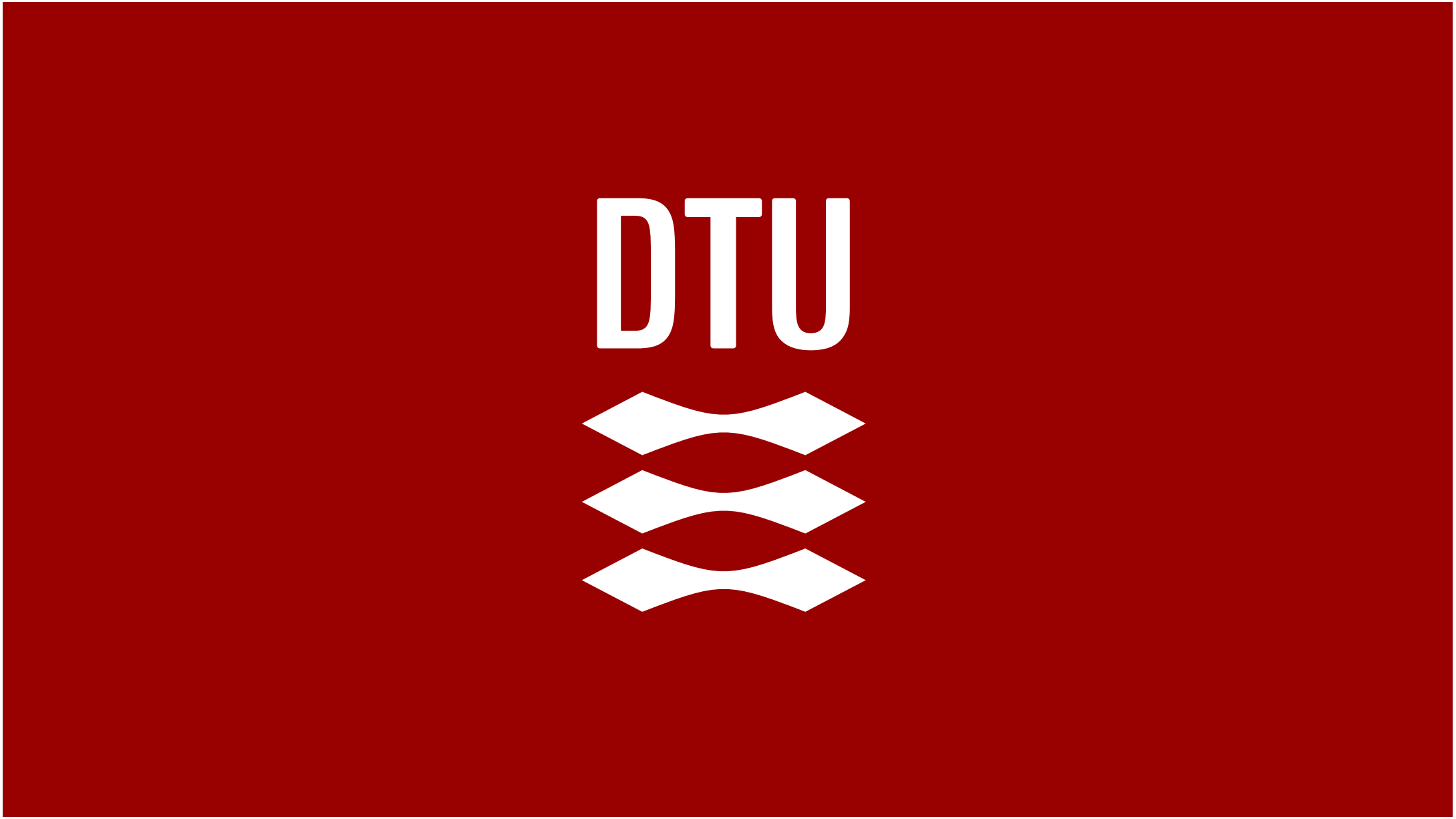

DTU

42578 Advanced Business Analytics

Text Analytics
word embeddings

Continuous Space Language Models

- Matrix Factorization
  - Latent Semantic Analysis (LSA)
  - Latent Dirichlet Allocation (LDA)

- **Neural Networks**
  - **Word embeddings** → real-valued vector representations of words

