Machine Learning for NLP

Readings in unsupervised Learning

Aurélie Herbelot

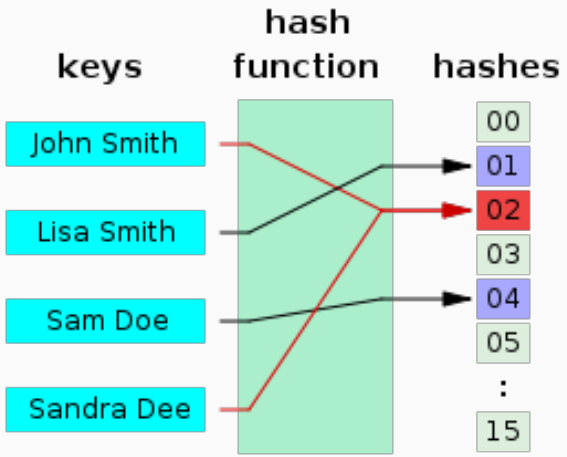
2018

Centre for Mind/Brain Sciences
University of Trento

Hashing

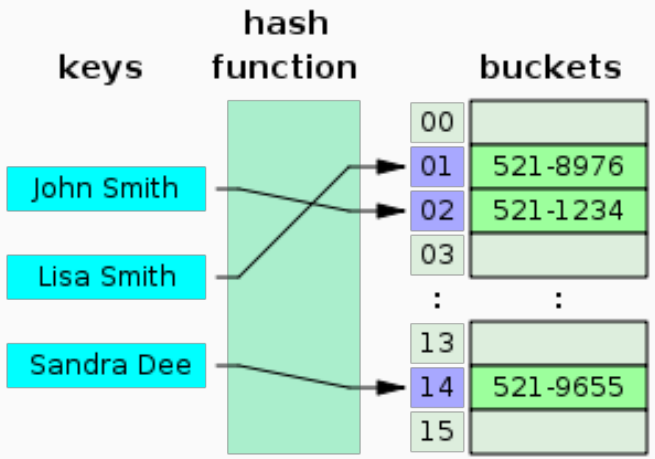
Hashing: definition

- Hashing is the process of converting data of arbitrary size into fixed size signatures.
- The conversion happens through a *hash function*.
- A *collision* happens when two inputs map onto the same *hash (value).*
- Since multiple values can map to a single hash, the slots in the hash table are referred to as *buckets*.



https://en.wikipedia.org/wiki/Hash_function

Hash tables



By Jorge Stolfi - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=6471238

Fixed size?

- **A bit:** a binary unit of information. Can take two values (0 or 1, or *True* or *False*).
- **A byte:** (usually) 8 bits. Historically, the size needed to encode one character of text.
- **A hash of fixed size:** a signature containing a fixed number of bytes.

Hashing in cryptography

- Let's convert a string S to a hash V. The hash function should have the following features:
  - whenever we input S, we always get V;
  - no other string outputs V;
  - S should not be retrievable from V.

Hashing in NLP

- **Finding duplicates documents:** hash each document. Once all documents have been processed, check whether any bucket contains several entries.
- **Random indexing:** a less-travelled distributional semantics method (more on it today!)

Hashing strings: an example

- An example function to hash a string s:

$$s[0] * 31^{n-1} + s[1] * 31^{n-2} + ... + s[n-1]$$

  where $s[i]$ is the ASCII code of the ith character of the string and n is the length of s.
- This will return an integer.

Hashing strings: an example

- An example function to hash a string s:

$$s[0] * 31^{n-1} + s[1] * 31^{n-2} + ... + s[n-1]$$

- **A test:** 65 32 84 101 115 116 Hash: 1893050673
- **a test:** 97 32 84 101 115 116 Hash: 2809183505
- **A tess:** 65 32 84 101 115 115 Hash: 1893050672

Modular hashing

- Modular hashing is a very simple hashing function with high risk of collision:

$$h(k) = k \ mod \ m$$

- Let's assume a number of buckets $m = 100$:
  - h(A test) = h(1893050673) = 73
  - h(a test) = h(2809183505) = 5
  - h (a tess) = h(1893050672) = 72
- Note: no notion of similarity between inputs and their hashes.

