Deep Neural Networks

Embeddings

16/17

Robert Bogucki

Embeddings

- Embedding is a mapping from one “world” into another one
- $f : X \to Y$
- Doesn’t have to be injective

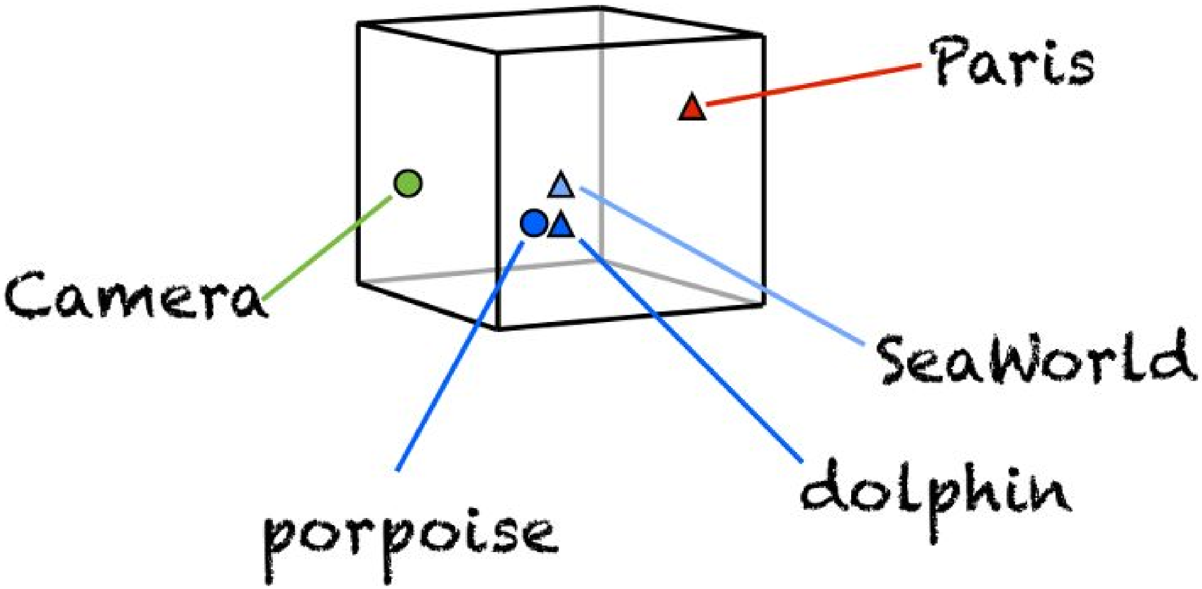
Most common scenario:

- X is either **high-dimensional** (e.g. tons of features) and/or inconvenient (for a machine) to work with (e.g. words)
- Y is a **vector space** with **lower dimensionality** and is convenient to work with (for a machine)

Word embedding

W(“cat")=(0.2, -0.4, 0.7, ...)

W(“mat")=(0.0, 0.6, -0.1, ...)



Sentence embedding

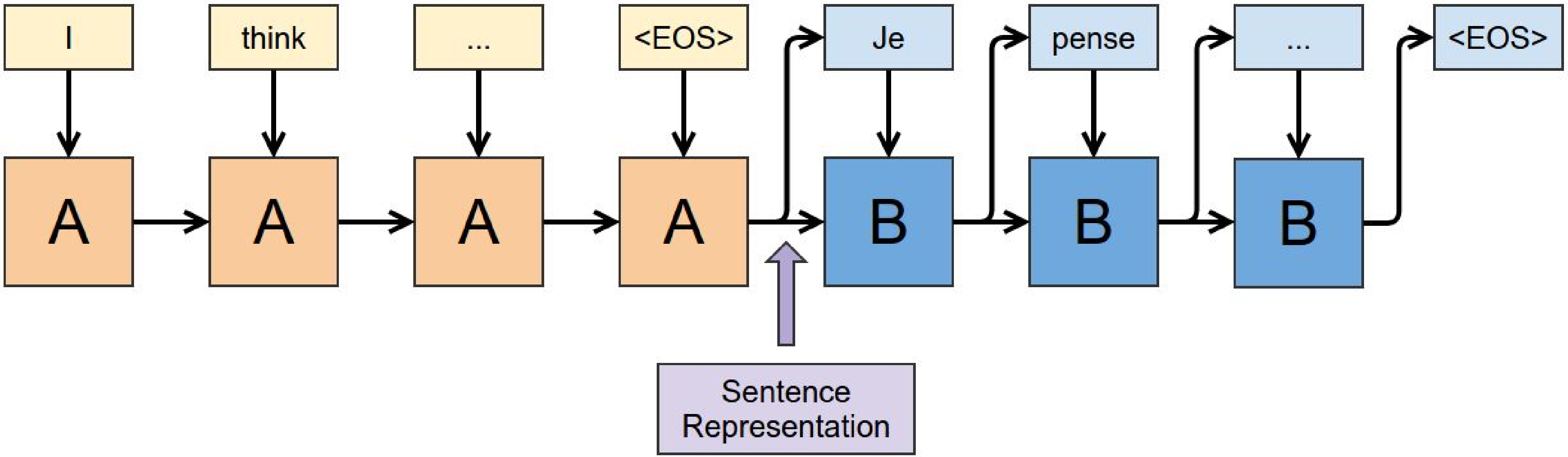
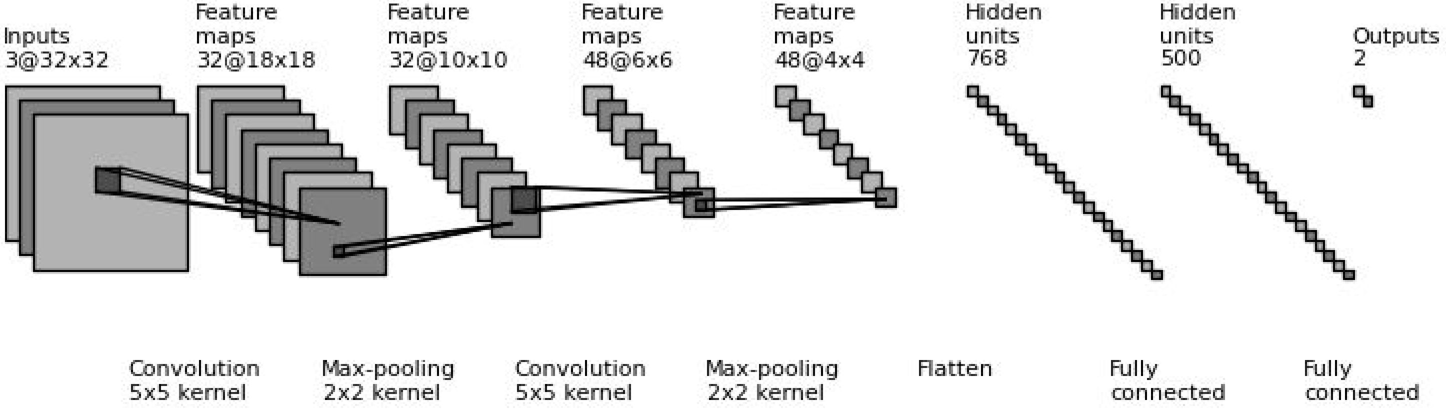
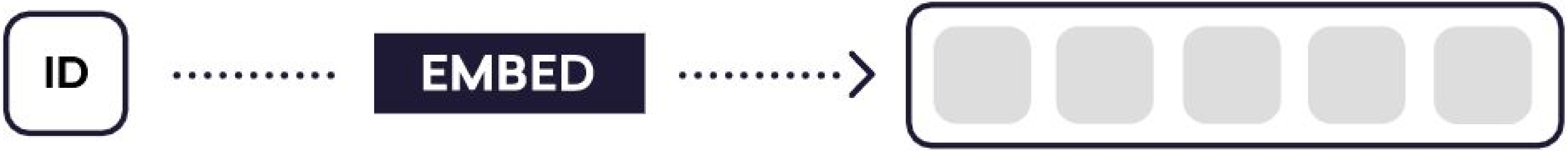


