# Embeddings

Jonathon Hare

Vision, Learning and Control
University of Southampton

# Introduction

- Sparse versus dense representations; similarity; context
- Dimensionality reduction
- Word Embeddings
- Joint Embeddings

# Problem statement

- Consider training a neural network on text
  - We need a vector representation of words
  - Obvious approach is One Hot Encoding into vectors of the same dimensionality as the vocabulary size
    - But, …
      - Very big vectors (>171k words in English vocab)
      - No notion of synonymy; all terms orthogonal

# Problem statement

- We'd really like much lower dimensional vectors (far fewer weights)
- … this is a **dimensionality reduction problem**
- Ideally vectors should capture similarity (cat->kitten should be **closer** than cat->dog)
  - … we need to **learn similarity**

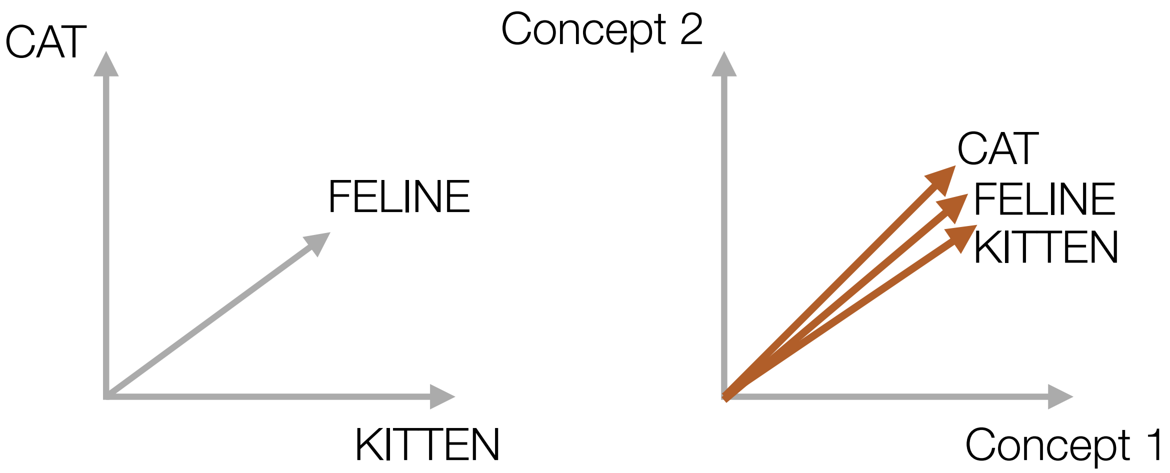
# Dimensionality Reduction

- **Learned** dimensionality reduction can be easily achieved through a linear projection (potentially followed by a non-linearity)
    - e.g. a fully-connected layer

# Learning Similarity: Distributional semantics

- **Distributional Hypothesis**:
    - **words that are close in meaning will occur in similar pieces of text**
- Exploit this to uncover *hidden meaning*
    - Latent Semantic Analysis
    - Word Embeddings

# Concept Spaces/Semantic Spaces



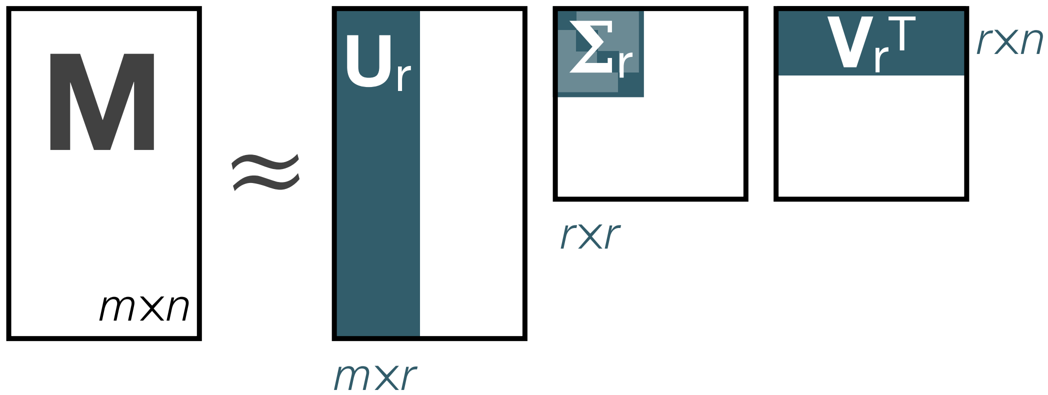

# Latent Semantic Analysis

- Consider a **term-document matrix** which described occurrences of terms in documents
    - Clearly going to be sparse
    - Could be weighted (c.f. TF-IDF)

