# Introduction to Word Embeddings

**Cristina España-Bonet**

**UdS & DFKI, Saarbrücken, Germany**

Artificial Intelligence with Deep Learning

23rd April 2019

## *A word embedding is...*

**... a numerical representation of a word**

- Allow arithmetic operations on text

  *Ex: time + flies*

# *A word embedding is...*

## ... a numerical representation of a word

- Allow arithmetic operations on text

  *Ex: time + flies*

- Several names
  - Semantic Representations of Words
  - Word Vector Representations
  - Word Embeddings

# A word embedding is...

*time* = (1.844012, 0.590383, 1.003636, -0.577031, 1.515419, 1.097797, 1.812856, 0.933615, -2.396581, -0.931116, -0.719396, -0.376134, -1.204231, 0.045771, -0.287482, 1.084627, 4.399265, 1.516829, -0.838133, -1.881685, 0.108117, 2.345857, -1.292667, -2.286168, 3.419926, 4.260052, -1.016988, 3.140229, -3.161504, -0.800707, -1.433775, 2.290546, 1.932333, 0.714649, -3.033084, -0.958289, -1.704687, -1.597345, 1.525060, 3.337017, -2.787743, 1.479353, 3.452092, -3.242210, 0.532302, -0.551804, 2.344314, -0.919049, -1.872516, 0.080137, 1.208913, -2.136555, -2.218254, 0.206410, 0.133225, -1.521032, 1.735609, 2.885288, -2.048691, 2.375038, 0.316599, -0.254595, 2.159168, 1.118603, -0.775468, 0.933521, -0.351797, 2.193516, 2.499064, 2.818742, -0.213898, 0.446962, 1.767461, 1.342941, 1.117215, -0.042004, 4.199081, 3.041796, -1.770649, -0.528354, -2.067354, 0.283046, -0.099049, -0.105402, 2.823484, -2.583724, -2.906962, 0.592174, -3.029664, -0.170582, 0.406366, 1.963008, -3.229250, -3.499467, -0.136623, -1.551140, 0.348241, -1.597526, 0.703598, 3.122618, 0.466473, -0.113320, -2.119155, 1.092863, -0.908410, 0.253259, -1.082862, 4.408773, 2.419691, 2.343239, 0.703793, 1.270707, 0.410221, -1.293057, -0.799147, 2.214563, -0.212623, 1.206766, -0.731273, 2.308388, -1.029362, -2.080709, 0.749148, -1.412619, 1.073051, -2.498955, -0.520858, 1.391912, -1.181121, 1.523457, -1.245448, -0.290742, -2.589719, -0.366162, 3.586508, 0.908829, -1.125176, -0.937035, -1.163619, 1.759209, 3.678231, 0.019263, -0.395732, 1.142848, -0.500150, -3.005232, 2.287069, -0.524648, -0.944902, 0.038368, -1.093538, -0.697787, 0.767664, 2.399855, 2.425945, 1.563581, -1.086811, 0.372100, 1.400303, -2.278863, 0.643208, -0.459837, 1.756295, 2.057359, 3.140241, -1.740582, 1.386243, -1.822378, 1.528883, -1.984250, 1.214508, -1.336822, -0.321478, -0.162113, 0.272326, -2.673072, 0.612675, -0.657483, -0.557969, -3.358420, -2.559981, -1.683046, -1.314229, -2.425110, -2.506184, -1.606668, 1.332781, -2.760878, -2.400824, -1.830618, -2.406664,-1.169146, -1.838281, 0.588559, 2.285466, -0.401462, 1.632473, -0.510084, -2.072332, -2.627897, 2.531830, -2.524195, 2.035469, 1.906113, -1.257332, -4.039220, -0.467614, -2.275054, -3.409202, -0.014383, 0.445576, 1.461529, -1.318478, 0.061049, 0.280523, 2.173227, -0.027133, 2.791830, -0.728346, -1.804815, 1.245291, 0.970318, 2.646388, 0.246842, -1.823608, 1.888760, 0.265116, -2.027269, -0.089802, 0.389976, -0.654499, 2.565478, -2.647825, 2.658914, 1.385568, 2.306623, 0.476923, -0.869644, -0.170338, 0.495097, -2.604649, 0.610231, 0.739677, 0.322778, -2.042915, -1.353154, 0.177016, 1.840185, -0.271689, -0.401560, -0.421108, -0.185526, 1.041765, -4.599578, -0.829409, 0.076258, -0.503421, 1.891007, -0.931777, 0.434825, -0.467926, -1.417658, -0.320597, -4.084039, -3.899607, 0.977403, 0.774670, 3.269479, -1.031264, -0.433907, -2.30576 0, 0.811788, 2.347483, -1.254061, -0.861366, 0.080974, -3.666142, -0.363376, -2.384475, -4.290071, -0.924723, 1.257435, 1.223927, 0.276726, 1.541471, 1.274240, 1.883040, -1.987514, -0.809325, 1.252716, 1.812783, -0.511801, -1.657522, 1.196169, 0.804855, -1.861488, -2.113367, 0.429888, -0.920844, 0.377247)

# A word embedding is...

*flies* = (0.101159, 0.550446, 0.543801, -0.973852, -0.680835, 0.417193, -0.247181, 0.209725, -1.136055, -0.059531, -0.401640, 0.171540, 0.925121, -0.143815, 0.781714, -1.482425, 0.347008, -0.112342, 0.442418, -1.020457, -0.071752, 1.873548, -0.222886, -0.729569, -0.830224, -0.868407, 0.203496, 0.469911, -0.191363, 0.565102, 0.687738, 0.480823, 0.842358, -0.173656, -0.265585, 0.685740, 0.488047, -0.359772, -0.576064, -0.802884, 0.081554, 0.046882, -0.861532, -0.461855, 0.613098, -1.534642, -0.884534, 0.207728, 1.396512, -0.242900, -0.383959, 0.570844, -0.703350, -1.368813, -1.008194, 1.534660, 0.171693, 0.640925, -0.233116, 0.324685, 0.483171, 0.337947, -0.963290, -0.400558, 0.830977, 0.913474, 0.251693, -0.589420, -0.299622, 1.047515, -0.266679, -1.247186, 1.087610, -0.549028, 1.600710, -1.538516, -1.703301, -1.393499, -0.894448, 0.717204, 0.105767, -0.189234, -0.615609, -0.658315, 0.051877, 0.014180, -0.791282, 0.150424, 1.343751, -0.464859, 0.871426, 1.542864, -1.202150, -0.767113, -1.734738, 0.073633, -1.012583, 0.747787, 0.476070, -0.454807, 0.642685, -0.854152, -0.071798, 0.233724, 0.712329, -0.097752, -0.531132, 0.323271, -0.447342, 0.657913, 1.199492, -0.107360, -0.154234, -1.131168, 1.354793, 1.721385, -0.240023, 0.655765, -0.217006, -0.801722, 0.553369, 0.213377, 0.323267, -1.516051, 2.106244, -0.134282, 0.742155, 0.426344, 0.197991, -0.806768, 0.372546, -0.160200, -1.552847, -0.286178, -0.707796, 0.527352, -0.259658, 0.230387, 0.105294, -0.194481, 0.301772, -1.022163, 0.557191, 1.096709, 0.058422, -1.036384, 0.353412, -0.623097, -0.689515, 0.091472, 0.783885, 0.184088, -0.367950, 0.952462, 0.183704, 0.677562, 0.293917, -0.214309, -0.487794, 0.934296, 0.311513, 0.286514, -0.085511, 0.777691, 1.232603, -0.309367, -0.225086, 0.005091, -0.099195, -0.293117, 1.305563, 0.595816, 0.950316, 0.568706, -0.561446, 0.911634, -0.383941, 0.758054, -0.197820, 0.506777, -0.290767, -0.356727, 1.229474, -0.156489, -0.782741, -0.210163, -0.029169, 0.602664, 0.418375, 0.148975, -0.761796, 1.322690, -0.173410, 0.204111, -1.344531, 1.081905, -0.660543, -0.225615, -0.444753, -0.929671, 0.054136, 0.052031, -0.164926, 0.159312, -1.316333, 0.837011, -1.290353, 0.958403, 1.247478, 0.442009, 0.455497, -1.856268, -0.358823, -0.230839, -0.206271, 0.227012, -0.454163, 0.747798, -1.252855, 1.436849, -0.427915, -0.810428, -0.628144, -0.288458, 0.087355, 0.356739, 0.153036, 0.516594, -0.504978, 0.814432, 1.052940, 1.094526, -0.219595, 0.722178, 0.267325, -0.087458, -1.270262, -0.039461, 0.991926, -0.112005, -0.009605, 0.149920, 0.164717, 0.280475, 0.966384, 0.327598, 0.189590, -0.208946, 0.838261, 0.051847, -0.277932, -0.788527, -0.768702, -1.688721, 0.388215, 0.170153, -0.555723, -0.529565, -0.528982, -0.659930, 0.588041, -0.368195, -0.850188, -0.004996, 0.925476, 1.046587, -0.731761, 0.519435, 0.193188, -0.709557, 0.123329, -0.454316, 1.885830, -0.201841, -0.728933, -0.953455, -0.205837, -0.724068, 0.120158, 1.765389, -0.192159, 1.062490, -0.002634, 0.125790, -0.846565, 0.548899, -1.062821, -2.146826, 0.134681, 0.570950, 0.851783, 0.436544, 0.688986, 1.229008, 1.435449, 0.118766, -0.132411, 2.527890, 0.778142, 0.269093)

## *The Purpose of the First Lecture is...*

**...to answer several questions:**

- How can we obtain those numbers?
- What's the well known word2vec?
- Is it the only way to obtain those numbers?
- Do the vectors (and components!) have any semantic meaning?
- Are we crazy by summing or multiplying words?

# Outline



# Introduction



# Introduction

## *Distributional Hypothesis, Contextuality*

**Never ask for the meaning of a word in isolation, but only in the context of a sentence**
(Frege, 1884)

# Introduction

## *Distributional Hypothesis, Contextuality*

**Never ask for the meaning of a word in isolation, but only in the context of a sentence**
(Frege, 1884)

**For a large class of cases... the meaning of a word is its use in the language**
(Wittgenstein, 1953)

**You shall know a word by the company it keeps**
(Firth, 1957)

# Introduction

## *Distributional Hypothesis, Contextuality*

**Words that occur in similar contexts tend to have similar meaning**

**(Harris, 1954)**

# Introduction

## *Similar Meanings...*

- ...need for a concept of **distance** to be defined.
- **Geometry** is the branch of mathematics that deals with distances
- **Vector spaces** and linear algebra are our tools

