Identifying and representing words

Benoît Sagot
Inria (ALMAnaCH)

MVA — Speech and Language Processing — Class #4 — 10th February, 2020

Credit and disclaimer: some of the following slides are taken from, illustrated or inspired by presentations and article figures by Jurafsky, Goldberg, Melamud, Mazaré and others.

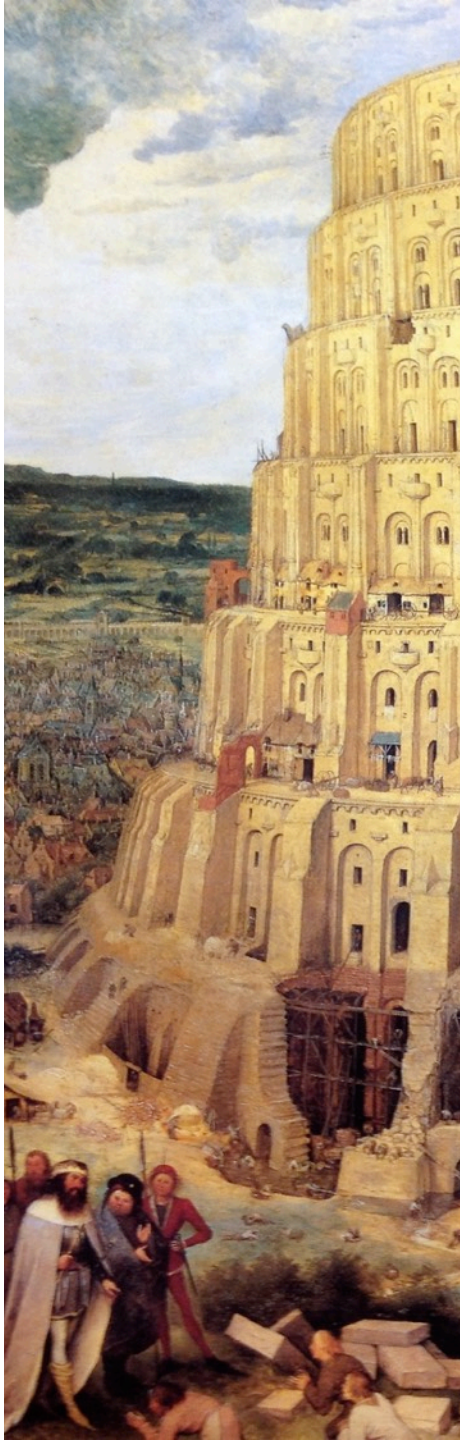
Processing textual data

- A text is a **sequence of characters**
  - letters, ideograms, syllabograms…
  - punctuation marks
  - whitespace characters (not in all writing systems)
- Most Natural Language Processing (NLP) systems rely on a **double structuring** of such a sequence
  - Macroscopic units: “**sentences**” (utterrances, speech turns)
  - Microscopic units: “**words**”
- Typically, sentences are processed individually as sequences of words
- This raises **two challenges**:

Processing textual data

- A text is a **sequence of characters**
  - letters, ideograms, syllabograms…
  - punctuation marks
  - whitespace characters (not in all writing systems)
- Most Natural Language Processing (NLP) systems rely on a **double structuring** of such a sequence
  - Macroscopic units: "**sentences**" (utterrances, speech turns)
  - Microscopic units: "**words**"
- Typically, sentences are processed individually as sequences of words
- This raises **two challenges**:
  - How do we represent words?
  - How do we identify words and sentences in raw texts?

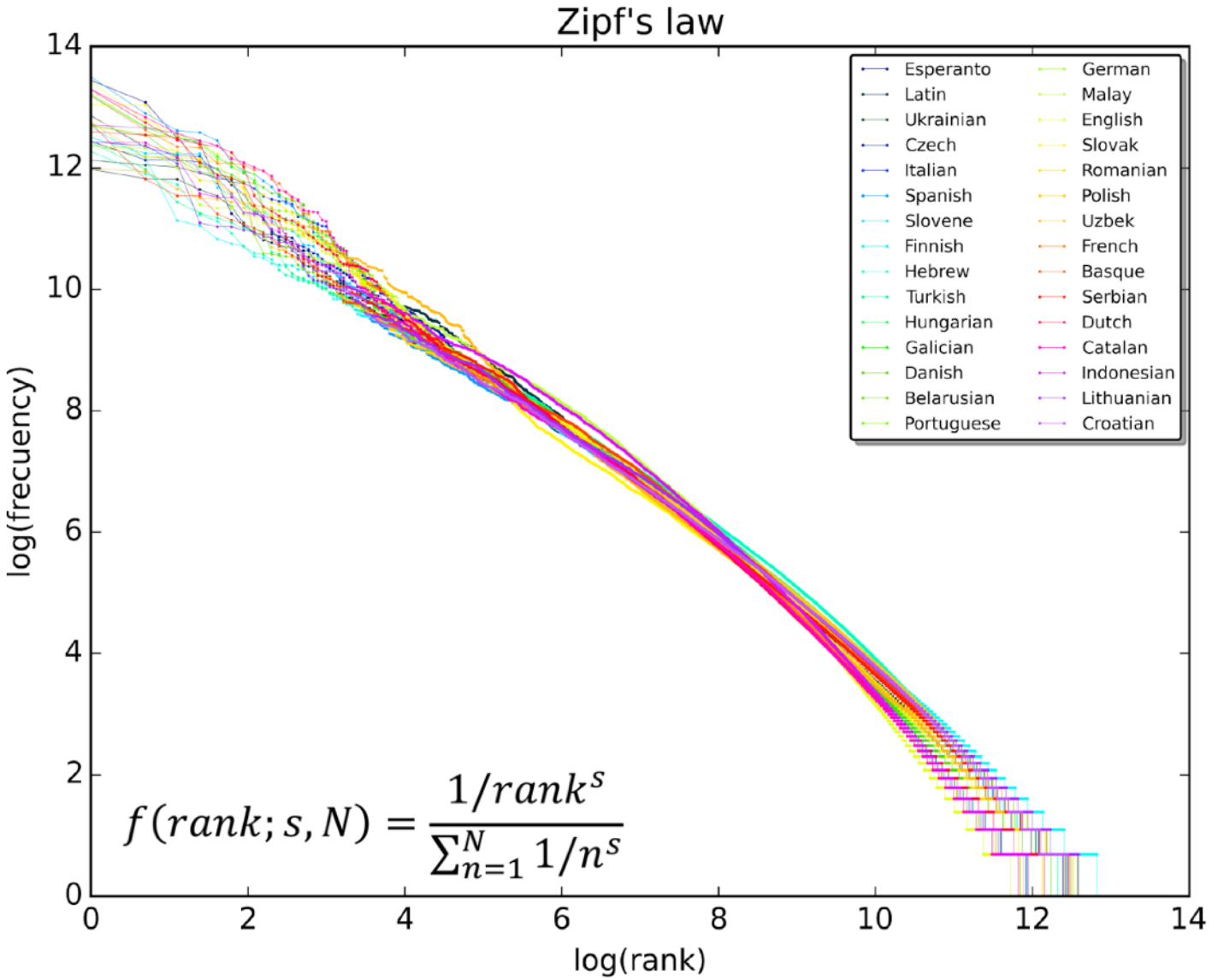
Word representations



Lexical sparsity

A plot of the rank versus frequency for the first 10 million words in 30 Wikipedias

(source: Wikipedia; data: dumps from Oct 2015)



Lexicons and thesauruses

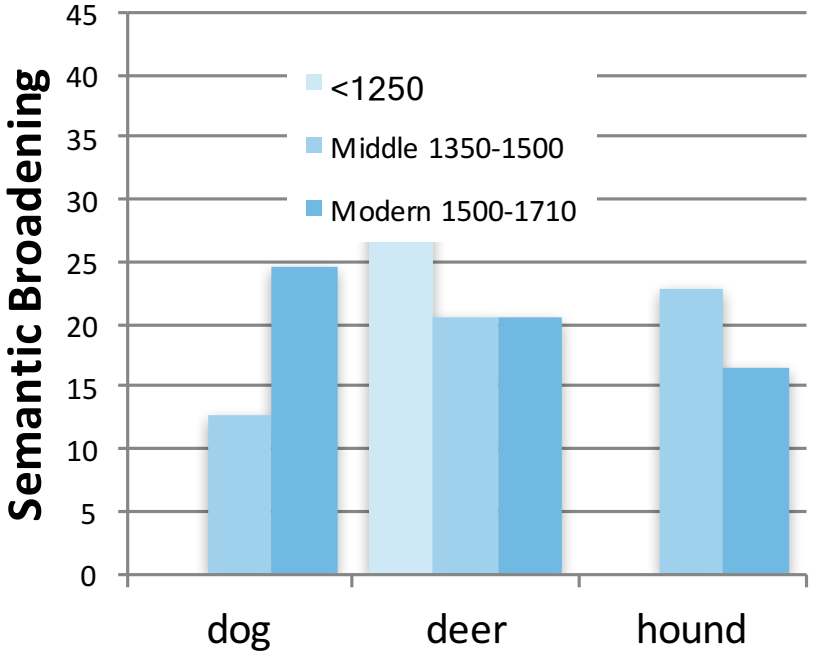
- **Advantages**
  - Possibility to encode rich linguistic information and to cover rare cases not seen in corpora
  - It is another source of linguistic information, next to annotated corpora
- **Drawbacks**
  - Costly to develop, do not exist for all languages
  - Static, fixed meanings and words
  - Limited coverage
  - Structural organisation not always relevant (it is difficult to create a hierarchy of meanings for adjectives and verbs)

Vector representations: what for?

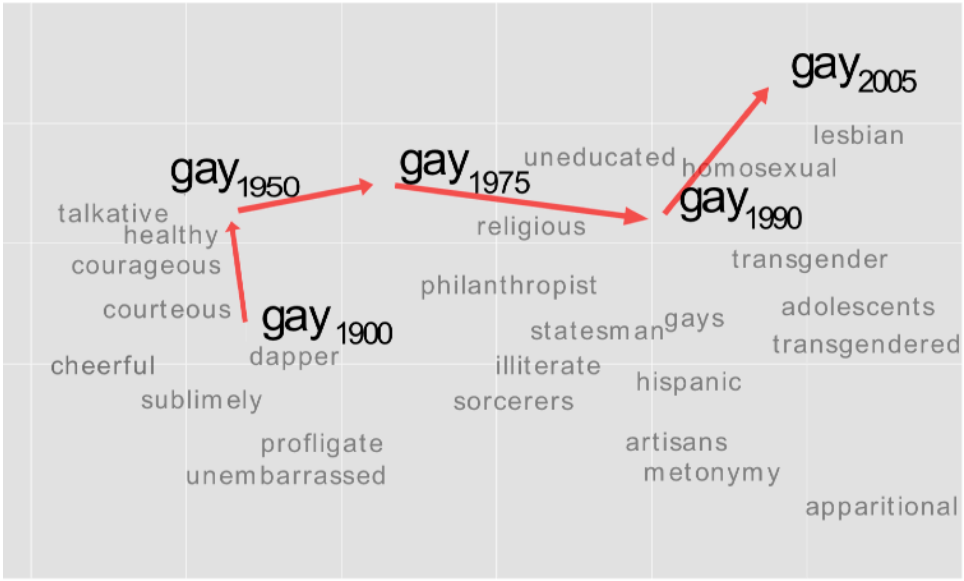
- Question answering:
  - Question: *How **tall** is Mt. Everest?*
    Candidate answer: *The official **height** of Mount Everest is 8848m*
- Plagiarism detection:
  - *Mainframes **are primarily** referred to as **large** computers with **rapid**, advanced processing capabilities that **can execute and** perform tasks **equivalent to many** Personal Computers (PCs)*
  - *Mainframes **usually are** referred to as large computers with **fast**, advanced processing capabilities that **could** execute and perform **by itself** tasks **that may require a lot of** Personal Computers (PCs)*
- We need to be able to measure **word similarity**

Word similarity for historical linguistics: semantic change over time

Sagi *et al.* 2013



Kulkarni *et al.* 2015



Distributional models of meaning

- Zellig Harris (1954):
  - “oculist and eye-doctor … occur in almost the same environments”
  - “If A and B have almost identical environments we say that they are synonyms.”

- Firth (1957):
  - “You shall know a word by the company it keeps!”

- **Distributional hypothesis**

Distributional models of meaning

- Nida example: Suppose I asked you what is tesgüino?

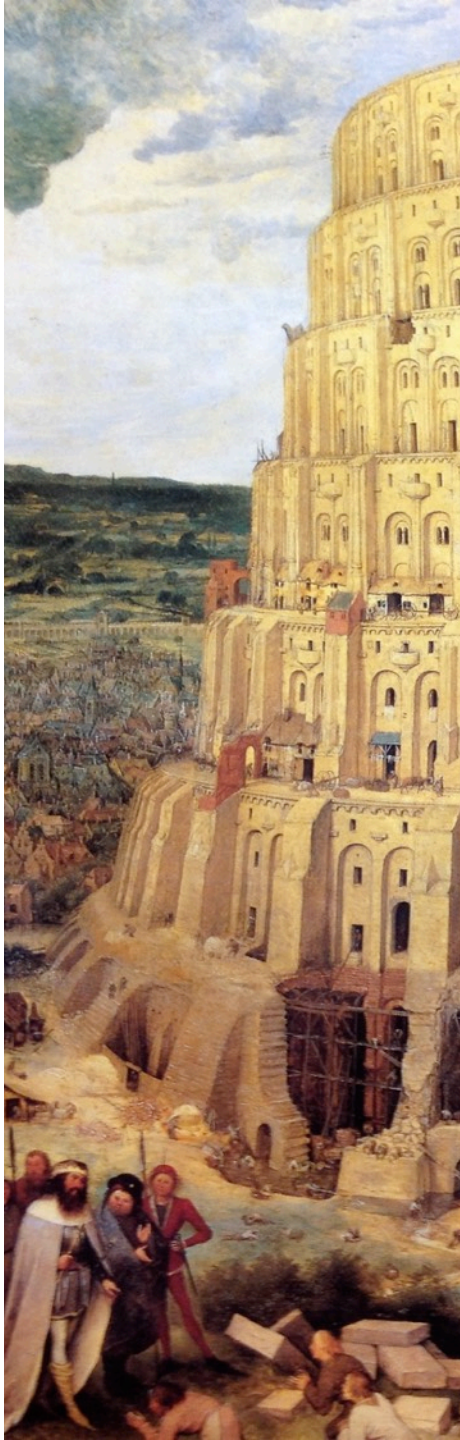
> *A bottle of **tesgüino** is on the table*
> *Everybody likes **tesgüino***
> ***Tesgüino** makes you drunk*
> *We make **tesgüino** out of corn.*

- From context words humans can guess tesgüino means
  - *an alcoholic beverage akin to beer*
- Intuition:
  - **Two words are similar if they have similar word contexts.**

