INFORMATION RETRIEVAL

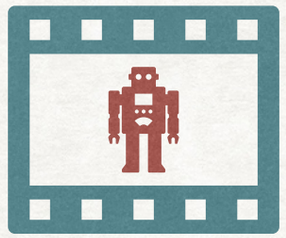
Luca Manzoni
lmanzoni@units.it

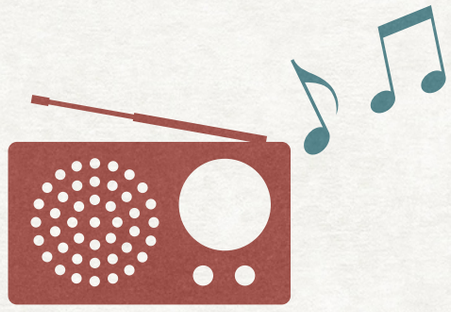
Lecture 11

LECTURE OUTLINE

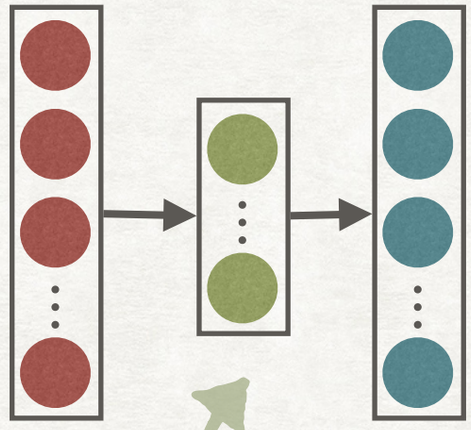
THANKS FOR ALL THE FISH

Multimedia Information Retrieval





Actually more like a brief list of essential concepts



Where to go next

Neural Networks in Information retrieval

MULTIMEDIA INFORMATION RETRIEVAL

MAIN PROBLEMS

FOR IMAGES/AUDIO/VIDEO

- Queries in one medium must be matched with other kinds of media (*cross-media IR*):
  - e.g., given a text retrieve an image.
  - Different media usually resides in different features spaces!
- There is an additional problem: the “semantic gap”.

POSSIBLE APPROACHES

OR A COMBINATION OF THEM

- There are possible approaches that can be used to index and answer query about multimedia documents:
  - Metadata-driven retrieval
  - Piggy-back text retrieval
  - Content-based retrieval
  - Automatic image annotation
  - Fingerprinting (images and video)

METADATA-DRIVEN RETRIEVAL

IT'S STILL TEXT!

- Many multimedia format have additional metadata attached:
  - Images can have EXIF data.
  - Audio files can have ID3 tags.
  - Other kind of files can be annotated, for example with XML metadata using the Dublin Core standard: https://www.dublincore.org
- Notice that metadata could be stored *outside* the document.

PIGGY-BACK TEXT RETRIEVAL

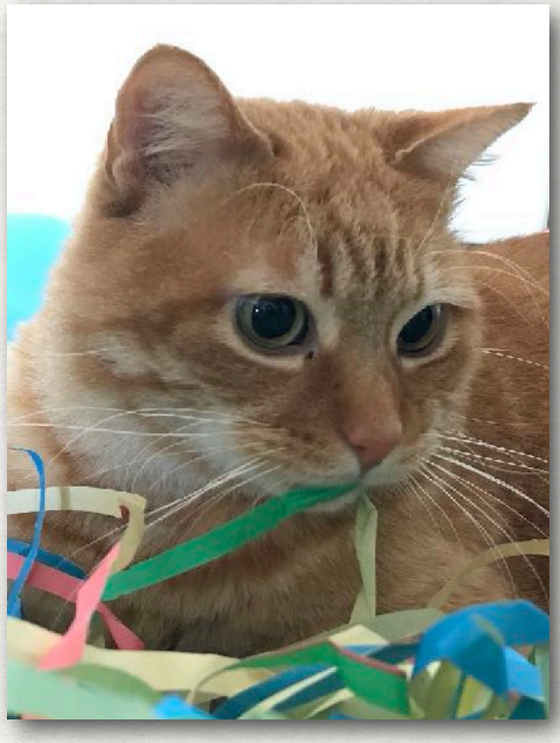
TEXT, AGAIN.

Sometimes we can use text that is not part of the document or its metadata to index a file:

…Bla bla…

Look at this wonderful [cat picture]!

…Bla bla…



…Bla bla…

This is [my cat]!

…Bla bla…

PIGGY-BACK TEXT RETRIEVAL

TEXT, AGAIN.

- Anchor text and captions are not the only possibilities.
- For movies with subtitles we can extract the subtitle (possibly using OCR)
- For Midi files we can extract the pitches and duration of the notes.
- Still, we are only using text to find the content of an image/audio/video.

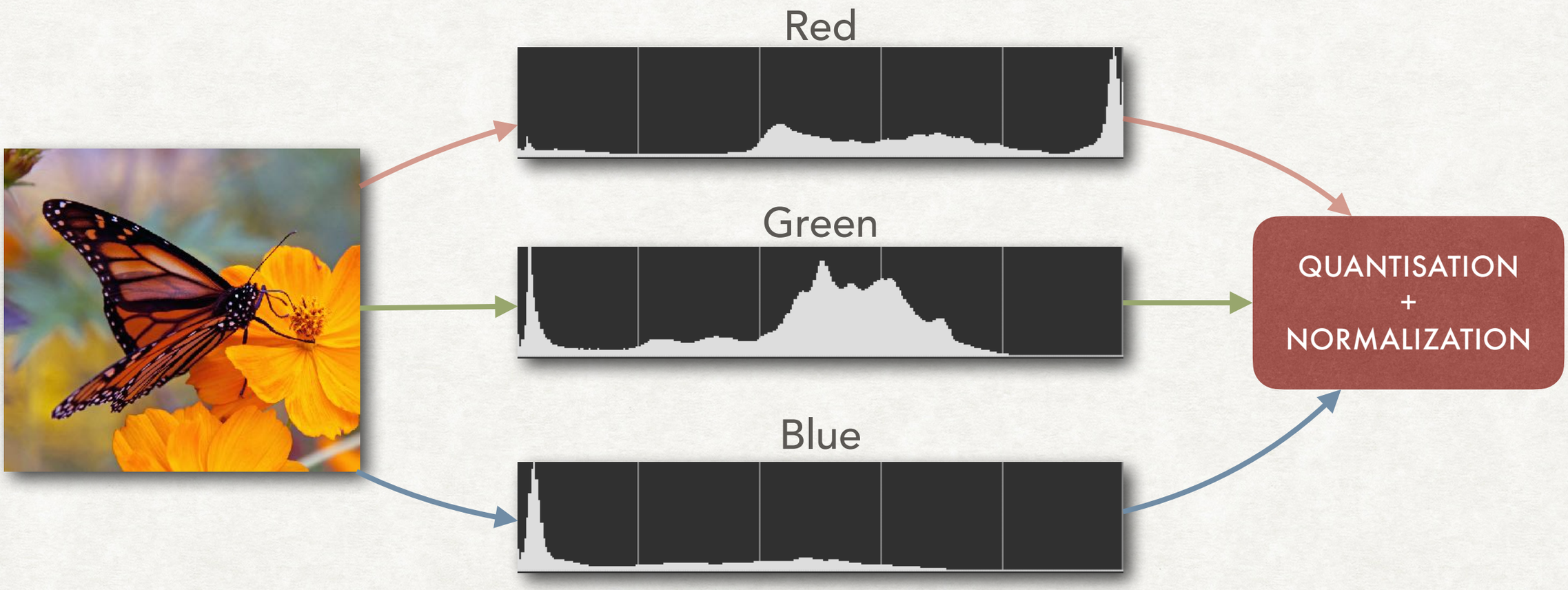
CONTENT-BASED RETRIEVAL

MORE THAN TEXT

- Create a collection of features describing, for example an image (we will consider images for content-based retrieval).
- The choice of feature will be the way we represent an image, so this choice has profound effect on the quality of the retrieval.
- We will see two examples for image retrieval:
  - Color histograms as features.
  - Statistical moments as features.

