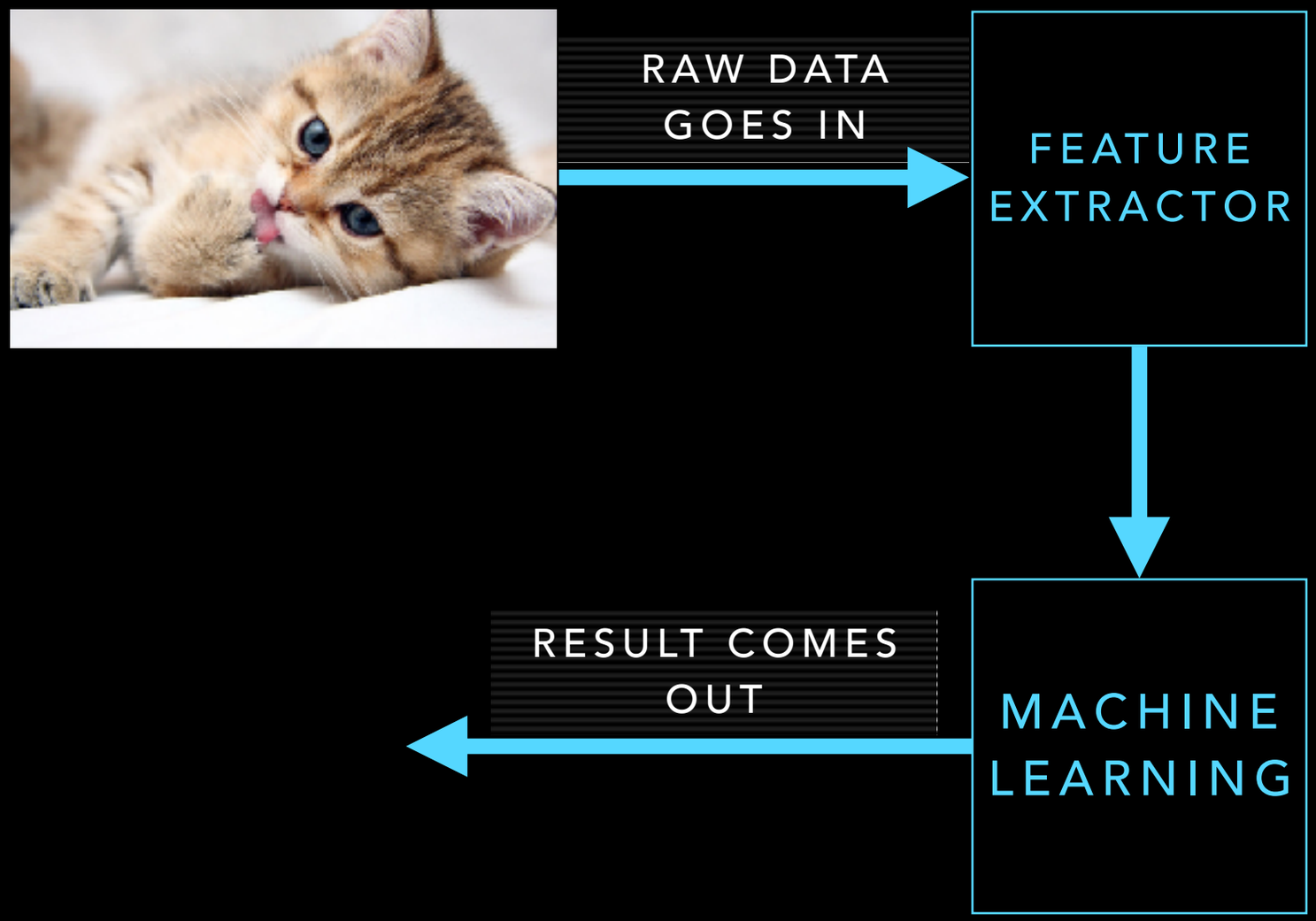
FEATURE EXTRACTION

Many applications involving machine learning take the following form:



Many applications involving machine learning take the following form:

RAW DATA GOES IN

FEATURE EXTRACTOR

THIS IS WHERE *INTERESTING* PROCESSING HAPPENS

MACHINE LEARNING

RESULT COMES OUT

Many applications involving machine learning take the following form:

RAW DATA GOES IN

FEATURE EXTRACTOR

FEATURE EXTRACTORS MAKE FEATUREVECTORS FROM DATA

MACHINE LEARNING

RESULT COMES OUT

Many applications involving machine learning take the following form:

RAW DATA GOES IN

FEATURE EXTRACTOR

MACHINE LEARNING SYSTEM USES FEATURE VECTORS TO MAKE INTELLIGENT DECISIONS

MACHINE LEARNING

RESULT COMES OUT

CHOOSING GOOD FEATUREVECTOR REPRESENTATIONS FOR MACHINE-LEARNING

- Choose features which allow to distinguish objects or classes of interest
  - Similar within classes
  - Different between classes
- Keep number of features small
  - Machine-learning can get more difficult as dimensionality of featurespace gets large

BINARY FEATURES

- For binary features we can build binary vectors [0,1,1,0,…]
- Each element corresponds to something that can be measured in the original data (and is either present or absent)

CATEGORICAL FEATURES

- How can we encode features that can take one value from a set of N>=2 choices?
  - We could choose to represent each value with a numeric index {A->0, B->2, C->3,…}
  - But this is perhaps not the best option unless we believe that A is closer to B than A is to C

- Solution: One Hot Encoding
  - Map each value to a vector in which the values are all 0, except for a single element with a 1 in the position that represents that value.
  - For example, taking the mapping {A->0, B->2, C->3}, the value “B” can be encoded as [0,1,0]
- All encodings are equally as distant from each other
  - We can think of this encoding as an idealised probability distribution when the possible values are independent of each other

ASIDE: SIMILARITY

- What about if we have prior knowledge about similarities?
- For example, “Dalmatian” is closer to “Poodle” than it is to “Horse”
- There are alternative embedding strategies that allow this to be exploited….