Different kinds of vector models

- **Sparse vector representations**
  1. Mutual-information weighted word co-occurrence matrices
- **Dense vector representations**
  2. Brown clusters
  3. Singular value decomposition (and Latent Semantic Analysis)
  4. Neural-network-inspired models (skip-grams, CBOW)
- **Shared intuition**
  - The meaning of a word is modelled by “embedding” the word in a vector space
  - A meaning is represented as a vector
  - The vector representing a word is called a “**word embedding**”

Word and co-occurrence vectors



Cooccurrence matrices

- **Cooccurrence** = appearing in the same environment
  - document
  - immediate context
- **Cooccurrence matrix** = frequency counts of *(word, environment)* pairs
- Two main categories of cooccurrence matrices
  - **Term-document matrix**: how often word *i* occurs in document *j*
  - **Term-term (word-word, word-context) matrix**: how often word *i* occurs in the immediate vicinity of word *j*

Term-document matrix

- **Each cell: count of word $w \in V$ in a document $d \in D$**

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Term-document matrix

- **Each cell: count of word *w* in a document *d***
  - Each document is a count vector in $\mathbb{N}^{|V|}$ (a column)

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Term-document matrix

- **Each cell: count of word *w* in a document *d***
  - Each document is a count vector in $\mathbb{N}^{|V|}$ (a column)
  - Each word is a count vector in $\mathbb{N}^{|D|}$ (a row)

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Similarity in term-document matrices

- Two documents are similar if their vectors are similar

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Similarity in term-document matrices

- Two documents are similar if their vectors are similar
- Two words are similar if their vectors are similar

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Similarity in word-word matrices

- Instead of entire documents, use **smaller contexts**
  - Paragraph
  - Window of ± 4 words
- A word is now defined by a vector over **counts of context words**
- Instead of each vector being of length $|D|$:
  - Each vector is now of length $|V|$
  - The word-word matrix is $|V|^2$

Word-word matrix

- Example with 7-word contexts

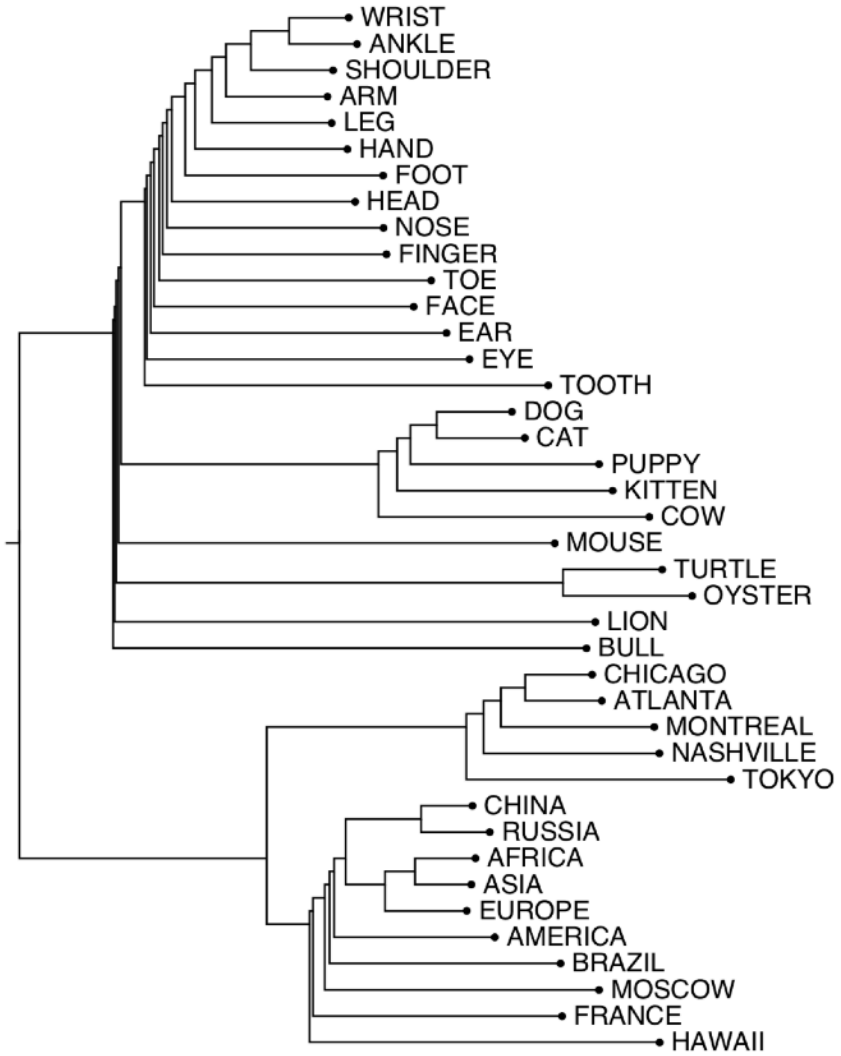
sugar, a sliced lemon, a tablespoonful of **apricot** preserve or jam, a pinch each of,
their enjoyment. Cautiously she sampled her first **pineapple** and another fruit whose taste she likened
well suited to programming on the digital **computer**. In finding the optimal R-stage policy from
for the purpose of gathering data and **information** necessary for the study authorized in the

	aardvark	computer	data	pinch	result	sugar	…	…
apricot	0	0	0	1	0	1		
pineapple	0	0	0	1	0	1		
digital	0	2	1	0	1	0		
information	0	1	6	0	4	0		

Word-word matrix

- We showed only 4x6, but the real matrix is 500,000 x 500,000
  - **Very sparse** (most values are 0)
  - That's acceptable, since there are lots of efficient algorithms for sparse matrices.
- Similarity is measured using the **cosine** between two vectors (i.e. the **dot product** between normalised vectors)
- The **size of context windows** depends on your goals
  - The **shorter** the windows, the more **syntactic** the representation 1-3 ~ syntactic similarity
  - The **longer** the windows, the more **semantic** the representation 4-10 ~ semantic/topical similarity

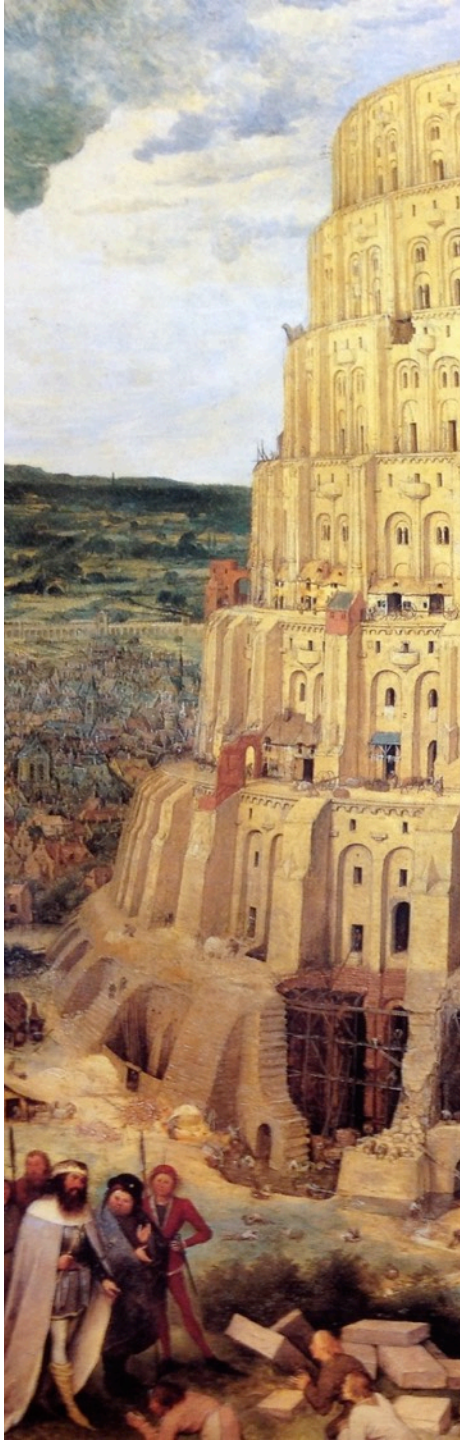
Similarity-based hierarchical clustering



First-order vs. second-order similarity

- First-order co-occurrence (**syntagmatic association**)
  - They are typically close to each other.
  - *wrote* is a first-order associate of *book* or *poem*.
- Second-order co-occurrence (**paradigmatic association**)
  - They have similar neighbours.
  - *wrote* is a second-order associate of words like *said* or *remarked*.

Positive Pointwise Mutual Information



Problems with raw counts

- Raw word frequency is not a great measure of association between words
  - It is very skewed
    - For ex.: “the” and “of” are very frequent, but maybe not the most discriminative

- We would be more interested in a measure that would give more importance to context words that are **more informative** about the target word
  - Positive Pointwise Mutual Information (PPMI)

Pointwise Mutual information (PMI)

- General definition: do events *x* and *y* cooccur more than if they were independent?

$$\text{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- PMI between two words: do words *x* and *y* cooccur more than if they were independent?

$$\text{PMI}(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

- PMIs range from -∞ to +∞

Positive Pointwise Mutual information (PPMI)

- **Negative PMI values are problematic**
  - They correspond to words co-occurring **less** than we expect by chance
  - Unreliable without a really huge corpus
    - Imagine two rare words w_1 and w_2 (say, frequency = 10^{-6}): it is hard to compare P(w_1,w_2) with 10^{-12} and know whether the difference is statistically significative…
  - It is unclear whether we have intuitions about unrelatedness
    - We are interested in similarity (relatedness)
- **We replace negative PMI values by 0 => PPMI**

$$\text{PPMI}(word_1, word_2) = \max\left(\log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}, 0\right)$$

PPMI on an example

Count(w,context)

	computer	data	pinch	result	sugar
apricot	0	0	1	0	1
pineapple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

PPMI(w,context)

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

Weighting PPMI

- **PPMI is biased toward infrequent events**
  - Very rare words have very high PMI values
- Two solutions
  - Transform counts using an exponential (exponent <1) to give rare context words slightly higher counts

$$\text{PPMI}_\alpha(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P_\alpha(c)}, 0)$$

$$P_\alpha(c) = \frac{count(c)^\alpha}{\sum_c count(c)^\alpha}$$

  - Use "**Laplace smoothing**", i.e. **add 1 to all counts** (it has a similar effect)
    - Or **add 2**, etc.

Example with add-2 smoothing

PPMI(w,context)

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

PPMI(w,context) [add-2]

	computer	data	pinch	result	sugar
apricot	0.00	0.00	0.56	0.00	0.56
pineapple	0.00	0.00	0.56	0.00	0.56
digital	0.62	0.00	0.00	0.00	0.00
information	0.00	0.58	0.00	0.37	0.00

Beyond cosine similarity

- Another way to improve over cosine similarity on count matrices is to use **more sophisticated similarity measures**
  - A few examples:

$$\text{sim}_{\text{cosine}}(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}| \cdot |\vec{w}|}$$

$$\text{sim}_{\text{Jaccard}}(\vec{v}, \vec{w}) = \frac{\sum_{i=1}^{N} \min(v_i, w_i)}{\sum_{i=1}^{N} \max(v_i, w_i)}$$

$$\text{sim}_{\text{Dice}}(\vec{v}, \vec{w}) = \frac{2 \sum_{i=1}^{N} \min(v_i, w_i)}{\sum_{i=1}^{N} (v_i + w_i)}$$

Beyond cosine similarity

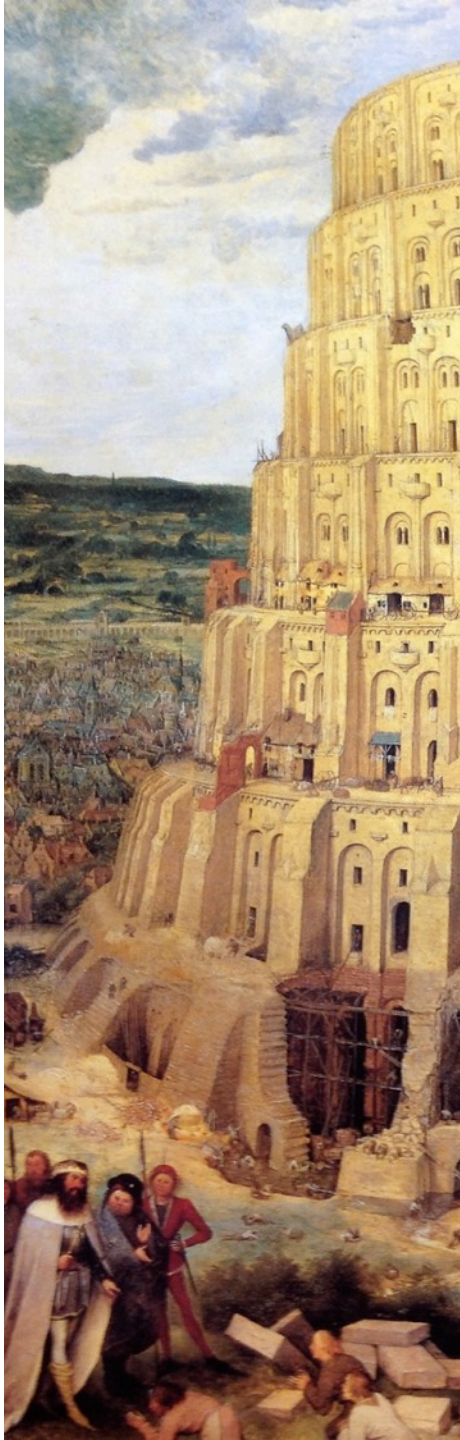
- When working on word-document matrices, the most frequently used transformation is not PPMI but **tf-idf**

$$(\text{tf-idf}_D)_{ij} = (\text{tf}_D)_{ij} \cdot (\text{idf}_D)_{ij}$$

$(\text{tf}_D)_{ij}$ = "term frequency" = #occurrences of the word w_i in document d_j (can be normalised by the document length, binarised, log…)

$(\text{idf}_D)_{ij}$ = "inverse document frequency" = $\log(|D|/\text{df}_i)$, where df_i is the number of documents containing w_i

