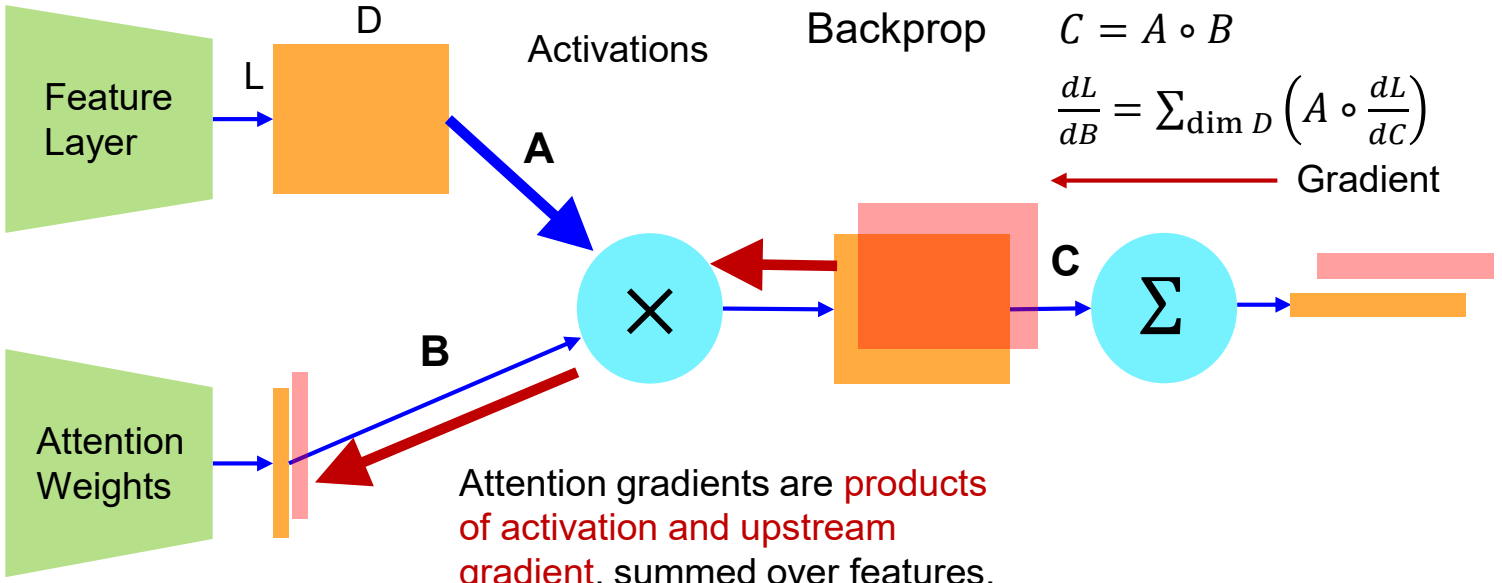
Soft attention: Compute a weighted combination (attention) over some inputs using an attention network. Can use backpropagation to train end-to-end.

Last Time: Attention for Captioning with RNNs



Last Time: Attention Mechanics: Salience

During training, the attention layers receives gradients which are the product of the upstream gradient and the feature layer activations (salience).



Text Semantics

- Propositional models
- Matrix factorization
- Word2vec
- Skip-Thought vectors
- Siamese models

Updates

Assignment 2 is due 11pm Tuesday 3/3.

Midterm should be graded some time tomorrow.

Text Semantics

- In Natural Language Processing (NLP), ***semantics*** is concerned with the meanings of texts.
- There are two main approaches:
  - ***Propositional or formal semantics:*** A block of text is to converted into a formula in a logical language, e.g. predicate calculus.
  - ***Vector representation.*** Texts are ***embedded*** into a high-dimensional space. Used for neural models…

Semantic Approaches

Propositional:

- “dog bites man” ➔ bites(dog, man)
- bites(*,*) is a binary relation. man, dog are objects.
- Probabilities can be attached.

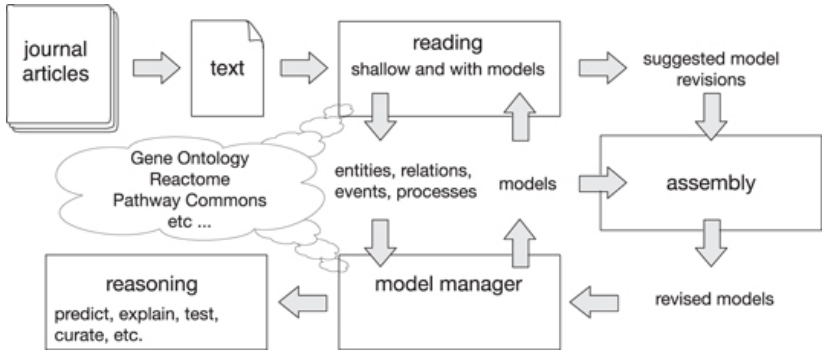
Vector representation:

- vec(“dog bites man”) = (0.2, -0.3, 1.5,…) ∈ $\mathbb{R}^n$
- Sentences similar in meaning should be close to this vector.

Propositional Semantics

- Allow logical inferences “Socrates is a man,” + “all men are mortal” → “Socrates is mortal”
- Important for inference in well-defined domains, e.g. inferring ***gene regulation*** from medical journals. Articles are in “natural language” but with a carefully controlled, standard vocabulary.

See DARPA’s “Big Mechanism” project



From “DARPA’s Big Mechanism program” Paul R Cohen, Phys. Biol. 12 (2015)

Vector Embedding of Words

Much recent work on text semantics has used word embeddings and **bag-of-words** representation:

vec(“dog”) = (0.2, -0.3, 1.5,…)

vec(“bites”) = (0.5, 1.0, -0.4,…)

vec(“man”) = (-0.1, 2.3, -1.5,…)

vec(“dog bites man”) = (0.6, 3.0, -0.4,…)

But notice that vec(“dog bites man”) = vec(“man bites dog”) ☹

Vector Embedding: Word Similarity

Word embeddings depend on a notion of ***word similarity***.

A very useful definition is *paradigmatic* similarity (Saussure 1916):

Similar words occur in ***similar contexts***. They are ***exchangable***.

This definition supports unsupervised learning: cluster or embed words according to their contexts.

government debt problems turning into banking crises as has happened in
saying that Europe needs unified banking regulation to replace the hodgepodge

Surrounding words represent “banking”

Paradigmatic Similarity

"My dog and I cuddled in front of the TV"

"My dog barked"

"…that dog is black and brown…"

"…her dog chased the ball…"

"…our dog Rex is lively but our dog Stanley is a sleeper…"

"…that dog has a long tail and heavy winter coat…"

We could *replace* the word "dog" with another word like "perro". A user who didn't know what a perro is should be able to figure it out with enough examples.

Some of these sentences contexts will be *shared* with related words like "cat". Those words will have similar embeddings.

Vector Embedding: Dimension Reduction

An embedding method should give different embeddings to different words, but a naïve (one-hot encoding) is very expensive:

Dog [0, 0, 0, 0,…, 0, 0, 1, 0, 0, 0, 0,…,0]

Man [0, 0, 0, 0,…, 0, 0, 0, 0, 1, 0, 0,…,0]

} Vector size = vocabulary size

But words are too different (inner products always zero) in this representation.

Instead we can use vectors of contexts (counts of other nearby words):

Dog [2, 5, 0, 1,…, 11, 3, 0, 5, 8, 1, 9,…,4]

Man [5, 3, 2, 0,…, 6, 2, 3, 1, 0, 8, 3,…,5]

} Vector size = vocabulary size

Captures the semantics better: Similar words will have large inner products.

