DENSE VECTORS

Luísa Coheur

TÉCNICO LISBOA

inesc id lisboa

OVERVIEW

- Learning objectives
- Topics
  - The road so far
  - Word Embeddings (general concept)
  - “Classic” dimensionality reduction methods
  - Neural Word Embeddings
  - Neural Sentence Embeddings
  - Representing and Evaluating Word Embeddings
- Key takeaways
- Suggested readings

LEARNING OBJECTIVES

LEARNING OBJECTIVES

- After this class, students should be able to explain:
  - The general concept of dimensionality reduction
  - How the SVD method can be applied to create dense vectors
  - How to create neural word embeddings, namely: skip-gram and BERT
  - How we can create sentence embeddings
  - How to evaluate neural word embeddings
  - Several concepts related to dense vectors

TOPICS

THE ROAD SO FAR

Before

Sparse vectors representation of language

Now

Dense vectors representation of language

THE ROAD SO FAR

- REMEMBER:
  - Distributional hypothesis: “you shall know a word by the company it keeps” (Firth 1957)
  - Distributional semantics: meaning can be represented by numerical vectors rather than symbolic structures

THE ROAD SO FAR

- Up to now: very high-dimensional space
  - Words represented as sparse vectors
    - Long vectors: length = $|V|$, 10.000-50.000
    - Sparse vectors: most entries = 0
      - => need for dimensionality reduction
- Now we will see that:
  - words can be represented as dense vectors
    - Short vectors: 50-1.000 dimensions
    - Dense vectors: most entries ≠ 0

OVERVIEW

- Learning objectives
- Topics
  - The road so far
  - Word Embedding (general concept)
  - "Classic" dimensionality reduction methods
  - Neural Word Embeddings
  - Neural Sentence Embeddings
  - Representing and Evaluating Word Embeddings
- Key takeaways
- Suggested readings

WORD EMBEDDING (GENERAL CONCEPT)

- A word embedding is a representation of a word in a vector space
  - words with similar meanings are (hopefully) represented by vectors that are close to each other in the vector space

cat →	0.6	0.9	0.1	0.4	−0.7	−0.3	−0.2
kitten →	0.5	0.8	−0.1	0.2	−0.6	−0.5	−0.1
dog →	0.7	−0.1	0.4	0.3	−0.4	−0.1	−0.3
houses →	−0.8	−0.4	−0.5	0.1	−0.9	0.3	0.8

man →	0.6	−0.2	0.8	0.9	−0.1	−0.9	−0.7
woman →	0.7	0.3	0.9	−0.7	0.1	−0.5	−0.4
king →	0.5	−0.4	0.7	0.8	0.9	−0.7	−0.6
queen →	0.8	−0.1	0.8	−0.9	0.8	−0.5	−0.9

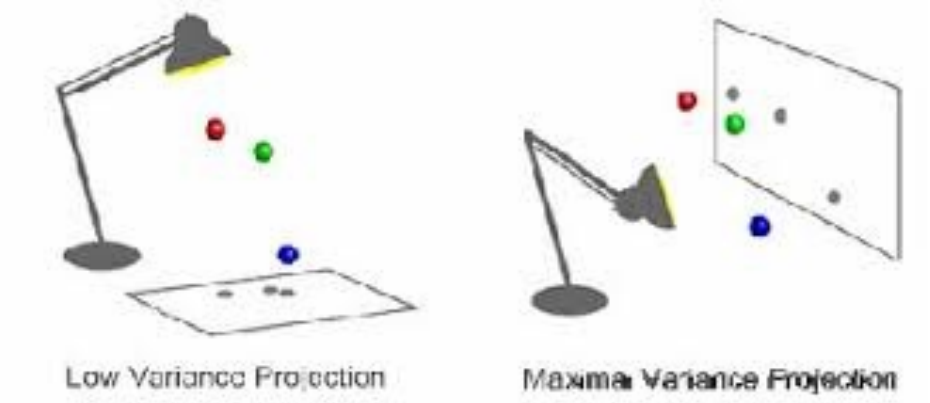
https://medium.com/@hari4om/word-embedding-d816f643140

OVERVIEW

- Learning objectives
- Topics
  - The road so far
  - Word Embedding (general concept)
  - "Classic" dimensionality reduction methods
  - Neural Word Embeddings
  - Neural Sentence Embeddings
  - Representing and Evaluating Word Embeddings
- Key takeaways
- Suggested readings

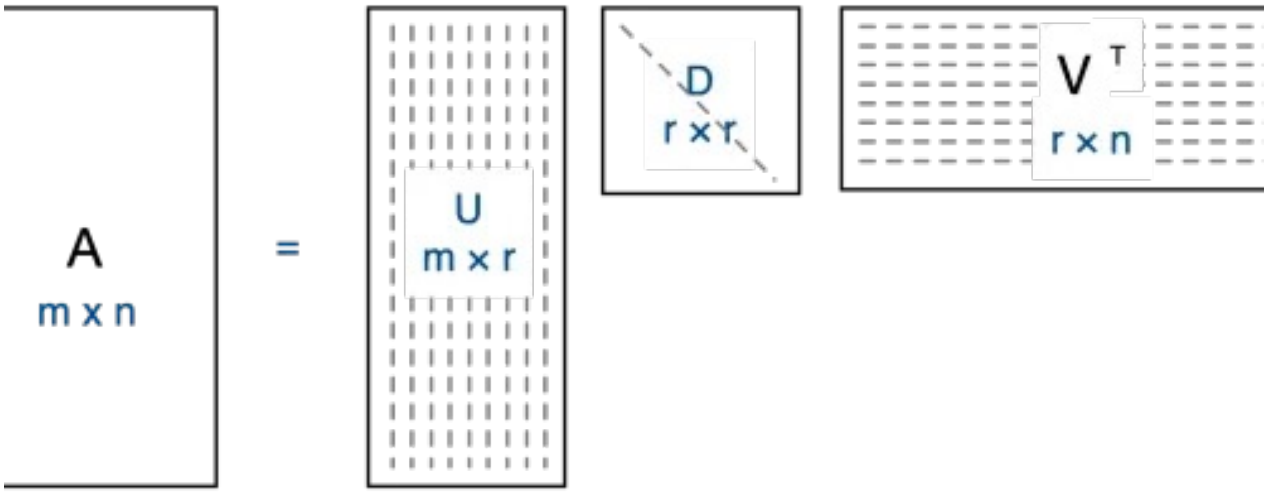
DIMENSIONALITY REDUCTION METHODS

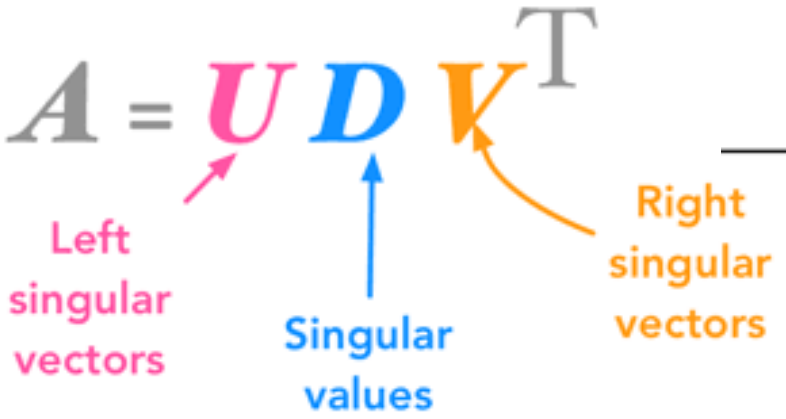
- Goal: reduce the number of dimensions (variables) in a data set without significant loss of information.



SINGULAR VALUE DECOMPOSITION

- Singular Value Decomposition (SVD): method (Algebra) for finding the most important dimensions of a data set (those dimensions along which the data varies the most)





