Text Mining and Sentiment Analysis

Prof. Annamaria Bianchi
A.Y. 2024/2025

Lecture 18

12 May 2025

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Scienze Economiche

Outline

Other topic models

Where is the field of NLP going?

Correlated Topic Modeling

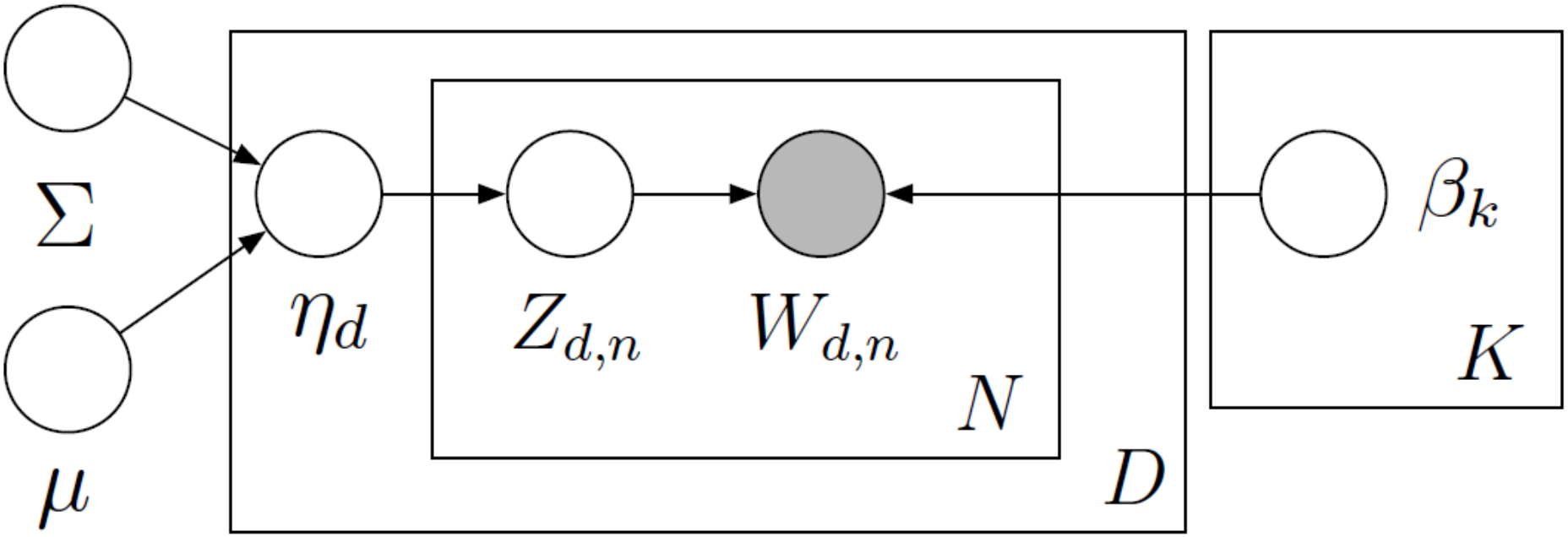
An LDA model sets the basic principles for topic modeling and CTM is an extension of this, building upon the LDA model.

Limitation of LDA: it fails to directly model correlation between the occurrence of topics. In most text corpora, it is natural to expect that the occurrences of the underlying topics will be highly correlated.
In LDA this limitation stems from the independence assumptions implicit in the Dirichlet distribution of the topic proportions.

In CTM, we model the topic proportions with an alternative, more flexible distribution that allows for covariance structure among components. This gives a more realistic model of latent topic structure where the presence of one latent topic may be correlated with the presence of another.

CTM is based on the logistic normal distribution, which allows for a general pattern of variability between the components.

Correlated Topic Modeling



Correlated Topic Modeling

The CTM is more expressive than LDA because the strong independence assumption imposed by the Dirichlet in LDA is not realistic when analyzing real document collections.

Quantitative results illustrate that the CTM better fits data than LDA.

The added flexibility of the CTM comes at computational cost. Implementation of CTM is not as fast and strightforward as LDA.

Correlated Topic Modeling

To perform correlated topic modeling we use the `CTM()` function from the **topicmodels** package. This produces CTM objects. Next, we tidy such models so that they can be manipulated with tidy tools.

The function `CTM()` estimates a CTM model.

```
CTM(x, k, method = "VEM", control = NULL,...)
```

`x`	Object of class "DocumentTermMatrix" with term-frequency weighting.
`k`	Integer; number of topics.
`method`	The method to be used for fitting; currently method = "VEM" or method= "Gibbs" are supported.
`control`	A named list of the control parameters for estimation.

Biterm Topic Modeling (BTM)

Directly applying conventional topic models (e.g. LDA or CTM) for uncovering topics within short texts, such as tweets and messages, may not work well.