Building dense vectors using SVD

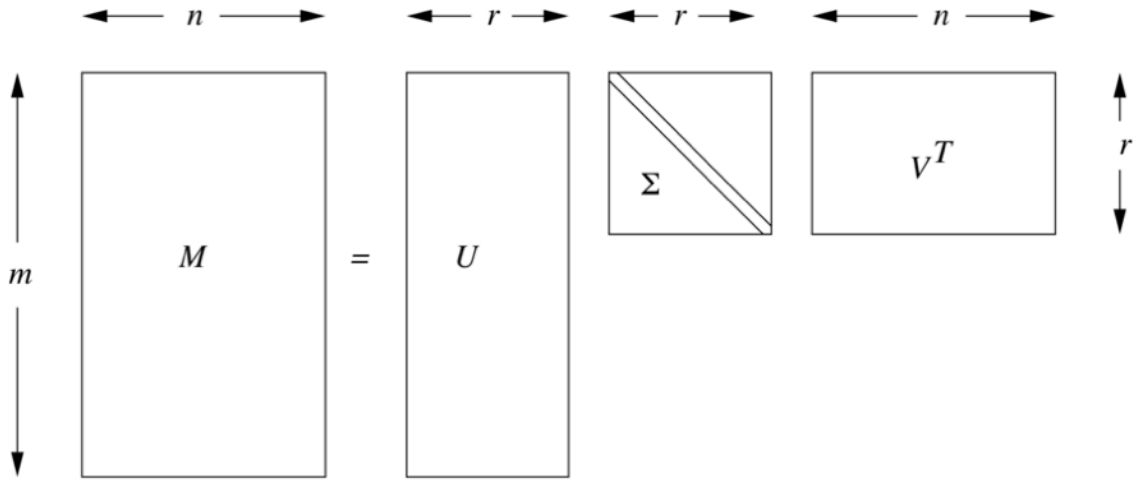


Sparse vs. dense vectors

- **PPMI vectors are sparse**
  - Up to 500,000 dimensions, if not more (depends on the language)
  - Most values are 0
  - Cosine distance not optimal on such vectors, no information is shared between similar words
  - Machine learning systems using such vectors have a large number of weights to train
- **How can we get dense, low-dimensionality vectors?**
  - Many techniques have been proposed. 2 examples:
    - Classical: singular value decomposition (SVD)
    - More recently: side-effect of predictive neural models

Singular Value Decomposition

- **Theorem**
  for any rectangular *m*×*n* real matrix M of rank *r*, there exist
  - an *m*×*r* orthogonal matrix U,
  - an *r*×*r* diagonal matrix Σ with diagonal values ≥ 0 ("**singular values**"),
  - and an *n*×*r* orthogonal matrix V
  - such that **M = U Σ V^T**
- If singular values are sorted in decreasing order (e.g. amount of variance captured by the corresponding dimension), then Σ is unique



Singular Value Decomposition an example

	Matrix	Alien	Star Wars	Casablanca	Titanic
John	1	1	1	0	0
Jack	3	3	3	0	0
Jill	4	4	4	0	0
Jenny	5	5	5	0	0
Jane	0	0	0	4	4
Joe	0	0	0	5	5
Jim	0	0	0	2	2

Singular Value Decomposition an example

	Matrix	Alien	Star Wars	Casablanca	Titanic
John	1	1	1	0	0
Jack	3	3	3	0	0
Jill	4	4	4	0	0
Jenny	5	5	5	0	0
Jane	0	0	0	4	4
Joe	0	0	0	5	5
Jim	0	0	0	2	2

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} .14 & 0 \\ .42 & 0 \\ .56 & 0 \\ .70 & 0 \\ 0 & .60 \\ 0 & .75 \\ 0 & .30 \end{bmatrix} \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \begin{bmatrix} .58 & .58 & .58 & 0 & 0 \\ 0 & 0 & 0 & .71 & .71 \end{bmatrix}$$

M U Σ V^{T}

SVD applied to term-document matrices: Latent Semantic Analysis

- **We discard all latent dimensions apart from the first *k* ones** (e.g. *k*=300)
  - U is replaced by an *m*×*k* matrix U_k,
    Σ is replaced by a *k*×*k* diagonal matrix Σ_k with only the *k* first singular values,
    U is replaced by an *n*×*k* matrix V_k
    and $\mathbf{M \approx U_k \Sigma_k V_k^T}$
  - We get an optimal approximation of M (least-square)
- We get a *k*-dimensional vector for each word (an "**embedding**") by reading U_k's rows

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} .14 & 0 \\ .42 & 0 \\ .56 & 0 \\ .70 & 0 \\ 0 & .60 \\ 0 & .75 \\ 0 & .30 \end{bmatrix} \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \begin{bmatrix} .58 & .58 & .58 & 0 & 0 \\ 0 & 0 & 0 & .71 & .71 \end{bmatrix}$$

$$M \qquad U \qquad \Sigma \qquad V^{\mathrm{T}}$$

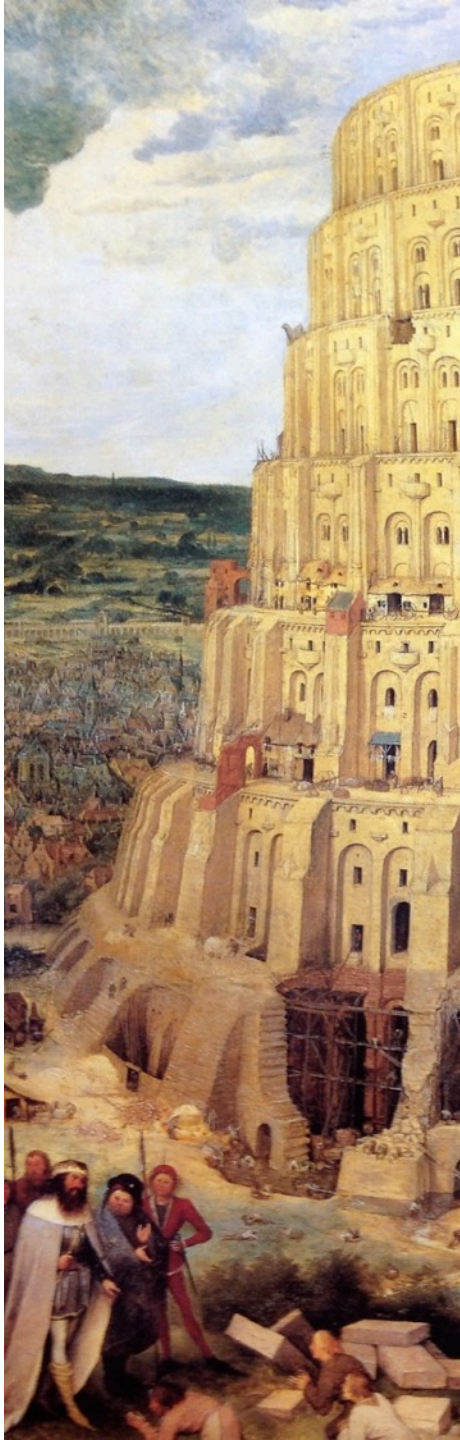
SVD applied to word-word matrices

- **Same technique**
  - Only difference: $m = n$
- Again, we only keep the top k dimensions
  - In fact, it might help to discard the first dimension(s) and keep the k following ones
- Again, we get a k-dimensional vector for each word (an "**embedding**") by reading U_k's rows

Does it work better than sparse vectors?

- **In short, it does**
  - Lower dimensions represent information that is not important, not necessarily significant: **denoising**
  - Removing lower dimensions results in **generalisations**, including capturing **higher-order cooccurrence**
  - Fewer dimensions = models easier to learn (**fewer weights**)

Building dense vectors using a neural network



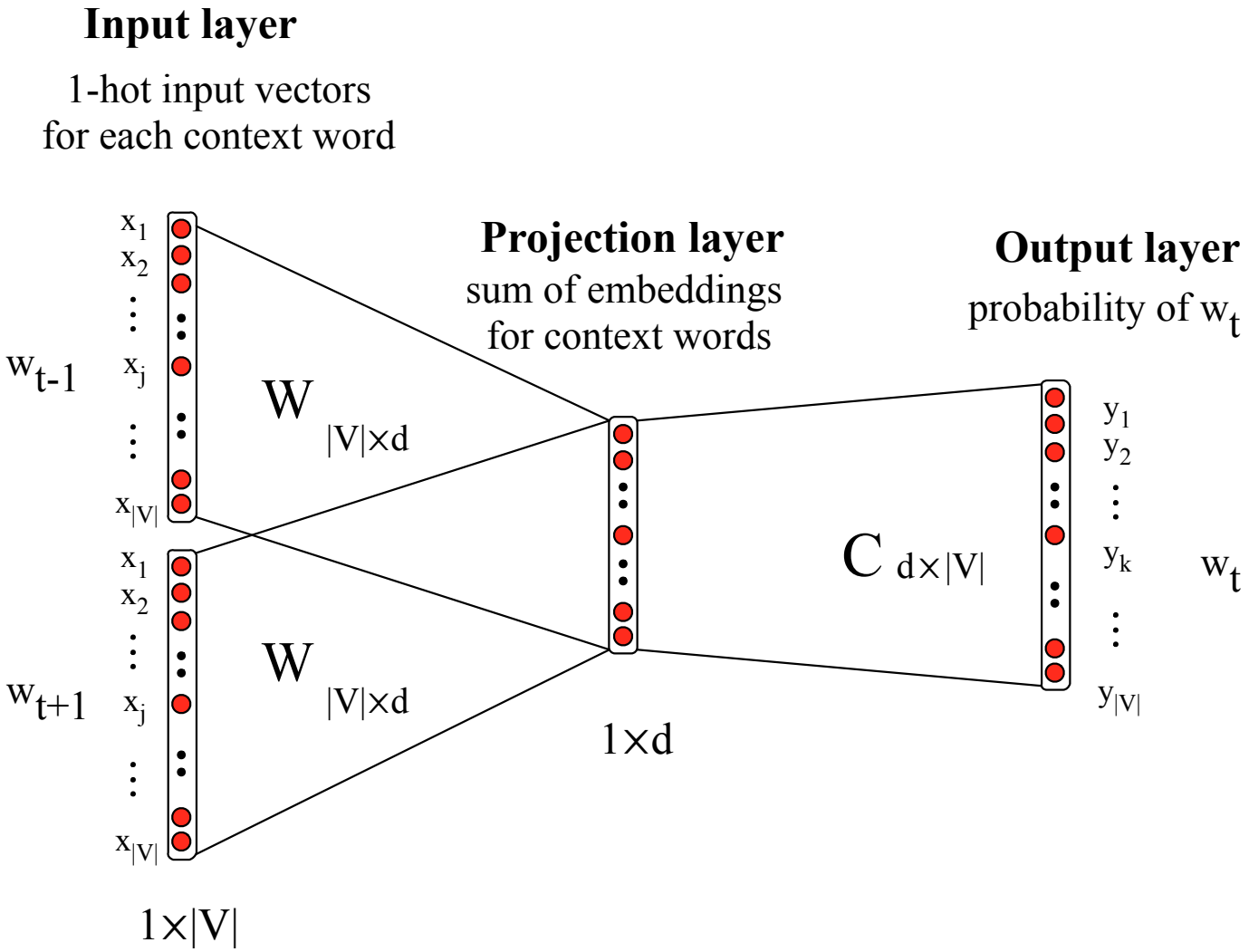
Prediction-based embeddings

- Different approach for implementing the same intuition
  - Underlying principle is still the distributional hypothesis
  - Instead of starting with **word counts to quantify the notion of distribution**, we will **learn to predict words based on distributional properties of their contexts**
  - We will do so **using a neural approach**
- Underlying idea: we will train a neural network to perform a given word-based task and extract **intermediate representations** as word representations (word embeddings)
- Approach popularised by the **word2vec** package (Mikolov et al. 2013a,b)
  - Easy and fast to train
  - Freely available, pre-trained embeddings

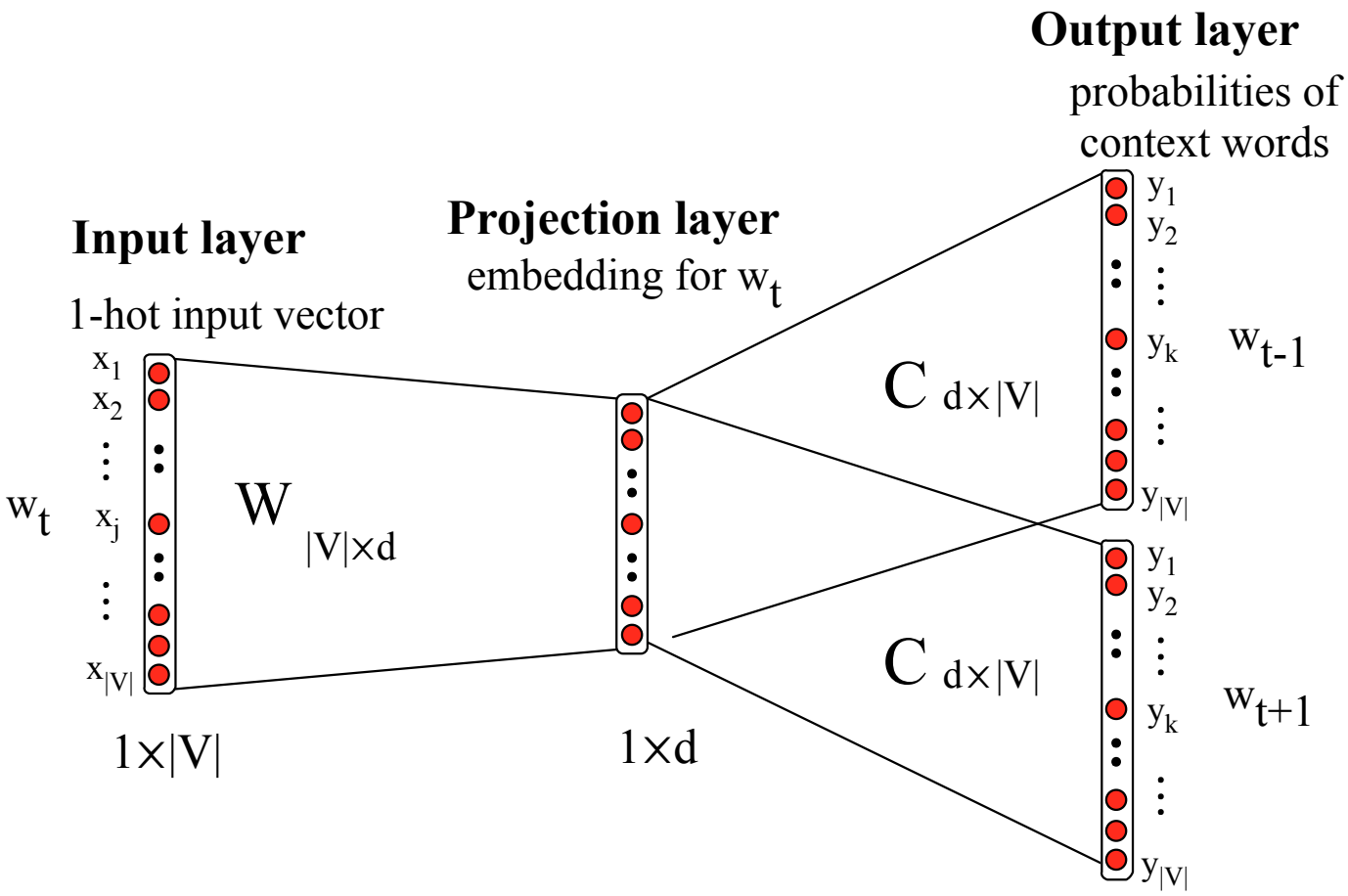
Prediction-based embeddings

- Neural network with one hidden layer
  - This hidden layer will provide the embeddings
- Input and output layers use **1-hot vector representations**
  - Word $w_i \in V$ is represented by a |V|-dimensional vector whose values are all 0 except for the *i*-th one which is 1
  - Contexts involving several words have 0 values except for those dimensions corresponding to these words
  - In the output layer, each value is interpreted as a probability (via the application of the SoftMax operation)
- Two types of predictions in word2vec:
  - Given a context surrounding a position, predict the word that should fill this position: **CBoW** (continuous bag of words)
  - Given a word, predict its neighbours: **skip-gram**