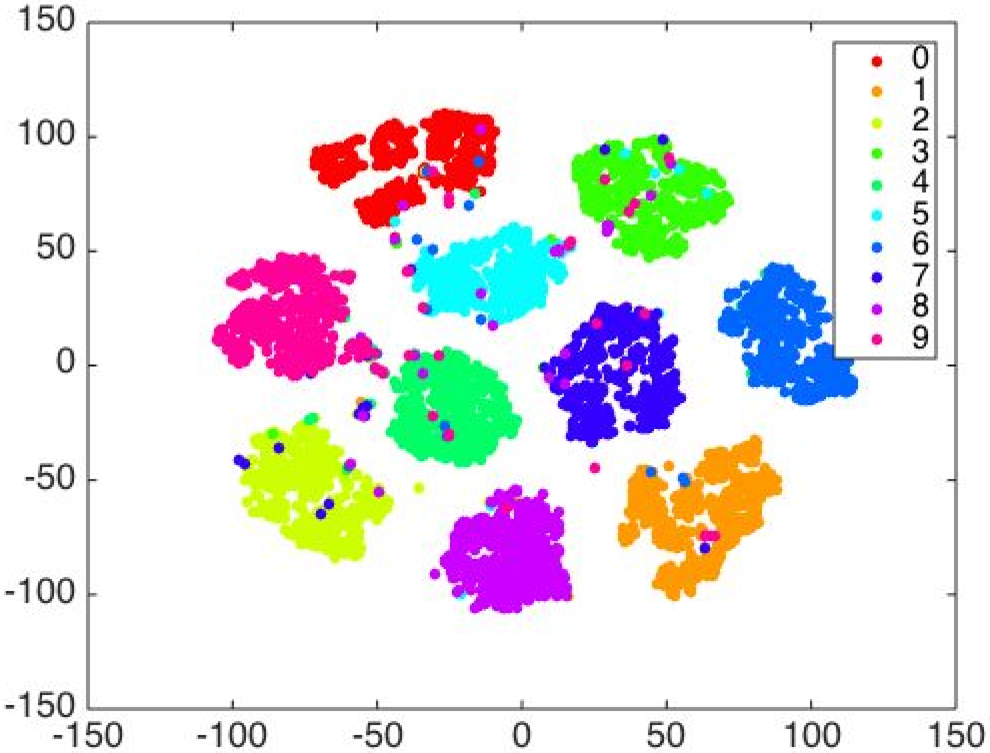
Image embedding



<whatever> embedding

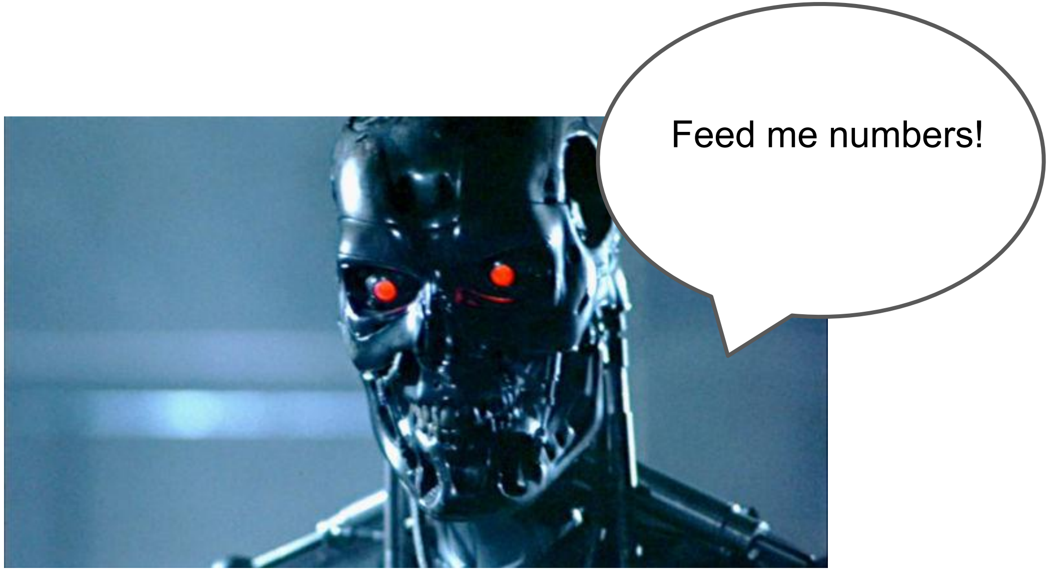


Embeddings for data visualization



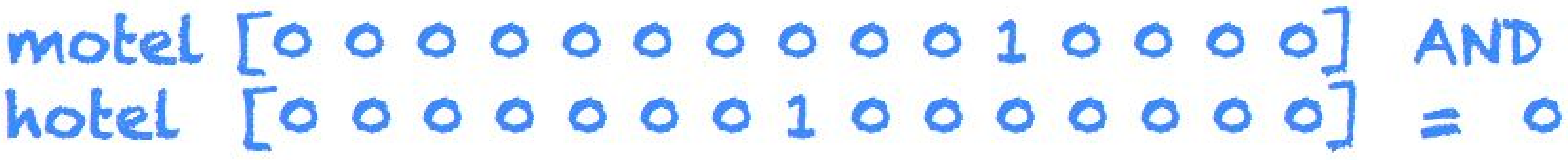
Motivation

- For some objects we don't have a natural representation that is amenable and feasible for ML algorithms



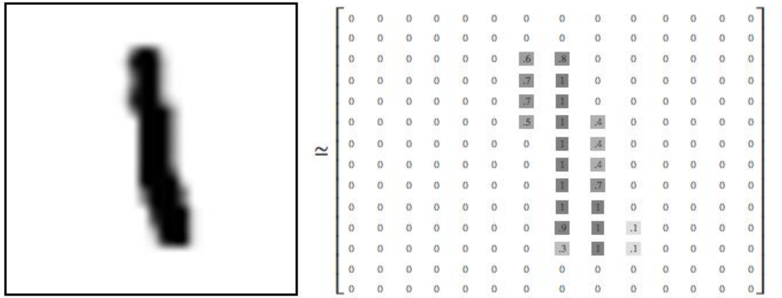
Motivation

- For some objects we don’t have a natural representation that is amenable and feasible for ML algorithms:
  - Text (words, sentences, paragraphs…)
  - Users, brands, products, anything “id” based
  - Images, movies...
- Dictionaries and one-hot encoding fail to encode meaningful information



Motivation

- For some objects we don’t have a natural representation that is amenable and feasible for ML algorithms:
    - Text (words, sentences, paragraphs…)
    - Users, brands, products, anything “id” based
    - Images, movies...
- Dictionaries and one-hot encoding fail to encode meaningful information
- Original representation is very high-dimensional



Motivation

- High-dimensional data is difficult to plot and difficult to understand
- Manifold hypothesis

Word embeddings

- Natural candidates for embeddings
  - No dense and canonical representation
  - Many non-trivial relationships (synonyms, antonyms, gender, similarity, …)

Word embeddings

- Natural candidates for embeddings
  - No dense and canonical representation
  - Many non-trivial relationships (synonyms, antonyms, gender, similarity, …)
- We would like to create a function W, mapping words into vectors (say 100-1000 dimensional)

Word embeddings

- Natural candidates for embeddings
  - No dense and canonical representation
  - Many non-trivial relationships (synonyms, antonyms, gender, similarity, …)