CONTENT-BASED RETRIEVAL

COLOR HISTOGRAMS



Given an image we retrieve the color histograms,
then we can quantise (for reasons of space)
and normalise (to make them comparable with other histograms) them

CONTENT-BASED RETRIEVAL

STATISTICAL MOMENTS

- The idea is to use a more compact representation of the distribution of pixels in an image.

- Let $p(i, j)$ be the intensity of the pixel in row i and column j of an image of height h and width w.

- We can compute the average intensity of a pixel:

$$\mu = \frac{1}{wh} \sum_{i=1}^{h} \sum_{j=1}^{w} p(i, j).$$

CONTENT-BASED RETRIEVAL

STATISTICAL MOMENTS

- We can actually compute every $k > 1$ the k-th central moment of the image:

$$\bar{p}_k = \frac{1}{wh} \sum_{i=1}^{h} \sum_{j=1}^{w} (p(i,j) - \mu)^k.$$

- To make the value comparable among them we define:

$$m_k = \text{sign}(\bar{p}_k) \sqrt{|\bar{p}_k|}$$

- And we use $(\mu, m_1, m_2, \ldots, m_\ell)$ for some ℓ as the feature vector.

AUTOMATIC IMAGE ANNOTATION

FINDING OBJECTS IN IMAGES

- We can have a “black box” automatically annotating a picture with the objects it contains.
- The black box could be, for example, a neural network.
- But we can also use a “human computer”, by having users of a service help us in tagging images.

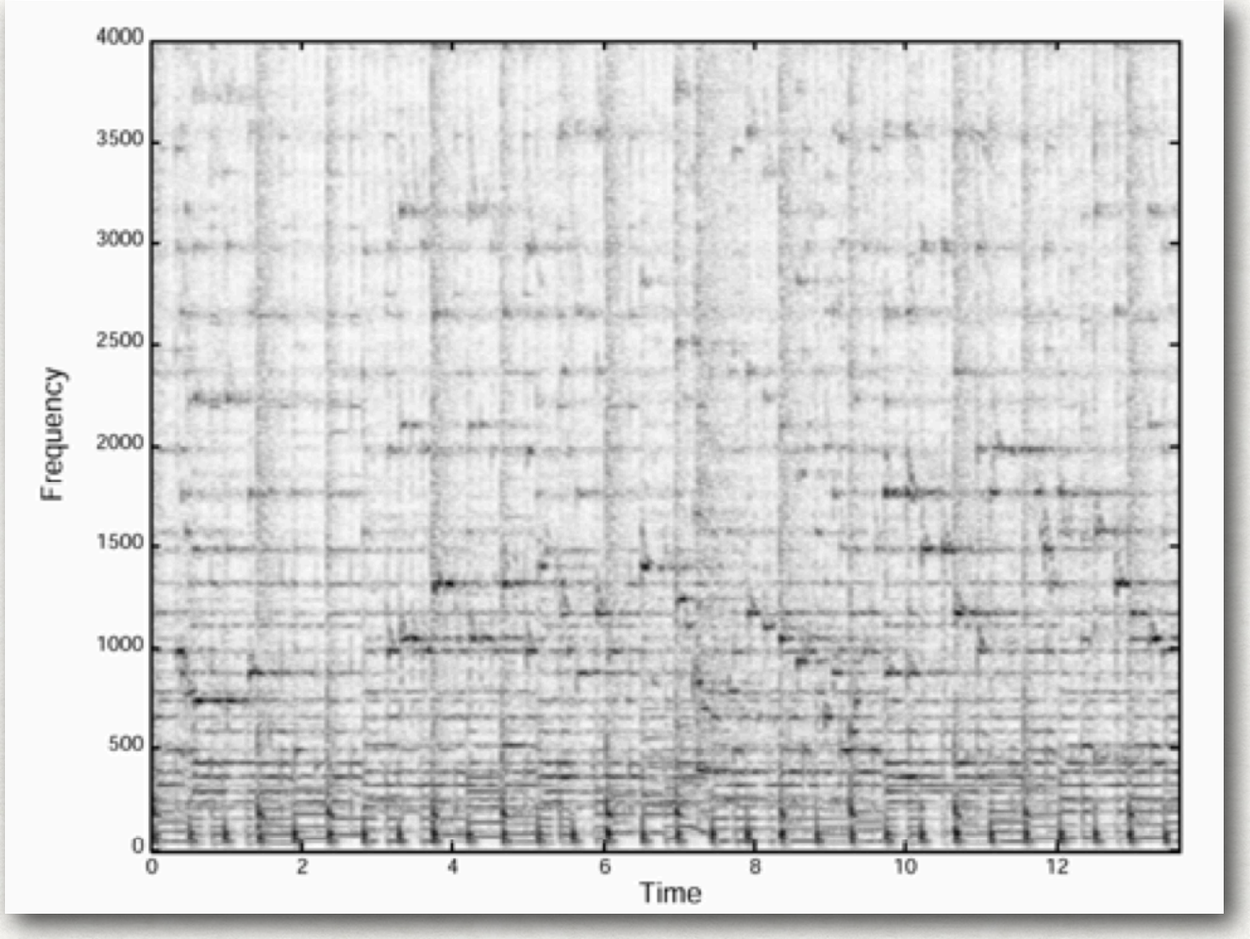
FINGERPRINTING

FOR IMAGES AND AUDIO

- The aim of fingerprinting is to uniquely identify a document inside a database.
- Recall the use of fingerprinting to remove duplicates and near-duplicates: in this case, however, we want to find the "duplicate" of our query.
- For multimedia content, the document is considered "the same" as long as it is "the same" according to the human perception.
- In fact we only need to reasonably identify the document.
- For an example of audio fingerprinting consider *Shazam*.

AUDIO FINGERPRINTING

SPECTROGRAM



The spectrogram represents the spectrum of frequencies of the sound when it varies with time.

We want to use the spectrogram to extract information that can be used to perform retrieval of audio.