CBoW


Input layer
1-hot input vectors
for each context word
Projection layer
sum of embeddings
for context words
Output layer
probability of w_t
w_{t-1}
w_{t+1}
x_1 x_2 x_j $x_{|V|}$
$W_{|V|\times d}$
$C_{d\times|V|}$
y_1 y_2 y_k $y_{|V|}$
w_t
$1\times d$
$1\times|V|$

Skip-grams

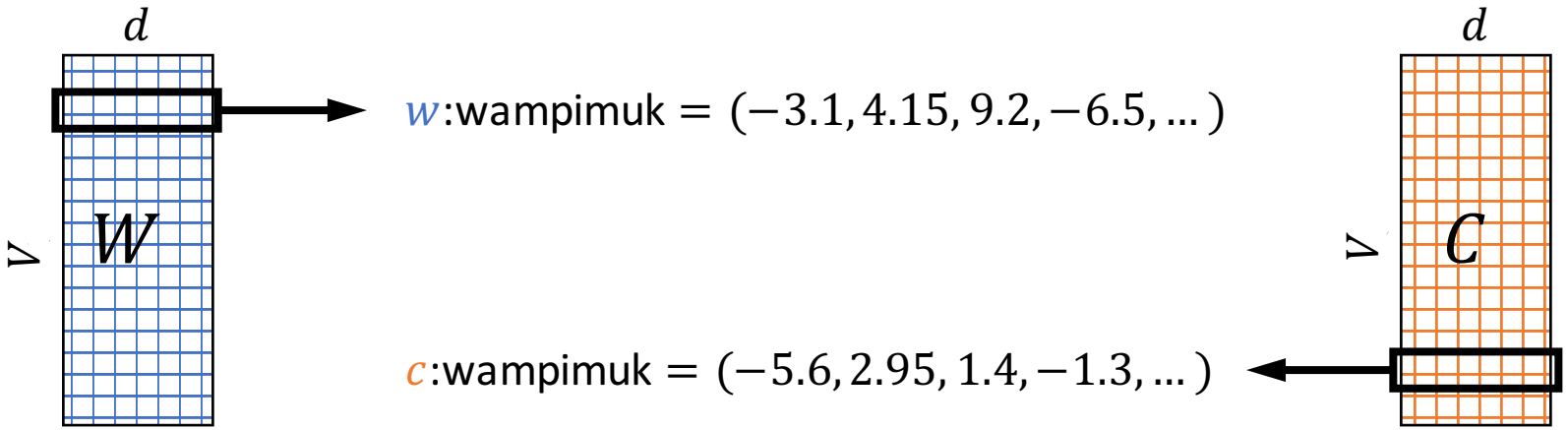

Output layer
probabilities of
context words
Input layer
1-hot input vector
Projection layer
embedding for w_t
w_t
x_1
x_2
x_j
$x_{|V|}$
$W_{|V|\times d}$
$C_{d\times|V|}$
$C_{d\times|V|}$
y_1
y_2
y_k
$y_{|V|}$
w_{t-1}
w_{t+1}
$1\times|V|$
$1\times d$

Focus on skip-grams

- The skip-gram approach creates a vector for each word $w \in V$
  - Each such vector has *d* latent dimensions (e.g. *d*=100)
- **It learns a matrix W** whose rows represent each word $w \in V$
- It also learns **a similar auxiliary matrix C** of context vectors
  - In fact, each word has two embeddings

Example (Goldberg apud Baroni):

Marco saw a furry little wampimuk hiding in the tree.



Negative sampling

Marco saw a furry little wampimuk hiding in the tree.

- Positive examples
  (wampimuk, furry), (wampimuk, little), (wampimuk, hiding), etc.

- No negative examples can be directly extracted from the data
  - **Negative sampling** is used to artificially create negative examples
  - How? By replacing the context word in such pairs with randomly selected words
    - "Randomly selected" in the sense of the unigram distribution
    - For each positive example, *k* negative examples are built

Negative sampling

Goldberg *et al.* 2014

- **Maximize:** $\sigma(\vec{w} \cdot \vec{c})$
  - c was **observed** with w

words	**contexts**
wampimuk	furry
wampimuk	little
wampimuk	hiding
wampimuk	in

- **Minimize:** $\sigma(\vec{w} \cdot \vec{c}')$
  - c' was **hallucinated** with w

words	**contexts**
wampimuk	Australia
wampimuk	cyber
wampimuk	the
wampimuk	1985

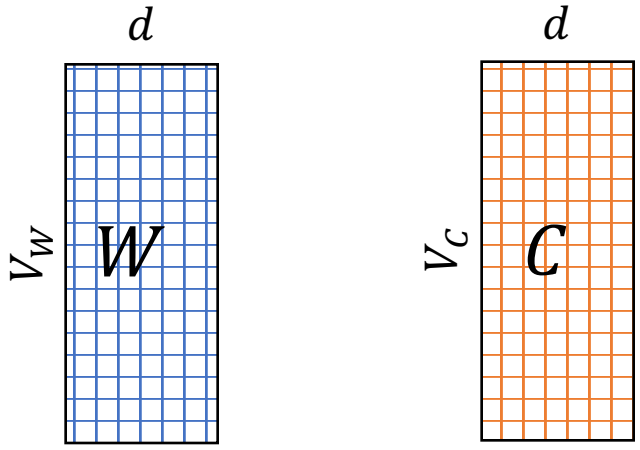
Skip-Gram with Negative Sampling (SGNS)

- Word embeddings created by the skip-gram model using negative sampling is the classical state-of-the-art

- word2vec's SGNS outperforms count-based (PPMI) vectors when used in virtually all NLP tasks
  - or does it?
  - Goldberg and colleagues have investigated what makes word2vec's SGNS so much better…

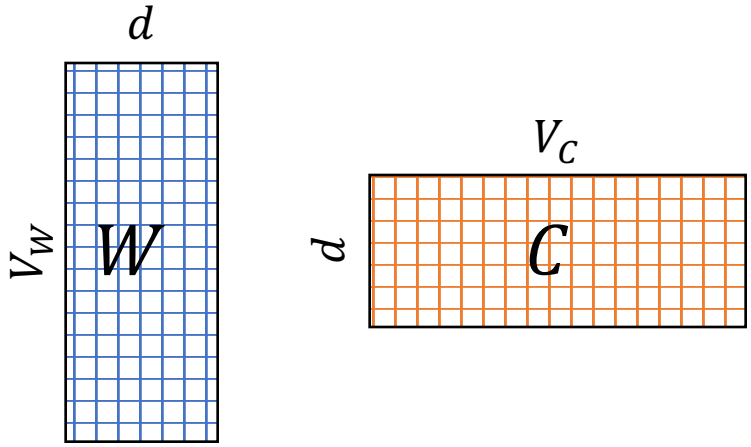
What is SGNS learning?

- Take both SGNS embedding matrices,



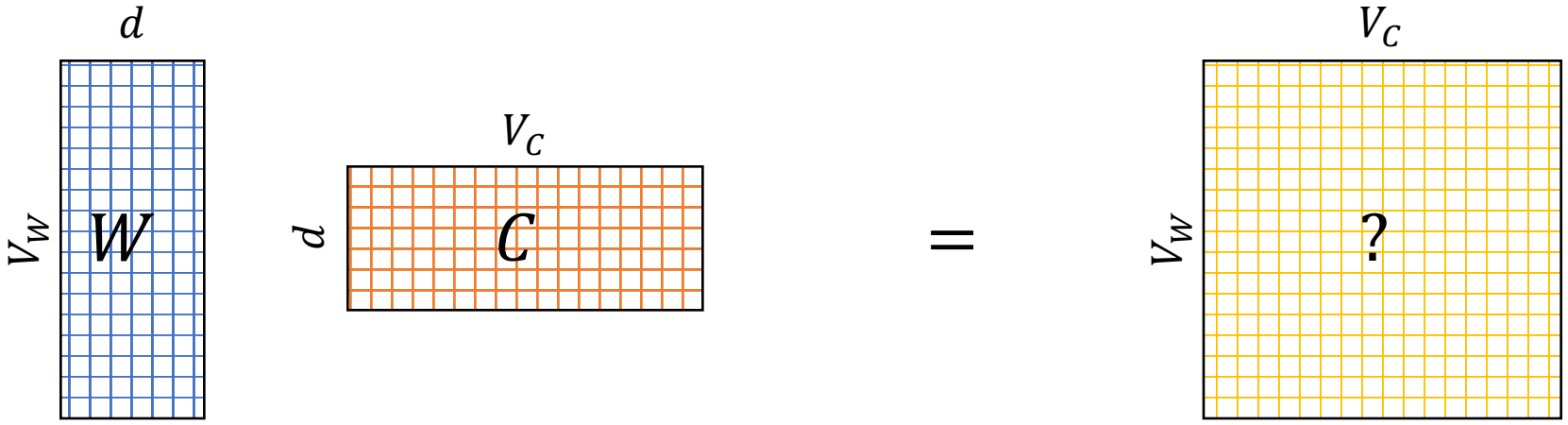
What is SGNS learning?

- Take both SGNS embedding matrices, multiply them



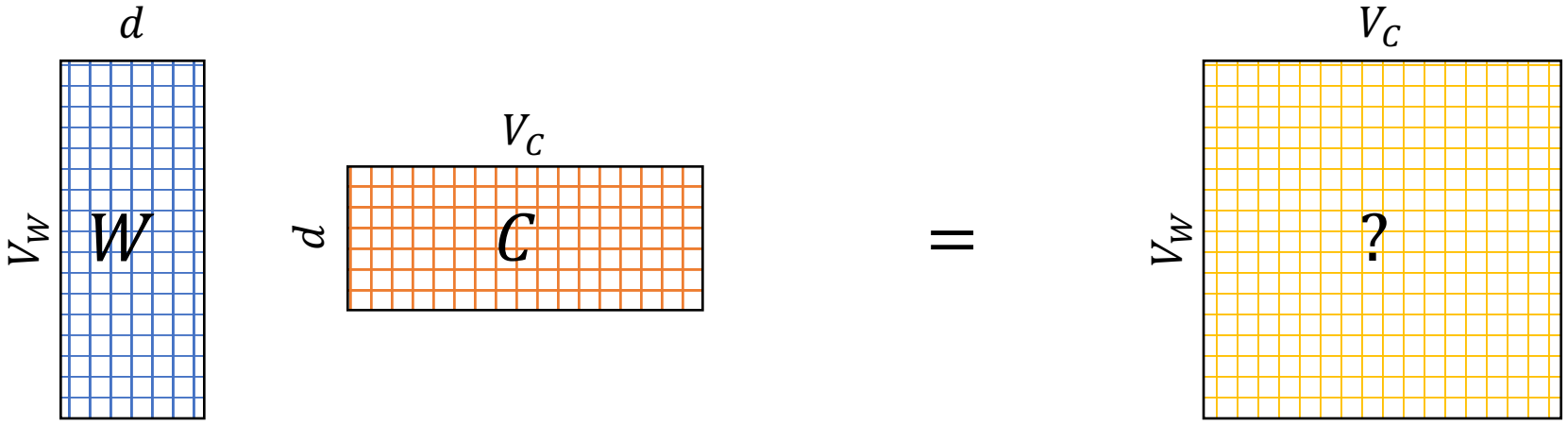
What is SGNS learning?

- Take both SGNS embedding matrices, multiply them,
- You get a $|V|^2$ matrix in which each cell describes the relation between a target word and a context word



What is SGNS learning?

- Take both SGNS embedding matrices, multiply them,
- You get a |V|² matrix in which each cell describes the relation between a target word and a context word
- It turns out that when *d* is large and after enough training iterations, **you get the PMI matrix**…



What is SGNS learning?

- Take both SGNS embedding matrices, multiply them,
- You get a $|V|^2$ matrix in which each cell describes the relation between a target word and a context word
- It turns out that when *d* is large and after enough training iterations, **you get the PMI matrix**… except for a constant:

$$W \cdot C^T \longrightarrow M_{PMI} - \log k$$

(Levy & Goldberg 2014)

What is SGNS learning?

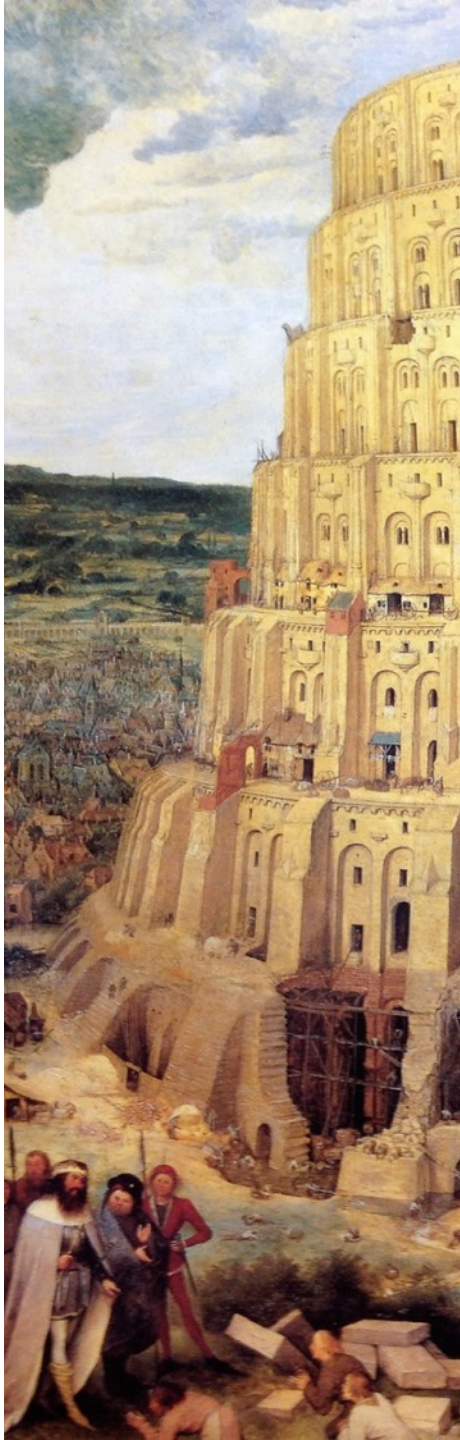
- Take both SGNS embedding matrices, multiply them,
- You get a $|V|^2$ matrix in which each cell describes the relation between a target word and a context word
- It turns out that when *d* is large and after enough training iterations, **you get the PMI matrix**… except for a constant:

$$W \cdot C^T \longrightarrow M_{PMI} - \log k$$

(Levy & Goldberg 2014)

- So **SGNS is factorising the word-word PMI matrix**
  - **SVD does this too!**
  - Mathematically, nothing really new then…

On the importance of little details



SGNS or SVD?