But dimension is too large (vocab size). We can use dimension reduction methods like PCA

Vector Embedding: Dimension Reduction

LSA: Dimensions of a few 100 are common

Word2vec: Dimensions of around 300 are common.

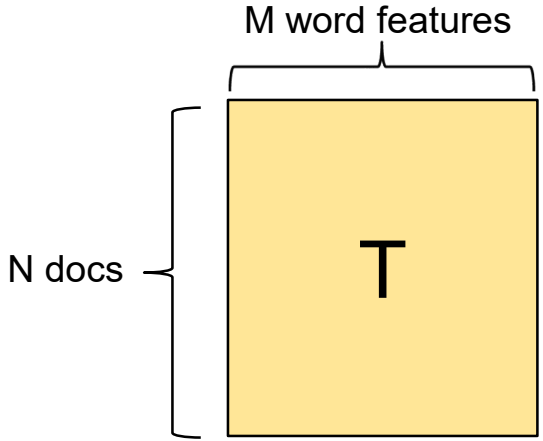
State-of-the-art sentence embedding methods may use dimensions of several thousand.

Word Embedding: Latent Semantic Analysis

Latent semantic analysis processes documents in bag-of-words (BoW) format (1988).

Encode a set of documents in a matrix T:

T_{ij} is the count* of word j in document i. Most entries are 0.



* Often tfidf or other “squashing” functions of the word counts are used.

Word Matrix Example

Example corpus: (R. Socher)

- I like deep learning.
- I like NLP.
- I enjoy flying.

	I	like	deep	learning	NLP	enjoy	flying
S1	1	1	1	1	0	0	0
S2	1	1	0	0	1	0	0
S3	1	0	0	0	0	1	1

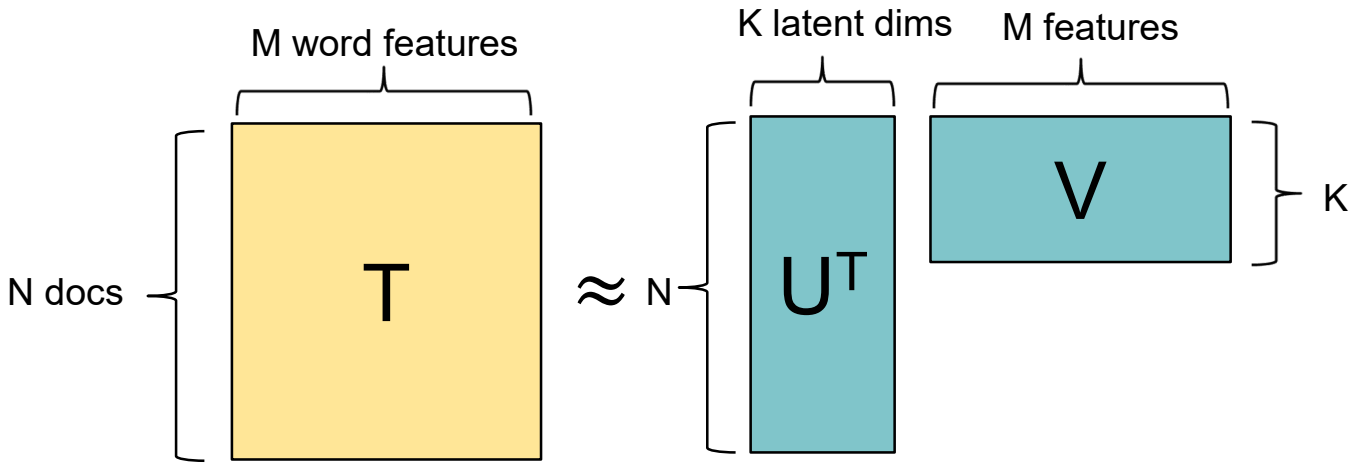
Note: this matrix is extremely sparse (mostly zero), which allows efficient sparse matrix methods to be used.

Word Embedding: Latent Semantic Analysis

Given a bag-of-words matrix T, compute a factorization $T \approx U^T * V$ (e.g. a best L_2 approximation to T) where $K << M, N$

Factors encode similar ***whole document contexts***.

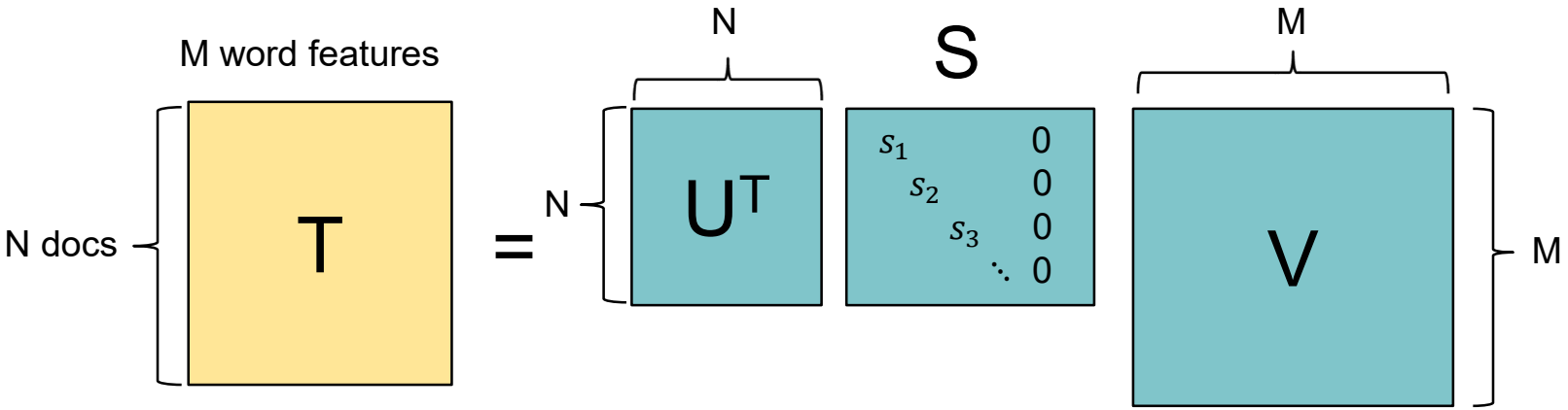
Factors are rows of V.



Singular Value Decomposition

$$T = U^T S V$$

where U and V are orthogonal, normal ($U^T U = U U^T = I$), S a rectangular diagonal matrix of singular values:



Singular Values and Eigenvalues

The singular values S of a matrix T can be computed from the eigenvalues of another matrix:

A. Eigenvalues of T

B. Square roots of Eigenvalues of TT^T

C. Eigenvalues of TT^T

D. Square roots of Eigenvalues of T^TT

Oops!

The singular values S of a matrix T can be computed from the eigenvalues of another matrix:

A. Eigenvalues of T

T is not square, its eigenvalues are not defined.

Try Again

Continue

Correct!

The singular values S of a matrix T can be computed from the eigenvalues of another matrix:

B. Square roots of Eigenvalues of TT^T

Since $T = U^T SV$, then $TT^T = U^T SVV^T SU = U^T S^2 U$, and the square roots are real because TT^T is positive semi-definite.

Try Again

Continue

Oops!

The singular values S of a matrix T can be computed from the eigenvalues of another matrix:

C. Eigenvalues of TT^T

Since $T = U^T SV$, then $TT^T = U^T SVV^T SU = U^T S^2 U$. So its eigenvalues are the squares of S and not S itself.