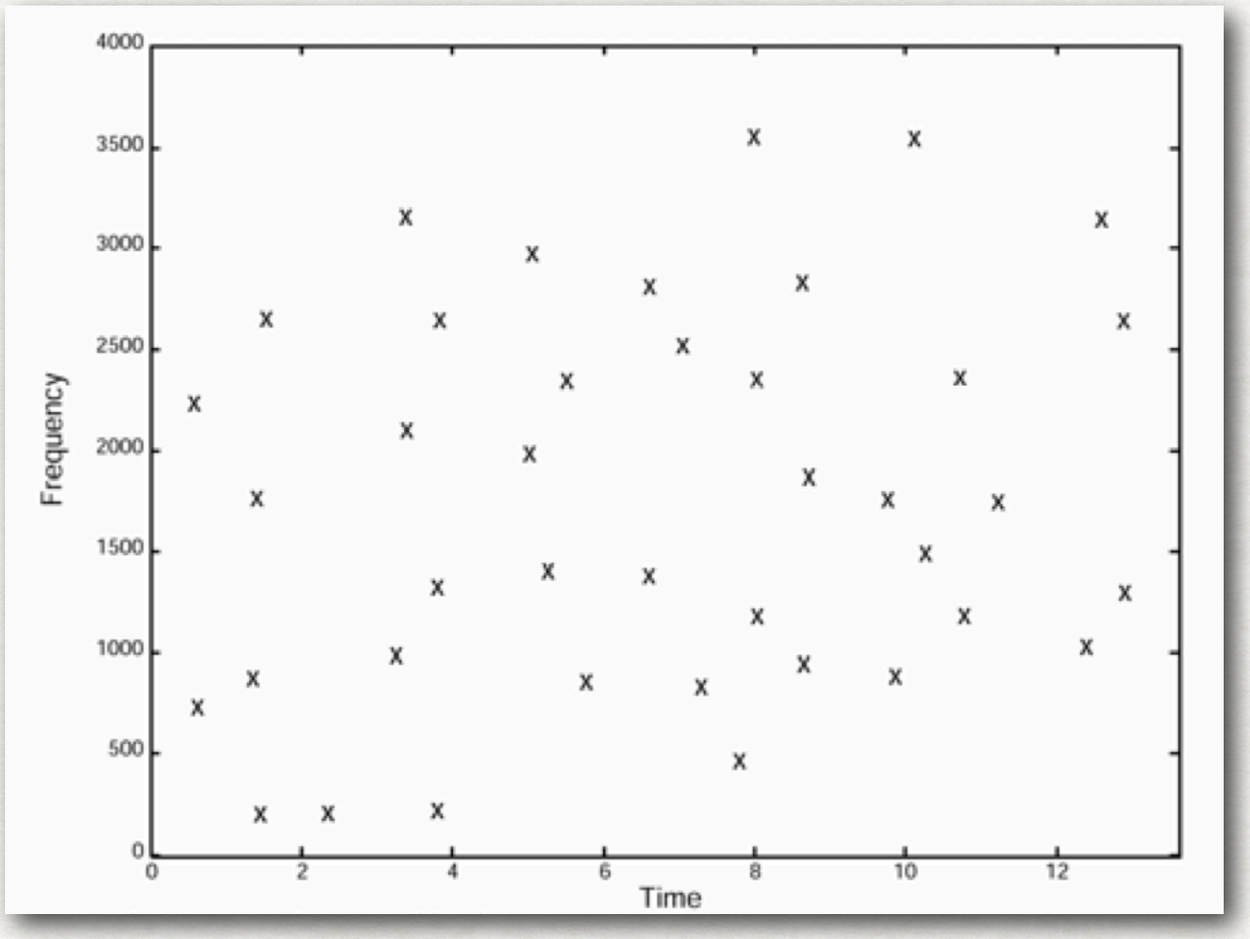
We start by highlighting the peaks in the spectrogram.

Paper:
Wang, Avery.
"An Industrial Strength Audio Search Algorithm."
In Ismir, vol. 2003, pp. 7-13. 2003.

Figures taken from (Wang 2003)

AUDIO FINGERPRINTING

CONSTELLATION DIAGRAM



By taking only the peaks in the spectrogram we obtain the *constellation diagram*.

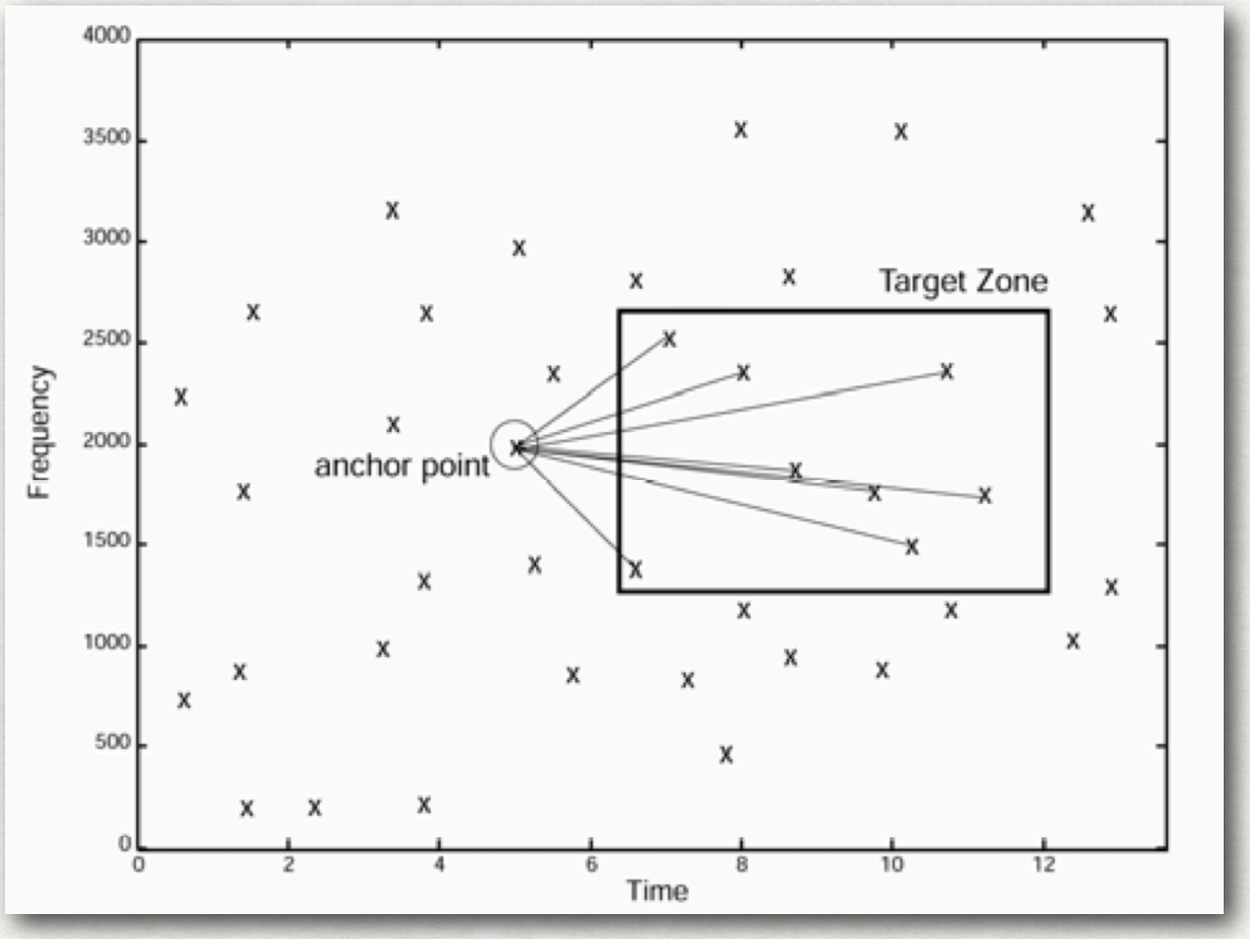
If we play again the same sound we will obtain the same constellation diagram.

However, we may also want to allow non-exact matches.

Wang (2003) defined an hash using pairs of points in the constellation diagram.

AUDIO FINGERPRINTING

FINGERPRINTS



For each point a target zone (a rectangle to the right) is considered

Each pair of anchor point and point in the target zone is encoded as:

- The time difference between the two points.
- The two frequencies.

The representation does not depend on the time we started listening.

AUDIO FINGERPRINTING

MATCHING

- Given an audio query we search all the pairs of points (has values) generated from our corpus matching the ones obtained by the audio query.