The fundamental reason lies in that conventional topic models implicitly capture the document-level word co-occurrence patterns to reveal topics, and thus suffer from the severe data sparsity in short documents.

Biterm topic model (BTM) learns the topics by directly modeling the generation of word co-occurrence patterns (i.e. biterms) in the whole corpus.

Major advantages: 1) BTM explicitly models the word co-occurrence patterns to enhance the topic learning;

2) BTM uses the aggregated patterns in the whole corpus for learning topics to solve the problem of sparse word co-occurrence patterns at document-level.

The Structural Topic Modeling

- **Key innovation**: allows to incorporate document metadata in the topic model. Metadata are defined as information about each document.

- **Goal:** of the structural topic model is to allow researchers to discover topics and estimate their relationship to document metadata. Outputs of the model can be used to conduct hypothesis testing about these relationships.

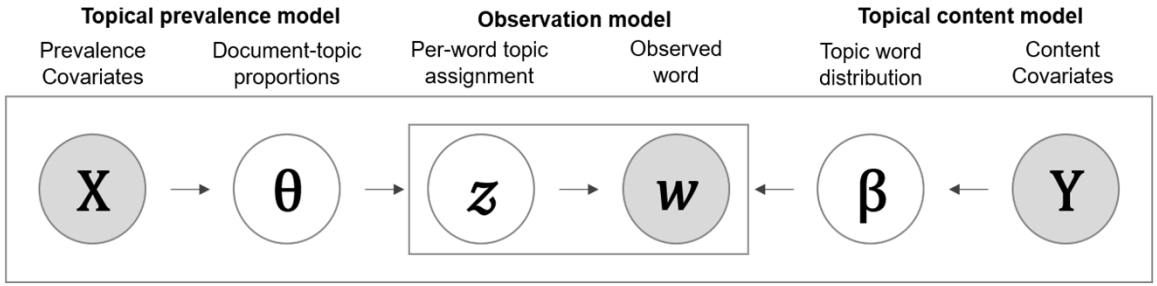
- **Examples**:

    - Analysis of open-ended survey questions to test the differences between males and females

    - Analysis of newspaper to compare how news are reported

    - Analysis of businesses' tweets to study differences across sectors

The Structural Topic Modeling



Source: Amended from Roberts et al. (2016).

- **Like LDA it is generative model** (Roberts, Stewart & Airoldi 2016): That means we define a data generating process for each document and then use the data to find the most likely values for the parameters within the model.
- **Like LDA is based on the bag of words representation**: we do not consider grammar and syntax features.

The Structural Topic Modeling

Elements:

- Set of D documents indexed by $d \in \{1 \dots D\}$
- Each document is composed by a mixture of words $w_{d,n}$, where $n \in \{1 \dots N_d\}$ indicates the position
- Θ is the proportion of a document about a specific topic
- β is the probability of a word being generated by a specific topic
- **X topical prevalence** covariates: affect the proportion of the document that is associated to a topic
- **Y topical content** covariates: affect the usage rate of word in a topic

Source: Amended from Roberts et al. (2016).