# Introduction

## *Similar Meanings...*

**Sumo**

**Basketball**

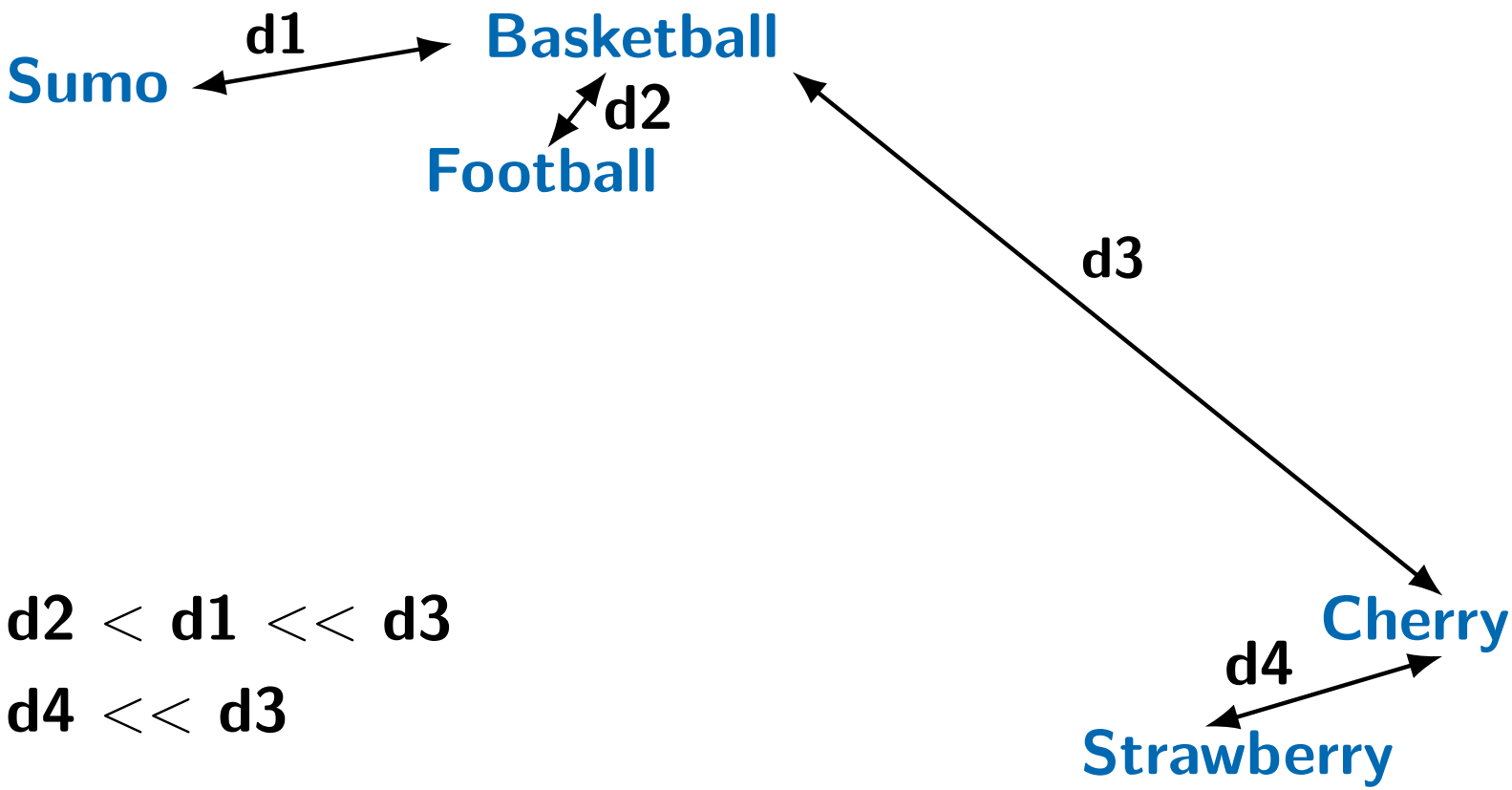
**Football**

**Cherry**

**Strawberry**

# Introduction

## *Similar Meanings...*

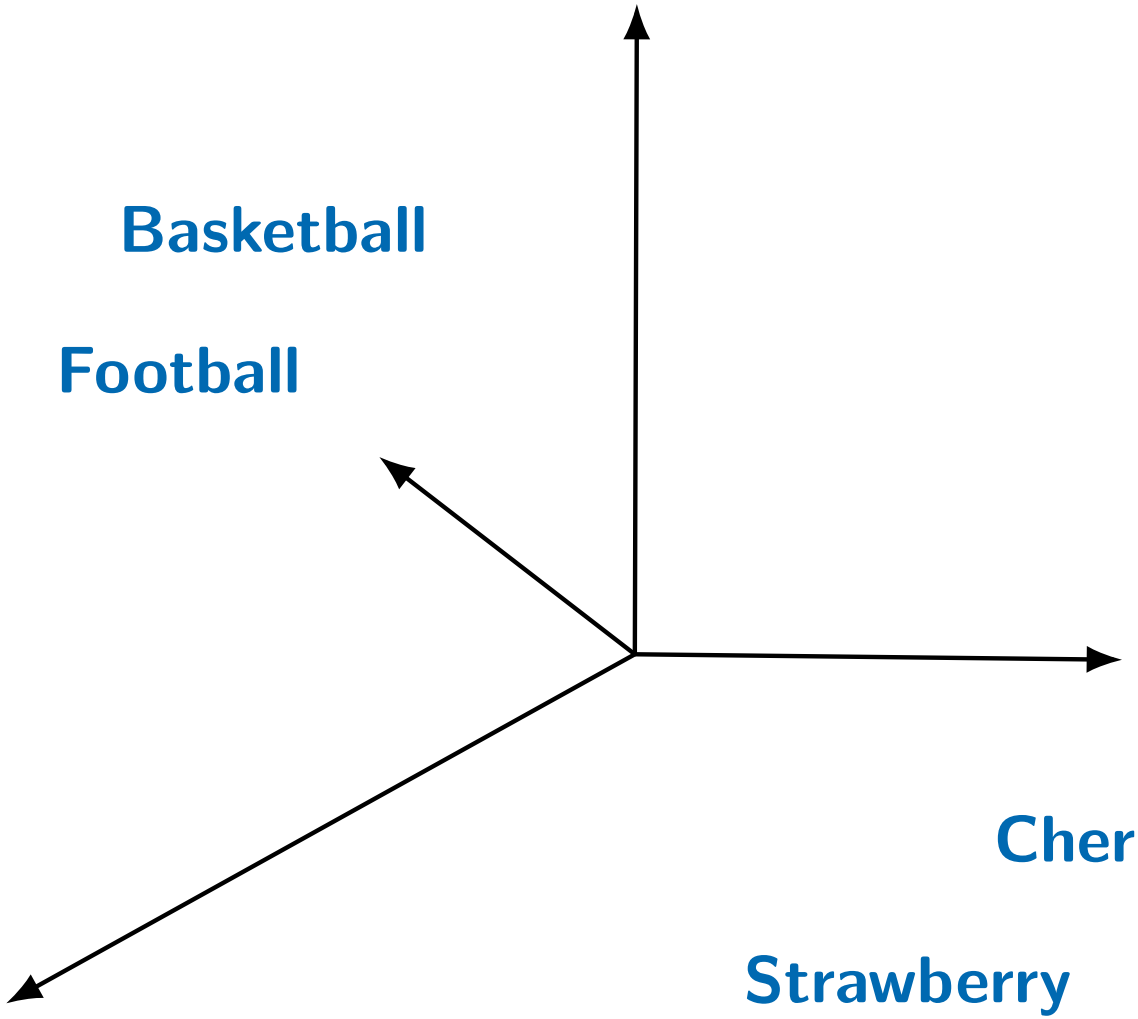



$d2 < d1 << d3$

$d4 << d3$

# Introduction

## *Word Vector Space*




***n*-dimensional vector space**

# Introduction

## *How to Obtain a Vector for a Word?*

Naïve example: **term frequencies** in a corpus

- The basis in our vector space is the vocabulary of the corpus
- Consider the document in which a word occurs its context
- Each word is characterised as the number of times it appears in each document

# Introduction

## *Example: Toy Corpus*

*S1*: We like to play some sport in the afternoon, I like basketball but John likes sumo more.

*S2*: Sumo is a sport where a rikishi attempts to force another wrestler out of a circular ring.

*S3*: Messi scored 4 goals yesterday and kept the ball as a memory of this fantastic sports afternoon!

*S4*: I ate too many cherries yesterday.

**Vocabulary**:{*like*, *play*, *sport*, *afternoon*, *basketball*, *John*, *ball*, *sumo*, *rikishi*, *attempt*, *force*, *werstler*, *circular*, *ring*, *Messi*, *score*, *goal*, *yesterday*, *keep*, *memory*, *fantastic*, *eat*, *cherry*}

# Introduction

## *Example: Occurrence Matrix*

| | like | play | sport | afternoon | basketball | John | ball | sumo | rikishi | attempt | force | werstler | circular | ring | Messi | score | goal | yesterday | keep | memory | fantastic | eat | cherry |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *S1* | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *S2* | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *S3* | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| *S4* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |

# Introduction

## *Example: Occurrence Matrix*

| | like | play | sport | afternoon | basketball | John | ball | sumo | rikishi | attempt | force | werstler | circular | ring | Messi | score | goal | yesterday | keep | memory | fantastic | eat | cherry |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *S1* | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *S2* | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *S3* | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| *S4* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |

**document vector**

# Introduction

## *Example: Occurrence Matrix*

| | like | play | sport | afternoon | basketball | John | ball | sumo | rikishi | attempt | force | werstler | circular | ring | Messi | score | goal | yesterday | keep | memory | fantastic | eat | cherry |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *S1* | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *S2* | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *S3* | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| *S4* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |

**word vector**

# Introduction

## *Example: Text Similarity*

**Euclidean distance** $\mathrm{d}(\overrightarrow{x}, \overrightarrow{y}) = \sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{x_i} - \mathrm{y_i})^2}$

basketball $\rightarrow \{1, 0, 0, 0\}$
sumo $\rightarrow \{1, 1, 0, 0\}$
cherry $\rightarrow \{0, 0, 0, 1\}$

$\mathrm{d}(\text{basketball, sumo}) = \sqrt{(1-1)^2 + (0-1)^2 + (0-0)^2 + (0-0)^2} = 1$
$\mathrm{d}(\text{basketball, cherry}) = \sqrt{(1-0)^2 + (0-1)^2 + (0-0)^2 + (0-0)^2} = \sqrt{2}$
$\mathrm{d}(\text{sumo, cherry}) = \sqrt{(1-0)^2 + (1-0)^2 + (0-0)^2 + (0-1)^2} = \sqrt{3}$

$\mathrm{d}(\text{basketball, sumo}) < \mathrm{d}(\text{basketball, cherry}) < \mathrm{d}(\text{sumo, cherry})$

# Introduction

## *Example: Text Similarity*

**Cosine similarity** $\quad \mathrm{sim}(\vec{x}, \vec{y}) = \frac{\sum_{i=1}^{n} x_i y_i}{|\vec{x}||\vec{y}|}$

basketball $\rightarrow \{1, 0, 0, 0\}$
sumo $\rightarrow \{1, 1, 0, 0\}$
cherry $\rightarrow \{0, 0, 0, 1\}$

$\mathrm{sim}(\mathrm{basketball}, \mathrm{sumo})=1$

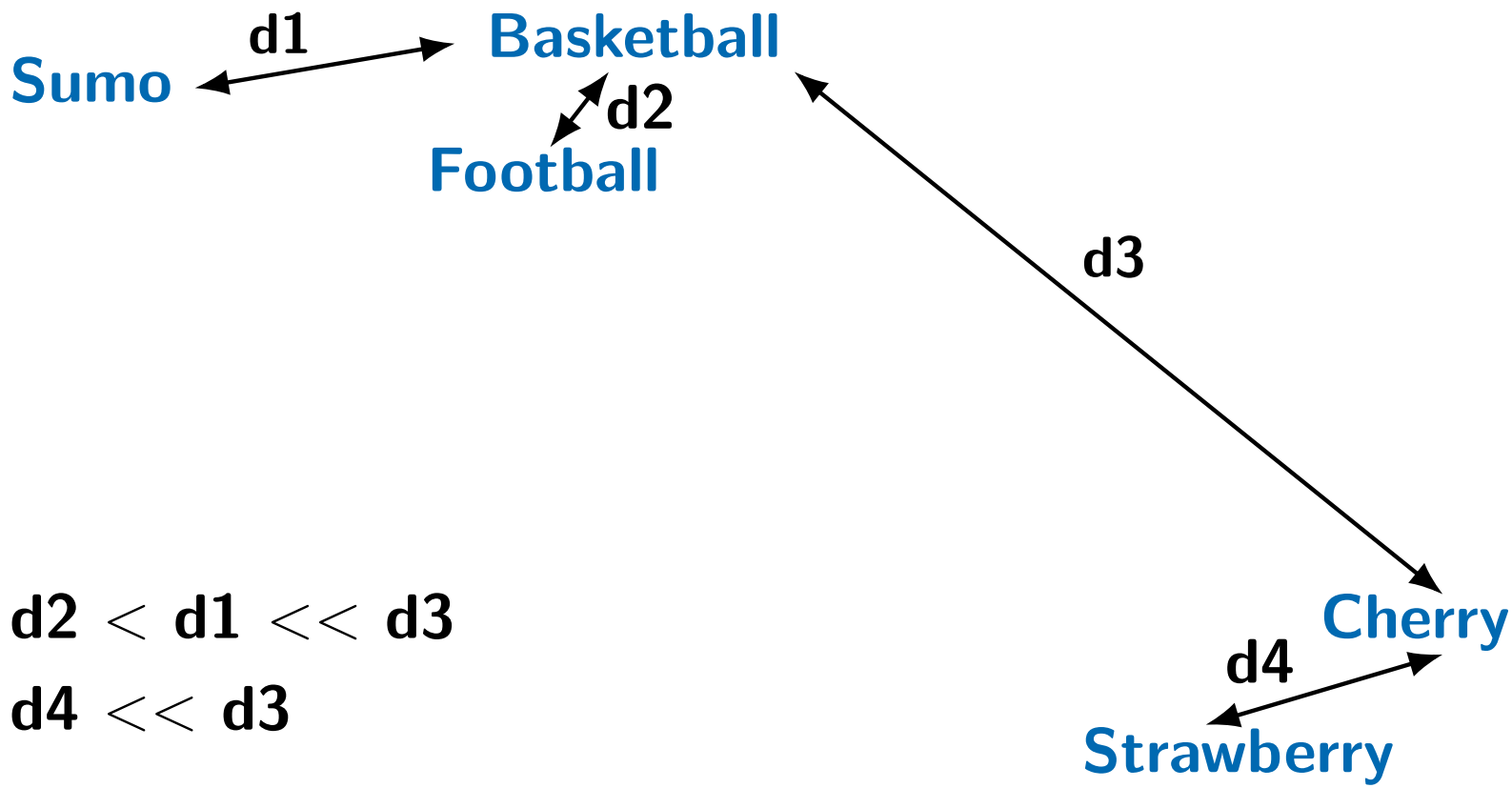
$\mathrm{sim}(\mathrm{basketball}, \mathrm{cherry})=0$

$\mathrm{sim}(\mathrm{sumo}, \mathrm{cherry})=0$

$\mathrm{sim}(\mathrm{basketball},\mathrm{sumo}) > \mathrm{sim}(\mathrm{basketball},\mathrm{cherry}) = \mathrm{sim}(\mathrm{sumo},\mathrm{cherry})$

# Introduction

## *Similarity vs. Distance*

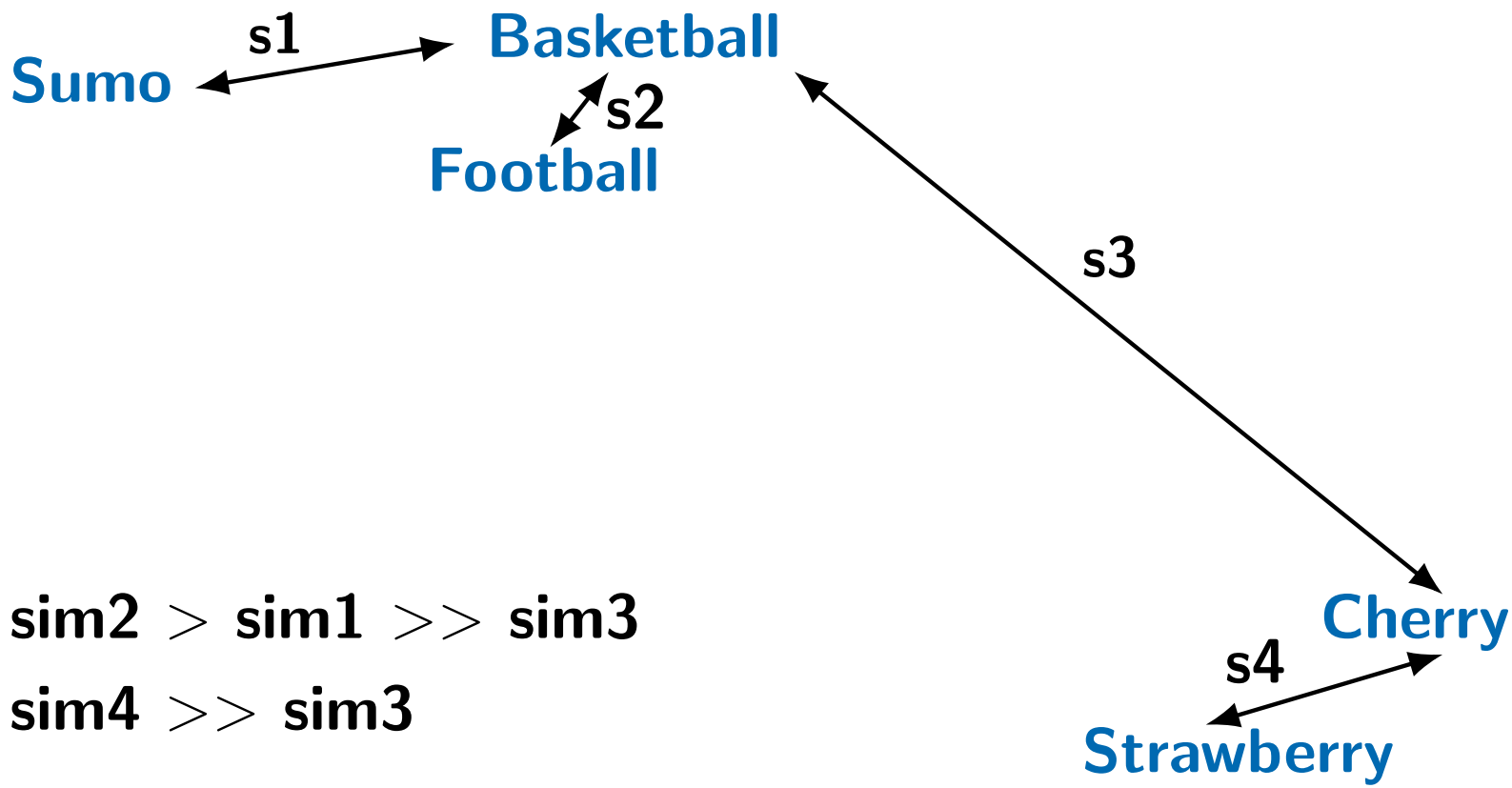



$d2 < d1 << d3$

$d4 << d3$

# Introduction

## *Similarity vs. Distance*




**sim2** $>$ **sim1** $>>$ **sim3**

**sim4** $>>$ **sim3**

# Frequency-based Embeddings



# Frequency-based Embeddings

## *Frequency-based Embeddings*

- *Term frequency word vectors*
- TF-IDF word vectors
- Co-occurrence word vectors

# Frequency-based Embeddings

## *Term Frequency-Inverse Document Frequency, TF-IDF*

### Term Frequency

How frequently a term occurs in a document $d$ normalised to account for $d$ length

$$\text{TF}(t,\, d) = \frac{\text{Number of times term } t \text{ appears in a document } d}{\text{Total number of terms in } d}$$

# Frequency-based Embeddings

## *Term Frequency-Inverse Document Frequency, TF-IDF*

### Term Frequency

How frequently a term occurs in a document $d$ normalised to account for $d$ length

$$\text{TF}(t,\, d) = \frac{\text{Number of times term } t \text{ appears in a document } d}{\text{Total number of terms in } d}$$

### Inverse Document Frequency

Measures how important a term is (low weight for stop words)

$$\text{IDF}(t,\, D) = \log_e \left( \frac{\text{Total number of documents } D}{\text{Number of documents with term } t \text{ in it}} \right)$$

# Frequency-based Embeddings

## *Term Frequency-Inverse Document Frequency, TF-IDF*

Trivially...

$$\text{TF-IDF}(t,\, d,\, D) = \text{TF}(t,\, d) \times \text{IDF}(t,\, D)$$

# Frequency-based Embeddings

## *Example: Toy Corpus*

*d1*: We like to play some sport in the afternoon, I like basketball but John likes sumo more.

