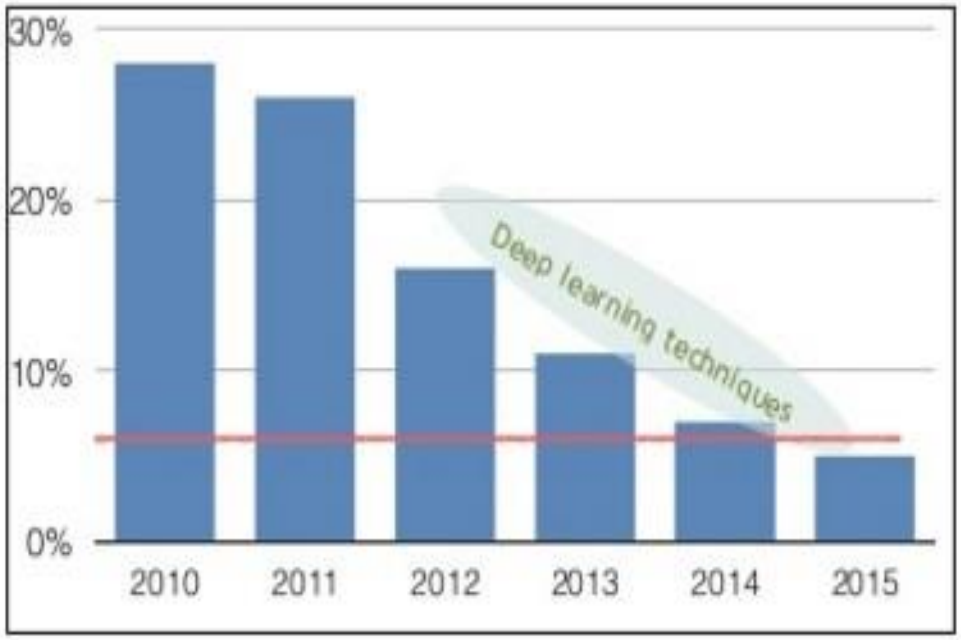
Deep Learning for Recommender Systems

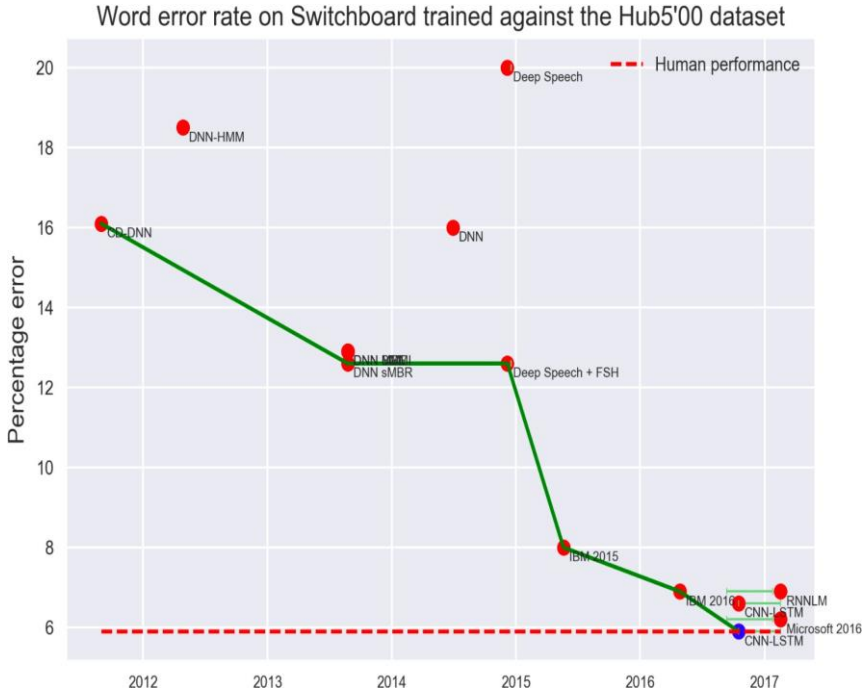
Alexandros Karatzoglou (Scientific Director @ Telefonica Research)
alexk@tid.es, @alexk_z

Balázs Hidasi (Head of Research @ Gravity R&D)
balazs.hidasi@gravityrd.com, @balazshidasi

RecSys'17, 29 August 2017, Como

Why Deep Learning?





ImageNet challenge [error rates] (red line = human performance)

Why Deep Learning?

