University of Ljubljana
Faculty of Computer and Information Science

Data Science WORKSHOP

Modern NLP through large pretrained language models

Prof Dr Marko Robnik-Šikonja
Andrej Miščič, Luka Vranješ

29 September 2020

Datascience@FRI workshop

University of Ljubljana
Faculty of Computer and Information Science

Contents

- Text preprocessing
- Text representations
- Basics of neural networks for text processing
- Neural language models
- BERT and transformers
- Hands-on (downstream task with transformers), lead by Andrej Miščič and Luka Vranješ

Some slides by Dan Jurafsky, Bhaskar Mitra, Nick Craswell, William Hamilton, Jacob Devlin and Jay Alammar

Semantic language technologies

- part of everyday communication in developed countries
  - communication with mobile devices
  - intelligent search
  - digital assistants and intelligent software
  - intelligent cars and other devices
  - electronic toys
  - household appliances
  - machine translation
  - automatic summarization
  - question answering
  - writting aids

- huge progress with deep neural networks
- strong need to cover low-resourced languages

Basic text preprocessing

- document → paragraphs → sentences → words
- words and sentences ← POS tagging
- sentences ← syntactical and grammatical analysis

Text preprocessing

- text normalization: transformation into a standard (canonic) form or any useful form
- upper/lower casing
- rediacritisation (for Slovene)
- notation of acronyms
- standard form of dates, time, and numbers
- stress marks, quotation marks punctuation,
- non-informative words
- spelling, e.g., US or GB
- emoticons, emoji, hashtags, web links
- editing and presentation markup, e.g., html tags
- usually start with tokenization


Primary quotation marks
in European languages
„x“
”x”
(also »x»)
«x»
”x”
(also »x»)
„x“
«x»
("x" for names)
„x“
«x»
»x«
„x“
‘x’
“x”
('x' preferred
in the UK;
"x" preferred
in the US)
“x”
„x”
rarely:
«x»
«x»
„x“
»x«
„x“
«x»
« x »
(with spaces)
„x“
»x«
„x”
„x”
«x»
«x»
“x”
„x“
„x“
«x»
(but “x” is
recommended
by El País)
«x»
in Brazil:
“x”
«x»
“x”
more maps at
jakubmarian.com

Text Normalization

- Every NLP task needs to do text normalization:
  1. Segmenting/tokenizing words in running text
  2. Normalizing word formats
  3. Segmenting sentences in running text

Token, type, term

- A *token* is an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing.
- A *type* is the class of all tokens containing the same character sequence.
- A *term* is a (perhaps normalized) type that is included in the system's dictionary. The set of index terms could be entirely distinct from the tokens, e.g., term can be semantic identifiers in a taxonomy, but in practice they are strongly related to the tokens in the document. However, they are usually derived from them by various normalization processes.
- *To sleep perchance to dream*,
- 5 tokens, 4 types (2 instances of *to*)
- if *to* is omitted from the index (as a stop word), then there will be only 3 terms: *sleep*, *perchance*, and *dream*

Is this this simple?

```
## tokenizing a piece of text
doc = "I wrote this sentence"
for i, w in enumerate(doc.split(" ")):
    print("Token " + str(i) + ": " + w)
```

```
Token 0: I
Token 1: wrote
Token 2: this
Token 3: sentence
```

Words, lemmas, word forms, stems

- I do uh main- mainly business data processing
  - Fragments, filled pauses
- Seuss's cat in the hat is different from other cats!
  - **Lemma**: same stem, part of speech, rough word sense
    - cat and cats = same lemma
  - **Wordform**: the full inflected surface form
    - cat and cats = different wordforms

Words

- Lexical analysis (tokenizer, word segmented), not just spaces
- 1,999.00€ 1.999,00€!
- Ravne na Koroškem
- Port-au-prince
- Lebensversicherungsgesellschaft
- Generalstaatsverordnetenversammlungen
- I'm rock 'n' roll
- Languages without spaces (e.g., Chineese)

- Rules, finite automata, statistical models, dictionaries (of proper names), lexicons, ML models

How many words?

N = number of tokens

V = vocabulary = set of types

$|V|$ is the size of the vocabulary

Church and Gale (1990): $|V| > O(N^{½})$

	Tokens = N	Types = |V|
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million

Issues in Tokenization

- `Finland's capital` → `Finland Finlands Finland's ?`
- `what're, I'm, isn't` → `What are, I am, is not`
- `Hewlett-Packard` → `Hewlett Packard ?`
- `state-of-the-art` → `state of the art ?`
- `Lowercase` → `lower-case lowercase lower case ?`
- `San Francisco` → one token or two?
- m.p.h., PhD. → ??

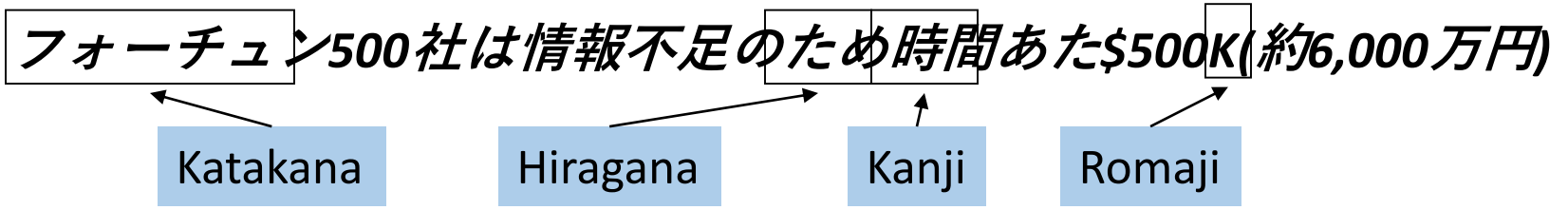
Tokenization: language issues

- French
  - ***L'ensemble*** $\rightarrow$ one token or two?
    - ***L*** ? ***L'*** ? ***Le*** ?
    - Want ***l'ensemble*** to match with ***un ensemble***

- German noun compounds are not segmented
  - ***Lebensversicherungsgesellschaftsangestellter***
  - 'life insurance company employee'
  - German information retrieval needs **compound splitter**

Tokenization: language issues

- Chinese and Japanese no spaces between words:
  - **莎拉波娃**现在居住在美国东南部的佛罗里达。
  - **莎拉波娃** 现在 **居住** **在** **美国** 东南部 **的** **佛**罗里达
  - Sharapova now lives in US southeastern Florida
- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats



End-user can express query entirely in hiragana!

Normalization

- Need to “normalize” terms
  - Information Retrieval: indexed text & query terms must have the same form.
    - We want to match ***U.S.A.*** and ***USA***
- We implicitly define equivalence classes of terms
  - e.g., deleting periods in a term
- Alternative: asymmetric expansion:
  - Enter: ***window*** Search: ***window, windows***
  - Enter: ***windows*** Search: ***Windows, windows, window***
  - Enter: ***Windows*** Search: ***Windows***
- Potentially more powerful, but less efficient

Case folding

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - e.g., ***General Motors***
    - ***Fed*** vs. ***fed***
    - ***SAIL*** vs. ***sail***
- For sentiment analysis, MT, information extraction
  - Case is helpful (***US*** versus ***us*** is important)

Lemmatization

- Reduce inflections or variant forms to base form
  - *am, are, is* → *be*
  - *car, cars, car's, cars'* → *car*
- *the boy's cars are different colors* → *the boy car be different color*
- Lemmatization: have to find correct dictionary headword form
- Machine translation
  - Slovene hočem (‘I want’), hočeš (‘you want’) the same lemma as hoteti ‘want’

Morphology

- **Morphemes**:
  - The small meaningful units that make up words
  - **Stems**: The core meaning-bearing units
  - **Affixes**: Bits and pieces that adhere to stems
    - Often with grammatical functions

Lemmatization

- Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item.
- Lemmatization difficulty is language dependent i.e., depends on morphology
- *English*
  - *walk, walked, walking, walks, ne pa walker*
  - *go, goes, going, gone, went*
- *Slovene*
  - *priti, pridem, prideš, pride, prideva, prideta, pridejo, pridemo, pridete, pridejo, ne pa prihod, prihodnost, prihajanje, prišlec*
  - *vlak, vlaka, vlaku, vlakom, vlakov,vlakoma,vlakih,vlaki, vlake*
  - *jaz, mene, meni, mano*
  - *Gori na gori gori!*
  - *Gori, na gori gori!*
- Use rules, dictionaries, lexicons, machine learning models
- Ambiguity resolution may be difficult

  Meni je vzel z mize (zapestnico).
- Quick solutions and heuristics, in English just remove suffixes: *–ing, -ation, -ed, …*
- essential approach for morphologically rich languages (Slavic, Arabic, Turkish, Spanish, etc)

Dealing with complex morphology is sometimes necessary

- Some languages requires complex morpheme segmentation
  - Turkish
  - Uygarlastiramadiklarimizdanmissinizcasina
  - `(behaving) as if you are among those whom we could not civilize'
  - Uygar `civilized' + las `become'
    + tir `cause' + ama `not able'
    + dik `past' + lar 'plural'
    + imiz 'p1pl' + dan 'abl'
    + mis 'past' + siniz '2pl' + casina 'as if'

Stemming

- stem: the root or main part of a word, to which inflections or formative elements are added
- in English
- simple solution: remove affixes

for example compressed and compression are both accepted as equivalent to compress.

for exampl compress and compress ar both accept as equival to compress

- Stemmer operates on a single word *without* knowledge of the context, and therefore cannot discriminate between words which have different meanings depending on part of speech (meeting: a lemma is to meet or a meeting). Speed!
- Potter algorithm

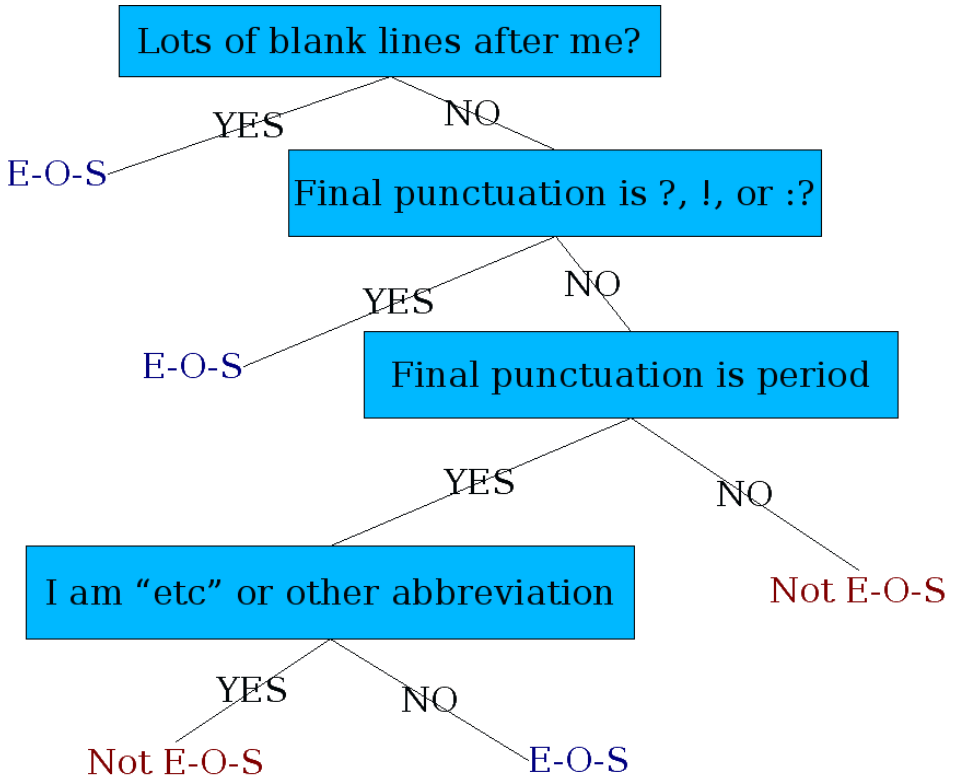
Sentences

- sentence delimiters – punctuation marks and capitalization are insufficient
- E.g., remains of 1. Timbuktu from 5c BC, were discovered by dr. Barth.
- Regular expressions, rules, manually segmented corpora

Sentence Segmentation

- !, ? are relatively unambiguous
- Period "." is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3
- Build a binary classifier
  - Looks at a "."
  - Decides EndOfSentence/NotEndOfSentence
  - Classifiers: hand-written rules, regular expressions, or machine-learning

Determining if a word is end-of-sentence: a Decision Tree



More sophisticated decision tree features

- Case of word with “.”: Upper, Lower, Cap, Number
- Case of word after “.”: Upper, Lower, Cap, Number

- Numeric features
  - Length of word with “.”
  - Probability(word with “.” occurs at end-of-s)
  - Probability(word after “.” occurs at beginning-of-s)

Tools

- every NLP library has tokenizer, sentence delimiter, lemmatizer, e.g., NLTK, spacy

- for Slovene
- https://www.cjvt.si/viri/
- https://github.com/clarinsi
- for nonstandard Slovene (twits, forum messages)
  - ***Nikola Ljubešić, Tomaž Erjavec, Darja Fišer:*** Orodja za procesiranje nestandardne slovenščine. V Fišer, D. (ur). 2018. Viri, orodja in metode za analizo spletne slovenščine. Ljubljana: Znanstveni založbi Filozofske fakultete Univerze v Ljubljani.