Try Again

Continue

Correct!

The singular values S of a matrix T can be computed from the eigenvalues of another matrix:

D. Square roots of Eigenvalues of $T^T T$

Since $T = U^T SV$, then $T^T T = V^T SUU^T SV = V^T S^2 V$, and the square roots are real because $T^T T$ is positive semi-definite.

Try Again

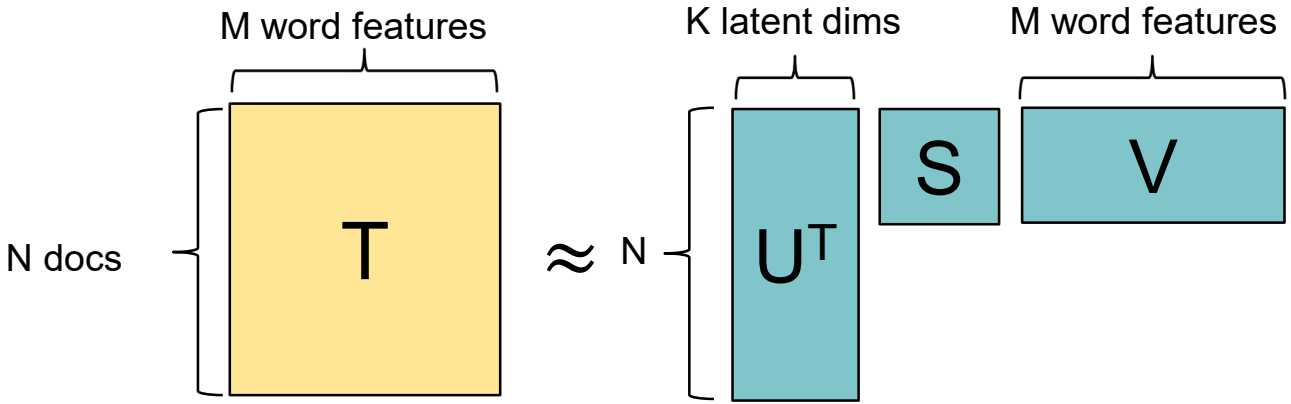
Continue

Embedding: Latent Semantic Analysis

We can also compute an **approximate** SVD by reducing the inner dimensions of U, S, and V, to some value $K \leq \min(M, N)$, keeping the largest singular values.

$v = tV^T$ is an embedding of the document in the latent space.

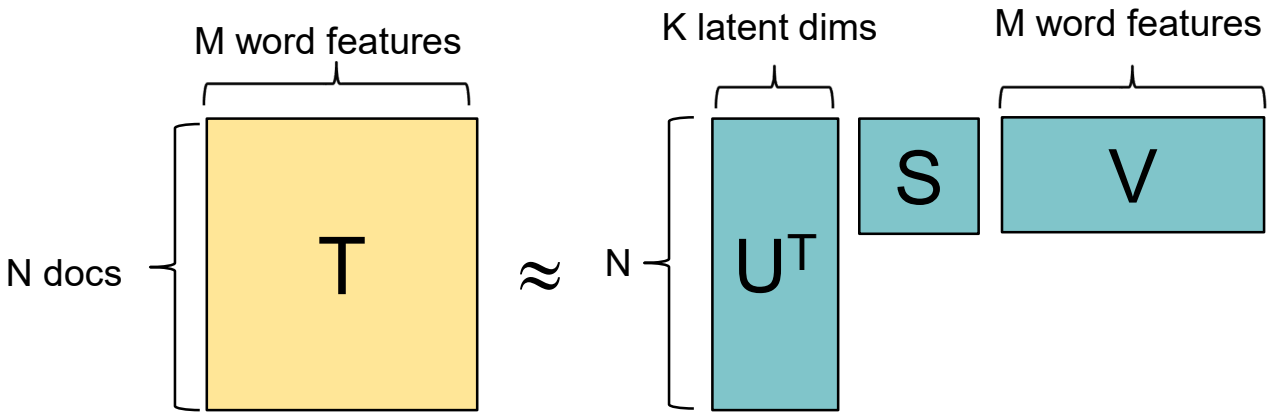
$t' = vV = tV^TV$ is the decoding of the document from its embedding.



Embedding: Latent Semantic Analysis

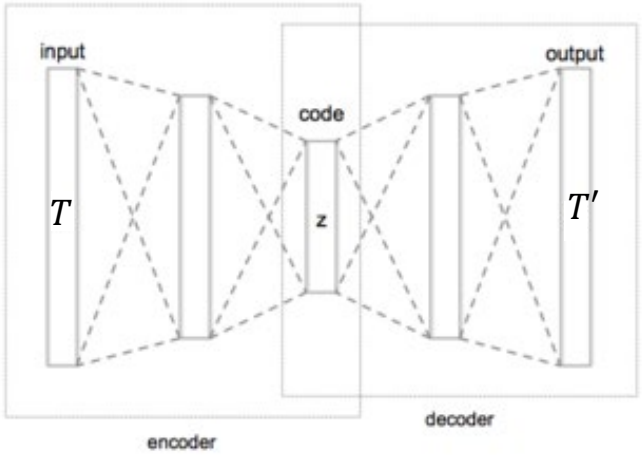
$t' = vV = tV^T V$ is the decoding of the sentence from its embedding.

An approximate SVD (Singular Value Decomposition) factorization gives the ***best possible reconstructions*** of the documents t' from their embeddings.



Embedding: Latent Semantic Analysis

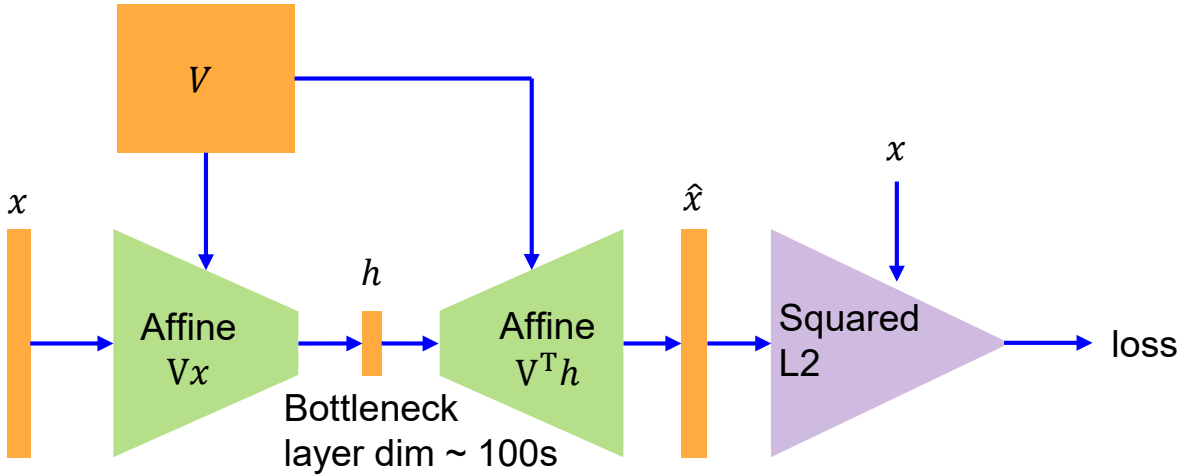
Thus LSA is an **auto-encoding** method, in fact the optimal linear auto-encoder for documents with L2 error.