- We would like to create a function W, mapping words into vectors (say 100-1000 dimensional)
- W can be represented as a matrix, where each row represents one word

Word embeddings

- Natural candidates for embeddings
  - No dense and canonical representation
  - Many non-trivial relationships (synonyms, antonyms, gender, similarity, …)
- We would like to create a function W, mapping words into vectors (say 100-1000 dimensional)
- W can be represented as a matrix, where each row represents one word
- How to construct W?
  - Start with random W
  - Optimize W to perform some task

Word embeddings

- Natural candidates for embeddings
  - No dense and canonical representation
  - Many non-trivial relationships (synonyms, antonyms, gender, similarity, …)
- We would like to create a function W, mapping words into vectors (say 100-1000 dimensional)
- W can be represented as a matrix, where each row represents one word
- How to construct W?
  - Start with random W
  - Optimize W to perform some task
- Let’s start with a toy example!

Word embeddings

Possible task: classify whether a given 5-gram is valid.

Word embeddings

Possible task: classify whether a given 5-gram is valid.

"cat sat on the mat"   "cat sat song the mat"

Word embeddings

Possible task: classify whether a given 5-gram is valid.

“cat sat on the mat” “cat sat song the mat”

- We can use a huge corpus to get the valid samples (e.g. Wikipedia)

Word embeddings

Possible task: classify whether a given 5-gram is valid.

"cat sat on the mat" "cat sat song the mat"

- We can use a huge corpus to get the valid samples (e.g. Wikipedia)
- How to get negative examples?

Word embeddings

Possible task: classify whether a given 5-gram is valid.

