DATA SCIENCE FOR DIGITAL HUMANITIES 1

TEXT ANALYSIS: LEXICAL SEMANTICS

PROF. DR. GORAN GLAVAŠ

Lexical semantics

Meaning compositionality

- Most commonly in natural language processing, we consider **words** to be atomic units of meaning
    - Most of the meaning is associated with content words
- **Compositional semantics:** inducing the meaning of sentences and larger units of text (from the meaning of words)
- **Lexical semantics**: modeling/capturing the meaning of words
    - But how do we encode the meaning of words?

Lexical semantics

- **Lexical semantics**: modeling/capturing the meaning of words
    - But how do we encode the meaning of words?
    - Through relations with other words

- **Lexico-semantic resources**
    - E.g., WordNet, BabelNet, ConceptNet
    - Define semantic relations between words
        - *Synonymy*
        - *Antonymy*
        - *Hyponymy-hypernymy* („*is-a*”, „*type of*” relation)
        - *Meronymy („part of” relation)*
        - *...*

Lexico-semantic resources

WordNet hierarchy

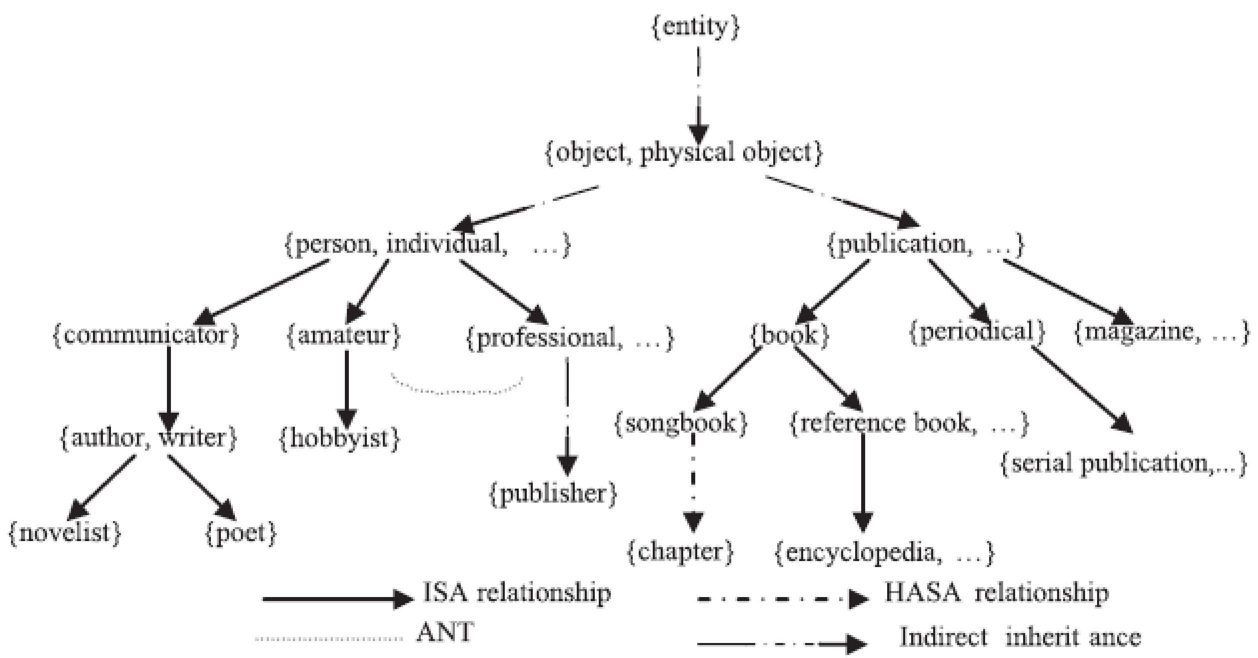


Image from: https://stackoverflow.com/questions/49355976/obtain-path-between-concepts-in-wordnet

Lexical semantics

- **Lexical semantics**: modeling/capturing the meaning of words
    - But how do we encode the meaning of words?
    - Through relations with other words

- **Lexico-semantic resources**
    - Manually curated
    - Limited coverage
    - Exist only for a handful of major languages
    - Hard to find a general-purpose meaningful measure of semantic similarity on these trees / graphs

Lexical semantics

- **Lexical semantics**: modeling/capturing the meaning of words
  - But how do we encode the meaning of words?
  - Through relations with other words

- **Distributional semantics**
  - co-occurrences of words in large corpora
  - Distributional hypothesis: „you'll know a word by the company it keeps" (Harris, 1954)
  - Assumption: the contexts in which the word appears, define its meaning

Distribution semantics: example

What is „ong choi”?