BAGS OF *WORDS*

The *bag* data structure

- A bag is an **unordered** data structure like a *set*, but which unlike a set allows elements to be **inserted multiple times**.
  - sometimes called a *multiset* or a *counted set*

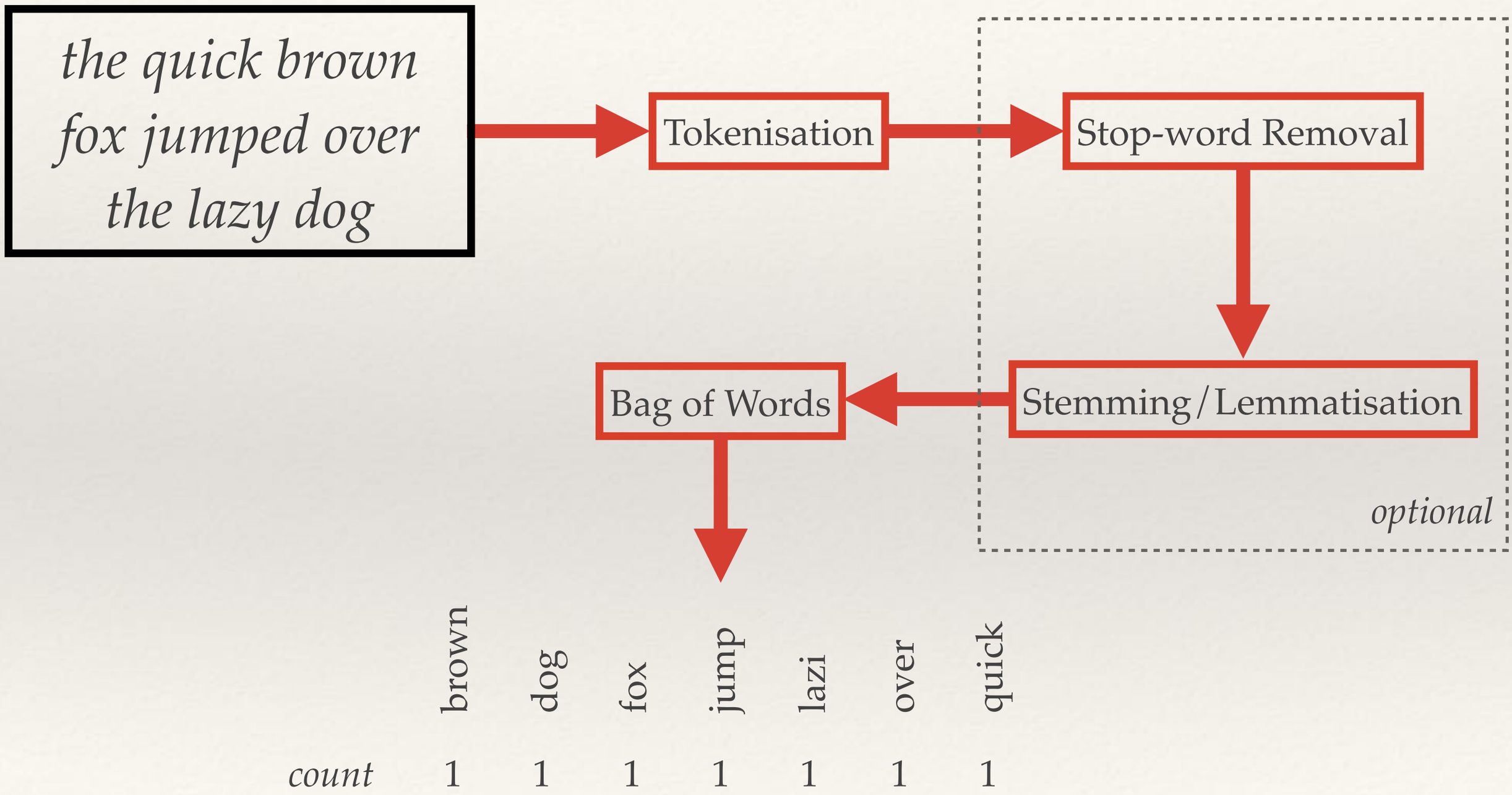
Bag of Words

A document

the quick brown fox jumped over the lazy dog

The bag of words describing the document

Text processing (feature extraction)



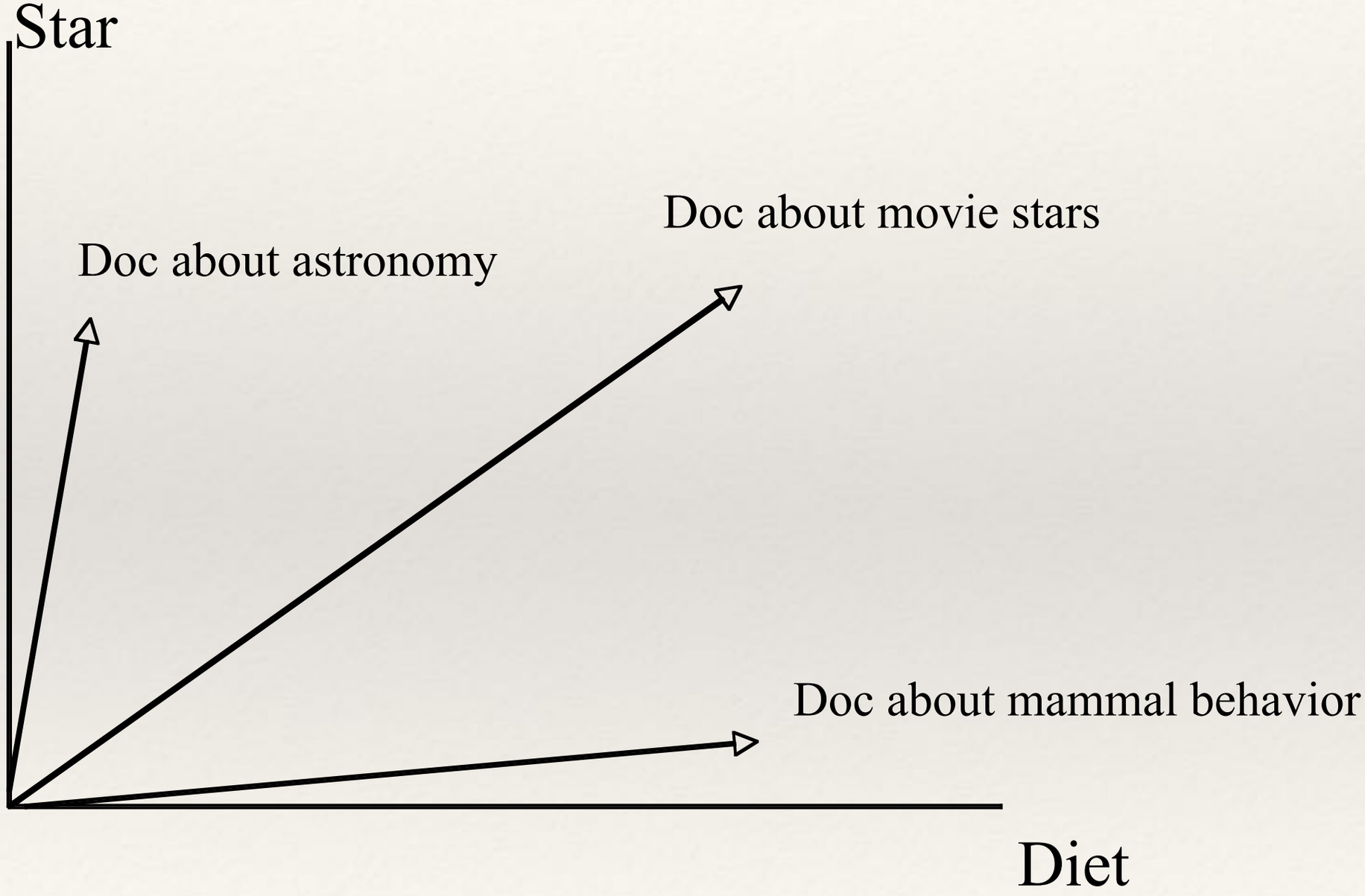
The Vector-Space Model

- Conceptually simple:
  - Model each document by a vector
  - Model each query by a vector
  - Assumption: documents that are “close together” in space are similar in meaning.
  - Use standard similarity measures to rank each document to a query in terms of decreasing similarity