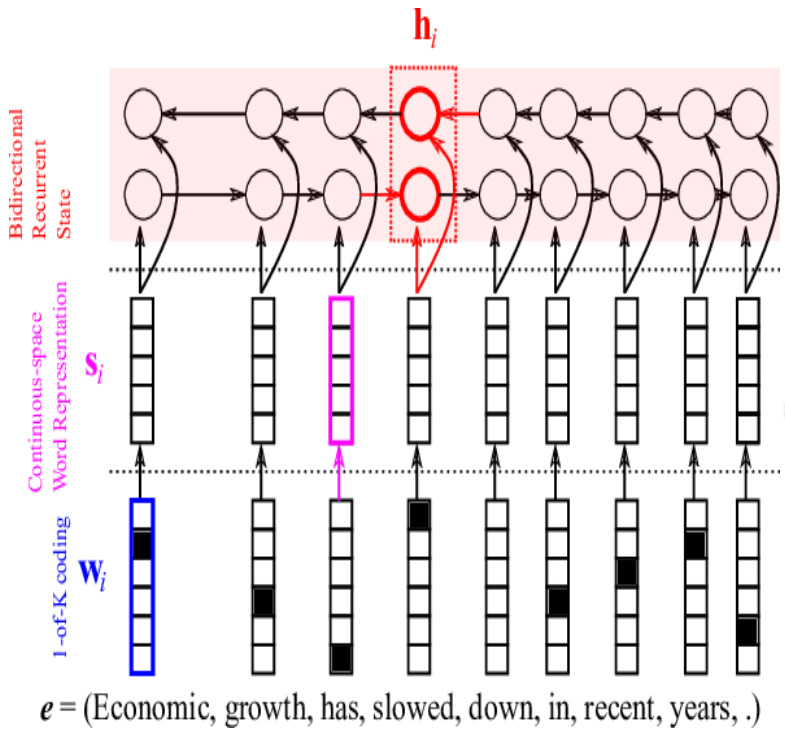
hand-crafted
Feature Extractor

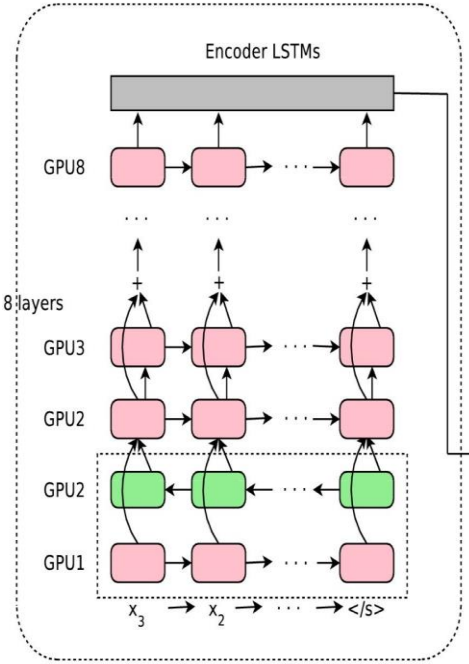
"Simple" Trainable
Classifier

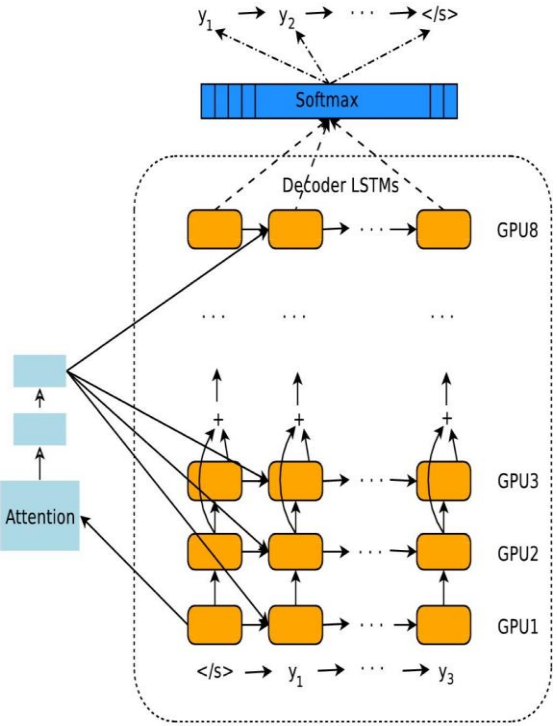
Trainable
Feature Extractor

Trainable
Classifier

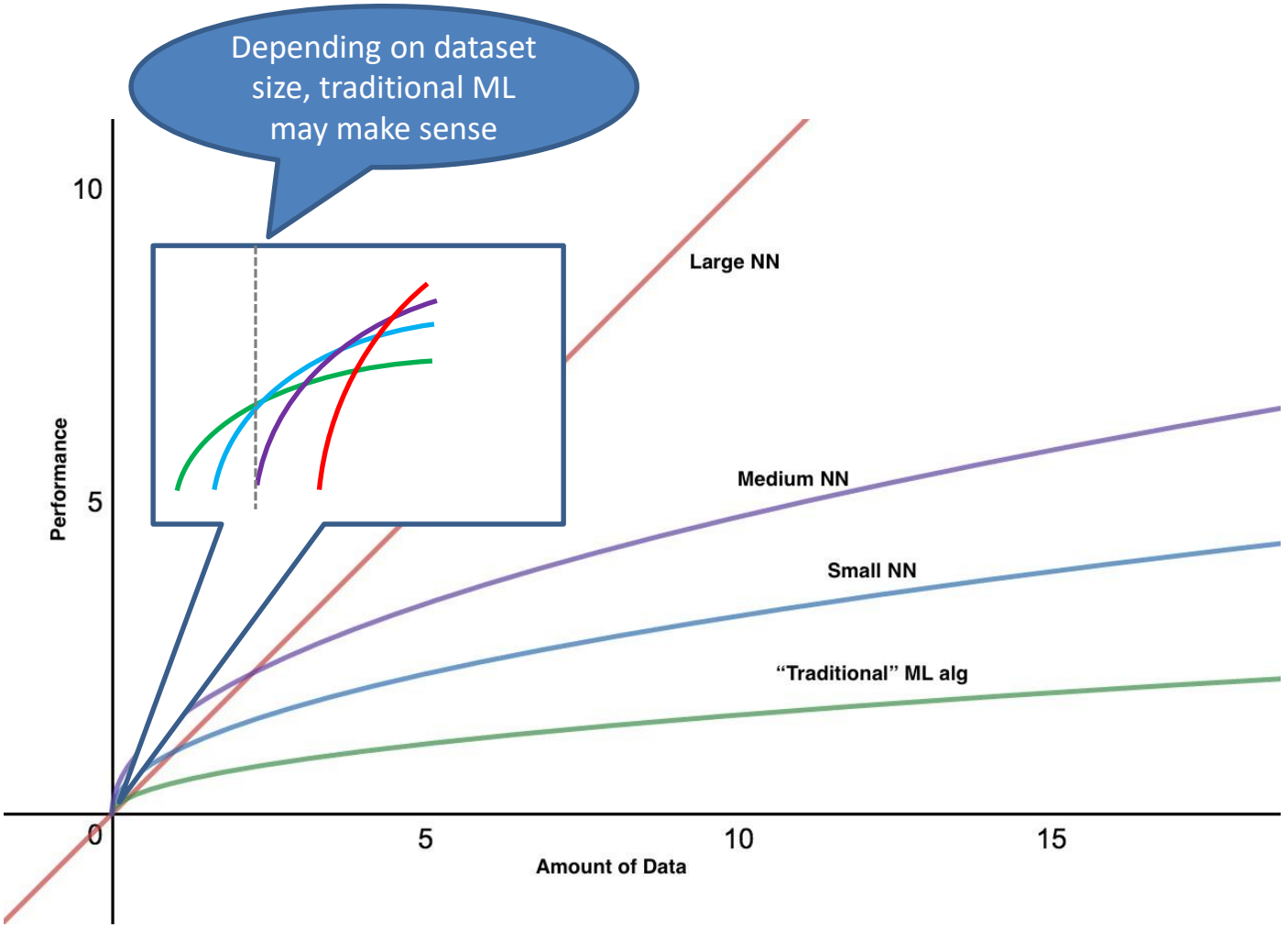
Complex Architectures