Locality Sensitive Hashing (LSH)

Locality Sensitive Hashing

- In ‘conventional’ hashing, similarities between datapoints are not conserved.
- LSH is a way to produces hashes that can be compared with a similarity function.
- The hash function is a projection matrix defining a hyperplane. If the projected datapoint $\vec{v}$ falls on one side of the hyperplane, its hash $h(\vec{v}) = +1$, otherwise $h(\vec{v}) = -1$.

Locality Sensitive Hashing

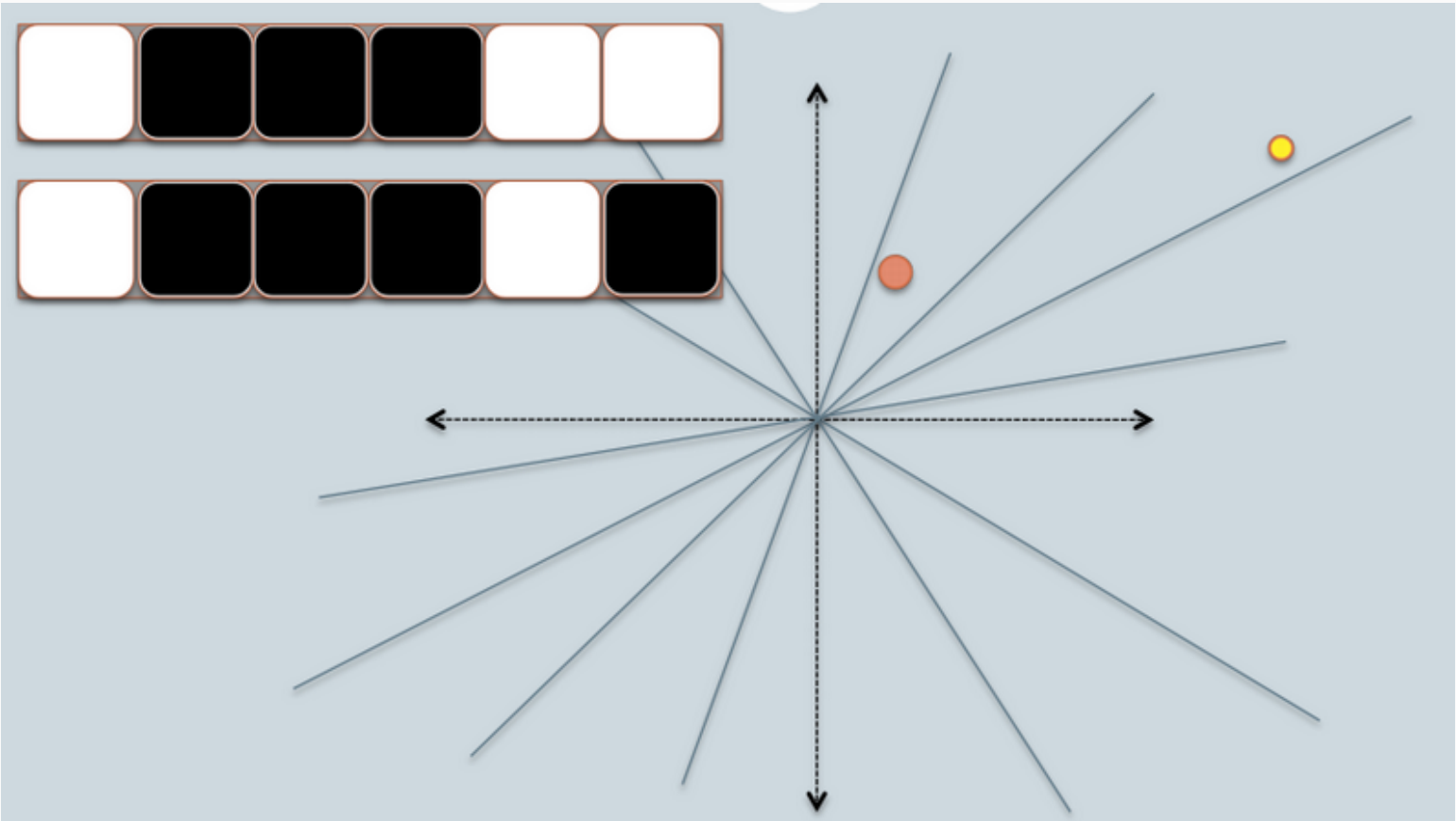


Image from VanDurme & Lall (2010): http://www.cs.jhu.edu/ vandurme/papers/VanDurmeLallACL10-slides.pdf