Bag of Words Vectors

- The lexicon or vocabulary is the **set** of all (processed) words across all documents known to the system.
- We can create vectors for each document with as many dimensions as there are words in the lexicon.
- Each word in the document's bag of words contributes a count to the corresponding element of the vector for that word.
  - In essence, each vector is a histogram of the word occurrences in the respective document.
  - **Vectors will have very high number of dimensions, but will be very sparse.**

The Vector-space Model



Weighting the vectors

- The number of times a word occurs in a document reflects the importance of that word in the document.
- Intuitions:
  - A term that appears in many documents is not important: e.g., the, going, come, …
  - If a term is frequent in a document and rare across other documents, it is probably important in that document.

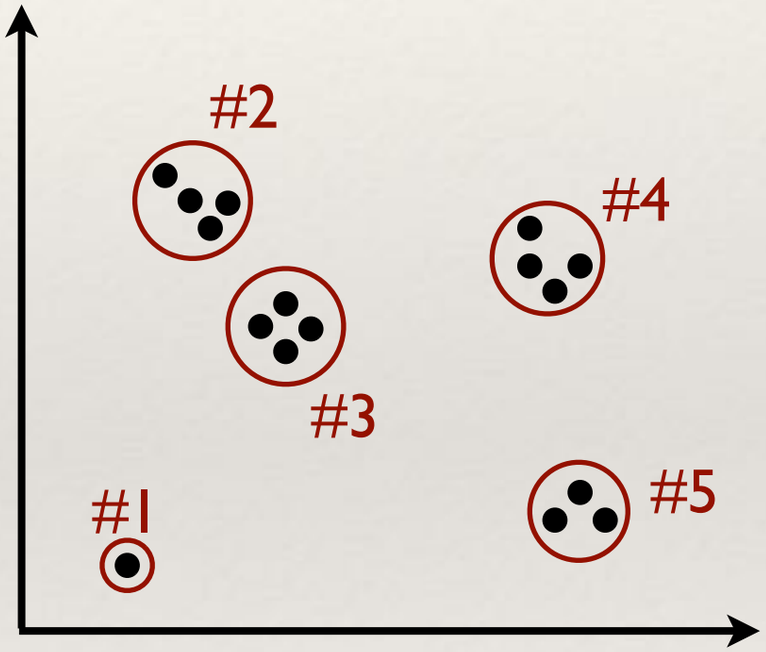
Possible weighting schemes

- Binary weights
  - Only presence (1) or absence (0) of a term recorded in vector.
- Raw frequency
  - Frequency of occurrence of term in document included in vector.
- TF-IDF
  - Term frequency is the frequency count of a term in a document.
  - Inverse document frequency (idf) provides high values for rare words and low values for common words.

Vector Quantisation

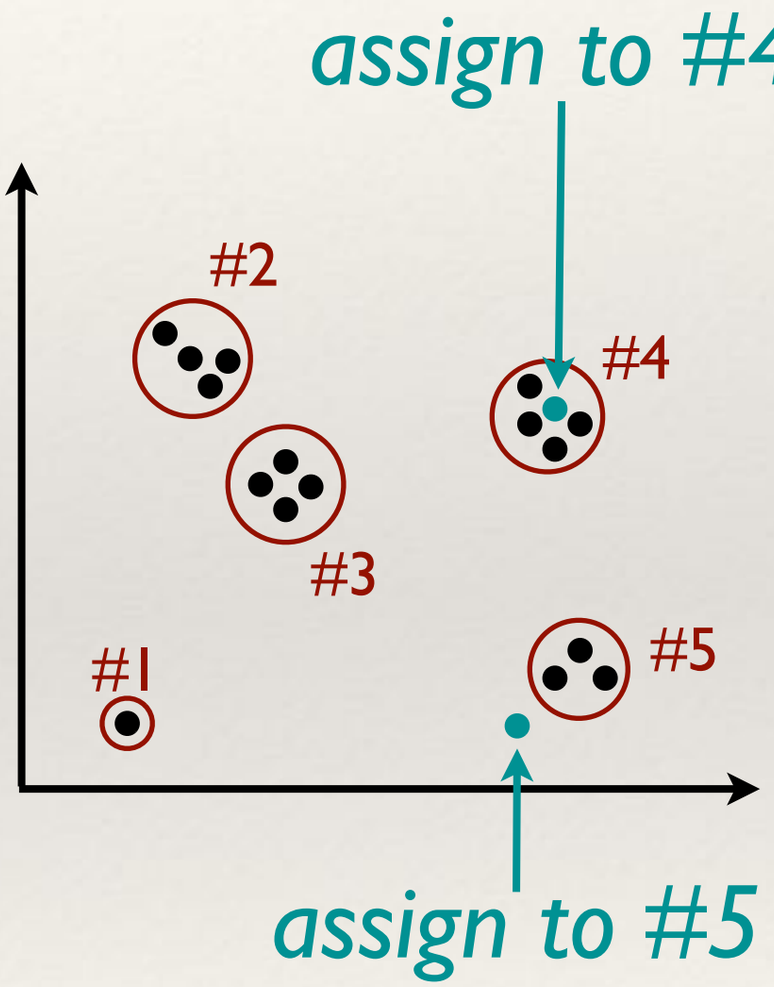
Learning a Vector Quantiser

- Vector quantisation is a lossy data compression technique.
- Given a set of vectors, a technique like K-Means clustering can be used to learn a fixed size set of representative vectors.
  - The representatives are the mean vector of each cluster in k-means
  - The set of representation vectors is called a **codebook**



Vector Quantisation

- Vector quantisation is achieved by representing a vector by another approximate vector, which
is drawn from a pool of representative vectors.
- Each input vector is assigned to the “closest” vector from the pool.