"cat sat on the mat" "cat sat song the mat"

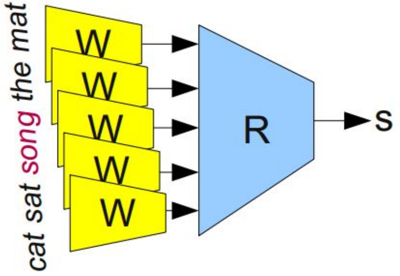
- We can use a huge corpus to get the valid samples (e.g. Wikipedia)
- Negative examples can be created by switching a random word

Word embeddings

Possible task: classify whether a given 5-gram is valid.

"cat sat on the mat" "cat sat song the mat"

- We can use a huge corpus to get the valid samples (e.g. Wikipedia)
- Negative examples can be created by switching a random word
- We plug W into a neural net so that it has to find the right values for it

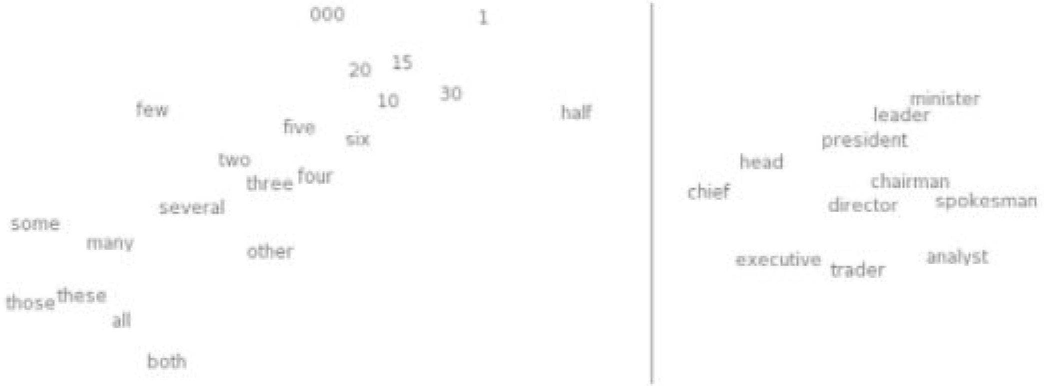


$$R(W(\text{``cat''}),\ W(\text{``sat''}),\ W(\text{``on''}),\ W(\text{``the''}),\ W(\text{``mat''})) = 1$$

$$R(W(\text{``cat''}),\ W(\text{``sat''}),\ W(\text{``song''}),\ W(\text{``the''}),\ W(\text{``mat''})) = 0$$

Word embeddings

- We don’t care about the task at all… We just care about learning W.
- We can visualize the embeddings with t-SNE (wait for it…)



Word embeddings

- Similar words ended up nearby!

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
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

Word embeddings

- Having similar words nearby is a useful thing to do for W (it allows it to perform better)

Word embeddings

- Having similar words nearby is a useful thing to do for W (it allows it to perform better)
- Note that two words can be similar in many ways:
  - Blue -> Red
  - Cat -> Dog
  - Jack -> John
  - France -> Austria

Word embeddings

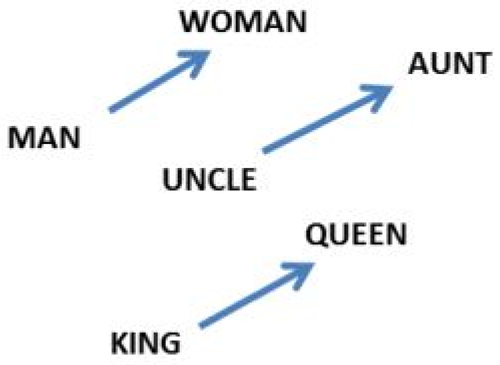
- Having similar words nearby is a useful thing to do for W (it allows it to perform better)
- Note that two words can be similar in many ways:
  - Blue -> Red
  - Cat -> Dog
  - Jack -> John
  - France -> Austria
- Seeing two sentences that differ just by one (similar) word encourages this effect

Word embeddings

(Probably) the most remarkable property is encoding analogies as differences

$$W(\text{``woman''}) - W(\text{``man''}) \simeq W(\text{``aunt''}) - W(\text{``uncle''})$$

$$W(\text{``woman''}) - W(\text{``man''}) \simeq W(\text{``queen''}) - W(\text{``king''})$$



Word embeddings

...and this goes beyond trivial relationships

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

Word embeddings

- We can use **pretrained** embeddings for a number of NLP tasks:
  - Entity recognition
  - Translation
  - Sentiment analysis
  - Part-of-speech tagging
  - Parsing
  - …
- Those embeddings can be further optimized for the selected task

Word embeddings

- All of those properties are **side effects**
- “a word is characterized by the company it keeps” - John Rupert Firth
- Distributional Hypothesis: words that appear in the same contexts share semantic meaning

Word embeddings

- All of those properties are **side effects**
- “a word is characterized by the company it keeps” - John Rupert Firth
- Distributional Hypothesis: words that appear in the same contexts share semantic meaning
- There are many ways to compute the embeddings:
  - Count-based methods (e.g. LSA)
  - Predictive methods (e.g. Word2Vec)
  - Combined (e.g. GloVe)