- A match is found when a diagonal line is present in the following diagram:

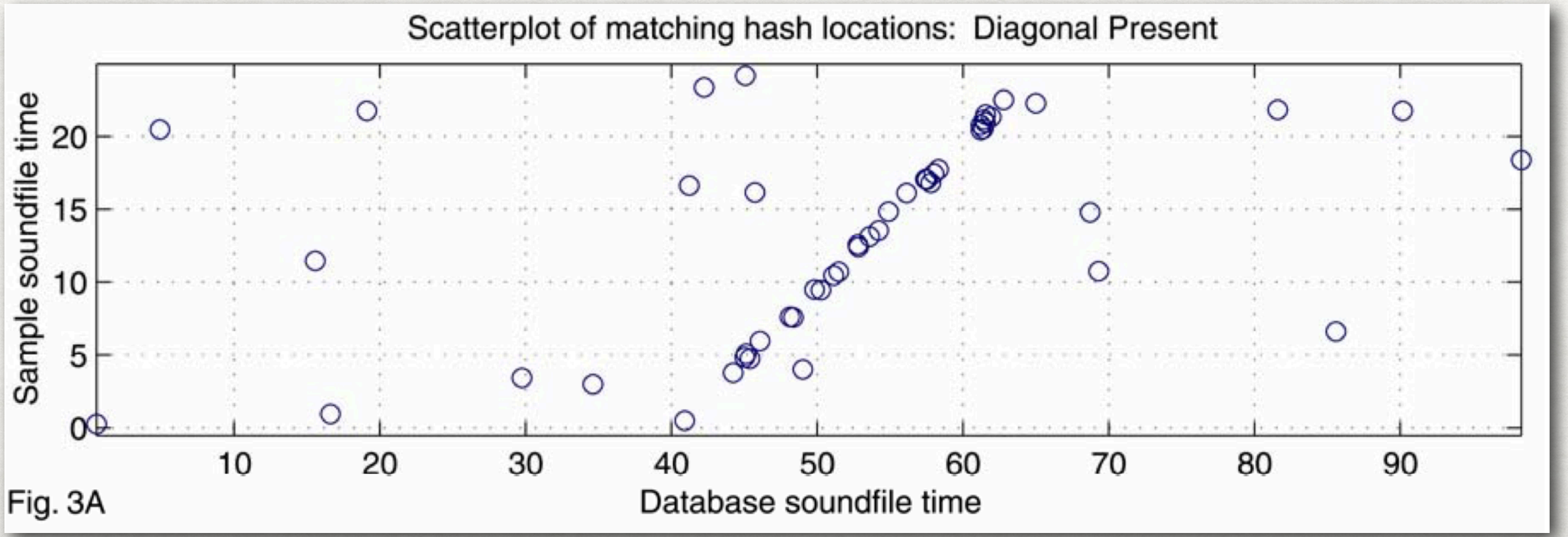


Fig. 3A

NEURAL NETWORKS IN IR

NEURAL NETWORKS AND IR

BECAUSE NEURAL NETWORKS ARE EVERYWHERE NOW

- Suggested reading:
  Bhaskar Mitra, Nick Craswell
  An Introduction to Neural Information Retrieval
  Foundations and Trends in Information Retrieval, 2018

- We assume some knowledge of neural networks

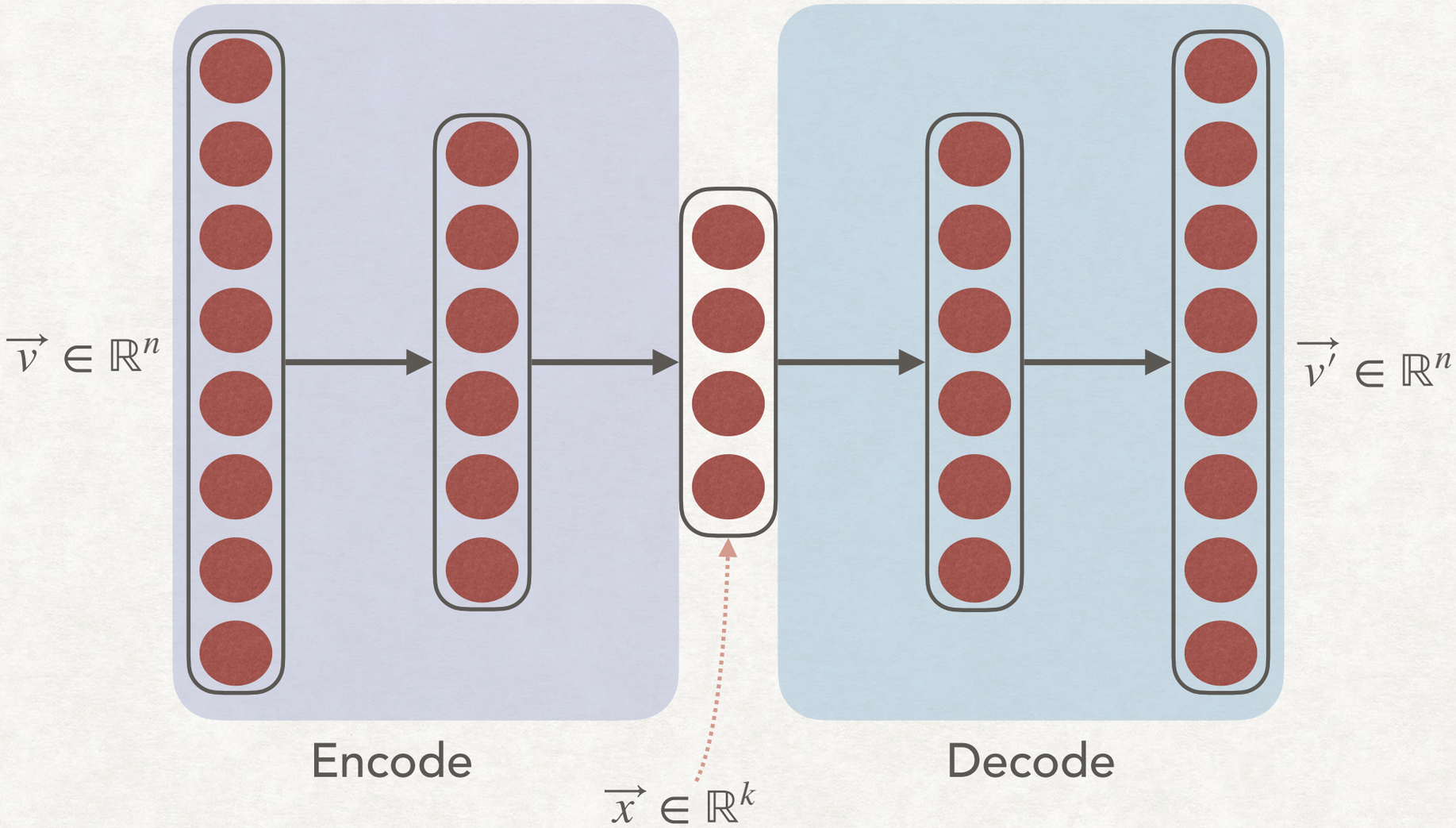
- We will see (briefly) some of the possible applications of neural networks in IR:

  - Learning of term representation

  - Recommender systems using NN

AUTOENCODER

AND DIMENSIONALITY REDUCTION



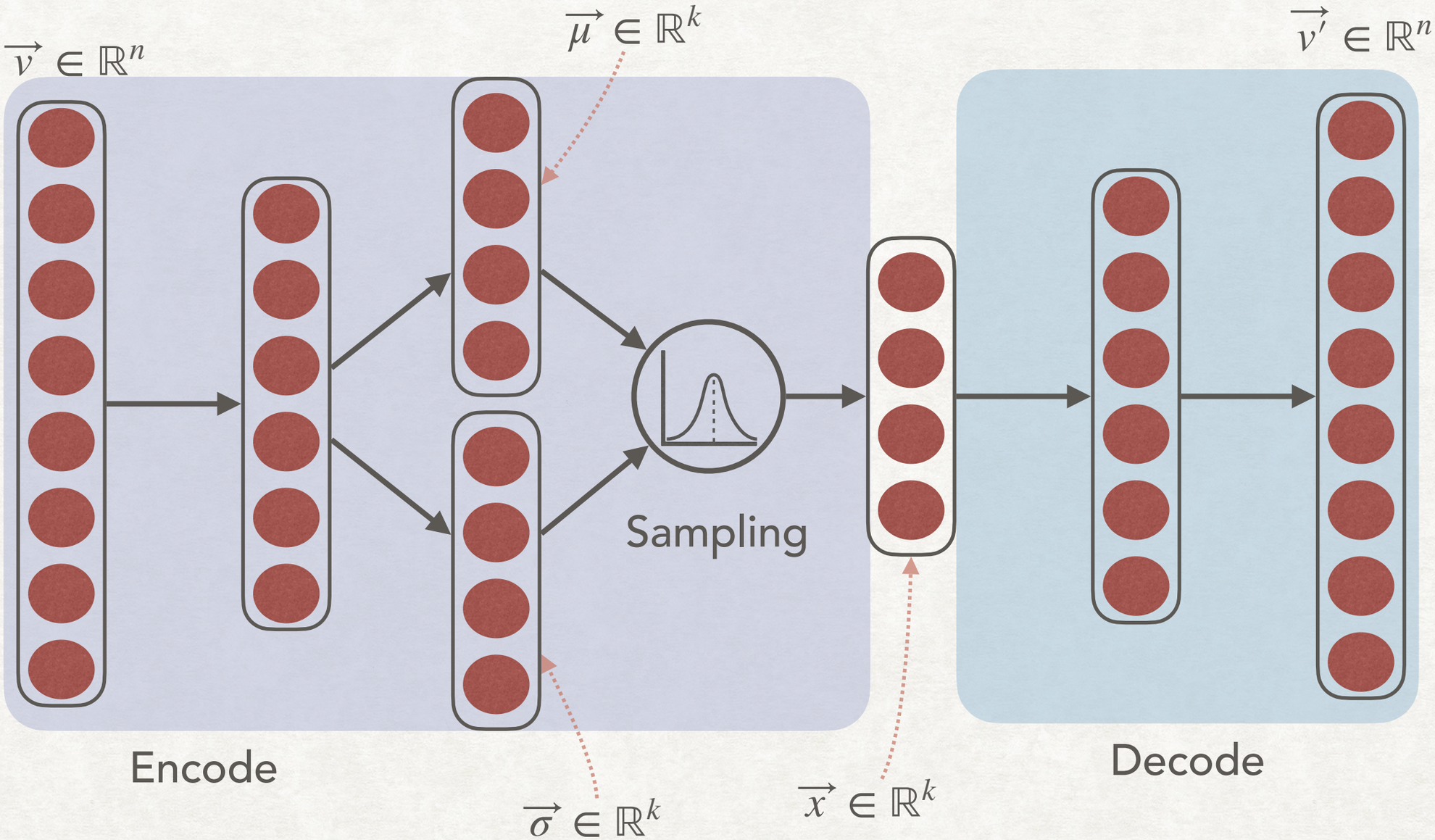
AUTOENCODER

AND DIMENSIONALITY REDUCTION

- Autoencoders are based on the *information bottleneck method* and typically have a "hourglass" shape.
- The input is a vector $\overrightarrow{v} \in \mathbb{R}^n$, the output is also a vector $\overrightarrow{v'} \in \mathbb{R}^n$.
- We want the output vector to be the same as the input vector (i.e., the network learns the identity function).
- The loss function is usually $\mathscr{L}_{\text{autoencoder}} = \|\overrightarrow{v} - \overrightarrow{v'}\|^2$.
- The "bottleneck" is a vector $\overrightarrow{x} \in \mathbb{R}^k$, with $k \ll n$ which represents an encoding of $\overrightarrow{v}$ in a lower dimensional space.

VARIATIONAL AUTOENCODER

A "SMOOTHER" AUTOENCODER



VARIATIONAL AUTOENCODER

A "SMOOTHER" AUTOENCODER

- Similar to an autoencoder, but the encoding part of the network generates two vectors:
  - $\vec{\mu} = (\mu_1, \mu_2, \ldots, \mu_k)$ of the means
  - $\vec{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_k)$ of the standard deviations
- The vector $\vec{x}$ is obtained by sampling k normal distributions with mean a variance obtained by $\vec{\mu}$ and $\vec{\sigma}$: $x_i \sim N(\mu_i, \sigma_i^2)$.
- This should allow to learn a "smoother" latent space.

VARIATIONAL AUTOENCODER

A "SMOOTHER" AUTOENCODER

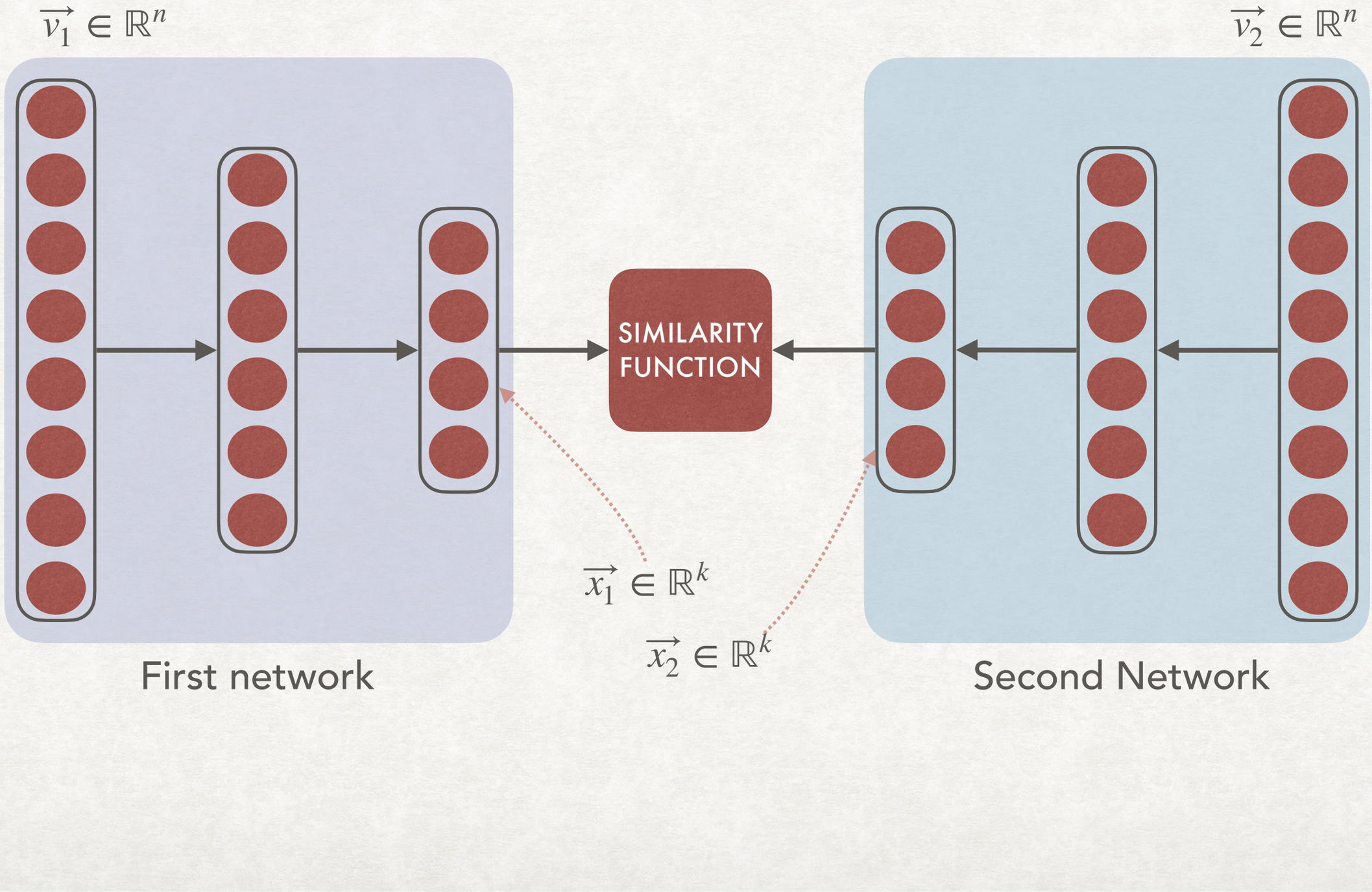
- The loss function should try to penalise setting the standard deviations too close to zero.
- This means that there are two components in the loss function:
  - Reconstruction error: $\mathscr{L}_{\text{reconstruction}} = \|\overrightarrow{v} - \overrightarrow{v'}\|^2$
  - Kullback–Leibler divergence with respect to a unit gaussian:

$$\mathscr{L}_{\text{KL-divergence}} = \sum_{i=1}^{k} \sigma_i^2 + \mu_i^2 - log(\sigma_i) + 1$$