Visual Words

(note that the following can equally apply to audio or other modalities of data)

SIFT Visual Words

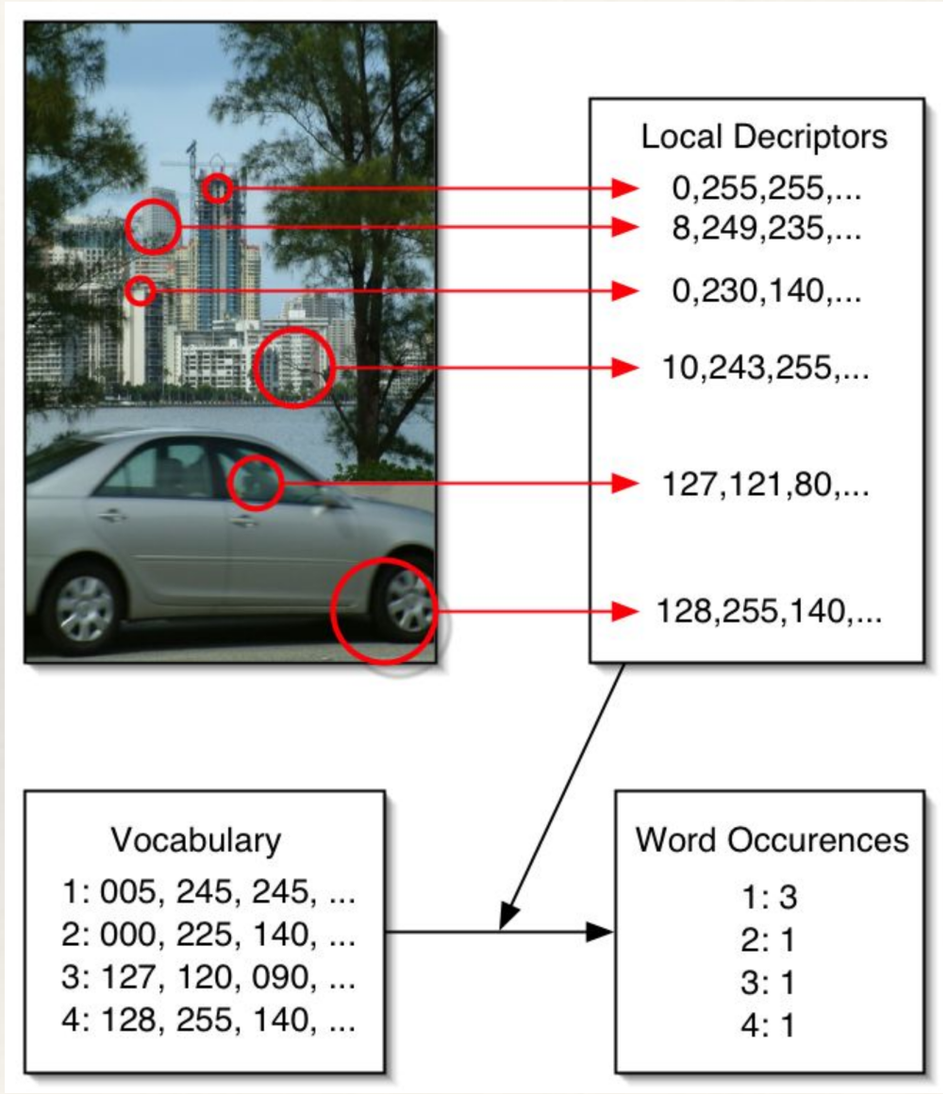
- We can vector quantise SIFT descriptors (or any other local feature)
  - Each descriptor is replaced by a representative vector known as a **visual word**
    - In essence the *visual word* describes a small image patch with a certain pattern of pixels
    - In many ways the process of applying vector quantisation to local features is analogous to the process of stemming words.
- The codebook is the visual equivalent of a lexicon or vocabulary.

Bags of Visual Words

- Once we've quantised the local features into visual words, they can be put into a bag.
- This is a **Bag of Visual Words (BoVW)**
- We're basically ignoring where in the image the local features came from (including ignoring scale)

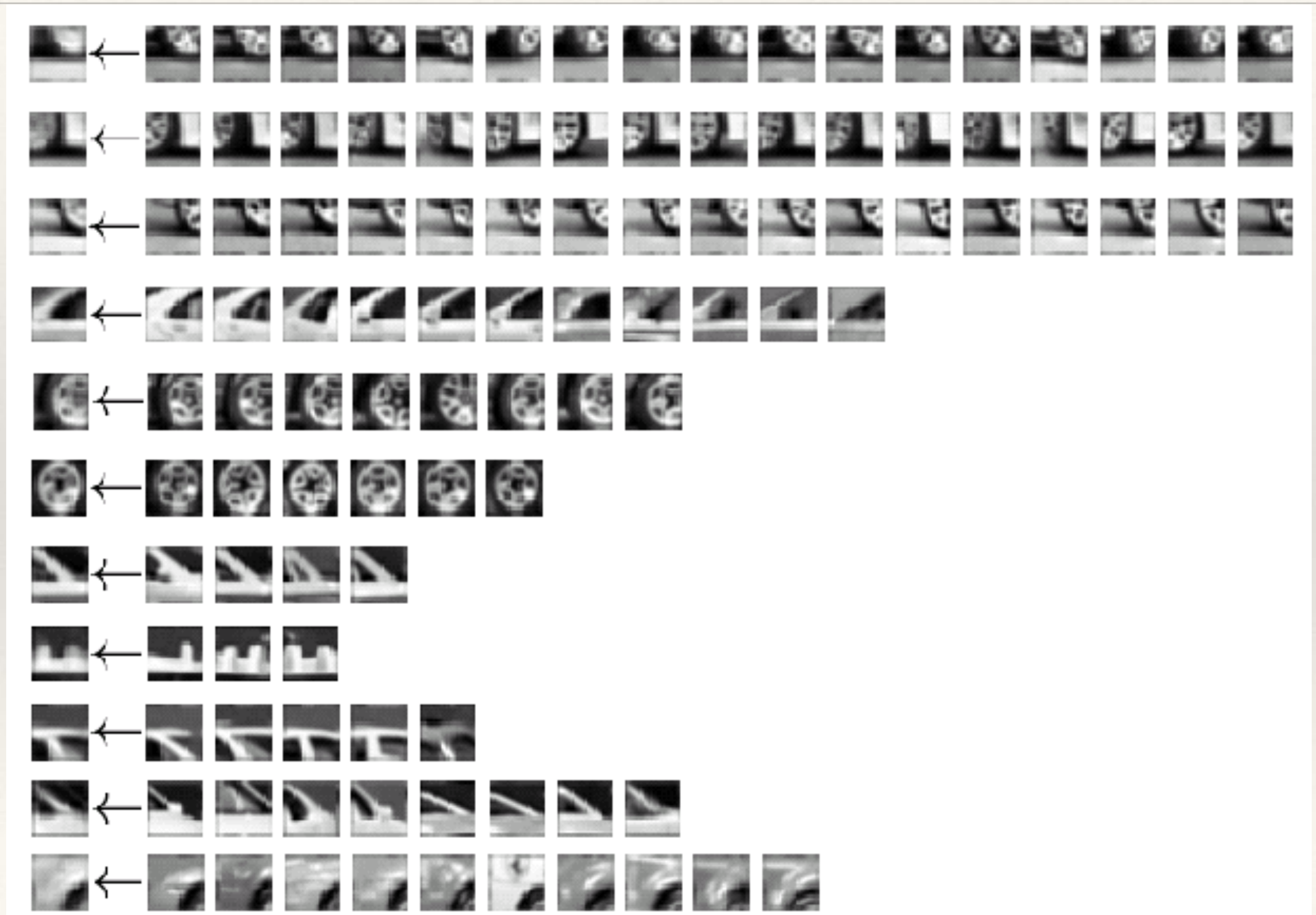
Histograms of Bags of Visual Words

- Like in the case of text, once we have a BoVW and knowledge of the complete vocabulary (the codebook) we can build histograms of visual word occurrences!
- This is rather nice… it gives us a way of aggregating a variable number of local descriptors into a fixed length vector.
  - Useful for machine learning
  - But also allows us to apply techniques for text retrieval to images