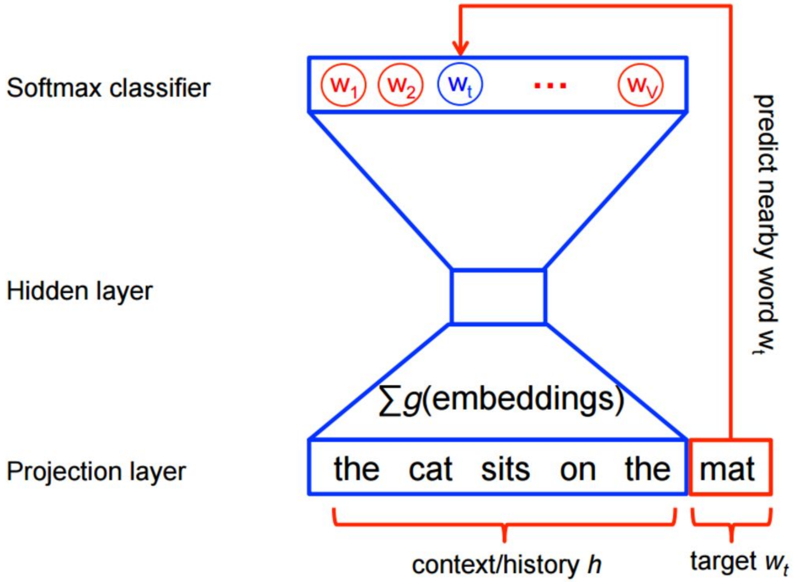
Word2Vec

- Comes in one of two flavours:
  - Continuous Bag-of-Words - predict the word based on the context words
  - Skip-Gram - predict the context words based on the word

Word2Vec

- Comes in one of two flavours:
  - Continuous Bag-of-Words - predict the word based on the context words
  - Skip-Gram - predict the context words based on the word
- Example: “the quick brown fox jumped over the lazy dog”
  - CBOW: ([the, brown], quick), ([quick, fox], brown), ([brown, jumped], fox), …
  - Skip-Gram: (quick, the), (quick, brown), (brown, quick), (brown, fox), ...

Word2Vec



$$P(w_t|h) = \text{softmax}(\text{score}(w_t, h))$$
$$= \frac{\exp\{\text{score}(w_t, h)\}}{\sum_{\text{Word w' in Vocab}} \exp\{\text{score}(w', h)\}}$$

Word2Vec

- Having the probability distribution, we can maximize the log-likelihood
- Problem: the denominator can be infeasible - we'd have to iterate over the whole Vocabulary

$$
\begin{aligned}
P(w_t|h) &= \text{softmax}(\text{score}(w_t, h)) \\
&= \frac{\exp\{\text{score}(w_t, h)\}}{\sum_{\text{Word w' in Vocab}} \exp\{\text{score}(w', h)\}}
\end{aligned}
$$

Word2Vec

- Having the probability distribution, we can maximize the log-likelihood
- Problem: the denominator can be infeasible - we'd have to iterate over the whole Vocabulary
- We can overcome this by using a binary loss instead - distinguishing between real words and "imaginary" noise words

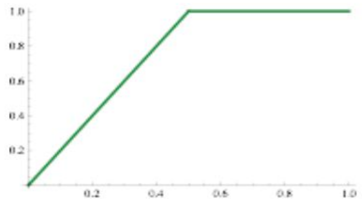
$$J(w) = \log \sigma(score(w_t, h)) + \sum_{\tilde{w} \sim \mathbf{P}_{noise}} \log \sigma(-score(\tilde{w}, h))$$

GloVe

- Combining the two worlds:
  - We learn the embeddings that correspond to co-occurrence counts

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^{W} f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$

$f \sim$



Nearest neighbours

Nearest words to frog:

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

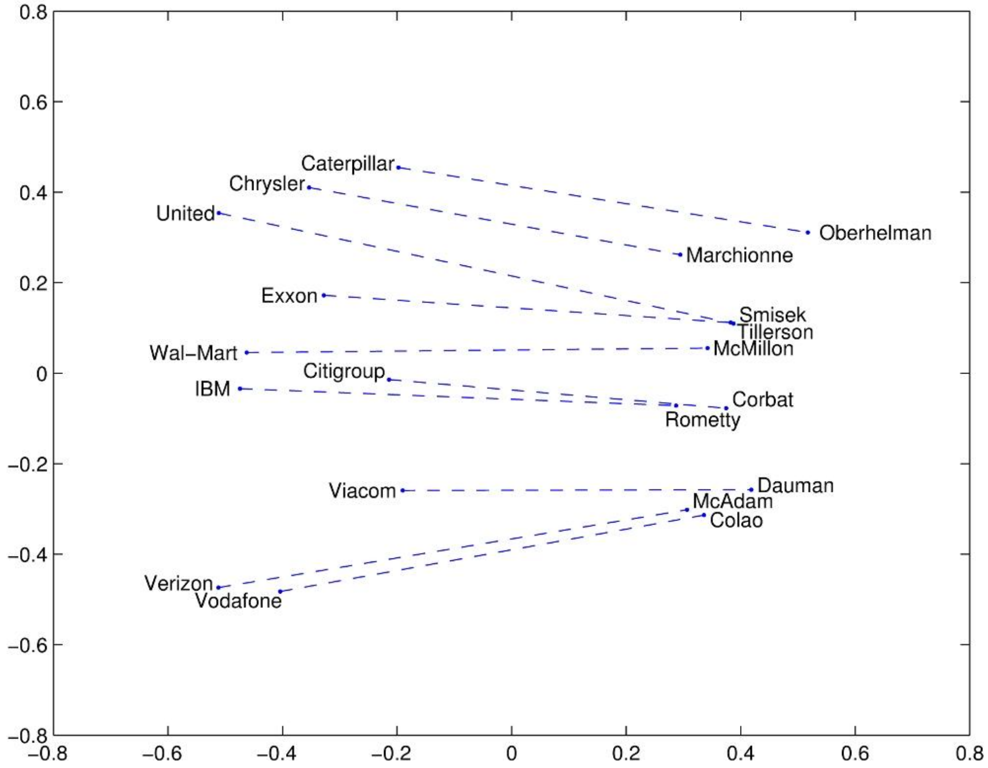
3. litoria

4. leptodactylidae

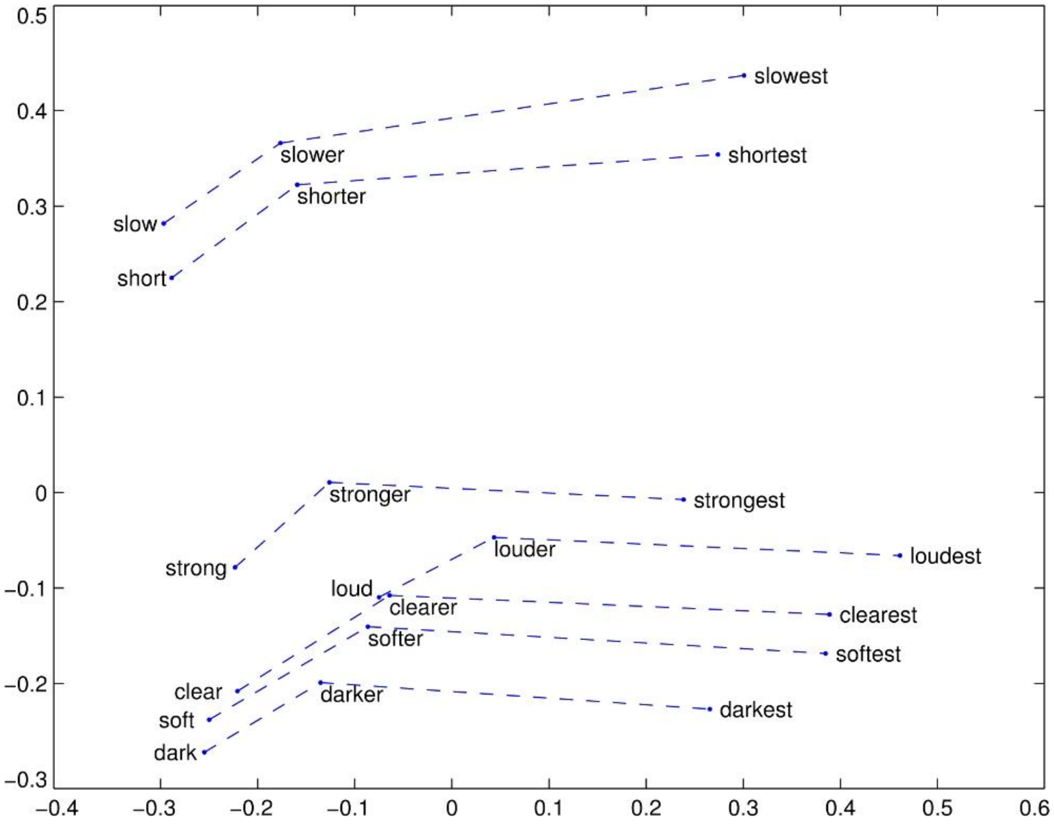
5. rana

7. eleutherodactylus

Company - CEO


Caterpillar
Chrysler
United
Exxon
Wal–Mart
Citigroup
IBM
Viacom
Verizon
Vodafone
Oberhelman
Marchionne
Smisek
Tillerson
McMillon
Corbat
Rometty
Dauman
McAdam
Colao
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8

Superlatives

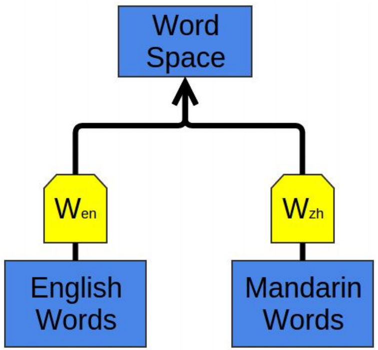


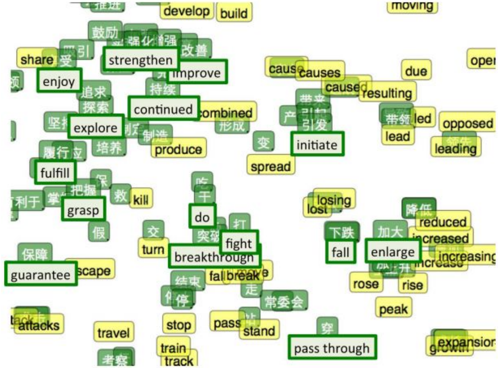
Shared representations

- Sometimes we would like to embed data from two or more distinct representations, e.g.:
  - Shared representation for two languages
  - Shared representation for images and words

Shared representations

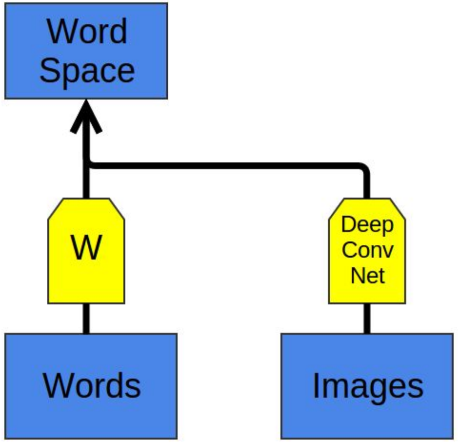
- Sometimes we would like to embed data from two or more distinct representations, e.g.:
  - Shared representation for two languages
  - Shared representation for images and words
- To achieve this we can force the embedding to map two representations that we know are of the same object to nearby points