$$Z = TV^T \qquad T' = ZV$$

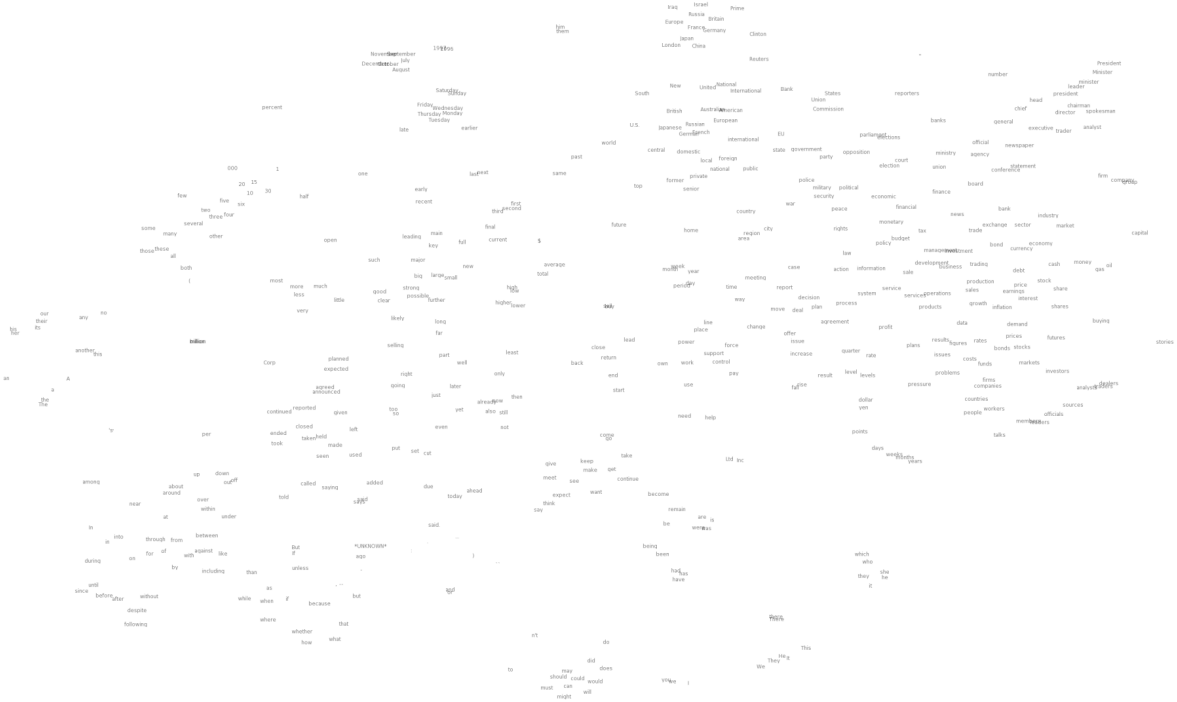
LSA as a Deep Network

Input x is a BoW encoded input sentence vector (column vector this time, so matrix products are transposed)



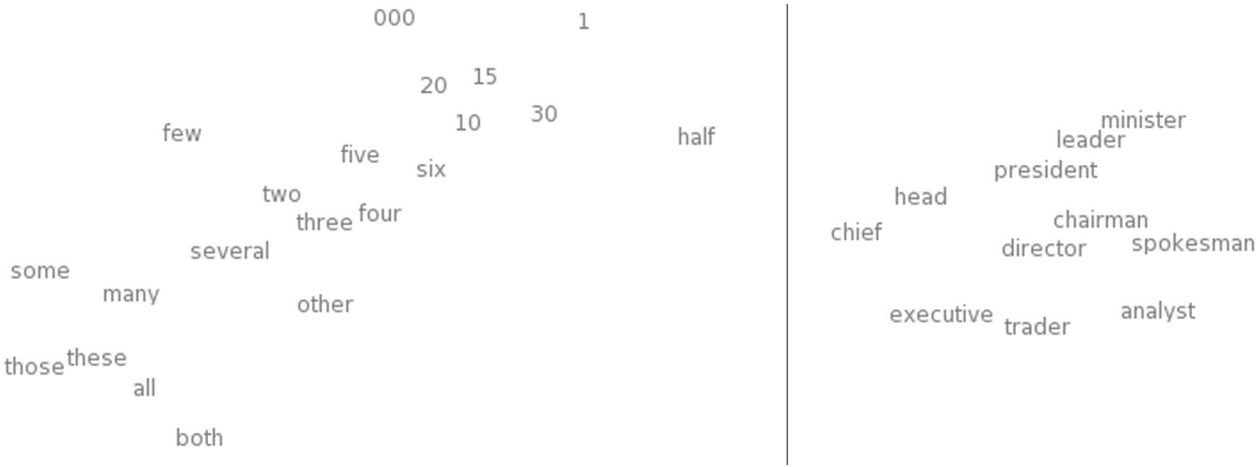
Compute loss over input sentences x. Note the computed V agrees with the SVD matrix V up to a rotation V' = RV. The predicted $\hat{x}$ is the same.

t-SNE of Word Embeddings



From: “Word representations: A simple and general method for semi-supervised learning” Joseph Turian, Lev Ratinov, Yoshua Bengio, ACL 2010.

t-SNE of Word Embeddings



Left: Number Region; Right: Jobs Region

from “Deep Learning, NLP, and Representations” by Chris Olah. See also

http://colah.github.io/posts/2015-01-Visualizing-Representations/

Word2vec: Local contexts

Instead of entire documents, Word2vec uses words a few positions away from each center word. The pairs of center word/context word are called **“skip-grams.”**

“It was **a bright cold day in April, and** the clocks were striking”

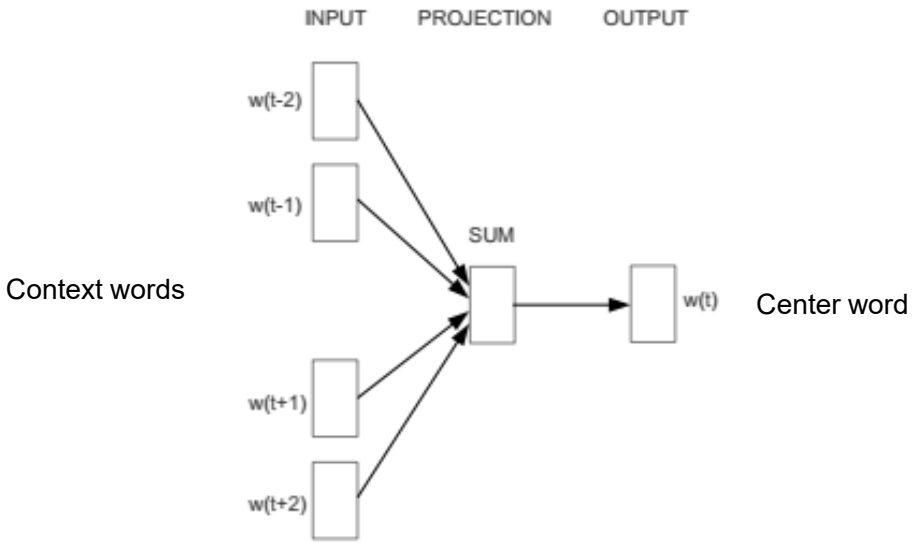
Center word: red

Context words: blue

Word2vec considers all words as center words, and all their context words.