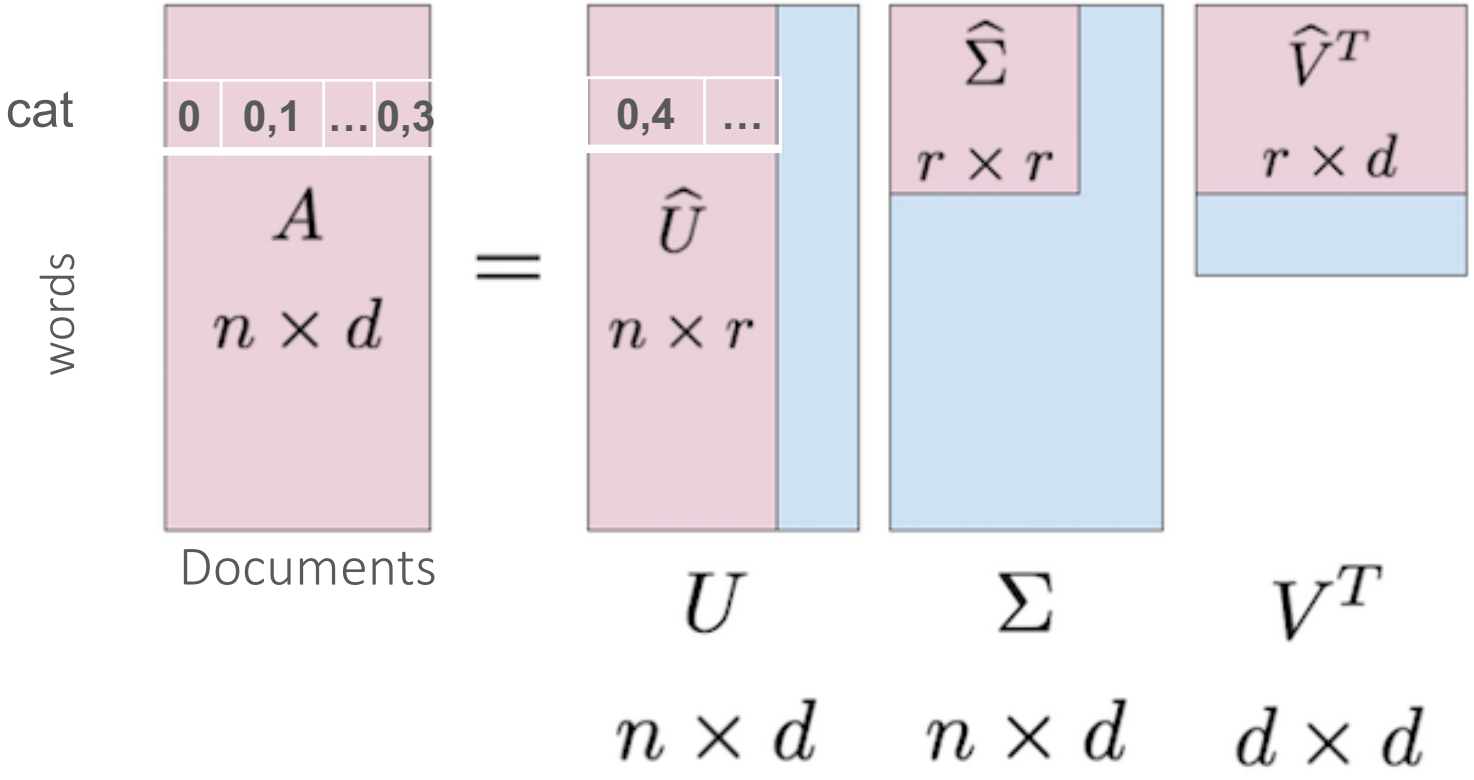
https://medium.com/

https://adel.ac/singular-value-decomposition-in-computer-vision/

Applied to NLP since 1988

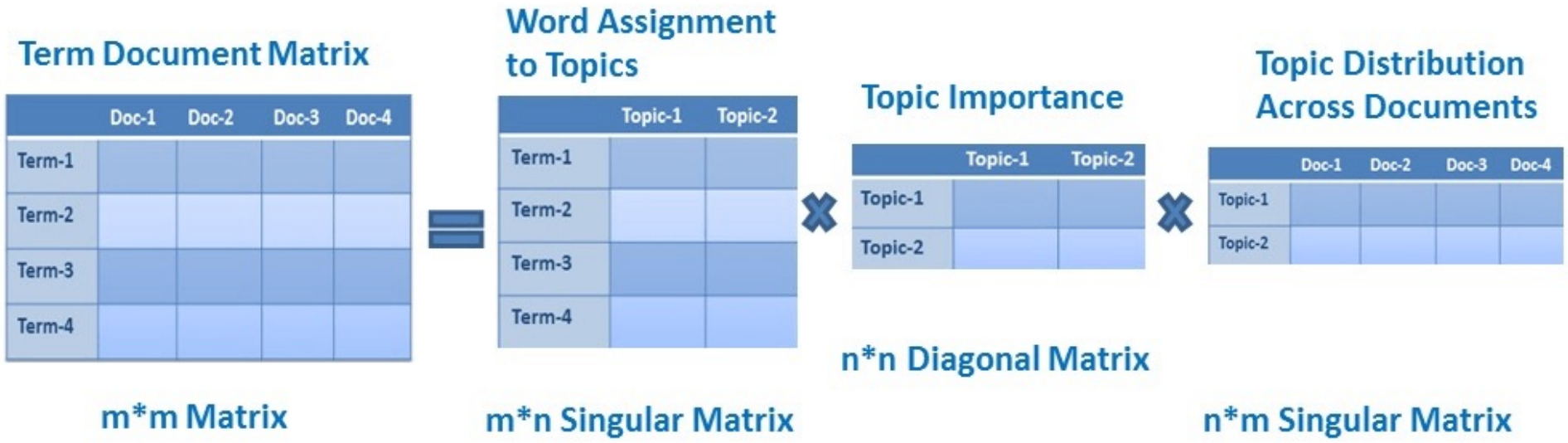
LATENT SEMANTIC ANALYSIS (OR INDEXING)

- Latent Semantic Analysis (or Indexing)(Landauer & Dumais, 1997): reduces the high-dimensional vector space by collapsing the original representation into a smaller (size r) set of latent dimensions. This is done based on the Singular Value Decomposition (SVD) method.



LATENT SEMANTIC ANALYSIS (OR INDEXING)

- Idea (suppose we only want to keep two topics)
  - Intuition: emergence of latent topics



https://www.datacamp.com/community/tutorials/discovering-hidden-topics-python

Latent Semantic Indexing (LSI) An Example

(taken from Grossman and Frieder's *Information Retrieval, Algorithms and Heuristics*)

A "collection" consists of the following "documents":

d1: *Shipment of gold damaged in a fire.*
d2: *Delivery of silver arrived in a silver truck.*
d3: *Shipment of gold arrived in a truck.*

Suppose that we use the term frequency as term weights and query weights. The following document indexing rules are also used:

- stop words were not ignored
- text was tokenized and lowercased
- no stemming was used
- terms were sorted alphabetically

LATENT SEMANTIC ANALYSIS (OR INDEXING)

Problem: Use Latent Semantic Indexing (LSI) to rank these documents for the query *gold silver truck*.

Step 1: Set term weights and construct the term-document matrix **A** and query matrix:

Terms	d1	d2	d3
a	1	1	1
arrived	0	1	1
damaged	1	0	0
delivery	0	1	0
fire	1	0	0
gold	1	0	1
in	1	1	1
of	1	1	1
shipment	1	0	1
silver	0	2	0
truck	0	1	1

A =

LATENT SEMANTIC ANALYSIS (OR INDEXING)

Step 2: Decompose matrix **A** matrix and find the **U**, **S** and **V** matrices, where

$A = USV^T$

U contains the Word Embeddings! (dim = 2)

$$U = \begin{bmatrix} -0.4201 & 0.0748 & -0.0460 \\ -0.2995 & -0.2001 & 0.4078 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.1576 & -0.3046 & -0.2006 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.3151 & -0.6093 & -0.4013 \\ -0.2995 & -0.2001 & 0.4078 \end{bmatrix}$$

$$S = \begin{bmatrix} 4.0989 & 0.0000 & 0.0000 \\ 0.0000 & 2.3616 & 0.0000 \\ 0.0000 & 0.0000 & 1.2737 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.4945 & 0.6492 & -0.5780 \\ -0.6458 & -0.7194 & -0.2556 \\ -0.5817 & 0.2469 & 0.7750 \end{bmatrix}$$

$$V^T = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \\ -0.5780 & -0.2556 & 0.7750 \end{bmatrix}$$

LATENT SEMANTIC ANALYSIS (OR INDEXING)

Latent Semantic Analysis — Deduce the hidden topic from the document

$$A = \begin{array}{c} \begin{array}{ccc} d1 & d2 & d3 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \end{bmatrix} \end{array}$$

$$U \approx U_k = \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix}$$

LATENT SEMANTIC ANALYSIS (OR INDEXING)

- LATENT SEMANTIC ANALYSIS
  - If applied to word-to-word matrix:
    - k = most important singular values (top k dimensions)
    - Truncated SVD

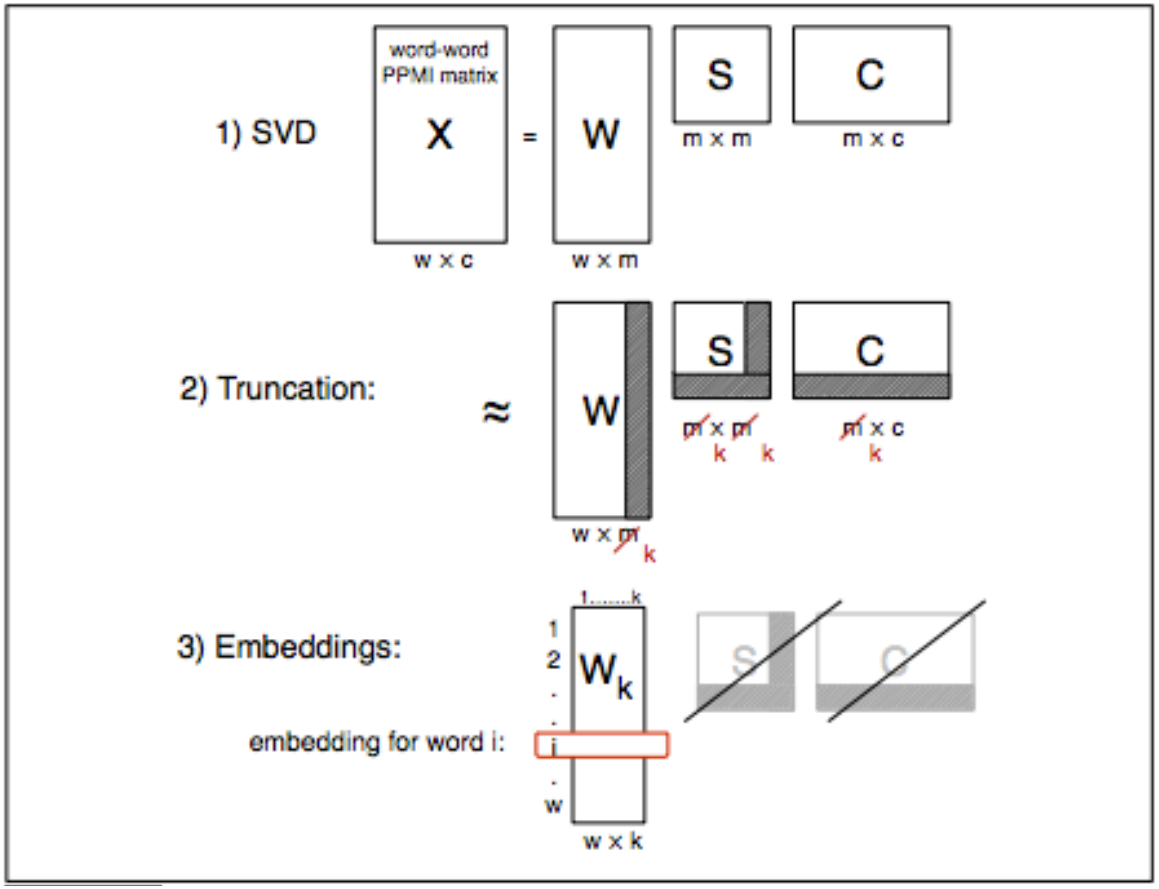


Figure from Jurafsky

RELATED TOPICS

- Topic models: type of statistical models for discovering the abstract "topics" that occur in a collection of documents.
- You have probably studied a dimensionality reduction technique called Principal Component Analysis (PCA)
  - Latent Semantic Analysis (LSA) is essentially truncated SVD, which is mathematically related to PCA.