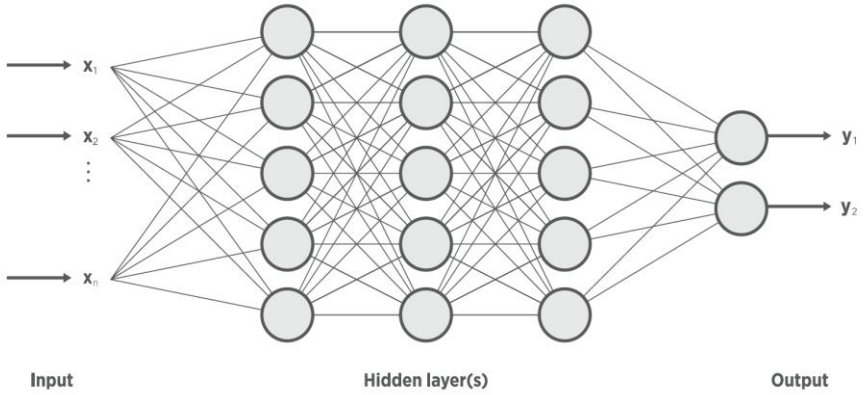
Deep neural networks for text

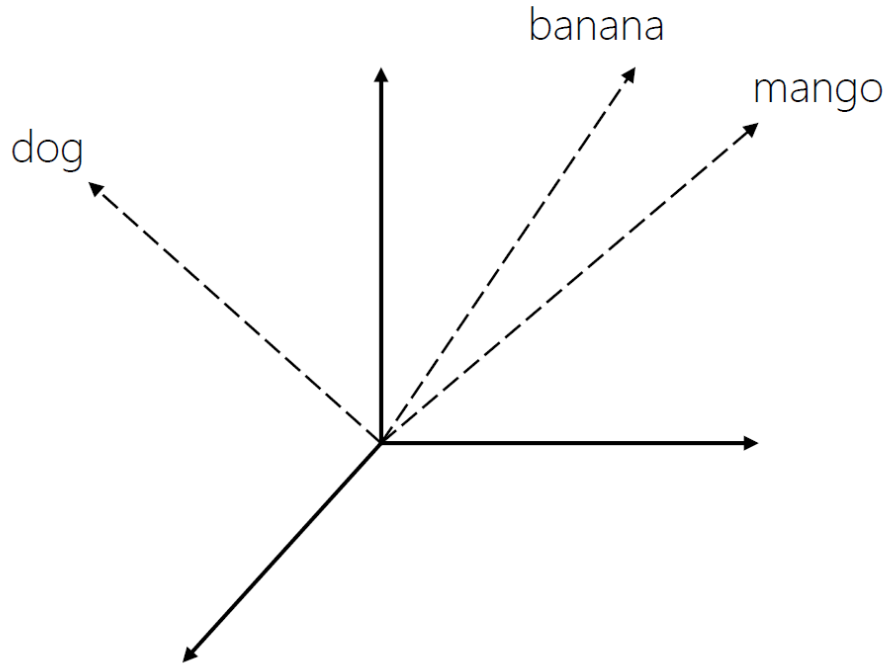
Currently the **most successful approach** to most natural language understanding tasks: machine translation, summarization, questions & answers, text generation, speech recognition and synthesis

Build knowledge representation automatically

Require **text as numeric input**
numeric input shall preserve similarity and relations between words

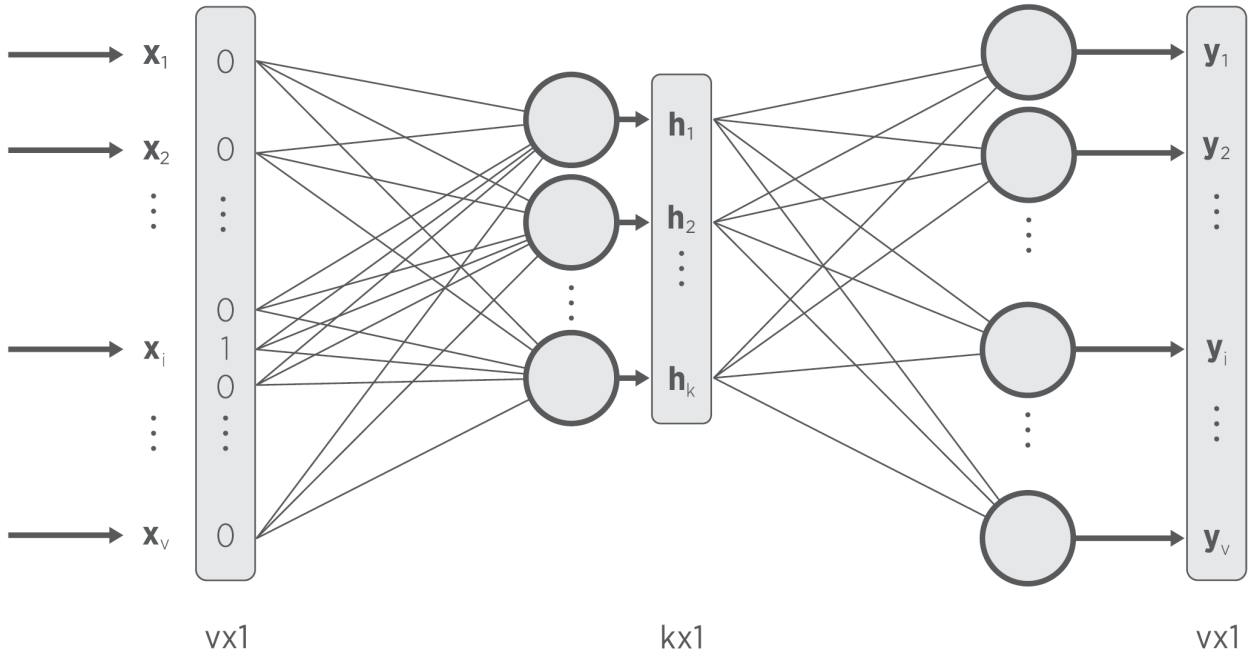
Solution: **text embeddings**





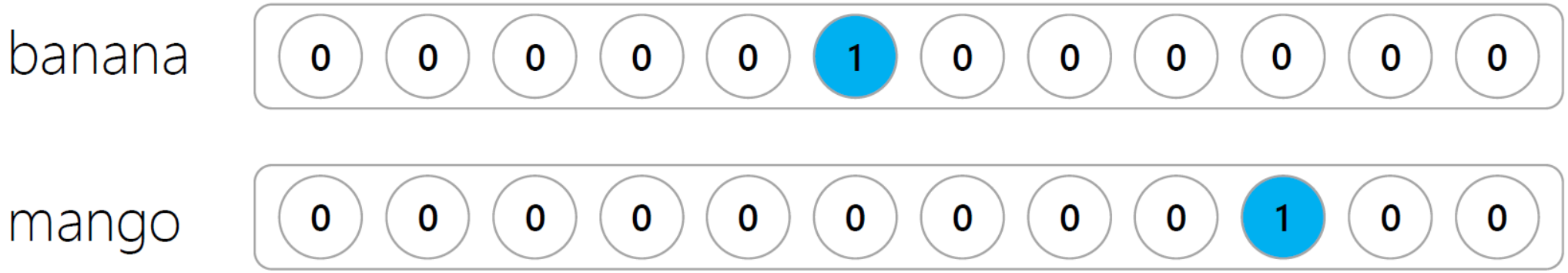
Word embeddings

- Representations of word meaning from **corpus statistics**
- Spatial relationships in *embedded* space correspond to **semantic relationships** expressed with language
- Called an "embedding" because it's embedded into a space
- Vectors are nowadays **learned** with neural networks



Initial textual embeddings

- Best machine learning models for text (SVM, neural networks) require numerical input.
- Simple representations like 1-hot-encoding and bag-of-words do not preserve semantic similarity.
- We need dense vector representation for text elements.



Sparse vector representation

- *An elephant is a mammal. Mammals are animals. Humans are mammals, too. Elephants and humans live in Africa.*

Africa	animal	be	elephant	human	in	live	mammal	too
1	1	3	2	2	1	1	3	1

9 dimensional vector (1,1,3,2,2,1,1,3,1)

In reality this is sparse vector of dimension |V| (vocabulary size in order of 10,000 dimensions)

Similarity between documents and queries in vector space.

But raw frequency is a bad representation

- Frequency is clearly useful; if *sugar* appears a lot near *apricot*, that's useful information.
- But overly frequent words like *the*, *it,* or *they* are not very informative about the context
- Need a function that resolves this frequency paradox!

TF-IDF: combine two factors

- **TF: term frequency**. frequency count (usually log-transformed):

$$\mathrm{tf}_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{count}(t,d) & \text{if } \mathrm{count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

- **IDF: inverse document frequency**

$$\mathrm{idf}_i = \log\left(\frac{N}{\mathrm{df}_i}\right)$$

Total # of docs in collection

# of docs that have word i

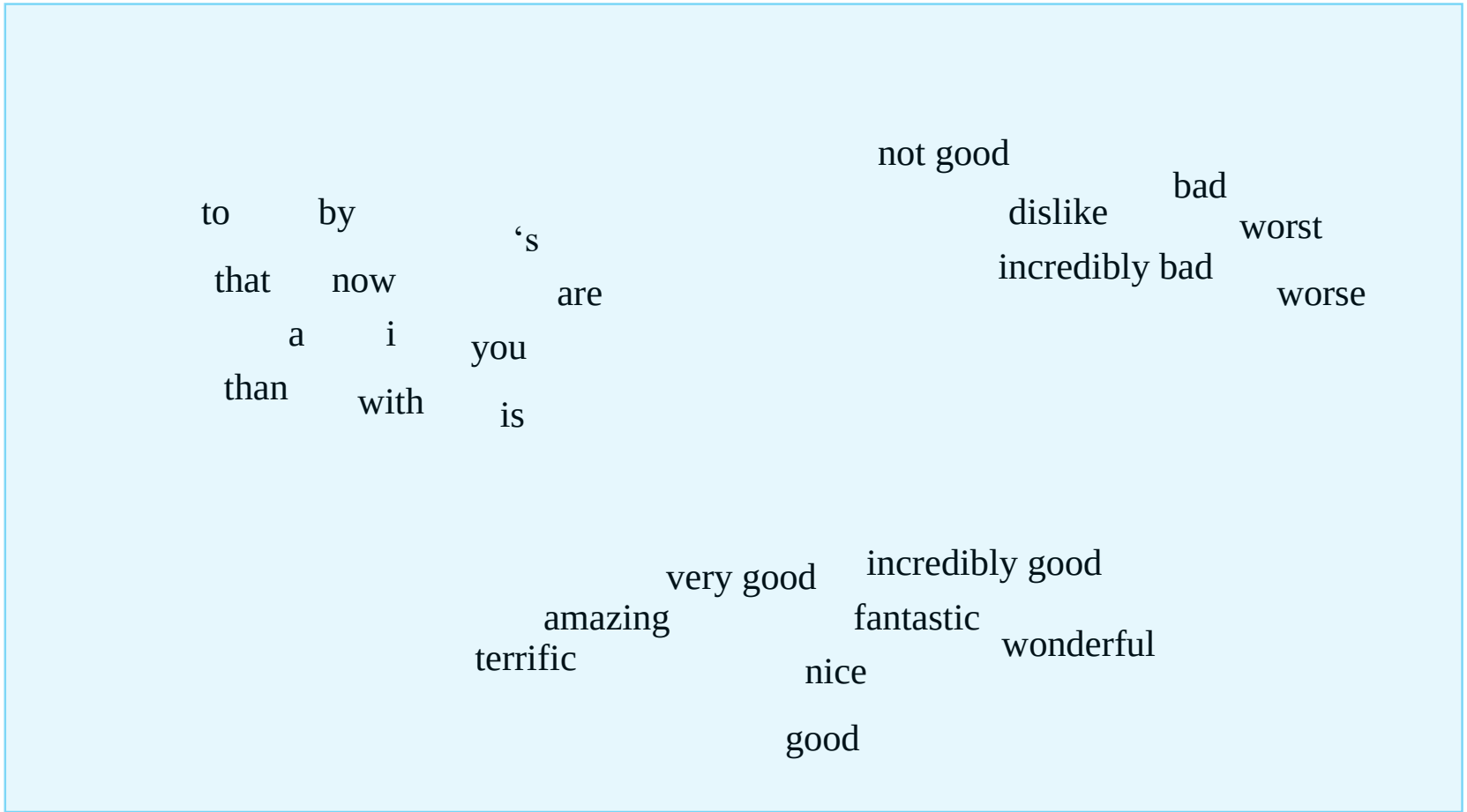
Words like "the" or "good" have very low IDF

TF-IDF value for word t in document d:

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

Dense embeddings: meaning transformed to similarity

- Each word = a vector
- Similar words are "nearby in space"



Distributional semantics

"You shall know a word
by the company it keeps"

Firth, J. R. (1957). A synopsis of linguistic theory 1930–1955. In Studies in Linguistic Analysis, p. 11. Blackwell, Oxford.