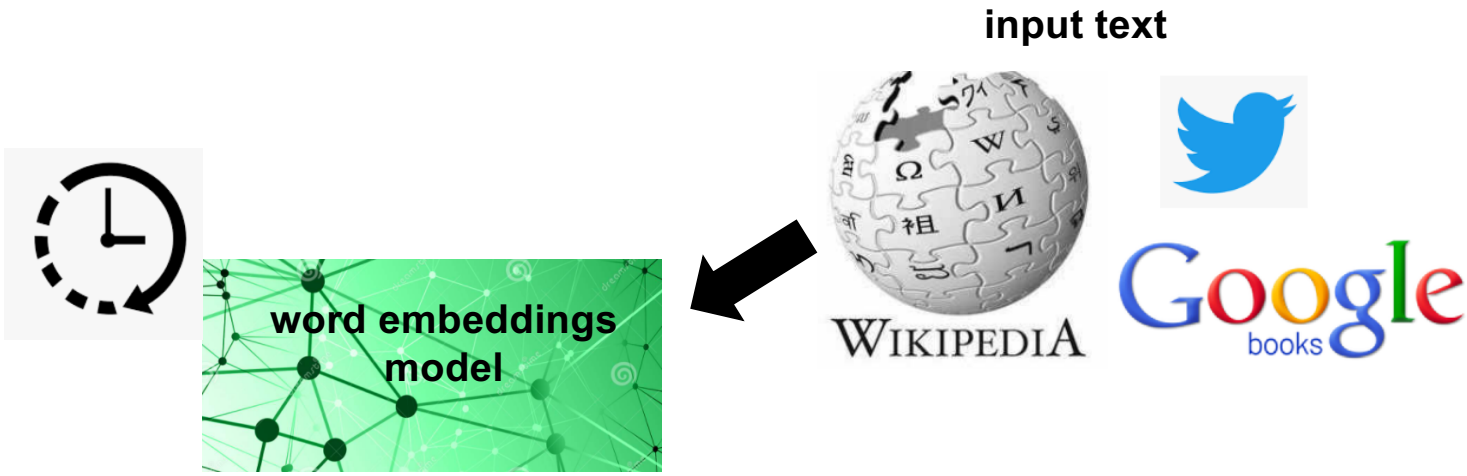
e.g. cat: [0.5 -0.2 0.7 0.9 …] → dense

D (*fixed* ~100-300 dims)

one-hot encoding → e.g. "cat": [0 0 1 0 …] → sparse

V (=vocab size ~thousands dims)

Word Embeddings – Generation



D

V

cat	→	[0.5	-0.2	0.7	0.9 ...]
dog	→	[-0.3	0.4	0.5	-0.8 ...]
house	→	[0.6	-0.1	-0.7	0.2 ...]
man	→	[-0.9	0.5	-0.6	0.7 ...]
.....					

W

VxD matrix W of embedding vectors

D: dimensionality of the embeddings

V: number of words in the input vocabulary

Each row is a vector representing a word!

Word Embeddings – Intuition

The Distributional Hypothesis:

"words that occur in the same contexts tend to have similar meanings" (Harris, 1954)

Learning distributional properties of the input words → able capture semantic associations

- *Similar words* have *similar vectors*
- All words are embedded into the *same continuous space*
- Formation of *meaningful word clusters* in the embedding space

Word Embeddings – Visualization

- Embedding vectors are of modest dimensionality (typically spanning from tens to hundreds)

- Visualization in 2-D using dimensionality reduction techniques:

Principal Component Analysis (PCA)
Multidimensional scaling (MDS)
t-Distributed Stochastic Neighbor Embedding (t-SNE)