Word2vec: Local contexts

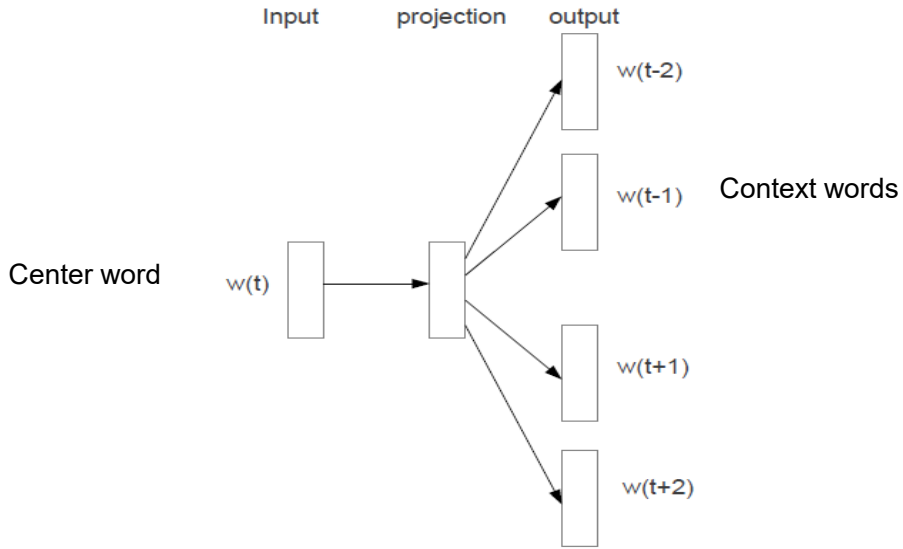
Models can predict center word from context, CBOW model. “Contextual Bag-Of-Words”. The context words are unordered.



Distributed Representations of Words and Phrases and their Compositionality
Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean, NIPS 2013

Word2vec: Local contexts

The pairs of center word/context word are called **“skip-grams.”** Typical distances are 3-5 word positions. Skip-gram model:



Distributed Representations of Words and Phrases and their Compositionality
Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean, NIPS 2013

Word2vec: Improving the loss

SVD minimizes the squared error in distances between original points and their embeddings.

This helps to minimize the errors between pairwise point distances in the original and embedded spaces. But this favors large distances instead of small ones.

Ideally we would preserve the distance errors for *close* points, like t-SNE.

Word2vec: Local contexts

Word2vec minimizes a softmax loss $-\log p\,(j|i)$ for each output word given an input word:

$$p(j|i) = \frac{\exp\left(u_j^T v_i\right)}{\sum_{k=1}^{V} \exp(u_k^T v_i)}$$

Where j is the output word, i is the input word. j ranges over a context of $\pm$ 3-5 positions around the input word.

u is an output embedding vector.

v is an input embedding vector.

Word2vec can be implemented with standard DNN toolkits, by backpropagating to optimize u and v.

Word2vec Loss

Word2vec minimizes a softmax loss $-\log p\,(j|i)$ for each output word given an input word:

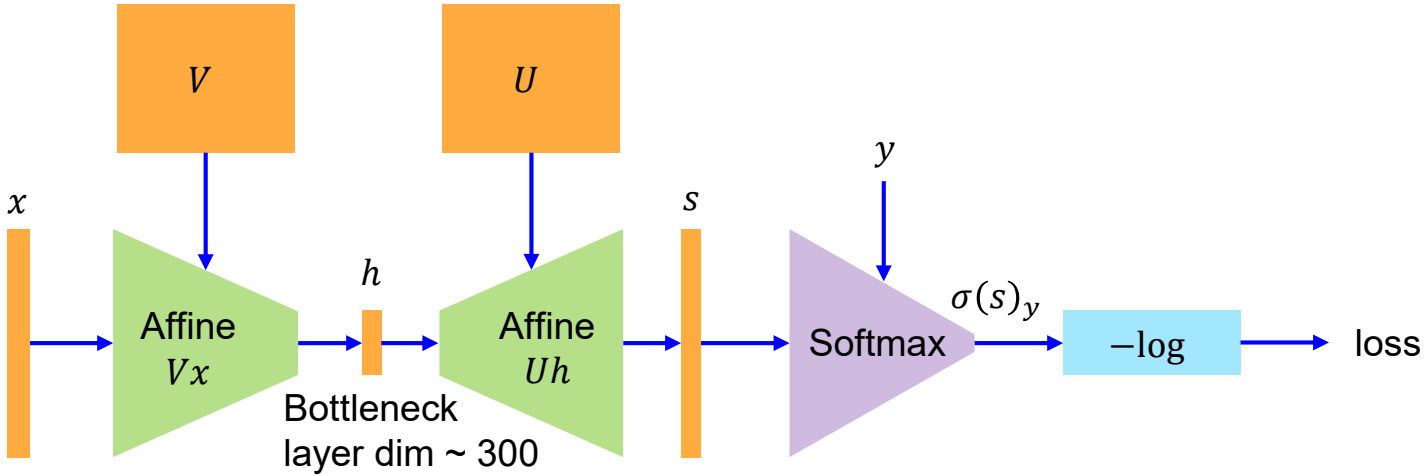
$$p(j|i) = \frac{\exp\left(u_j^T v_i\right)}{\sum_{k=1}^{V} \exp(u_k^T v_i)}$$

The probability is largest for the most similar word j to i.

Unfortunately, Word2vec uses the cross entropy – the log of the above softmax, which re-emphasizes distant word relations (but not as much as SVD).

Word2vec: As a Deep Network

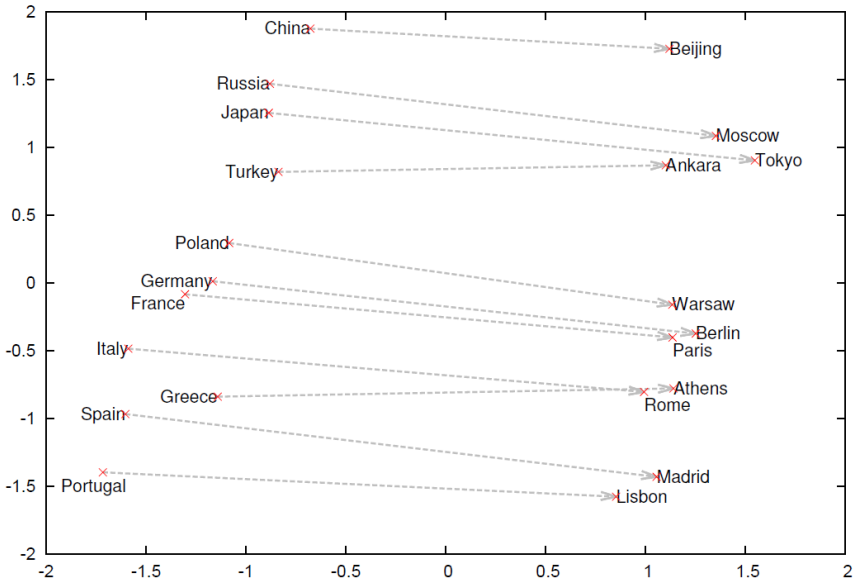
Input x is a one-hot encoded input word, y is the index of the output (context) word:



Compute loss over all (x,y), (input, context) word pairs.