- Plenty of evidence that (neural) embeddings outperform traditional (e.g., SVD) methods
  - “Don’t Count, Predict!” (Baroni et al., ACL 2014)
  - Cf. also fastText embeddings (using subword information; Bojanowski et al. 2016)
- This is not systematic:
  - GloVe (Pennington et al., EMNLP 2014) is a recent, non-neural word embedding approach
- How is it possible, since we just shown that SGNS performs a similar computation to the traditional, (P)PMI-based approach?
- The answer is: *the devil is in the detail*

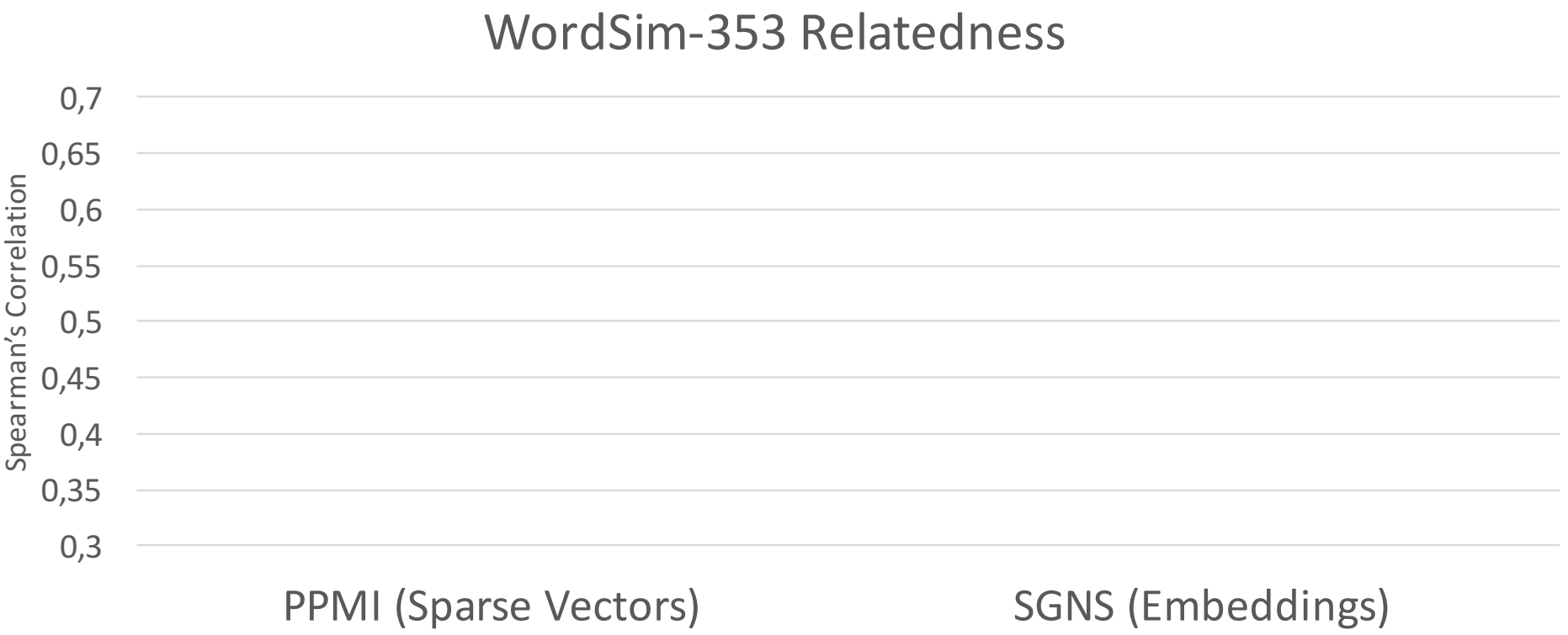
Hyperparameters

- word2vec does not only introduce **2 new algorithms**
- **It also includes and tunes several hyperparameters**
- For word2vec's SGNS, here are a few of these hyperparameters:
  - **Preprocessing**
    - Dynamic Context Windows
    - Subsampling
    - Deleting Rare Words
  - **Association Metric**
    - Shifted PMI
    - Context Distribution Smoothing

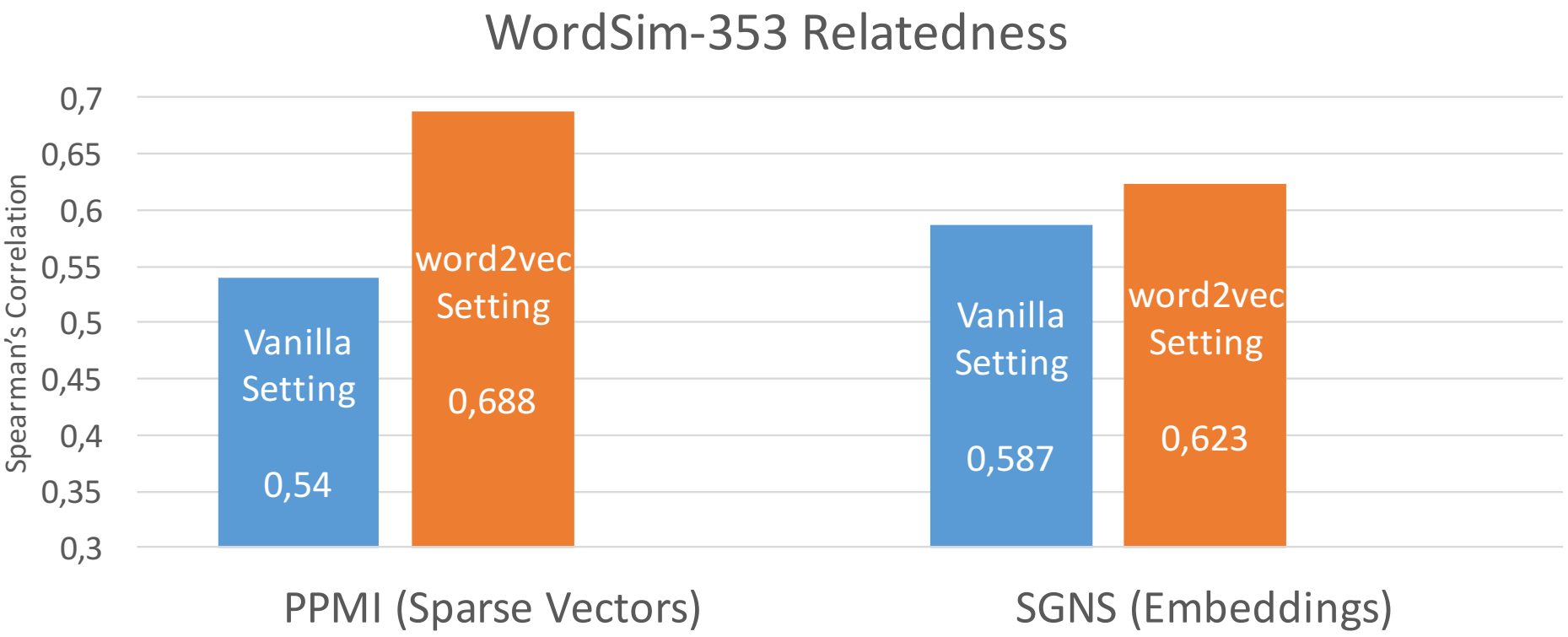
Hyperparameters

- word2vec does not only introduce **2 new algorithms**
- **It also includes and tunes several hyperparameters**
- For word2vec's SGNS, here are a few of these hyperparameters:
  - **Preprocessing**
    - Dynamic Context Windows
    - Subsampling
    - Deleting Rare Words
  - **Association Metric**
    - Shifted PMI
    - Context Distribution Smoothing

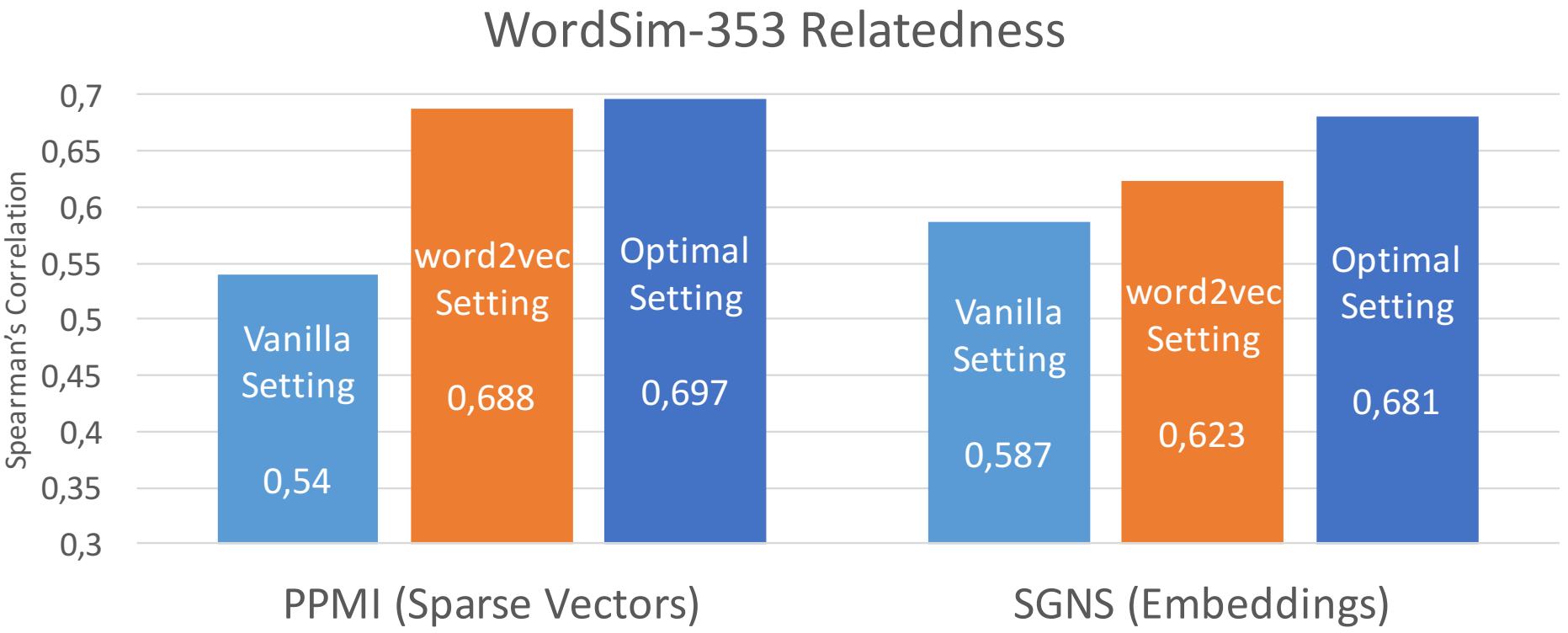
Results


WordSim-353 Relatedness
Spearman's Correlation
0,7
0,65
0,6
0,55
0,5
0,45
0,4
0,35
0,3
PPMI (Sparse Vectors)
SGNS (Embeddings)

Results


WordSim-353 Relatedness
Spearman's Correlation
0,7
0,65
0,6
0,55
0,5
0,45
0,4
0,35
0,3
Vanilla Setting
0,54
word2vec Setting
0,688
Vanilla Setting
0,587
word2vec Setting
0,623
PPMI (Sparse Vectors)
SGNS (Embeddings)

Results


WordSim-353 Relatedness
Spearman's Correlation
0,7
0,65
0,6
0,55
0,5
0,45
0,4
0,35
0,3
Vanilla Setting
0,54
word2vec Setting
0,688
Optimal Setting
0,697
Vanilla Setting
0,587
word2vec Setting
0,623
Optimal Setting
0,681
PPMI (Sparse Vectors)
SGNS (Embeddings)

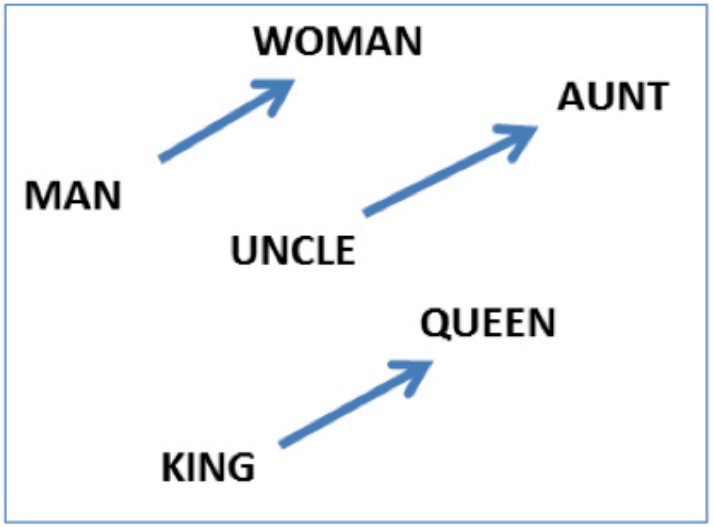
Results

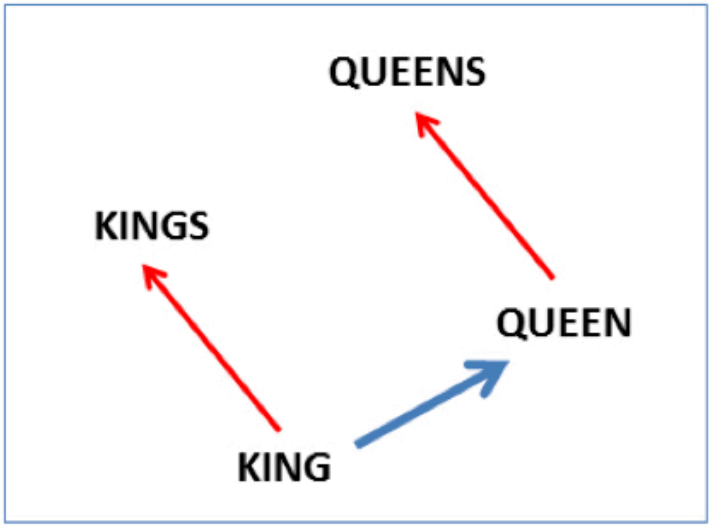
- Impact of hyperparameters is higher than that of algorithms
- Better hyperparameters often have stronger effects than more data
- Neural-based approaches do not always outperform traditional approaches
  - contra Baroni et al; 2014 (“Don’t count, predict!”): predicting is not better, but word2vec hyperparameters make it better
  - SGNS does, sometimes. It seems that it performs better than traditional methods on syntactic analogies (not on semantic analogies)