Suppose you see these sentences:

- **Ong choi** is delicious sauteed with garlic.
- **Ong choi** is superb over rice
- **Ong choi** leaves with salty sauces

– And you've also seen these:

- ... **spinach** sauteed with garlic over rice
- **Chard** stems and leaves are delicious
- **Collard** greens and other salty leafy greens

Distribution semantics: example

What is **„ong choi"**?

Word representations

- An **embedding** of a word is nothing but a numeric vector that aims to capture some properties (typically meaning) of the word

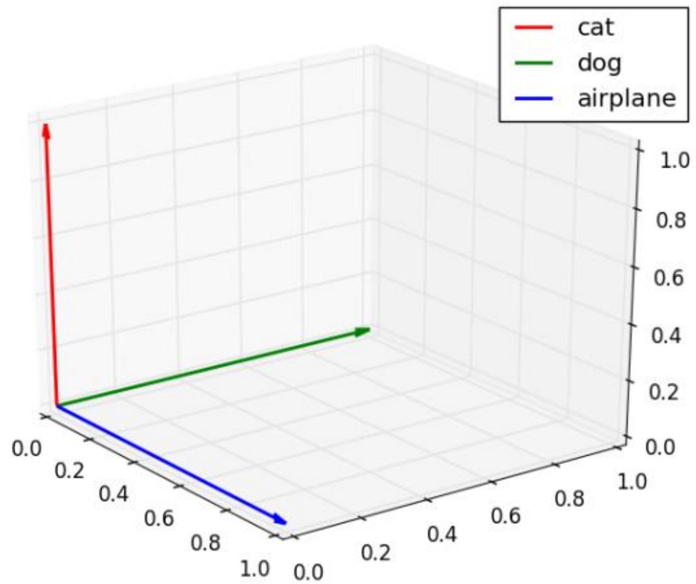
- Word can be represented with sparse or dense vectors
- Sparse vectors: one hot encoding
- Dense vectors: all rely on the distributional hypothesis
    - Co-occurrence vectors
    - Latent semantic vectors (obtained with Latent Semantic Analysis)
    - Topical distribution vectors (obtained using Latent Dirichlet Allocation)
    - **Word embeddings** (obtained using „neural” algorithms like SkipGram or CBOW)

Word representations

Sparse representation

- Each word is represented by a one-hot vector, i.e., it is given a unique symbolic ID
- The dimension of the symbolic representation for each word is equal to the size of the vocabulary V (number of words)
- All but one dimension are equal to zero, and one is set to one

$$v_{word} = (\ldots, 0, 1, 0, \ldots)$$

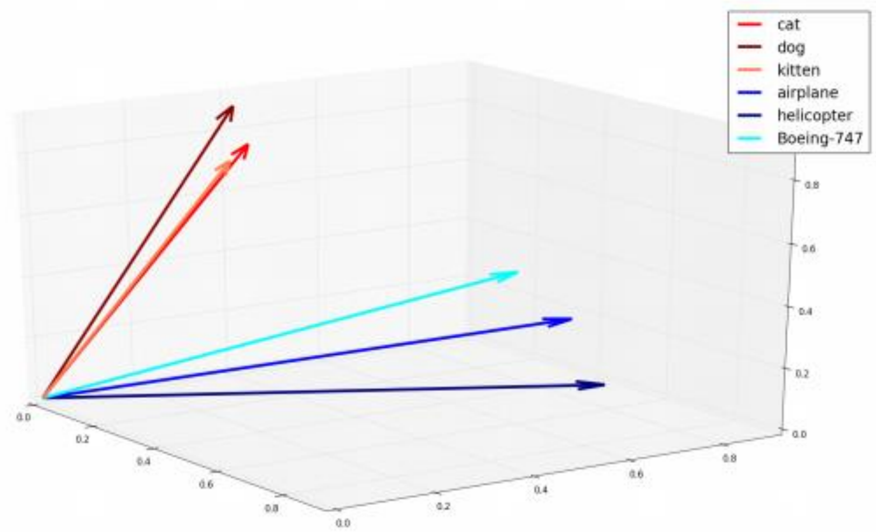


Word representations

Dense representations

- Each word is represented by a dense vector, a point in a vector space
- The dimension of the semantic representation d is usually much smaller than the size of the vocabulary (d << V)
- All dimensions contain real-valued numbers (possibly normalized between −1 and 1)