Locality Sensitive Hashing

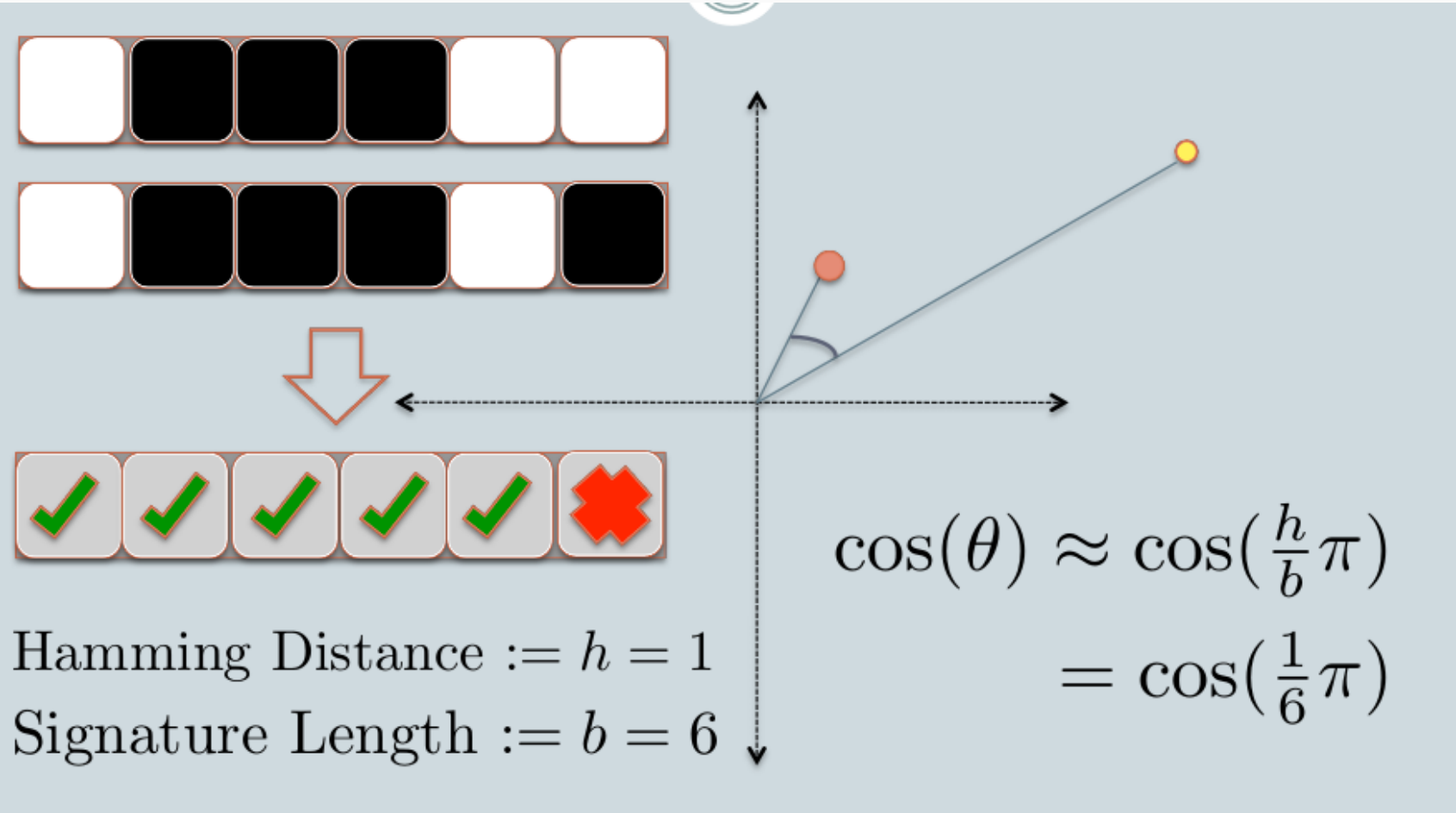


Image from VanDurme & Lall (2010): http://www.cs.jhu.edu/ vandurme/papers/VanDurmeLallACL10-slides.pdf

So what is the hash value?