"The meaning of a word is its use in the language"

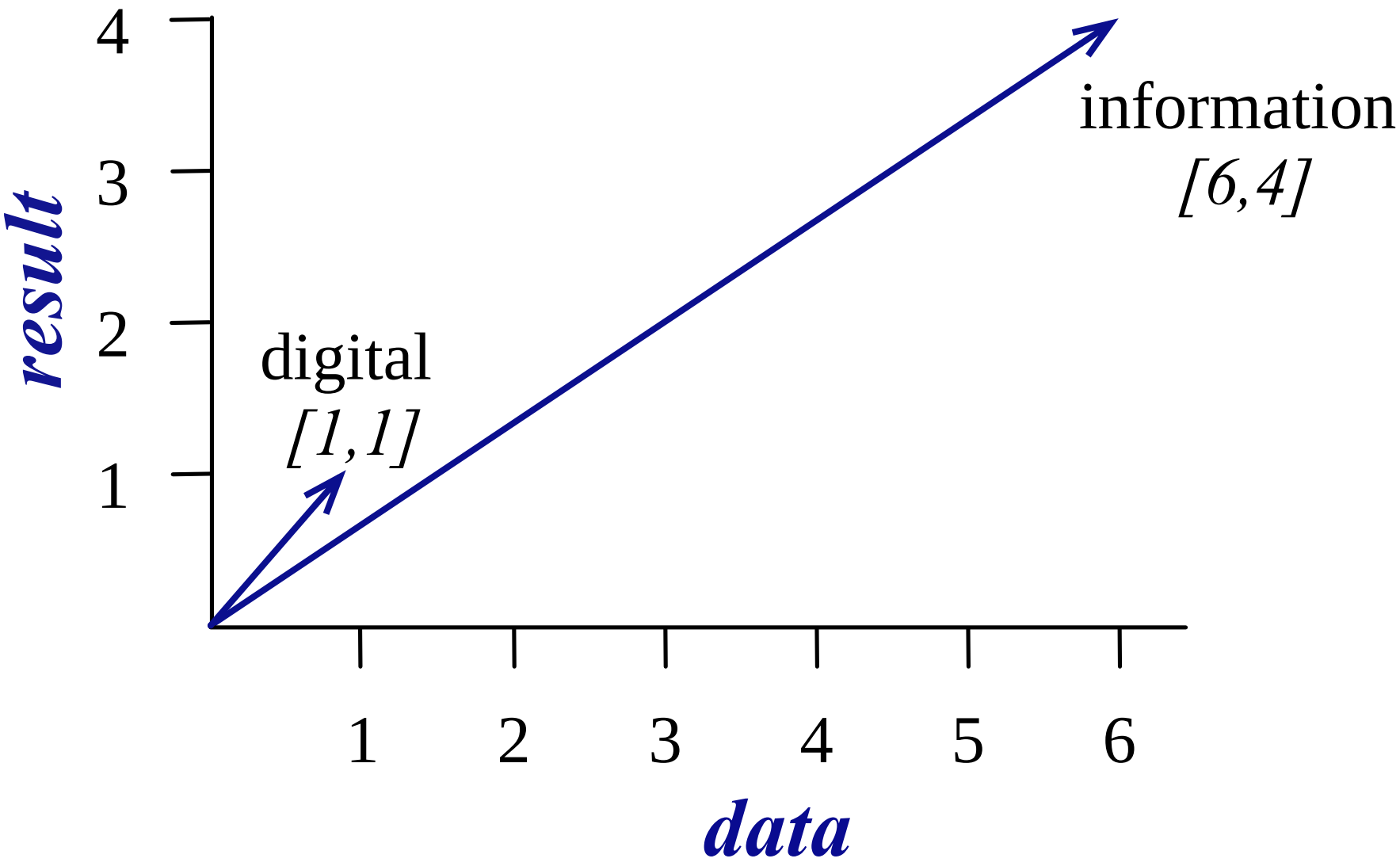
Ludwig Wittgenstein, PI #43

Word-word matrix (or "term-context matrix")

- Two **words** are similar in meaning if their context vectors are similar.

sugar, a sliced lemon, a tablespoonful of	**apricot**	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	


4
3
2
1
result
digital
[1,1]
information
[6,4]
1
2
3
4
5
6
data

Types of dense embeddings for text

- Latent sematic analysis (LSA) - based on word-context matrix decomposition
- Neural embeddings, e.g., word2vec
- Context-sensitive neural embeddings: ELMo and BERT

Idea of LSA – Latent Semantic Analysis

- Decomposition of word-context matrix with SVD
- Approximation with the most important dimensions

Word-word matrix (or "term-context matrix")

- Two **words** are similar in meaning if their context vectors are similar.

sugar, a sliced lemon, a tablespoonful of	**apricot**	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	

SVD for matrices

- SVD (singular value decomposition) for arbitrary matrices, generalizes decomposition of eigenvalues

$$M = U\Sigma V^T$$

- Approximation of N-dimensional space with lower dimensional space (similarly to PCA)
- In machine learning used for feature extraction
- Rotation in the direction of largest variance

Principle components analysis

- Principle components analysis, PCA
- We iteratively find the orthogonal axes of the largest variance
- We use the new dimensions to approximate the original space

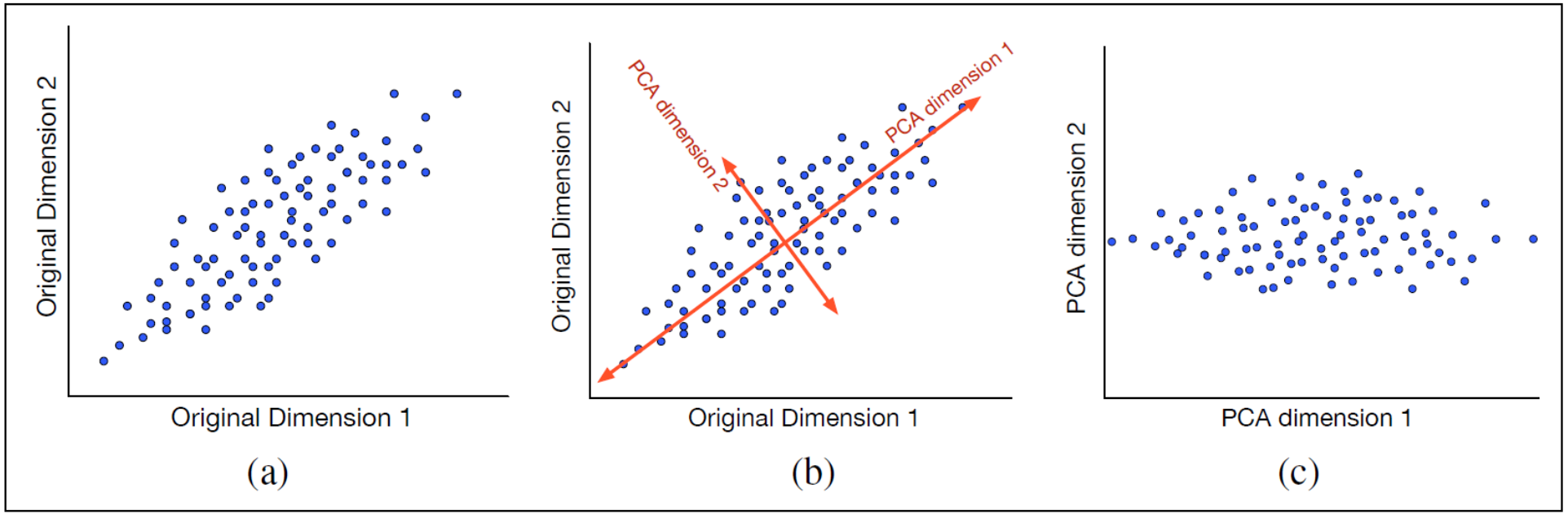
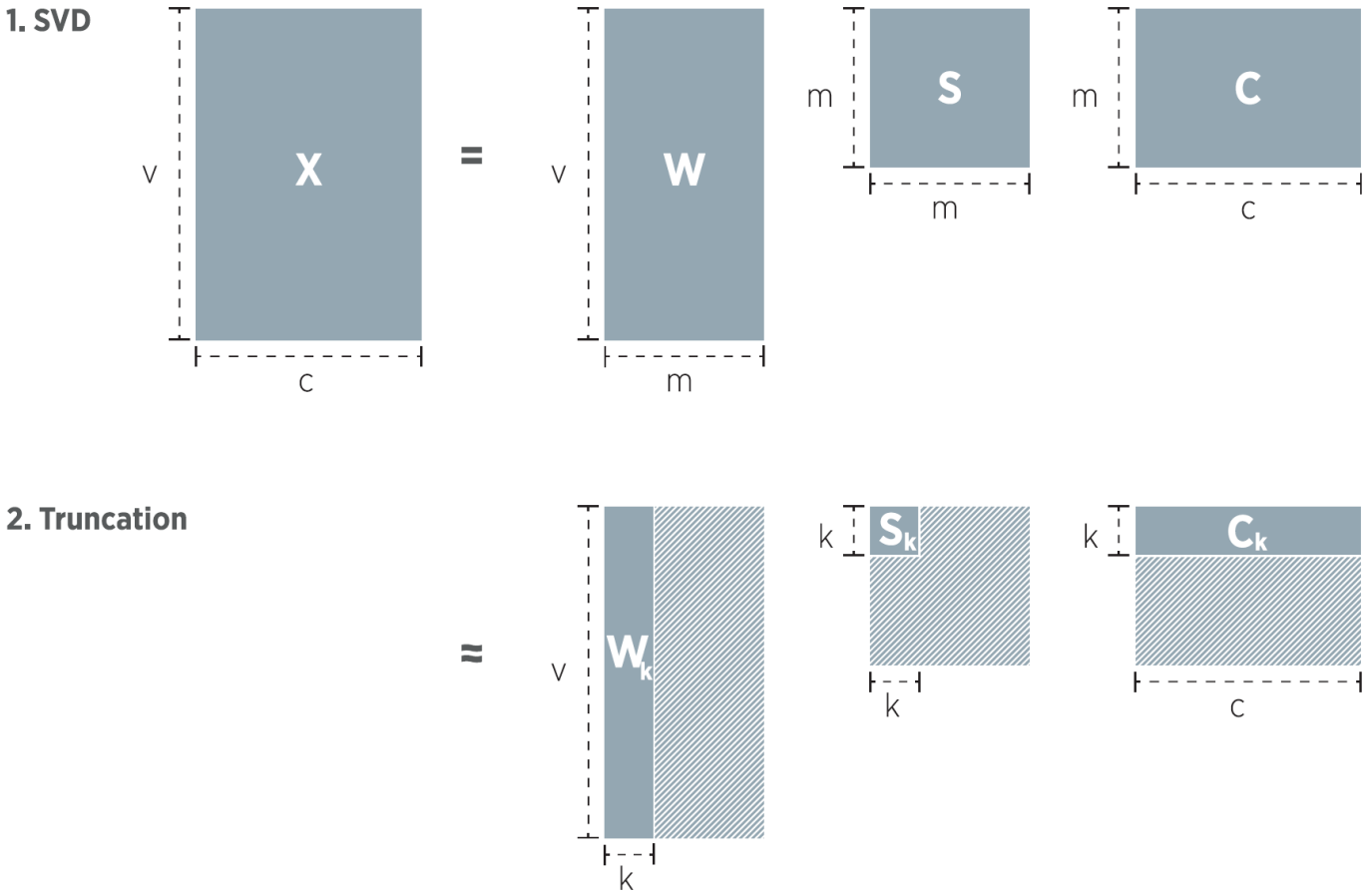