$$v_{word} = (..., 0.3, -0.5, 0.1, ...)$$



Word representations

Shortcomings of sparse word representations

- There is no notion of similarity between words
- All words are equidistant in this vector space
- V = (cat, dog, airplane)

  $v_{cat} = (0, 0, 1)$

  $v_{dog} = (0, 1, 0)$

  $v_{airplane} = (1, 0, 0)$

  sim(cat, airplane) = sim(dog, cat) = sim(dog, airplane)

- The size of the vocabulary matrix D
- $V \cdot V$, as we have a V-dimensional vector for each out of V words
- Usually we have to remove some words from the vocabulary due to memory footprint

Word representations

- Distributional hypothesis: „you'll know a word by the company it keeps" (Harris, 1954)
- Dense representations are derived from word co-occurrences in a large corpus of text

. . . the quick brown | fox | jumps over the . . .

- Assumption: the contexts in which the word appears, define its meaning
- This allows to create a (still rather sparse) V x V dimension matrix of co-occurrences between words
- Word vectors from the co-occurrence matrix can now be compared (similar words will appear in similar contexts, hence have similar vectors)

Word representations

Exploiting co-occurrences for deriving dense word representations

1. **Count-based / dimensionality reduction strategies**
    - Idea: don't need all the dimensions representing a word, just the most important ones
    - Dense vectors obtained through factorization of the co-occurrence matrix
        - Latent semantic analysis (LSA), based on SVD decomposition

Word representations

Exploiting co-occurrences for deriving dense word representations

1. **Prediction-based models**
   - Start with dense random vectors for all word in the vocabulary
   - Go through the corpus and try to predict the center word from the context (or the context from the center word)
   - Update dense word vectors based on the prediction error
   - Word2Vec models (Mikolov et al., 2013): **Continuous Bag-of-Words (CBOW)** and **Skip-Gram (SG)**

Count-based distributional methods

- Start by counting (co-)occurrences on a large corpus
- Occurrences of words in contexts
- Contexts can be:
  - A symmetric or asymmetric word window of some size
  - A sentence
  - A parag
  - ...

$$
A = \begin{array}{l} \\ \textit{president} \\ \textit{minister} \\ \textit{speech} \\ \textit{law} \\ \textit{ball} \\ \textit{score} \\ \textit{player} \\ \textit{run} \\ \textit{person} \\ \textit{piano} \\ \textit{mouse} \end{array}
\begin{array}{c}
\begin{array}{cccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\
\begin{pmatrix}
3 & 2 & 0 & 1 & 0 & 0 \\
4 & 1 & 3 & 0 & 0 & 0 \\
2 & 5 & 1 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 & 1 \\
0 & 0 & 0 & 4 & 0 & 2 \\
0 & 0 & 0 & 3 & 2 & 3 \\
0 & 0 & 1 & 1 & 4 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0
\end{pmatrix}
\end{array}
$$

Latent Semantic Analysis

- **Latent Semantic Analysis (LSA)** – a distributional lexical semantics model based on a factorization of a sparse (co)ocurrence matrix
- Namely Singular Value Decomposition (SVD)

- We decompose the sparse co-ccurrence into factor matrices
- Which we use to obtain dense vector representations of words

- Obtained dense vectors better capture meaning of words that raw sparse distributional vectors
- Comparing dense vectors of words better captures their semantic similarity than comparing their sparse distributional vectors

Latent Semantic Analysis

- Given a matrix **A** (with non-negative elements!), the Singular Value Decomposition finds orthogonal matrices **U** and **V** and a rectangular diagonal matrix **Σ** such that:

$$A = U\Sigma V^T$$

- Matrix **U** is of dimensions **M** x **M**
- M is the number of words, i.e., the vocabulary size
- Matrix **V** is of dimensions **N** x **N**
- N is the number of contexts
- Matrix **Σ** is of dimensions **M** x **N**
- U and V are orthogonal: $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}$
- Values of the diagonal matrix **Σ** are singular values of **A**