# Latent Semantic Analysis

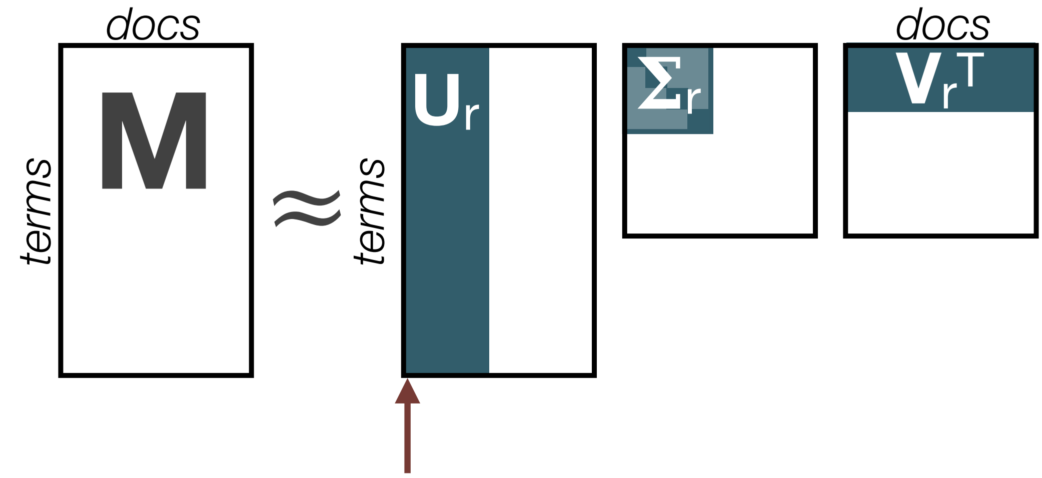
- LSA works by making a **low-rank approximation** under the following assumptions:
  - The original term-document matrix is **noisy**
    - anecdotal instances of terms are to be eliminated.
      - the approximated matrix is **de-noised**
  - The original term-document matrix is **overly sparse** relative to the "*true*" term-document matrix
    - We want to capture **synonymy**

## **Truncated** Singular Value Decomposition



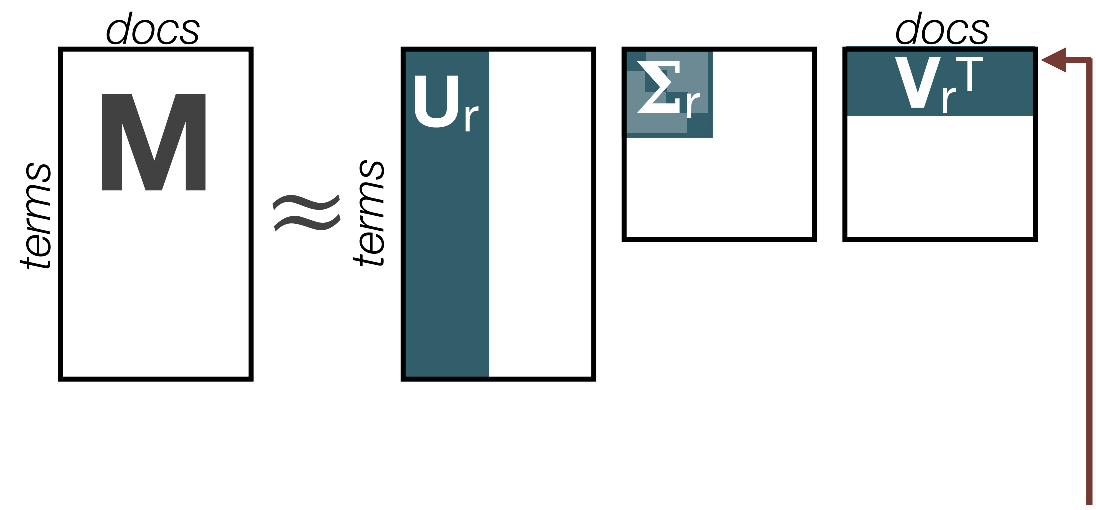

*Truncated SVD considers only the* ***largest*** *r singular values (and corresponding left & right vectors)*

# Latent Semantic Analysis



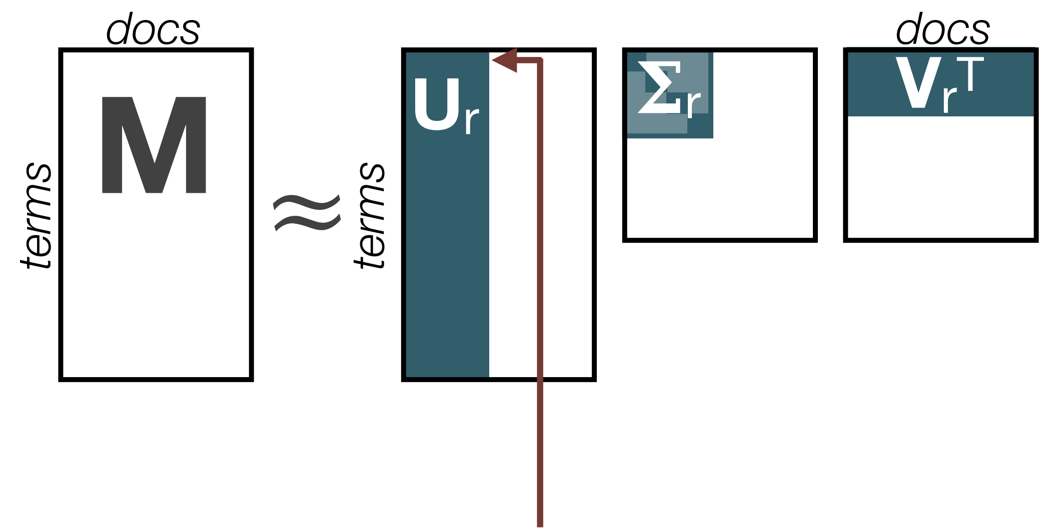

*Each column corresponds to an eigenvector of* $\mathbf{MM}^T$
*(i.e. proportional to covariance or correlation of the terms)*
*These are the "**concepts**"*