Diagram of LSA

$$\underset{|V| \times c}{\begin{bmatrix} X \end{bmatrix}} = \underset{|V| \times m}{\begin{bmatrix} W \end{bmatrix}} \underset{m \times m}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_m \end{bmatrix}} \underset{m \times c}{\begin{bmatrix} C \end{bmatrix}}$$

$$\underset{|V| \times c}{\begin{bmatrix} X \end{bmatrix}} = \underset{|V| \times k}{\begin{bmatrix} W_k \end{bmatrix}} \underset{k \times k}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_k \end{bmatrix}} \underset{k \times c}{\begin{bmatrix} C \end{bmatrix}}$$

SVD for embeddings



LSA parameters

- Usually k=300 or k=500
- Term context matrix is weighted with local and global weights
- Local weight of each word *i* is log of its frequency in document *j:*
  1+ log f(i, j)
- Global weight of each word is a variant of entropy, where ndocs is the number of documents

$$1 + \frac{\sum_j p(i,j) \log p(i,j)}{\log \text{ndocs}}$$

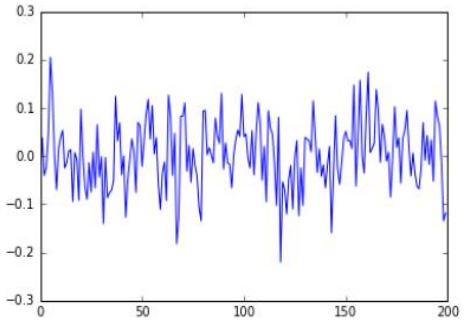
Dense embeddings

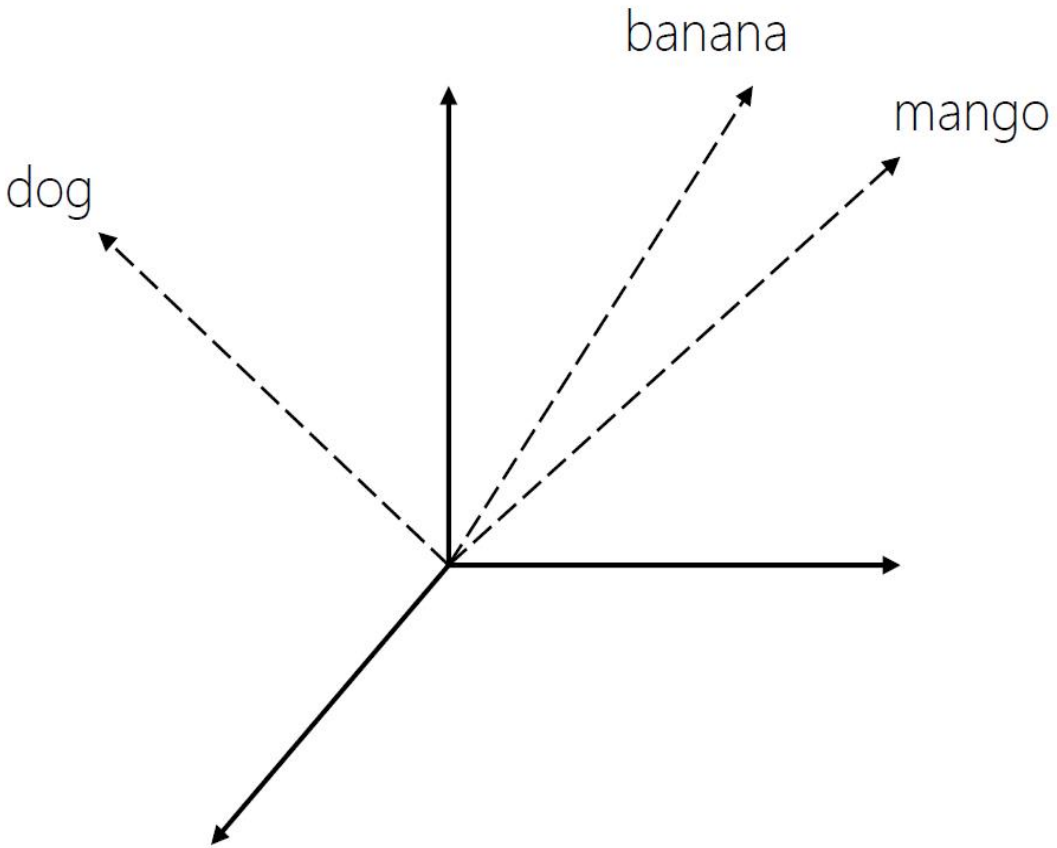
Dense. Dim = 200 (for example)

```
In [67]: print(vec['banana'])
         plt.plot(vec['banana'])
```

```
[-0.065091, 0.037847, -0.040299, -0.022862, 0.046481, 0.204306, 0.132157, 0.000275, -0.069716, 0.014626, 0.038425, 0.053029, -
0.024947, -0.013991, 0.010317, 0.012735, -0.094237, 0.007101, -0.007268, -0.091869, 0.097138, -0.002357, -0.065102, -0.089856,
 -0.013727, -0.074923, 0.007938, -0.066188, 0.064525, -0.0436, -0.001177, -0.140017, -0.003096, -0.086315, -0.0763, -0.071214,
 -0.051458, 0.123467, 0.031151, 0.068839, -0.039029, 4e-06, -0.127185, -0.049415, -0.007708, 0.035502, 0.009538, -0.075545, 0.0
69583, 0.062794, -0.021556, 0.031155, 0.087352, 0.117663, 0.034883, 0.104613, 0.004534, 0.037999, -0.058016, -0.110679, -0.0353
5, -0.012488, -0.0924, 0.126315, 0.080949, -0.040334, 0.047046, -0.182169, -0.1268, 0.082376, 0.082963, 0.110073, -0.031732, 0.
022219, -0.054332, 0.015394, -0.019853, -0.04169, -0.106969, -0.134253, 0.093094, 0.094716, 0.002643, 0.017417, 0.00309, -0.014
145, 0.078464, 0.041464, 0.026328, 0.12988, -0.02715, 0.027002, -0.014312, -0.017305, -0.066002, 0.002747, 0.033995, 0.053829,
 0.040628, 0.127369, 0.040216, 0.045803, -0.003395, -0.024843, 0.052411, -0.039267, 0.043378, 0.110868, 0.067947, -0.050505, 0.
019753, -0.094825, 0.094058, 0.057547, 0.045447, -0.016258, -0.102323, 0.080506, -0.219969, -0.053595, -0.069609, -0.120579, -
0.048799, -0.019837, -0.109987, -0.002571, 0.031825, -0.124037, -0.024646, -0.102276, 0.038512, 0.035166, 0.031713, 0.008979,
 0.114415, 0.0421, -0.034152, 0.014497, -0.04199, -0.018534, -0.065822, -0.020059, 0.019861, -0.159393, -0.03374, 0.083666, -0.
025234, -0.058921, -0.014924, 0.035292, 0.050979, 0.031609, 0.0322, 0.015638, 0.146793, -0.062475, 0.042192, 0.157084, 0.00237
1, -0.035507, 0.08275, 0.173776, 0.007175, 0.016044, 0.025942, 0.137863, 0.094541, -0.013125, 0.065621, 0.040823, -0.010574, 0.
007796, -0.085031, -0.003617, 0.102267, 0.018047, 0.037613, -0.056187, 0.036693, 0.053867, 0.094616, 0.015941, -0.041536, 0.005
796, -0.03694, -0.063241, -0.067796, -0.026023, 0.069142, -0.008786, 0.042428, -0.017718, 0.03318, -0.052277, 0.114012, 0.08154
2, 0.063282, -0.012149, -0.134274, -0.118431]
```

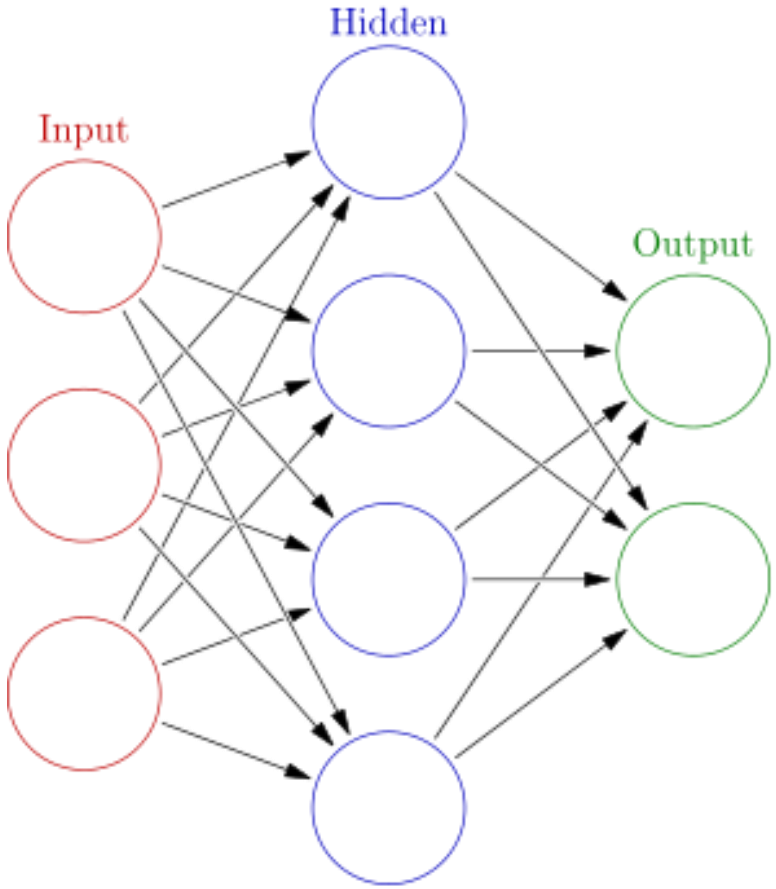
```
Out[67]: [<matplotlib.lines.Line2D at 0x12a60774e48>]
```





Artificial neural networks (ANN)

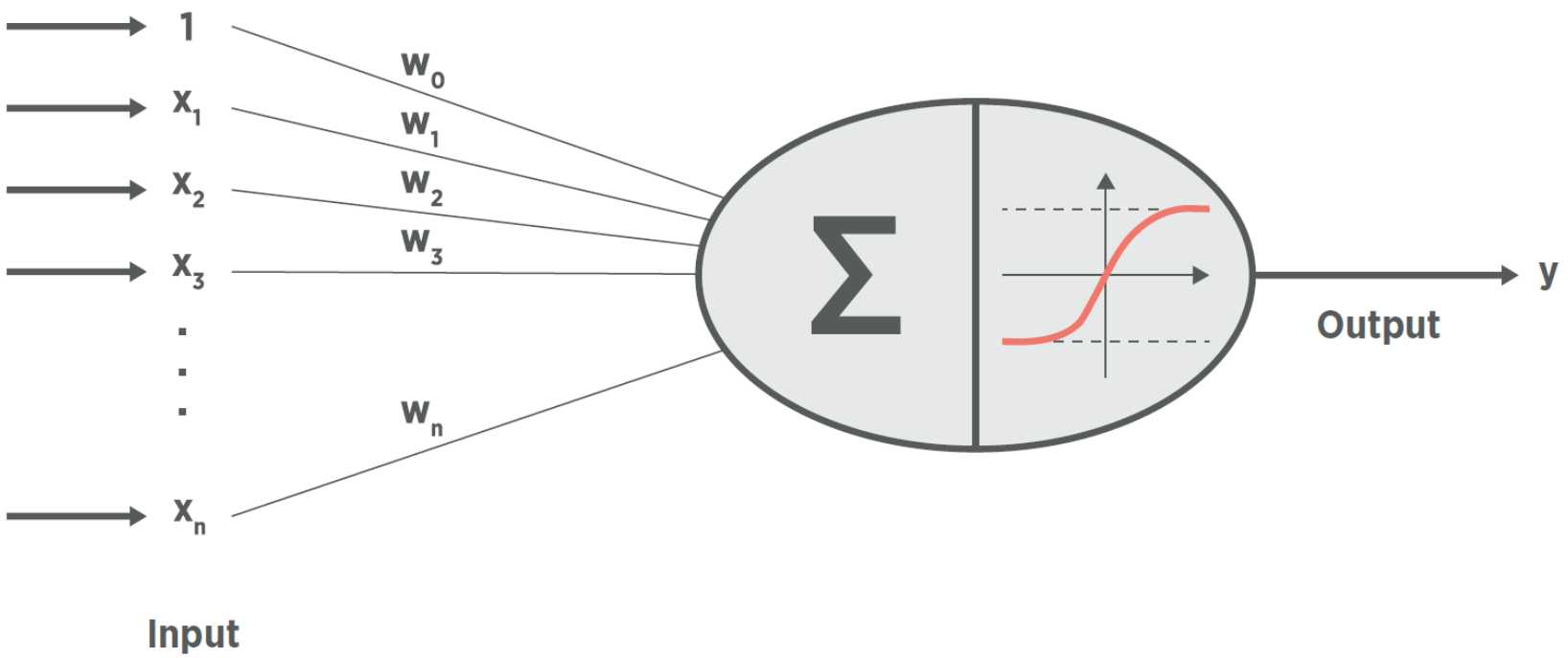
- universal function Approximator
- intuition: neurons in successive layers encode useful features



Artificial neural networks and brain analogy – a neuron

- more than a hundred types of neurons in brain

Artificial neuron



The brain analogy is far from realistic: a neuron cell is highly complex, and so are interconnections.