Analogy: the structure of word similarity

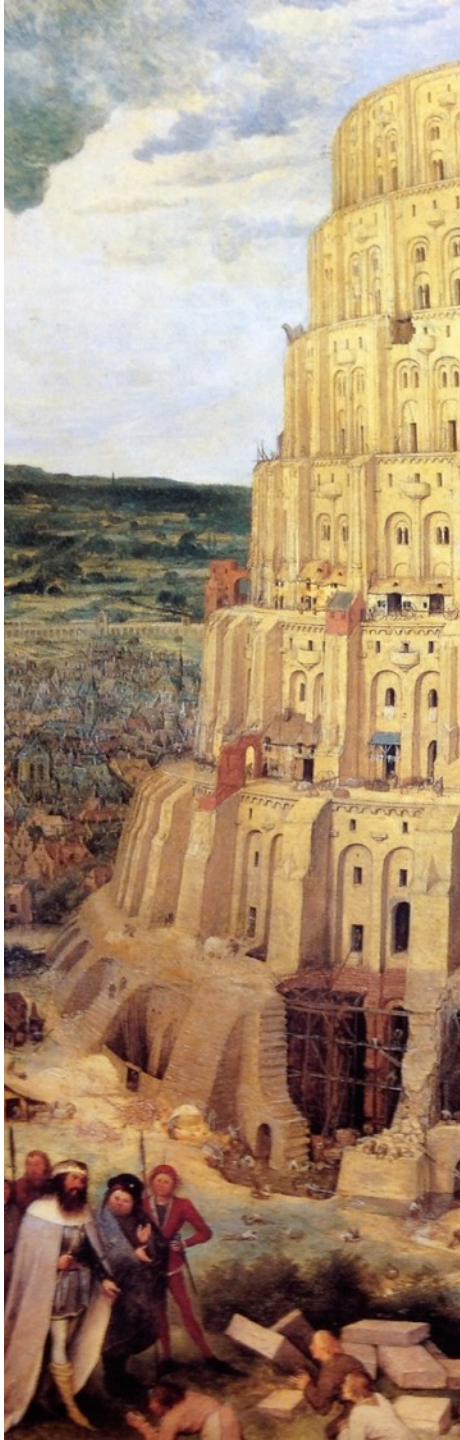
vector(*'king'*) – vector(*'man'*) + vector(*'woman'*) ≈ vector('queen')

vector(*'Paris'*) – vector(*'France'*) + vector(*'Italy'*) ≈ vector('Rome')





Contextual word embeddings



Context in word2vec

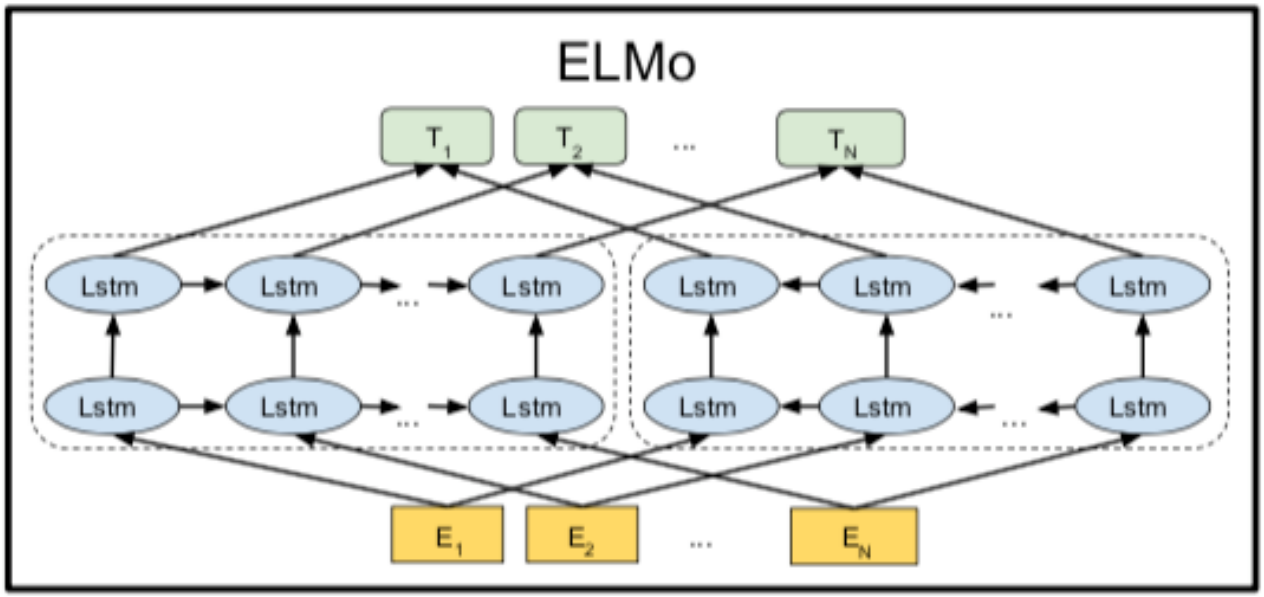
- Context in word2vec is a (weighted) bag of words
  - Context words are taken independently of one another
  - …and independently of their position with respect to the target word
- Such information is crucial for correctly modelling the context
  - *I am not happy to see you*
    vs. *I am happy to see you*
  - *The cat eats the mouse*
    vs. *The mouse eats the cat*

Refining context representation

- What we want to do
  - Build a representation that takes the **whole context** into account
  - We need to **selectively accumulate information** from the (left and right) context
  - Obvious answer: use a **sequence modelling NN**
    - Typically, use an **LSTM** (Long Short-Term Memory) layer
      - **ELMo** (Peters 2018)
    - … or **Transformers**
      - **BERT** (Devlin et al. 2018)

ELMo (Peters et al. 2018)

- Train a BiLSTM on a large dataset for bidirectional language modelling, in this case to predict the next word
- Encode the sentence by running it through the sentence through both forward and backward LSTMs
- Combine forward and backward representations into final contextual embeddings



ELMo (Peters et al. 2018)

- Train a BiLSTM on a large dataset for bidirectional language modelling, in this case to predict the next word
- Encode the sentence by running it through the sentence through both forward and backward LSTMs
- Combine forward and backward representations into final contextual embeddings

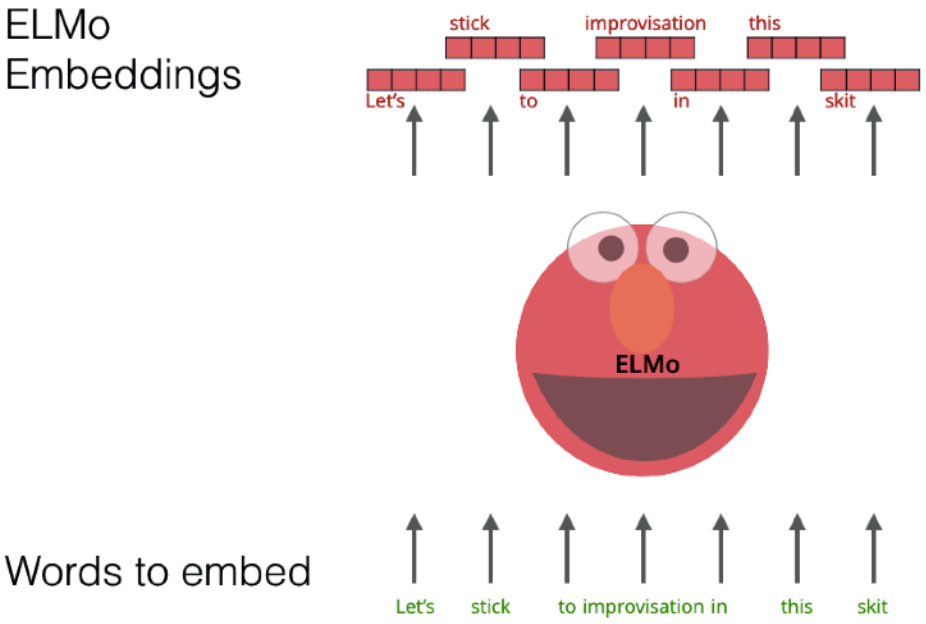


Figure (adapted) from http://jalammar.github.io/illustrated-

ELMo (Peters et al. 2018)

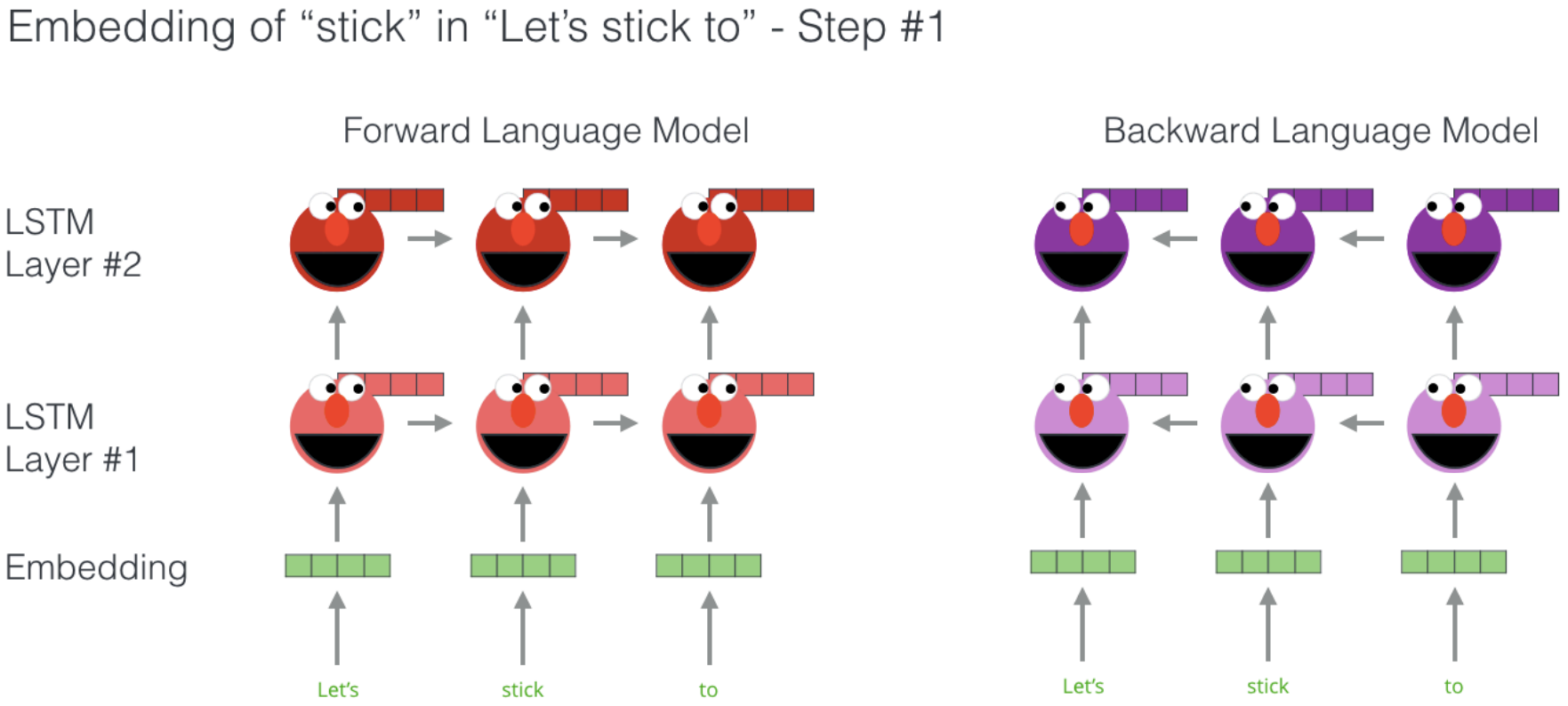


Figure (adapted) from http://jalammar.github.io/illustrated-

ELMo (Peters et al. 2018)

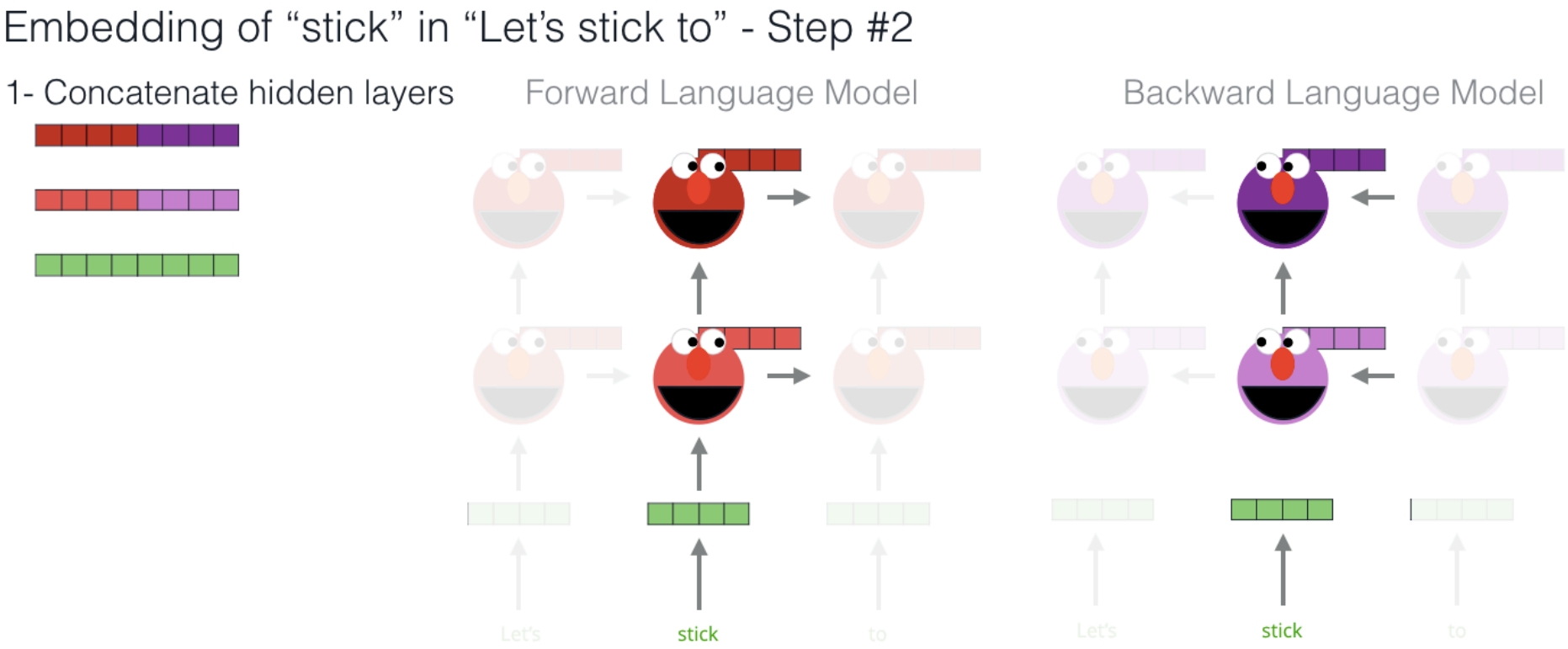


Figure (adapted) from
http://jalammar.github.io/illustrated-

ELMo (Peters et al. 2018)