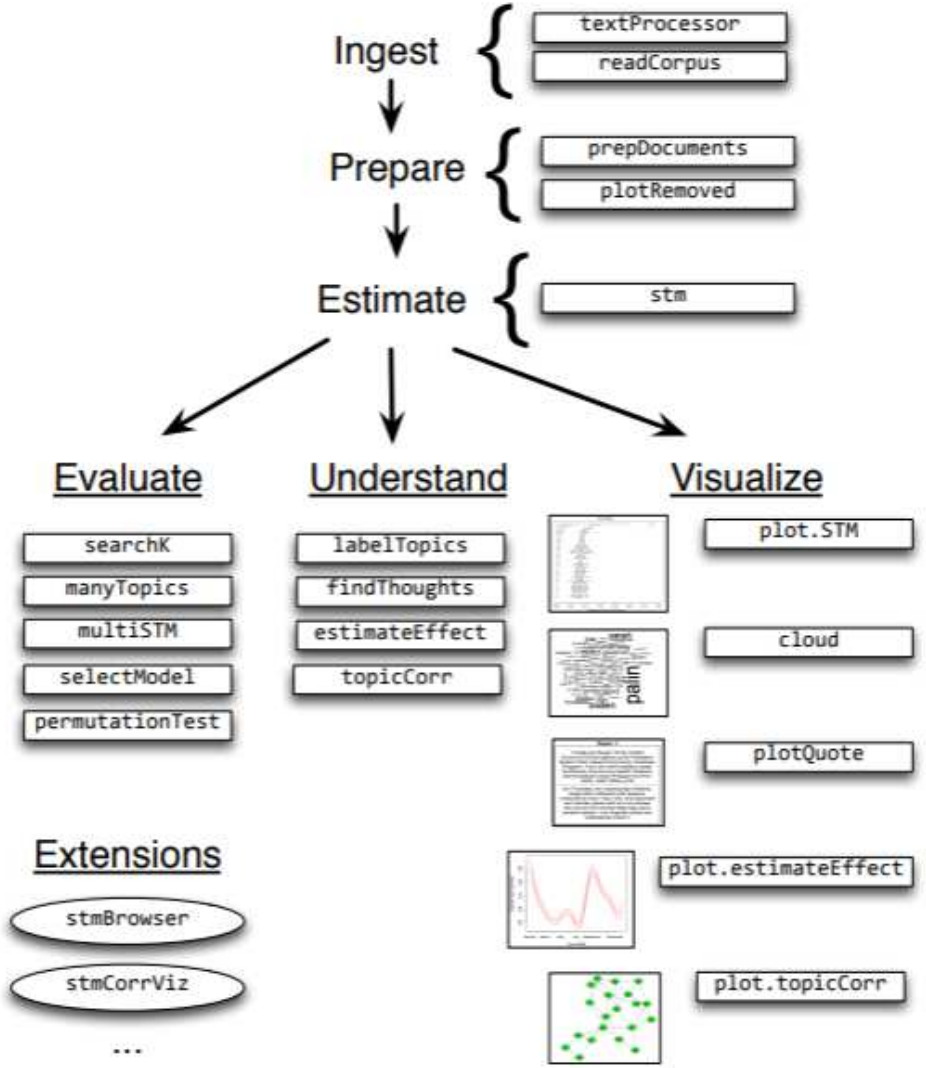
The Structural Topic Modeling

- **Model estimation and inference**: variational expectation-maximization algorithm
- **Model convergence**: the relative change in the approximate variational lower bound is below a defined tolerance level
- **Model Initialization**: spectral initialization, a deterministic algorithm based on the method of moments, is suggested due to its stability

The STM

The stm package allows to:

- Estimate
- Evaluate
- Understand & Visualize the results



Vector semantics

- How we build a theory of how to represent word meaning? → We'll introduce **vector semantics**
  - Suppose you see these sentences:
    - Ong choi is delicious **sautéed with garlic**.
    - Ong choi is superb **over rice**
    - Ong choi **leaves** with salty sauces
- And you've also seen these:
    - …spinach **sautéed with garlic over rice**
    - Chard stems and **leaves** are **delicious**
    - Collard greens and other **salty** leafy greens
- Conclusion:
  - Ongchoi is a leafy green like spinach, chard, or collard greens
    - We could conclude this based on words like "leaves" and "delicious" and "sauteed"

Vector semantics

- Idea 1: Let's define the meaning of a word by its distribution in language use, meaning its neighboring words or grammatical environments.

Vector semantics

- Idea 2: Meaning as a point in space (Osgood et al. 1957)
- 3 affective dimensions for a word
  - **valence**: pleasantness
  - **arousal**: intensity of emotion
  - **dominance**: the degree of control exerted

	Valence	Arousal	Dominance
courageous	8.05	5.5	7.38
music	7.67	5.57	6.5
heartbreak	2.45	5.65	3.58
cub	6.71	3.95	4.24

NRC VAD Lexicon
(Mohammad 2018)

- Hence the connotation of a word is a vector in 3-space

Vector semantics

- Each word = a vector (not just "good" or "w_{45}")
- Similar words are "**nearby in semantic space**"
- We build this space automatically by seeing which words are **nearby in text**

Vector semantics

Called an "embedding" because it's embedded into a space

The standard way to represent meaning in NLP

Every modern NLP algorithm uses embeddings as the representation of word meaning

Fine-grained model of meaning for similarity

Intuition: why vectors?

- Consider sentiment analysis:
  - With **words**, a feature is a word identity
    - Feature 5: 'The previous word was "terrible"'
    - requires **exact same word** to be in training and test
  - With **embeddings**:
    - Feature is a word vector
    - 'The previous word was vector [35,22,17…]
    - Now in the test set we might see a similar vector [34,21,14]
    - We can generalize to **similar but unseen** words!!!