OVERVIEW

- Learning objectives
- Topics
  - The road so far
  - Word Embedding (general concept)
  - "Classic" dimensionality reduction methods
  - Neural Word Embeddings
  - Neural Sentence Embeddings
  - Representing and Evaluating Word Embeddings
- Key takeaways
- Suggested readings

DENSE VECTORS BASED ON NEURAL NETWORKS

- Neural Word Embeddings:
  - F_θ: words → Rn
    - θ is randomly initialized (thus, random vectors for each word)
    - We learn (Deep Learning) to have meaningful vectors
      - Several methods to do this

WORD2VEC

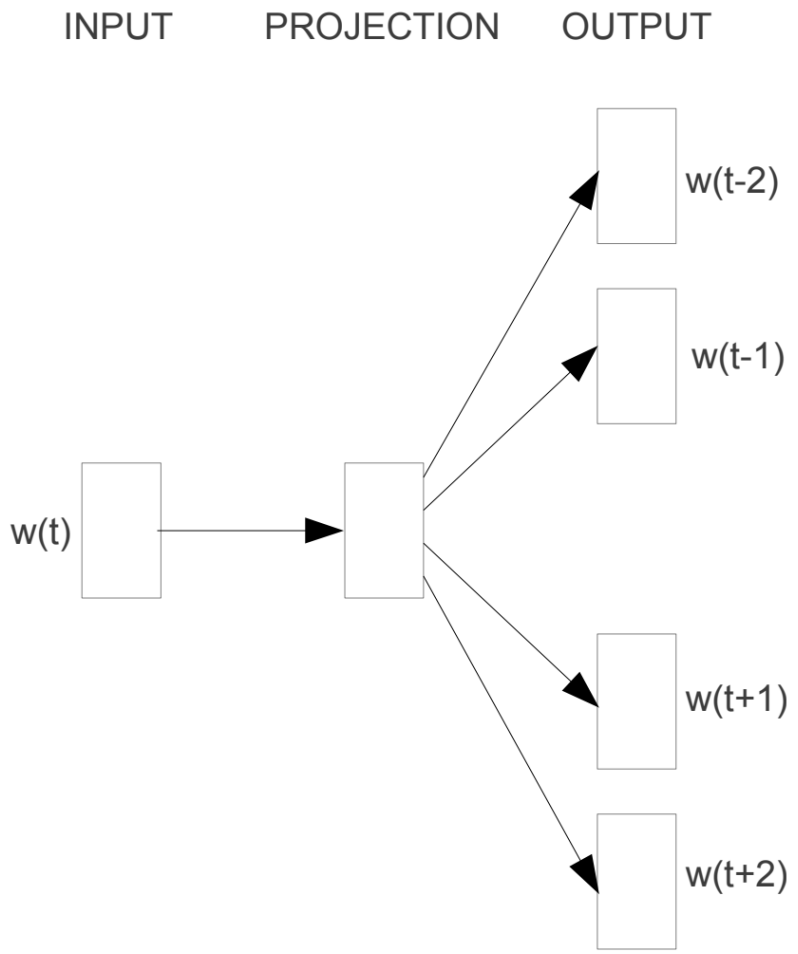
- WORD2VEC (Mikolov et al. 2013 and 2013a): group of models that are used to produce word embeddings.
  - Skip-gram model: uses the current word to predict the surrounding window of context words.
  - CBOW model: predicts the current word from a window of surrounding context words.

TEASER

- Skip-gram and CBOW model perform “Fake” tasks

SKIP-GRAM

- Skip-gram:
  - Task intuition: given the word "Soviet", the output probabilities are going to be higher for words like "Union" and "Russia" than for "table" or "watermelon".

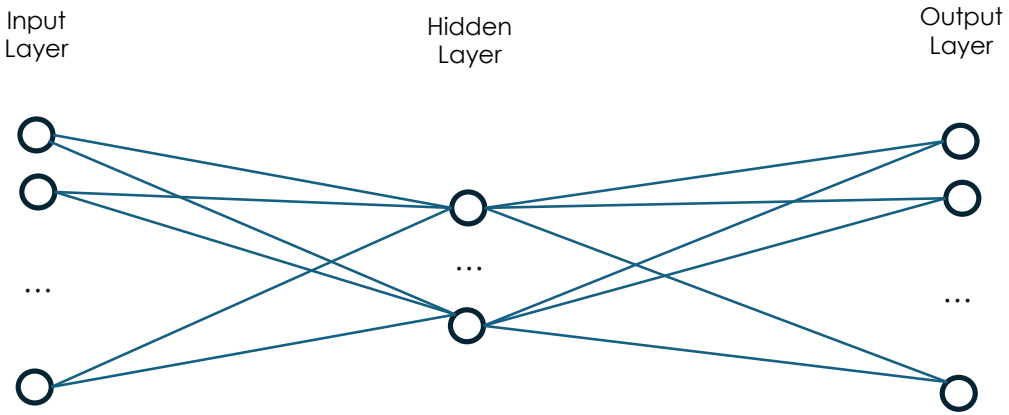


Skip-gram

https://paperswithcode.com/method/skip-gram-word2vec

SKIP-GRAM

- Vocabulary size = |V| (unique words)
- Training:
  - 1-hot input vector with size |V| x 1
  - 1 hidden layer
  - 1-hot output vector with size |V| x 1



Dictionary D= { I; cat; dog; have; a}

1	0	0	0	0
0	**1**	0	0	0
0	0	**1**	0	0
0	0	0	**1**	0
0	0	0	0	**1**

One hot vector

Also, consider the dimension of each dense vector (ex: N = 300)

SKIP-GRAM

- Skip-gram: unsupervised (building the training data)

My cat has powers

Training samples
(window = 1)

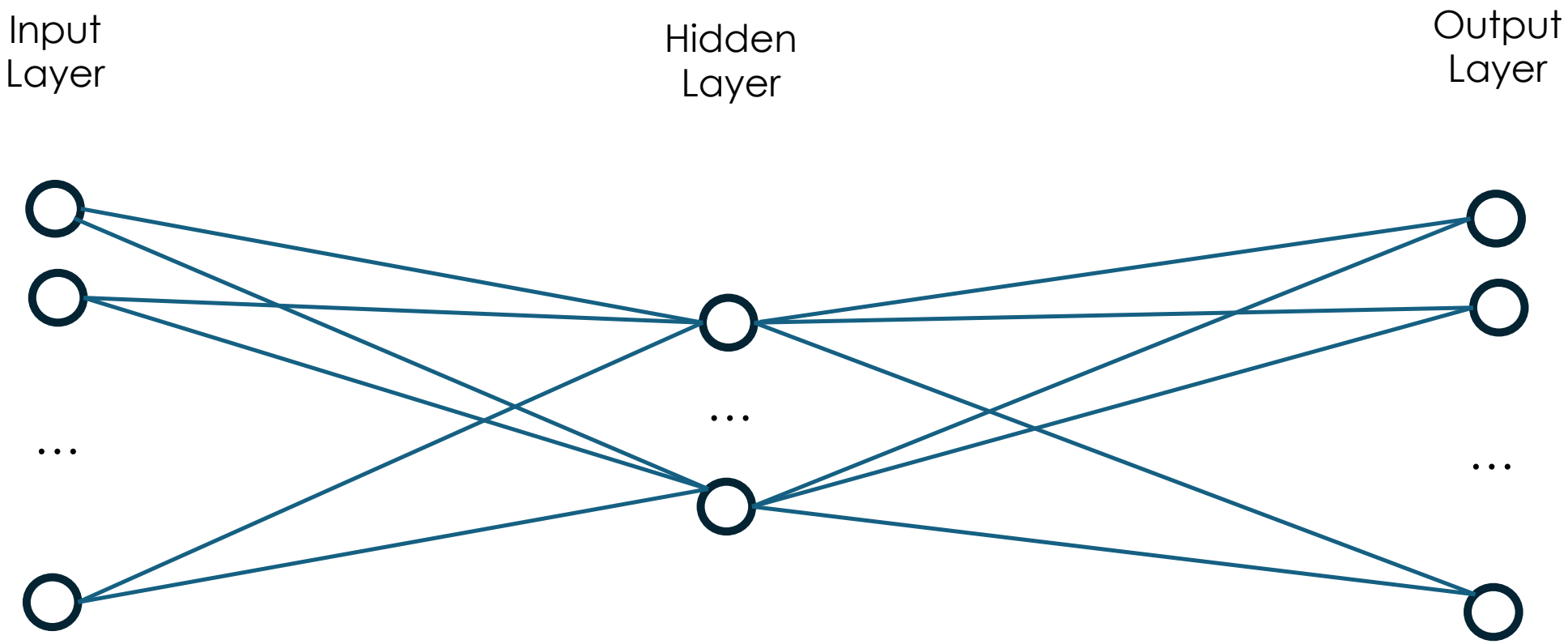
(my, cat)
(cat, my)
(cat, has)
(has, cat)
(has, powers)
(powers, has)

