HIGH-DIMENSIONAL REPRESENTATION AND COMPUTING:

Word Embeddings

Guntis Barzdins

LATVIJAS UNIVERSITATE
ANNO 1919

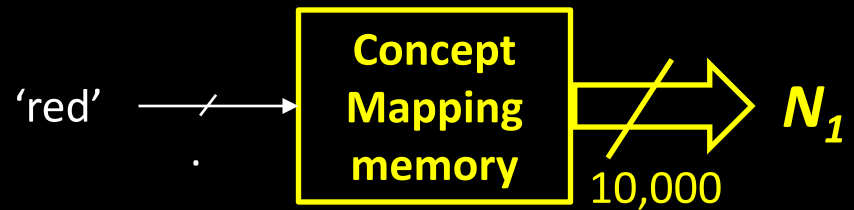
LETA
nacionālā informācijas aģentūra

Concept Mapping to Hypervectors

- Each concept is represented by a 10,000–D hypervector chosen at *random* (independent components) *:*

N_1 = [−1 +1 −1 −1 −1 +1 −1 −1 ...]
N_2 = [+1 −1 +1 +1 +1 −1 +1 −1 ...]
N_3 = [−1 −1 −1 +1 +1 −1 +1 −1 ...]
N_4 = [−1 −1 −1 +1 +1 −1 +1 −1 ...]
...
N_{64} = [−1 −1 +1 −1 +1 +1 +1 −1 ...]

- Every hypervector is dissimilar to others, e.g., $\langle N_1, N_2 \rangle = \mathbf{0}$
- This assignment is fixed throughout computation



HD Arithmetic

- **Addition (+)** is good for representing sets, since sum vector is similar to its constituent vectors.
  - $\langle A, B \rangle = 0$ ~0.1 (100D); ~0.05 (1000D); ~0.01 (10000D)
  - $\langle A, A \rangle = 1$
  - $\langle A+B, A \rangle = \sim 0.70$
  - $\langle A+B+C, A \rangle = \sim 0.57$
  - $\langle A+B+C+D, A \rangle = \sim 0.50$
  - $\langle A+B+C+D+E, A \rangle = \sim 0.45$
  - ...

[P. Kanerva, Cognitive Computation'09]

Concepts → Words

```
import numpy as np
D=10000
def similar(A,B): # Calculate Cosine similarity (normalized dot-product)
    return np.sum(A*B)/(np.sqrt(np.sum(A*A))*np.sqrt(np.sum(B*B)))

human   = np.random.randn(D)
leader   = np.random.randn(D)
male   = np.random.randn(D)
monarch   = np.random.randn(D)
female   = np.random.randn(D)
single   = np.random.randn(D)
plural   = np.random.randn(D)
estland   = np.random.randn(D)
capital   = np.random.randn(D)
country   = np.random.randn(D)
letland   = np.random.randn(D)

king      = human + leader + monarch + male + single
man       = human + male
woman     = human + female
queen     = human + leader + monarch + female + single

print  similar( queen, king - man + woman)

estonia   = estland + country
tallinn   = estland + capital
latvia = letland + country
riga   = letland + capital

print similar( riga, tallinn - estonia + latvia)
```

```
Guntiss-MacBook-Pro-2:seq2seq guntis$ python we.py
1.0
1.0
Guntiss-MacBook-Pro-2:seq2seq guntis$
```

Operations on Hypervectors: An example

. **Seed vectors**: 10,000 randomly placed 1s and -1s

```
            1  2  3           ....          10,000
          .-------------------------------------.
  A =     | +1 -1 -1 +1 -1 -1 .... +1 +1 -1 +1 |
          '-------------------------------------'
```

. A seed vector can represent a letter of the alphabet, for example

. **Addition** (+): Coordinate by coordinate

```
    A =  +1 -1 -1 +1 -1 -1 .... +1 +1 -1 +1
    B =  +1 +1 +1 +1 -1 +1 .... -1 +1 +1 +1
    C =  -1 -1 +1 -1 -1 +1 .... -1 -1 -1 +1
 -------------------------------------------
A+B+C =  +1 -1 +1 +1 -3 +1 .... -1 +1 -1 +3
```