Model for embedding

Word2vec

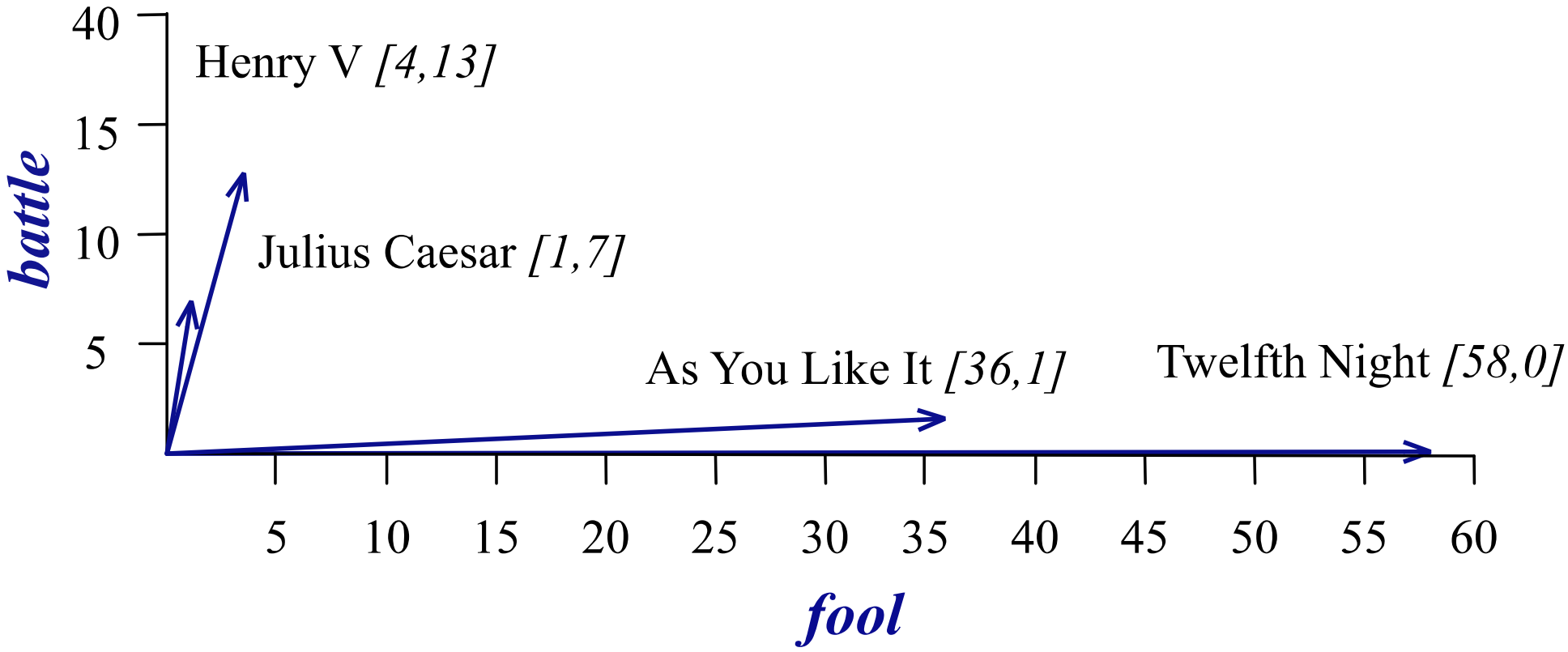
- **Dense** vectors
- Representation is created by training a classifier to **predict** whether a word is likely to appear nearby
- Extensions called **contextual embeddings**

Model for embedding

Each document is represented by a vector of words

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Visualizing document vectors



Visualizing document vectors

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Vectors are similar for the two comedies

But comedies are different than the other two

Comedies have more *fools* and *wit* and fewer *battles*.

Words can be vector too

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

battle is "the kind of word that occurs in Julius Caesar and Henry V"

fool is "the kind of word that occurs in comedies, especially Twelfth Night"