Training samples
(window = 2)

(my, cat)
(my, has)
(cat, my)
(cat, has)
(cat, powers)
(has, my)
(has, cat)
(has, powers)
(powers, cat)
(powers, has)

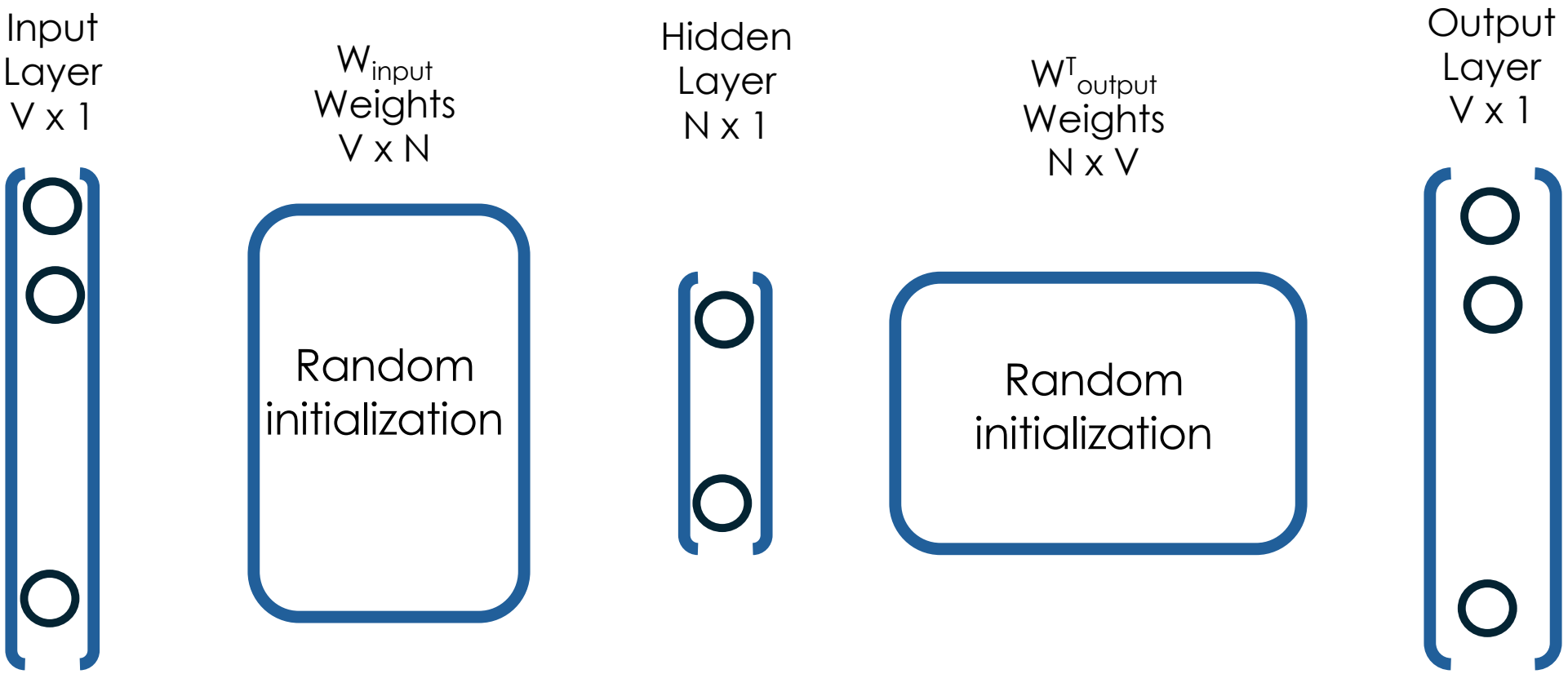
SKIP-GRAM

- Skip Gram model architecture



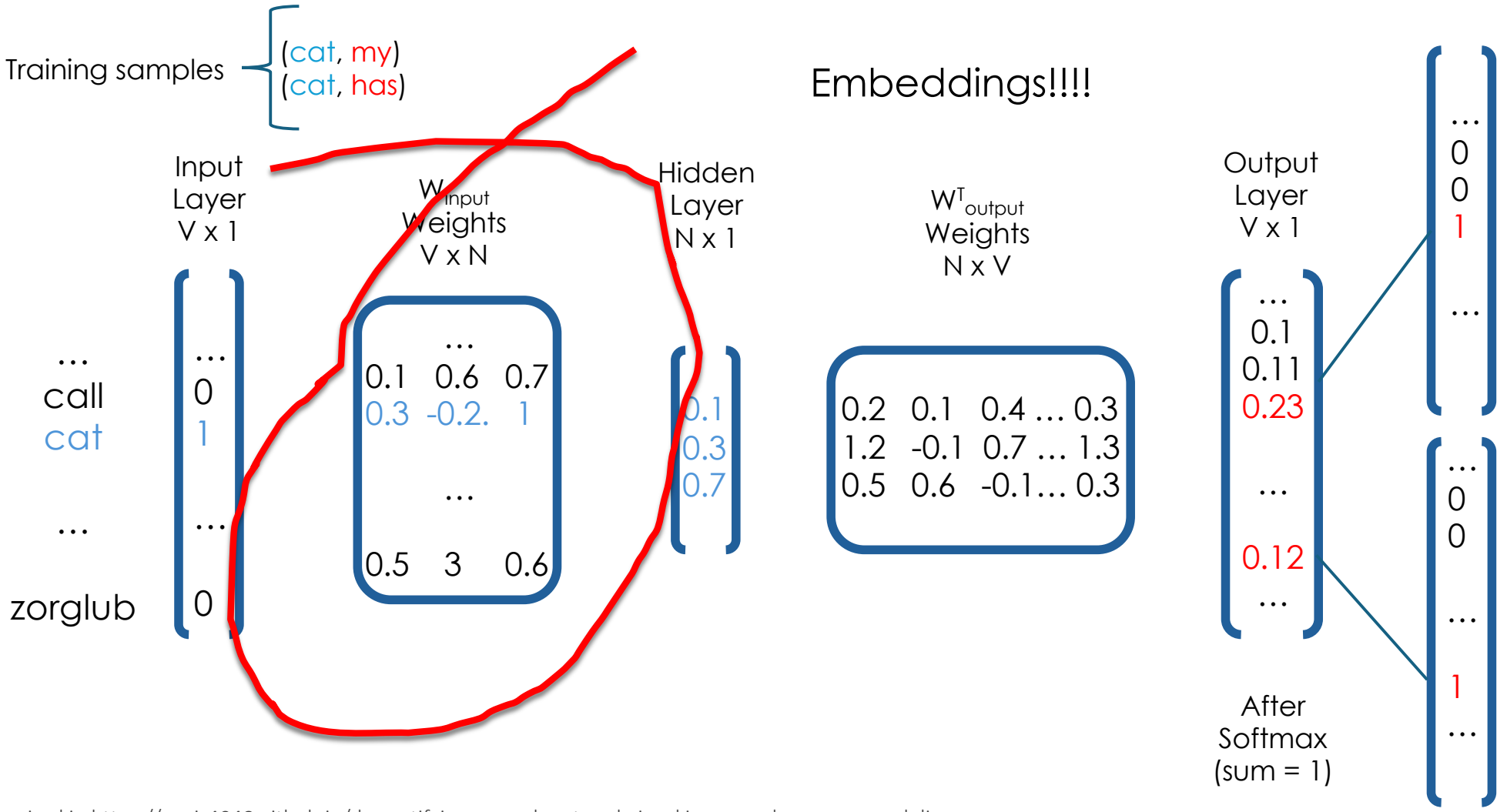
SKIP-GRAM

- Skip Gram model architecture



SKIP-GRAM

- Skip Gram model training



Inspired in https://aegis4048.github.io/demystifying_neural_network_in_skip_gram_language_modeling

SKIP-GRAM

- As each word is represented by a one-hot vector, multiply that vector by the (lookup) matrix and you have the vector that corresponds to that word embedding

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

From Chris McCormick

SKIP-GRAM

- Now you understand why Skip-gram and CBOW model perform “fake” tasks

ACTIVE LEARNING MOMENT: THINK-PAIR-SHARE

EXERCISE

- Write a short comment justifying the veracity/falsity of the sentence:

"Embeddings created with word2vec

are the result of a fake task"