Activation functions

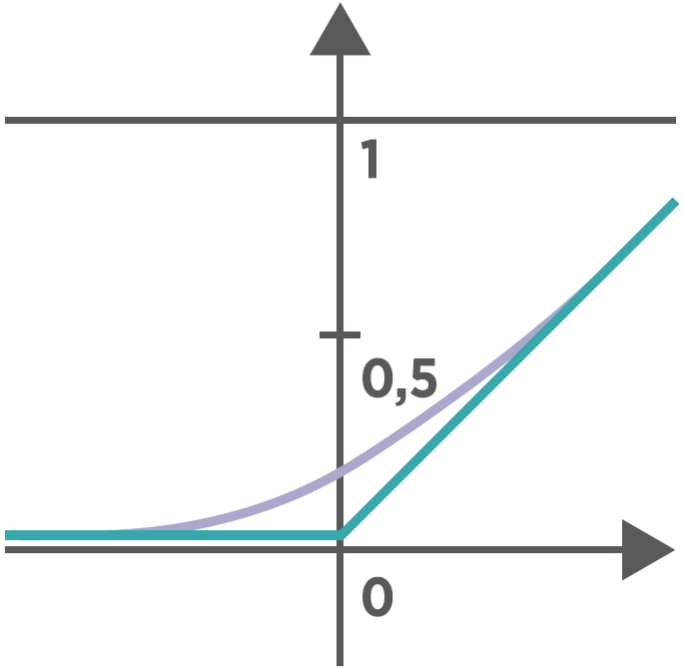
- ReLU (rectified linear unit)

  $f(x) = \max(0, x)$

- softplus / approximation of ReLU with continuous derivation

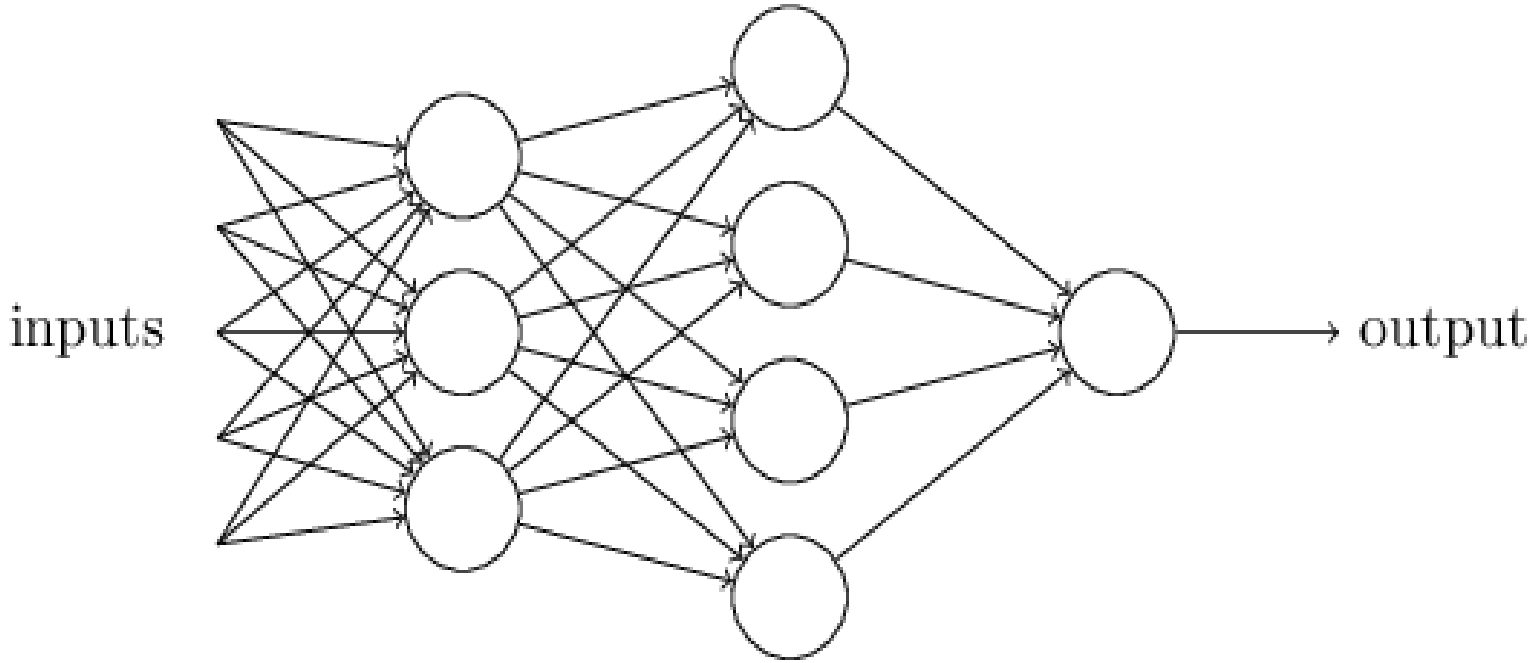
  $f(x) = \ln(1+e^x)$

- many others



Learning: error backpropagation

- a single neuron is weak
- a network of neurons can approximate any continuous function
- deep neural network: more than one hidden level

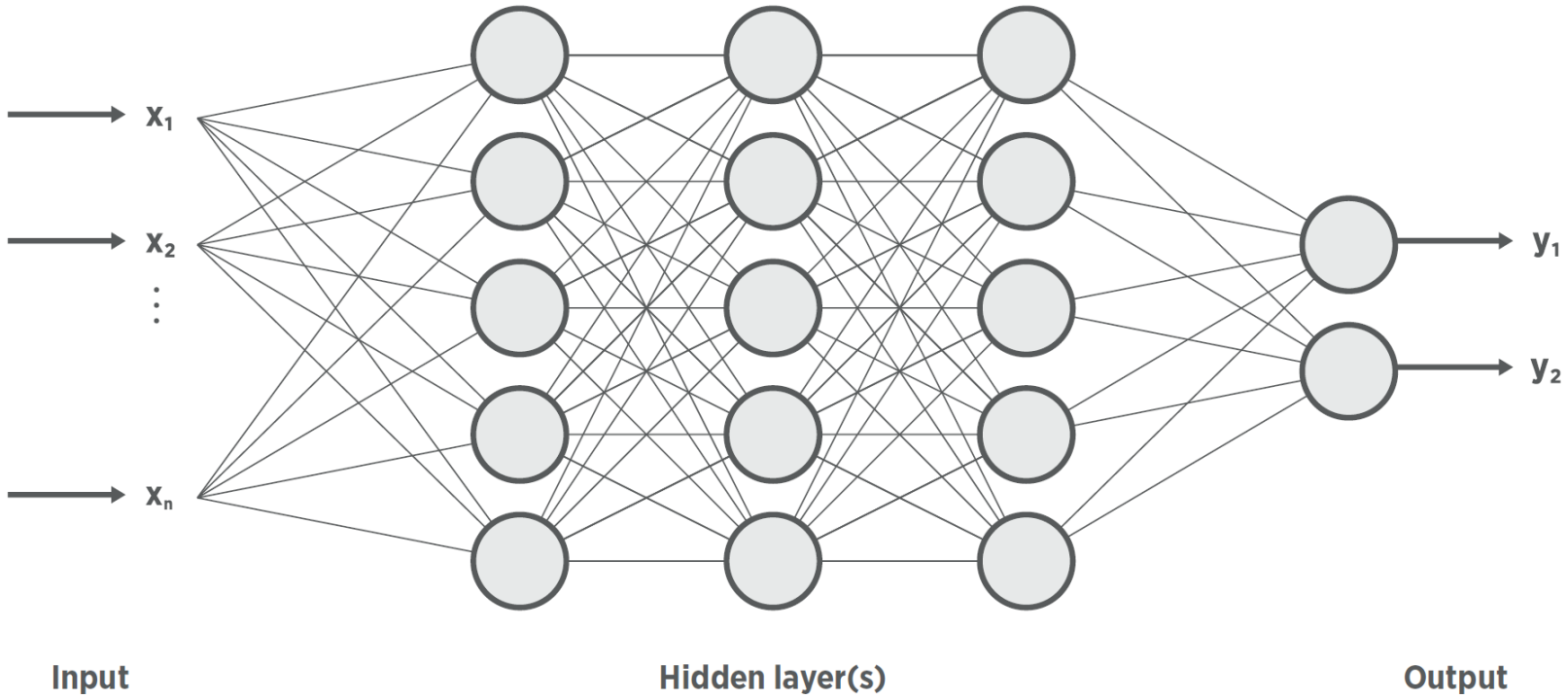


- learning: error backpropagation

Deep learning =

deep neural networks + large data sets + GPU

(+many new ideas)



Why deep learning works?

Why DNNs are better than standard ML approaches?

Feature construction!

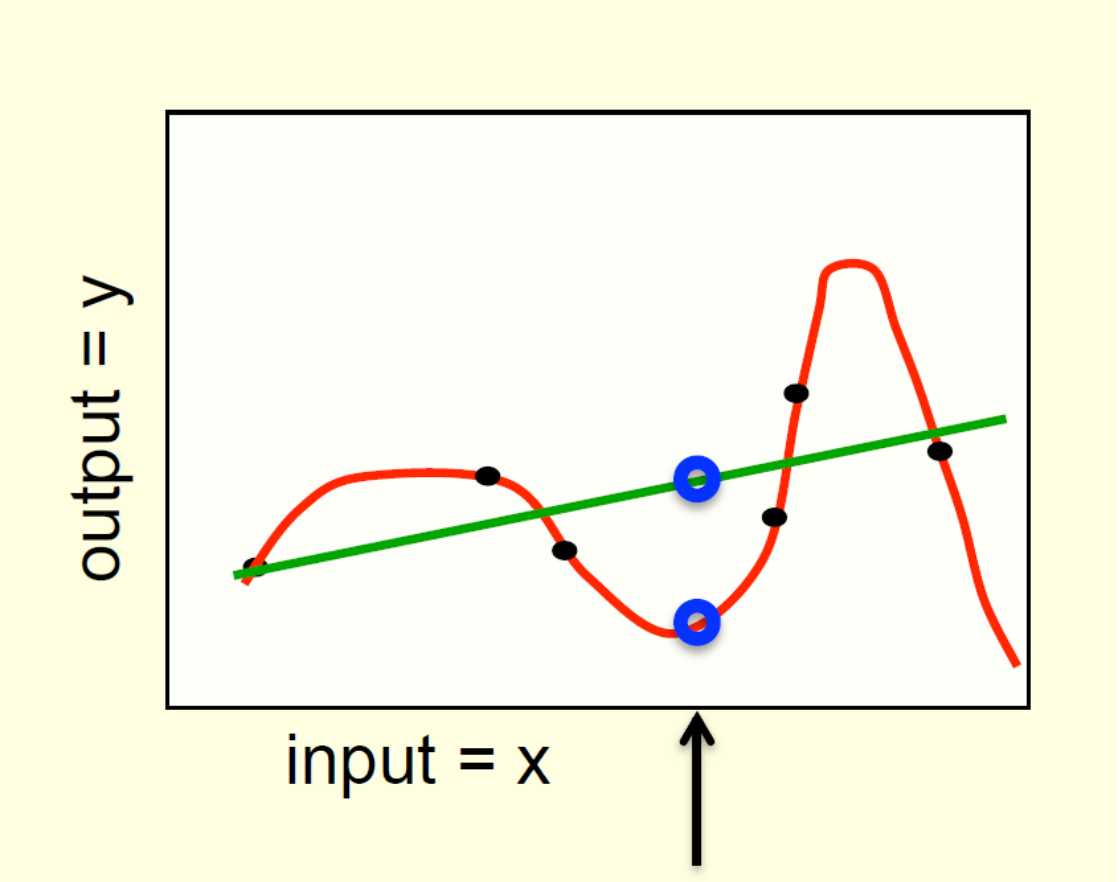
DNN in vision: convolutional nets

- CNNs are currently very successful approach in image analysis
- Idea: many copies of small detectors used all over the image, recursively combined
- Detectors are learned, combinations are learned

Weights of the same colors have equal weights
Learning with adapted backpropagation

Overfitting and model complexity

- which curve is more plausible given the data?
- overfitting
- neural nets are especially prone to overfitting
- why?