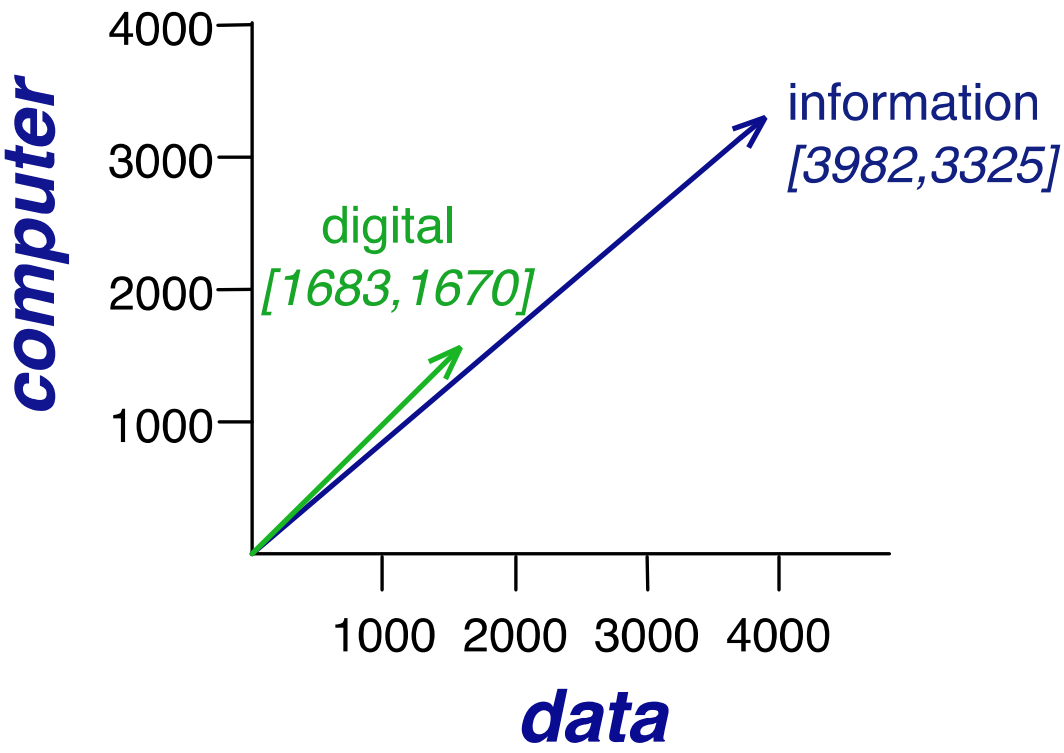
Word-word matrix

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

is traditionally followed by	**cherry**	pie, a traditional dessert
often mixed, such as	**strawberry**	rhubarb pie. Apple pie
computer peripherals and personal	**digital**	assistants. These devices usually
a computer. This includes	**information**	available on the internet

	aardvark	...	**computer**	**data**	**result**	**pie**	**sugar**	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Word-word matrix



Embeddings

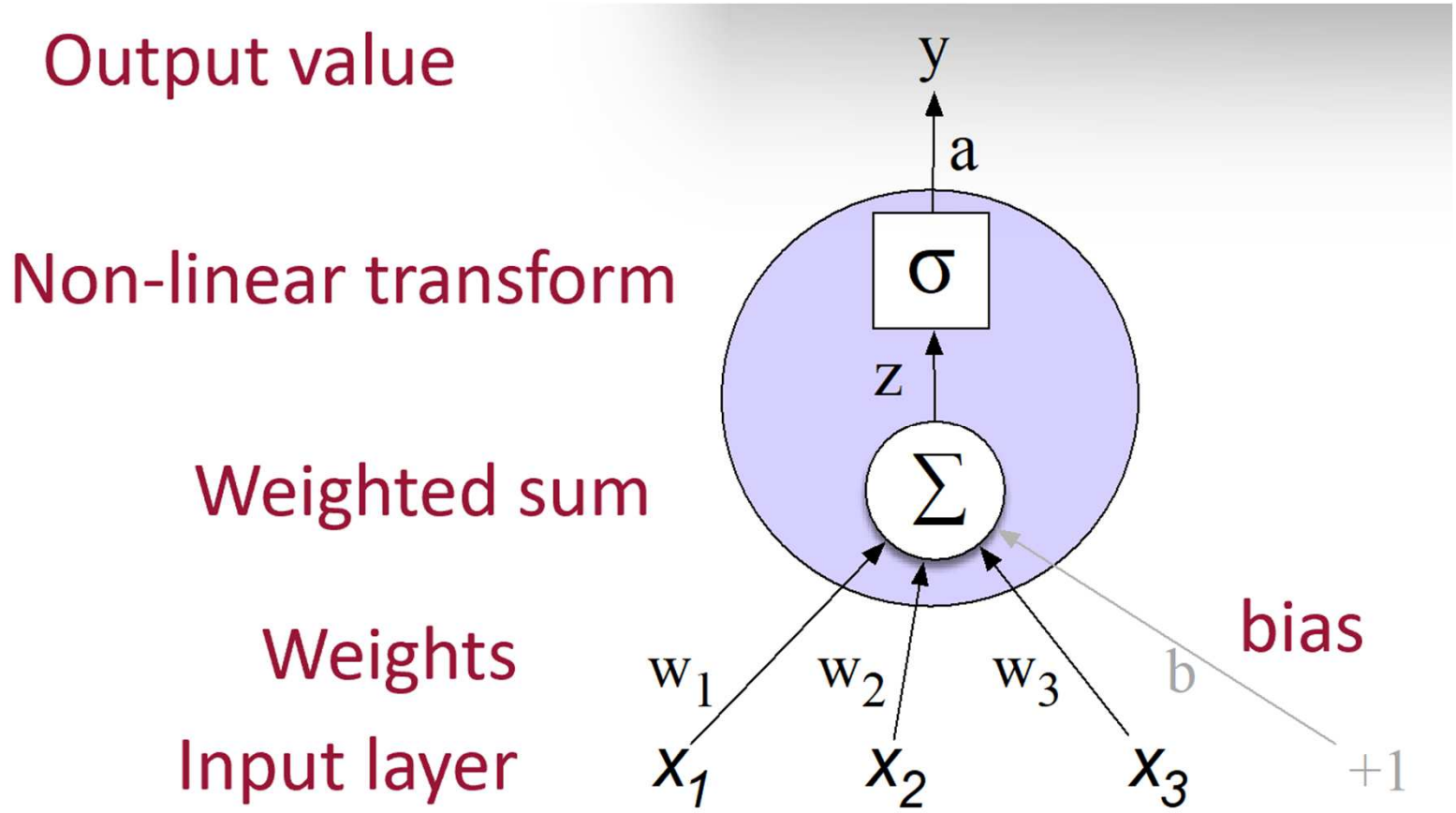
- So far words as vectors
  - **long** (length |V|= 20,000 to 50,000)
  - **sparse** (most elements are zero)
- Alternative: learn vectors which are
  - **short** (length 50-1000)
  - **dense** (most elements are non-zero)

Embeddings

Why dense vectors?