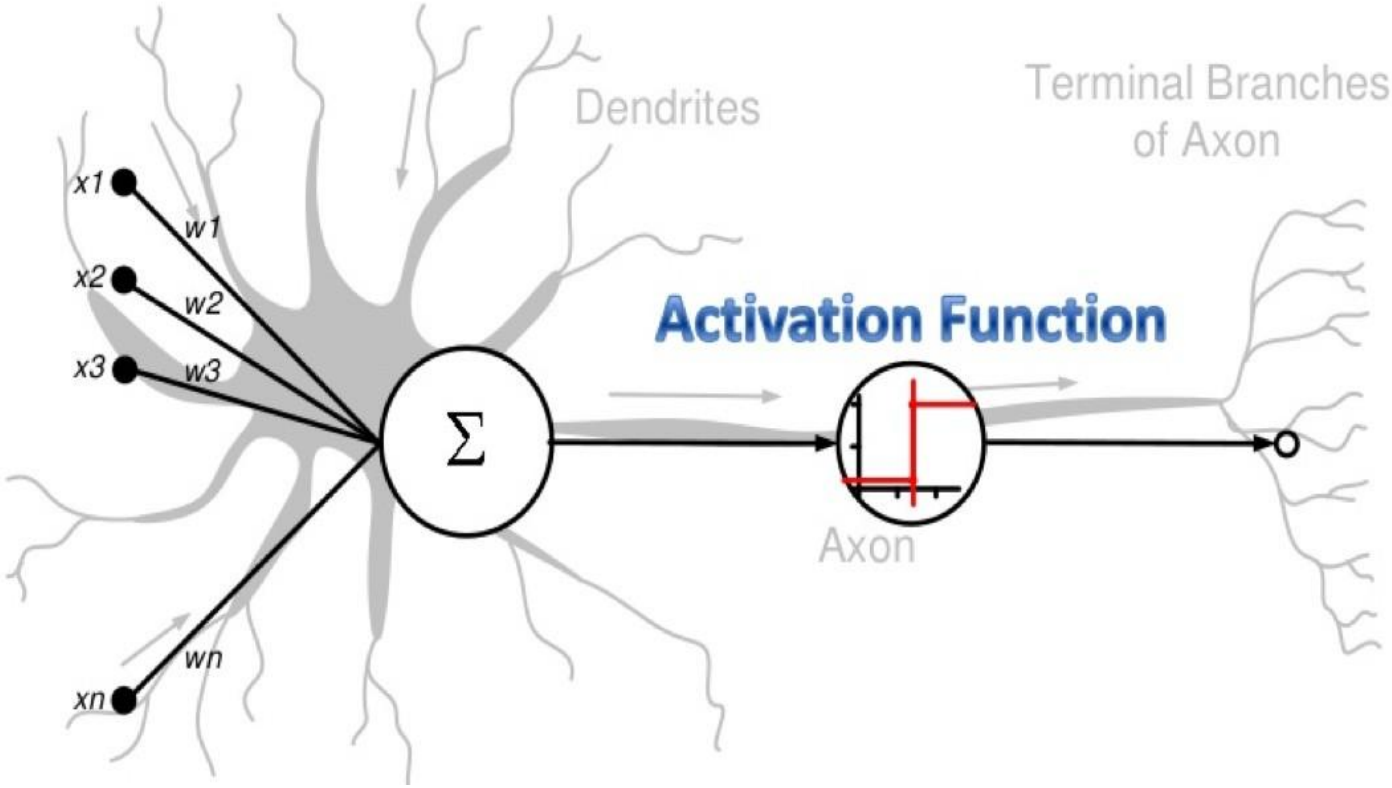


Neural Networks are Universal Function Approximators

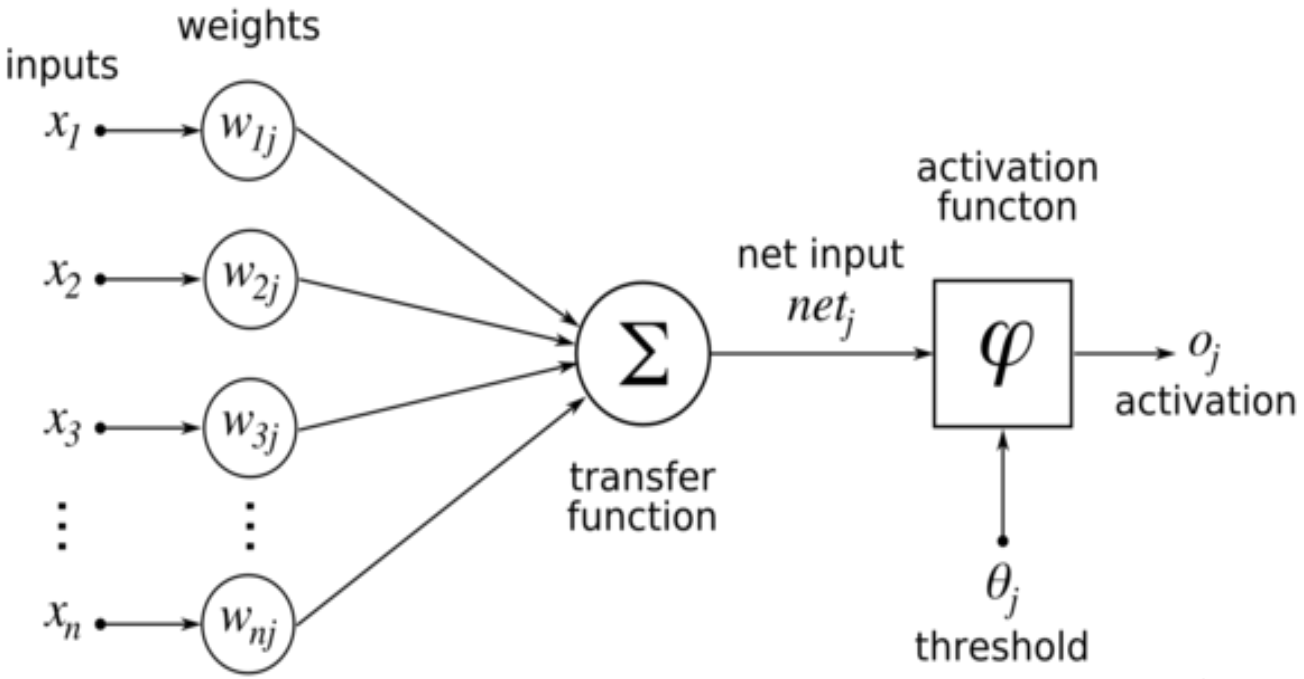


Inspiration for Neural Learning

Neural Model

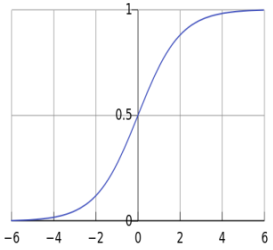


Neuron a.k.a. Unit

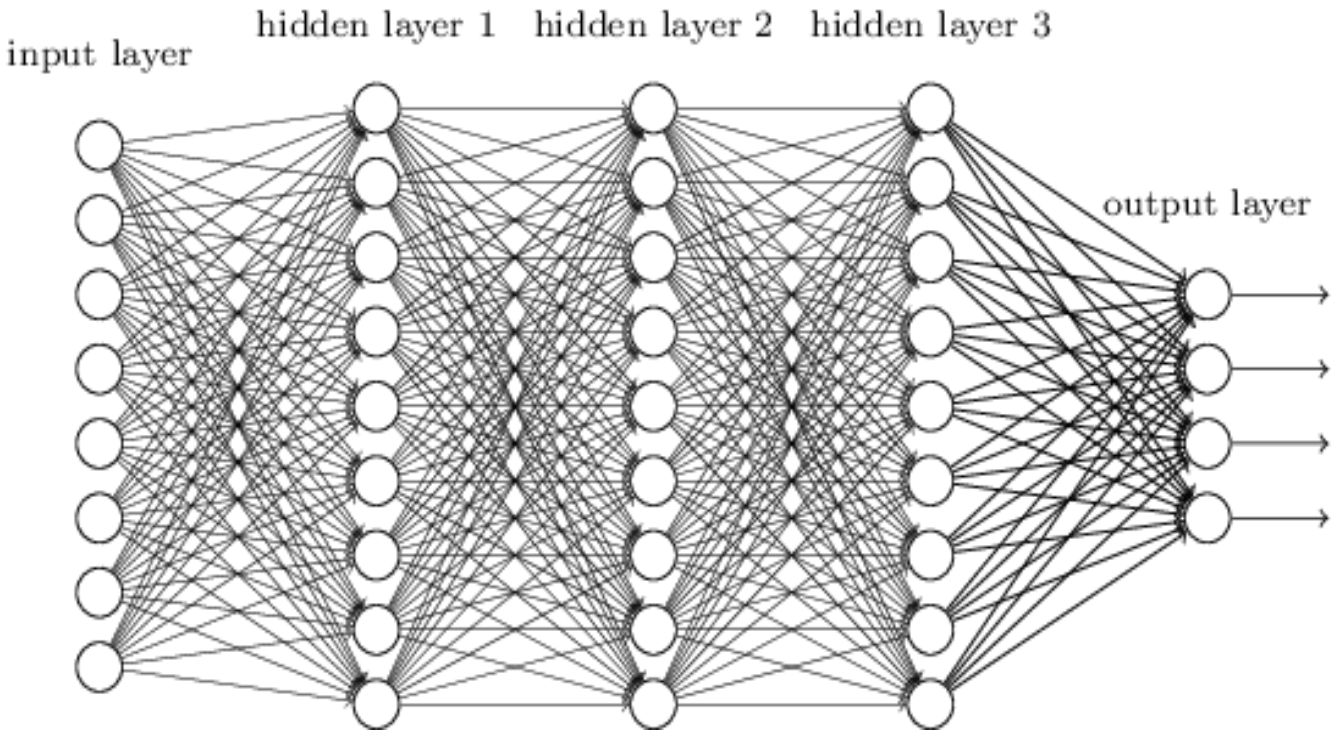