# Latent Semantic Analysis



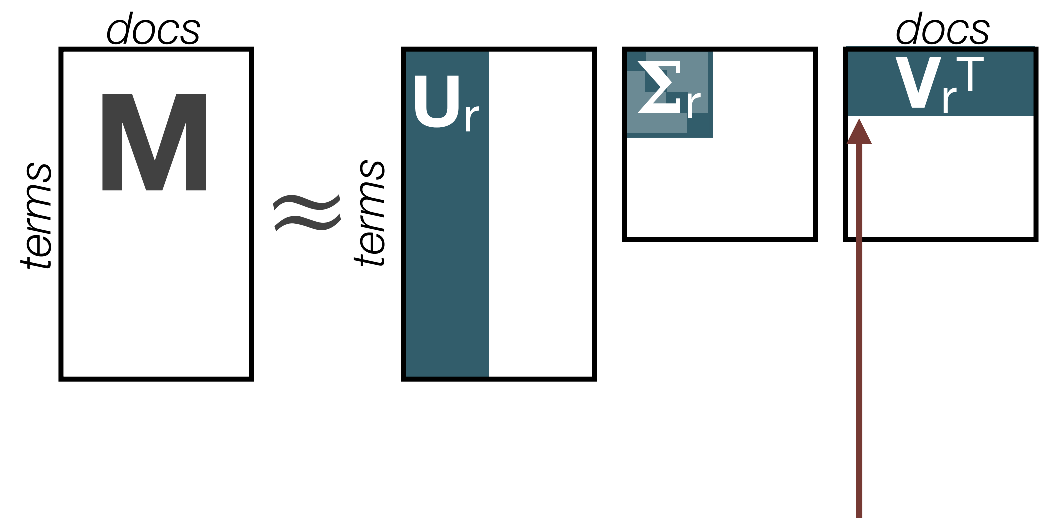

*Each row corresponds to an eigenvector of* $\mathbf{M}^T\mathbf{M}$
*(i.e. proportional to covariance or correlation of the documents)*
*These are the "**concepts**"*

# Latent Semantic Analysis



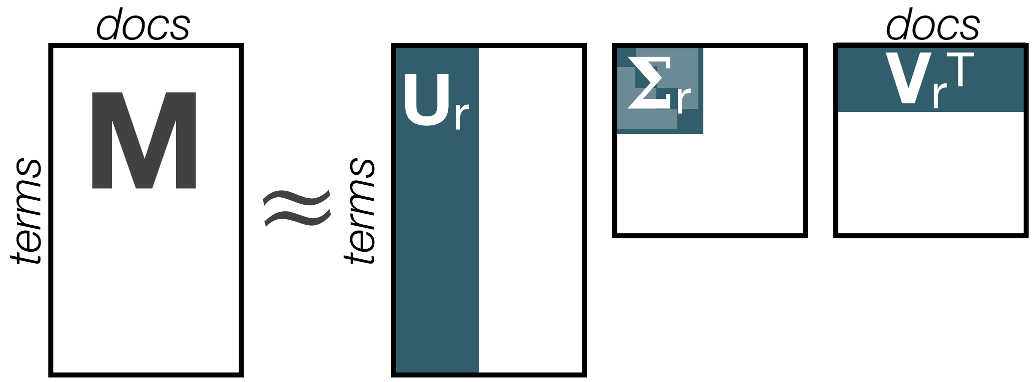

*Each row corresponds to an r dimensional vector that describes a term as a vector of weights with respect to the r concepts*

# Latent Semantic Analysis



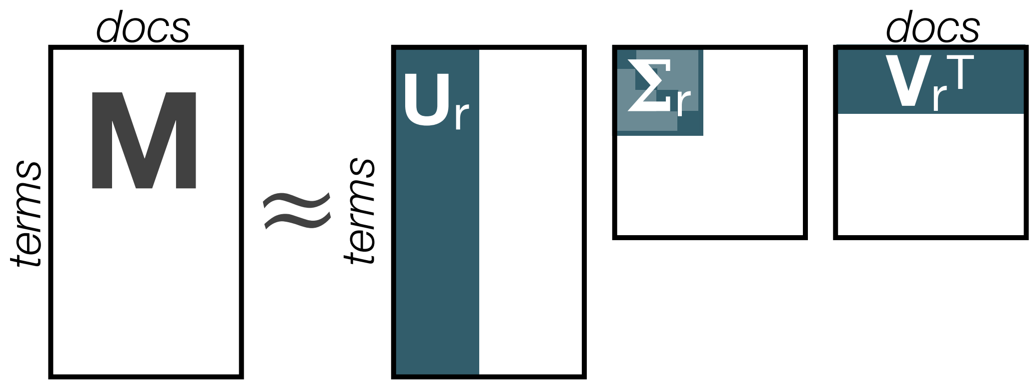

*Each column corresponds to an r dimensional vector that describes a term as a vector of weights with respect to the r concepts*