- Short vectors may be easier to use as **features** in machine learning (fewer weights to tune)
- Dense vectors may **generalize** better than explicit counts
- Dense vectors may do better at capturing synonymy
- **In practice, they work better**

Neural networks

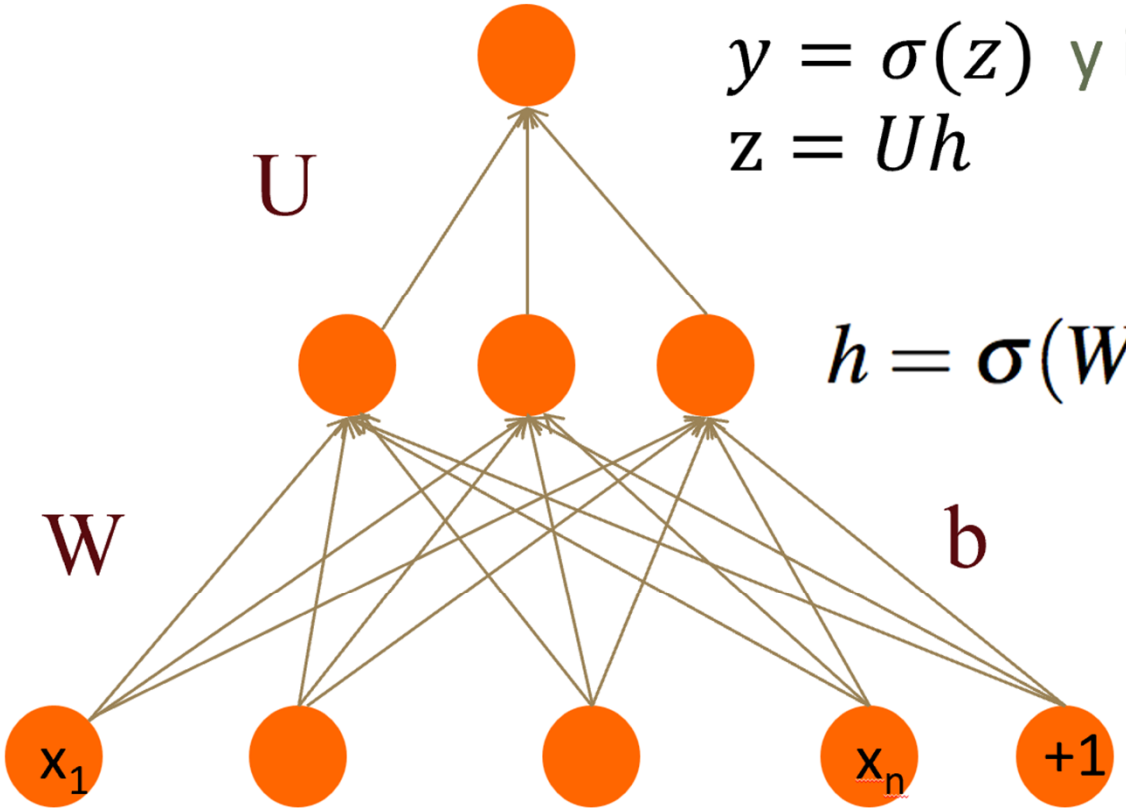


Two-layer network with scalar output

Output layer
(σ node)

hidden units
(σ node)

Input layer
(vector)