Demo: SIFT visual word histogram

Visualising Visual Words



The effect of codebook size

- There is one **key parameter** in building visual words representations - **the size of the vocabulary**.
  - Too small, and all vectors look the same
    - Not distinctive
  - Too big, and the same visual words might never appear across images
    - Too distinctive

Feature Engineering: Improved Text Features

- The simple BOW type features we looked at earlier are OK, but can we do better?
  - Feature engineering

Fields

- Emails have structure:
  - sender; to; cc
  - title; subject; body
  - etc
- Can we use these to create features?
  - e.g. introduce the sender as a feature…

Virtual features

- SPAM emails might have language features that exhibit certain characteristics
  - THEY MIGHT CONTAIN LOTS OF SHOUTING!!!
- Could engineer a virtual feature to measure this
  - If more than 30% of words are uppercase then record the presence of a *virtual* "uppercase" feature

	money	viagra	dad	mom	dinner	mail	v.Uppercase	…
SPAM	1	15	0	2	1	8	5	
HAM	0	0	3	3	8	2	0	

Word Context: N-Grams

- Rather than looking at individual words, could use pairs of words or triplets of words as the base feature

  - More generally called *n-grams*

ratherthan thanlooking lookingat atindividual individualwords

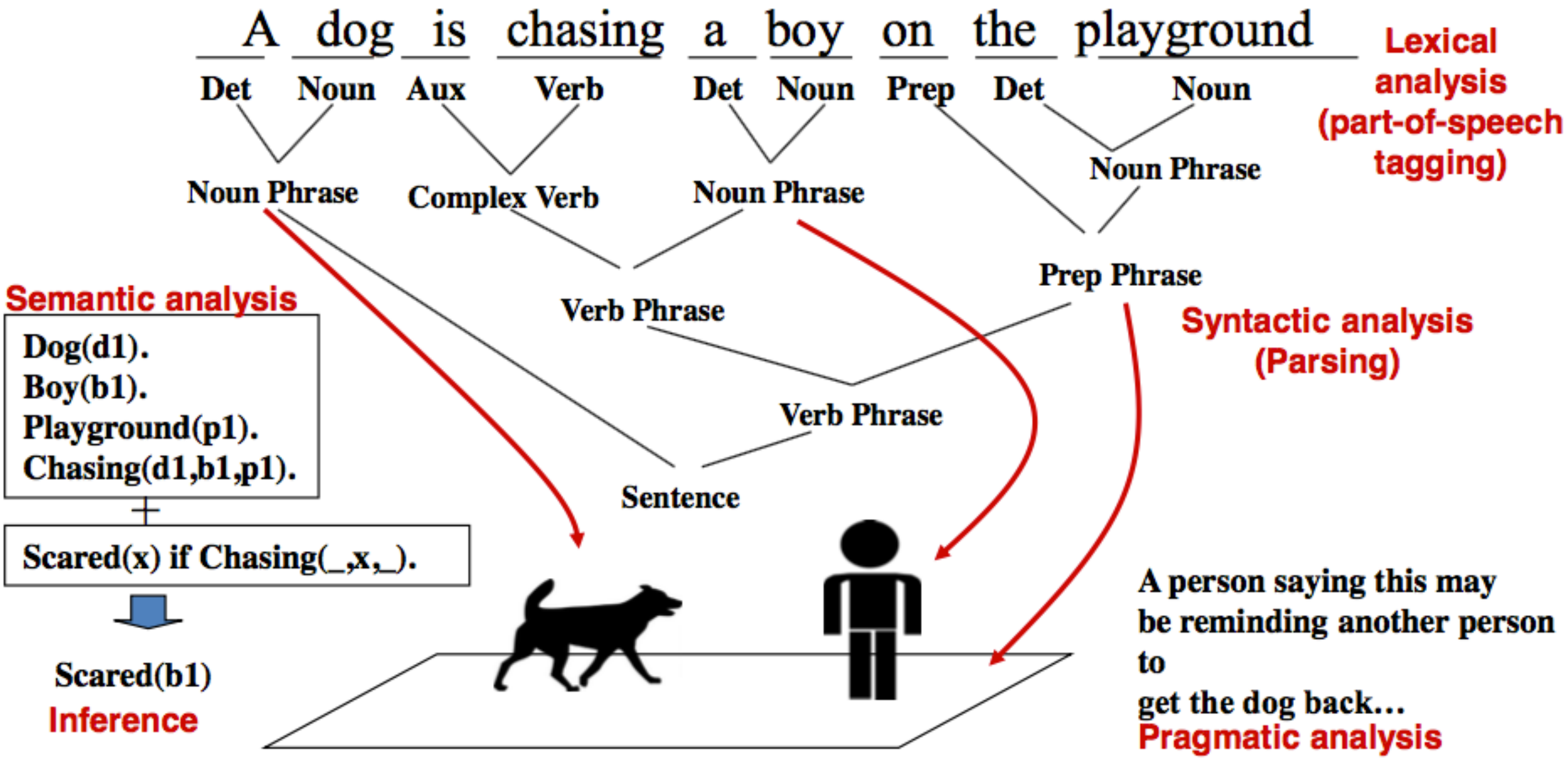
ratherthanlooking thanlookingat lookingatindividual

- Advantages: context capture; names

- Disadvantages: feature explosion

Deeper parsing of text

- Could we more intelligently parse the text to better know what features we should record?
  - *Natural Language Processing*
  - *POS-tagging*
  - *NE-Extraction*



From the slides by ChengXiang “Cheng” Zhai at UIUC
https://class.coursera.org/textretrieval-001

NLP is hard!

- POS tagging
  - *State-of-the-art: Close to 97% accuracy for English*
  - “He turned off the motorway.” vs “He turned off the PC.”
- General complete parsing
  - “A man saw a boy with a telescope.”
- **Robust NLP tends to be *shallow***

Named Entities

Jim bought 300 shares of Acme Corp. in 2006.

[Jim]$_{\text{Person}}$ bought 300 shares of [Acme Corp.]$_{\text{Organisation}}$ in [2006]$_{\text{Time}}$.

Could we use this as a basis for better features?