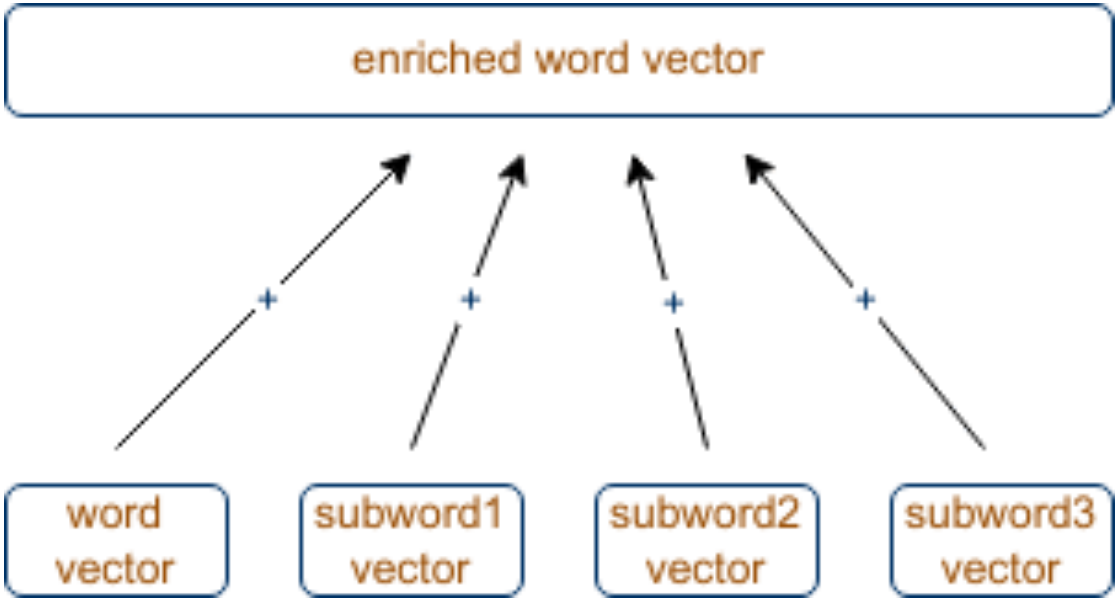
After training (word2vec), the goal is not to apply the created model, but to extract the weights that connect the input and the hidden layers. These weights are going to be the Word Embeddings.

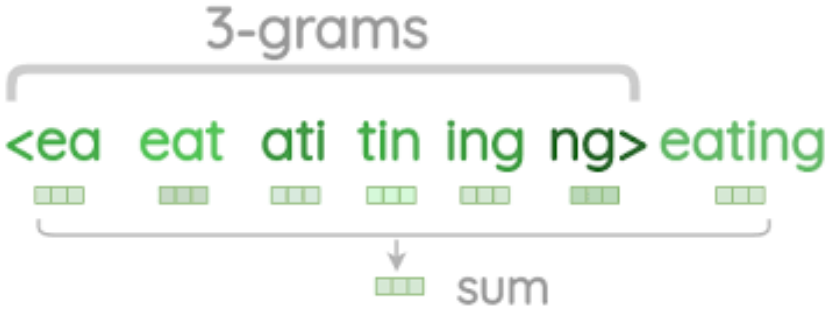
GloVe

- After Word2Vec there were many models to create embeddings.
  - You probably heard about Global Vectors (GloVe) (Stanford, Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014)

fastText

- fastText (Facebook, P. Bojanowski, E. Grave, A. Joulin. 2016) was specifically interesting as it used the concept of subword:
  - a powerful way to handle misspelling words and unknown words.





CONTEXT-FREE MODELS

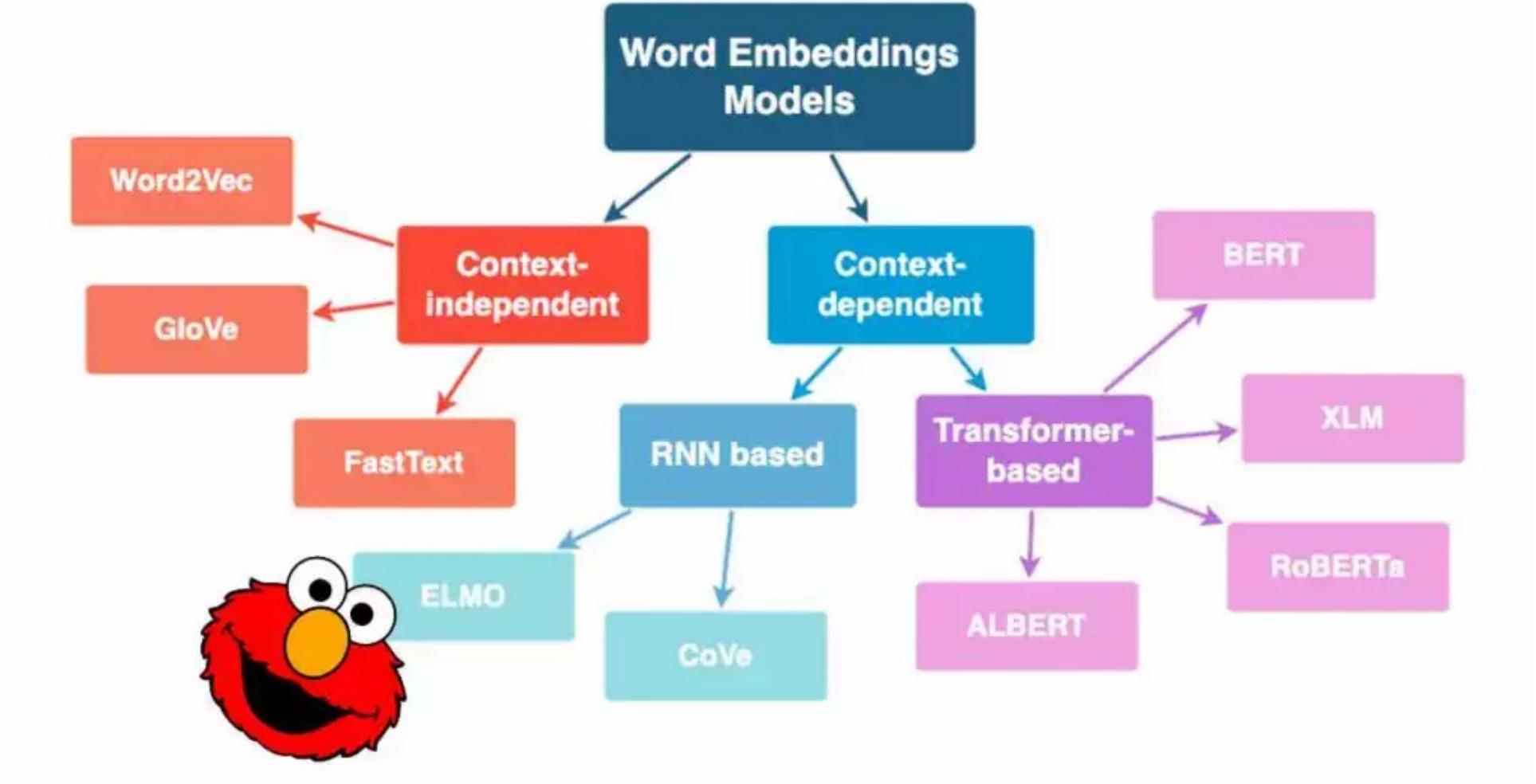
- However, the first models were context-free models: they generated a single word embedding representation for each word in the vocabulary.
  - So, bank has the same representation in bank deposit and river bank

CONTEXTUAL MODELS

- Then, the contextual models arrived: these generate a representation of each word based on the other words in the sentence
  - Examples: ELMo, ULMFit, and BERT

DENSE VECTORS BASED ON NEURAL NETWORKS

- Before we move to BERT (and just to mention a few)