- The hash value of an input point in LSH is made of all the projections on all chosen hyperplanes.
- Say we have 10 hyperplanes $h_1 ... h_{10}$ and we are projecting the 300-dimensional vector of *dog* on those hyperplanes:
  - dimension 1 of the new vector is the dot product of *dog* and h_1: $\sum dog_i h_{1i}$
  - dimension 2 of the new vector is the dot product of *dog* and h_2: $\sum dog_i h_{2i}$
  - ...
- We end up with a ten-dimensional vector which is the hash of *dog*.

Interpretation of the LSH hash

- Each hyperplane is a discriminatory feature cutting through the data.
- Each point in space is expressed as a function of those hyperplanes.
- We can think of them as new ‘dimensions’ relevant to explaining the structure of the data.

Random indexing

Random projections

- Random projection is a dimensionality reduction technique.
- Intuition (Johnson-Lindenstrauss lemma):

  “If a set of points lives in a sufficiently high-dimensional space, they can be embedded into a space of much lower dimension in such a way that distances between the points are nearly preserved.”

- The hyperplanes of LSH are random projections.

Method

- The original data – a matrix M in d dimensions – is projected into k dimensions, where $k << d$.
- The random projection matrix R is of the shape $k \times d$.
- So the projection of M is defined as:

$$M^{RP}_{k \times N} = R_{k \times d} M_{d \times N}$$

Gaussian random projection

- The random matrix R can be generated via a Gaussian distribution.
- For each row r_k in the original matrix M:
  - Generate a unit-length vector v_k according to the Gaussian distribution such that...
  - v_k is orthogonal to $v_{1\ldots k}$ (to all other row vectors produced so far).

Simplified projection

- It has been shown that the Gaussian distribution can be replaced by a simple arithmetic function with similar results (Achlioptas, 2001).
- An example of a projection function:

$$R_{i,j} = \sqrt{3} \begin{cases} +1 & \text{with probability } \frac{1}{6} \\ 0 & \text{with probability } \frac{2}{3} \\ -1 & \text{with probability } \frac{1}{6} \end{cases}$$

Random Indexing (RI)

- Building semantic spaces with random projections.
- Basic idea: we want to derive a semantic space S by applying a random projection P to co-occurrence counts C:

$$C_{p \times n} \times P_{n \times x} = S_{p \times x}$$

- We assume that $x << n$. So this has in effect dimensionality-reduced the space.

Why random indexing?

- No distributional semantics method so far satisfies all ideal requirements of a *semantics acquisition model*:
  1. show human-like behaviour on linguistic tasks;
  2. have low dimensionality for efficient storage and manipulation ;
  3. be efficiently acquirable from large data;
  4. be transparent, so that linguistic and computational hypotheses and experimental results can be systematically analysed and explained ;
  5. be incremental (i.e. allow the addition of new context elements or target entities).

Why random indexing?

- **Count-models** fail with regard to incrementality. They also only satisfy transparency *without* low-dimensionality, or low-dimensionality *without* transparency.
- **Predict models** fail with regard to transparency. They are more incremental than count models, but not fully.

Why random indexing?

- A random indexing space can be simply and incrementally produced through a two-step process:
  1. Map each *context* item c in the text to a random projection vector.
  2. Initialise each *target* item t as a null vector. Whenever we encounter c in the vicinity of t we update $\vec{t} = \vec{t} + \vec{c}$.
- The method is extremely efficient, potentially has low dimensionality (we can choose the dimension of the projection vectors), and is fully incremental.

Is RI human-like?

- Not without adding PPMI weighting at the end of the RI process... (This kills incrementality.)