*d2*: Sumo is a sport where a rikishi attempts to force another wrestler out of a circular ring.

*d3*: Messi scored 4 goals yesterday and kept the **ball** as a memory of this fantastic sports afternoon!

*d4*: I ate too many cherries yesterday.

$$\textbf{TF(ball)} = \left(0, 0, \frac{1}{17}, 0\right);$$

# Frequency-based Embeddings

## *Example: Toy Corpus*

*d1*: We like to play some sport in the afternoon, I like basketball but John likes sumo more.

*d2*: Sumo is a sport where a rikishi attempts to force another wrestler out of a circular ring.

*d3*: Messi scored 4 goals yesterday and kept the **ball** as a memory of this fantastic sports afternoon!

*d4*: I ate too many cherries yesterday.

$$\textbf{TF(ball)} = \left(0, 0, \frac{1}{17}, 0\right); \qquad \textbf{IDF(ball)} = \log_e \left(\frac{4}{1}\right);$$

# Frequency-based Embeddings

## *Example: Toy Corpus*

*d1*: We like to play some sport in the afternoon, I like basketball but John likes sumo more.

*d2*: Sumo is a sport where a rikishi attempts to force another wrestler out of a circular ring.

*d3*: Messi scored 4 goals yesterday and kept the **ball** as a memory of this fantastic sports afternoon!

*d4*: I ate too many cherries yesterday.

$$\textbf{TF(ball)} = \left(0, 0, \frac{1}{17}, 0\right); \qquad \textbf{IDF(ball)} = \log_e\left(\frac{4}{1}\right);$$

$$\textbf{TF-IDF(ball)}_3 = \frac{1}{17} \times \log_e(4) = 0.08$$

# Frequency-based Embeddings

## *Example: Toy Corpus*

*d1*: We like to play some sport in the afternoon, I like basketball but John likes sumo more.

*d2*: Sumo is **a** sport where **a** rikishi attempts to force another wrestler out of **a** circular ring.

*d3*: Messi scored 4 goals yesterday and kept the ball as **a** memory of this fantastic sports afternoon!

*d4*: I ate too many cherries yesterday.

$$\textbf{TF(a)} = \left(0, \frac{3}{17}, \frac{1}{17}, 0\right);$$

# Frequency-based Embeddings

## *Example: Toy Corpus*

*d1*: We like to play some sport in the afternoon, I like basketball but John likes sumo more.

*d2*: Sumo is **a** sport where **a** rikishi attempts to force another wrestler out of **a** circular ring.

*d3*: Messi scored 4 goals yesterday and kept the ball as **a** memory of this fantastic sports afternoon!

*d4*: I ate too many cherries yesterday.

$$\textbf{TF(a)} = \left(0, \frac{3}{17}, \frac{1}{17}, 0\right); \qquad \textbf{IDF(a)} = \log_e\left(\frac{4}{2}\right);$$

# Frequency-based Embeddings

## *Example: Toy Corpus*

*d1*: We like to play some sport in the afternoon, I like basketball but John likes sumo more.

*d2*: Sumo is **a** sport where **a** rikishi attempts to force another wrestler out of **a** circular ring.

*d3*: Messi scored 4 goals yesterday and kept the ball as **a** memory of this fantastic sports afternoon!

*d4*: I ate too many cherries yesterday.

$$\textbf{TF(a)} = \left(0, \frac{3}{17}, \frac{1}{17}, 0\right); \qquad \textbf{IDF(a)} = \log_e\left(\frac{4}{2}\right);$$

$$\textbf{TF-IDF(a)}_2 = \frac{3}{17} \times \log_e(2) = 0.12; \qquad \textbf{TF-IDF(a)}_3 = 0.04$$

# Frequency-based Embeddings

## *Term Frequency-Inverse Document Frequency, TF-IDF*

- Word vectors of $D$ dimensions
- Distances between words as before:
  - Euclidean distance
  - Cosine similarity
  - ...

# Frequency-based Embeddings

## *Co-Occurence Matrix, Count Vectors*

- Words co-occurrence statistics describes how words occur together
- Counts how two or more words occur together in a given corpus

# Frequency-based Embeddings

## *Example: Toy Corpus*

*d1*: We like to play some sport in the afternoon, I like basketball but John likes sumo more.

*d2*: Sumo is a sport where a rikishi attempts to force another wrestler out of a circular ring.

*d3*: Messi scored 4 goals yesterday and kept the ball as a memory of this fantastic sports afternoon!

*d4*: I ate too many cherries yesterday.

# Frequency-based Embeddings

## *Example: Toy Corpus*

*d1*: We **like** to play some sport in the afternoon, I **like** basketball but John likes sumo more.

*d2*: Sumo is **a** sport where **a** rikishi **attempts** to force another wrestler out of **a** circular ring.

*d3*: **Messi** scored 4 goals yesterday and kept the **ball** as **a** memory of this fantastic sports afternoon!

*d4*: I ate too many cherries yesterday.

# Frequency-based Embeddings

## *Example: Co-Occurrence Matrix*

| | *like* | *to* | *sport* | *afternoon* | *basketball* | *John* | *ball* | *a* | *rikishi* | *attempt* | *force* | *werstler* | *circular* | *as* | *Messi* | *score* | *goal* | *yesterday* | *keep* | *memory* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *like* | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *attempt* | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *Messi* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| *a* | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| *ball* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| *...* | | | | | | | | | | | | | | | | | | | | |

# Frequency-based Embeddings

## *Co-occurence matrix, count vectors*

- Simple bigram frequencies of all possible word-pairs need a size $N \times N$ matrix to represent $N$ words in a corpus
- Real models use context windows, not only bigrams
- Counts are converted into probabilities
- In general, one has sparse matrices
- Dimensionality reduction (SVD, for instance)

# Frequency-based Embeddings

## *Aside Comment: One-Hot Encodings*

| | *like* | *to* | *sport* | *afternoon* | *basketball* | *John* | *ball* | *a* | *rikishi* | *attempt* | *force* | *werstler* | *circular* | *as* | *Messi* | *score* | *goal* | *yesterday* | *keep* | *memory* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *like* | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *attempt* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *Messi* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| *a* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *...* | | | | | | | | | | | | | | | | | | | | |

One-hot encoding for *like* with this vocabulary:
*like* = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

# Frequency-based Embeddings

## *Aside Comment: One-Hot Encodings*

| | *like* | *to* | *sport* | *afternoon* | *basketball* | *John* | *ball* | *a* | *rikishi* | *attempt* | *force* | *werstler* | *circular* | *as* | *Messi* | *score* | *goal* | *yesterday* | *keep* | *memory* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *like* | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *attempt* | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *Messi* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| *a* | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| ... | | | | | | | | | | | | | | | | | | | | |

Co-occurence word vector for *like* with this vocabulary in the previous corpus:
*like* = (0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

# Prediction-based Embeddings



# Prediction-based Embeddings

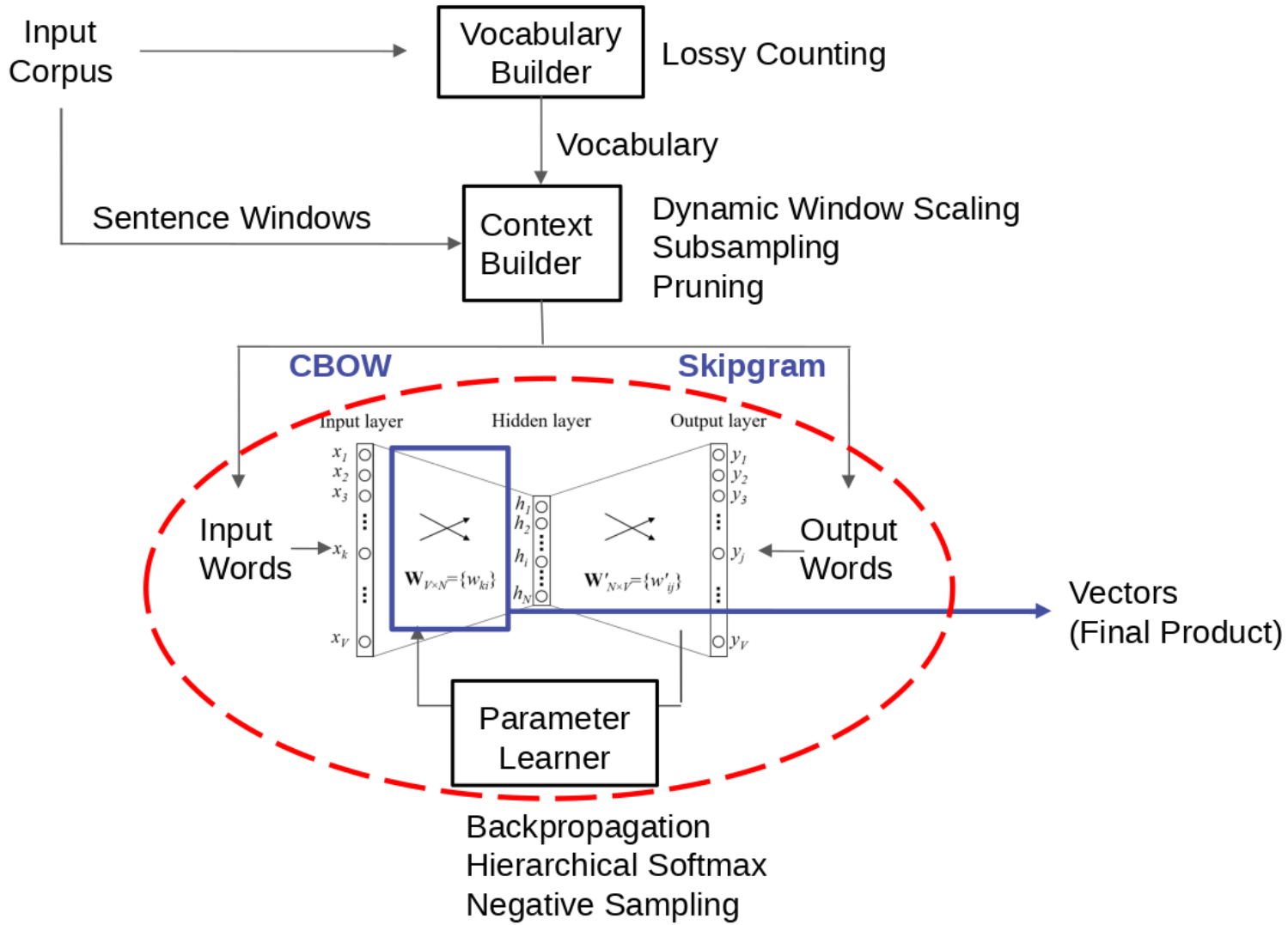
## *Word Embeddings (word2vec example)*

Word vectors learned by a neural network in two tasks:

1. predict the probability of a **word given a context** (CBoW)
2. predict the **context given a word** (skip-gram)

# Prediction-based Embeddings

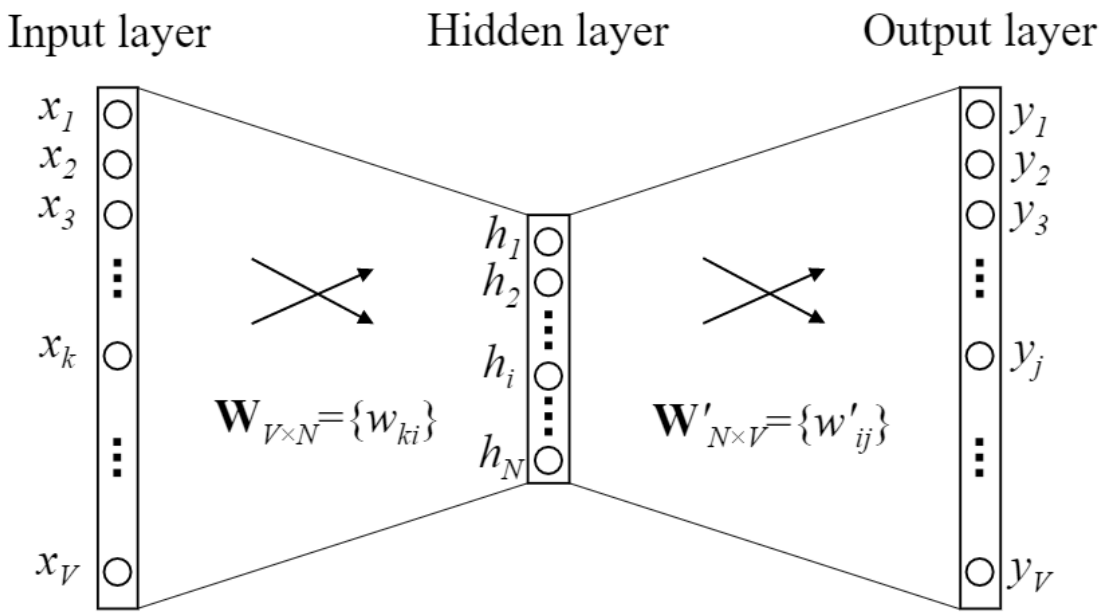
## *Word Embeddings (word2vec example)*




*Credits: Xin Rong*

# Prediction-based Embeddings

## *Word Embeddings (word2vec example)*




Look at the network: simple feed-forward network learned by backpropagation with cross-entropy loss

No deep learning at all!