$$P(o_j = 1|x) = \phi\left(\sum_{i=1}^{n} w_{ij}x_i + \theta_j\right)$$

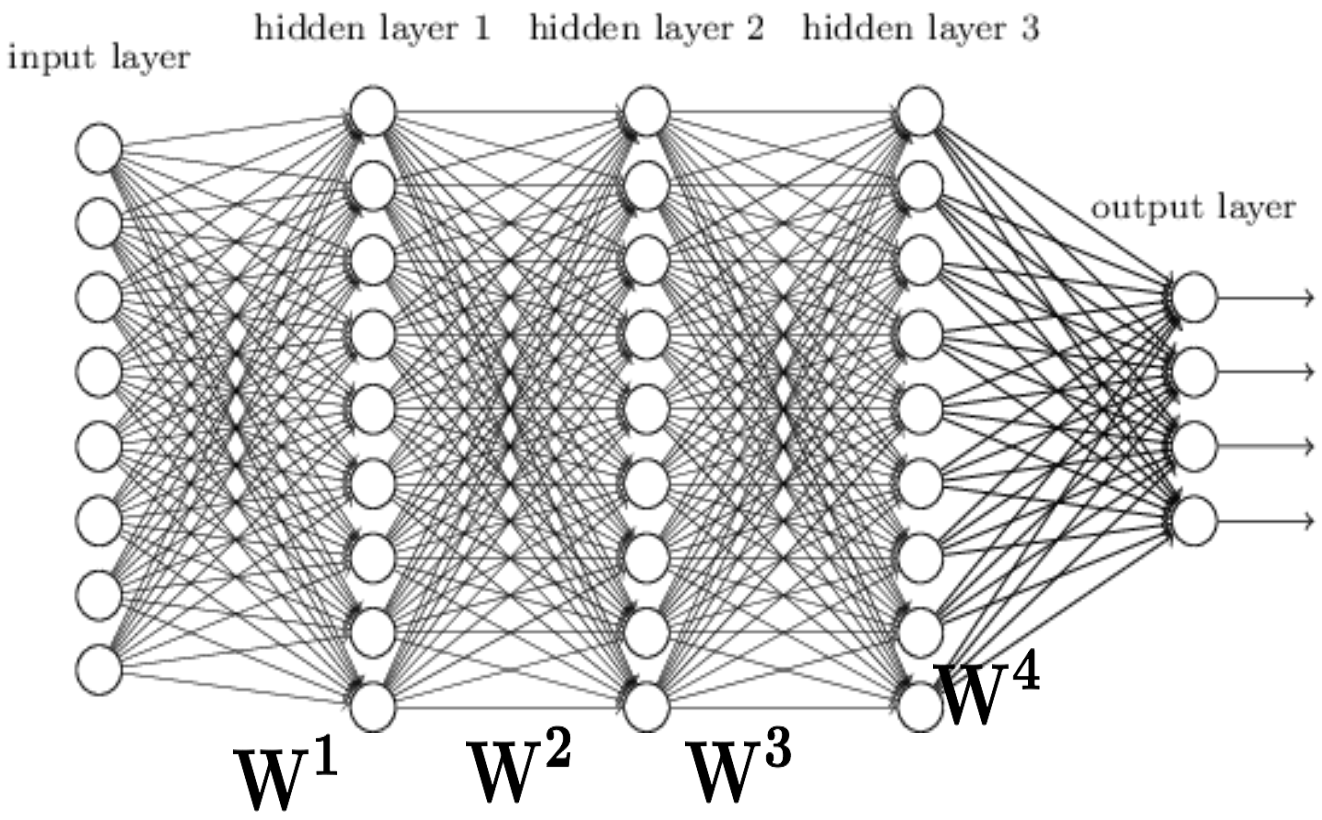
$$\phi = \frac{1}{1 + e^{-\Sigma}}$$



Feedforward Multilayered Network



Learning


input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer
W^1
W^2
W^3
W^4

Stochastic Gradient Descent