. **Similarity** between vectors: Cosine

cos(**A**, **A**) = 1

cos(**A**,-**A**) = -1

cos(**A**, **B**) = 0 if **A** and **B** are **orthogonal**

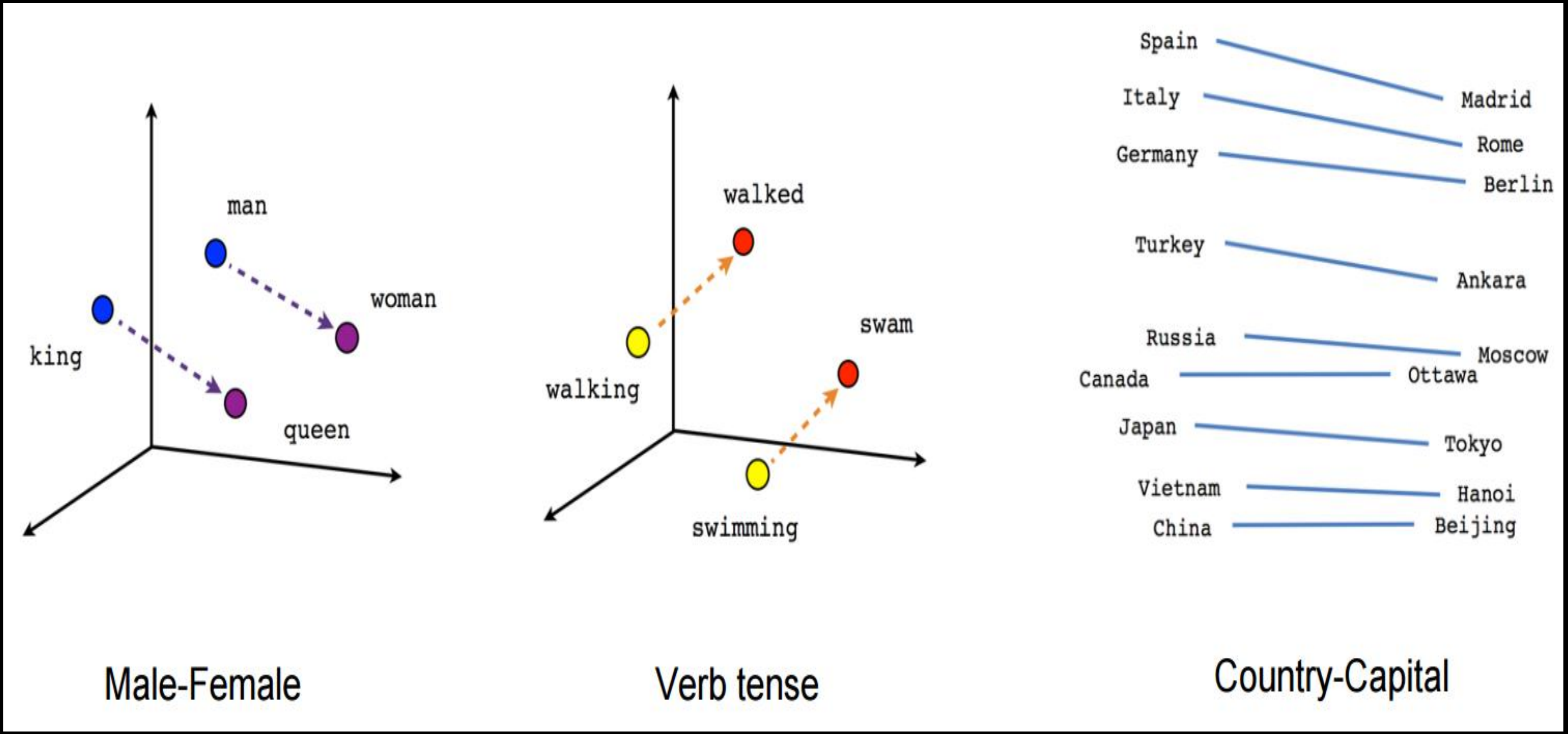
The **blessing of dimensionality:** A *randomly* chosen hypervector is *approximately orthogonal* (dissimilar) to any vector seen so far