# Prediction-based Embeddings

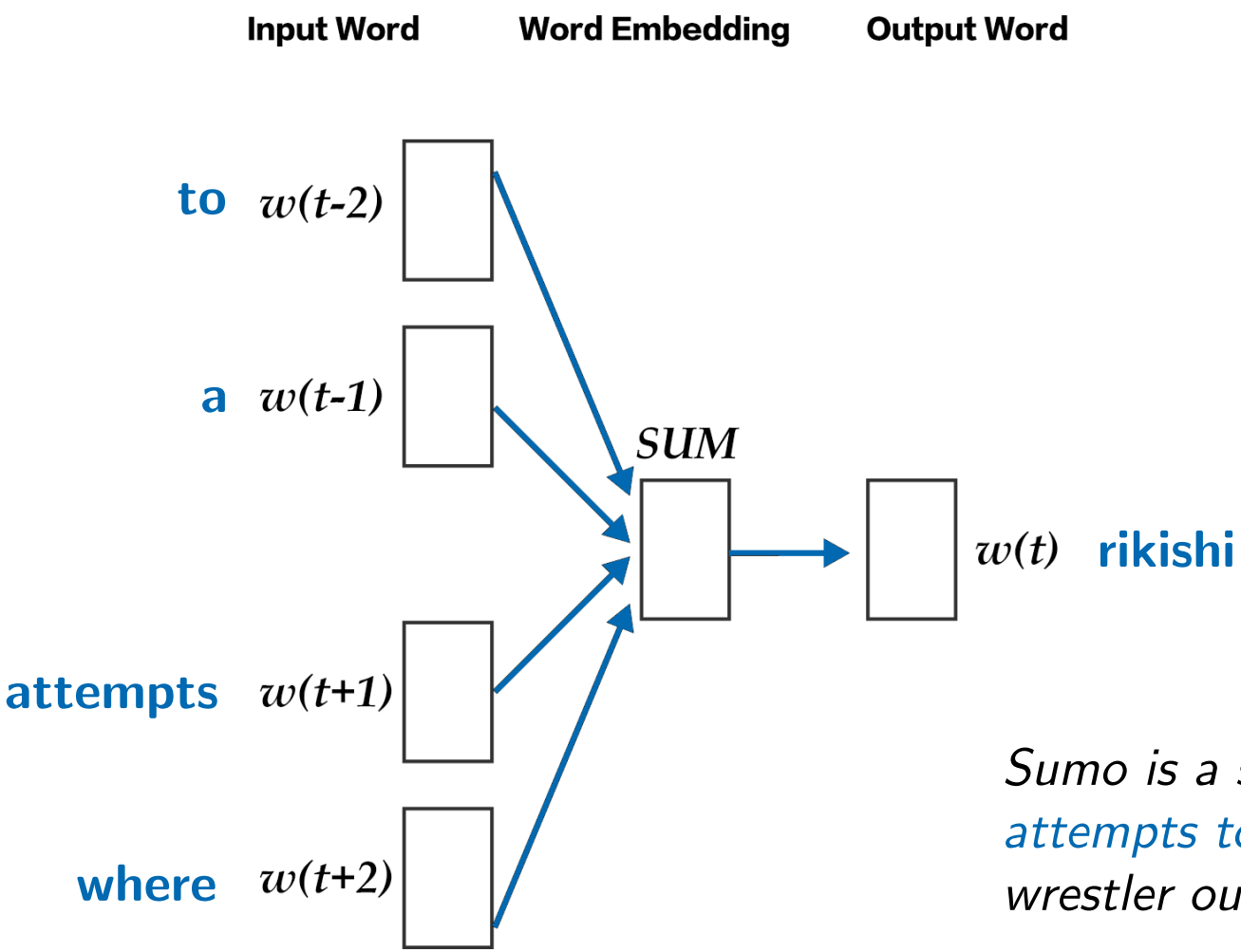
## *Word Embeddings (word2vec example)*

**Comments:**

- A hidden layer in a NN interprets the input in his own way to optimise his work in the concrete task
- The size of the hidden layer gives you the dimension of the word embeddings
- Too few neurons could not have enough capacity to learn everything needed
- Too many neurons would need a very large corpus to be meaningful

# Prediction-based Embeddings

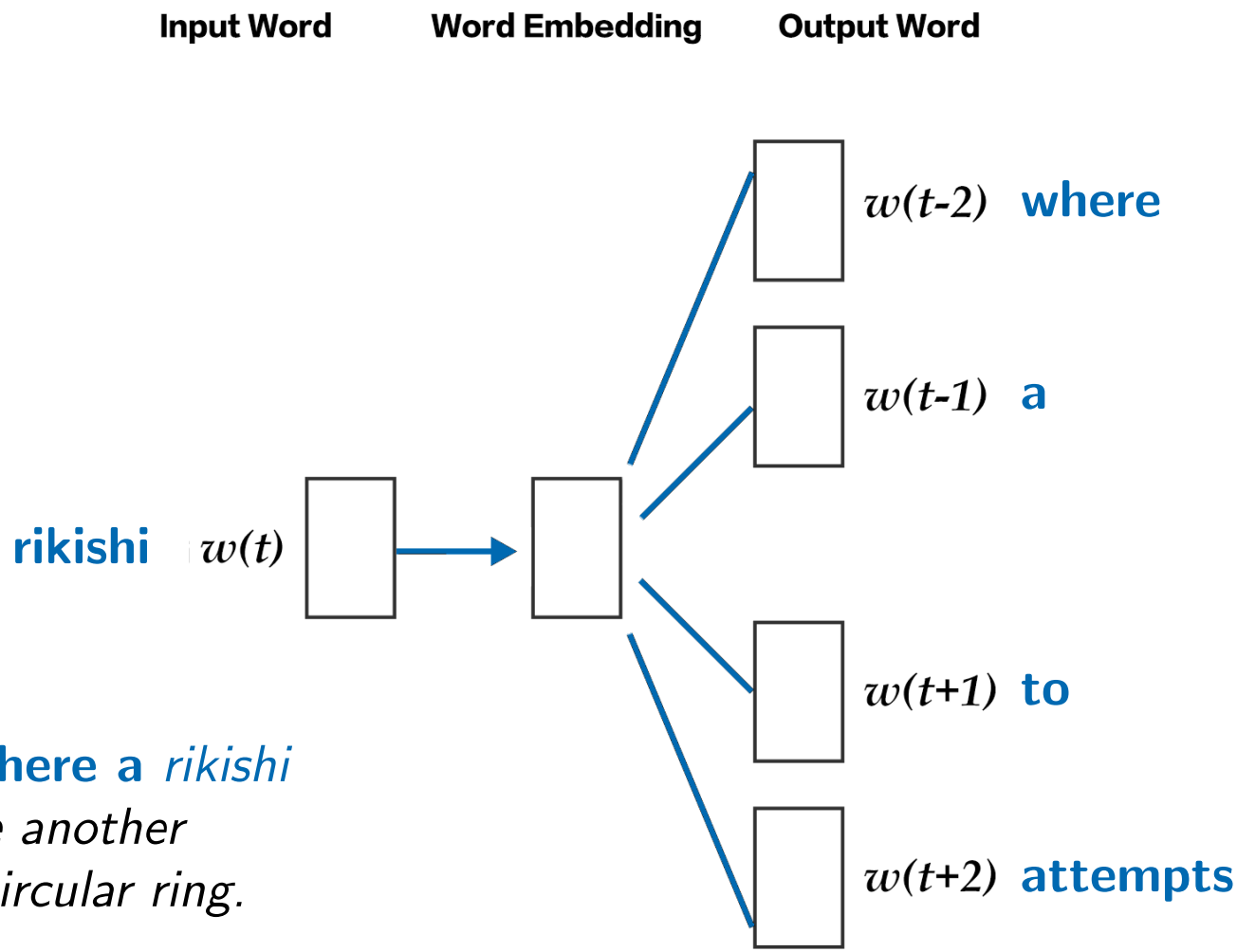
## *Continuous Bag of Words, CBoW*




*Sumo is a sport where a* **rikishi** *attempts to force another wrestler out of a circular ring.*

# Prediction-based Embeddings

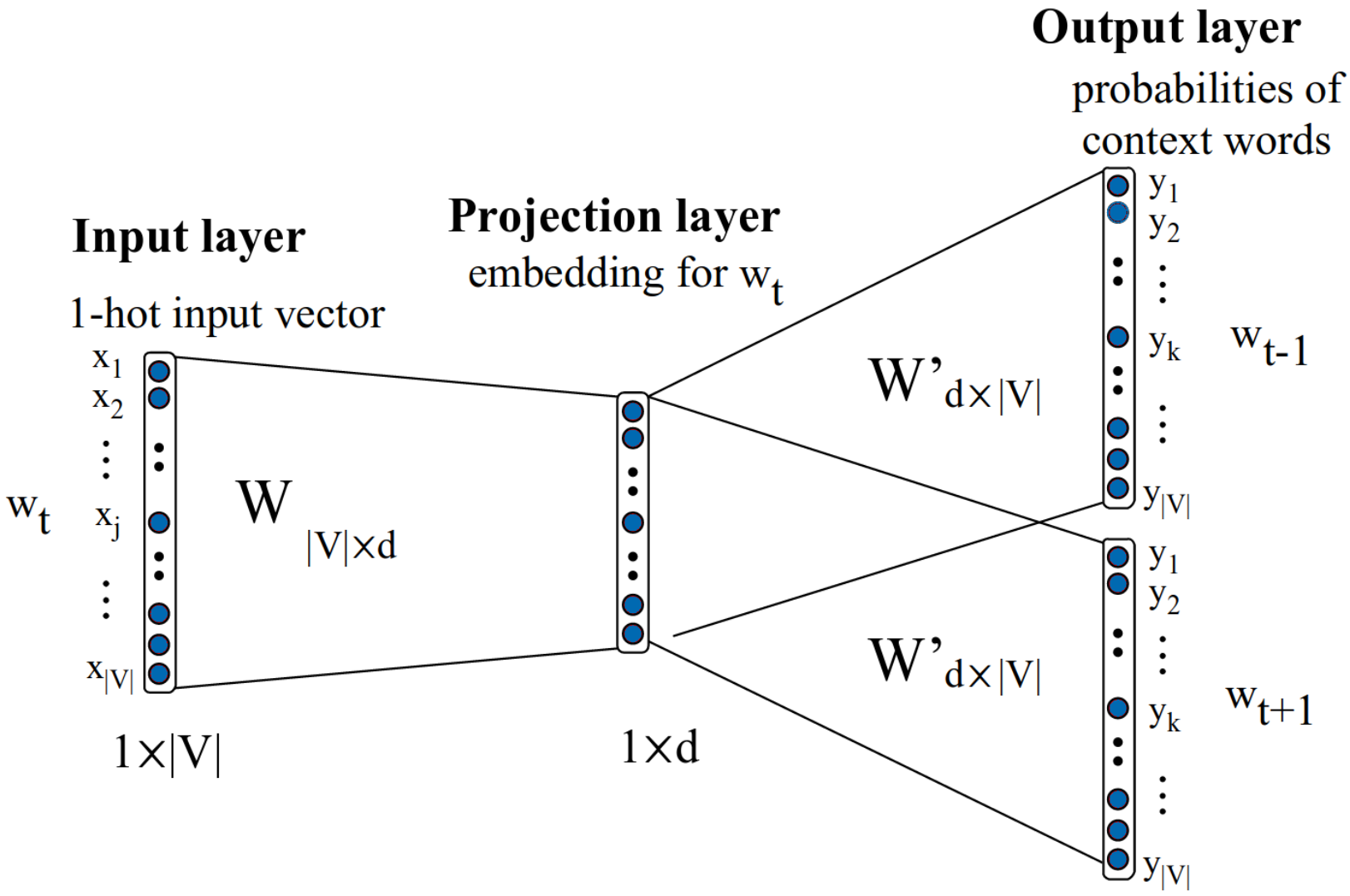
## *Skip-Gram Model*




*Sumo is a sport* **where a** *rikishi* **attempts to** *force another wrestler out of a circular ring.*

# Prediction-based Embeddings

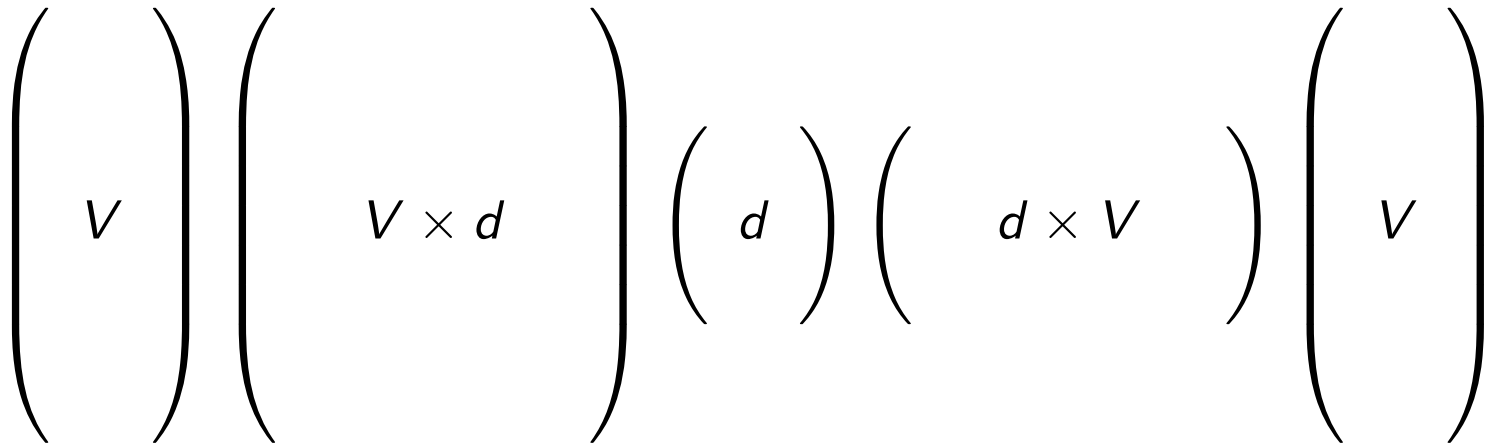
## *More Detailed Architecture (skip-gram)*




*Credits: Xin Rong*

# Prediction-based Embeddings

## *More Detailed Architecture (schematic matrix visualisation)*

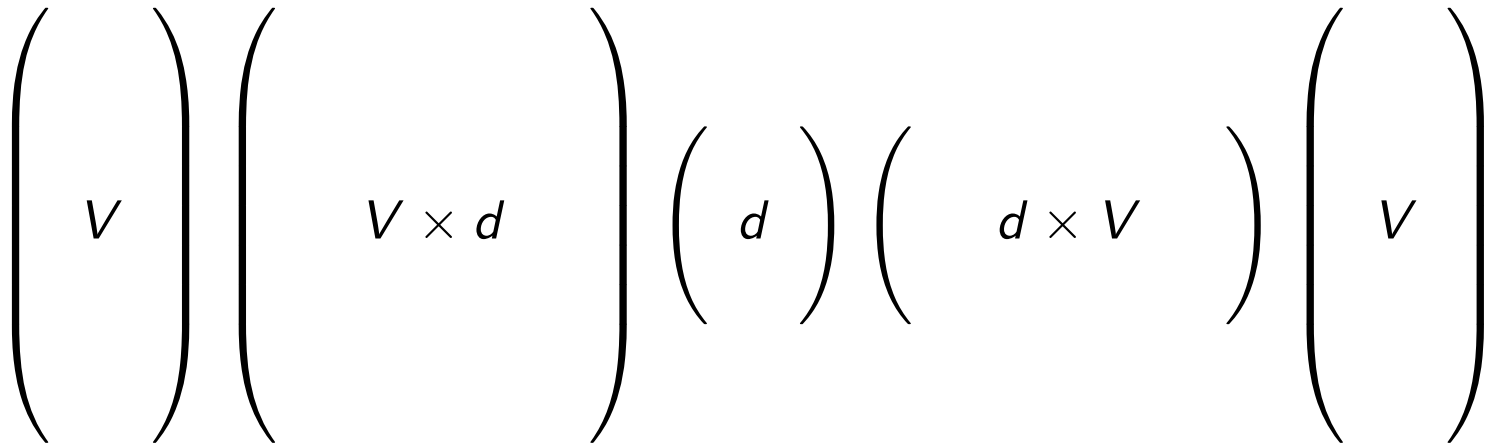



**Input Embedding**

The row *i* of the input matrix $\mathrm{W}$ is the $1 \times \mathrm{d}$ for word *i* in the vocabulary