LSI – SVD Example

$$\text{terms}\left\{ U = \overbrace{\begin{bmatrix} -0.43 & 0.13 & 0.22 & -0.01 & -0.55 & -0.09 & \\ -0.53 & 0.25 & -0.28 & 0.62 & -0.09 & -0.07 & \\ -0.58 & 0.33 & 0.18 & -0.56 & 0.37 & 0.06 & \\ -0.12 & -0.05 & -0.19 & 0.28 & 0.64 & 0.26 & \\ -0.22 & -0.51 & 0.53 & 0.17 & 0.10 & -0.32 & \\ -0.26 & -0.62 & 0.08 & -0.05 & -0.03 & 0.41 & \cdots \\ -0.22 & -0.40 & -0.69 & -0.25 & -0.12 & -0.21 & \\ -0.03 & -0.06 & -0.18 & -0.11 & -0.12 & -0.07 & \\ -0.11 & -0.03 & 0.02 & 0.13 & -0.18 & 0.60 & \\ -0.10 & -0.02 & -0.12 & -0.29 & 0.01 & -0.06 & \\ -0.09 & -0.08 & 0.01 & 0.16 & 0.26 & -0.47 & \end{bmatrix}}^{\text{topics}} \right. \begin{array}{l} \mathit{president} \\ \mathit{minister} \\ \mathit{speech} \\ \mathit{law} \\ \mathit{ball} \\ \mathit{score} \\ \mathit{player} \\ \mathit{run} \\ \mathit{person} \\ \mathit{piano} \\ \mathit{mouse} \end{array}$$

- The first column („topic") seems to have weights of large magnitude for *politics* terms, and the second column for *sports* terms

Latent Semantic Analysis

- **Goal**: reduce the dimensionality of word and context vectors and obtain dense semantic vectors of words (and contexts)

- We reduce the size of the matrix $\mathbf{\Sigma}$ with singular values
  - We keep only the top K largest singular values: $\sigma_1, ..., \sigma_k$
  - We denote the reduced matrix with $\mathbf{\Sigma}_k$
  - Dense vectors for terms and contexts will be then be of dimension K

- By reducing the rank of the matrix with singular values, we are effectively retaining only the K most prominent „topics”
  - Retained topics carry the most of the „meaning”
  - The topics/dimensions we discard are assumed to be noise

Latent Semantic Analysis

- This leaves us with the best possible approximation of rank A_K of the original term-document occurrence matrix A

$$\text{Dense vectors of words}\left\{\begin{array}{c} \\ \textit{president} \\ \textit{minister} \\ \textit{speech} \\ \textit{law} \\ \textit{ball} \\ \textit{score} \\ \textit{player} \\ \textit{run} \\ \textit{person} \\ \textit{piano} \\ \textit{mouse} \end{array}\right. \begin{array}{c} U_K \\ \begin{bmatrix} -0.43 & 0.13 \\ -0.53 & 0.25 \\ -0.58 & 0.33 \\ -0.12 & -0.05 \\ -0.22 & -0.51 \\ -0.26 & -0.62 \\ -0.22 & -0.40 \\ -0.03 & -0.06 \\ -0.11 & -0.03 \\ -0.10 & -0.02 \\ -0.09 & -0.08 \end{bmatrix} \end{array} \cdot \overbrace{\begin{array}{c} \begin{array}{cccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\ \begin{bmatrix} -4.66 & -4.37 & -2.71 & -2.37 & -1.51 & -1.65 \\ 2.01 & 2.12 & 0.49 & -4.23 & -2.93 & -3.35 \end{bmatrix} \end{array}}^{\text{Dense vectors of contexts}} \Sigma_K V^T_K$$

- A_K has the same dimensions as original A ($M \times N$)
- U_K is of size $M \times K$, and $\Sigma_K V^T_K$ of size $K \times N$

Latent Semantic Analysis

- In practice, we don't compute $\mathbf{A}_K$
- $\mathbf{A}_K$ is not sparse – it's explicit computation is computationally expensive!
- We don't need to have $\mathbf{A}_K$ to compare pairs of words