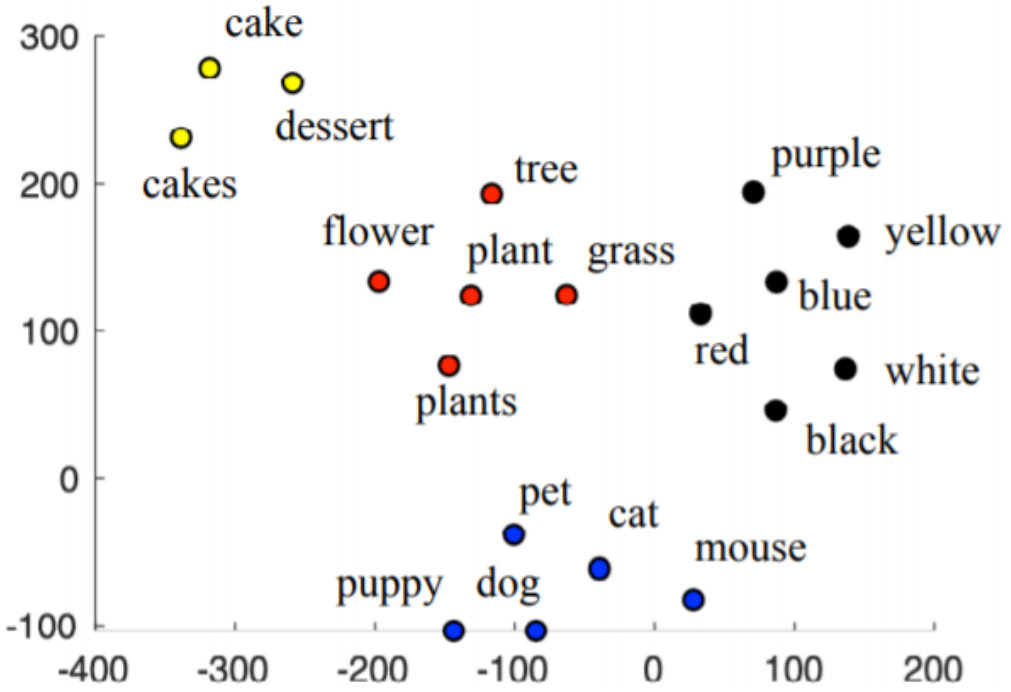


image by (Fang et al., 2019)

Semantically close words occupy neighboring positions in the embedding space.

Where do word embeddings come from?

Word embeddings toolkits available today:

- word2vec → introduced by (Mikolov et al., 2013) → popularization of word embeddings!
- GloVe → (by Standford University)
- ELMo
- BERT

} different embeddings for homographs (= words with the same spelling but different meanings)

- fastText → (by Facebook's AI Research lab) / Better for smaller training sets

……

Ready-to-use *pretrained* word embeddings:

- ***on a variety of text collections***

- ***on different languages***

word2vec models

How do they work? → What is the model learning to predict ?

1. **Continuous Bag-of-Words (CBOW) model** → *predicts one word based on context words*

   e.g. **Madrid is the** ___capital___ **of Spain.** *(Fill-in-the-blank logic)*

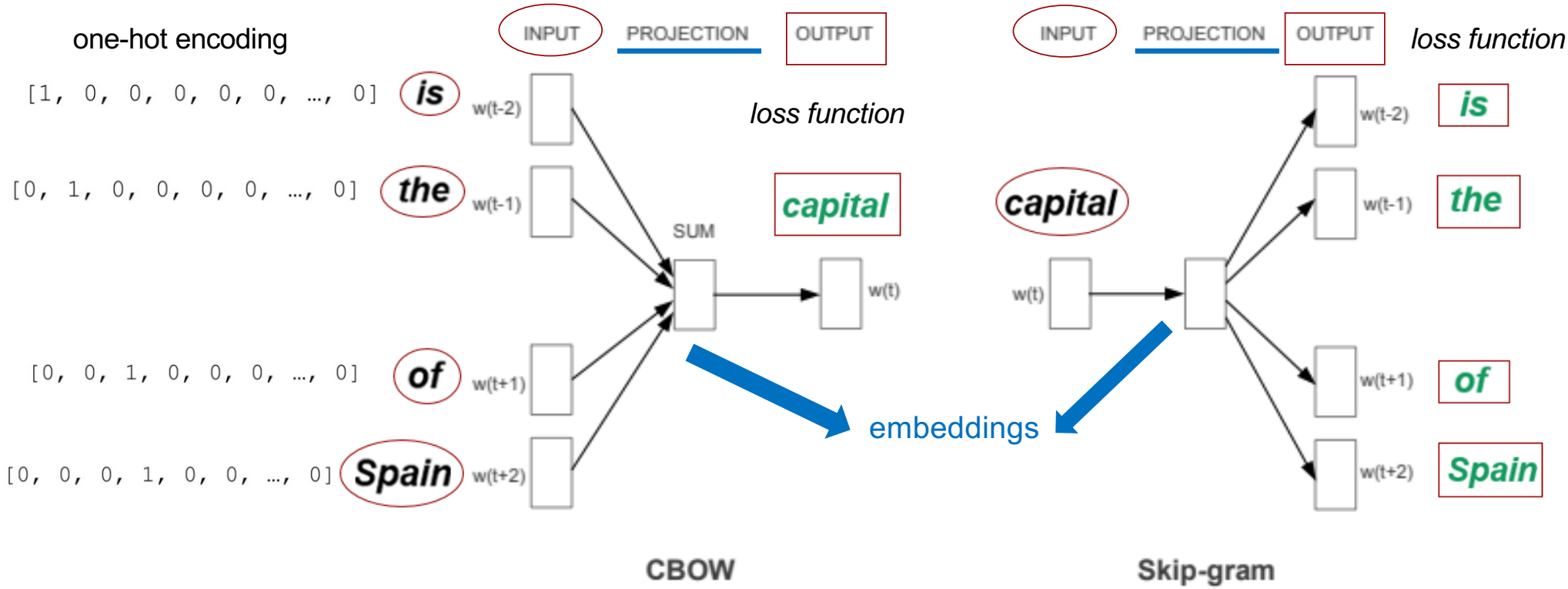
   → more intuitive
   → faster to train