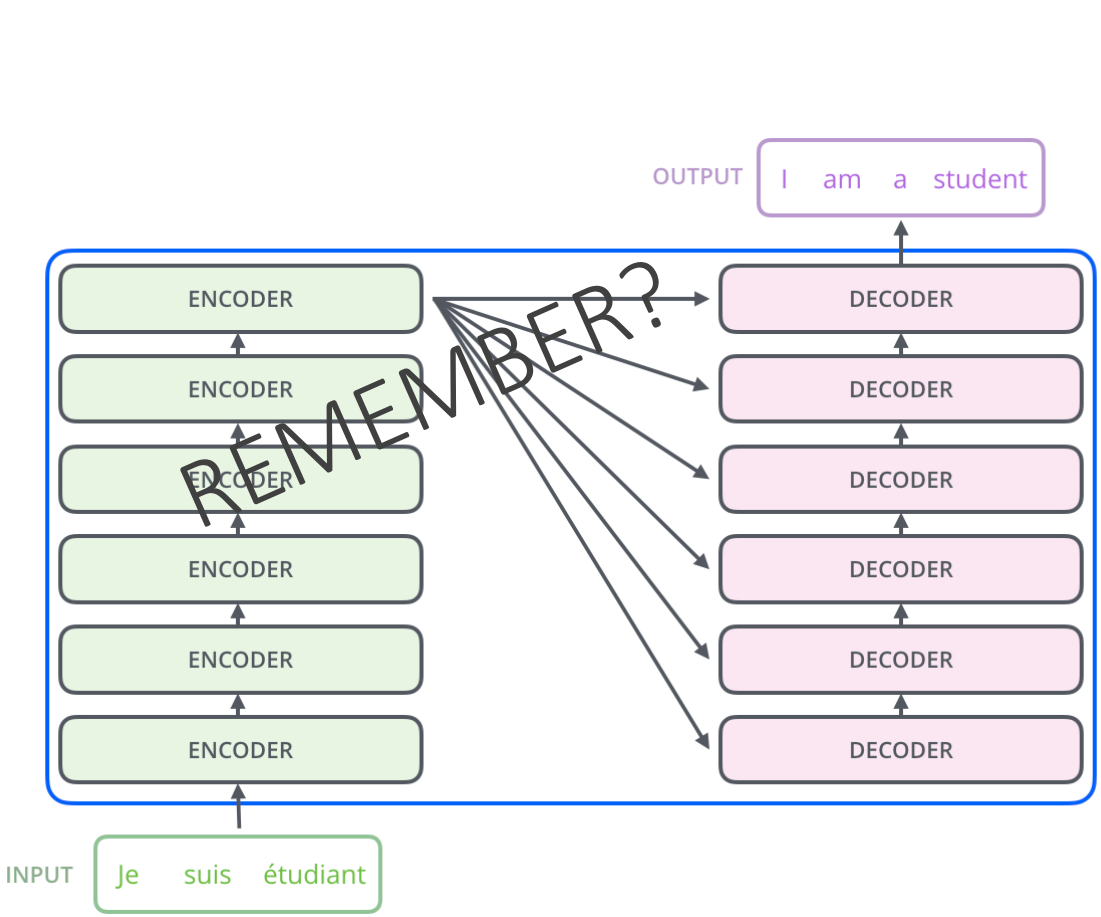


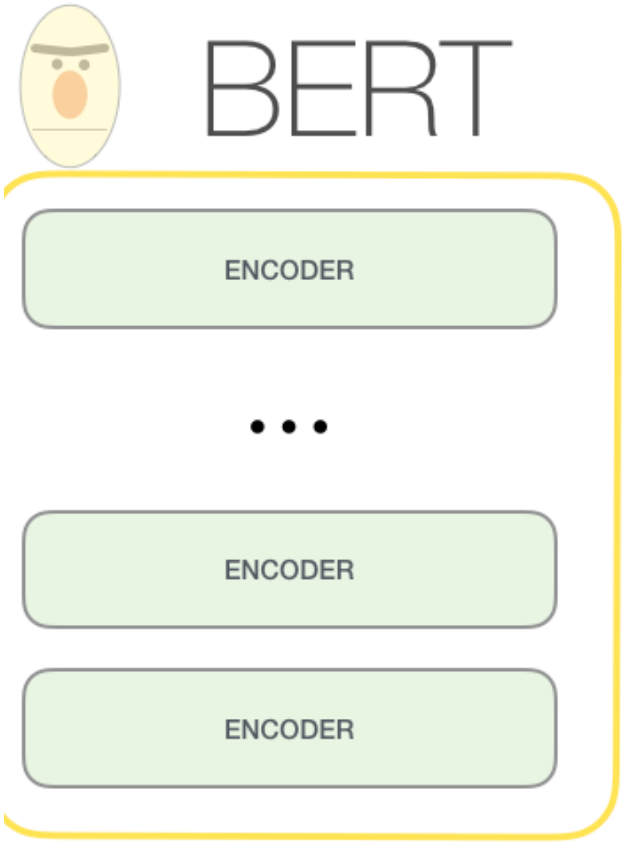
From https://spotintelligence.com

Generated by DALL-E

BERT

- BERT (Google, late 2018)





BERT TASK 1

- BERT is trained in two tasks:
  - TASK 1 (Masked Language Model): BERT masks out 15% of the words in the input, run the entire sequence through a bidirectional Transformer encoder, and then predict only the masked words.

- Example:
  - Input: the man went to the [MASK1] . he bought a [MASK2] of milk. Labels: [MASK1] = store; [MASK2] = gallon

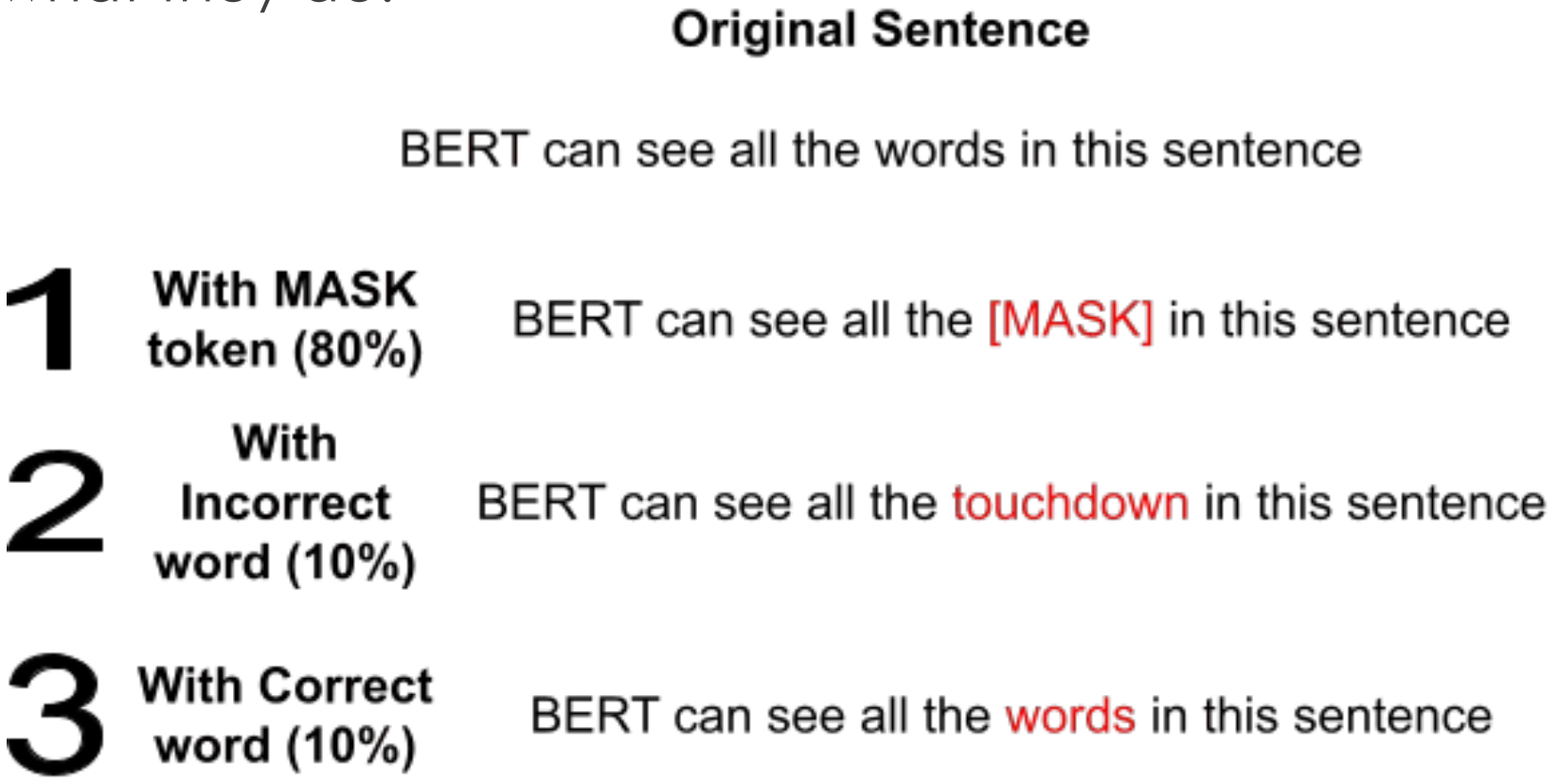
ACTIVE LEARNING MOMENT: THINK-PAIR-SHARE

THINK-PAIR-SHARE

- What do you think bidirectional means in this context?

BERT TASK 1

- BERT Training (TASK 1 – Masked Language Model): for the 15% of selected input tokens we want to mask, what they do:



https://neptune.ai/blog/unmasking-bert-transformer-model-performance

BERT TASK 2

- BERT Training (Task 2 – Next Sentence Prediction): given two sentences A and B, find out if B is the actual next sentence that comes after A, or just a random sentence from the corpus.
  - Example:

Sentence A: the man went to the store .

Sentence B: he bought a gallon of milk .

Label: IsNextSentence

Sentence A: the man went to the store .

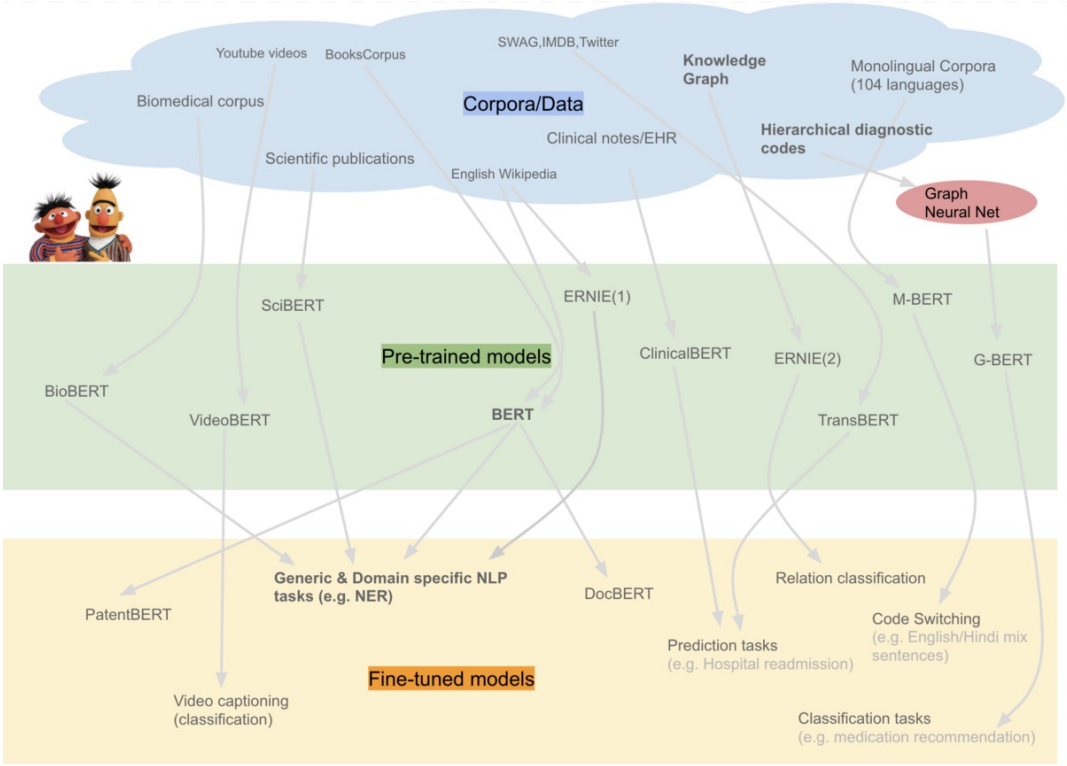
Sentence B: penguins are flightless .