# Important Note




*The term-concepts and the document-concepts are not the same - they have the same dimensionality, but represent different spaces*
*They are intrinsically linked though, and it is possible to project one into the other*

# What exactly is a concept?



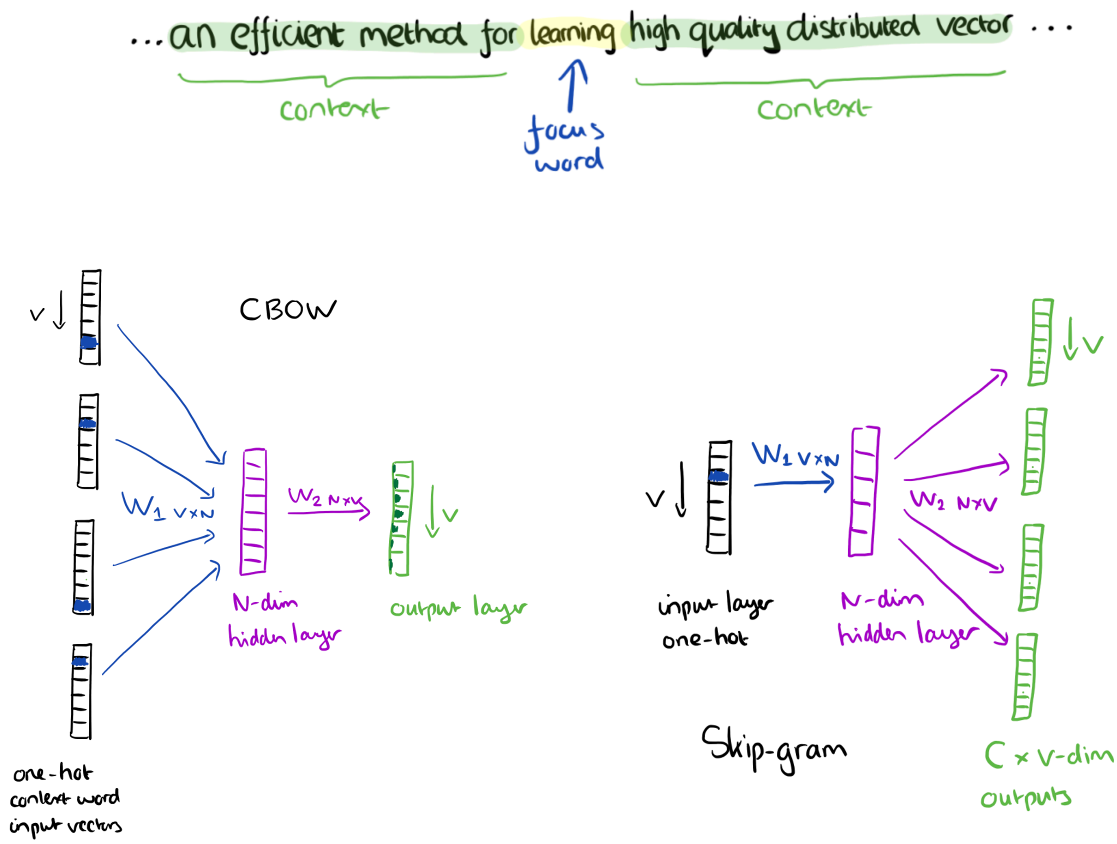

*A linear combination of terms (or documents).*
*Not necessarily "comprehensible" -*
*e.g. 1.3452 * **car** - 0.2828 * **bottle***

# Word Embeddings

- Can we build a better vector representation of words?
  - Lower dimensionality (but dense)
  - Capturing synonymous words
  - What about capturing algebraic semantics?
    - word2vec(“Brother”) - word2vec(“Man”) + word2vec(“Woman”) = word2vec(“Sister”)

# Word Embeddings…

- Many models of mapping words to vectors have been proposed.
  - A pair of commonly used models is known as “word2vec” and was introduced by Mikolov *et al.* at Google
    - They’re both shallow two-layer neural nets, but trained on *lots* of data
    - Ironically, although the paper introducing the models has 9700 citations, it was never officially published after being rejected (and heavily slated by the reviewers) of ICLR 2013!
  - Another popular model is GloVe “Global Vectors for Word Representation” by Pennington *et al.*
  - All these models have all the features from the previous slides!
  - *Note that practically speaking, you don’t have to train the models - you can just download a pretrained variant*