Given two vectors of attributes, *A* and *B*, the cosine similarity, *cos(θ)*, is represented using a dot product and magnitude as

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}}$$

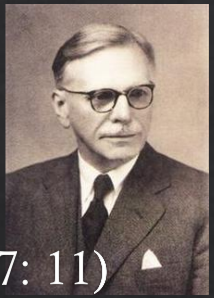
, where A_i and B_i are components of vector A and B respectively.

Examples from word2vec, GloVe



vector[Queen] = vector[King] - vector[Man] + vector[Woman]

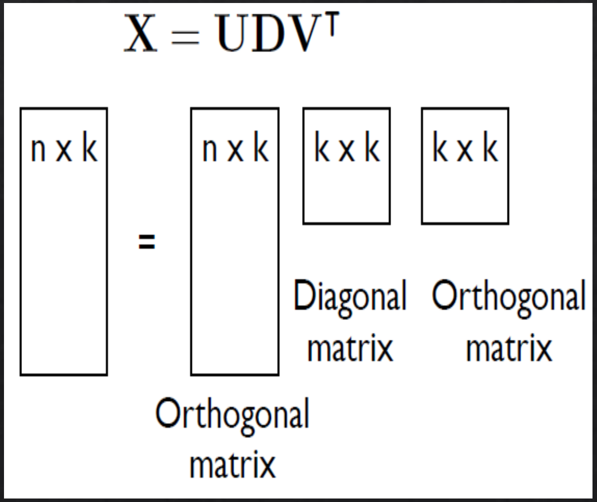
Building these magical vectors . . .



- How do we actually build these super-intelligent vectors, that seem to have such magical powers?
  - "You shall know a word by the company it keeps"(J. R. Firth 1957: 11)

- Most famous methods to build such lower-dimension vector representations for words based on their context
  1. Co-occurrence Matrix with SVD
  2. word2vec (*Google)*
  3. Global Vector Representations (GloVe) (*Stanford)*