2. **Skip-gram model**→ *predicts all context words given one word*

   e.g. ___Madrid___ ___is___ ___the___ **capital** ___of___ ___Spain___ ___.___

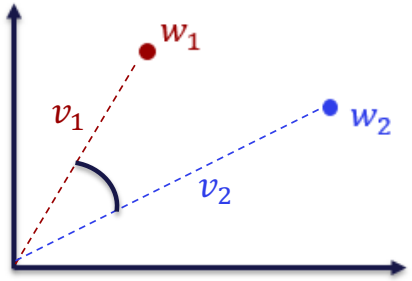
   → slower to train
   → ++ for rare words

word2vec models



Word Embeddings – Similarity between words

The semantic similarity between 2 words w_1 and w_2 can be measured through a similarity metric between their corresponding embedding vectors v_1 and v_2 in the embedding space



cosine similarity

$$similarity(w_1, w_2) = cosine\ sim(v_1, v_2) = \frac{v_1 \cdot v_2}{||v_1|| \cdot ||v_2||}$$

→ ***range*** $[-1,\ 1]$

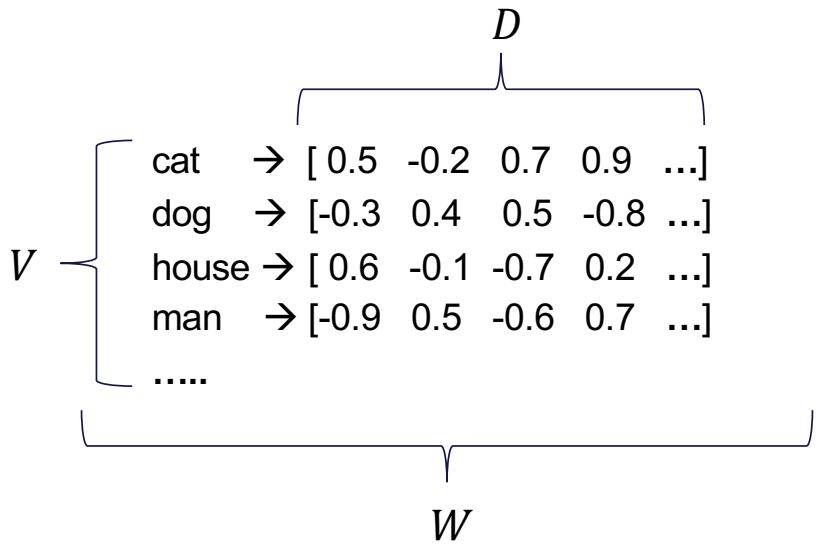
$$\boldsymbol{cosine\ distance = 1\ - cosine\ sim}$$

- For normalized vectors ($||v|| = 1$) :

$$cosine(v_1, v_2) = v_1 \cdot v_2$$

Word Embeddings – similarity search

Example: **Given** a VxD **embedding matrix** ***W*** with *normalized* vectors v



Find top-n most similar words to "cat"