Label: NotNextSentence

MULTI-TASK LEARNING IS ALSO A CURRENT TREND!!

DENSE VECTORS BASED IN NEURAL NETWORKS

- BERT train:
  - A large model is trained (12-layer to 24-layer Transformer) on a large corpus (Wikipedia + BookCorpus) for a long time.
  - There are several different versions of BERT →



And many more…

DENSE VECTORS BASED IN NEURAL NETWORKS

- Using BERT:
  - BERT has two stages: Pre-training and fine-tuning
  - Pre-training is fairly expensive (four days on 4 to 16 Cloud TPUs), but is a one-time procedure
  - Fine-tuning is inexpensive

- We will talk about this a future class

OVERVIEW

- Learning objectives
- Topics
  - The road so far
  - Word Embedding (general concept)
  - "Classic" dimensionality reduction methods
  - Neural Word Embeddings
  - Neural Sentence Embeddings
  - Representing and Evaluating Word Embeddings
- Key takeaways
- Suggested readings

SENTENCE EMBEDDINGS

- And… what about sentence embeddings?
  - Sentence embedding techniques represent entire sentences and their semantic information as vectors

SENTENCE EMBEDDINGS

- There are also models that are trained to create sentence embeddings:
  - Doc2Vec (2014): adds on to the Word2Vec
  - SentenceBERT (2019, Nils Reimers, Iryna Gurevych)
  - InferSent (facebook)
  - Universal Sentence Encoder (Google): multitask learning

SENTENCE EMBEDDINGS

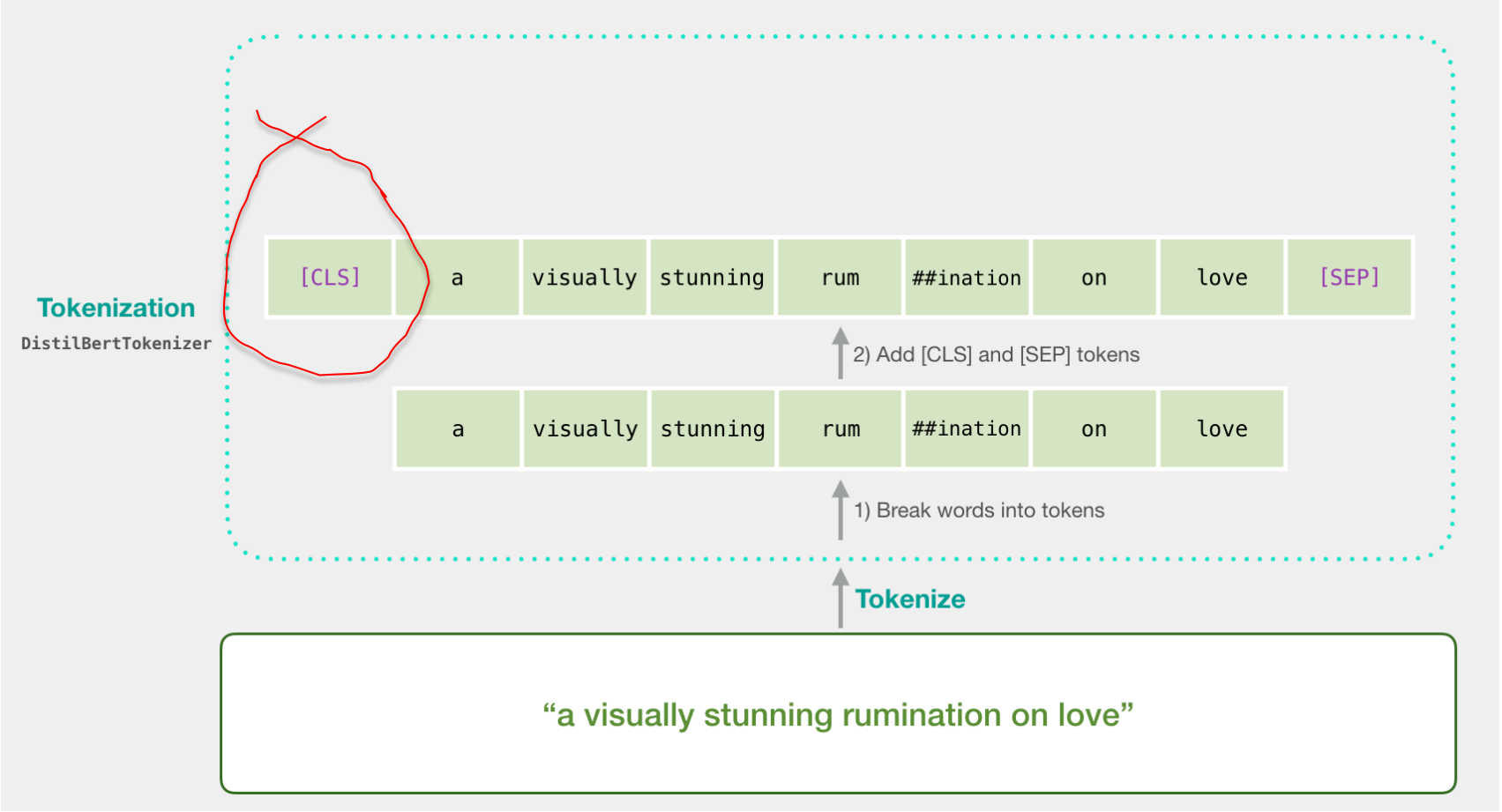
- But… the most straightforward sentence embeddings can be obtained:
  - by summing the word embeddings of the words in the sentence;

  or
  - by taking an average of the word embeddings of the words in the sentence

You must be joking…

SENTENCE EMBEDDINGS

- Or use the CLS token (we will talk about this in a future class)



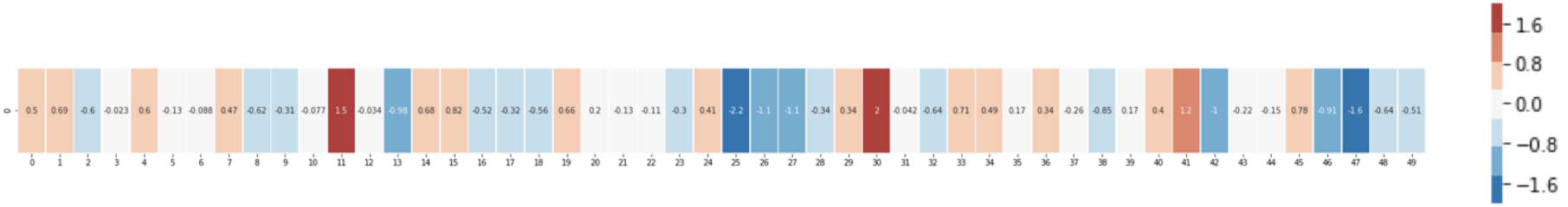
https://discuss.pytorch.org/t/transformers-to-encode-a-sequence-into-a-fixed-lenght-vector/102290

OVERVIEW

- Learning objectives
- Topics
  - The road so far
  - Word Embedding (general concept)
  - "Classic" dimensionality reduction methods
  - Neural Word Embeddings
  - Neural Sentence Embeddings
  - Representing and Evaluating Word Embeddings
- Key takeaways
- Suggested readings

REPRESENTING WORD EMBEDDINGS

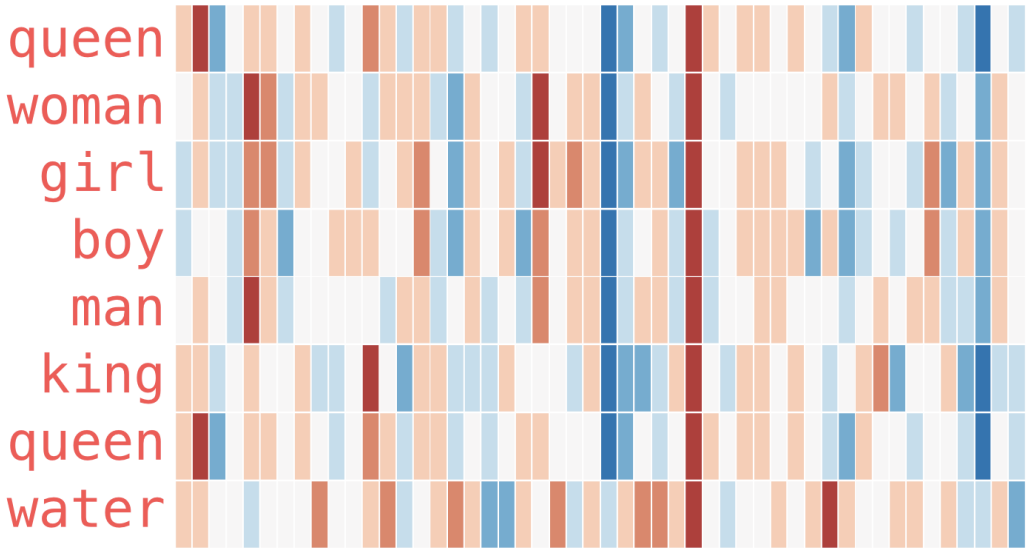
[0.50451 , 0.68607 , -0.59517 , -0.022801, 0.60046 , -0.13498 , -0.08813 , 0.47377 , -0.61798 , -0.31012 , -0.076666, 1.493 , -0.034189, -0.98173 , 0.68229 , 0.81722 , -0.51874 , -0.31503 , -0.55809 , 0.66421 , 0.1961 , -0.13495 , -0.11476 , -0.30344 , 0.41177 , -2.223 , -1.0756 , -1.0783 , -0.34354 , 0.33505 , 1.9927 , -0.04234 , -0.64319 , 0.71125 , 0.49159 , 0.16754 , 0.34344 , -0.25663 , -0.8523 , 0.1661 , 0.40102 , 1.1685 , -1.0137 , -0.21585 , -0.15155 , 0.78321 , -0.91241 , -1.6106 , -0.64426 , -0.51042]