Embedding

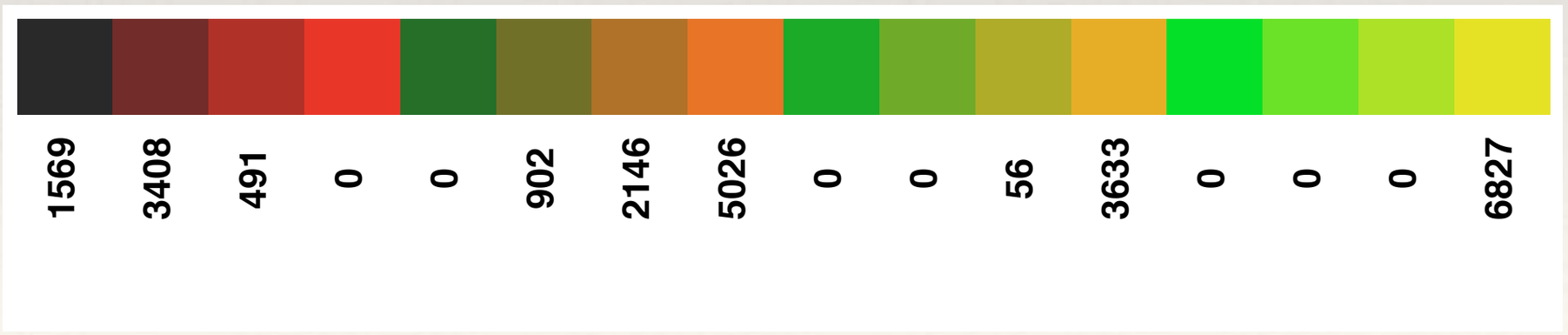
- Learn a vector representation of words (or documents)
- Use the “distributional hypothesis” - words with similar meanings tend to occur in similar contexts

- Latent Semantic Analysis
  - Family of techniques for embedding documents
- Learned embeddings, word2vec & GloVe
  - Modern techniques for learning word similarity

Over time the features used to create BoVW representations have improved

Early global colour visual terms

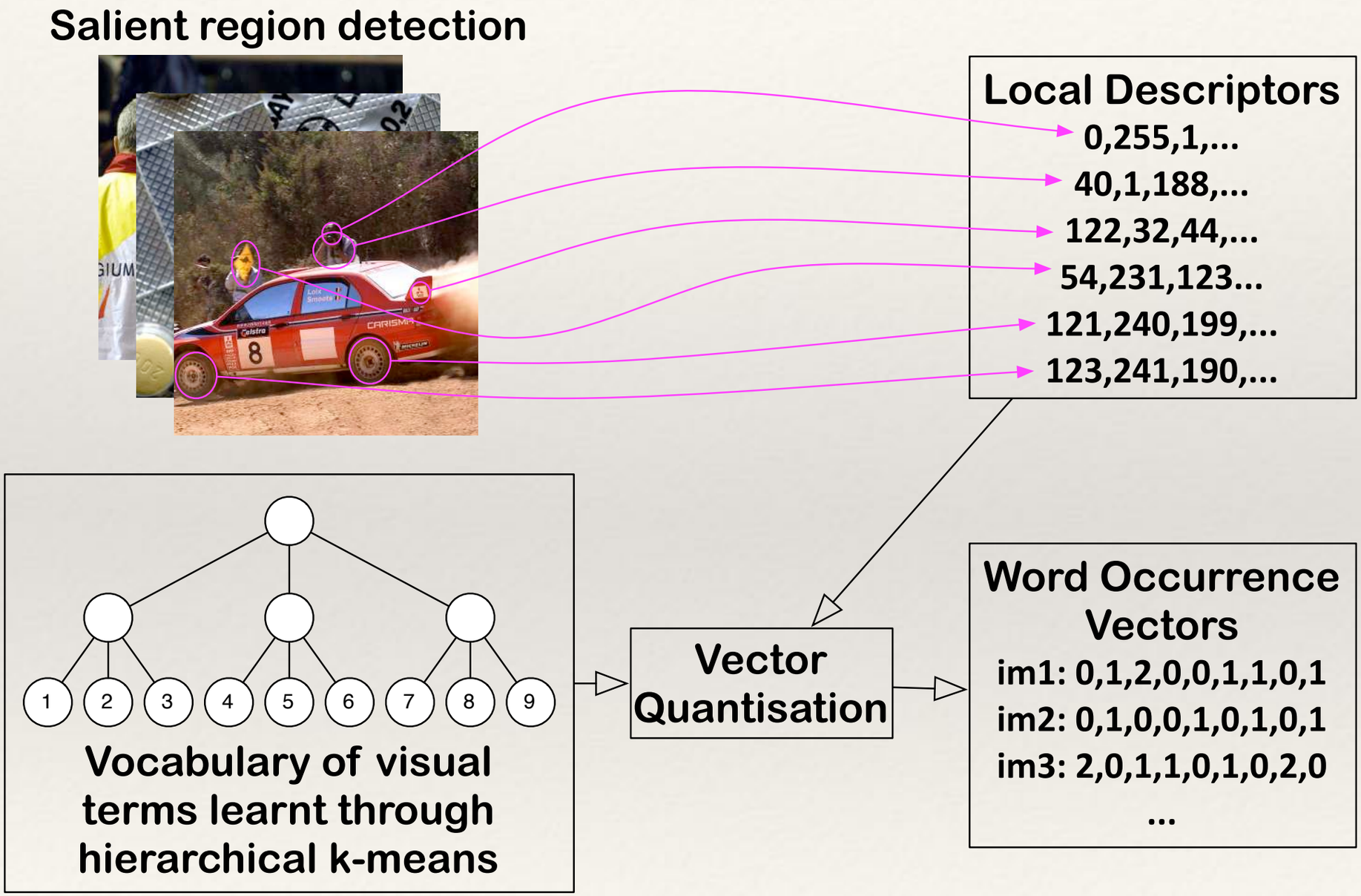
- Consider each pixel as a visual word based on the quantisation of its colour to a discrete set of values.
- The BoVW Histogram is just a joint colour histogram that we saw earlier



Visual words from regions/segments

[1 2 0 0 6]