- Get a vector S_{cat} of cosine similarities between v_{cat} and all the word vectors in W:

$$\boldsymbol{S_{cat}} = [\,1 \quad 0.8 \quad -0.5 \quad 0.6 \quad \ldots.] = \boldsymbol{W \cdot v_{cat}^{T}}$$

V: cat, dog, house, man

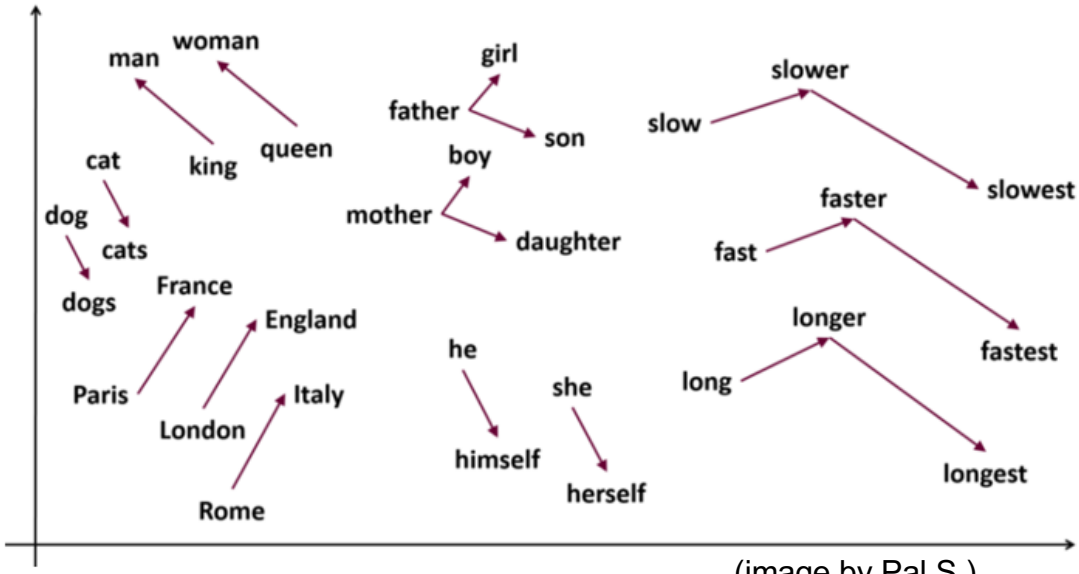
- Get the words corresponding to the ***highest n similarity values*** **in** $\boldsymbol{S_{cat}}$ (ignoring the first word→ cat)

$$\boldsymbol{sorted}(\boldsymbol{S_{cat}})[\mathbf{1} : \boldsymbol{n} + \mathbf{1}] = [\,0.8 \quad 0.75 \quad 0.72 \quad 0.7\,]$$

dog, pet, cats, rabbit

computational tools and packages → fast + efficient similarity search (even for hundreds of thousands of words)

Word Embeddings → Compositionality



(image by Pal S.)

embedding vector offset → **words relation**

cat − cats ≈ dog – dogs ≈ leg – legs ≈ ….→ (singular/plural)

France – Paris ≈ England − London ≈ …. → (country/capital)

king − man ≈ queen − woman ≈ …. → (royalty/gender)

king – queen ≈ man – woman ≈ …. → (male/female)

Answering analogy questions!

"man is to king as woman is to ..?"

king − man + woman ≈ queen

Word Analogies

$$If \quad w_1 - w_2 \approx w_4 - w_3$$
$$w_1 - w_2 + w_3 \approx w_4$$

Then

"w_1 *is to* w_2 *as* w_4 *is to* w_3"

Word Embeddings – NLP Applications

- Alone (as features) or combined with traditional NLP methods (e.g. with tf-idf weighting)
  improved performance in:
  - → document classification
  - → topic modeling
  - → sentiment analysis

- Efficient for a variety of challenging NLP tasks
  such as:
  - → automatic translations
  - → text normalization
  - → automatic sentence completion
  - → named entity recognition
  - → question answering
  - → metaphor detection

  .....

Training your own word embeddings?

Word2vec FastText GloVe

PRETRAINED MODEL

Why?