*Images from https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/*

# word2vec limitations

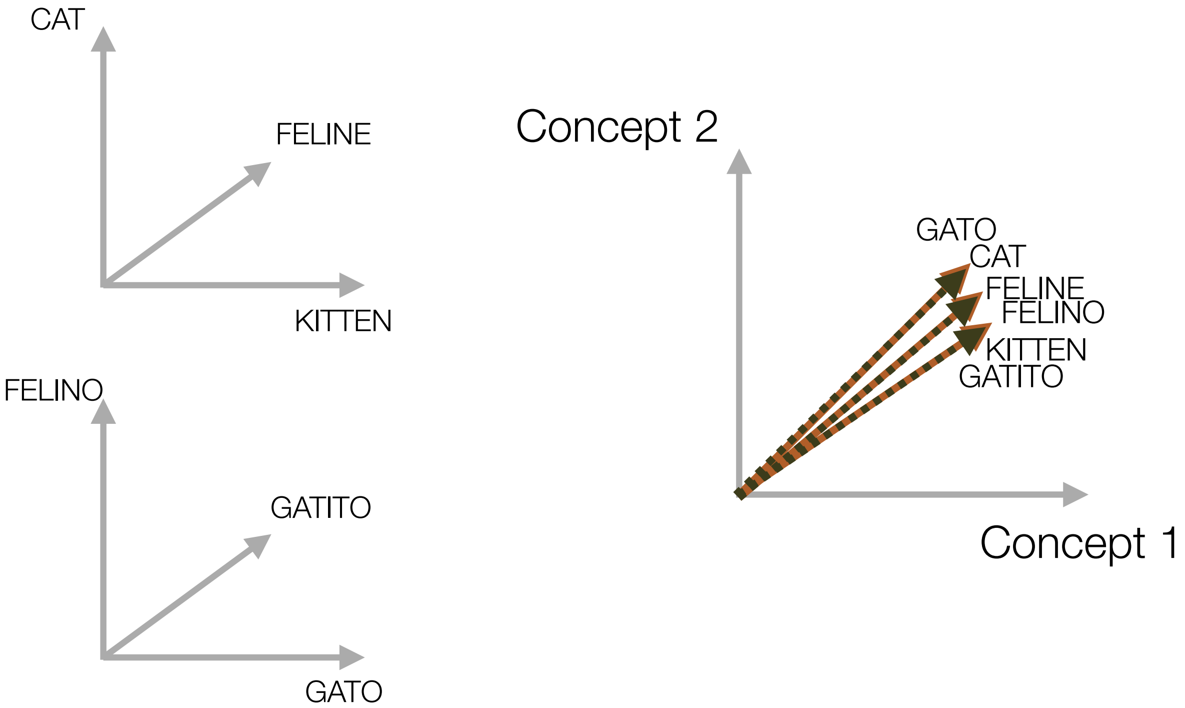
- word2vec works well, but doesn't deal with out of vocabulary (OOV) words
    - A newer model called FastText attempts to solve this problem by building the embeddings from character n-grams
        - The idea is that words with similar meaning often have similar sub-strings (e.g. locat[e/ing/ion])

# Implementation Note

- In PyTorch, we don't ever represent a word in one-hot form
  - Too expensive & unnecessary
  - Rather, the **Embedding Layer**, is implemented as a lookup-table between the input word index and the corresponding output vector
    - Can still be differentiated though of course as it's functionally equivalent to multiplication of the layer weights by an OHE vector

# Mining semantic correspondences across feature domains

# Embedding across languages




# Cross-Language LSI

- Use a bilingual (or multilingual) **training** corpus to build a single term-document matrix
    - each document vector contains terms from the original language and its translation(s)

| | CAT | KITTEN | FELINE | FELINO | GATO | GATITO | … |
|---|---|---|---|---|---|---|---|
| doc1 | 1 | 0 | 0 | 0 | 1 | 0 | … |
| doc2 | 1 | 1 | 1 | 1 | 1 | 1 | … |
| … | | | | | | | |

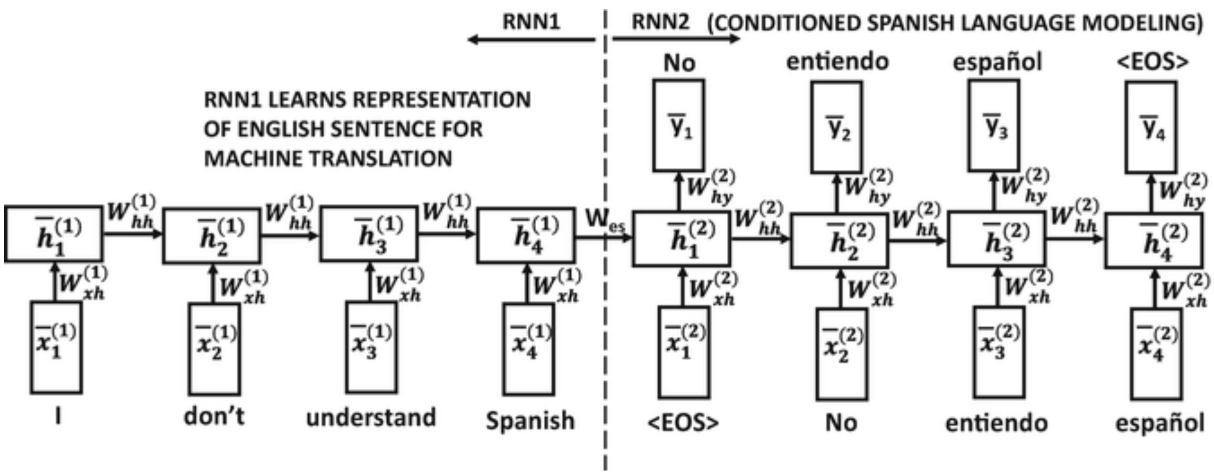
# Cross-Language LSI

- Decompose with SVD as per standard LSI
- Perform queries by projecting into the lower dimensional space as before
  - but just use one language and set the rest to 0

| | CAT | KITTEN | FELINE | FELINO | GATO | GATITO | … |
|---|---|---|---|---|---|---|---|
| query | 1 | 0 | 0 | 0 | 0 | 0 | … |