Word2vec: Local contexts

Local contexts capture more information about relations and properties than LSA:

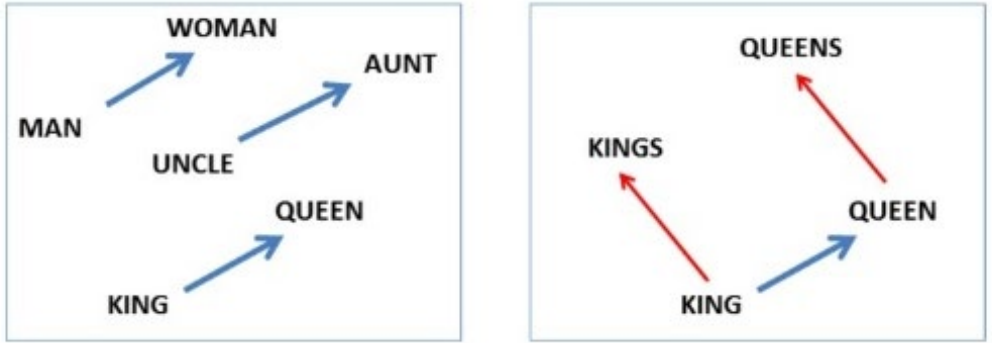


Composition

Algebraic relations:

vec(“woman")−vec(“man") ≃ vec(“aunt")−vec(“uncle")

vec(“woman")−vec(“man") ≃ vec(“queen")−vec(“king")



From “Linguistic Regularities in Continuous Space Word Representations”
Tomas Mikolov , Wen-tau Yih, Geoffrey Zweig, NAACL-HLT 2013

Relations Learned by Word2vec

Word2vec model computed from 6 billion word corpus of news articles

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

“Efficient Estimation of Word Representations in Vector Space” Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Arxiv 2013

Relations Learned by Word2vec

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

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

“Efficient Estimation of Word Representations in Vector Space” Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Arxiv 2013

Relations Learned by Word2vec

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

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

"Efficient Estimation of Word Representations in Vector Space" Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Arxiv 2013

Relations Learned by Word2vec

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

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

"Efficient Estimation of Word Representations in Vector Space" Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Arxiv 2013

Relations Learned by Word2vec

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

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

"Efficient Estimation of Word Representations in Vector Space" Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Arxiv 2013

Relations Learned by Word2vec

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

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

"Efficient Estimation of Word Representations in Vector Space" Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Arxiv 2013

Relations Learned by Word2vec

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

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

"Efficient Estimation of Word Representations in Vector Space" Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Arxiv 2013

Relations Learned by Word2vec

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

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

"Efficient Estimation of Word Representations in Vector Space" Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Arxiv 2013

Word2vec Visualizations

- Embedding projector

Word2vec: Criticisms

- Use of cross-entropy loss puts much emphasis on small (unlikely) combinations of word/contexts.

- Very expensive to normalize the softmax over all words (can be 10s or 100s of thousands of them). Approximations are often used.

- Uses heuristic down-weighting of frequent words

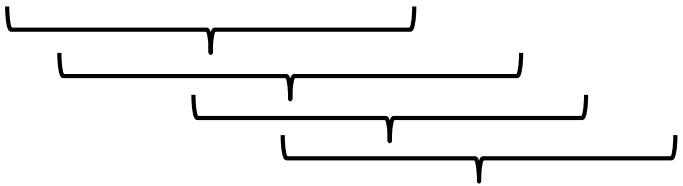
Co-occurrence Matrices

C_{ij} counts the number of documents containing both words i and j.

Full-document co-occurrence matrix gives results similar to LSA.

We can also use a window-based co-occurrence matrix, by considering words that co-occur within a given distance from each other. This gives results similar to word2vec.

“It might take some time for the results of this paper…”



Window-Based Co-occurrence Matrix Example

Example corpus: (R. Socher)

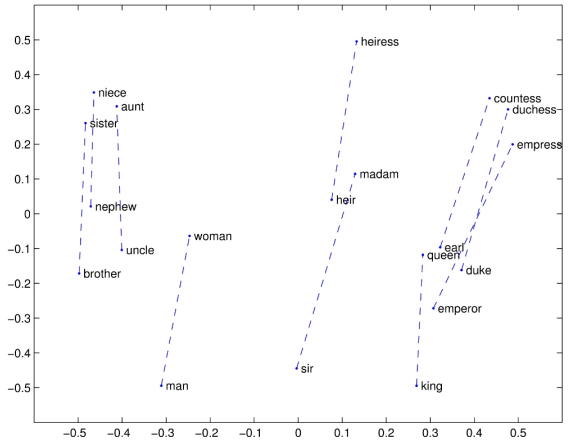
- I like deep learning.
- I like NLP.
- I enjoy flying.

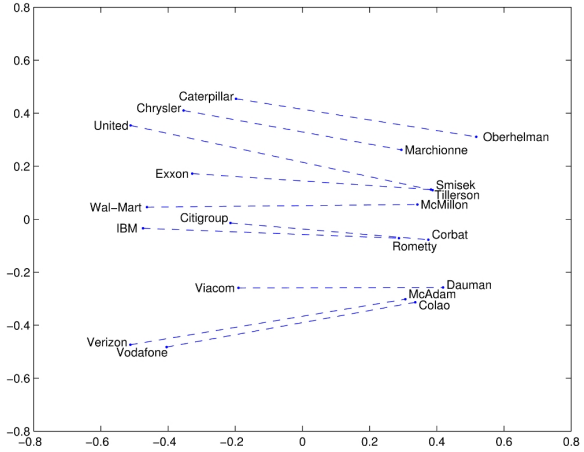
What's the window size here?

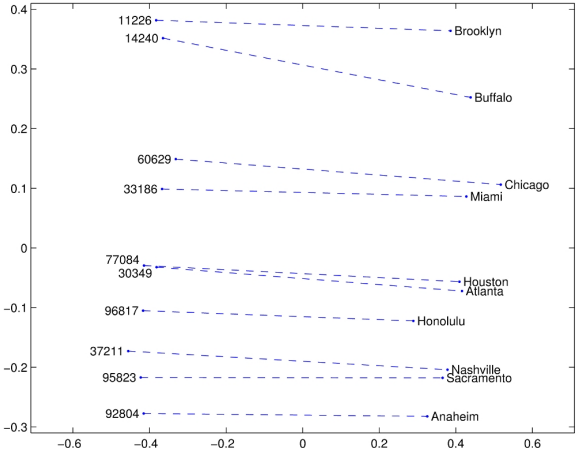
counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

A different model: GloVe

We can optimize a model explicitly for analogies:







A different model: GloVe

Let C_{ij} denote the number of times that word j occurs in the context of word i.

GloVe Minimizes:

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

Where u_i is the word embedding, v_j is the context word embedding, b_i and $\tilde{b}_j$ are bias vectors, and $f(.)$ is a function that satisfies:

$f(0) = 0$

$f(x)$ non-decreasing

$f(x)$ “saturates” – not too large for large x.

A different model: GloVe

Let C_{ij} denote the number of times that word j occurs in the context of word i.

Can have $C_{ij} = 0$ so $\log C_{ij}$ undefined, but $f(C_{ij})$ is zero in such cases, and that term in the sum is treated as 0:

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

A sensible choice of $f(.)$ is :

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} .$$

Typical α = ¾, $x_{\max}$=100

Glove results

Nearest words to frog:

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

litoria

leptodactylidae

rana

eleutherodactylus

Lexical and Compositional Semantics

Lexical Semantics: focuses on the meaning of individual words.

Compositional Semantics: meaning depends on the words, and on how they are combined.

Beyond Bag-Of-Words: Skip-Thought Vectors

The models we discussed so far embed texts as **the sum of their words** (lexical semantics).