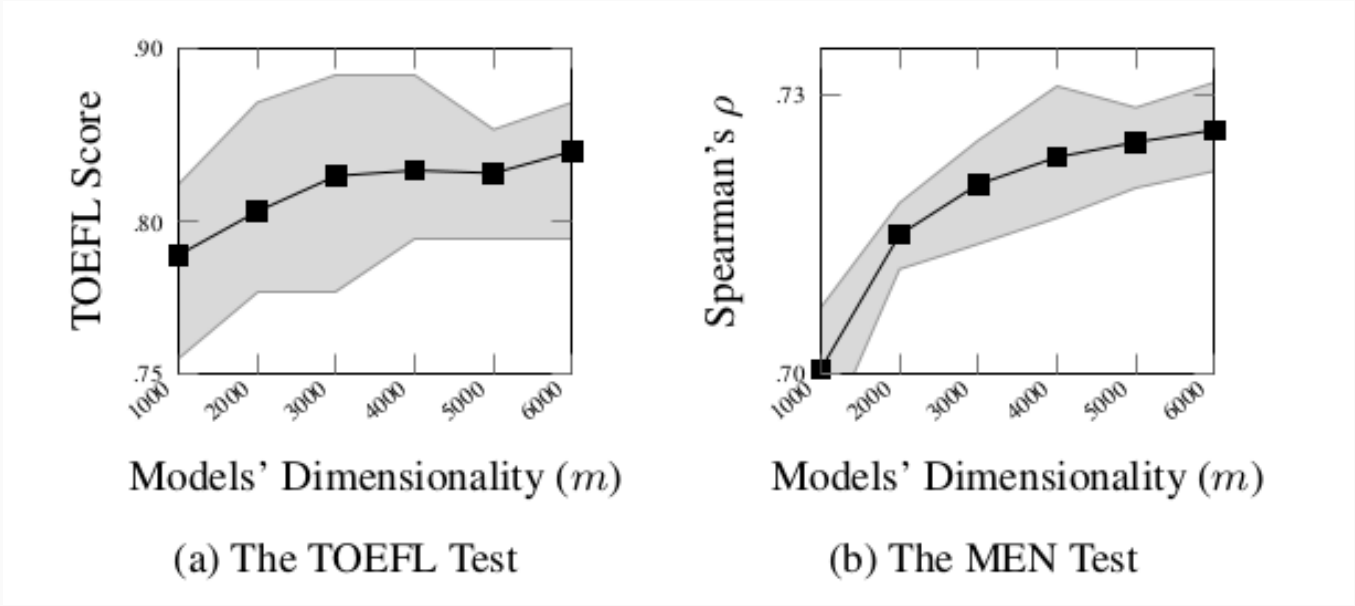
	Context Window Size (a+a)					
Method	2+2	3+3	5+5	7+7	9+9	11+11
OM-raw+γ	0.543	0.541	0.546	0.545	0.545	0.545
OM-PPMI+r	0.712	0.731	0.745	0.745	0.751	0.744
RI	0.331	0.361	0.393	0.422	0.443	0.461
Count-raw	0.326	0.356	0.391	0.419	0.440	0.457
Count-PPMI	0.731	0.748	0.749	0.749	0.750	0.751
W2V-CBOW	0.769	0.757	0.773	0.767	0.769	0.773

TABLE 2: The MEN Test : Various Context Size.

QasemiZadeh et al (2017)

Is RI human-like?

- Not at a particulary low dimensionality...



QasemiZadeh et al (2017)

Is RI interpretable?

- To the extent that the random projections are extremely sparse, we get semi-interpretability.
- **Example:**
  - context *bark* 0 0 0 1
  - context *hunt* 1 0 0 0
  - target *dog* 23 0 1 46

Questions

- What does weighting do that is not provided by RI per se?
- Can we retain the incrementality of the model by not requiring post-hoc weighting?
- Why the need for such high dimensionality? Can we do something about reducing it?

Random indexing in fruits flies

Similarity in the fruit fly

- Living organisms need efficient nearest neighbour algorithms to survive.
- E.g. given a specific smell, should the fruit fly:
  - approach it;
  - avoid it.
- The decision can be taken by comparing the new smell to previously stored values.