Approaches to prevent overfitting

- Weight-decay
- Weight-sharing
- Early stopping
- Model averaging
- Bayesian fitting of neural nets
- Dropout
- Generative pre-training
- etc.

Weaknesses of deep learning

Attacks on neural networks

Sources

- Richard Socher: *Deep Learning for Natural Language Processing*. Coursera
- Ian Goodfellow and Yoshua Bengio and Aaron Courville: *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org
- Yoav Goldberg: A Primer on Neural Network Models for Natural Language Processing. *Journal of Artificial Intelligence Research* 57:345-420, 2016
- Keras library
- PyTorch

More representation: Neural embeddings

- word2vec
- GloVe
- Contextual embeddings: ELMo, BERT
- cross-lingual embeddings

Word2vec

- Instead of **counting** how often each word *w* occurs near "*apricot*“
- Train a classifier on a binary **prediction** task:
  Is *w* likely to show up near "*apricot*"?
- We don’t actually care about this task
- But we'll take the learned classifier weights as the word embeddings

- Words near apricot acts as ‘correct answers’ to the question
  “Is word w likely to show up near apricot?”
- No need for hand-labeled supervision

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

Word2vec – two variants

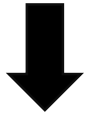
Model	Approach	Speed and Performance	Use case
Continuous Bag-of-Words model (CBOW)	The CBOW predicts the current word based on the context.	Faster to train than the skip-gram model	Predicts frequent words better
Skip-Gram model	Skip-gram predicts surrounding words given the current word.	Usually performs better than CBOW	Predicts rare words better

- Word2vec comes with two models:

Word2vec – skip-gram

- Skip-gram learning:
  - Given $w_{0,}$ predict w_{-2}, w_{-1}, w_1, and w_2

w_{-2}	w_{-1}	w_0	w_1	w_2
?	?	Network	?	?



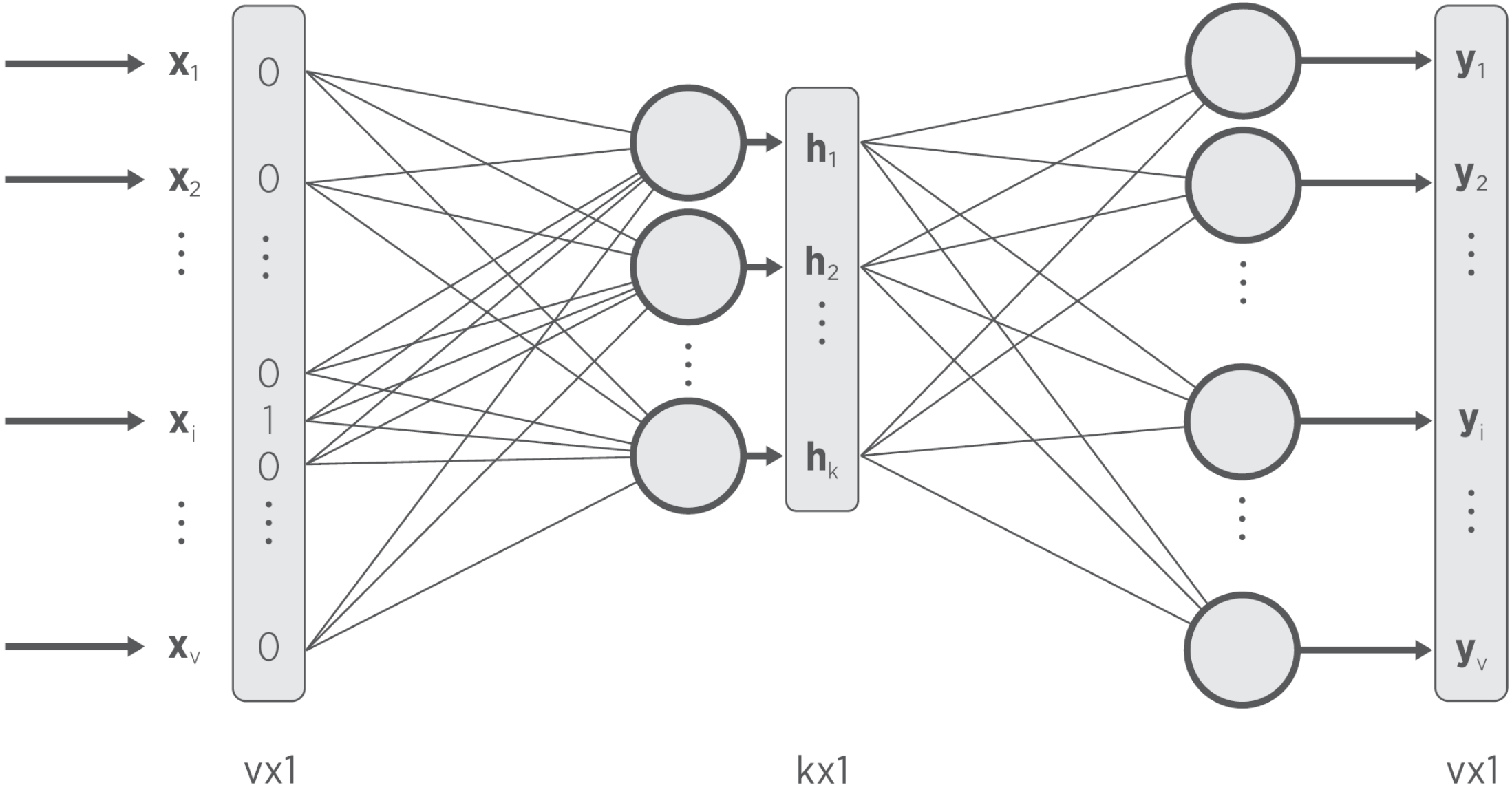
w_{-2}	w_{-1}	w_0	w_1	w_2
Recurrent	Neural		Language	Model

- Conversely, CBOW tries to predict w_0 when given w_{-2}, w_{-1}, w_1, and w_2

Skip-gram algorithm

1. Treat the target word and a neighboring context word as positive examples.
2. Randomly sample other words in the lexicon to get negative samples
3. Use logistic regression to train a classifier to distinguish those two cases
4. Use the weights as the embeddings

Word2vec neural network schema



Word2vec (skip-gram) training data

- Training sentence:
- ... lemon, a tablespoon of **apricot** jam a pinch ...
- c1 c2 target c3 c4

- Assume context words are those in +/- 2 word window
- Get negative training examples randomly
- Train a neural network to predict probability of a co-occurring word

Skip-gram goal

- Given a tuple (tic) = target, context
  - (*apricot*, *jam*)
  - (*apricot*, *aardvark*)
- Return probability that c is a real context word:

- P(+|t,c)
- $P(-|t,c) = 1-P(+|t,c)$

GloVe: Global Vectors for word representations

- It leverages the statistics of the word occurrences in the corpus and uses neural network to represent the meaning of such statistics
- Idea similar to Latent Semantic Analysis (LSA)
- Captures the global and local context of the word

$$J = \sum_{i,j=1}^{|V|} f(X_{ij}) \left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij} \right)^2, \qquad f(x) = \begin{cases} \left(\frac{x}{x_{\max}} \right)^{\alpha}, & x < x_{\max}, \\ 1, & \text{otherwise.} \end{cases}$$

Pre-trained models https://nlp.stanford.edu/projects/glove/

Pennington, J., Socher, R. and Manning, C.D., 2014, October. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on Empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

FastText: Fast Text Classifier

- Based on the word2vec **skipgram** model, only that is uses the subword information (later in the slides)
- A word is represented as a sum of character n-gram embeddings that appeared in the word

$$\sum_{t=1}^{T}\left[\sum_{c\in\mathcal{C}_t}\mathcal{L}(s(w_t,w_c))+\sum_{n\in\mathcal{N}_{t,c}}\mathcal{L}(-s(w_t,n))\right], \qquad s(w,c)=\sum_{g\in\mathcal{G}_w}\mathbf{z}_g^T\mathbf{v}_c$$

$$\mathcal{L}: x \rightarrow \log(1+e^{-x})$$

FastText compared with word2vec

- FastText outperforms skipgram in most scenarios and datasets when dealing with syntactic tasks
- For sematic tasks, the fastText is (2-5 per cent) less accurate than the skipgram model
- Is able to generate out-of-vocabulary word embeddings

Pre-trained models (157 languages, aligned vectors)
https://fasttext.cc/docs/en/english-vectors.html

Sub-word inputs for NNs

- good for morphologically rich languages
- Byte Pair Encoding
  - Most frequent byte pair↦a new byte
  - in NLP, we use character ngrams instead of bytes
- Start with a unigram vocabulary of all (Unicode) characters in data
- Most frequent ngram pairs ↦a new ngram
- Have a target vocabulary size and stop when you reach it
- Do deterministic longest piece segmentation of words
- Segmentation is only within words identified by some prior tokenizer
- Automatically decides vocabulary for system
- No longer strongly “word” based in conventional way

Comparison of standard word embeddings

Model	Advantages	Disadvantages
Continuous BOW	• Mediocre semantic accuracy • Absence in papers; unpopular in practice	• Ignores global vocabulary information • Does not handle out-of-vocabulary words
Skipgram	• Good semantic accuracy • Pre-trained models available	
GloVe	• Uses global information of vocabulary • Captures local and global context of words • Good syntactic and semantic accuracy • Pre-trained models available	• Does not handle out-of-vocabulary words
FastText	• Handles out-of-vocabulary words • Good at syntactic tasks • Pre-trained models available • Available aligned word vectors	• Takes 1.5x longer to train than *skipgram*

Evaluation of embeddings

- Related to general evaluation in NLP: intrinsic vs. extrinsic
- Intrinsic:
  - Evaluation on a specific/intermediate subtask
  - Fast to compute
  - Helps to understand that system
  - Not clear if really helpful unless correlation to real task is established
- Extrinsic:
  - Evaluation on a real task
  - Can take a long time to compute accuracy
  - Unclear if the subsystem is the problem or its interaction or other subsystems
  - If replacing exactly one subsystem with another improves accuracy then we are doing well

Embeddings capture relational meaning

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

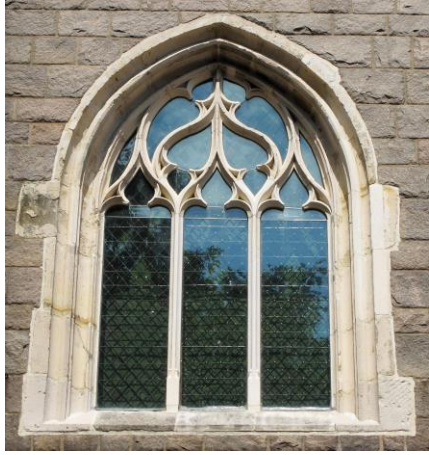
vector(*'Paris'*) - vector(*'France'*) + vector(*'Italy'*) ≈ vector('Rome')

Embeddings reflect cultural biases

- Ask “Paris : France :: Tokyo : x”
  - x = Japan
- Ask “father : doctor :: mother : x”
  - x = nurse
- Ask “man : computer programmer :: woman : x”
  - x = homemaker

Bolukbasi, Tolga, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." In *Advances in Neural Information Processing Systems*, pp. 4349-4357. 2016.