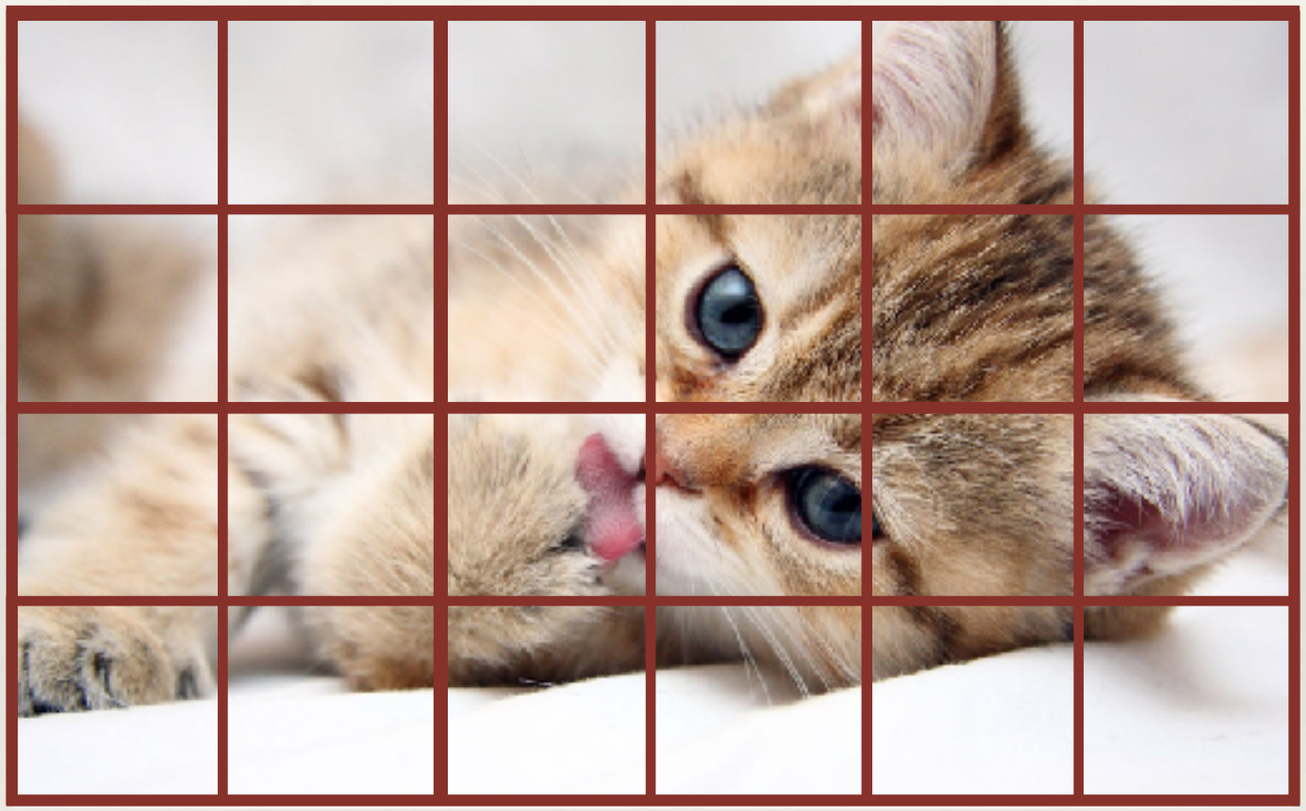
Visual words from interest points



Local features extracted around interest points work okay for classification, but there are more recent strategies that can work better…

densely sampled features

Dense Local Image Patches



Dense SIFT

Rather than extracting your SIFT features at DoG interest points, you could extract them across a dense grid - this gives much more coverage of the entire image.

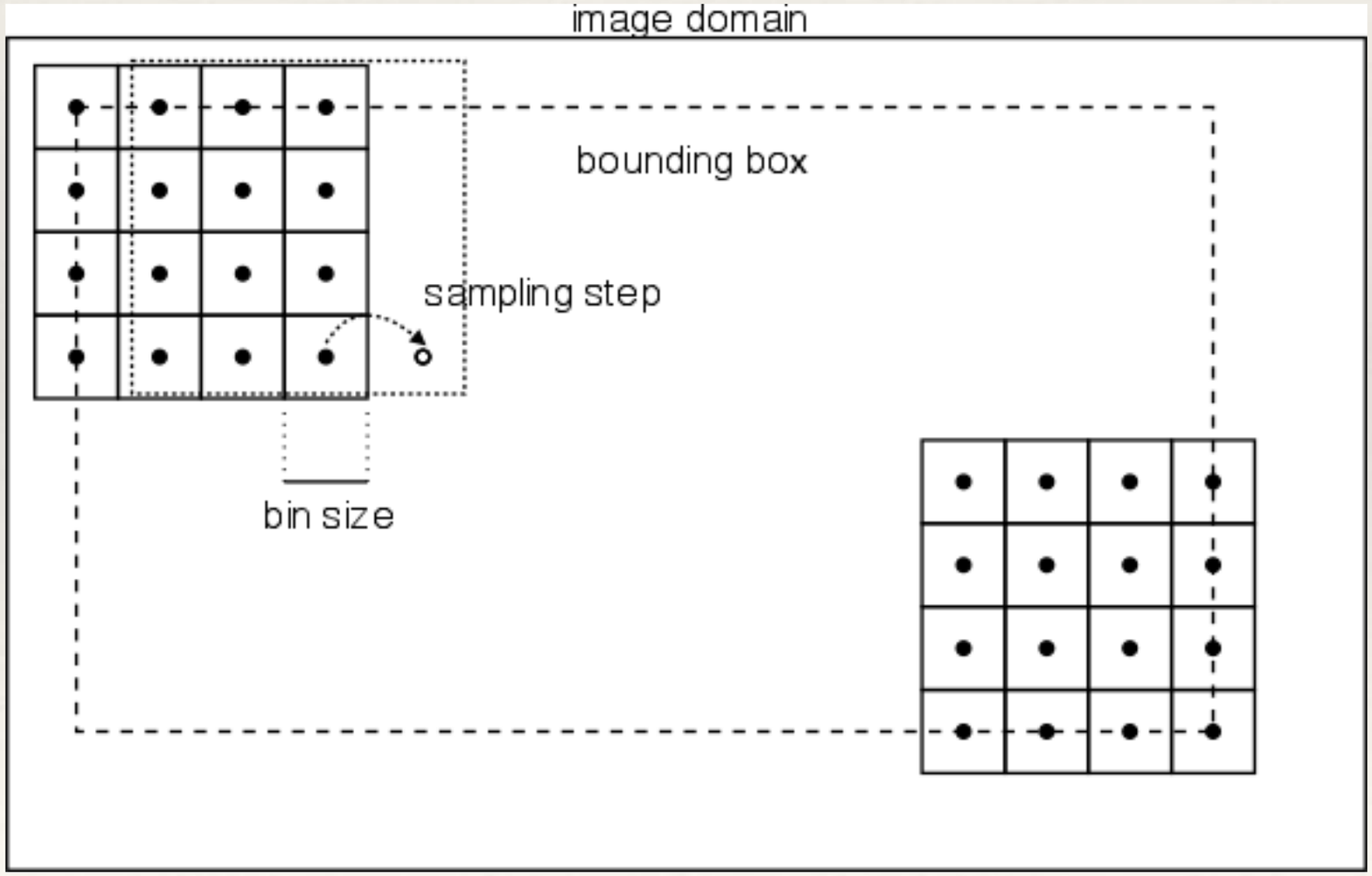
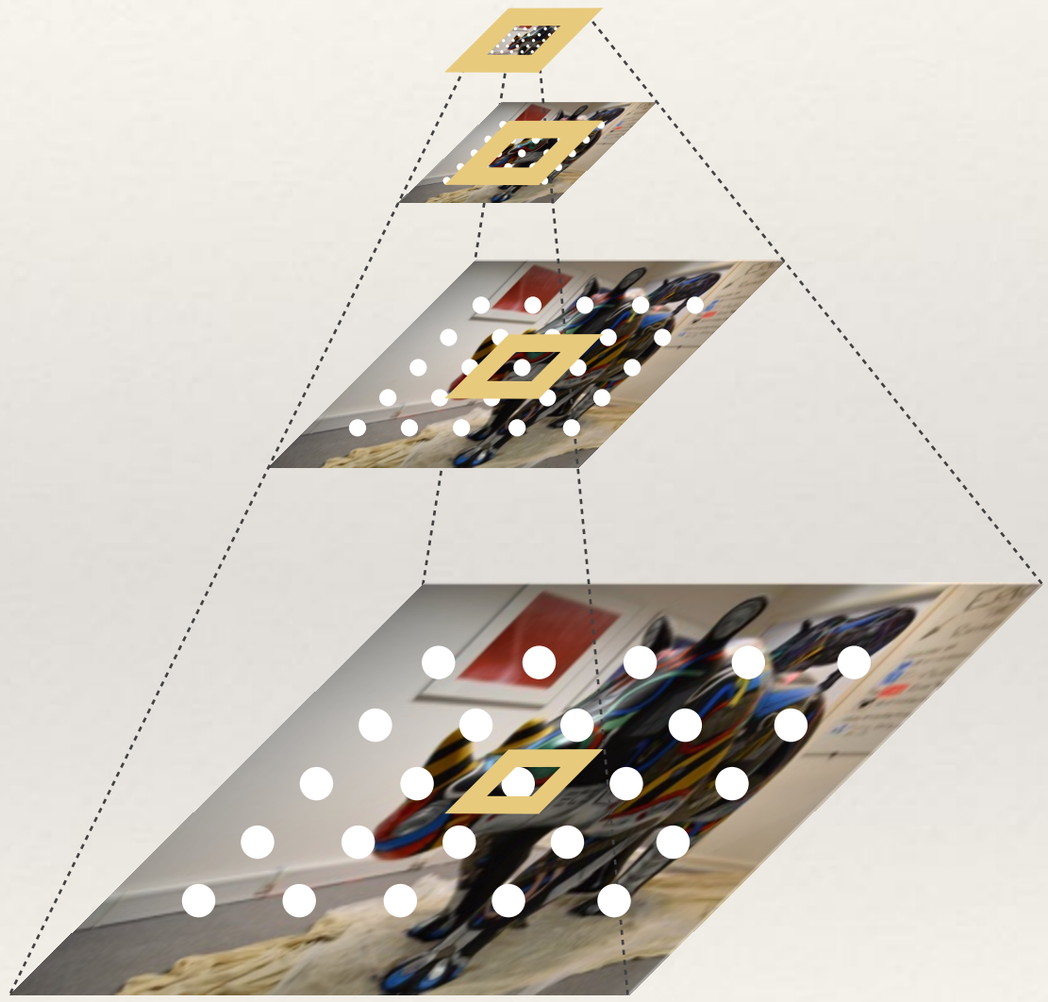