Co-occurrence Matrix with Singular Value Decomposition



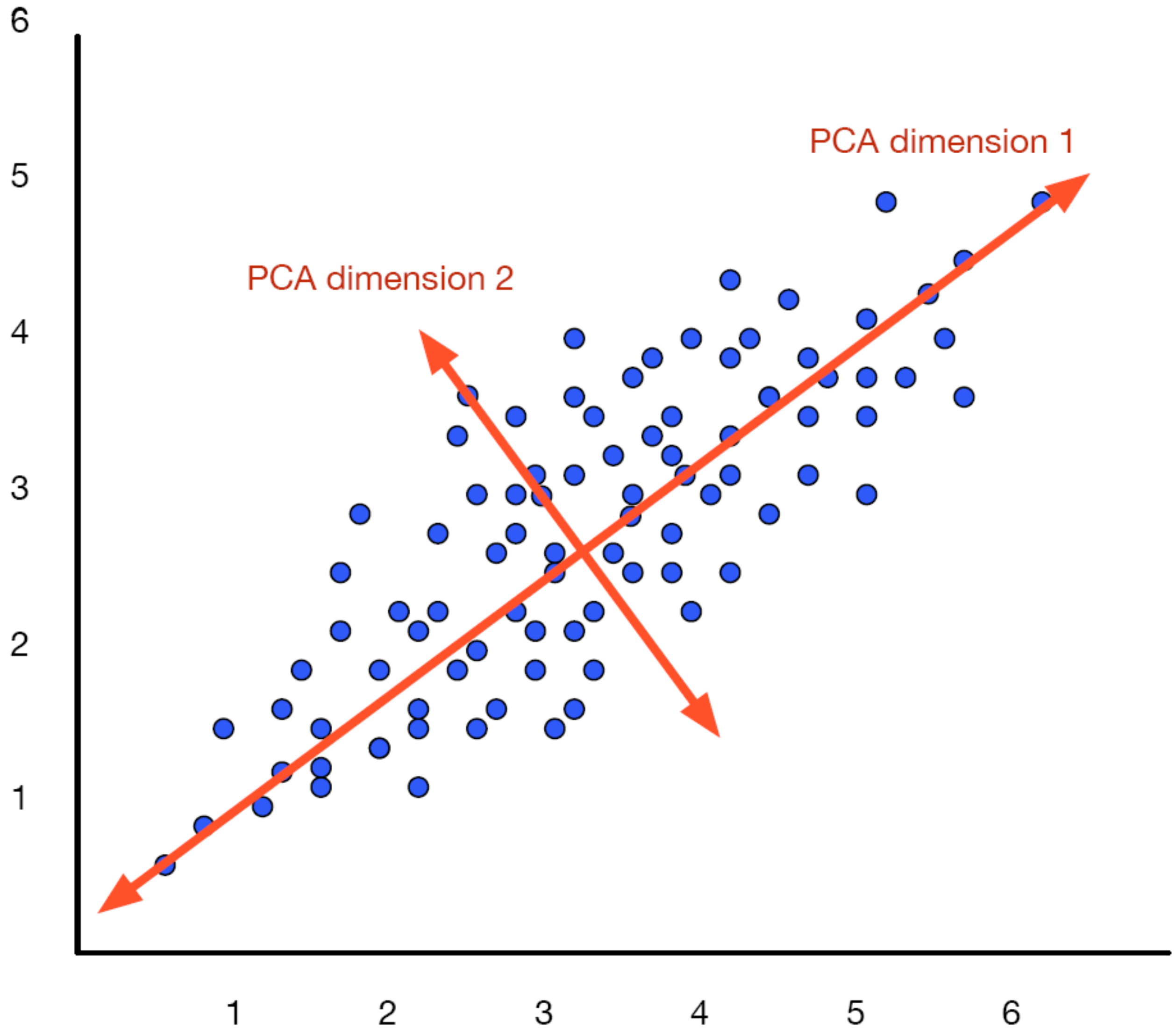
Building a co-occurrence matrix

Corpus = {"I like deep learning"
"I like NLP"
"I enjoy flying"}

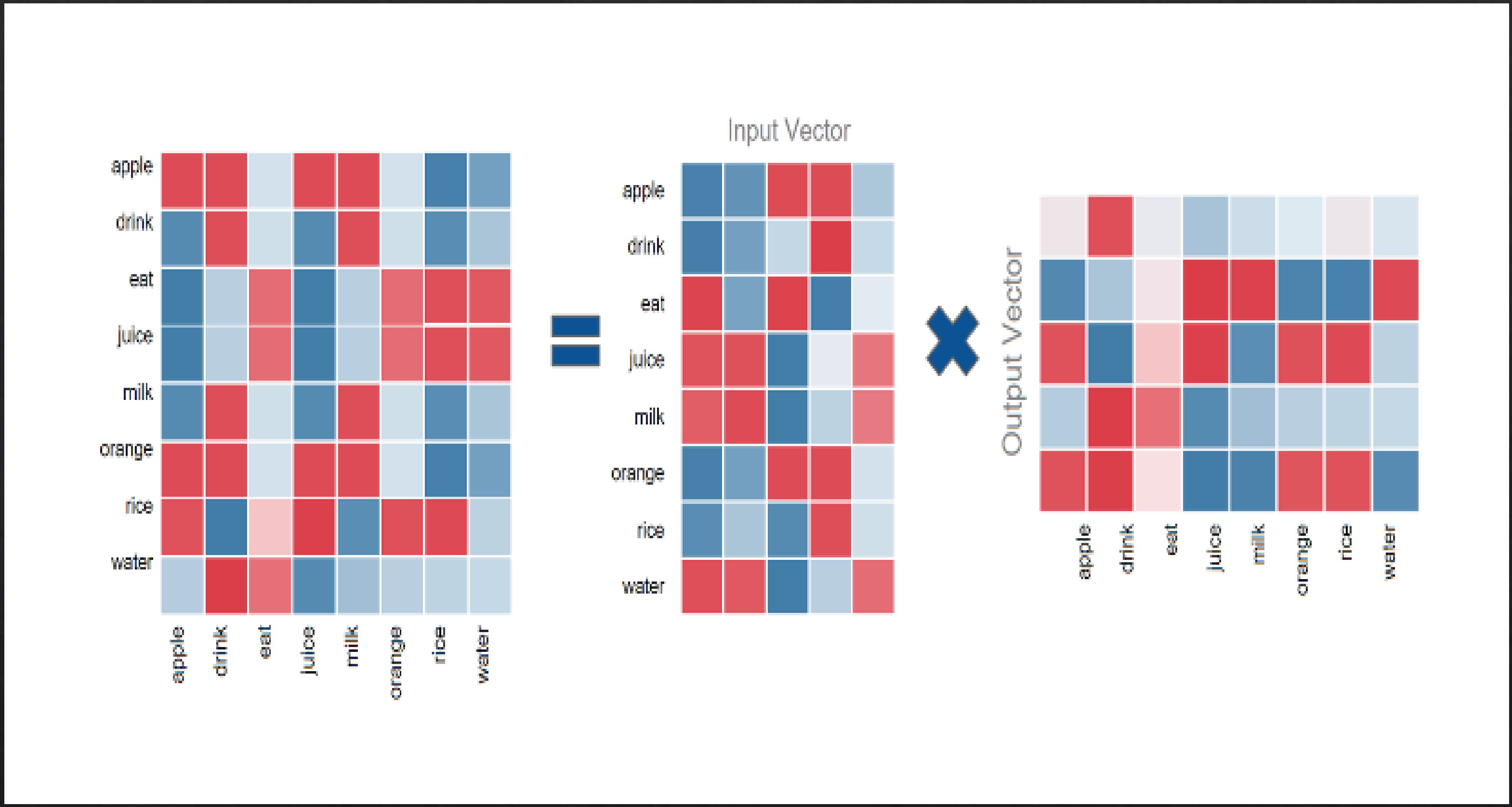
Context = previous word and next word

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

SVD: Intuition of Dimensionality reduction



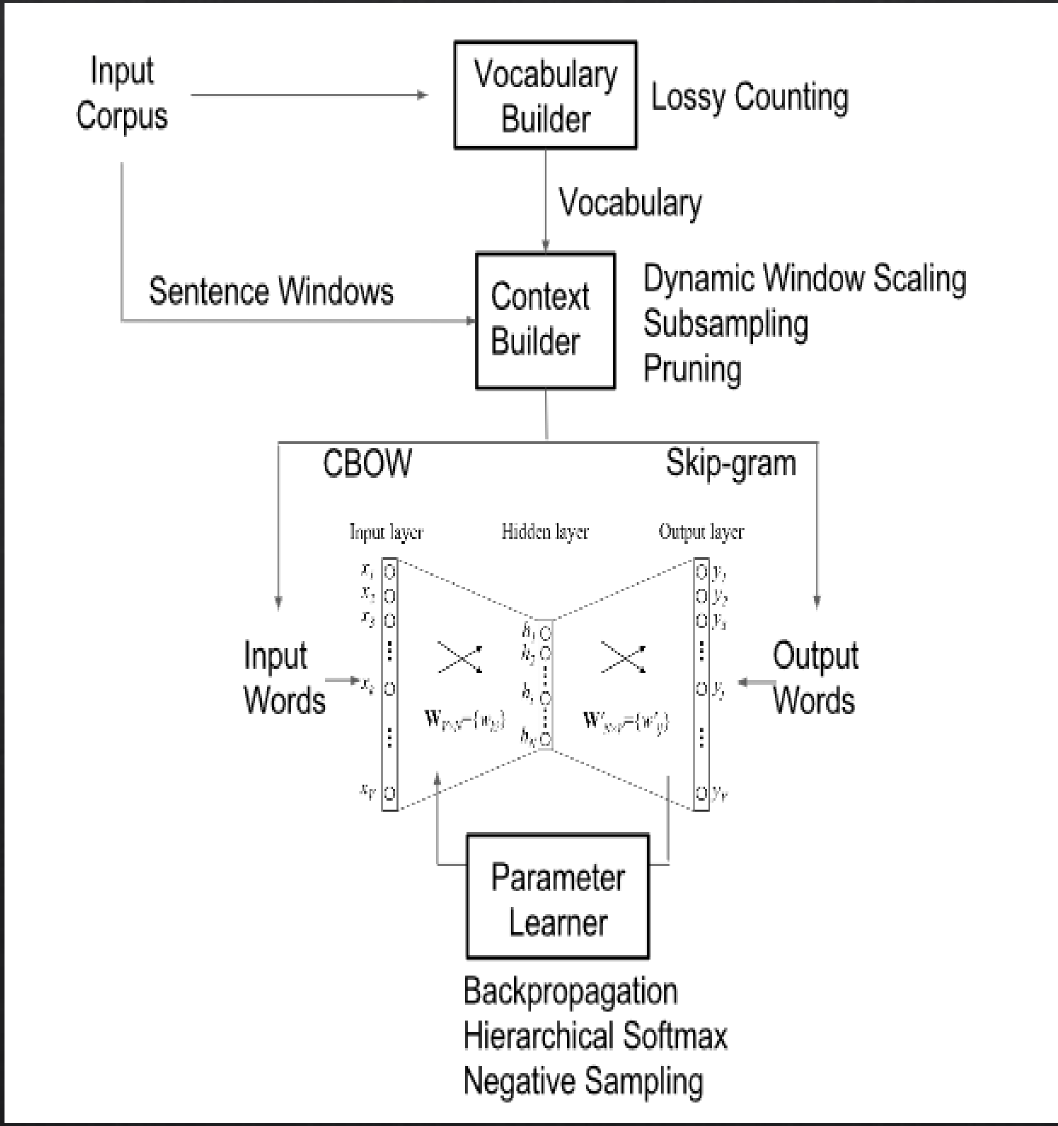
Singular Value Decomposition



The problem with this method, is that we may end up with matrices having billions of rows and columns, which makes SVD computationally restrictive.

word2vec

Architecture



The Research Question

- How to decompose the real-world (*word2vec* from billion word corpora) **word-vectors** into the orthogonal **concept-vectors**?

Concept-vectors: **<<**	Word-vectors **(binary):**
<1,0,0,0,0,0> - singular	<1,0,1,1,1,1> = king
<0,1,0,0,0,0> - female	<1,1,1,1,1,0> = queen
<0,0,1,0,0,0> - leader	<1,0,0,1,0,1> = man
<0,0,0,1,0,0> - human	<1,1,0,1,0,0> = woman
<0,0,0,0,1,0> - monarch	
<0,0,0,0,0,1> - male	**king – man + woman = queen**

- If this is possible, then word-embeddings are effectively **discrete**!

This question is part of my ERC Advanced grant submission. For overview of State-of-the-art: http://blog.aylien.com/a-review-of-the-recent-history-of-natural-language-processing/

Using word2vec in your research . . .

- Easiest way to use it is via the Gensim libarary for Python (tends to be slowish, even though it tries to use C optimizations like Cython, NumPy)

https://radimrehurek.com/gensim/models/word2vec.html

- Original word2vec C code by Google

https://code.google.com/archive/p/word2vec/

Word Embedding Visualization
http://ronxin.github.io/wevi/