Clearly there is a lot missing from these representations:

“man bites dog” = “dog bites man”

“the quick, brown fox jumps over the lazy dog” =

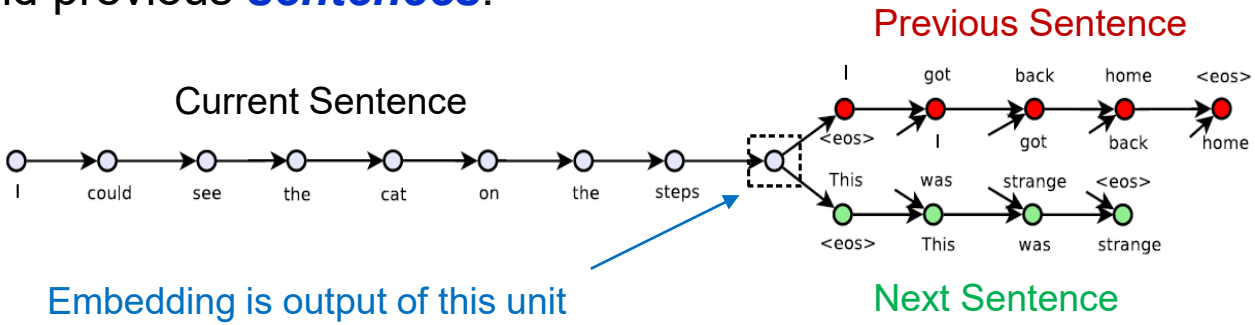
“the lazy fox over the brown dog jumps quick”

…

How can we model text structure as well as word meanings?

Beyond Bag-Of-Words: Skip-Thought Vectors

Skip-thought embeddings use sequence-to-sequence RNNs to predict the next and previous ***sentences***.



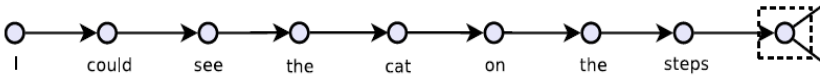
The output state vector of the boundary layer (dotted box) forms the embedding. RNN units are GRU units.

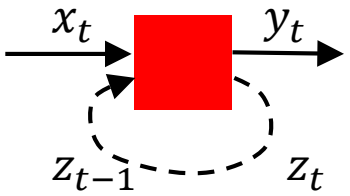
Once the network is trained, we can discard the red and green sections of the network.

From “Skip-Thought Vectors,” Ryan Kiros et al., Arxiv 2015.

Beyond Bag-Of-Words: Skip-Thought Vectors

Skip-thought embeddings use sequence-to-sequence RNNs to predict the next and previous ***sentences***.



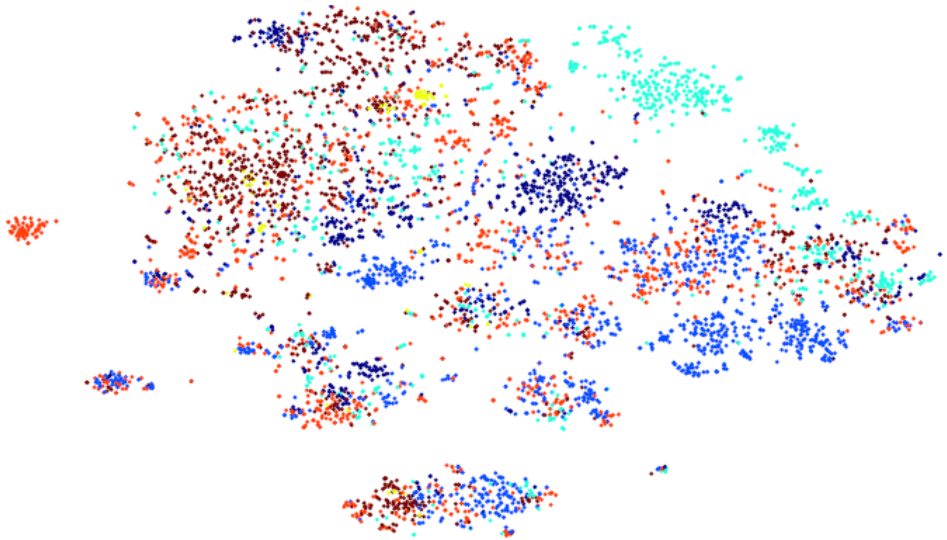


Encoding doesn't require backpropagation, so we can represent the encoder as a (truly) recurrent network.

Thus we can encode longer units of text: sentences or paragraphs.

Embedding of TREC queries

Points are colored by query type (t-SNE embedding):



From “Skip-Thought Vectors,” Ryan Kiros et al., Arxiv 2015.

Sentence Similarity

Query and nearest sentence
he ran his hand inside his coat , double-checking that the unopened letter was still there . he slipped his hand between his coat and his shirt , where the folded copies lay in a brown envelope .
im sure youll have a glamorous evening , she said , giving an exaggerated wink . im really glad you came to the party tonight , he said , turning to her .
although she could tell he had n't been too invested in any of their other chitchat , he seemed genuinely curious about this . although he had n't been following her career with a microscope , he 'd definitely taken notice of her appearances .
an annoying buzz started to ring in my ears , becoming louder and louder as my vision began to swim . a weighty pressure landed on my lungs and my vision blurred at the edges , threatening my consciousness altogether .
if he had a weapon , he could maybe take out their last imp , and then beat up errol and vanessa . if he could ram them from behind , send them sailing over the far side of the levee , he had a chance of stopping them .
then , with a stroke of luck , they saw the pair head together towards the portaloos . then , from out back of the house , they heard a horse scream probably in answer to a pair of sharp spurs digging deep into its flanks .
" i 'll take care of it , " goodman said , taking the phonebook . " i 'll do that , " julia said , coming in .
he finished rolling up scrolls and , placing them to one side , began the more urgent task of finding ale and tankards . he righted the table , set the candle on a piece of broken plate , and reached for his flint , steel , and tinder .

Approximately two weeks of training on a billion-word Books corpus

Semantic Relatedness Evaluation

SICK semantic relatedness task: score sentences for semantic similarity from 1 to 5 (average of 10 human ratings)

Sentence A: A man is jumping into an empty pool
Sentence B: There is no biker jumping in the air
Relatedness score: 1.6

Sentence A: Two children are lying in the snow and are making snow angels
Sentence B: Two angels are making snow on the lying children
Relatedness score: 2.9

Sentence A: The young boys are playing outdoors and the man is smiling nearby
Sentence B: There is no boy playing outdoors and there is no man smiling
Relatedness score: 3.6

Sentence A: A person in a black jacket is doing tricks on a motorbike
Sentence B: A man in a black jacket is doing tricks on a motorbike
Relatedness score: 4.9

Semantic Relatedness Evaluation

Note: a separate model is trained to predict the scores from pairs of embedded sentences.