Embeddings as a window onto history



- Use the historical embeddings
- The cosine similarity of embeddings for decade X for occupations (like teacher) to male vs female names
  - Is correlated with the actual percentage of women teachers in decade X

Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, *115*(16), E3635–E3644

Embeddings reflect ethnic stereotypes over time

- Attitudes toward ethnic groups (1933, 1951, 1969) scores for adjectives
  - *industrious, superstitious, nationalistic*, etc.
- Cosine of Chinese name embeddings with those adjective embeddings correlates with human ratings.

Recurrent network (RNN)

- back connections
- biologically more realistic
- store information from the past – back connections correspond to memory
- more difficult to learn

Contextual embeddings

- **Simple embeddings**, like word2vec, produce the **same vector** for a word like “cold” irrespective of its meaning and context
- Recent embeddings take the **context** into account
- Already established as a standard:
  ELMo and BERT

Language models

- DNNs shall contain knowledge about language
- Knowledge is in the form of language models: $P(w_n|w_1w_2…w_{n-1})$
- Neural network is trained to predict the context of words

ELMo

- ELMo looks at the entire sentence before assigning each word in it an embedding.
- ELMo predicts the next word in a sequence of words - a task called *Language Modeling*.
- It uses a bi-directional LSTM recurrent neural network
- Includes subword units
- As an embedding ELMO uses several layers of the network
- First layers capture morphological and syntactic properties, deeper layers encode semantical properties
- Uses several fine tuned parameters
- Publicly available; for many languages

Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L., 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pp. 2227-2237.

Ulčar, M. and Robnik-Šikonja, M., 2020, May. High Quality ELMo Embeddings for Seven Less-Resourced Languages. In *Proceedings of The 12th Language Resources and Evaluation Conference* (pp. 4731-4738).

Sub-word inputs for NNs

- good for morphologically rich languages
- Byte Pair Encoding
  - Most frequent byte pair↦a new byte
  - in NLP, we use character ngrams instead of bytes
- Start with a unigram vocabulary of all (Unicode) characters in data
- Most frequent ngram pairs ↦a new ngram
- Have a target vocabulary size and stop when you reach it
- Do deterministic longest piece segmentation of words
- Segmentation is only within words identified by some prior tokenizer
- Automatically decides vocabulary for system
- No longer strongly “word” based in conventional way

Word/sentence piece encoding

- Google NMT uses a variant of Byte Pair Encoding
- wordpiece model
- sentencepiece model
- Rather than char *n*-gram count, uses a greedy approximation to maximizing language model log likelihood to choose the pieces
- Add *n*-gram that maximally reduces perplexity

- Wordpiece model tokenizes inside words
- Sentencepiece model works from raw text
- Whitespace is retained as special token (_) and grouped normally
- You can reverse things at end by joining pieces and recoding them to spaces

ELMo embeddings

- Embeddings from Language Models
- Learn word token vectors using long contexts (whole sentence, could be longer)
- Learn a deep Bi-NLM and use all its layers in prediction

Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L., 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT* (pp. 2227-2237).

ELMo embeddings details

- Train a bidirectional LM
- Aim at performant but not overly large LM:
- Use 2 biLSTM layers
- Use character CNN to build initial word representation (only)
  - 2048 char n-gram filters and 2 highway layers, 512 dim projection
- User 4096 dim hidden/cell LSTM states with 512 dim projections to next input
- Use a residual connection
- Tie parameters of token input and output (softmax) and tie these between forward and backward LMs

ELMo weights

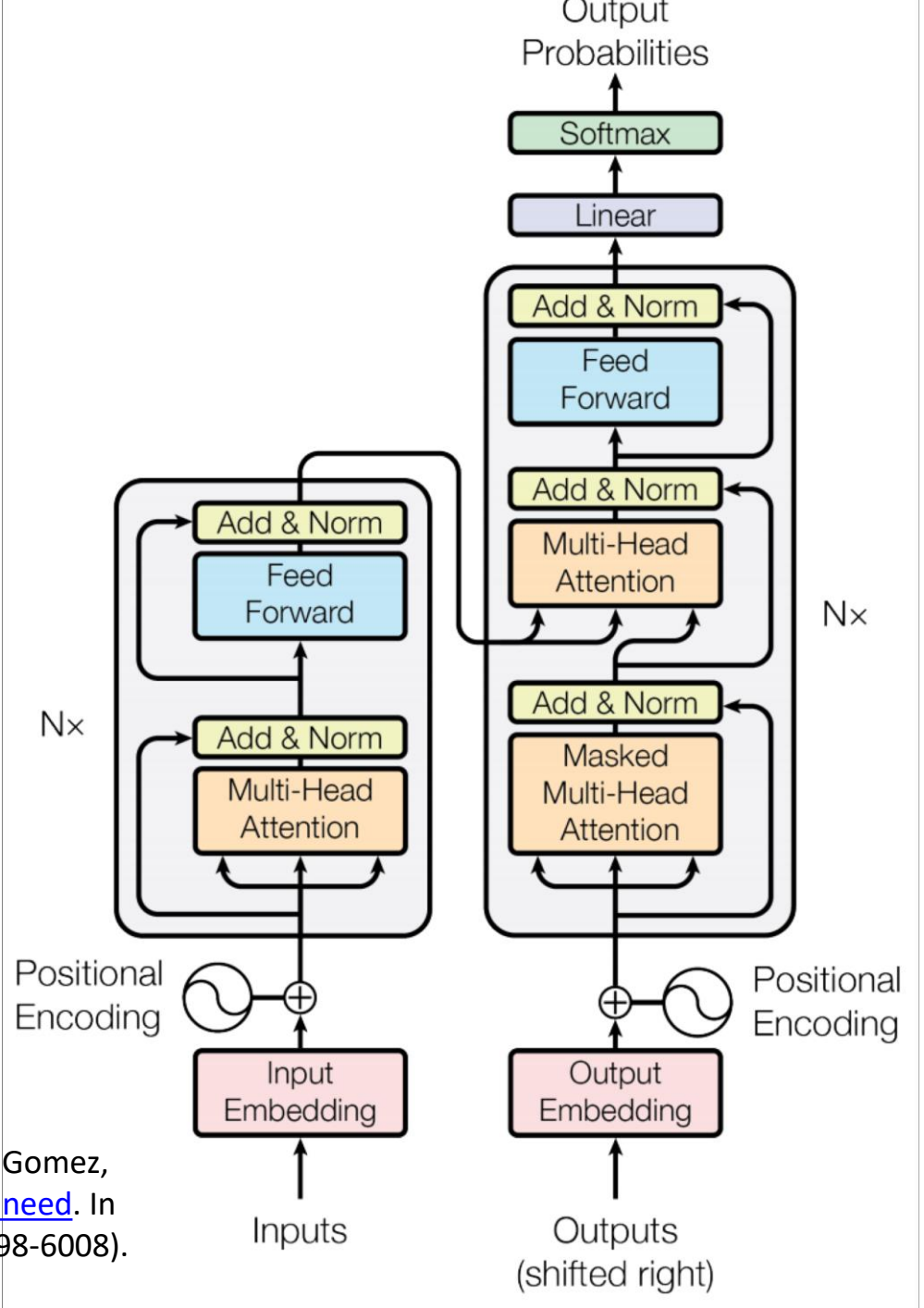
- The two biLSTM NLM layers have differentiated uses/meanings
- Lower layer is better for lower-level syntax: Part-of-speech tagging, syntactic dependencies, NER
- Higher layer is better for higher-level semantics: Sentiment, Semantic role labeling, question answering, SNLI

Transformer model

- currently the most successful DNN
- non recurrent
- architecturally it is encoder-decoder model
- fixed input length
- can be parallelized
- adapted for GPU (TPU) processing
- based on extreme use of attention

Transformer overview

- Initial task: machine translation with parallel corpus
- Predict each translated word
- Final cost/loss/error function is
- standard cross-entropy erroron top of a softmax classifier



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

BERT

- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
- state-of-the-art pretained LM based on transformer architecture

- some slides from Jacob Devlin and Jay Alammar

Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Volume 1, pp. 4171-4186.

BERT: motivation 1/3

- **Problem**: Language models only use left or right context, but language understanding is bidirectional.
- Why are LMs unidirectional?
  - Reason 1: Directionality is needed to generate a well-formed probability distribution.
    - We don't care about this.
  - Reason 2: Words can "see themselves" in a bidirectional encoder.

BERT: motivation 3/3

- **Solution**: Mask out *k*% of the input words, and then predict the masked words
- BERT uses *k* = 15%

store gallon

↑ ↑

the man went to the [MASK] to buy a [MASK] of milk

- Too little masking: Too expensive to train (not enough masks)
- Too much masking: Not enough context

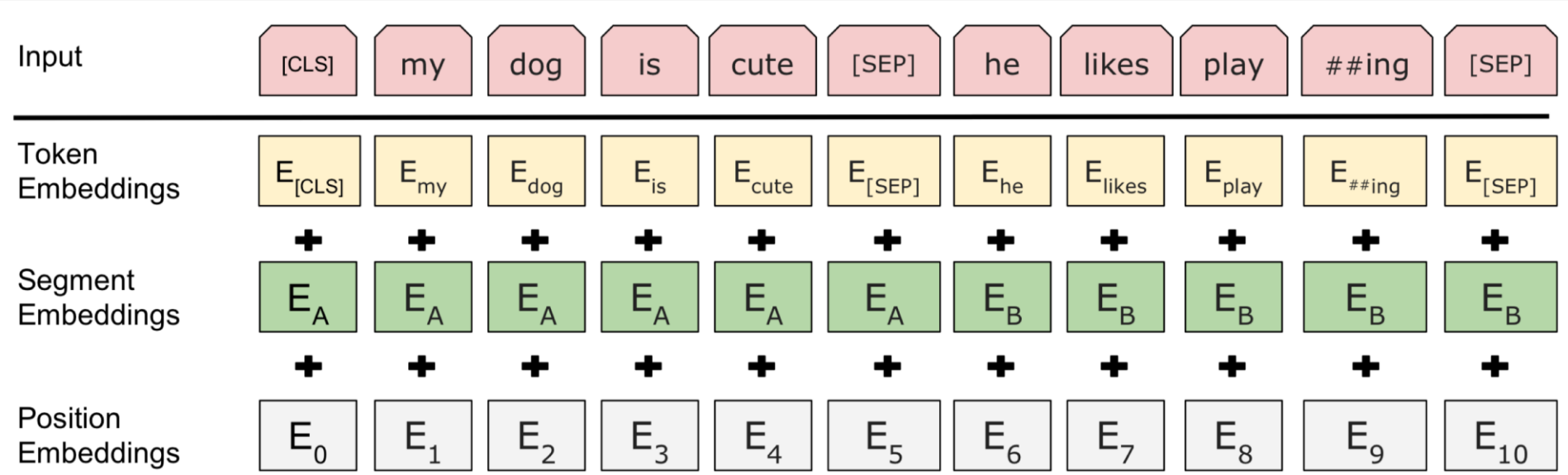
BERT uses several tasks

- besides masked LM, BERT learns relationships between sentences,
- predict whether Sentence B is actual sentence that proceeds Sentence A, or a random sentence

- some follow-up BERT-like models, e.g., RoBERTa, drop this task and claim better performance on downstream tasks

Sentence-pair encoding for BERT

- Token embeddings are word pieces (sub-word encoding)
- (Relatively) common words are in the vocabulary: *at, fairfax, 1910s*
- Other words are built from wordpieces: *hypatia = h ##yp ##ati ##a*
- Learned segmented embedding represents each sentence
- Positional embedding is the same as for other transformer architectures



BERT training

- Transformer encoder
- Self-attention ⇒ no locality bias
- Long-distance context has “equal opportunity”
- Single multiplication per layer ⇒ efficiency on GPU/TPU
- Trained on Wikipedia + BookCorpus
- English BERT was trained on 2 model sizes:
  - BERT-Base: 12-layer, 768-hidden neurons, 12-head, 110M parameters
  - BERT-Large: 24-layer, 1024-hidden neurons, 16-head, 340M parameters
- Trained on 4x4 or 8x8 TPU slice for 4 days

Examples of GLUE tasks

- GLUE benchmark is dominated by natural language inference tasks, but also has sentence similarity and sentiment

MultiNLI

Premise: Hills and mountains are especially sanctified in Jainism.

Hypothesis: Jainism hates nature.