- The loss function is then: $\mathscr{L}_{\text{VAE}} = \mathscr{L}_{\text{reconstruction}} + \mathscr{L}_{\text{KL-divergence}}$

SIAMESE NETWORKS

A REPRESENTATION FOR COMPUTING SIMILARITY



SIAMESE NETWORKS

A REPRESENTATION FOR COMPUTING SIMILARITY

- Autoencoders and VAE use a latent space representation that is useful for reconstructing the original input…
- …but sometimes we are interested in a latent space representation that is useful for computing similarities.
- Two networks (models) maps two inputs $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ into the same latent space, obtaining $\overrightarrow{x_1}$ and $\overrightarrow{x_2}$.
- We compute the similarity between $\overrightarrow{x_1}$ and $\overrightarrow{x_2}$ in the latent space using a classical similarity measure, like cosine similarity.

SIAMESE NETWORKS

A REPRESENTATION FOR COMPUTING SIMILARITY

- A possible way of learning for siamese networks is to consider each input sample as a triple.
- We obtain tree outputs: $\overrightarrow{x_1}$, $\overrightarrow{x_2}$, and $\overrightarrow{y}$.
- We know that $\overrightarrow{x_1}$ should be more similar to $\overrightarrow{y}$ that $\overrightarrow{x_2}$.
- We define the loss function to represent this relation:
- $\mathscr{L}_{\text{siamese}} = \log\left(1 + e^{-\gamma(sim(\overrightarrow{y},\overrightarrow{x_1}) - sim(\overrightarrow{y},\overrightarrow{x_2})}\right)$
  where γ is a parameter, usually set to 10.

DOCUMENT AUTOENCODER

LEARNING A LATENT REPRESENTATION

- Relevant paper:
  Salakhutdinov, R. and G. Hinton. 2009. "*Semantic hashing*".
  International Journal of Approximate Reasoning. 50(7): 969–978.

- Idea: use auto-encoders to learn a latent-space representation of a document.

- A network is trained using a one-hot encoding of the 2000 most common terms (without stopwords) to produce a binary vector encoding of the documents.

DOCUMENT AUTOENCODER

LEARNING A LATENT REPRESENTATION

- Similar documents with the same hash vector can be efficiently retrieved.
- The auto encoder acts as an hash function where similar documents ends up in the same “bin”.
- In other works also variational auto encoders were used.
- Main problem: a vocabulary of a few thousand words might be too small in practical problems.
- Possible solution: use of trigraphs instead of words as input.

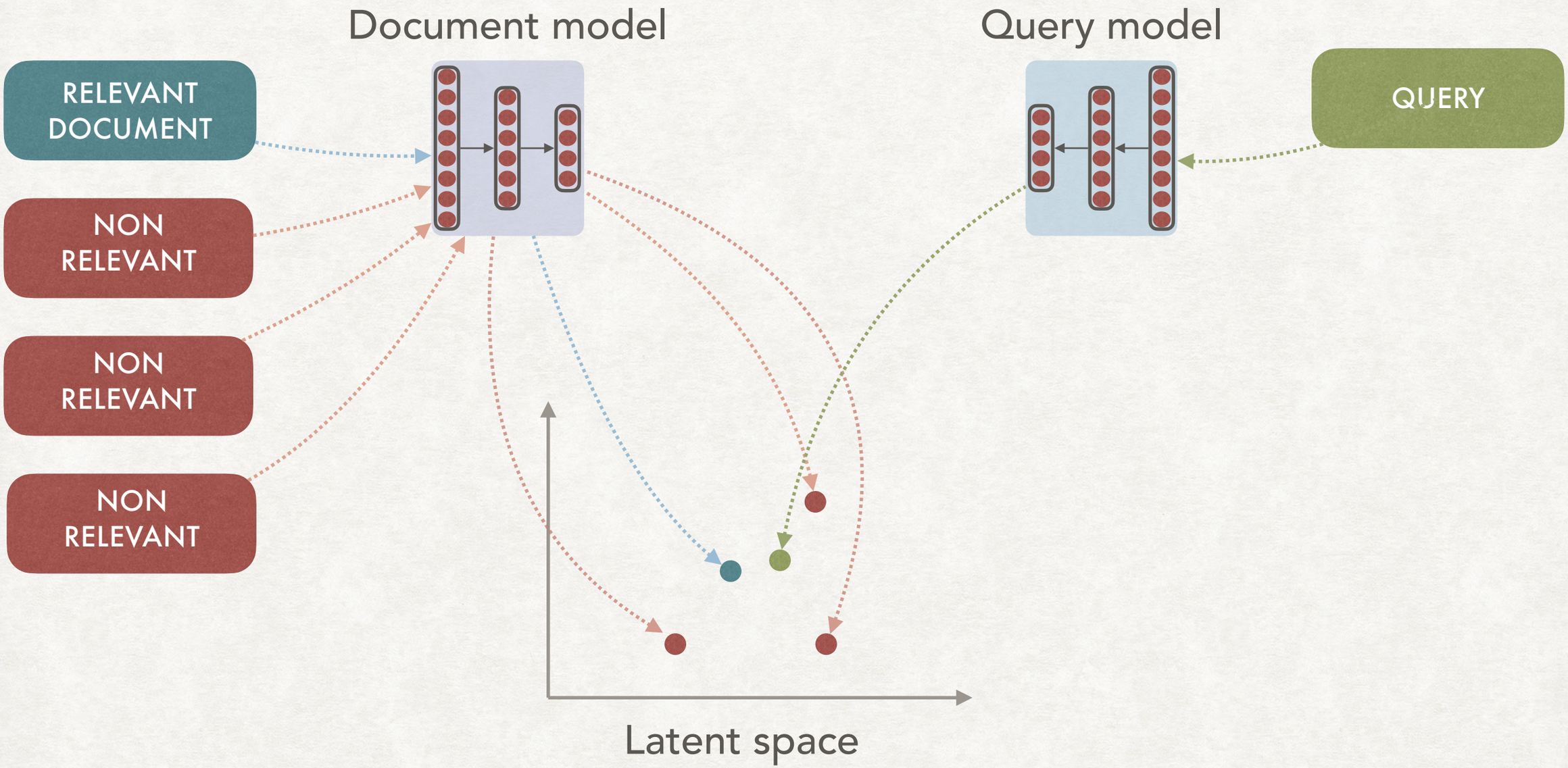
SIAMESE NETWORKS

LEARNING BY DOCUMENTS AND QUERIES

- One approach is to learn a representation using both documents and queries at the same time.
- An approach using siamese networks is the *Deep Semantic Similarity Model* (DSSM).
- Relevant paper:
  Huang, P.-S., X. He, J. Gao, L. Deng, A. Acero, and L. Heck. 2013. "Learning deep structured semantic models for web search using clickthrough data". In: Proc. CIKM. ACM. 2333–2338.
- Two models, one for the query and one for the documents.