Similarity in the fruit fly

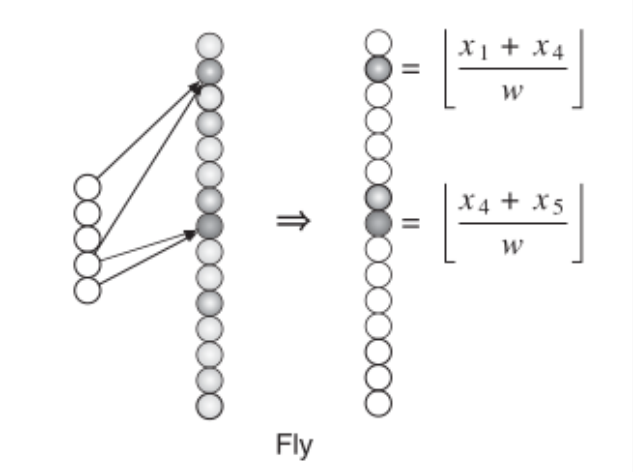
- The fruit fly assigns ‘tags’ to different odors (a signature made of a few firing neurons).
- Its algorithm follows three steps:
  - feedforward connections from 50 Odorant Receptor Neurons (ORNs) to 50 Projection Neurons (PNs), involving normalisation;
  - **expansion** of the input to 2000 Kenyon Cells (KCs) through a sparse, binary random matrix;
  - winner-takes-all (WTA) circuit: only keep the top 5% activations to produce odor tag (hashing).

ML techniques used by the fly

- **Normalisation:** all inputs must be on the same scale in order not to confuse smell intensity with feature distribution.
- **Random projection:** a number of very sparse projections map the input to a *larger-dimensionality* output.
- **Locality-sensitive hashing:** dimensionality-reduced tags for two similar odors should themselves be similar.

More on the random projection

- Each KC sums the activations from ≈ 6 randomly selected PNs.
- This is a *binary* random projection. For each PN, either it contributes activation to the KC or not.



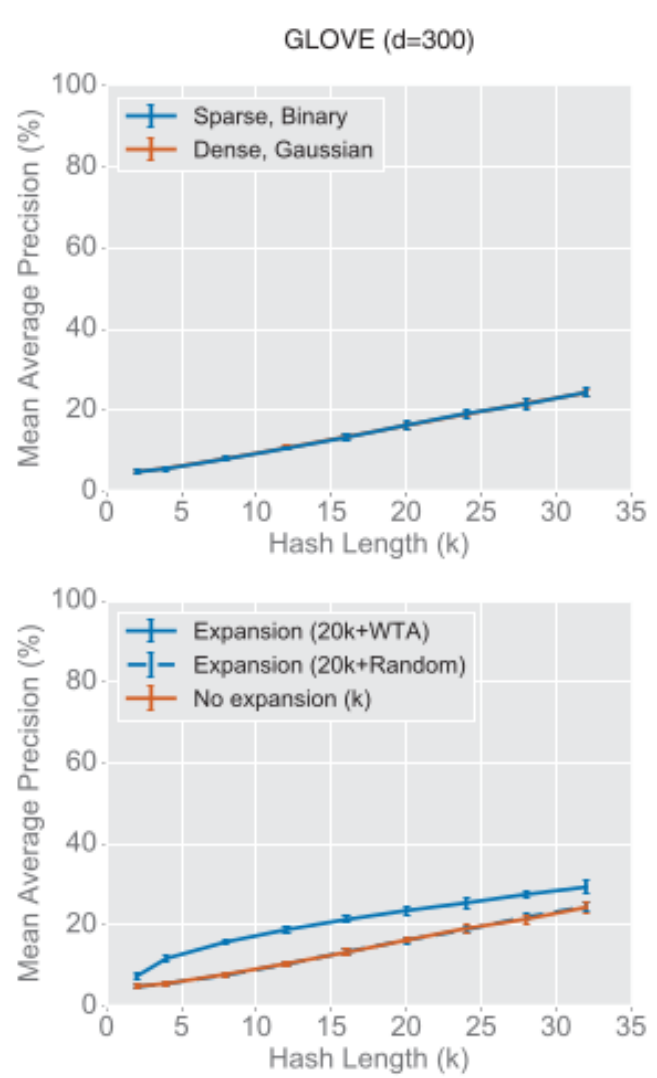
Evaluation

- The fly's algorithm is evaluated on GLOVE distributional semantics vectors.
- For 1000 random words, compare true nearest neighbours to predicted ones.
- Check effect of dimensionality expansion.
- Vary k: the number of KCs used to obtain the final hash.