- Term comparison is performed by comparing rows of $\mathbf{U}_K$
  - sim(„president", „minister") = cos([-0.43, 0.13], [-0.53, 0.25])
  - sim(„president", „player") = cos([-0.43, 0.13], [-0.22, -0.40])

- Context comparison is performed by comparing columns of $\mathbf{\Sigma}_K\mathbf{V}^T{}_K$
  - sim(c_1, c_2) = cos([-4.66, 2.01], [-4.37, 2.12])
  - sim(c_4, c_6) = cos([-2.37, -4.23], [-1.65, -3.35])

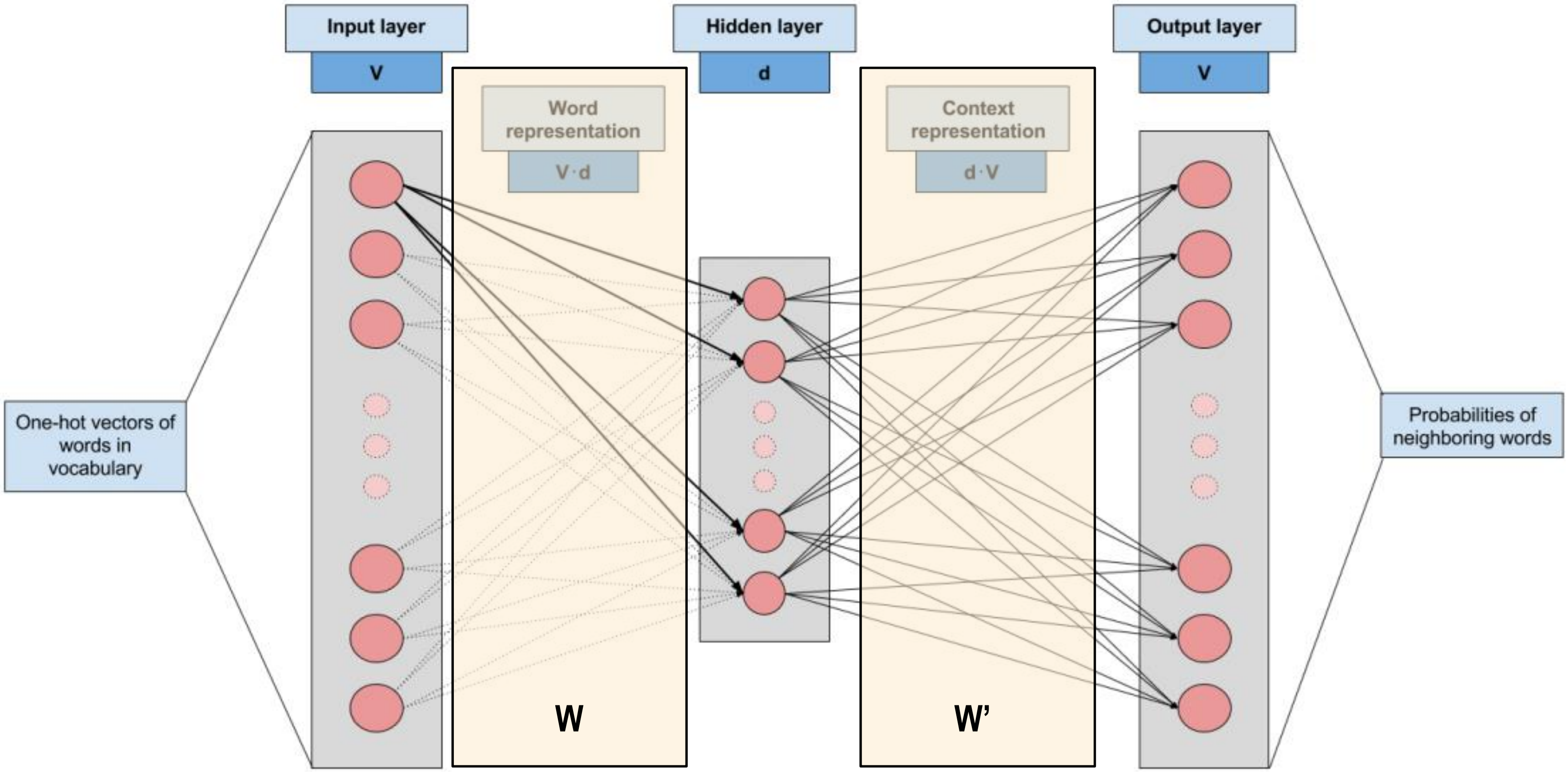
Prediction-based model: Skip-Gram

- Start by assigning two different dense random vectors to each word
- **Center vector** and **context vector** (each of size d << V)