SIAMESE NETWORKS

LEARNING BY DOCUMENTS AND QUERIES



SIAMESE NETWORKS

LEARNING BY DOCUMENTS AND QUERIES

- The document titles and the queries are represented as a collection of trigraphs.
- Each sample consists of a query $\overrightarrow{q}$, a relevant document $\overrightarrow{d^+}$ and a set of non-relevant document D^- randomly sampled from the full collection.
- The cosine similarity was used as the similarity measure.
- The loss function used was:

$$\mathcal{L}_{\text{dssm}}(\overrightarrow{q}, \overrightarrow{d^+}, D^-) = -\log\left(\frac{e^{\gamma \cos(\overrightarrow{d^+}, \overrightarrow{q})}}{\sum_{\overrightarrow{d} \in D^- \cup \{\overrightarrow{d^+}\}} e^{\gamma \cos(\overrightarrow{d}, \overrightarrow{q})}}\right)$$

LEXICAL AND SEMANTIC MATCHING

AND THE PROBLEM WITH RARE TERMS

- Embeddings into a latent space as the ones produced by NN have one problem: they tend to produce poor embeddings for rare terms.
- For rare terms a “classical” lexical matching is more effective.
- But for other queries, looking at the semantics via the embedding is more effective (the documents do not contain the same terms as the query).
- In general, lexical and semantic matching tends to perform well on different kinds of queries.

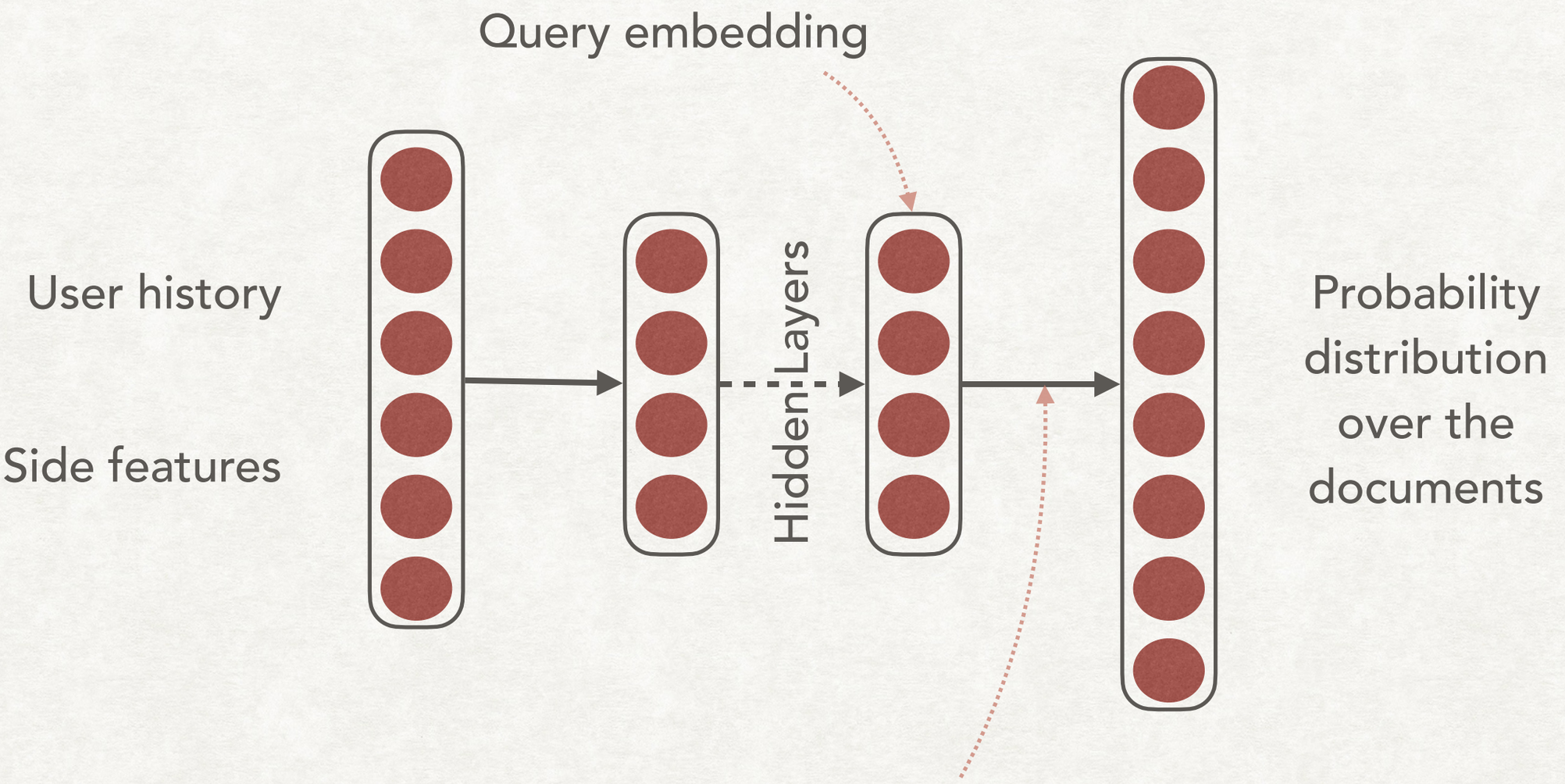
DNN FOR RECOMMENDER SYSTEMS

AN EXAMPLE

- It is possible to use a DNN to build a recommender system to improve with respect to matrix factorisation:
  - Input: a vector $\overrightarrow{x}$ representing the user query. It can contain sparse features (e.g., watch history, liked items) and dense features (e.g., time of the last interaction with the system).
  - Output $\hat{p}$ is a probability distribution across all documents in the corpus representing the probability that the user will like/be interested/watch them.
    This can be obtained using a softmax activation in the last layer.