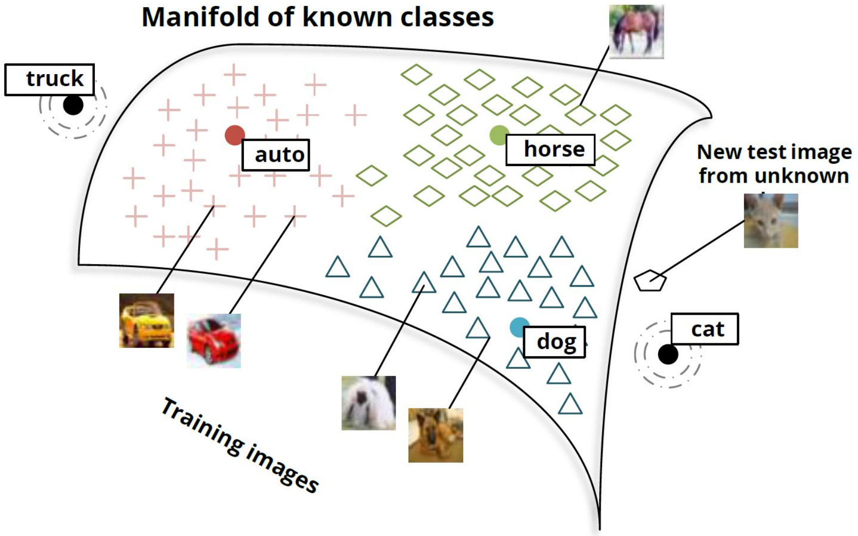




Shared representations

-





Embedding for classification/matching

- Are those two pictures of the same individual?
- Are those two descriptions of the same object?
- Finding relevant/similar objects
- One-shot or zero-shot learning

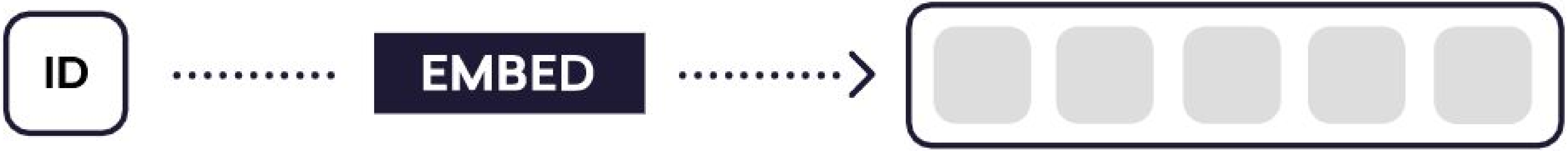
Embedding for classification/matching

- Are those two pictures of the same individual?
- Are those two descriptions of the same object?
- Finding relevant/similar objects
- One-shot or zero-shot learning

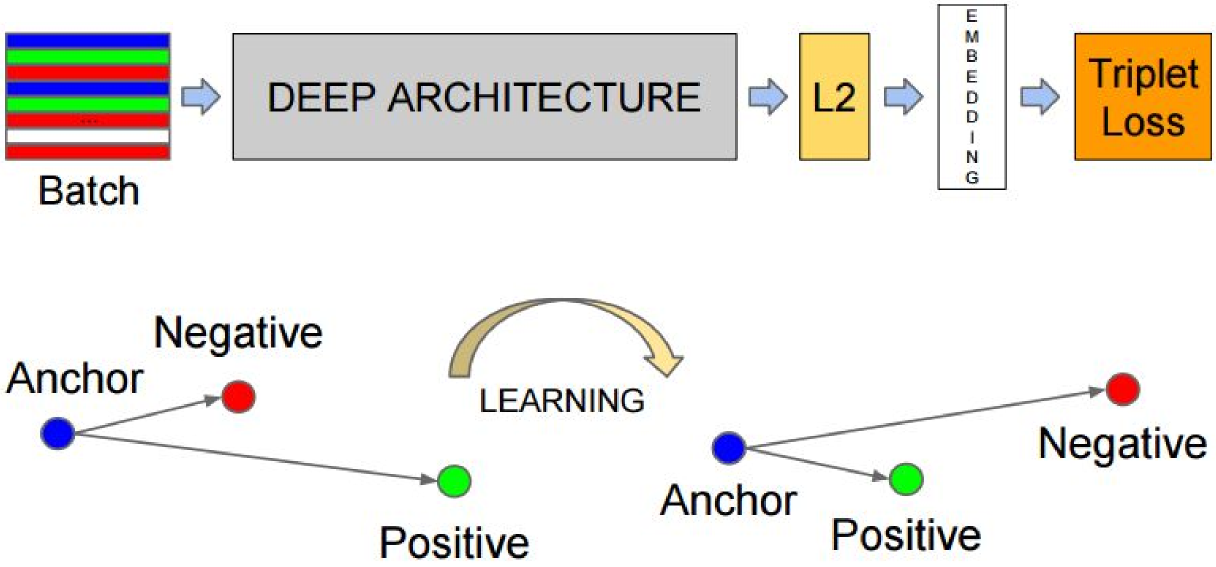
Why not ordinary classification?

- Possibly too many classes
- Not enough training samples per class
- Frequent updates (adding new classes/objects)

Embedding for classification/matching



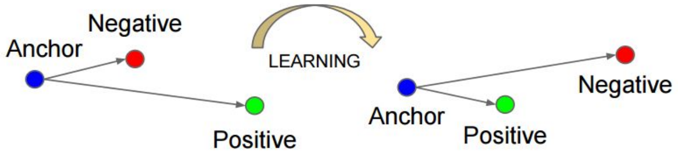
Triplet loss



Triplet loss

Positive sample should be closer than the negative one:

$$D(f(p_i), f(p_i^+)) < D(f(p_i), f(p_i^-))$$



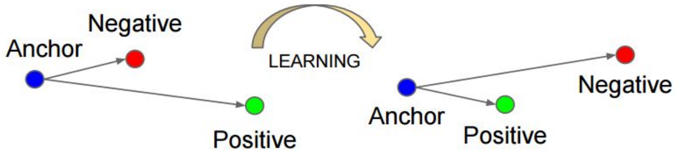
Triplet loss

Positive sample should be closer than the negative one:

$$D(f(p_i), f(p_i^+)) < D(f(p_i), f(p_i^-))$$

Based on this we can create a loss function (with a regularizer g)

$$l(p_i, p_i^+, p_i^-) =$$
$$\max\{0, g + D(f(p_i), f(p_i^+)) - D(f(p_i), f(p_i^-))\}$$