- Generalization of (Stochastic) Gradient Descent

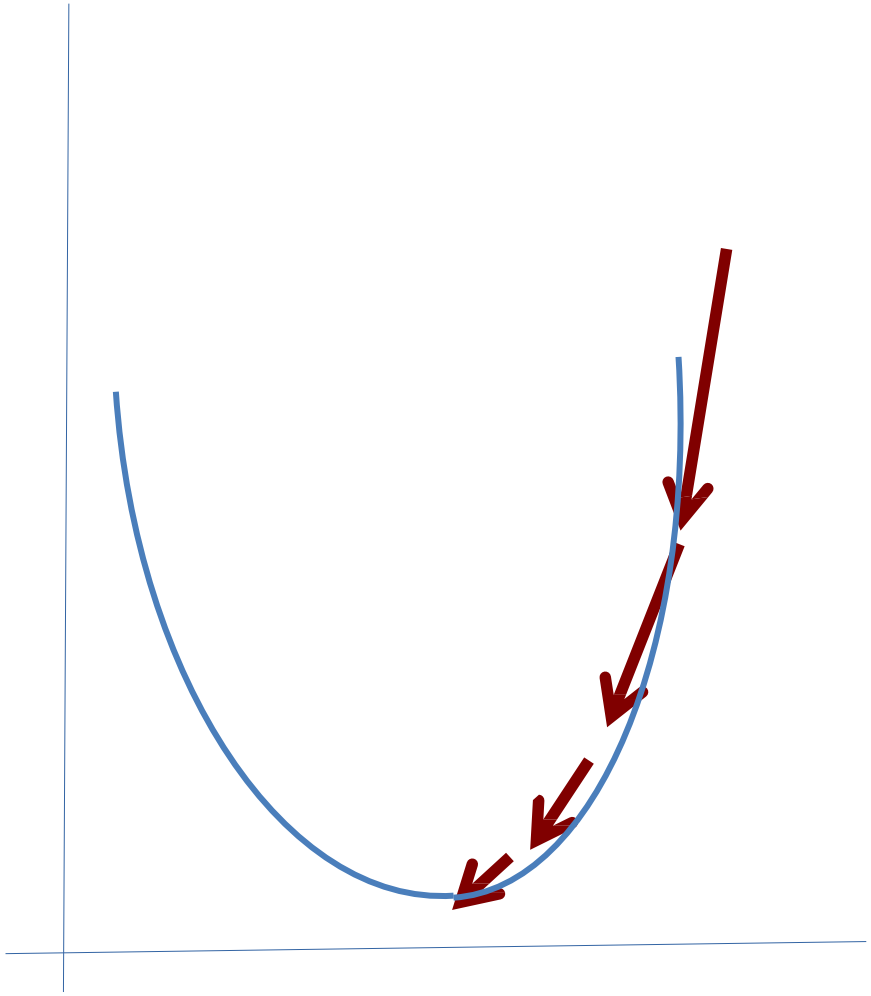
$$E = \frac{1}{2}(f - y)^2$$

$$f = \mathbf{w}^T \mathbf{x}$$

$$for \quad i = 1, 2, \ldots, n$$

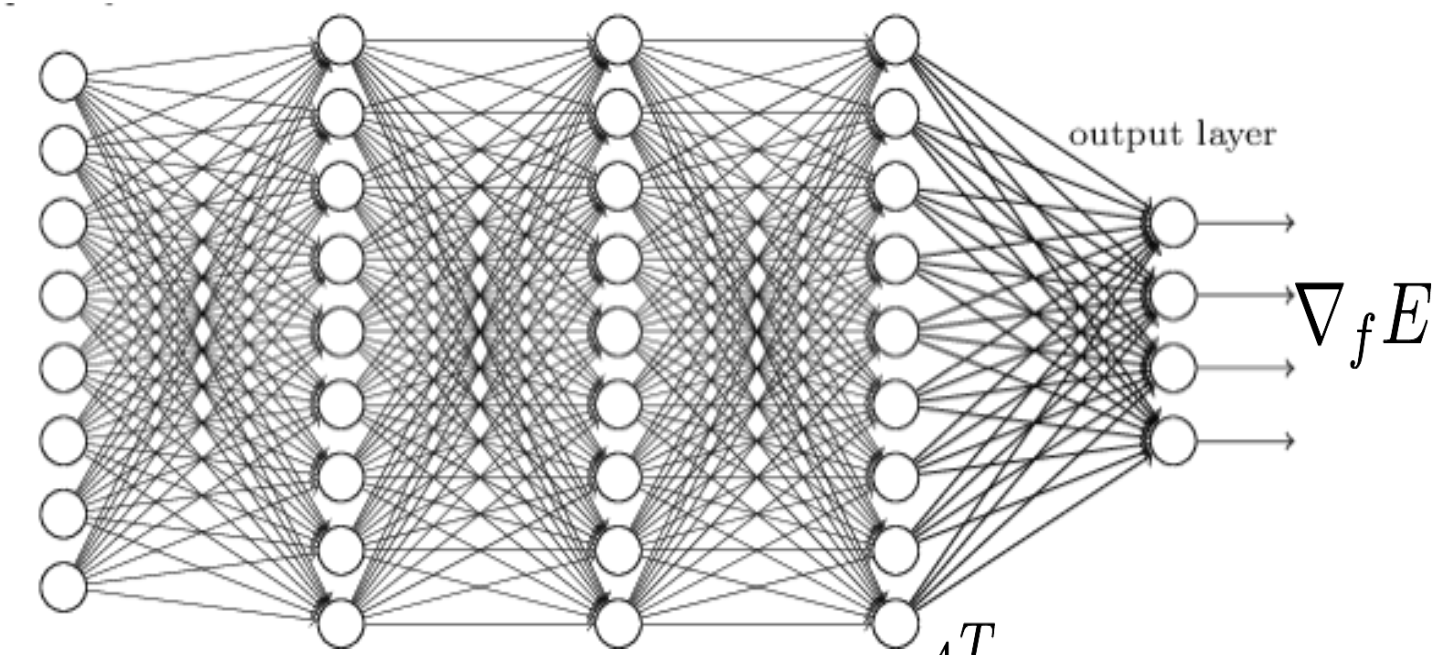
$$\mathbf{w} := \mathbf{w} - \eta \nabla_f E \mathbf{x}_i$$