DNN FOR RECOMMENDER SYSTEMS

AN EXAMPLE



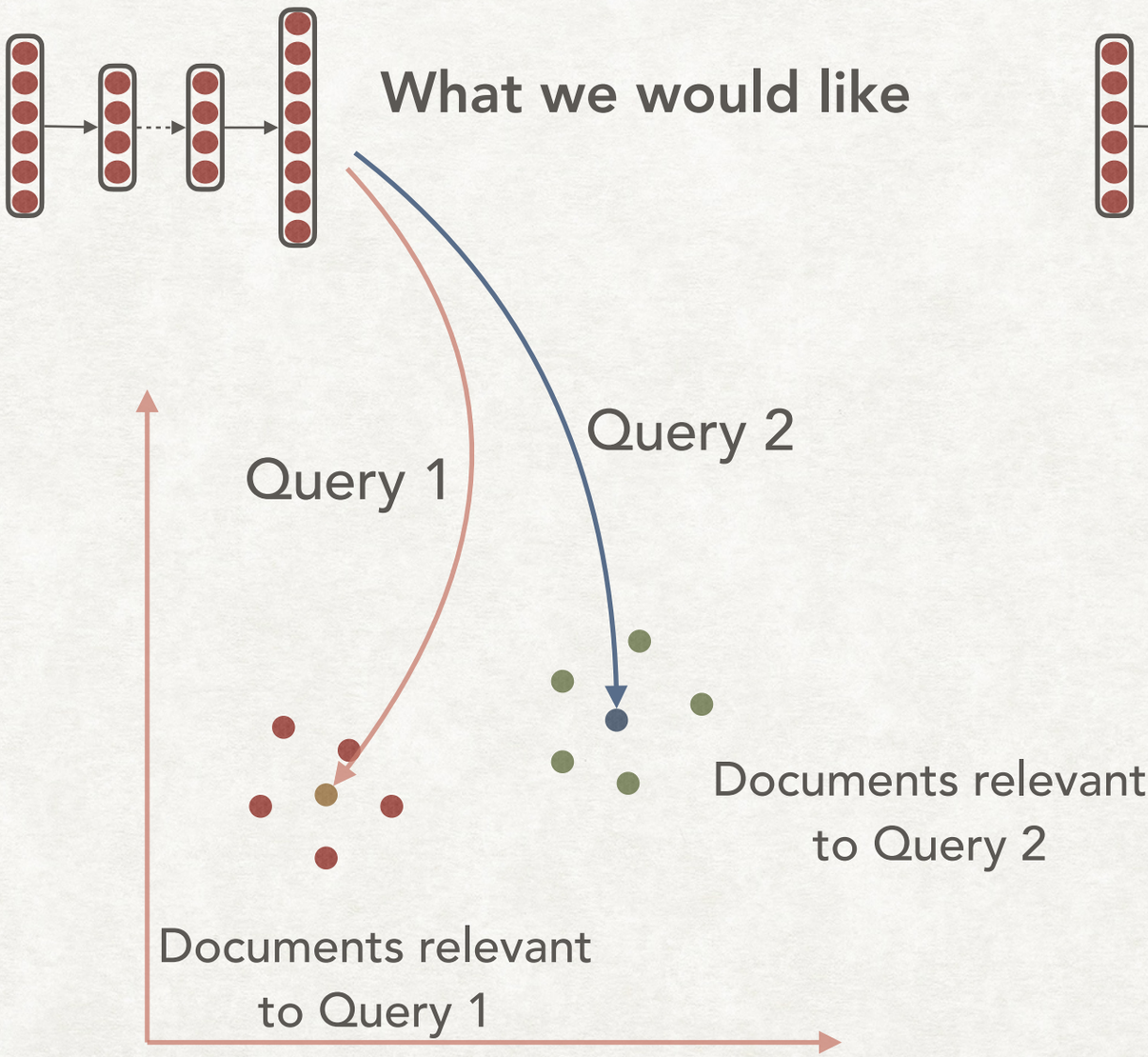
DNN FOR RECOMMENDER SYSTEMS

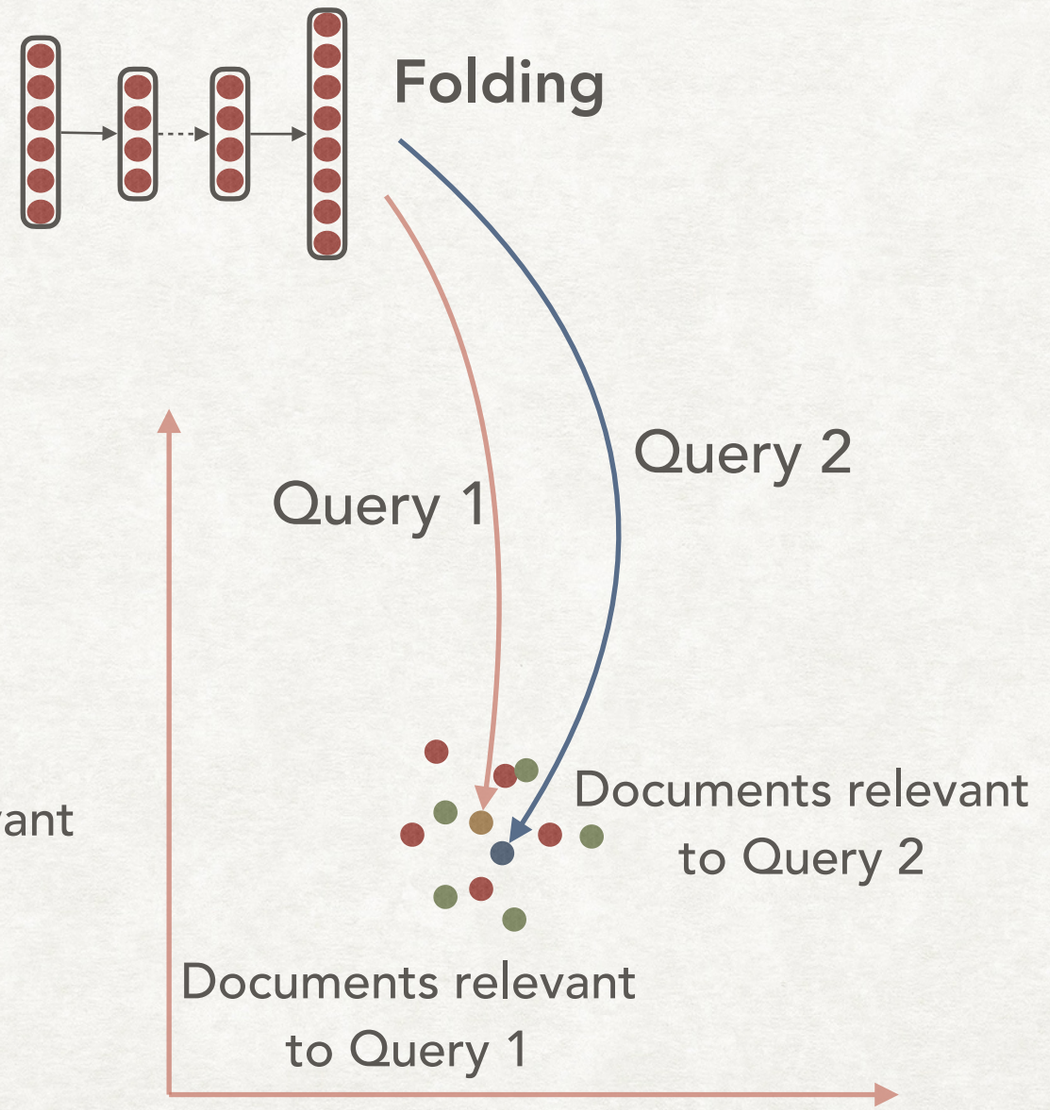
AN EXAMPLE

- How to compute the loss function?
- We might want to consider a function of the difference between $\hat{p}$ (the predicted distribution) and p (the real one)…
- Except that we do not know the entirety of p.
- We can try to compute the gradient only for the positive item of p (the one that the user liked)…
- …but we can have the problem of *folding*.

DNN FOR RECOMMENDER SYSTEMS

AN EXAMPLE





DNN FOR RECOMMENDER SYSTEMS

AN EXAMPLE

- We use *negative sampling.*

- Instead of learning only from positive example we sample a set of irrelevant documents as negative examples.
  We can do it in two ways:

  - Uniform sampling

  - Higher probability of being sampled to items with a large output value. They contribute more to the gradient.

DNN FOR RECOMMENDER SYSTEMS

ADVANTAGES AND DISADVANTAGES

- DNN can easily incorporate additional features for personalisation.
- DNN can adapt to new queries.
- DNN are more difficult to scale to handle a very large corpus.
- WALS is less prone to folding than DNN.
- The item embeddings (weights of the last layer) can be stored, but the query embedding (output of all layers but the last) must be re-computed every time.

WHERE TO GO NEXT

SWIRL WORKSHOP

STRATEGIC WORKSHOP ON IR IN LORNE

- Workshop started in 2004, with second edition in 2012 and third in 2018.

- The first edition produced a list of “recommended reading” for research students in IR: https://people.eng.unimelb.edu.au/ammoffat/swirl2004/homework-forum.pdf

- The second edition has a list of interesting papers with a brief description of their content and why they are interesting: http://www.cs.rmit.edu.au/swirl12/discussion.php

- Website for SWIRL 3: https://sites.google.com/view/swirl3/