# Prediction-based Embeddings

## *More Detailed Architecture (schematic matrix visualisation)*




**Output Embedding**

The column $j$ of the output matrix $\mathrm{W}'$ is the $\mathrm{d} \times 1$ for word $j$ in the vocabulary

# Prediction-based Embeddings

## *Observations (Tensorflow Tutorial)*

**CBoW**

- Smoothes over a lot of the distributional information by treating an entire context as one observation. This turns out to be a useful thing for **smaller datasets**

**Skip-gram**

- Treats each context-target pair as a new observation, and this tends to do better when we have **larger datasets**

# Prediction-based Embeddings

## *Let's Play!*

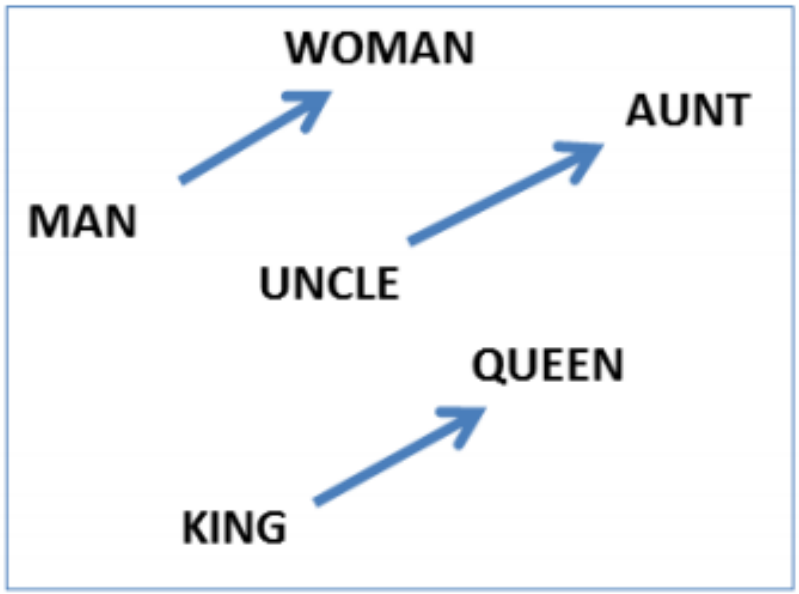
Word Embedding Visual Inspector, wevi

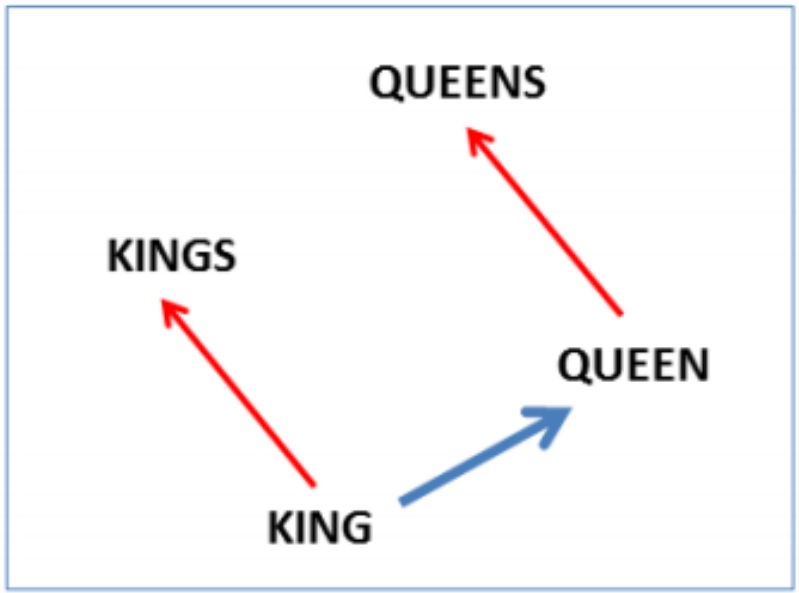
`https://ronxin.github.io/wevi/`

# Prediction-based Embeddings

## *Nice Properties*

**King - Man + Woman = Queen**



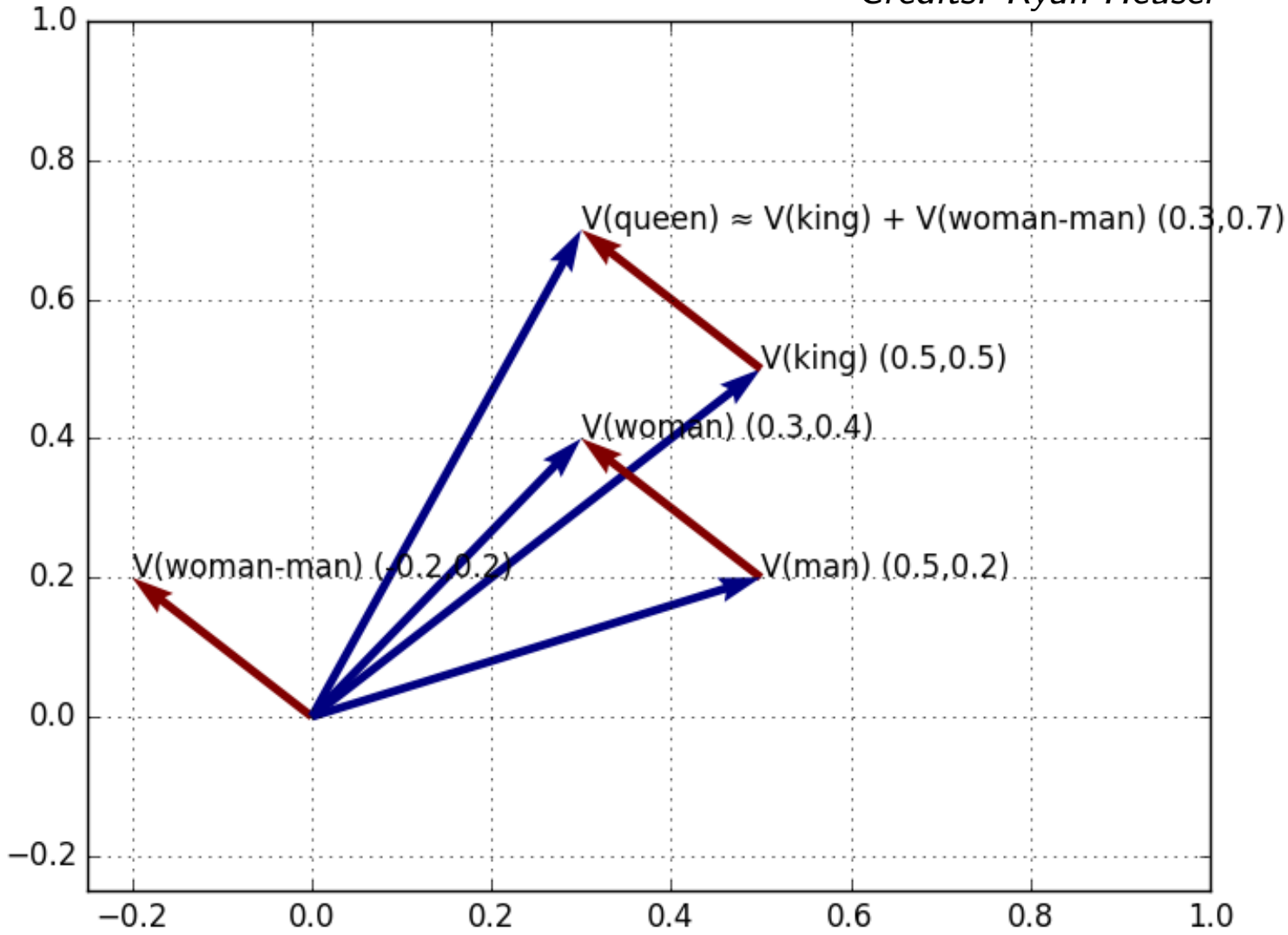




(Mikolov et al., NAACL HLT, 2013)

# Prediction-based Embeddings

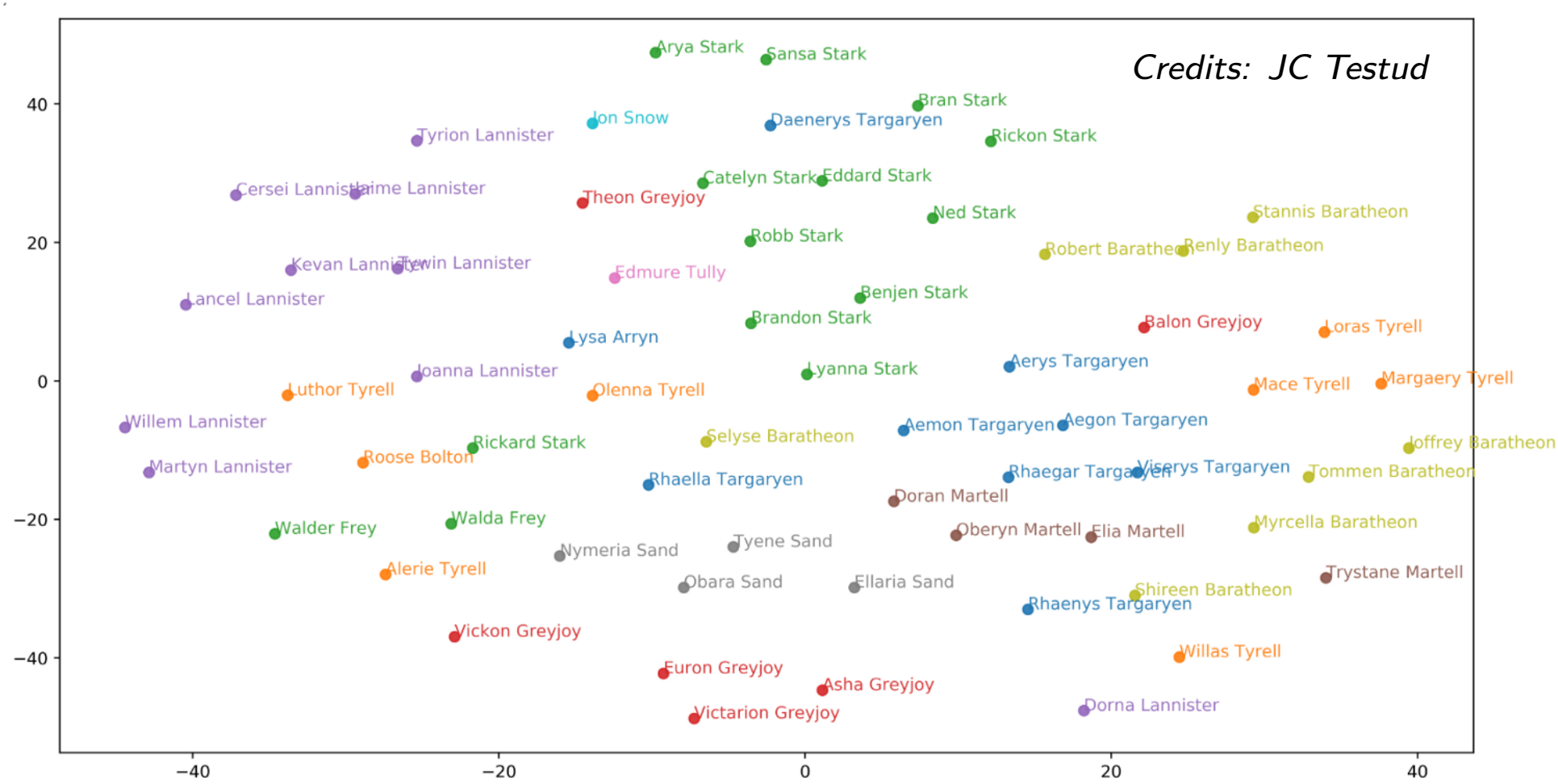
## *Nice Properties*

*Credits: Ryan Heuser*




# Prediction-based Embeddings

## *Nice Properties*

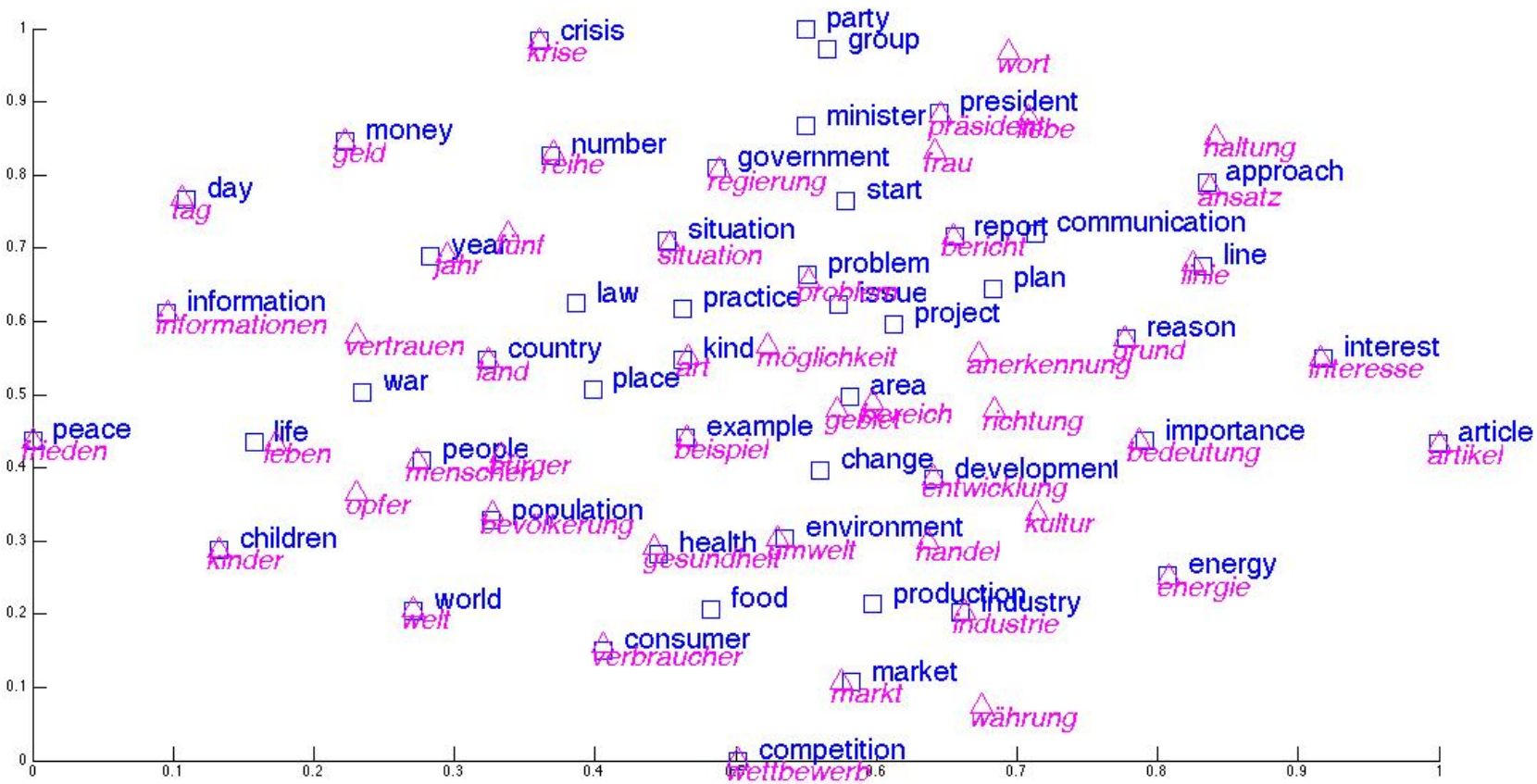



*2D tSNE projection of the main characters of Game of thrones colored by House*

# Prediction-based Embeddings

## *Nice Properties*

*(Luong, Pham & Manning, NAACL, 2015)*




*Barnes-Hut-SNE visualisation of bilingual embeddings German/English*

# Prediction-based Embeddings

## *Let's Explore!*

Embedding Projector

`http://projector.tensorflow.org/`

# Beyond *Word* Embeddings



# Beyond *Word* Embeddings

## *Is Language Compositional?*

**The meaning of a compound expression is a function of the meanings of its parts and of the way they are syntactically combined.**