Stochastic Gradient Descent



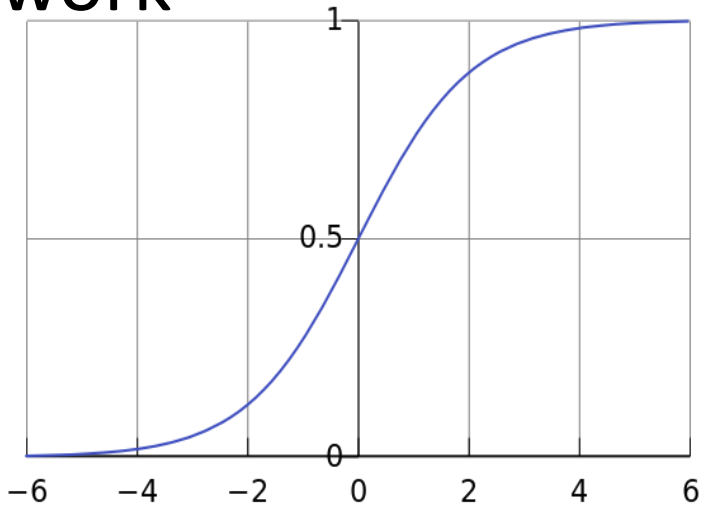
Backpropagation

$$F(x) := \sigma(\sigma(\mathbf{W^1}x)\mathbf{W^2}\dots)$$



Backpropagation

- Does not work well in plain a normal" multilayer deep network
- Vanishing Gradients
- Slow Learning



- SVM's easier to train
- 2nd Neural Winter

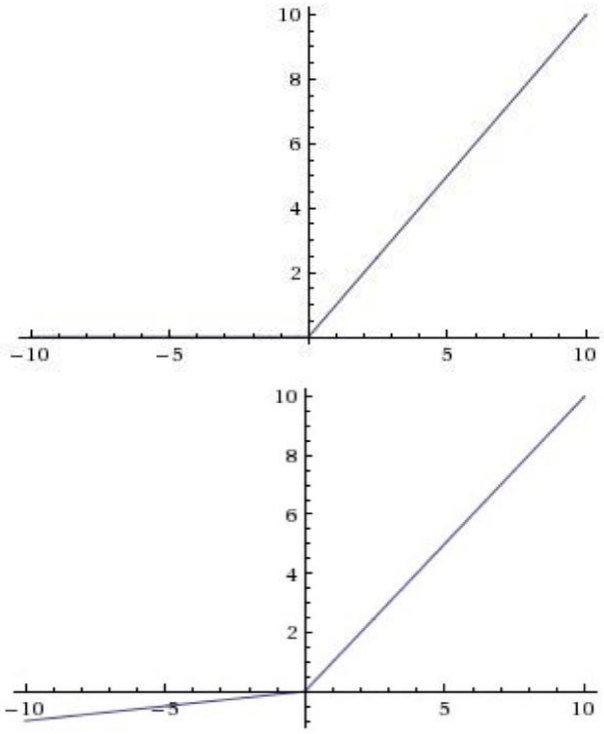
Modern Deep Networks

- Ingredients:

- Rectified Linear Activation function a.k.a. ReLu

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

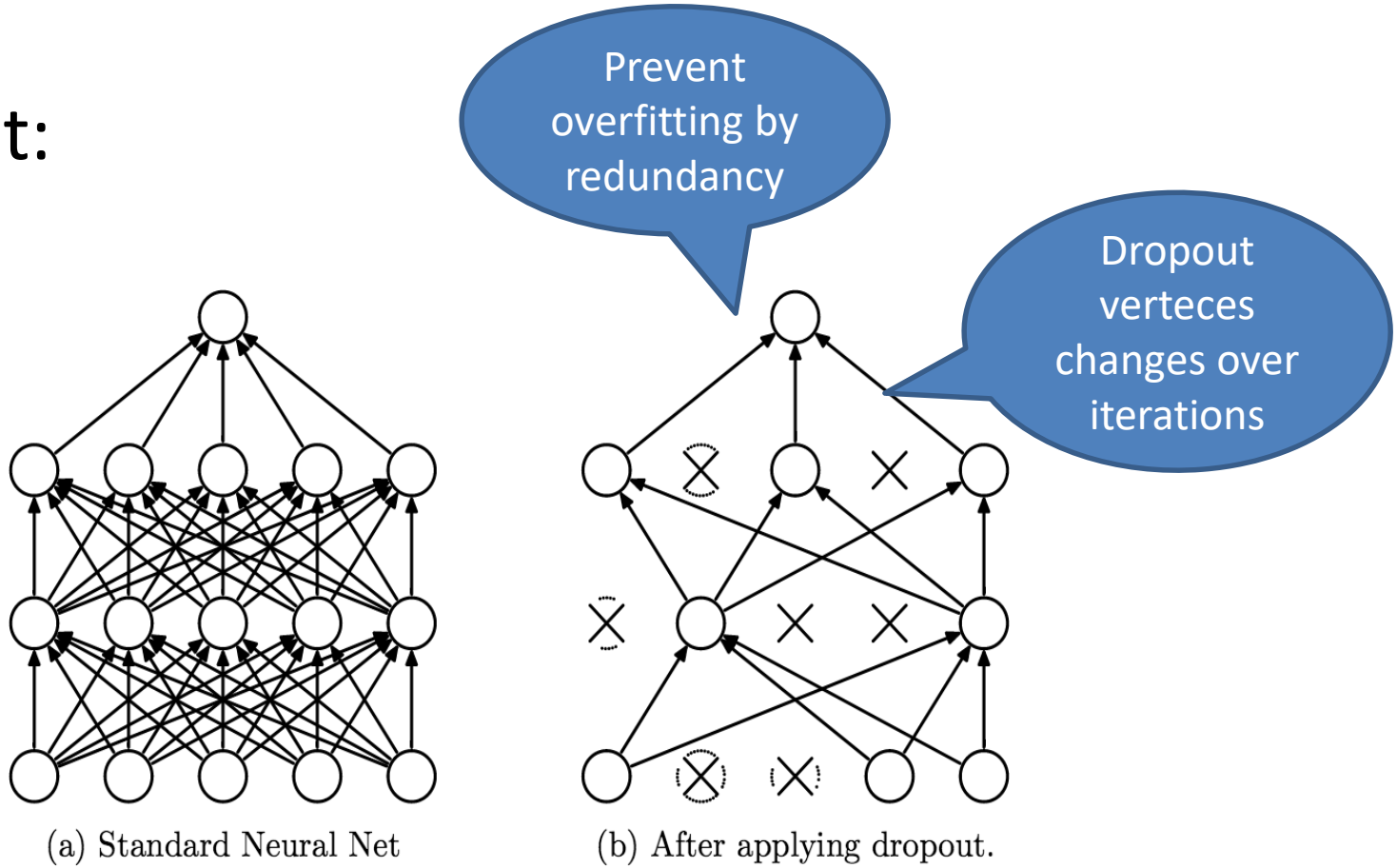
$$\sigma(x) = max(\alpha x, x) \quad \alpha < 1$$