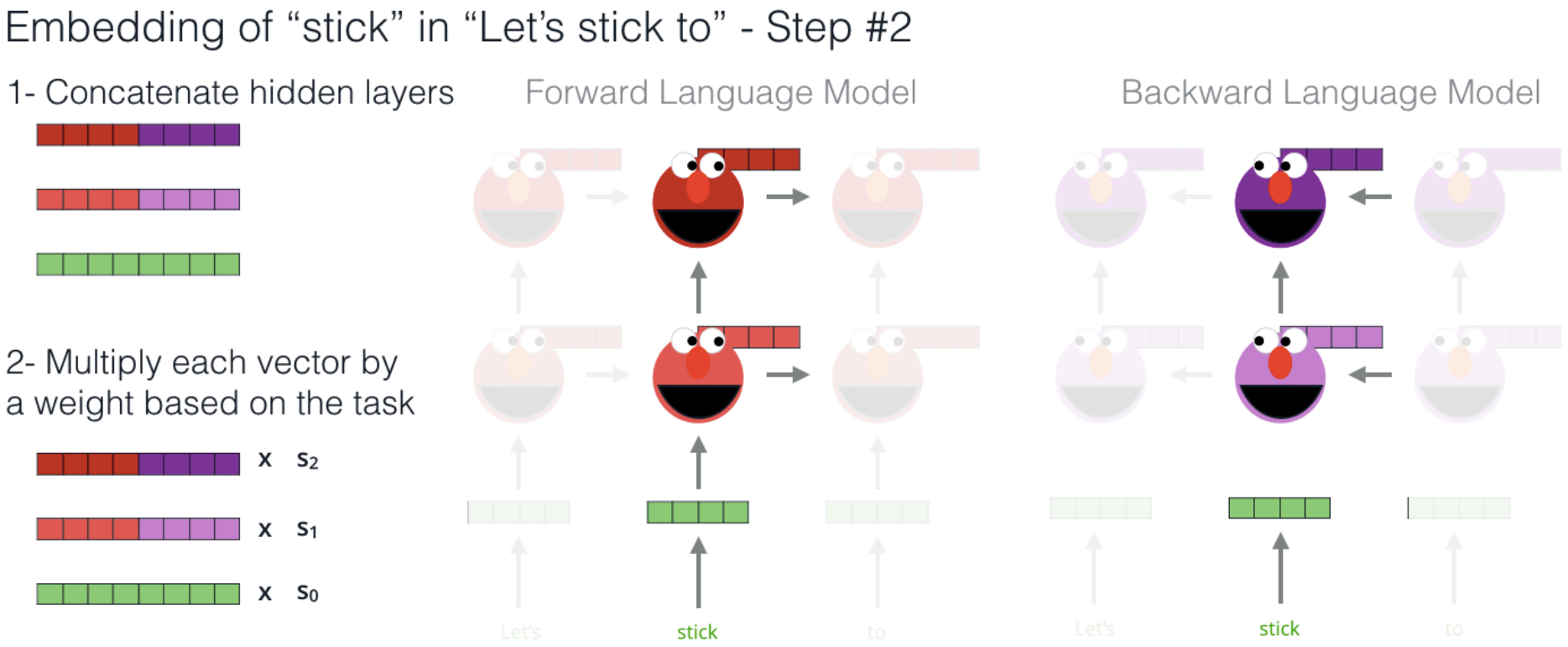


Figure (adapted) from http://jalammar.github.io/illustrated-

ELMo (Peters et al. 2018)

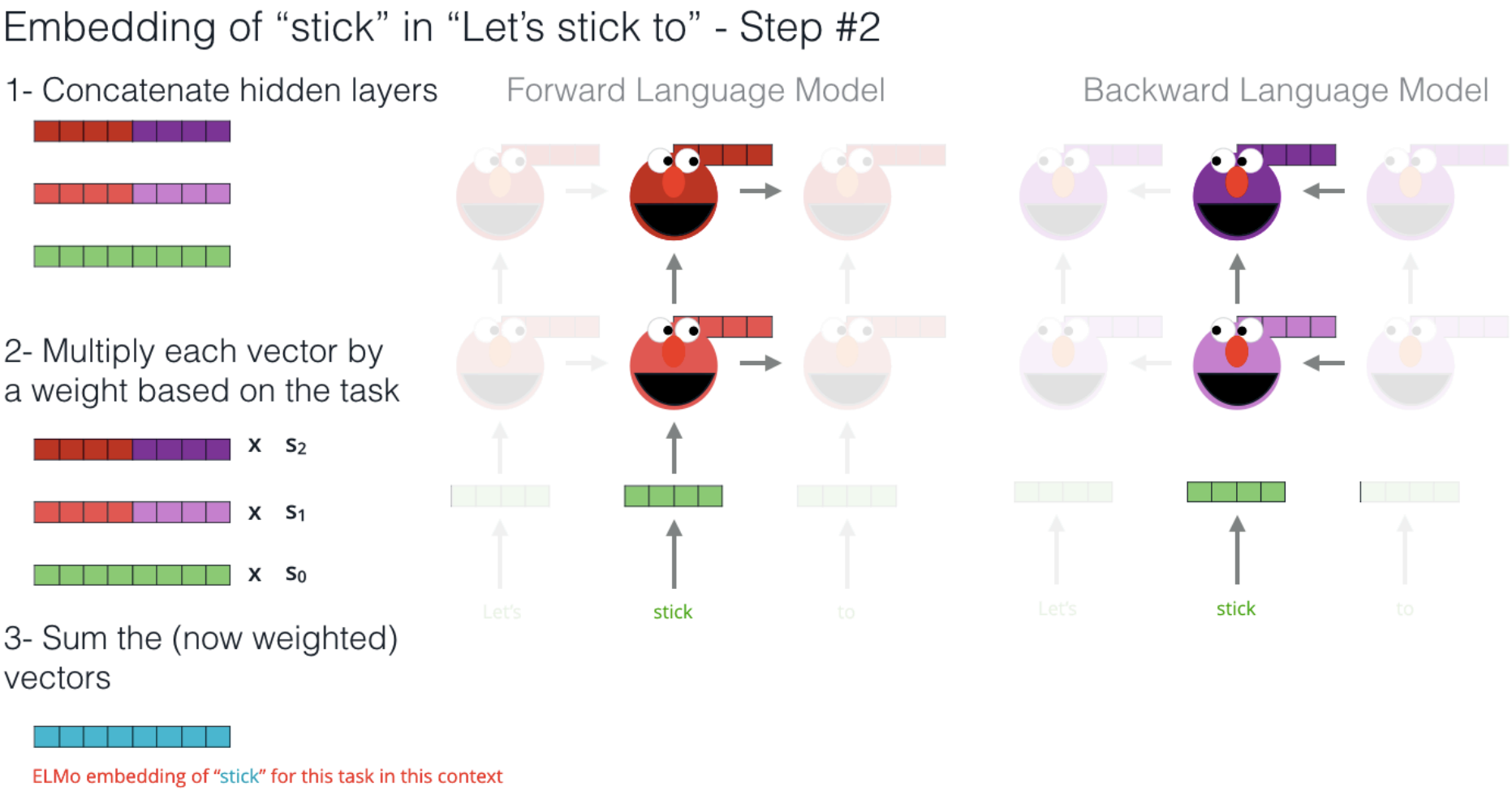


Figure (adapted) from http://jalammar.github.io/illustrated-

ELMo (Peters et al. 2018)

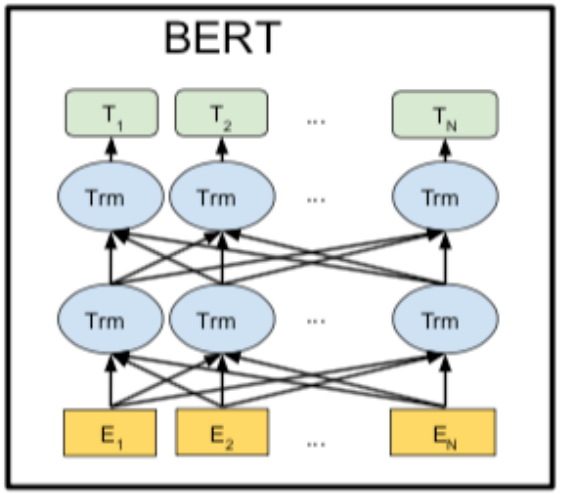
- So how contextual are ELMo embeddings?

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

- Additional benefit : standard ELMo models use as input vectors the result of a CNN layer to capture the character-level content of words
  - So they can even handle unknown words!
- A task-specific NN can then be trained to perform a task when provided as an input with the ELMo contextual embeddings of words

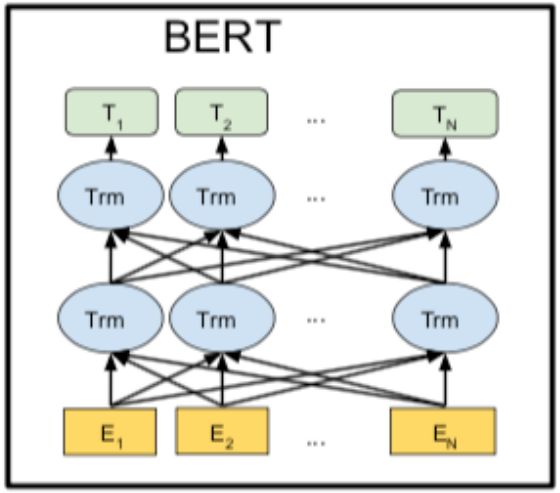
BERT (Devlin et al. 2019)

- Same underlying idea : contextual embeddings via word prediction
- Use a Transformer instead of a BiLSTM



BERT (Devlin et al. 2019)

- Same underlying idea : contextual embeddings via word prediction
- Use a Transformer instead of a BiLSTM

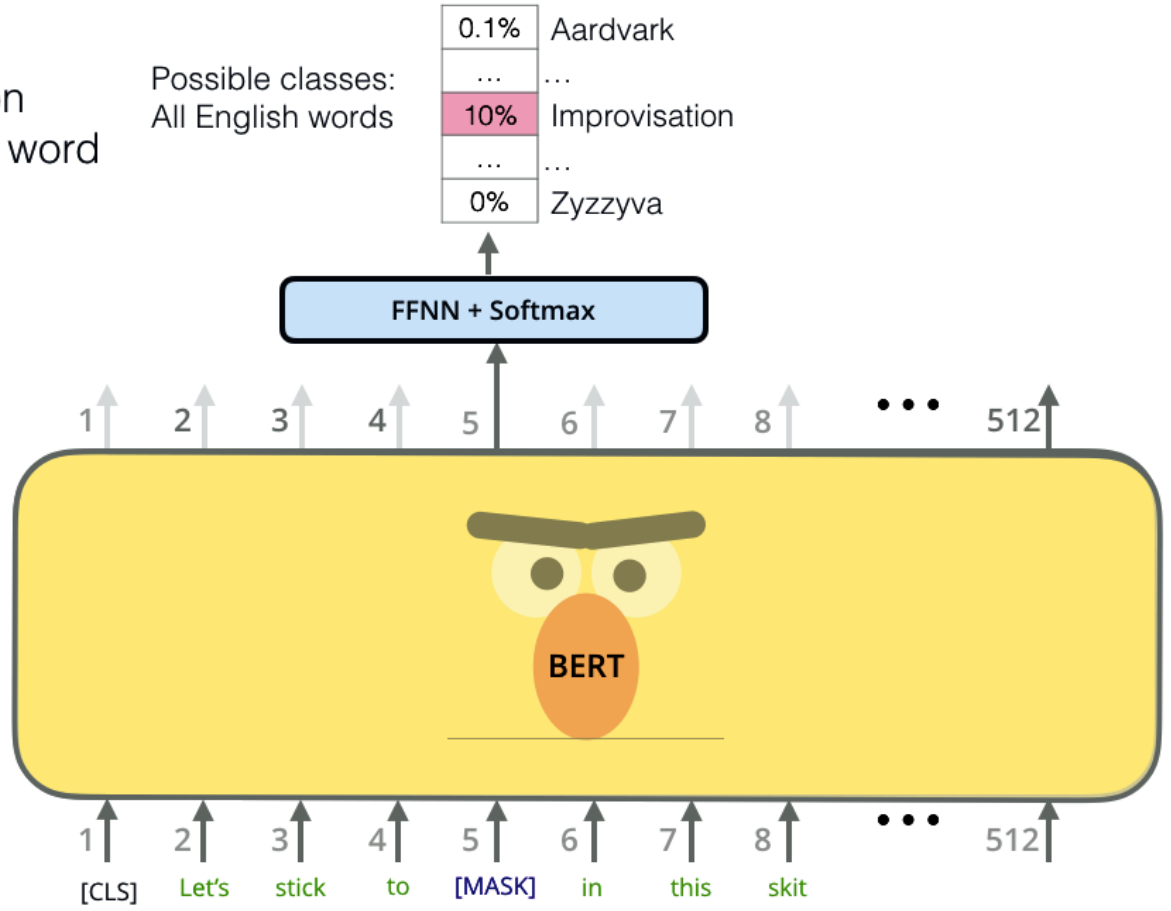


Each level sees the whole level below, contrarily to RNNs such as LSTMs

- Different Language Modelling objective: the Masked Language Model
  - Given a sentence with some words masked at random, can we predict them? (cf. Taylor 1954, “Cloze task”)
  - Randomly select 15% of tokens to be replaced with “<MASK>”

BERT (Devlin et al. 2019)

Use the output of the masked word's position to predict the masked word



Randomly mask 15% of tokens

Input

[CLS] Let's stick to improvisation in this skit

Figure (adapted) from http://jalammar.github.io/illustrated-

BERT (Devlin et al. 2019)

- In fact, BERT combines this objective with another one:
  - Given two sentences, does the first follow the second?
  - Teaches BERT about the relationship between two sentences
  - 50% of the time the actual next sentence, 50% random

BERT (Devlin et al. 2019)