**(Partee, 1984)**

# Beyond *Word* Embeddings

## *Is Language Compositional?*

Computer Scientist-like Background Yes!

$$\texttt{meaning}(\text{Eat the icecream}) =$$
$$\texttt{meaning}(\text{Eat}) + \texttt{meaning}(\text{the}) + \texttt{meaning}(\text{icecream})$$

# Beyond *Word* Embeddings

## *Is Language Compositional?*

Computer Scientist-like Background Yes!

$$\texttt{meaning}(\text{Eat the icecream}) = \texttt{meaning}(\text{Eat}) + \texttt{meaning}(\text{the}) + \texttt{meaning}(\text{icecream})$$

Linguist-like Background No!

$$\texttt{meaning}(\text{Break the ice}) \neq \texttt{meaning}(\text{Break}) + \texttt{meaning}(\text{the}) + \texttt{meaning}(\text{ice})$$

# Beyond *Word* Embeddings

## *Is Language Compositional?*

**Computer Scientist-like Background** Yes!

$$\texttt{meaning}(\text{Eat the icecream}) = \texttt{meaning}(\text{Eat}) + \texttt{meaning}(\text{the}) + \texttt{meaning}(\text{icecream})$$

Linguist-like Background No!

$$\texttt{meaning}(\text{Break the ice}) \neq \texttt{meaning}(\text{Break}) + \texttt{meaning}(\text{the}) + \texttt{meaning}(\text{ice})$$

# Beyond *Word* Embeddings

## *Representations for Phrases, Sentences or Paragraphs*

- **Composition** of word embeddings using operations $(+, \times)$ on vectors and matrices
- **Latent paragraph vectors** in word2vec-like NNs
- Internal representations in **seq2seq** architectures or auto-encoders (NMT context vectors, skip-thought vectors...)
- **Contextual** word vectors (LM by-products for instance)

# Beyond *Word* Embeddings

## *Composition I*

*(Mitchell & Lapata, 2010)*

Table 5
Composition functions considered in our experiments

| Model | Function |
|---|---|
| Additive | $p_i = u_i + v_i$ |
| Kintsch | $p_i = u_i + v_i + n_i$ |
| Multiplicative | $p_i = u_i \cdot v_i$ |
| Tensor product | $p_{i,j} = u_i \cdot v_j$ |
| Circular convolution | $p_i = \sum_j u_j \cdot v_{i-j}$ |
| Weighted additive | $p_i = \alpha v_i + \beta u_i$ |
| Dilation | $p_i = v_i \sum_j u_j u_j + (\lambda - 1) u_i \sum_j u_j v_j$ |
| Head only | $p_i = v_i$ |
| Target unit | $p_i = v_i (t_1 t_2)$ |

# Beyond *Word* Embeddings

## *Composition II*

*(Mitchell & Lapata, 2010)*

Table 6
Correlation coefficients of model predictions with subject similarity ratings (Spearman's $\rho$) using a simple semantic space

| Model | Adjective–Noun | Noun–Noun | Verb–Object |
|---|---|---|---|
| Additive | .36 | .39 | .30 |
| Kintsch | .32 | .22 | .29 |
| Multiplicative | .46 | .49 | .37 |
| Tensor product | .41 | .36 | .33 |
| Convolution | .09 | .05 | .10 |
| Weighted additive | .44 | .41 | .34 |
| Dilation | .44 | .41 | .38 |
| Target unit | .43 | .34 | .29 |
| Head only | .43 | .17 | .24 |
| Humans | .52 | .49 | .55 |

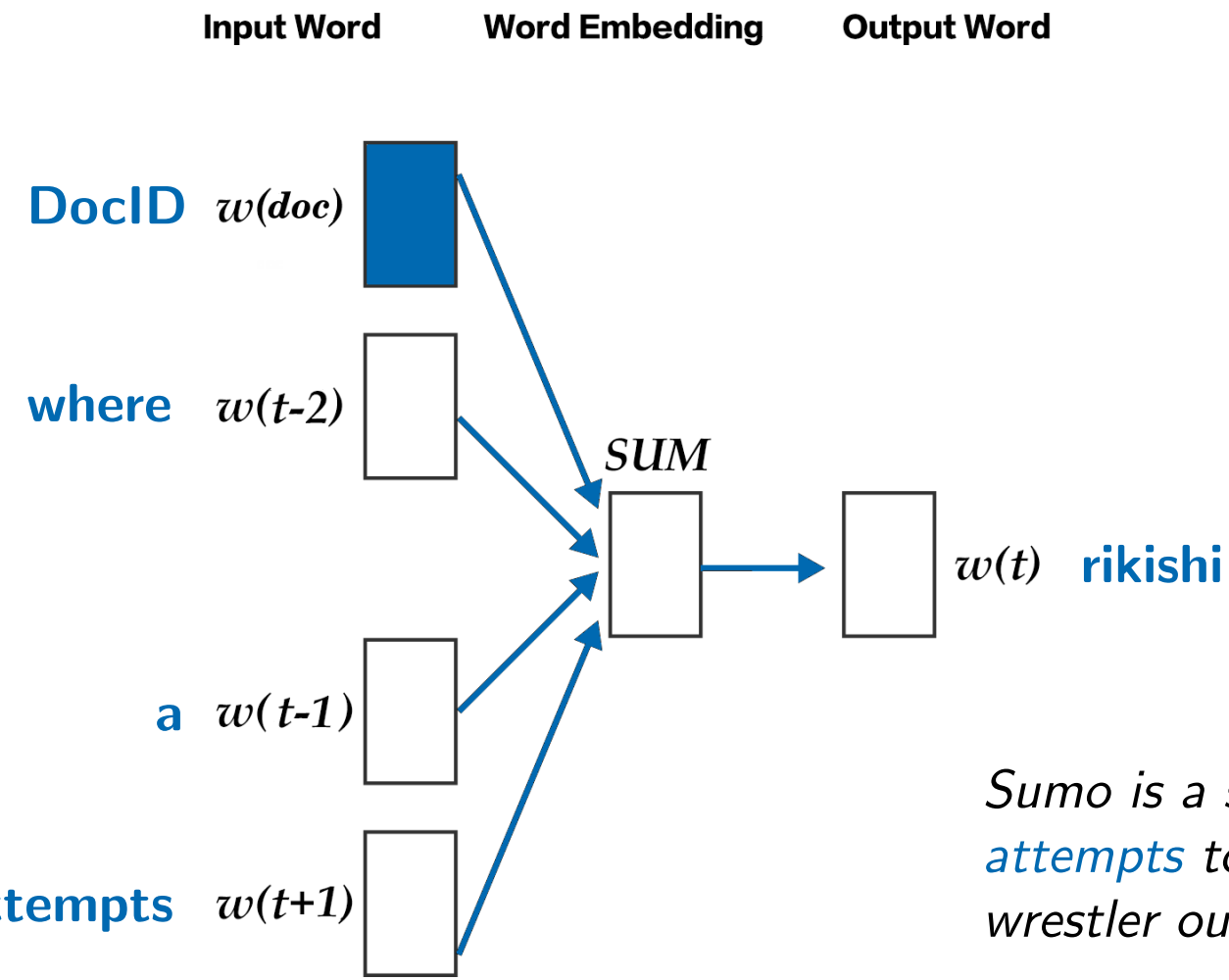
# Beyond *Word* Embeddings

## *Distributed Representations of Sentences and Documents*

- word2vec-like architecture where a **document vector** is added to word vectors and learned simultaneously (PV-DM)
- **Generation** of words from a document from its document vector (skip-gram-like architecture!) (PV-DBoW)

# Beyond *Word* Embeddings

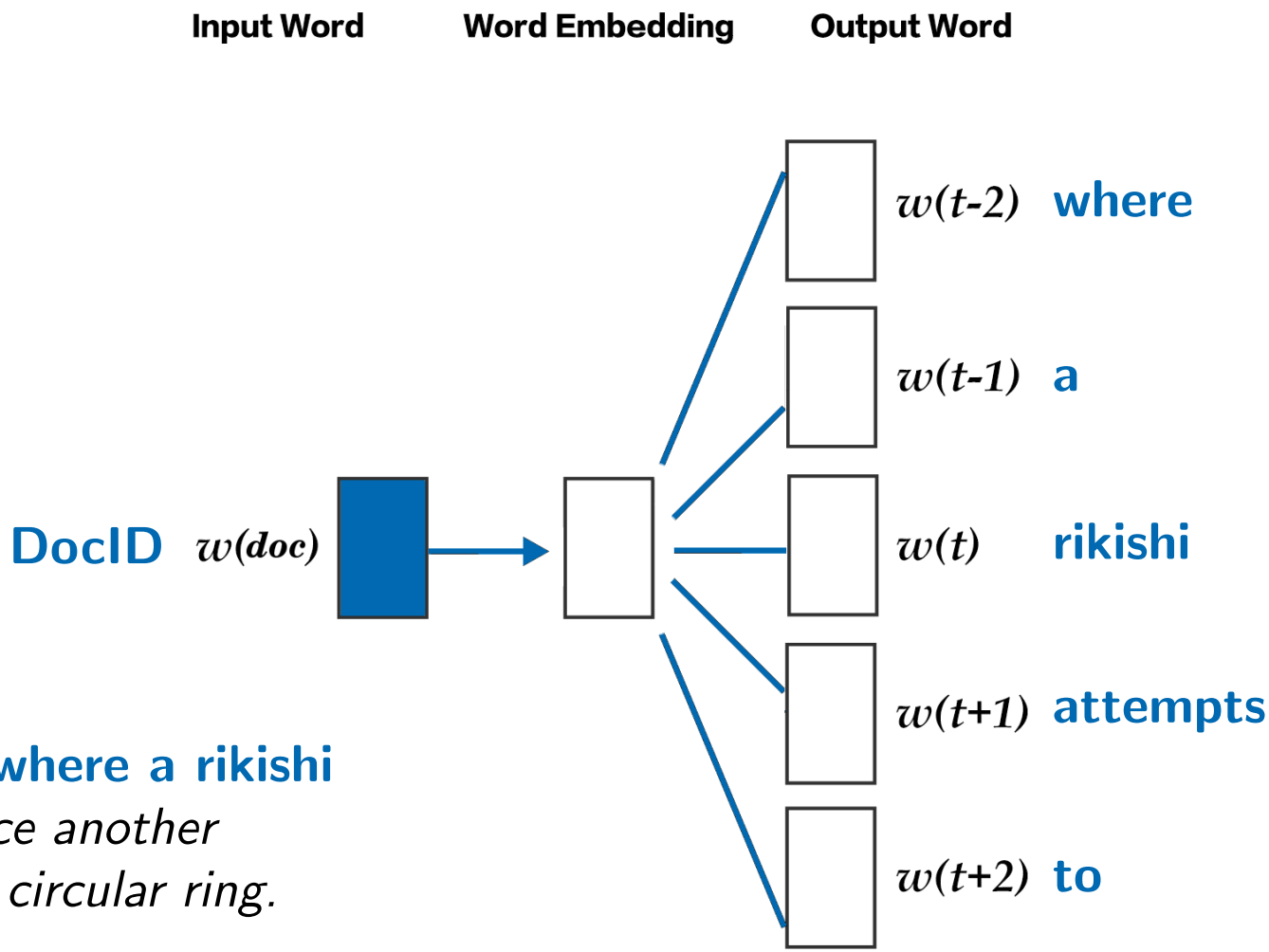
## *Distributed Memory Model of Paragraph Vectors, PV-DM*




*Sumo is a sport where a* **rikishi** *attempts to force another wrestler out of a circular ring.*

# Beyond *Word* Embeddings

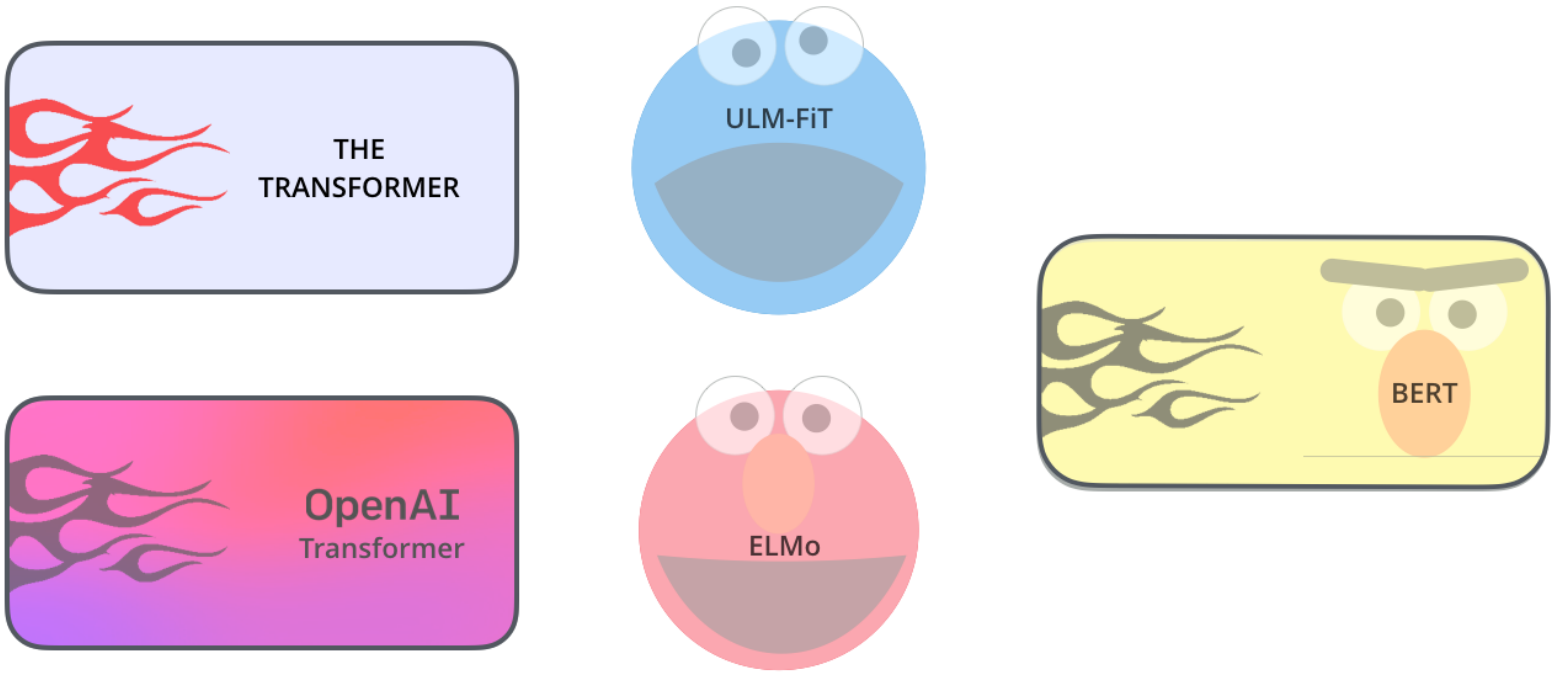
## *Distributed Bag of Words version of Paragraph Vector*