Label: Contradiction

CoLA (Corpus of Linguistic Acceptability)

Sentence: The wagon rumbled down the road. Label: Acceptable

Sentence: The car honked down the road. Label: Unacceptable

Multilingual BERT

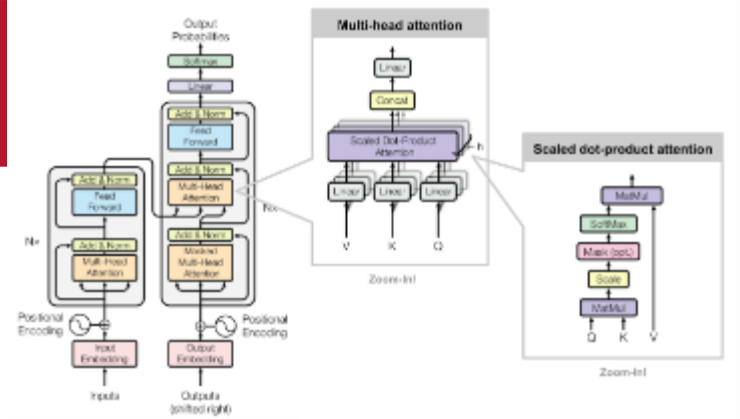
- BERT base architecture was trained on 104 languages with largest Wikipedia (at least 1 million words at the time)
- achieves good results for many tasks
- good cross-lingual transfer

BERT can produce embeddings

- one can extract fixed size contextual vectors from BERT, achieving slightly lower accuracy than using the whole BERT as the first stage model

University of Ljubljana
Faculty of Computer and Information Science

The new king: BERT



- combines several tasks
- predicts masked words in a sentence
- also predicts order of sentences: is sentence A followed by sentence B or not
- as embeddings combines several hidden layers of the network
- uses transformer neural architecture, e.g., 14 layers
- basic variant 110 million weights
- uses several fine tuned hyperparameters
- multilingual variant supports 104 languages by training on Wikipedia
- publicly available

Devlin, J., Chang, M.W., Lee, K. and Tout nova, K., 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *ArXiv preprint arXiv:1810.04805*.

University of Ljubljana
Faculty of Computer and Information Science

Existing embeddings

- recent XLM-R was trained on 2.5 TB of texts in 100 languages
- for Slovene: fastText, ELMo,
- trilingual BERT – CroSloEngual
- on Clarin.si
- more to follow: hundreds of papers investigating BERT-like models in major ML & NLP conferences

Ulčar, Matej and Marko Robnik-Šikonja. FinEst BERT and CroSloEngual BERT: less is more in multilingual models. In Proceedings of Text, Speech, and Dialogue, TSD2020 (accepted), 2020.

Cross-lingual embeddings

- embeddings are trained on monolingual resources
- words of one language form a cloud in high dimensional space
- clouds for different languages can be aligned

$$W_1 S \approx E$$

Cross-lingual embeddings

- Aligning embedding spaces across languages

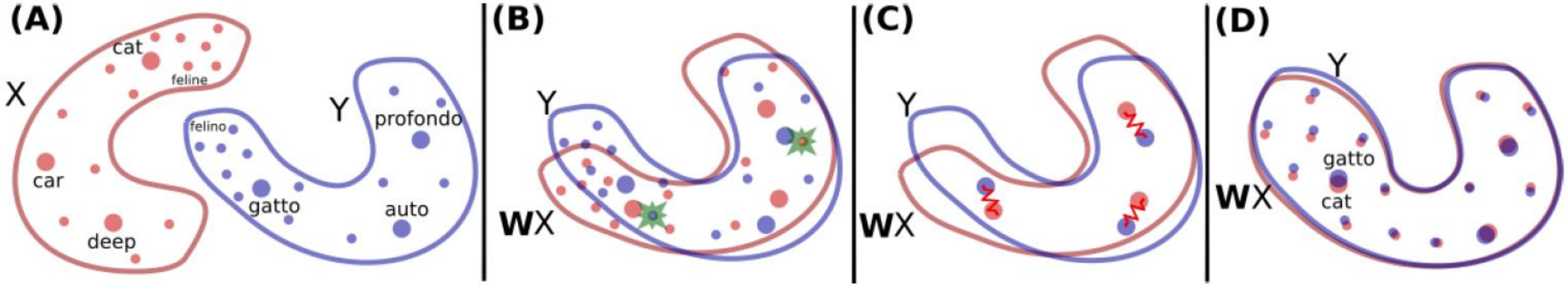
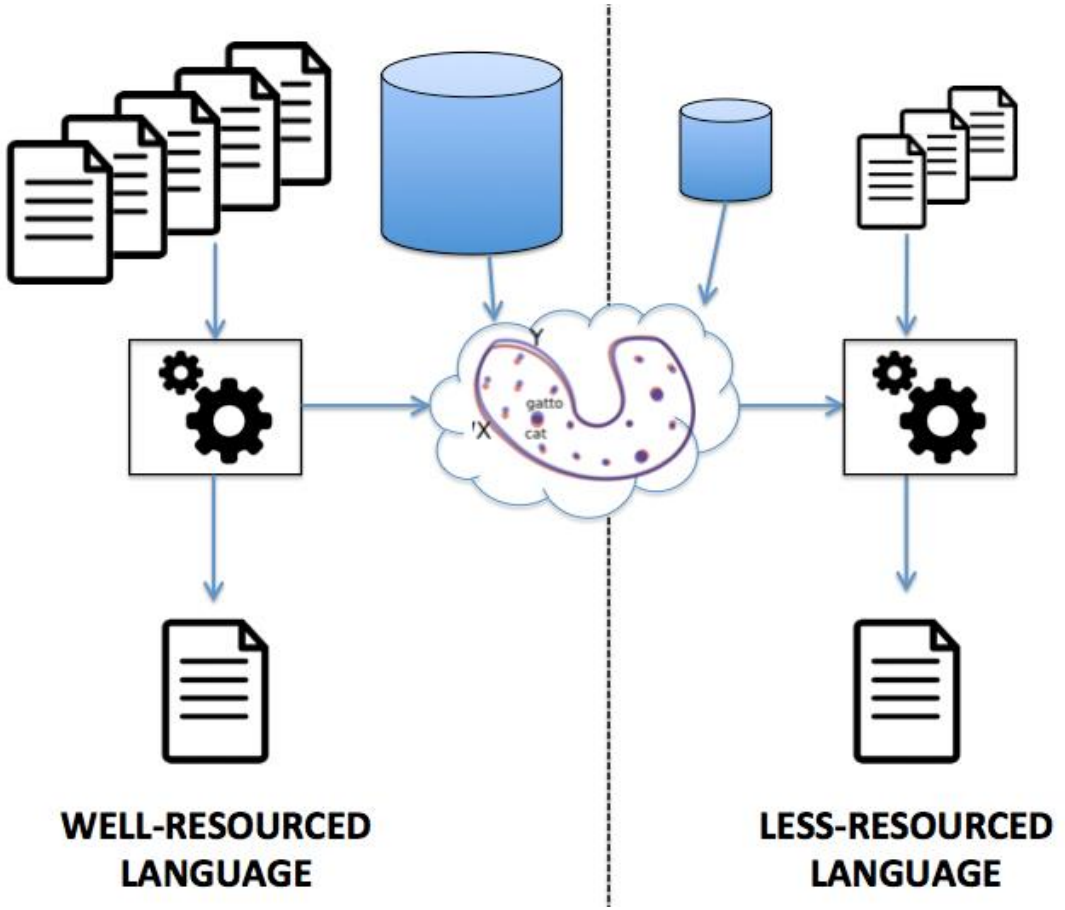


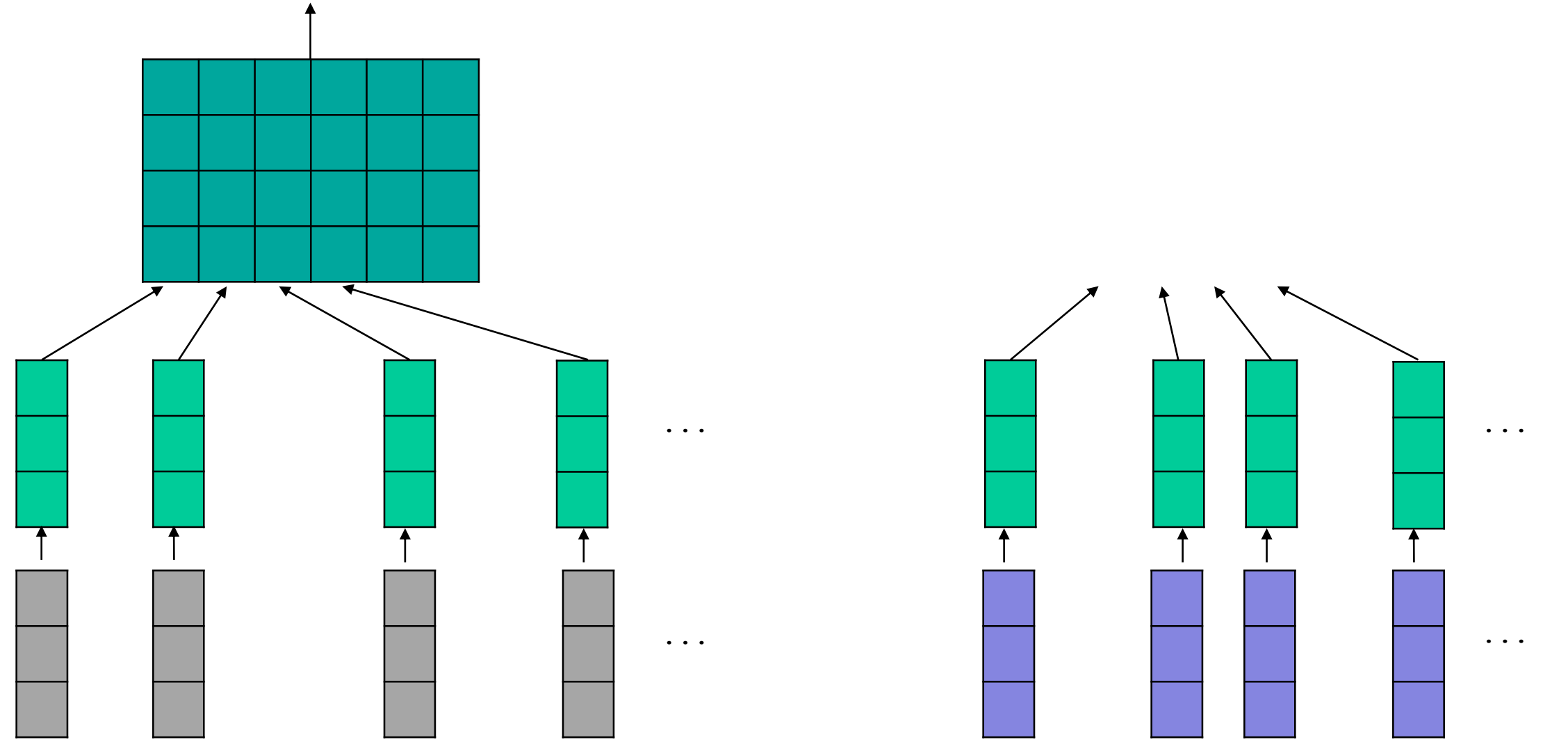
Image credit to Conneau, Lample, Ranzato, Denoyer, Jegou: Word translation without parallel data. ICLR 2018.

- Supervised, semi-supervised, and unsupervised approaches

Cross-lingual model transfer based on embeddings

- Transfer of tools trained on mono-lingual resources




the president announced today ...
predsednik je danas najavio ...

EU H2020 project

- EMBEDDIA: Cross-Lingual Embeddings for Less-Represented Languages in European News Media

- develops textual embeddings and in particular
  - cross-lingual embeddings
  - for less-resourced languages
  - for media industry

- http://www.embeddia.eu
- @embeddiaproject

This project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 825153

University *of Ljubljana*
Faculty *of Computer and Information Science*

Conclusion of dense text embeddings

- **Concepts** or word senses
  - Have a complex many-to-many association with **words** (homonymy, multiple senses)
  - Have relations with each other
    - Synonymy, Antonymy, Superordinate
  - But are hard to define formally (necessary & sufficient conditions)
- **Embeddings** = vector models of meaning
  - More fine-grained than just a string or index
  - Especially good at modeling similarity/analogy
    - Just download them and use cosines
  - Useful in practice but know they encode cultural stereotypes

Summary: Embed all the things!

- Neural networks require numeric input
- Embedding shall preserve relations from the original space
- Representation learning seems to be crucial topic in nowadays machine learning
- Lots of applications whenever enough data is available to learn the representation
- In text BERT-like models rule
- Similar ideas applied to texts, speech, graphs, electronic health records, relational data, time series, etc.