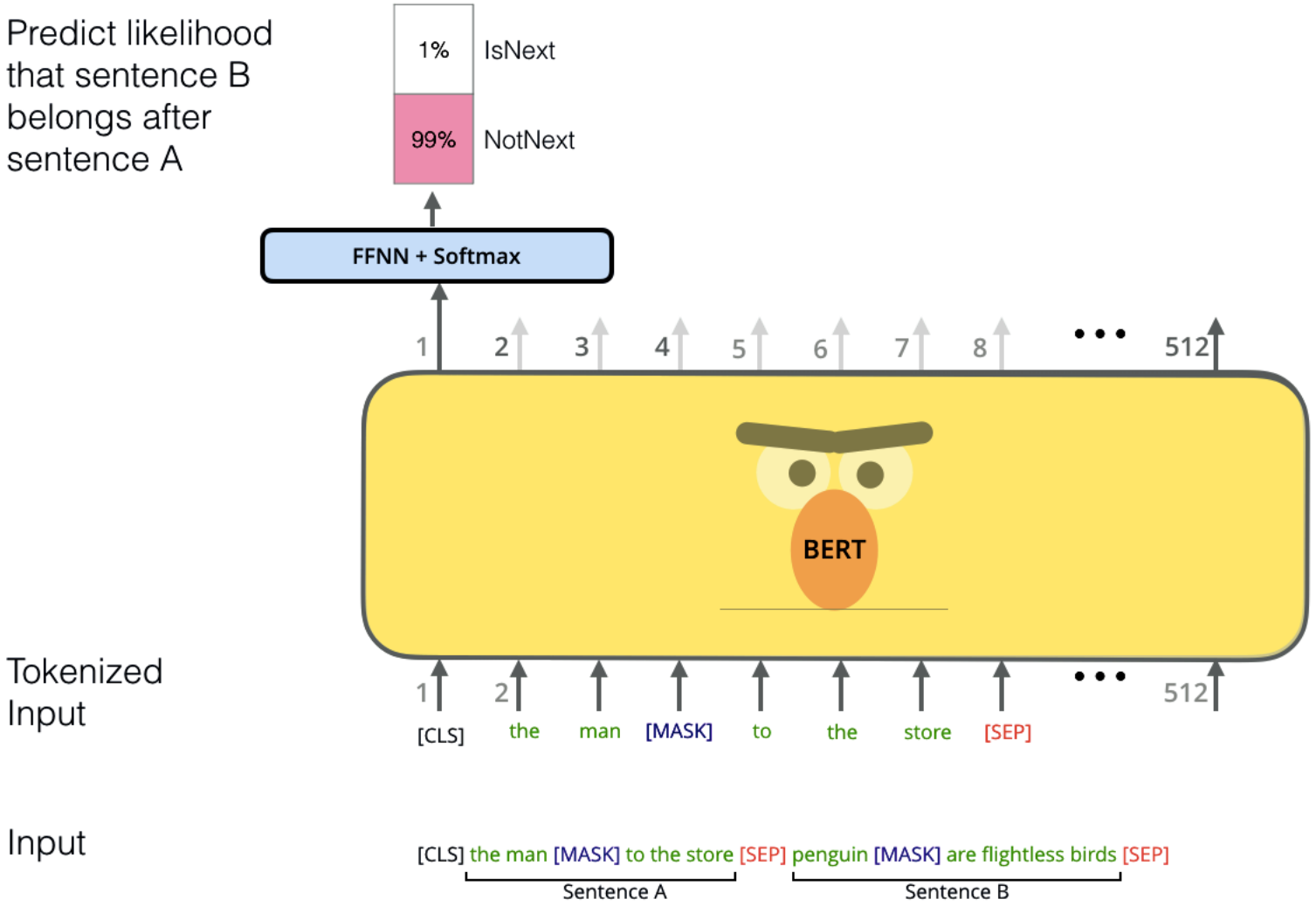


Figure (adapted) from http://jalammar.github.io/illustrated-

BERT (Devlin et al. 2019)

- A task-specific NN can then be trained to perform a task when BERT contextual embeddings of words are provided as input
  - "BERT as **embeddings**"
- BERT is now often used in **fine-tuning** settings
  - Additional NN layers are added on top of the BERT architecture
  - BERT layers are initialised with the output of the BERT LM training step, which is then named the **pre-training** step
  - The whole architecture is then trained on a task-specific dataset
- ELMo outperformed previous architectures
  - It seems it can be trained on less data than BERT
- But training BERT can benefit from the fact that training Transformer architectures can be better parallelised, and therefore be trained on more data, if available
  - The resulting models outperforms ELMo in most if not all tasks

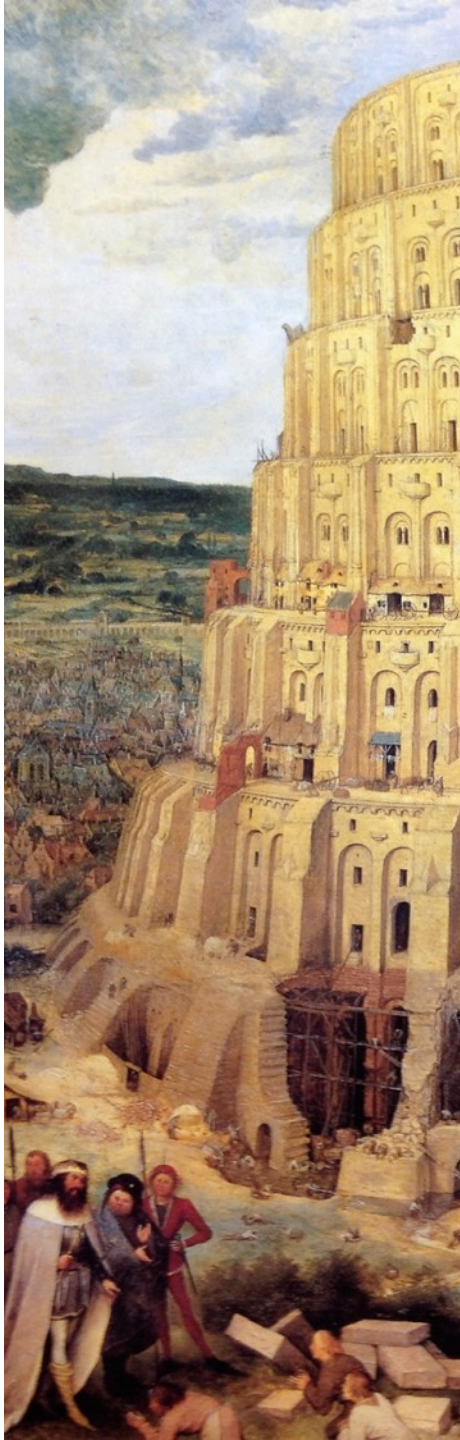
BERT variants and extensions

- **Multilingual BERT** (same paper as BERT) : BERT trained on texts covering ~100 languages (mostly Wikipedia editions)
- **ROBERTa** (Liu et al. 2019) : more training, better hyperparameters, only the masked word model objective, and other technical changes
- **ALBERT** (Lan et al. 2019) : technique for having fewer parameters than in vanilla BERT
- **CamemBERT** (Martin et al. 2019, our work ;-) a ROBERTa-like model trained on French dat with a few (technical) differences
  - Works significantly better than the multilingual BERT

Training BERT

- **How much data** is required to train a BERT (or ROBERTa) ?
- Short answer : we don't really know yet
- The original BERT model was trained on 16GB of uncompressed text (~4B tokens, Wikipedia + Book Corpus)
- The authors of the ROBERTa paper have shown that more data helps
  - They use 160GB of uncompressed text (~40B tokens, more diverse)
- For CamemBERT, we experimented with the French Wikipedia (1B tokens) and our web-based, Common-Crawl-derived corpus, OSCAR (32B tokens)
  - *Paper currently under review: do not tell anyone ;-)*
  - Using 1B tokens randomly selected from OSCAR works as well as the whole corpus,
  - …but better than the French Wikipedia : heterogeneity helps!
  - At least 38 languages have an OSCAR corpus with 1B tokens or more

Identifying words and sentences



What is a sentence?

- Output of a **macroscopic segmentation**
  - **Self-contained syntactic structure**
  - Semantically related to other sentences via specific phenomena (the "**discursive**" level)
    - anaphora
    - discourse relations
  - (dubious) Prosodically marked (pauses, intonation)
  - Typographically marked
    - In the Latin script, the full stop (or period) is marked with an ambiguous, sometimes overloaded symbol; the same holds for the uppercase first letter of a "sentence"

What is a sentence?

- This is an **idealised view**
  - Self-contained syntactic structures are not that frequent in speech
  - …and do not always match the “intuitive” notion of sentence
    “Malheur à toi si tu refuses”, le menaça-t-il.
  - Typography is sometimes misleading
    Best. Movie. Ever.
    Dès maintenant, la mobilisation est de mise. Pour l’amour des mots.
    The grocery sells cucumbers, lettuce, radishes, etc.
  - Nested structures, e.g. with quotes:
    "It is basically the perfect sort of tool to find objects like 'Oumuamua. We expect to find 100s of them with the LSST," Dr Fraser says.

What is a word?

- No linguist would dare define a unique notion “word”
- At least four notions must be distinguished:
  - The prosodic word
  - The typographic word, or **token**
  - The morphosyntactic word, or **wordform** (or **form**)
  - The **semantic word**
- Let us review the last three concepts
  - And mismatches between these three notions

Tokens

- **A token is a purely typographic unit**
  - **Conventionally**, deterministically defined
- Starting point:
  - Many writing systems have “punctuation marks”
  - Some of them have a typographic separator (e.g. whitespace)
- In writing systems with a typographic separator, a token can be defined as:
  - A sequence of characters containing no separators and no punctuation marks,
  - or a punctuation mark (anything that is not a letter or a digit)
  - Ex.: *All of a sudden, he started playing table tennis with John Doe.*

Tokens

- **A token is a purely typographic unit**
  - **Conventionally**, deterministically defined
- Starting point:
  - Many writing systems have “punctuation marks”
  - Some of them have a typographic separator (e.g. whitespace)
- In writing systems with a typographic separator, a token can be defined as:
  - A sequence of characters containing no separators and no punctuation marks,
  - or a punctuation mark (anything that is not a letter or a digit)
  - Ex.: *All of a sudden , he started playing table tennis with John Doe .*

Tokens

- In writing systems without a typographic separator, the simplest way to define a token is to consider each individual character as a token
  - Remember splitting a sentence into tokens must be **deterministic** by definition
  - Ex.: 我的漢語說得不太好。

ฉันฟังไม่เข้าใจ

Tokens

- In writing systems without a typographic separator, the simplest way to define a token is to consider each individual character as a token
  - Remember splitting a sentence into tokens must be **deterministic** by definition
  - Ex.: 我 的 漢 語 說 得 不 太 好 。

    ฉ ั น ฟ ั ง ไ ม่ เ ข ้ า ใ จ

Wordforms (or forms)

- A **wordform** is a syntactically atomic unit
  - It can receive annotations such as a part-of-speech (noun, adjective, etc…), morphological features (plural, dative…)
  - It corresponds to the leaves in syntactic structures
  - Not easy to identify! Ambiguities…
  - Sometimes, the token modifies the form (uppercase, errors…)
- **Mismatches** between tokens and forms are numerous:
  - 1 token corresponding to multiple forms = **amalgam**
    Ex.: aux (= à les), du (= de les OR du) won't (= will not)
    Sp. dámelo (= da me lo)
  - multiple tokens corresponding to 1 form = **compound word**
    Ex.: au fur et à mesure all of a sudden
  - multiple tokens corresponding to multiple forms
    Ex.: au fur et à mesure du (= au_fur_et_à_mesure_de le)

Named entities

- A **named entity** is a real-world object that can be denoted individually
  - Standard named entities: people, locations, organisations
  - Extended named entities: dates, addresses, URLs, e-mail addresses, numbers…
- A named entity **mention** is an utterance denoting a named entity
  - Examples: Emmanuel Macron, Los Angeles, Apple Inc.
  - This denotation can be ambiguous…
- Named entity mentions have specific properties, apart from their specific, individual denotation:
  - Specific internal structure (**local grammar**), often culture-dependent more than language-dependent
    > **They can be viewed as atomic for the grammar of the language**, and therefore as **special forms**
- Ex.: *All of a sudden , he started playing table tennis with John Doe .*

Named entities

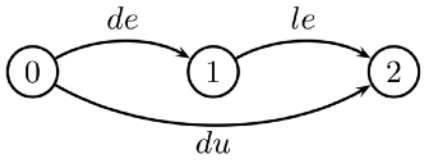
- A **named entity** is a real-world object that can be denoted individually
  - Standard named entities: people, locations, organisations
  - Extended named entities: dates, addresses, URLs, e-mail addresses, numbers…
- A named entity **mention** is an utterance denoting a named entity
  - Examples: Emmanuel Macron, Los Angeles, Apple Inc.
  - This denotation can be ambiguous…
- Named entity mentions have specific properties, apart from their specific, individual denotation:
  - Specific internal structure (**local grammar**), often culture-dependent more than language-dependent
    > **They can be viewed as atomic for the grammar of the language**, and therefore as **special forms**
- Ex.: *all_of_a_sudden , he started playing table tennis with John_Doe .*

Semantic words

- A **semantic word** is a (sequence of) form(s) with non-compositional meaning
  - I.e. its meaning cannot be inferred from the meaning of its constitutive forms
    Ex.: *pomme de terre*, *red herring*
  - Often ambiguous!
    Ex.: *Il a sorti la pomme de terre / Il a modelé une pomme de terre cuite*
    W. C. wrote how he used red herring to lay a false trail, while training hunting dogs
  - Close but distinct form the notion of **term** (a conventional unit)
    Ex.: *machine à laver* (not **appareil à nettoyer* !)

Which words do we need?

- The aim of word embeddings was to represent “words” in a way that captures similarities between them
- Small context window => syntactic-ish similarity
  - Wordforms?
- Larger context window => semantic-ish similarity
  - Semantic words?
- In fact…
  - **Many people use tokens**, because they are easy to get
  - Identifying forms is non-trivial because **non-deterministic**
    We can represent this ambiguity using automata



  - Identifying semantic words is even more difficult; in fact, it is an **ill-defined task**

Noisy tokens?

- The situation is in fact even worse with noisy inputs and language variation
- Example: web texts
- It would be helpful to correct/normalise this before (or jointly with) further processing (including wordform identification)

Phenomenon	Attested example	Std. counterpart	Gloss
	Ergographic phenomena		
Diacritic removal	*demain c'est l'***ete**	*demain c'est l'été*	'tomorrow is summer'
Phonetization	*je suis* **oqp**	*je suis occupé*	'I'm busy'
Simplification	*je* **sé**	*je sais*	'I know'
Spelling errors	*tous mes* **examen**	*tous mes examens*	'All my examinations
	son *normaux*	*sont normaux*	are normal'
	Transverse phenomena		
Contraction	**nimp**	*n'importe quoi*	'rubbish'
	qil	*qu'il*	'that he'
Typographic diaeresis	**c a dire**	*c'est-à-dire*	'namely'
	c t	*c'était*	'it was'
	Marks of expressiveness		
Punctuation transgression	*Joli !!!!!!*	*Joli !*	'nice!'
Graphemic stretching	*superrrrrrrrr*	*super*	'great'
Emoticons/smileys	*:-), <3*	–	–


That’s all for today!