- For a center word, predict the words will appeat in its context
  - E.g., given „fox” predict „quick”; „brown”; „jumps”; „over”

- **Algorithm**
  - Single-layer neural network (not really deep :)
  - The input **X** is the one-hot encoding representation of the **center word**
  - Two parameter matrices: **W** and **W’**
    - **W** (V x d) transforms the one-hot encoding vector of the center word into a dense vector
    - **W’** (d x V) transforms the dense vector into the sparse vector of the context

Skip-Gram (SG) model



Skip-Gram (SG) model

- Let $\mathbf{v_w}$ be the sparse vector of the **center** word w

- The dense **center vector** (dimension d) is then computed as:

$$\mathbf{c_w = v_w\, W}$$

- The predicted vector of the context is computed as:

$$\mathbf{c_p = c_w{}^T\, W'}$$

- Let $\mathbf{c_t}$ be the sparse, one hot-encoding vector of some **context** word

- We compute the prediction error by comparing the true vector of the context word $\mathbf{c_t}$ and the predicted context vector $\mathbf{c_p}$

Skip-Gram (SG) model – softmax

- The predicted context vector $\mathbf{c_p}$ is not a probability distribution over vocabulary terms, and it should be

- Thus, we apply the **softmax function**, to transform $\mathbf{c_p}$ into a probability distribution

$$\mathrm{softmax}(c_p^i) = \frac{\exp(cpi)}{\sum_j \exp(cpj)}$$

- Now, both the predicted vector $\mathbf{c}_p$ and context one-hot encoding vector $\mathbf{c}_t$ are probability distributions

- We compute how dissimilar they are and propagate the error to update the weights in **W** and **W’**

Skip-Gram

- **One matrix** (**W** of dimensions V x d) to encode the center word into low-dimensional dense vector
- **One matrix** (**W’** of dimensions d x V) to „reconstruct” the context word
- Each vocabulary word has a corresponding row in **W** and a corresponding column in **W’**
- When training finishes (i.e., we learn good values in W and W’), the **word embedding** of i-th vocabulary word is the concatenation of:
  1. The i-th row of the matrix **W**
  2. The i-th column of the matrix **W’**

Word embeddings – results

Airplane

word	cosine
plane	0.835
airplanes	0.777
aircraft	0.764
planes	0.734
jet	0.716
airliner	0.707
jetliner	0.706