Embedding for dimensionality reduction

- Commonly used:
  - As a part of feature engineering pipeline (e.g. to battle overfitting)
  - To map the data to 2D or 3D and visualize it
  - PCA, Autoencoders, t-SNE

Embedding for dimensionality reduction

- Commonly used:
  - As a part of feature engineering pipeline (e.g. to battle overfitting)
  - To map the data to 2D or 3D and visualize it
  - PCA, Autoencoders, t-SNE
- A naive idea for embedding:
  - Preserve the distances between points (as much as possible)
  - One can actually to do this with gradient descent

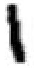
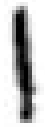
$$C = \sum_{i \neq j} (d^*_{i,j} - d_{i,j})^2$$

Embedding for dimensionality reduction

- Commonly used:
  - As a part of feature engineering pipeline (e.g. to battle overfitting)
  - To map the data to 2D or 3D and visualize it
  - PCA, Autoencoders, t-SNE
- A naive idea for embedding:
  - Preserve the distances between points (as much as possible)
  - One can actually to do this with gradient descent

$$C = \sum_{i \neq j} (d^*_{i,j} - d_{i,j})^2$$

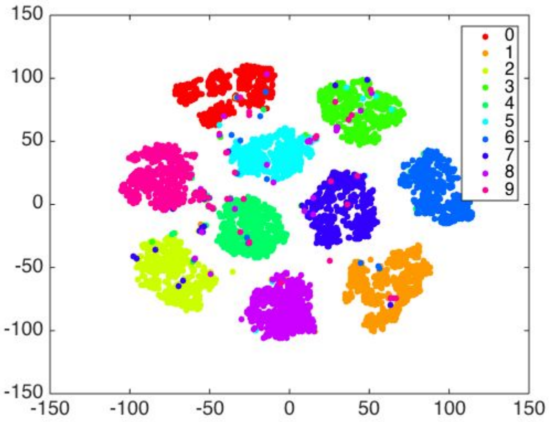
  - Distances in the original space may misbehave (e.g. images)

$d($  $,$  $) = 4.53$

$d($  $,$  $) = 12.0$

t-SNE

- State-of-the-art dimensionality reduction
- Very well suited for visualization of high-dimensional data
- Works with large datasets
- Can be used for various types of data (numerical, images, words)



t-SNE

- Idea - preserve the neighbourhood and similarities (in terms of points):

t-SNE

- Idea - preserve the neighbourhood and similarities (in terms of points):
  - Define a probability distribution of similarities in **the original space** - how much does this guy look like my nearest neighbour

t-SNE

- Idea - preserve the neighbourhood and similarities (in terms of points):
  - Define a probability distribution of similarities in **the original space** - how much does this guy look like my nearest neighbour
  - Define a probability distribution of similarities **in the embedded space**

t-SNE

- Idea - preserve the neighbourhood and similarities (in terms of points):
  - Define a probability distribution of similarities in **the original space** - how much does this guy look like my nearest neighbour
  - Define a probability distribution of similarities **in the embedded space**
  - Make sure that those two distributions are close to each other - use gradient descent to optimize embeddings

t-SNE

- Idea - preserve the neighbourhood and similarities (in terms of points):
  - Define a probability distribution of similarities in **the original space** - how much does this guy look like my nearest neighbour

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)}$$

  - Define a probability distribution of similarities **in the embedded space**
  - Make sure that those two distributions are close to each other - use gradient descent to optimize embeddings

t-SNE

- Idea - preserve the neighbourhood and similarities (in terms of points):
  - Define a probability distribution of similarities in **the original space** - how much does this guy look like my nearest neighbour

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)} \qquad p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

  - Define a probability distribution of similarities **in the embedded space**
  - Make sure that those two distributions are close to each other - use gradient descent to optimize embeddings

t-SNE

- Idea - preserve the neighbourhood and similarities (in terms of points):
  - Define a probability distribution of similarities in **the original space** - how much does this guy look like my nearest neighbour

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)} \qquad p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

  - Define a probability distribution of similarities **in the embedded space**

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq m} (1 + \|\mathbf{y}_k - \mathbf{y}_m\|^2)^{-1}}$$

  - Make sure that those two distributions are close to each other - use gradient descent to optimize embeddings

t-SNE

- Idea - preserve the neighbourhood and similarities (in terms of points):
  - Define a probability distribution of similarities in **the original space** - how much does this guy look like my nearest neighbour

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)} \qquad p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

  - Define a probability distribution of similarities **in the embedded space**

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq m} (1 + \|\mathbf{y}_k - \mathbf{y}_m\|^2)^{-1}}$$

  - Make sure that those two distributions are close to each other - use gradient descent to optimize embeddings

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Where to read more about it?

- Word embeddings:
  - http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/
  - http://p.migdal.pl/2017/01/06/king-man-woman-queen-why.html
  - http://cs224d.stanford.edu/syllabus.html (especially lectures 2 and 3)
- Visualizing high-dimensional data:
  - http://colah.github.io/posts/2014-10-Visualizing-MNIST/
  - http://colah.github.io/posts/2015-01-Visualizing-Representations/
- Triplet-loss:
  - https://arxiv.org/pdf/1503.03832.pdf (FaceNet)
  - https://users.eecs.northwestern.edu/~jwa368/pdfs/deep_ranking.pdf (Ranking with triplet-loss)
- t-SNE:
  - https://lvdmaaten.github.io/tsne/
  - http://distill.pub/2016/misread-tsne/