Modern Deep Networks

- Ingredients:

- Dropout:



Modern Deep Networks

- Ingredients:

- Mini-batches:
  - Stochastic Gradient Descent
  - Compute gradient over many (50 -100) data points (minibatch) and update.

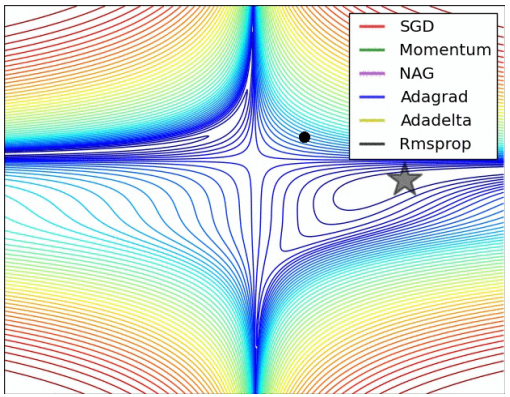
Modern Feedforward Networks

- Ingredients:
- Adagrad a.k.a. adaptive learning rates

$$\mathbf{w} := \mathbf{w} - \frac{\eta}{\sqrt{\sum_{j=0}^{j=i} \nabla_w E_j^2}} \nabla_f E \mathbf{x}_i$$

Decrease step size over time

In a factor-wise fashion



https://ruder.io/optimizing-gradient-descent/

Deep Learning for RecSys

- Feature extraction directly from the content
  - Image, text, audio, etc.
  - Instead of metadata
  - For hybrid algorithms
- Heterogenous data handled easily
  - CB + CF + context + …
- Sequential behaviour modeling (RNNs/Transformers)
- Beyond RecSys paradigms
  - RecSys + LLM / conversational RecSys
- RecSys is a complex domain
  - Deep learning worked well in other complex domains
  - Worth a try

Research directions in DL-RecSys

- As of 2017, main topics:
  - Learning item embeddings
  - ~~*Deep collaborative filtering*~~
  - Feature extraction directly from content
  - Session-based recommendations with RNN
- In 2025
  - Most of newly proposed RecSys algorithms are based on neural networks
  - LLM integration at various stages of recommendations
  - Combining heterogeneous inputs
  - Modality translation (e.g., job posting <-> applicant's CV)

Best practices

- Start simple
  - Add improvements later
- Optimize code
  - GPU/CPU optimizations may differ
- Scalability is key
- Opensource code
- Experiment (also) on public datasets
- Don’t use very small datasets
- Don’t work on irrelevant tasks, e.g. rating prediction

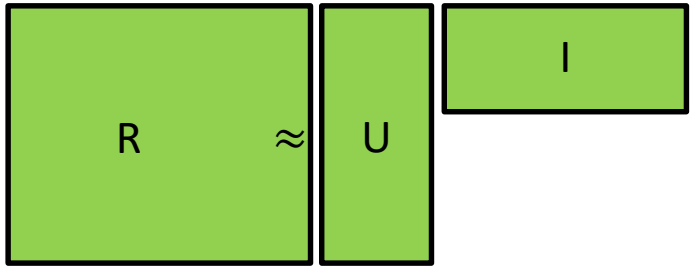
Item embeddings & 2vec models

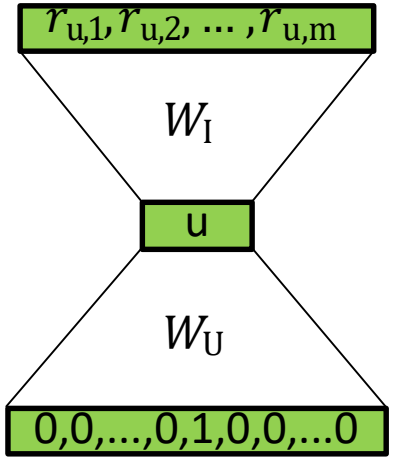
Embeddings

- Embedding: a (learned) real value vector representing an entity
  - Also known as:
    - Latent feature vector
    - (Latent) representation
  - **Similar entities' embeddings are similar**
- Use in recommenders:
  - Initialization of item representation in more advanced algorithms
  - **Item-to-item recommendations**

Matrix factorization as learning embeddings

- MF: user & item embedding learning
  - Similar feature vectors
    - Two items are similar
    - Two users are similar
    - User prefers item
  - MF representation as a simplictic neural network
    - Input: one-hot encoded user ID
    - Input to hidden weights: user feature matrix
    - Hidden layer: user feature vector
    - Hidden to output weights: item feature matrix
    - Output: preference (of the user) over the items
- „Matrix factorization“ in EasyStudy is implemented like this…