*Sumo is a sport* **where a rikishi attempts to** *force another wrestler out of a circular ring.*

# Beyond *Word* Embeddings

## *Language Model as Complementary Task for Contextual Vectors*




http://jalammar.github.io/illustrated-bert/

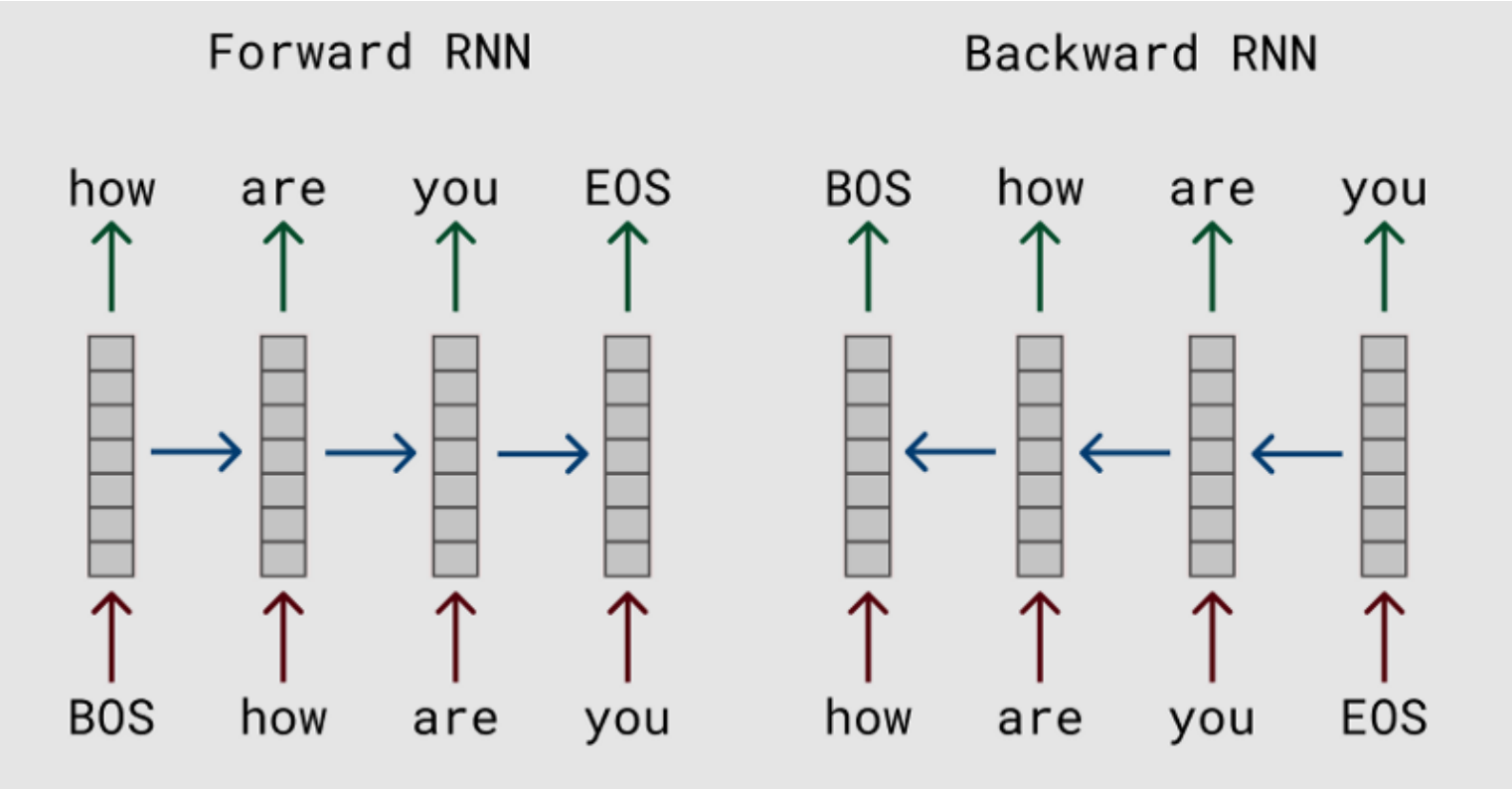
# Beyond *Word* Embeddings

## *Language Model as Complementary Task for Contextual Vectors*

- Deep **contextualised** word representations
- Mainly based on LSTMs or transformer architectures
- Example: ELMo, Embeddings from Language Models
- **Bilingual language models** with LSTMs

# Beyond *Word* Embeddings

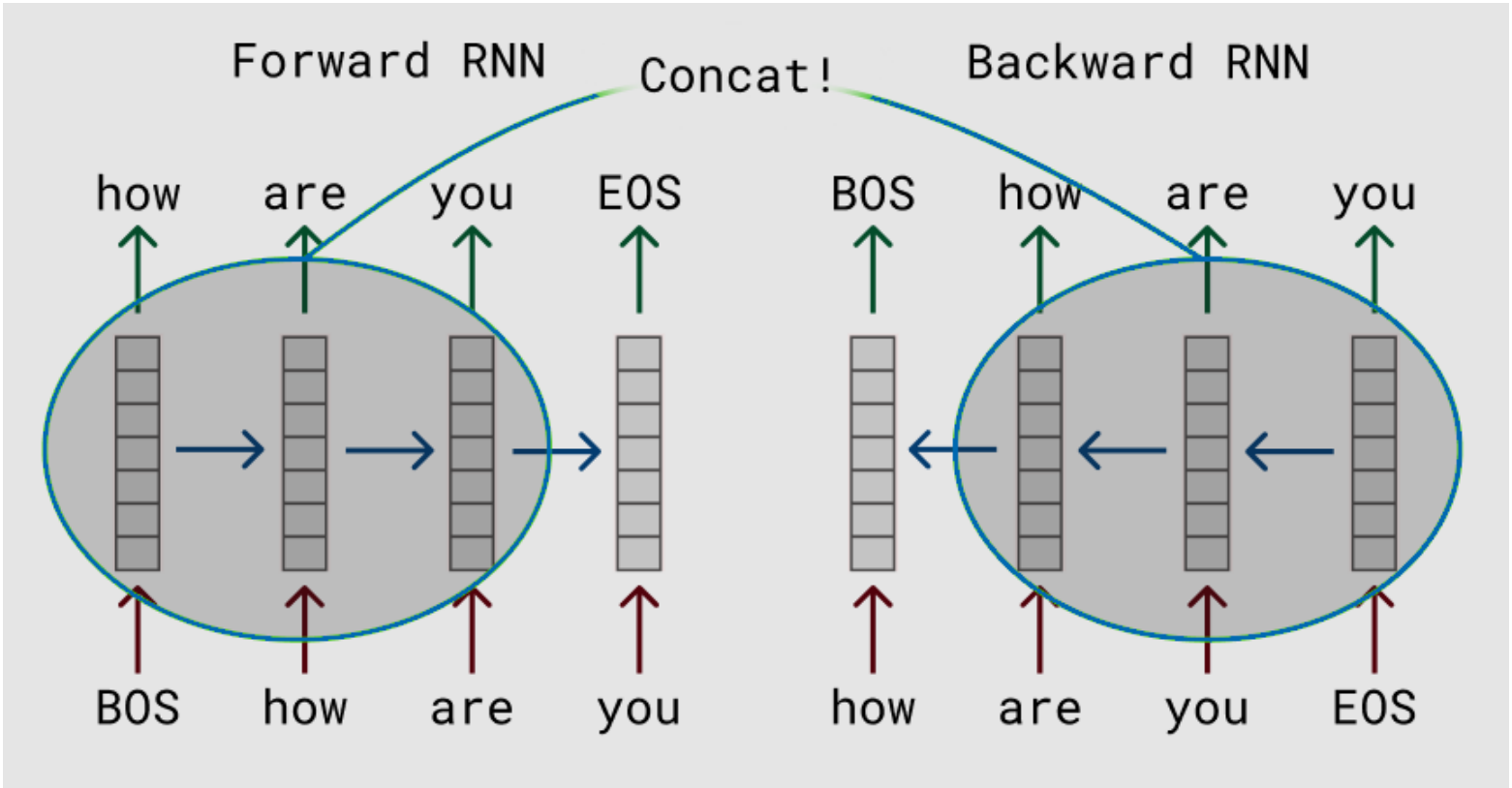
## *Language Model as Complementary Task for Contextual Vectors*




https://medium.com/@plusepsilon/
the-bidirectional-language-model-1f3961d1fb27

# Beyond *Word* Embeddings

## *Language Model as Complementary Task for Contextual Vectors*




https://medium.com/@plusepsilon/
the-bidirectional-language-model-1f3961d1fb27

# Beyond *Word* Embeddings

## *Language Model as Complementary Task for Contextual Vectors*

Word vectors and contextual word vectors are used differently:

- **Word vectors**: dictionary look-up of words and their corresponding vectors, they are static entities
  - Good to initialise input word embeddings in several NLP tasks

# Beyond *Word* Embeddings

## *Language Model as Complementary Task for Contextual Vectors*

Word vectors and contextual word vectors are used differently:

- **Word vectors**: dictionary look-up of words and their corresponding vectors, they are static entities
  - Good to initialise input word embeddings in several NLP tasks
- **Contextual word vectors**: vectors on-the-fly by passing text through a deep learning model, they would only be static if we could generate all sentences in a language!
  - Good for transfer learning into several NLP tasks (SotA in lots of tasks, more in the **LM lecture**!)

# Summary



# Summary

## *Assuming Contextuality Holds...*

- A word can be represented by a vector that describes it with respect to the context it is usually used
- This vector can be estimated by **counts** in a corpus
- This vector can be **learned** by examples in a corpus
- In fact, skipgram and co-occurence matrix factorisation are equivalent under certain conditions
- They are very useful to characterise text, but we haven't talked about **ambiguity** problems for instance

# Summary

## *Assuming Compositionality Holds...*

- A sentence can be decomposed into the vectors of its constituents
- Simple operations such as **sum** and **product** work surprisingly well
- Sentence/Document vectors can also be **learned** in more general tasks such as language modeling or translation
- They are very useful to characterise text, but we haven't talked about **idioms** or **negation** problems for instance

# Summary

## *Can you Answer now the Initial Basic Questions?*

*flies* = (0.101159, 0.550446, 0.543801, -0.973852, -0.680835, 0.417193, -0.247181, 0.209725, -1.136055, -0.059531, -0.401640, 0.171540, 0.925121, -0.143815, 0.781714, -1.482425, 0.347008, -0.112342, 0.442418, -1.020457, -0.071752, 1.873548, -0.222886, -0.729569, -0.830224, -0.868407, 0.203496, 0.469911, -0.191363, 0.565102, 0.687738, 0.480823, 0.842358, -0.173656, -0.265585, 0.685740, 0.488047, -0.359772, -0.576064, -0.802884, 0.081554, 0.046882, -0.861532, -0.461855, 0.613098, -1.534642, -0.884534, 0.207728, 1.396512, -0.242900, -0.383959, 0.570844, -0.703350, -1.368813, -1.008194, 1.534660, 0.171693, 0.640925, -0.233116, 0.324685, 0.483171, 0.337947, -0.963290, -0.400558, 0.830977, 0.913474, 0.251693, -0.589420, -0.299622, 1.047515, -0.266679, -1.247186, 1.087610, -0.549028, 1.600710, -1.538516, -1.703301, -1.393499, -0.894448, 0.717204, 0.105767, -0.189234, -0.615609, -0.658315, 0.051877, 0.014180, -0.791282, 0.150424, 1.343751, -0.464859, 0.871426, 1.542864, -1.202150, -0.767113, -1.734738, 0.073633, -1.012583, 0.747787, 0.476070, -0.454807, 0.642685, -0.854152, -0.071798, 0.233724, 0.712329, -0.097752, -0.531132, 0.323271, -0.447342, 0.657913, 1.199492, -0.107360, -0.154234, -1.131168, 1.354793, 1.721385, -0.240023, 0.655765, -0.217006, -0.801722, 0.553369, 0.213377, 0.323267, -1.516051, 2.106244, -0.134282, 0.742155, 0.426344, 0.197991, -0.806768, 0.372546, -0.160200, -1.552847, -0.286178, -0.707796, 0.527352, -0.259658, 0.230387, 0.105294, -0.194481, 0.301772, -1.022163, 0.557191, 1.096709, 0.058422, -1.036384, 0.353412, -0.623097, -0.689515, 0.091472, 0.783885, 0.184088, -0.367950, 0.952462, 0.183704, 0.677562, 0.293917, -0.214309, -0.487794, 0.934296, 0.311513, 0.286514, -0.085511, 0.777691, 1.232603, -0.309367, -0.225086, 0.005091, -0.099195, -0.293117, 1.305563, 0.595816, 0.950316, 0.568706, -0.561446, 0.911634, -0.383941, 0.758054, -0.197820, 0.506777, -0.290767, -0.356727, 1.229474, -0.156489, -0.782741, -0.210163, -0.029169, 0.602664, 0.418375, 0.148975, -0.761796, 1.322690, -0.173410, 0.204111, -1.344531, 1.081905, -0.660543, -0.225615, -0.444753, -0.929671, 0.054136, 0.052031, -0.164926, 0.159312, -1.316333, 0.837011, -1.290353, 0.958403, 1.247478, 0.442009, 0.455497, -1.856268, -0.358823, -0.230839, -0.206271, 0.227012, -0.454163, 0.747798, -1.252855, 1.436849, -0.427915, -0.810428, -0.628144, -0.288458, 0.087355, 0.356739, 0.153036, 0.516594, -0.504978, 0.814432, 1.052940, 1.094526, -0.219595, 0.722178, 0.267325, -0.087458, -1.270262, -0.039461, 0.991926, -0.112005, -0.009605, 0.149920, 0.164717, 0.280475, 0.966384, 0.327598, 0.189590, -0.208946, 0.838261, 0.051847, -0.277932, -0.788527, -0.768702, -1.688721, 0.388215, 0.170153, -0.555723, -0.529565, -0.528982, -0.659930, 0.588041, -0.368195, -0.850188, -0.004996, 0.925476, 1.046587, -0.731761, 0.519435, 0.193188, -0.709557, 0.123329, -0.454316, 1.885830, -0.201841, -0.728933, -0.953455, -0.205837, -0.724068, 0.120158, 1.765389, -0.192159, 1.062490, -0.002634, 0.125790, -0.846565, 0.548899, -1.062821, -2.146826, 0.134681, 0.570950, 0.851783, 0.436544, 0.688986, 1.229008, 1.435449, 0.118766, -0.132411, 2.527890, 0.778142, 0.269093)

# Summary

## *Can you Answer now the Initial Basic Questions?*

- How can we obtain those numbers?
  - ✓ Co-occurrences in a corpus by either frequency counts or machine learning and dimensionality reduction

# Summary

## *Can you Answer now the Initial Basic Questions?*

- How can we obtain those numbers?
  - ✓ Co-occurrences in a corpus by either frequency counts or machine learning and dimensionality reduction
- What's word2vec?
  - ✓ A framework to learn embeddings in a couple of artificial tasks using a simple feed-forward network

# Summary

## *Can you Answer now the Initial Basic Questions?*

- How can we obtain those numbers?
  - ✓ Co-occurrences in a corpus by either frequency counts or machine learning and dimensionality reduction
- What's word2vec?
  - ✓ A framework to learn embeddings in a couple of artificial tasks using a simple feed-forward network
- Is it the only way to obtain those numbers?
  - ✓ Nope! We have seen that also simple counts work well, but we haven't talk about other models such as GloVe. More in the following lectures and in the research talk.