  - Obviously this still has a problem in the sense that all the indexed documents needed translation…

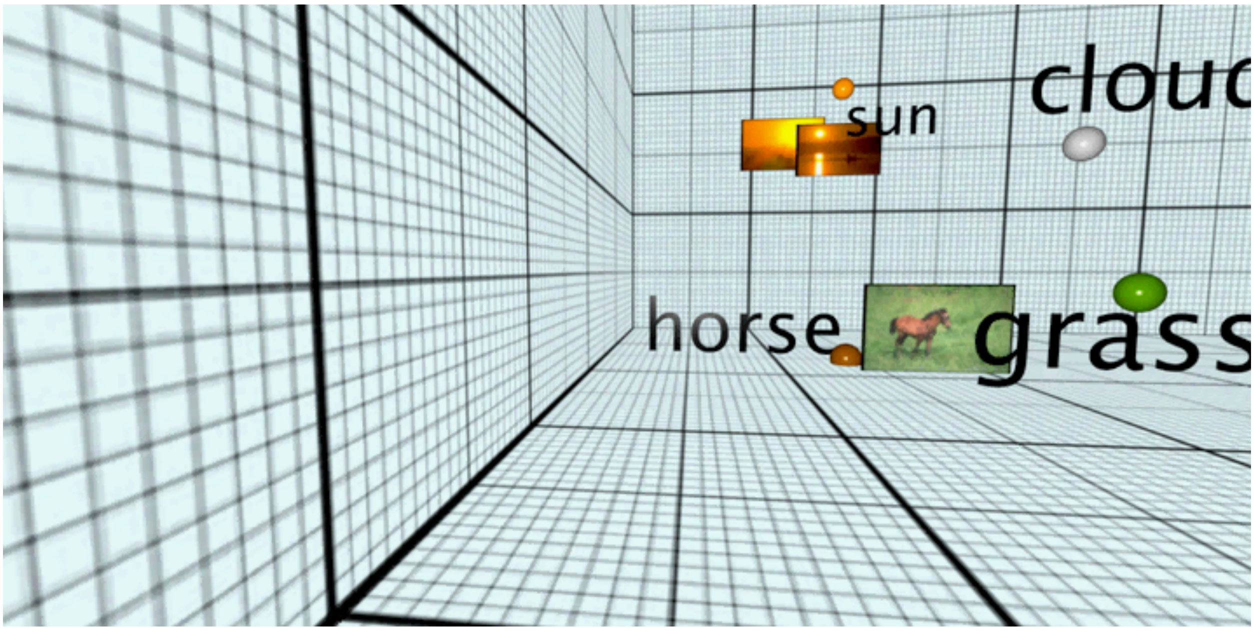
# Sequence-Sequence Translation



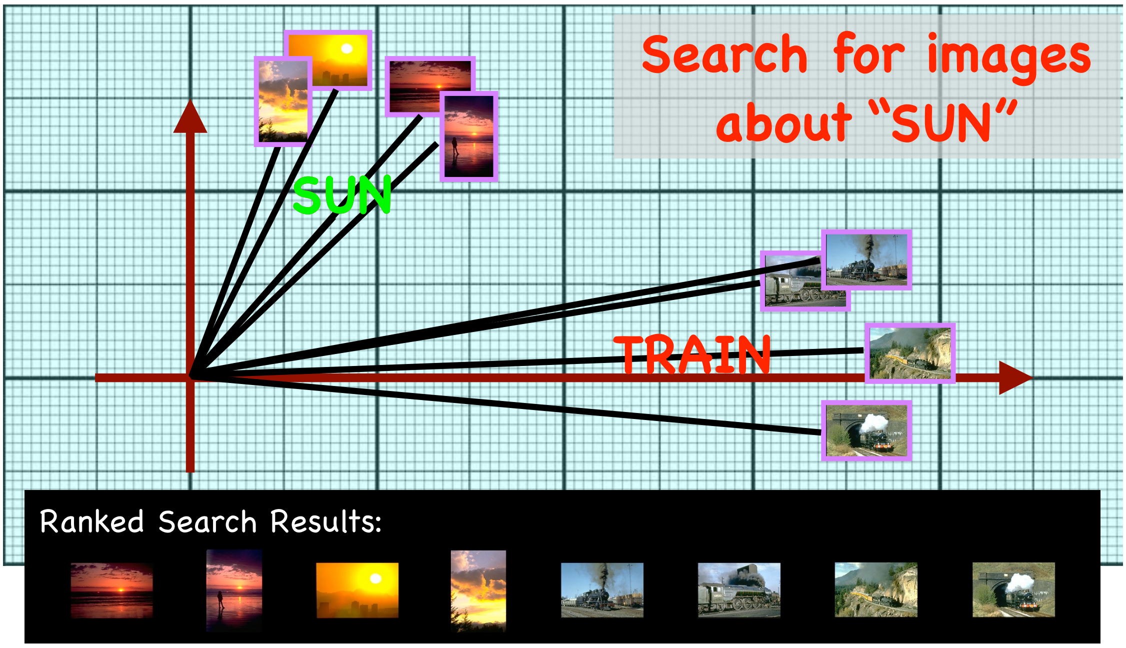

*https://link.springer.com/chapter/10.1007/978-3-319-73531-3_10*

# Image-Concept Embedding

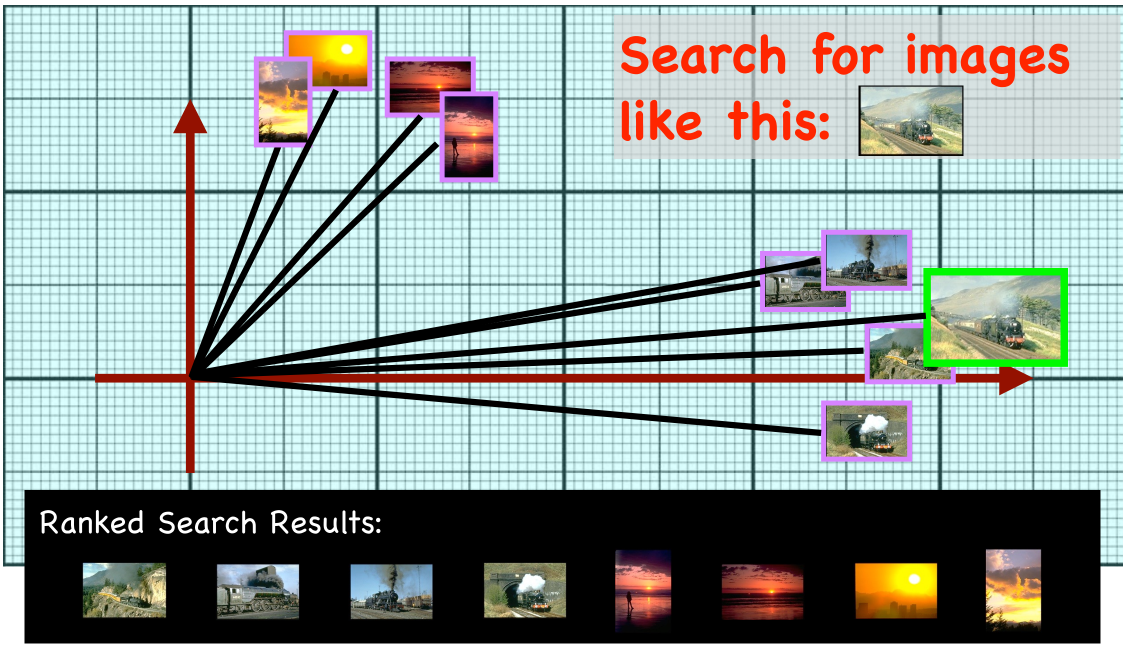
- Basic idea: Create a large multidimensional space in which images, keywords (or other metadata) and visual information can be placed.
- In the training stage learn how keywords are related to visual terms and images.
  - Place related visual terms, images and keywords close-together within the space.
- In the projection stage unannotated images can be placed in the space based upon the visual terms they contain.
  - The placement should be such that they lie near keywords that describe them.