Training your own word embeddings with word2vec

Things to consider:

(1) **Size of the corpus**!!! → crucial for obtaining high-quality representations
(2) **Text Preprocessing** → cleaning, tokenization…

word2vec parameters:

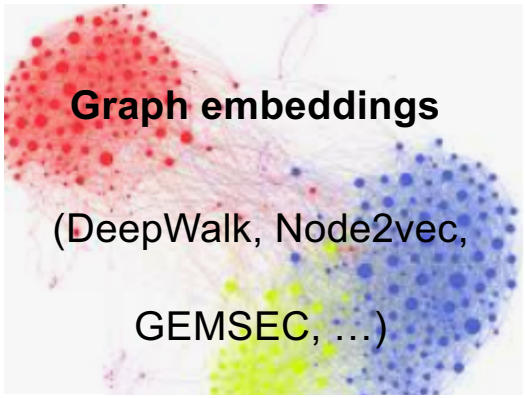
(3) **model**: skip-gram (slower, better for infrequent words) *vs* CBOW (faster)

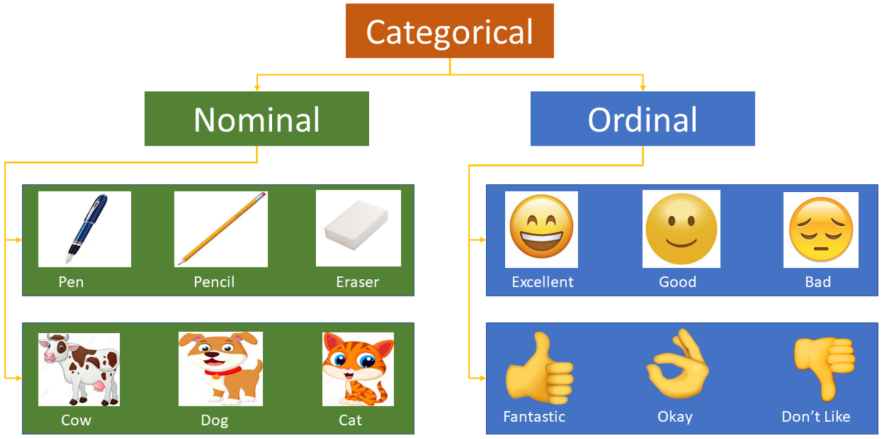
(4) embeddings **dimensionality**: typical values are in range ~100 to ~300

(5) **training algorithm**: hierarchical softmax (better for infrequent words)
vs negative sampling (better for frequent words + low vector dimensions)

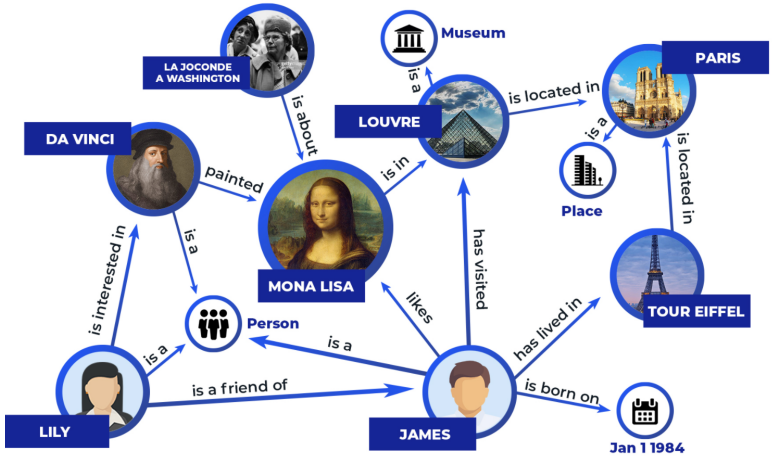
(6) **context window size:** ~5 (for skip-gram) to ~10 words (for CBOW)

Beyond word embeddings….





Embeddings for categorical variables
(cat2vec, PyTre)



Knowledge Graph embeddings

(RESCAL, TransE, …)

Image embeddings (DeViSE, …)

References

Harris, Z. S. (1954). Distributional structure. *Word*, *10* (2-3), 146-162.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Fang, L., Luo, Y., Feng, K., Zhao, K., & Hu, A. (2019, May). Knowledge-enhanced ensemble learning for word embeddings.

In *The World Wide Web Conference* (pp. 427-437).

Pal S. (2019, Jun 22). Implementing Word2Vec in Tensorflow. Analytics Vidhya. Retrieved from: https://medium.com/analytics-vidhya/implementing-word2vec-in-tensorflow-44f93cf2665f . access: 01.02.2021.