# Summary

## *Can you Answer now the Initial Basic Questions?*

- Do the vectors (and components!) have any semantic meaning?
  - ✓ Mmmm... we should talk more about this. For today, let's say they have very nice general semantic properties and are useful for many NLP tasks

# Summary

## *Can you Answer now the Initial Basic Questions?*

- Do the vectors (and components!) have any semantic meaning?
  - ✓ Mmmm... we should talk more about this. For today, let's say they have very nice general semantic properties and are useful for many NLP tasks

- Are we crazy by summing or multiplying words to get the meaning of a larger unit?
  - ✓ Yes, probably a bit... But, hey, it also works! We can use them in many NLP, but nowadays contextual embeddings perform better

# Software & References



# Software & References

## *Libraries & Packages*

- **word2vec**
  `https://github.com/dav/word2vec`
- **fastText**
  `https://github.com/facebookresearch/fastText`
- **Gensim**
  `https://radimrehurek.com/gensim/models/word2vec.html`
- **GloVe**
  `https://nlp.stanford.edu/projects/glove/`

# Software & References

## *Libraries & Packages*

- **ELMo**
  `https://github.com/allenai/allennlp/blob/master/allennlp/modules/elmo.py`
- **Open AI**
  `https://github.com/openai/gpt-2`
- **BERT**
  `https://github.com/google-research/bert`
- **InferSent**
  `https://github.com/facebookresearch/InferSent`

# Software & References

## *Basic References*


- Stephen Clark. 2015. **Vector Space Models of Lexical Meaning**. *Handbook of Contemporary Semantic Theory*. Second edition, edited by Shalom Lappin and Chris Fox. Chapter 16. Pages 493–522. Wiley-Blackwell.

- Jeff Mitchell and Mirella Lapata. 2010. **Composition in distributional models of semantics**. *Cognitive Science, 34(8)*. Pages 1388–1429.

- Daniel Jurafsky and James H. Martin. 2017. **Speech and Language Processing.** Chapter 16: Semantics with Dense Vectors.

- Xin Rong. 2015. **word2vec Parameter Learning Explained**. *eprint arXiv:1411.2738*.


# Software & References

## *Basic References II*


- Tomas Mikolov, Kai Chen, Greg Corrado and Jeffrey Dean. 2013. **Efficient Estimation of Word Representations in Vector Space**. *Proceedings of the Workshop at International Conference on Learning Representations (ICLR)*. Pages 1–12.

- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. **GloVe: Global Vectors for Word Representation**. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Pages 1532–1543

- Quoc V. Le and Tomas Mikolov. 2014. **Distributed Representations of Sentences and Documents**. *Proceedings of the 31st International Conference on Machine Learning (ICML)*, in PMLR 32(2). Pages 1188–1196


# Software & References

## *Basic References III*


- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer. 2018. **Deep contextualized word representations**. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Pages 2227–2237.

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. 2018. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**. *https://arxiv.org/abs/1810.04805*.

- Alan Akbik, Duncan Blythe, Roland Vollgraf. 2018. **Contextual String Embeddings for Sequence Labeling**. *Proceedings of the 27th International Conference on Computational Linguistic)*. Pages 1638–1649.


# Everything clear? Too much information?

**Questions?**

# Introduction to Word Embeddings

**Cristina España-Bonet**

**UdS & DFKI, Saarbrücken, Germany**

Artificial Intelligence with Deep Learning

23rd April 2019

# Extra Slides

## *Singular-Value Decomposition, SVD*

- Linear algebra

# Extra Slides

## *Singular-Value Decomposition, SVD*

- Linear algebra
- **Factorisation** of a matrix $\mathbf{M}$ as $\mathbf{M} = \mathbf{U\Sigma V^T}$

# Extra Slides

## *Singular-Value Decomposition, SVD*

- Linear algebra
- **Factorisation** of a matrix $\mathbf{M}$ as $\mathbf{M} = \mathbf{U\Sigma V^T}$
  - ✓ $\mathbf{U}$ is an $m \times m$ orthogonal matrix,

# Extra Slides

## *Singular-Value Decomposition, SVD*

- Linear algebra
- **Factorisation** of a matrix $\mathbf{M}$ as $\mathbf{M} = \mathbf{U\Sigma V^T}$
  - ✓ $\mathbf{U}$ is an $m \times m$ **orthogonal matrix**,
    - $\mathbf{U^T U} = \mathbf{U U^T} = \mathbf{I}$
    - or, equivalently, $\mathbf{U^T} = \mathbf{U^{-1}}$

# Extra Slides

## *Singular-Value Decomposition, SVD*

- Linear algebra
- **Factorisation** of a matrix $\mathbf{M}$ as $\mathbf{M} = \mathbf{U\Sigma V^T}$
  - ✓ $\mathbf{U}$ is an $m \times m$ orthogonal matrix,
  - ✓ $\mathbf{\Sigma}$ is a diagonal $m \times n$ matrix with non-negative real numbers,

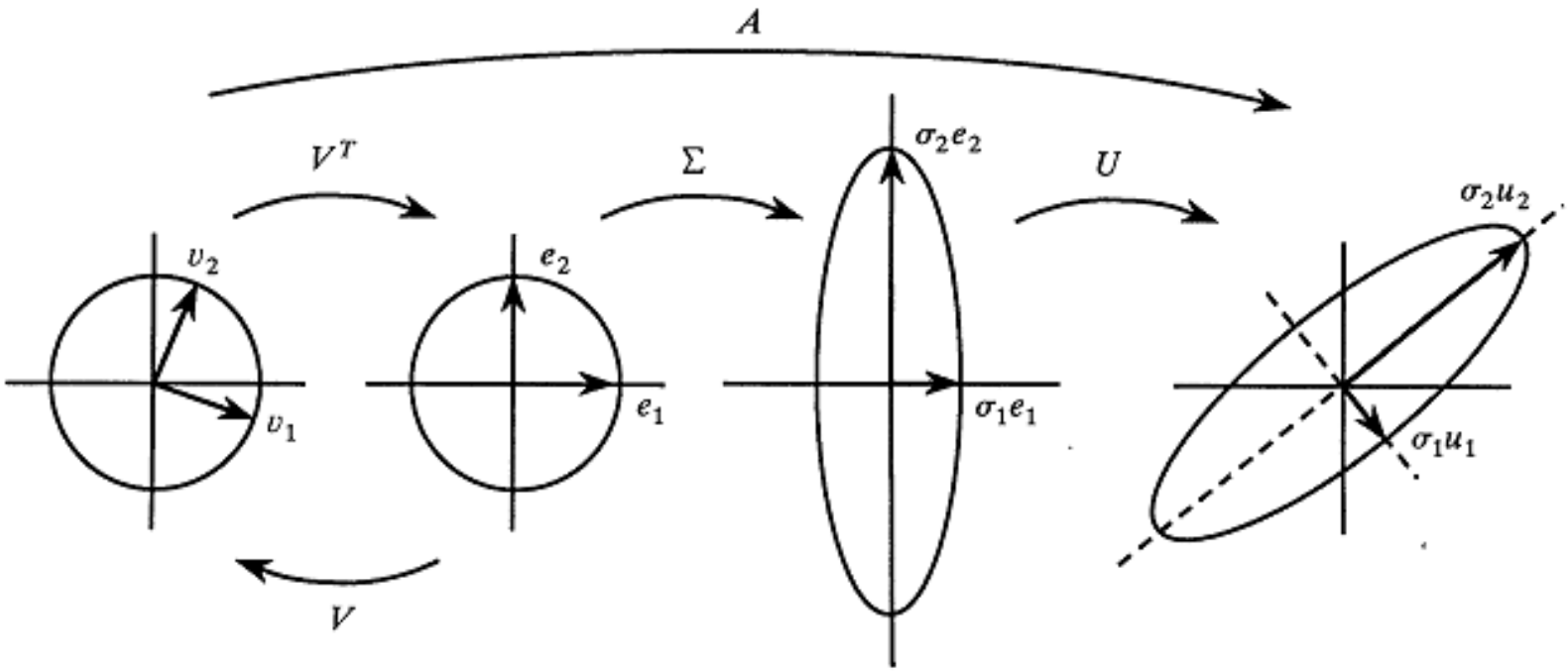
# Extra Slides

## *Singular-Value Decomposition, SVD*

- Linear algebra
- **Factorisation** of a matrix $\mathbf{M}$ as $\mathbf{M} = \mathbf{U\Sigma V^T}$
  - ✓ $\mathbf{U}$ is an $m \times m$ orthogonal matrix,
  - ✓ $\mathbf{\Sigma}$ is a diagonal $m \times n$ matrix with non-negative real numbers,
  - ✓ $\mathbf{V^T}$ is the conjugate transpose of an $n \times n$ orthogonal matrix

# Extra Slides

## *SVD: $2 \times 2$ Geometric Interpretation*




a linear transformation is a rotation or reflection, followed by a scaling, followed by another rotation or reflection

https://blogs.sas.com

# Extra Slides

## *Singular-Value Decomposition, SVD*

$$\begin{pmatrix} \\ m \times n \\ \\ \end{pmatrix} = \begin{pmatrix} \\ m \times m \\ \\ \end{pmatrix} \begin{pmatrix} \\ m \times n \\ \\ \end{pmatrix} \begin{pmatrix} \\ n \times n \\ \\ \end{pmatrix}$$

$$\mathbf{M} \quad = \quad \mathbf{U} \quad \mathbf{\Sigma} \quad \mathbf{V^T}$$

# Extra Slides

## *SVD: Singular Values*

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & 0 \\ & & \ddots & \\ & 0 & & \sigma_r \\ & & & & 0 \end{pmatrix};$$

$\sigma_1$ ... $\sigma_r$, singular values of $\mathbf{M}$ (in decreasing order)

$r$, rank of $\mathbf{M}$

# Extra Slides

## *SVD: Singular Values*

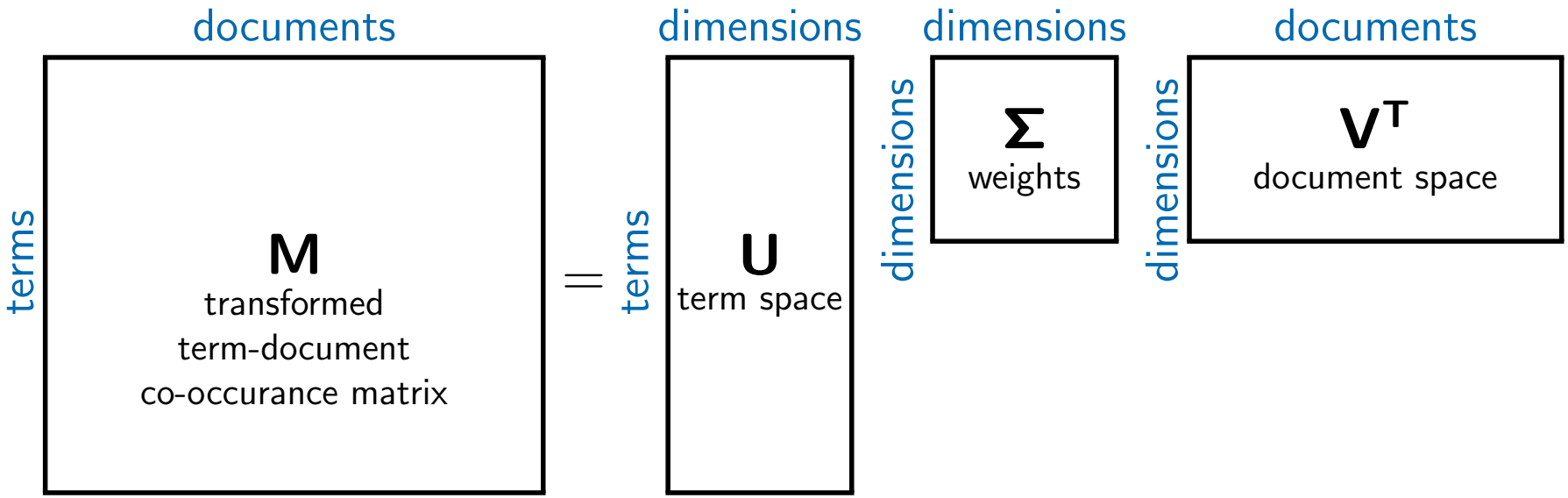
$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & & & \\ & . & & 0 \\ & & . & \\ & 0 & & \sigma_r \\ & & & & 0 \end{pmatrix}; \qquad \mathbf{M_r} = \sum_{i=1}^{r} \sigma_i \overrightarrow{u}_i \overrightarrow{v}_i^T$$

$\sigma_1$ ... $\sigma_r$, singular values of $\mathbf{M}$ (in decreasing order)

$r$, rank of $\mathbf{M}$

# Extra Slides

## *SVD: Application, Latent Semantic Analysis*




$$\Sigma_r \Longrightarrow \mathbf{M}_r$$

# Extra Slides

## *SVD: Learn & Practice*

`https://nlp.stanford.edu/IR-book/pdf/18lsi.pdf`

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. **Introduction to Information Retrieval**. Cambridge University Press, New York, NY, USA.