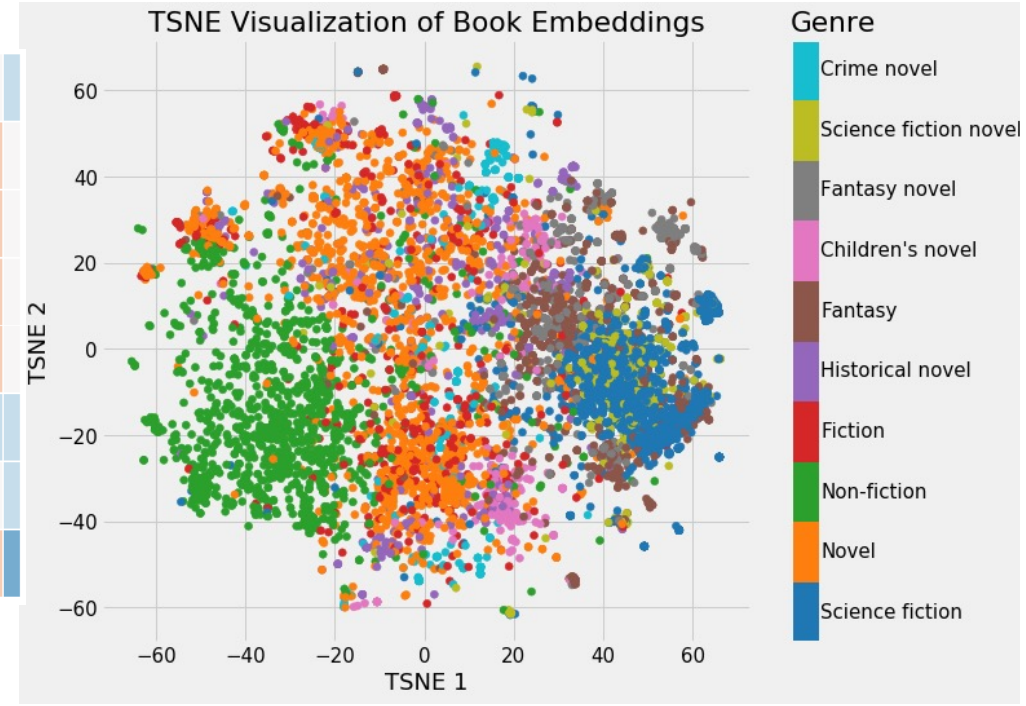
From http://jalammar.github.io/illustrated-word2vec/

REPRESENTING WORD EMBEDDINGS

- https://devopedia.org/word-embedding

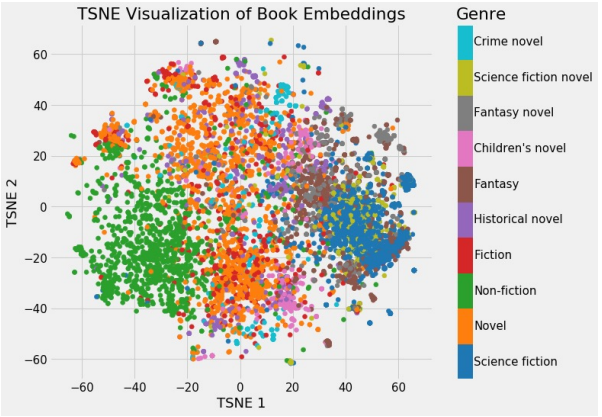


From http://jalammar.github.io/illustrated-word2vec/



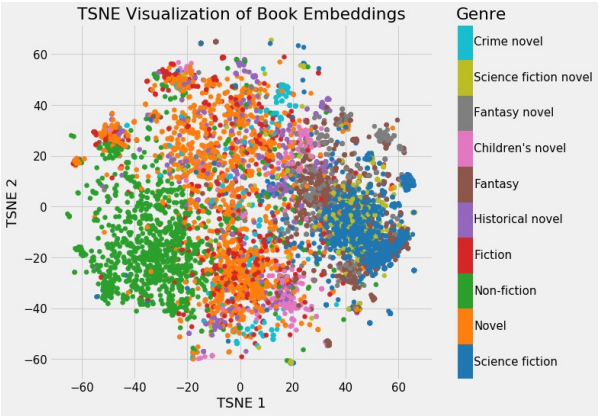
BY THE WAY…

- t-SNE is a dimensionality reduction technique designed to visualize high-dimensional data (like word embeddings) in 2D or 3D.
  - Input: High-dimensional vectors (e.g., 512-dimensional embeddings)
  - Output: 2D or 3D points that you can plot
  - Goal: Preserve the local structure of the data
    - points that are close in high dimensions remain close in the 2D/3D visualization



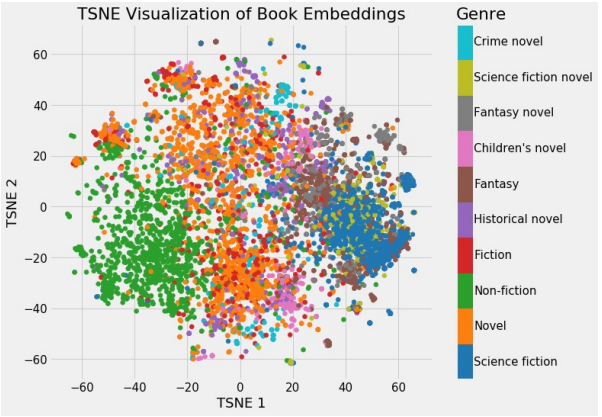
BY THE WAY…

- How t-SNE works (intuitions):
  - Measure distances between points in high dimensions
  - Convert distances to probabilities representing similarity
    - closer points → higher probability
  - Start with a random 2D layout
  - Adjust positions iteratively, so that the 2D probabilities match the high-dimensional probabilities as closely as possible
    - End result: clusters of similar points appear together, separated from dissimilar points.



BY THE WAY…

- And this takes to another concept, also named Perplexity (remember?):
  - Perplexity is a hyperparameter in t-SNE: it tell us how many neighbours a point cares about (each point "attends" to roughly perplexity neighbours when computing similarity)
    - Low perplexity → the algorithm focuses on very local structure (few neighbours).
    - High perplexity → it considers broader neighbourhoods (more neighbours).



EVALUATING WORD EMBEDDINGS

- And… how do we know if we have created good embeddings?
  - Use then in an extrinsic evaluations (remember?)
    - QA, Spell Checking, …
  - Evaluate them in intrinsic evaluations
    - Correlation between algorithm and human word similarity ratings
      - Example: sim(plane, car) = 5,77
    - TOEFL multiple-choice vocabulary tests
      - Example: “Levied” is closest in meaning to: Imposed, believed, requested, correlated

EVALUATING WORD EMBEDDINGS

- Language
  - Θfrance – Θfrench ≅ Θmexico – Θspanish
- Gender
  - Θking – Θman ≅ Θqueen – Θwoman
    - => King - Queen + Woman = Man
- Plural
  - Θcar – Θcars ≅ Θapple – Θapples

EVALUATING WORD EMBEDDINGS

- How do we know that our embeddings are not biased?
- From Jurafsky: "For example African-American names like 'Leroy' and 'Shaniqua' had a higher GloVe cosine with unpleasant words while European-American names ('Brad', 'Greg', 'Courtney') had a higher cosine with pleasant words."

ACTIVE LEARNING MOMENT

EXERCISE

- With your colleagues:
  - say how related the words in the table are (score them between -1 and 1).
  - say which is the word closest to: water, rain, person, pencil, tree, flower
  - complete the analogies: woman/girl -> man/?, woman/aunt -> man/?, uncle/boy -> aunt/?, France/Paris -> Portugal/?, water/fire -> sea/?
- Test everything in
  - http://epsilon-it.utu.fi/wv_demo/
  - Think a little bit about this

Word 1	Word 2
tiger	cat
tiger	dog
book	paper
computer	keyboard
computer	internet
plane	car
train	car

KEY TAKEWAYS

KEY TAKEWAYS

- The idea of creating/using dense vectors is not new
  - SVDs can be used for dimensionality reduction
    - You should be able to explain this
- There are many neural word embeddings: everything started with word2vec
- Skip-gram is based on a neural network with a very simple architecture
- You should be able to explain:
  - Skip-gram training
  - Why Word2vec is a context-free model
  - Why BERT is a contextual model
  - BERT training (in two tasks: a masked language model and a next sentence prediction)

KEY TAKEWAYS (CONT.)

- There are several ways to obtain the embeddings of a sentence – you should be able to explain some
- There are several ways to evaluate embeddings – you should be able to describe some ways
- Be careful with bias in embeddings

SUGGESTED READINGS

READINGS

- Jurafsky: 6.8, 6.9, 6.10, 6.11, 11.1, 11.2