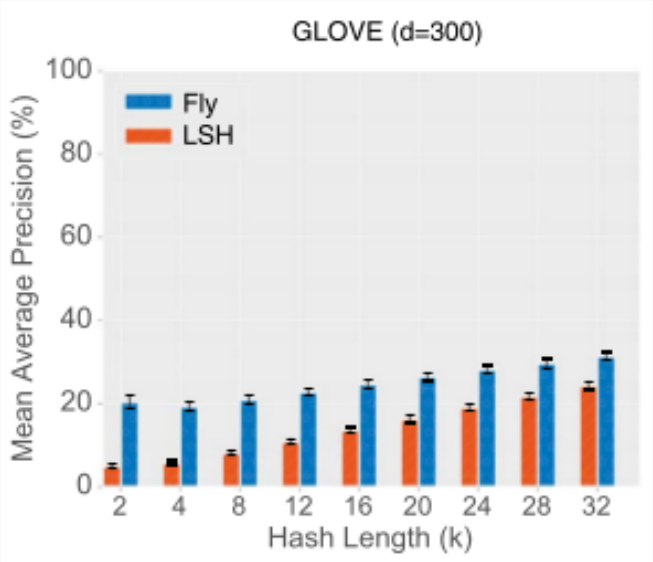
Evaluation

- Evaluation metric: Mean Average Precision (MAP).
- Average Precision (AP) is an Information Retrieval measure. Given a query q to e.g. a search engine, how many relevant documents are retrieved?
- **Example:** query *jaguar animal*. The system returns 80 documents about jaguars (animals), 18 about Jaguars (cars) and 2 about tigers. Then $AP = 80/100 = 0.8$.
- MAP is a mean of APs over many queries.
- Can you see the problem with using MAP for this evaluation?

Results

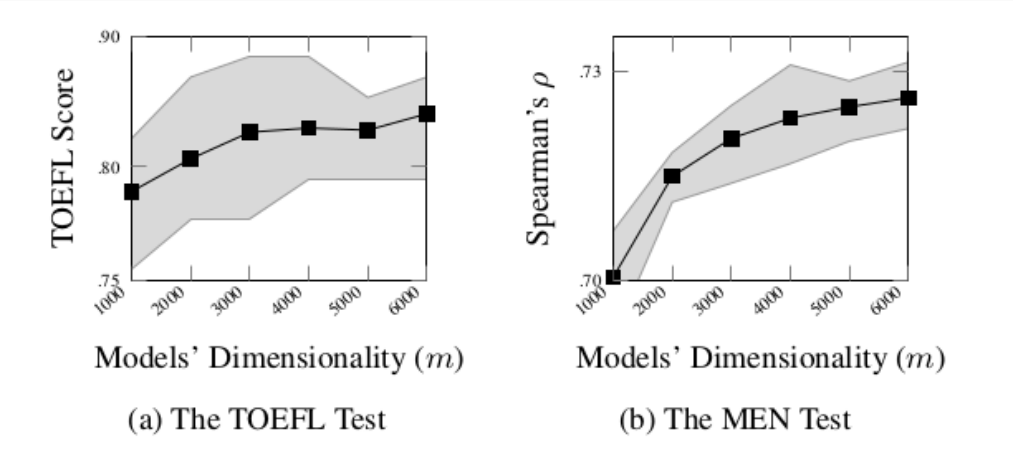

GLOVE (d=300)
Sparse, Binary
Dense, Gaussian
Mean Average Precision (%)
Hash Length (k)
Expansion (20k+WTA)
Expansion (20k+Random)
No expansion (k)
Mean Average Precision (%)
Hash Length (k)