Sentence A: A man is jumping into an empty pool
Sentence B: There is no biker jumping in the air
Relatedness score: 1.6

Sentence A: Two children are lying in the snow and are making snow angels
Sentence B: Two angels are making snow on the lying children
Relatedness score: 2.9

Sentence A: The young boys are playing outdoors and the man is smiling nearby
Sentence B: There is no boy playing outdoors and there is no man smiling
Relatedness score: 3.6

Sentence A: A person in a black jacket is doing tricks on a motorbike
Sentence B: A man in a black jacket is doing tricks on a motorbike
Relatedness score: 4.9

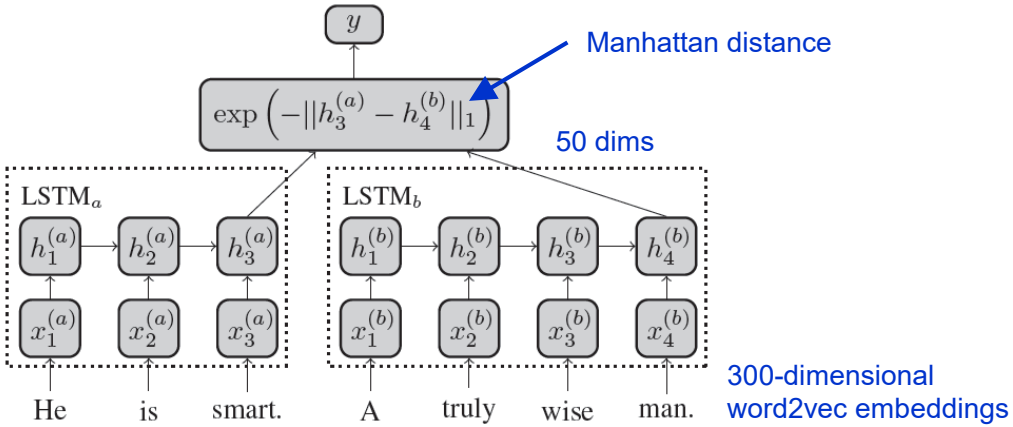
Semantic Relatedness Evaluation

SICK semantic relatedness scores for skip-thought methods:

Method	r	ρ	**MSE**
Illinois-LH [18]	0.7993	0.7538	0.3692
UNAL-NLP [19]	0.8070	0.7489	0.3550
Meaning Factory [20]	0.8268	0.7721	0.3224
ECNU [21]	0.8414	–	–
Mean vectors [22]	0.7577	0.6738	0.4557
DT-RNN [23]	0.7923	0.7319	0.3822
SDT-RNN [23]	0.7900	0.7304	0.3848
LSTM [22]	0.8528	0.7911	0.2831
Bidirectional LSTM [22]	0.8567	0.7966	0.2736
Dependency Tree-LSTM [22]	**0.8676**	**0.8083**	**0.2532**
uni-skip	0.8477	0.7780	0.2872
bi-skip	0.8405	0.7696	0.2995
combine-skip	0.8584	0.7916	0.2687
combine-skip+COCO	0.8655	0.7995	0.2561

A Siamese Network for Semantic Relatedness

This network is trained on pairs of sentences a, b with a similarity label y.



Parameters are shared between the two networks.

From “Siamese Recurrent Architectures for Learning Sentence Similarity” Jonas Mueller, Aditya Thyagarajan, AAAI-2016

A Siamese Network for Semantic Relatedness

The network is trained on Semeval similar sentence pairs, expanded by substituting for random words using WordNet (a dataset of synonyms).
Results:

Method	r	ρ	**MSE**
Illinois-LH (Lai and Hockenmaier 2014)	0.7993	0.7538	0.3692
UNAL-NLP (Jimenez et al. 2014)	0.8070	0.7489	0.3550
Meaning Factory (Bjerva et al. 2014)	0.8268	0.7721	0.3224
ECNU (Zhao, Zhu, and Lan 2014)	0.8414	–	–
Skip-thought+COCO (Kiros et al. 2015)	0.8655	0.7995	0.2561
Dependency Tree-LSTM (Tai, Socher, and Manning 2015)	0.8676	0.8083	0.2532
ConvNet (He, Gimpel, and Lin 2015)	0.8686	0.8047	0.2606
MaLSTM	**0.8822**	**0.8345**	**0.2286**

From “Siamese Recurrent Architectures for Learning Sentence Similarity” Jonas Mueller, Aditya Thyagarajan, AAAI-2016

A Siamese Network for Semantic Relatedness

The network is trained on Semeval similar sentence pairs, expanded by substituting for random words using WordNet (a dataset of synonyms).
Results:

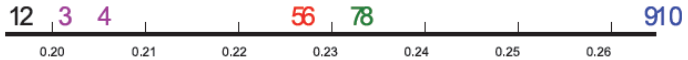
Method	r	ρ	**MSE**
Illinois-LH (Lai and Hockenmaier 2014)	0.7993	0.7538	0.3692
UNAL-NLP (Jimenez et al. 2014)	0.8070	0.7489	0.3550
Meaning Factory (Bjerva et al. 2014)	0.8268	0.7721	0.3224
ECNU (Zhao, Zhu, and Lan 2014)	0.8414	–	–
Skip-thought+COCO (Kiros et al. 2015)	0.8655	0.7995	0.2561
Dependency Tree-LSTM (Tai, Socher, and Manning 2015)	0.8676	0.8083	0.2532
ConvNet (He, Gimpel, and Lin 2015)	0.8686	0.8047	0.2606
MaLSTM	**0.8822**	**0.8345**	**0.2286**

r = Pearson correlation, ρ = Spearman's rank correlation.

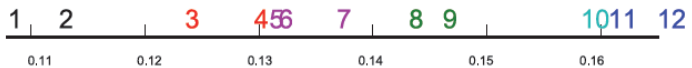
Moral: Train for your evaluation metric!

Hidden Unit Factors 1,2, and 6

1 There is no man pointing at a car
2 The woman is not playing the flute
3 The man is not riding a horse
4 A man is pointing at a silver sedan
5 The woman is playing the flute
6 A man is riding a horse



1 Two kids are bouncing on colorful balls
2 Two children are bouncing on colorful balls
3 The golden dog is running through a field of tall grass
4 A brown dog is running through tall green grass
5 A woman is putting on makeup carefully
6 A woman is carefully removing her makeup
7 A woman is applying cosmetics to her eyelid
8 A woman is carefully applying cosmetics to her eyelid
9 There is no woman cutting potatoes
10 A woman is slicing carrots



1 The cat is running across the gravel
2 A cat is playing a keyboard
3 The brown animal is jumping in the air
4 The animal with big eyes is eating
5 A dog is bouncing on a trampoline
6 A dog is running on the ground
7 A dog is running on the road
8 Several boys are jumping on a trampoline
9 A little boy is running on the ground and playing with a little girl
10 Someone is playing a piano
11 A man is running on the road
12 A man is playing an electronic keyboard

Semantic Entailment Evaluation

SICK semantic entailment task: score sentences for relations: ENTAILMENT, CONTRADICTION, NEUTRAL:

Sentence A: Two teams are competing in a football match
Sentence B: Two groups of people are playing football
Entailment judgment: ENTAILMENT

Sentence A: The brown horse is near a red barrel at the rodeo
Sentence B: The brown horse is far from a red barrel at the rodeo
Entailment judgment: CONTRADICTION

Sentence A: A man in a black jacket is doing tricks on a motorbike
Sentence B: A person is riding the bicycle on one wheel
Entailment judgment: NEUTRAL

Semantic Entailment for MaLSTM

Method	Accuracy
Illinois-LH (Lai and Hockenmaier 2014)	**84.6**
ECNU (Zhao, Zhu, and Lan 2014)	83.6
UNAL-NLP (Jimenez et al. 2014)	83.1
Meaning Factory (Bjerva et al. 2014)	81.6
Reasoning-based n-best (Lien and Kouylekov 2015)	80.4
LangPro Hybrid-800 (Abzianidze 2015)	81.4
SNLI-transfer 3-class LSTM (Bowman et al. 2015)	80.8
MaLSTM features + SVM	84.2

Summary

- Embed text using word vectors
- Auto-encoder perspective: LSA
- Local contexts: Word2vec
- Bespoke (custom) performance measures for analogies: GloVe.
- Skip-Thought vectors
- Siamese models