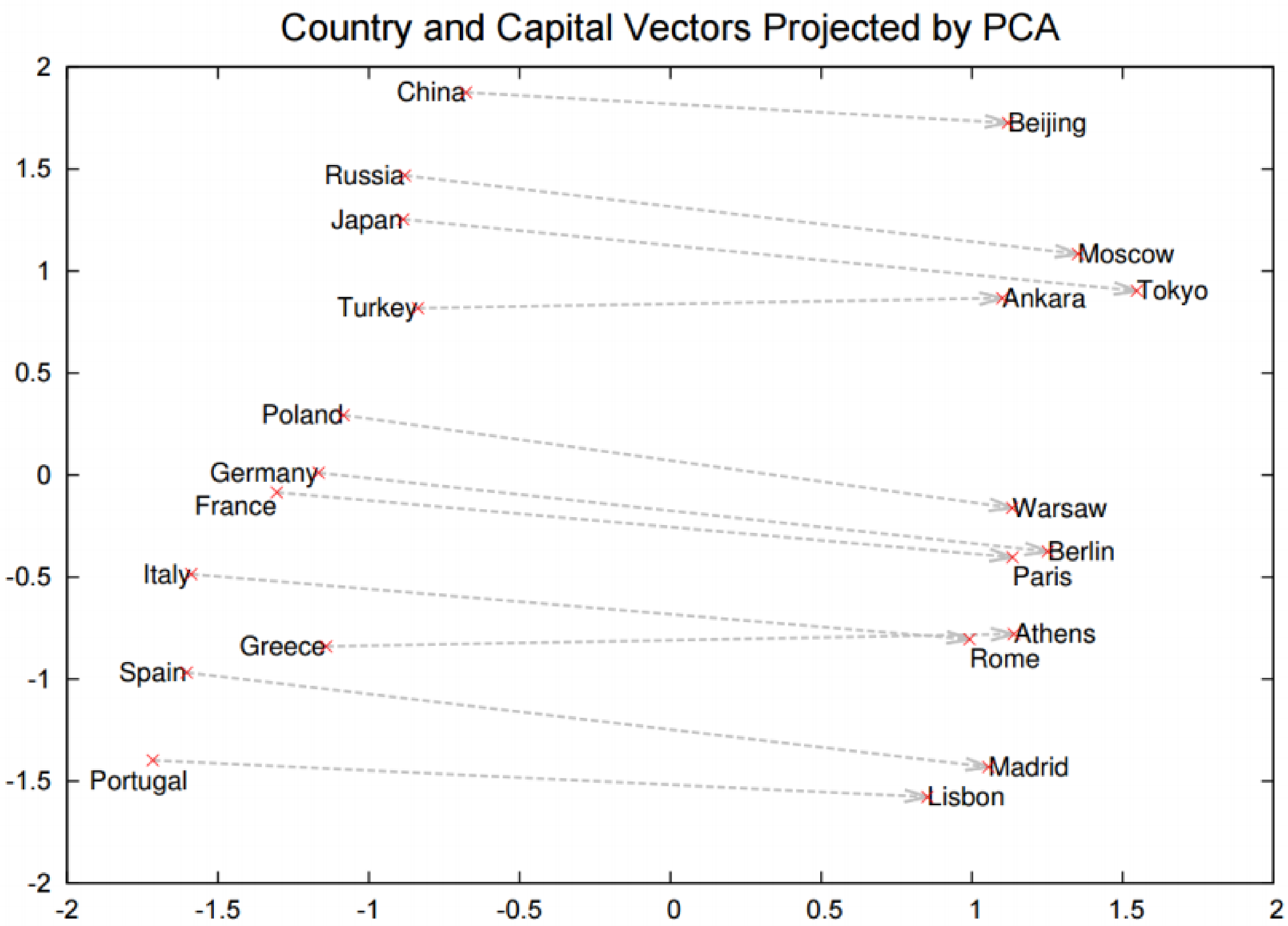
Cat

word	cosine
cats	0.810
dog	0.761
kitten	0.746
feline	0.732
puppy	0.707
pup	0.693
pet	0.689

Dog

word	cosine
dogs	0.868
puppy	0.811
pit_bull	0.780
pooch	0.763
cat	0.761
pup	0.741
canines	0.722

Word embeddings – results



Evaluating Word Representations

- **Q:** How do we measure if the obtain dense vectors representing words are good?
- We don't really know how the vectors should look like (no gold vectors!)
- **A:** We evaluate whether word similarities perceived by humans correspond to similarities computed based on the obtained vectors

- We need manually created evaluation resources:
  - Consisting of triples (w_1, w_2, *score*)
  - *Score* is the human-assigned degree of semantic similarity/relatedness between the words w_1 and w_2

Some Evaluation Resources

- **WordSim-353**
  - 353 word pairs annotated with scores of general semantic relatedness
  - http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/

- **SimLex-999**
  - 999 word pairs annotated for *semantic similarity*
  - *Car* is similar to *vehicle*, but not to *driver*
  - *https://fh295.github.io/simlex.html*

- **SimVerb-3500**
  - 3500 verb pairs judged for semantic similarity
  - Verbs are typically more difficult to model in a vector space
  - http://people.ds.cam.ac.uk/dsg40/simverb.html

Evaluation Measures

- Two sets of scores:
  1. Manually assigned scores by the annotatord
  2. Automatically obtained scores based on dense word vectord
     - Most commonly, **cosine similarity** between the vectors of the two words

- We measure a correlation measure between the two sets of scores:
  1. **Pearson correlation** – correlation between the actual scores
  2. **Spearman correlation** – correlation between rankings
     - We rank the word-pairs according to both gold-standard scores and predicted scores
     - We compute Pearson correlation between sets of ranks