Results


GLOVE (d=300)
Fly
LSH
Mean Average Precision (%)
100
80
60
40
20
0
2
4
8
12
16
20
24
28
32
Hash Length (k)

Comparison with RI

- The gains observed over LSH happen with $\approx 10d$ KCs where d is the dimensionality of the input.
- For the GLOVE vectors, with $d = 300$, the expansion layer has dimensionality 3000.
- This is strikingly similar to our RI results on the MEN dataset:



(a) The TOEFL Test (b) The MEN Test

Comparison with RI

- So how does the fly do dimensionality reduction?
- With the winner-takes-all (WTA) strategy: one sorting operation.
- Note: this is an incrementality-friendly operation. (Compare with SVD, which requires a matrix of datapoints to compute singular values.)

Comparison with RI

- What about weighting? We saw it was essential for good performance.
- At first glance, there is no weighting function in the fly's algorithm.
- But we have the data expansion layer...

Fruit flies do reference??

What PPMI does

- Remember that PMI is a function of the strength of association between two words:

$$\text{pmi}(x;y) \equiv \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)}$$

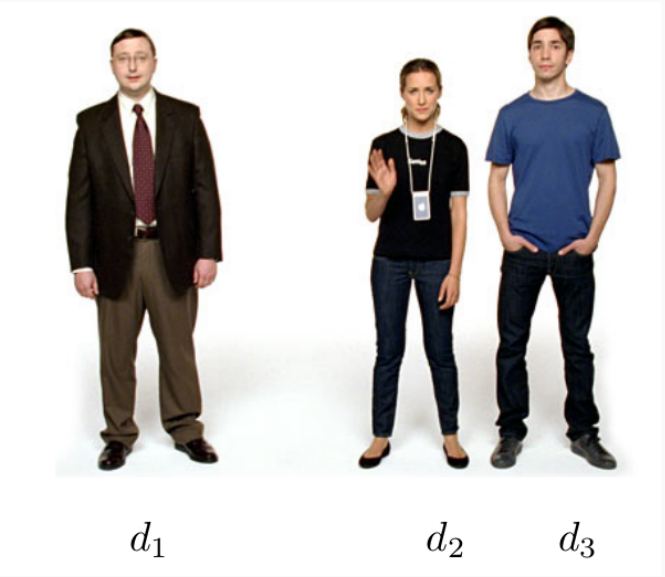
- That is: we are computing how often x and y occur together in comparison to the number of times they occur alone.

What reference does

- Reference is a process by which a *referring expression* is used to select an object out of a *confusion set*.
- Here is an example from the Referring Expression Generation literature:

"the man in the blue T-shirt"

*T*he co-occurrences *man/blue* and *man/T-shirt* identify d_3 successfully (*man* and *T-shirt* on their own are not good signatures for d_3).



Krahmer & van Deemter (2010)

(P)PMI and reference

- When we calculate PMI over word co-occurrences, we are emphasising the idiosyncracies of the target word.
- **Example:** it is idiosyncratic of *cat* that it occurs a lot with *meow* (no other word does that). So the PMI of *cat/meow* is high.
- PMI has a referential effect: it picks out the features that will make sure that a target is *distinguishable* from other words

What the fly does

- The fly *expands* its input data. Why?
- Expansion acts as a magnifying glass. In contrast to input reduction, it makes sure that the idiosyncracies of the data are preserved.
- If the idiosyncracies are salient enough (they contribute high activations in the random projection), they will be conserved in the final hashing.

What does the final hash look like?

- We don't know. Further analysis would be needed.
- Ideally, it needs to have two contrasting features:
  - a general shape that puts it close to similar elements in the space;
  - an indication of *what* makes it different from other elements, and to which degree.
- Note: this is an essential feature of any linguistic constituent:
  - Consider the sentence: *This is a fly.*
  - It has the features of a fly (it is similar to other flies).
  - It has the features that are not found in something that is not a fly (it is dissimilar to non-flies).

Does the fly do reference?

- To *do* reference, you need the ability to produce referring expressions.
- As far as we know, the fly cannot do generation for the benefit of another fly.
- However, the mechanism that produces the representations necessary to successfully refer seems to be there.