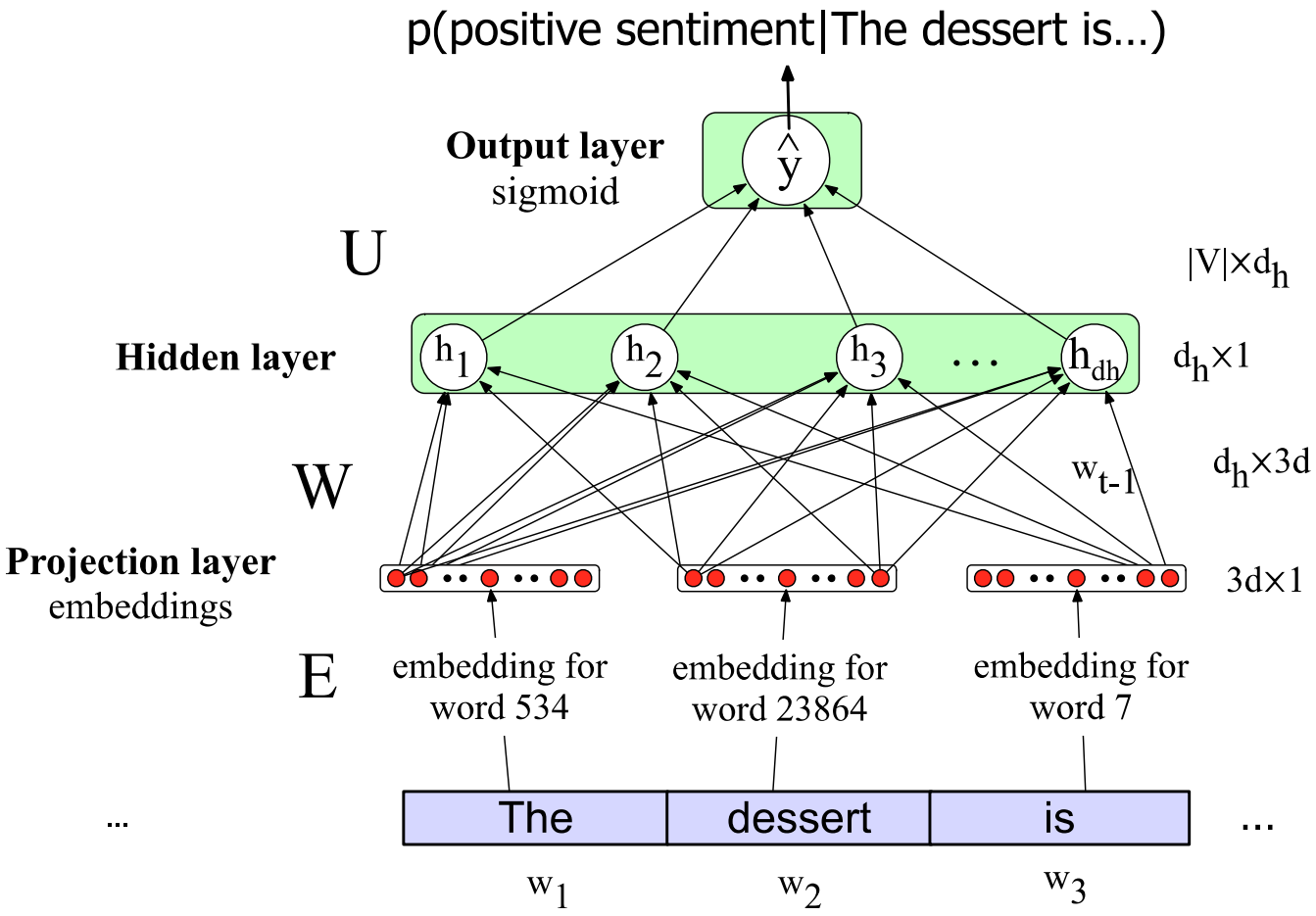
$y = \sigma(z)$ y is a scalar

$z = Uh$

$h = \sigma(Wx + b)$

Could be ReLU
Or tanh

Applying feedforward networks to NLP tasks



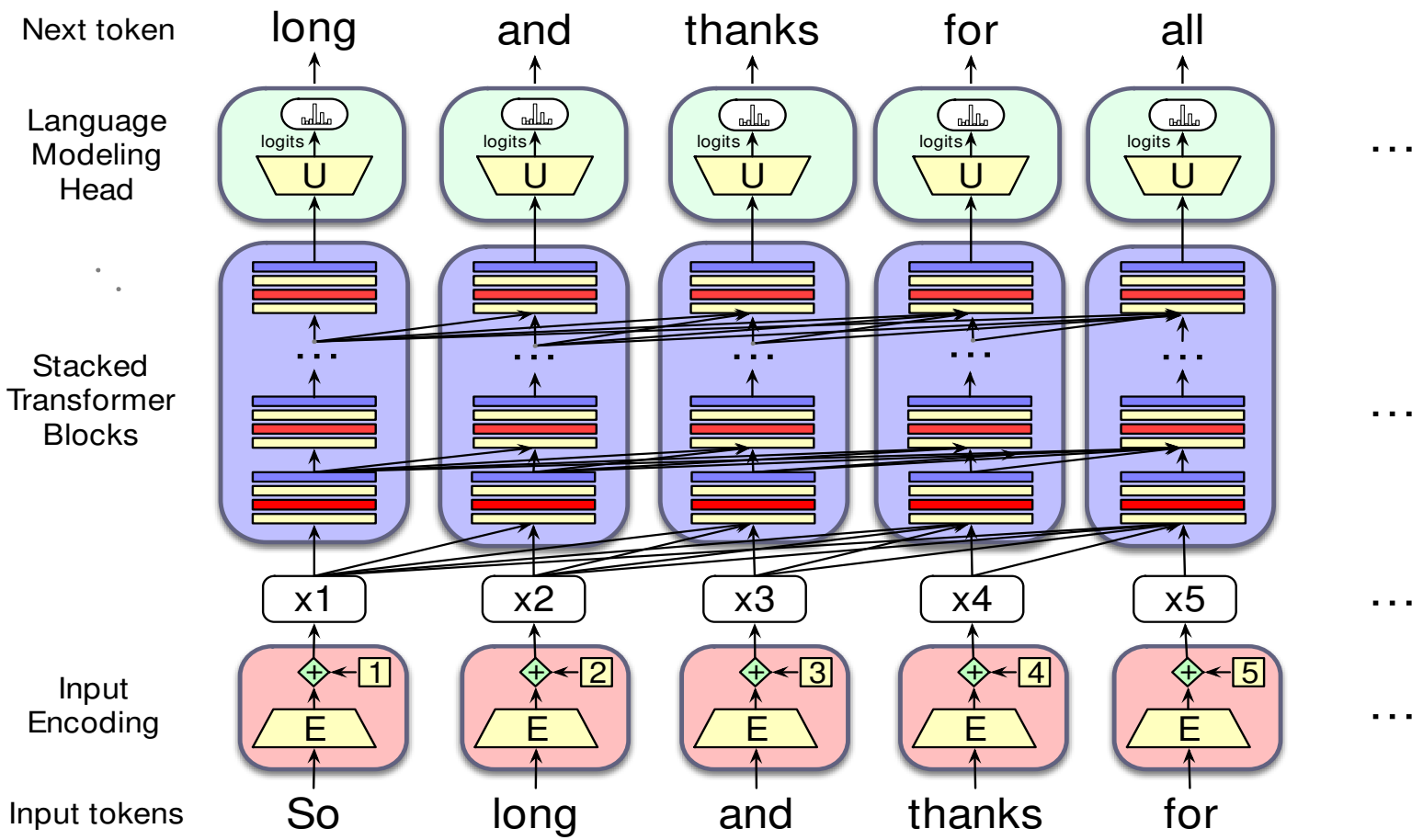
LLMS are built of transformers

Transformer: a network architecture with specific structure that includes a mechanism called attention.

Attention can be thought of as a way to build contextual representations of a token'n meaning by attending to and integrating information from surrounding tokens.

The transformer is the standard architecture for building **large language models**.

LLMS are built of transformers



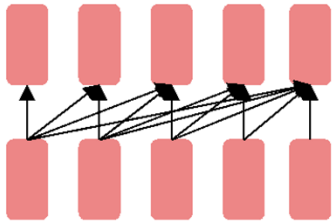
Language models

Large language models:

- Assigns probabilities to sequences of words
- Generate text by sampling possible next words
- **Are trained by learning to guess the next word**

- Even through pretrained only to predict words
- Learn a lot of useful language knowledge
- Since training on a **lot** of text

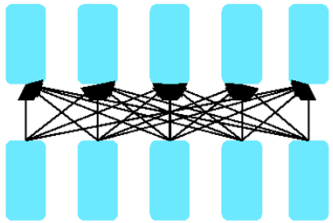
Three architectures for LLMs



Decoders

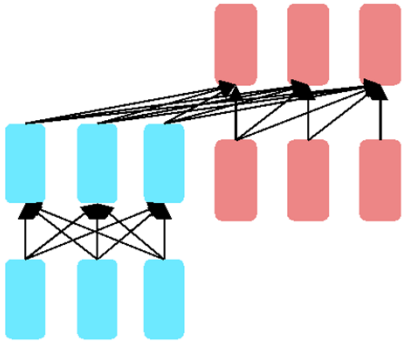
GPT, Claude, Llama

Mixtral



Encoders

BERT family, HuBERT



Encoder-decoders

Flan-T5, Whisper

Encoders

Many varieties!

- Popular: Masked Language Models (MLMs)
- BERT family

- Trained by predicting words from surrounding words on both sides
- Are usually **finetuned** (trained on supervised data) for classification tasks.

LLMs and LLM-based NLP: advantages and issues

- (+) Huge pre-training improves overall performance on many tasks
- (+) Capable of solving many tasks with the same or just slightly modified model (zero/few-shot)
- (+) capable of resolving common ambiguities, e.g. Winograd Schema Challenge (Kocijan2022)
- (-) "Hallucinations": competent-sounding nonsense. Variation of often-seen (catastrophic) failure modes with ML, esp. DNNs
- (-) Hard to control bias, performance and bias depends on selection of TB of data
- (-) As with most ML-based approaches: does not know what it doesn't know
- (-) As with most DNN-based ML approaches: missing model explainability
- (-) Currently: not always accessible (effort in hardware, data, know-how) ⇒ dependency on large-player models, APIs and solutions.