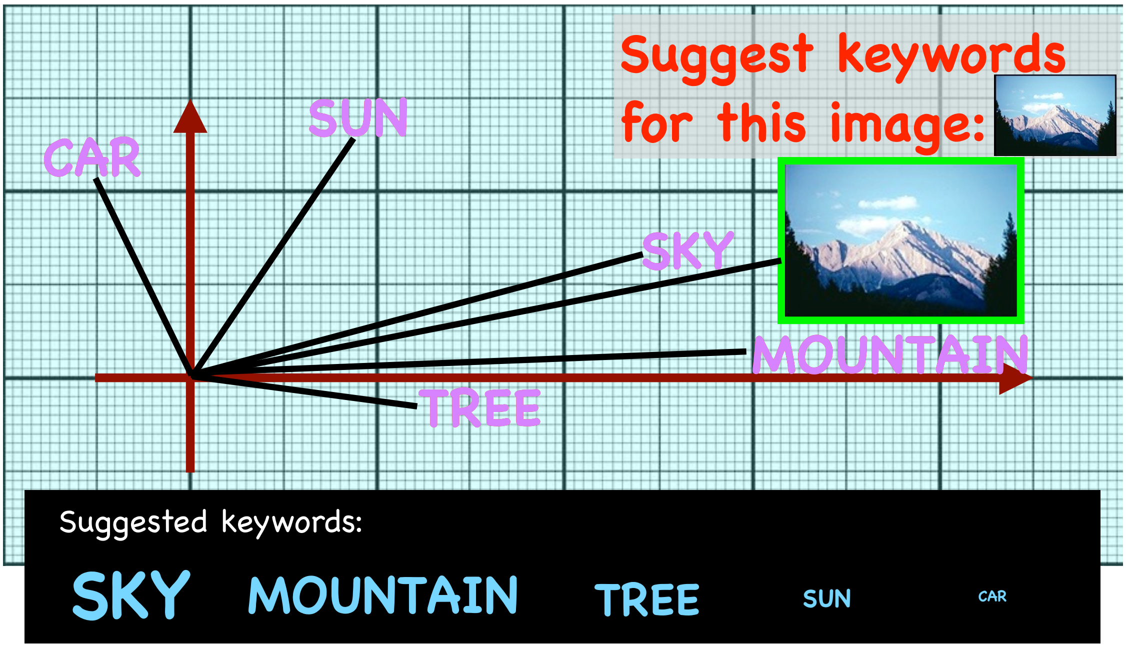
# Searching by Keyword



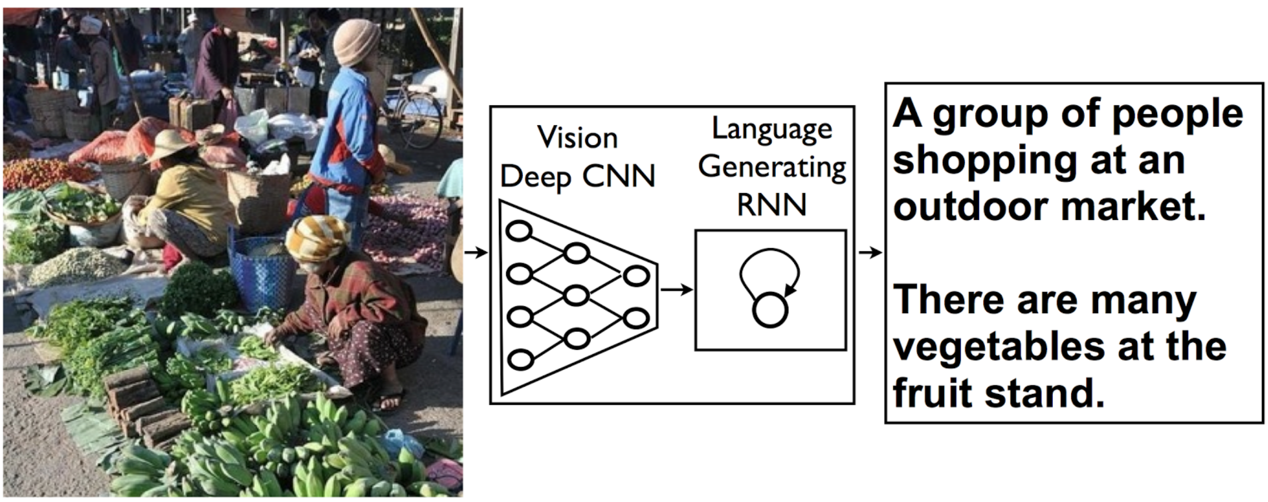

# Searching by Image



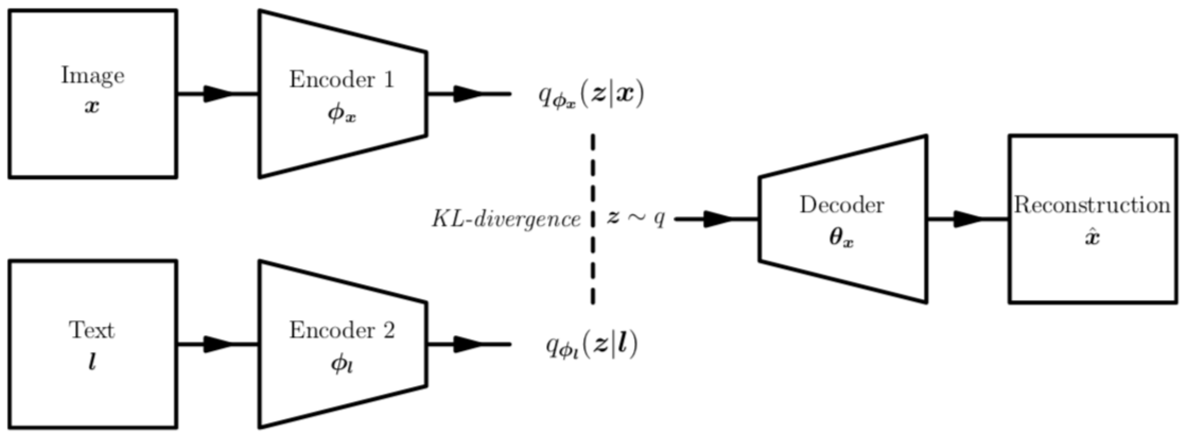
# Suggesting Keywords




# Image captioning



# Probabilistic Semantic Embedding




*https://openreview.net/pdf?id=r1xwqjRcY7*