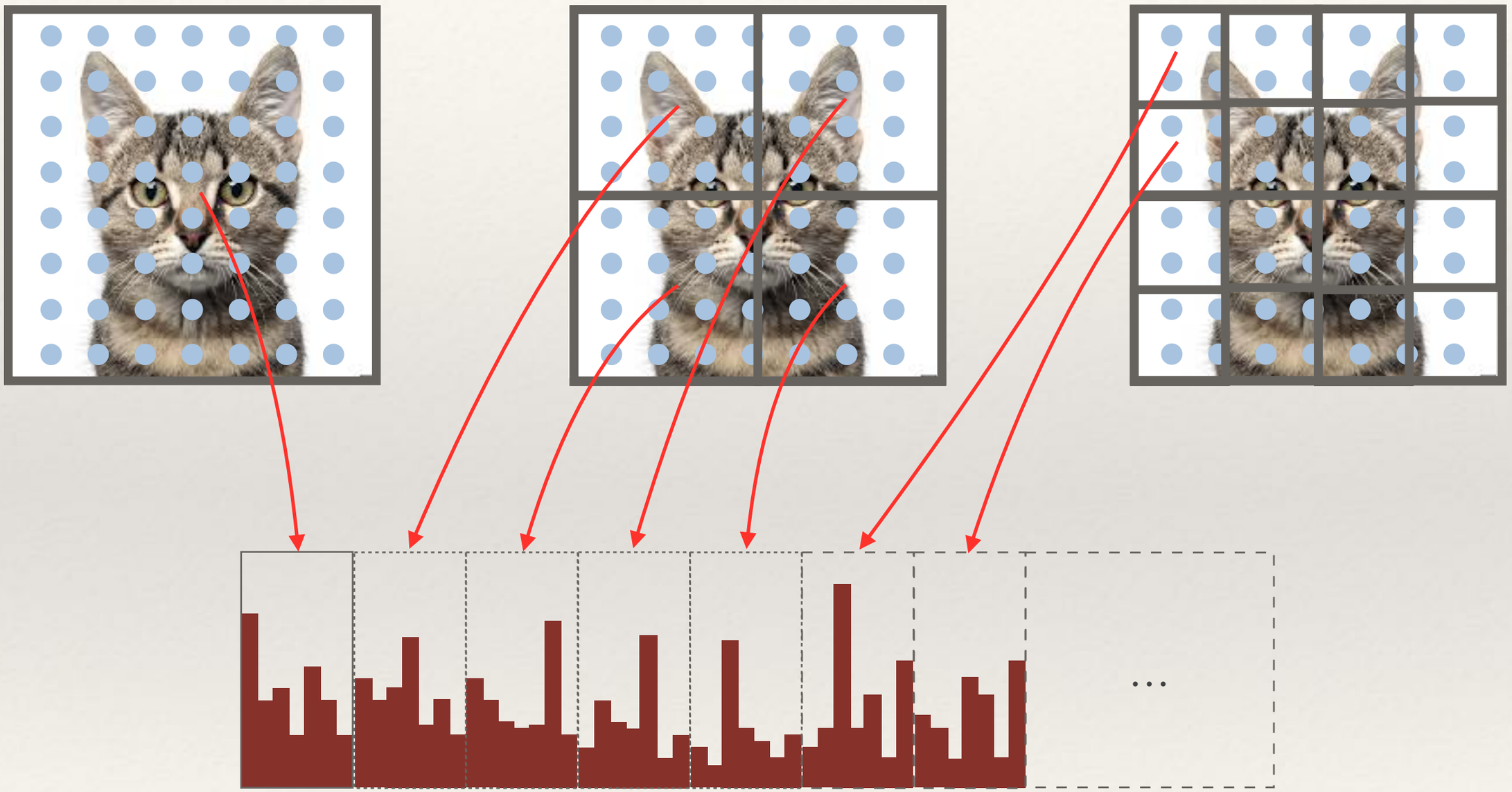
Image from http://www.vlfeat.org/api/dsift.html

Pyramid Dense SIFT

- For even better performance and coverage, you can sample in a Gaussian pyramid
- Note that the sampling region is a fixed size, so at higher scales you sample more content



Spatial Pyramids



PHOW: Pyramid Histogram of Words = Hist(VQ(Pyramid Dense SIFT)) + Spatial Pyramid

All this feature-engineering seems to be based on a lot of heuristics…

Is there an alternative?
Is it better?