Word2Vec

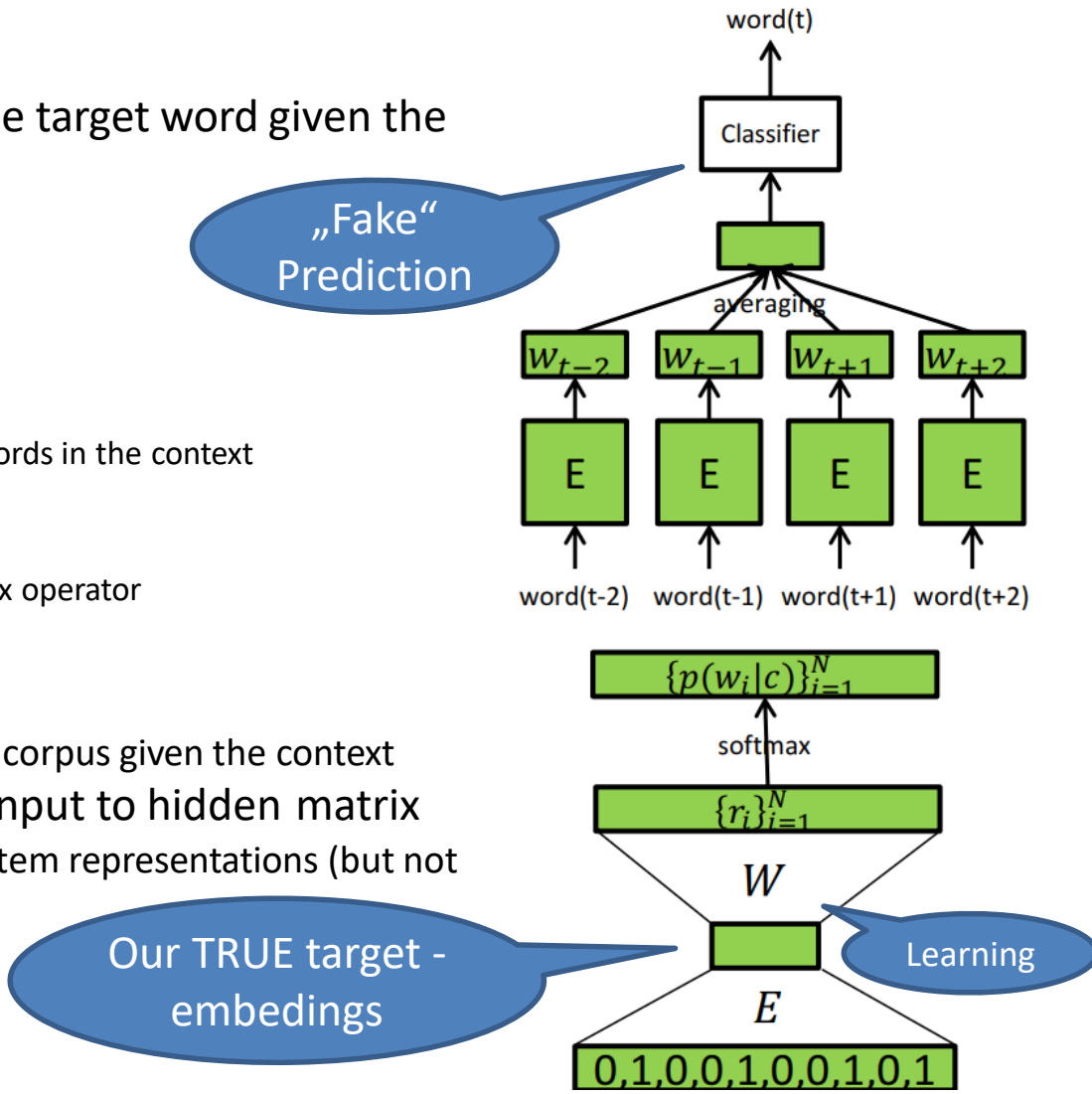
- [Mikolov et. al, 2013a]
- Representation learning of words
- Shallow model
- Data: (target) word + context pairs
  - Sliding window on the document
  - Context = words near the target
    - In sliding window
    - 1-5 words in both directions
- Two models
  - Continous Bag of Words (CBOW)
  - Skip-gram (pairwise relevance)

Word2Vec - CBOW

- Continuous Bag of Words
- Maximalizes the probability of the target word given the context
- Model
  - Input: one-hot encoded words
  - Input to hidden weights
    - Embedding matrix of words
  - Hidden layer
    - Sum of the embeddings of the words in the context
  - Hidden to output weights
  - Softmax transformation
    - Smooth approximation of the max operator
    - Highlights the highest value

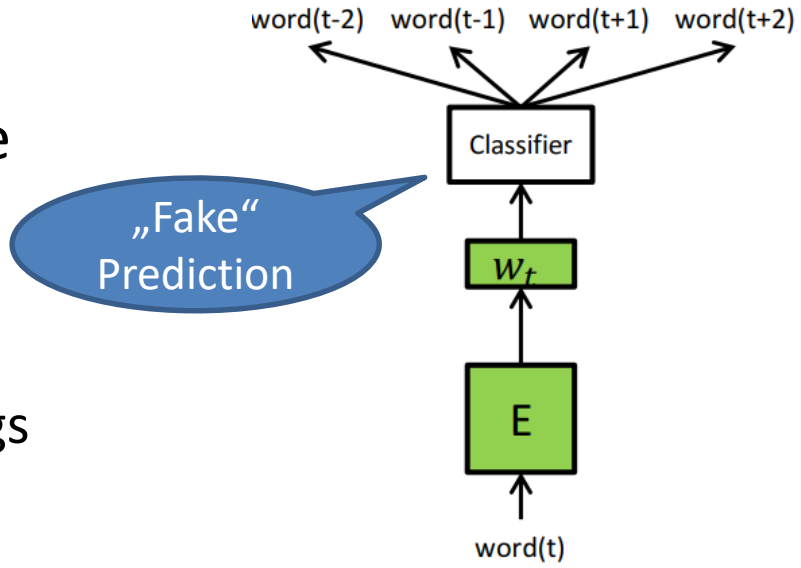
$$s_i = \frac{e^{r_i}}{\sum_{j=1}^{N} e^{r_j}}, (r_j: \text{scores})$$

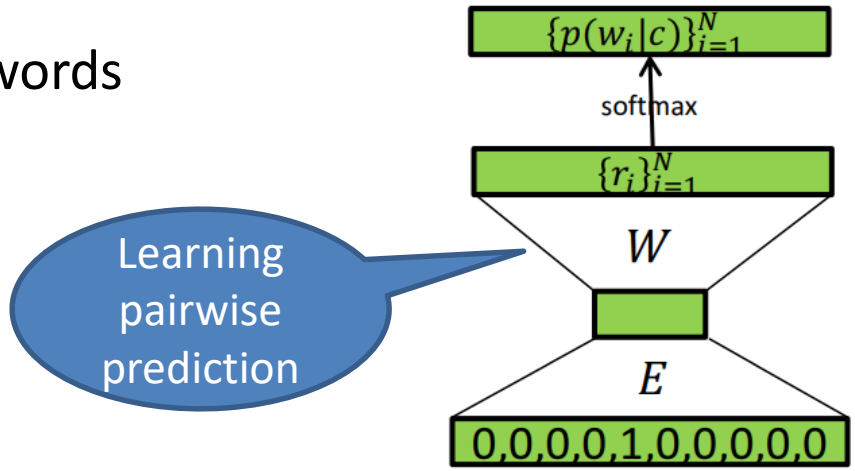
  - Output: likelihood of words of the corpus given the context
- Embeddings are taken from the input to hidden matrix
  - Hidden to output matrix also has item representations (but not used)



Word2Vec – Skip-gram

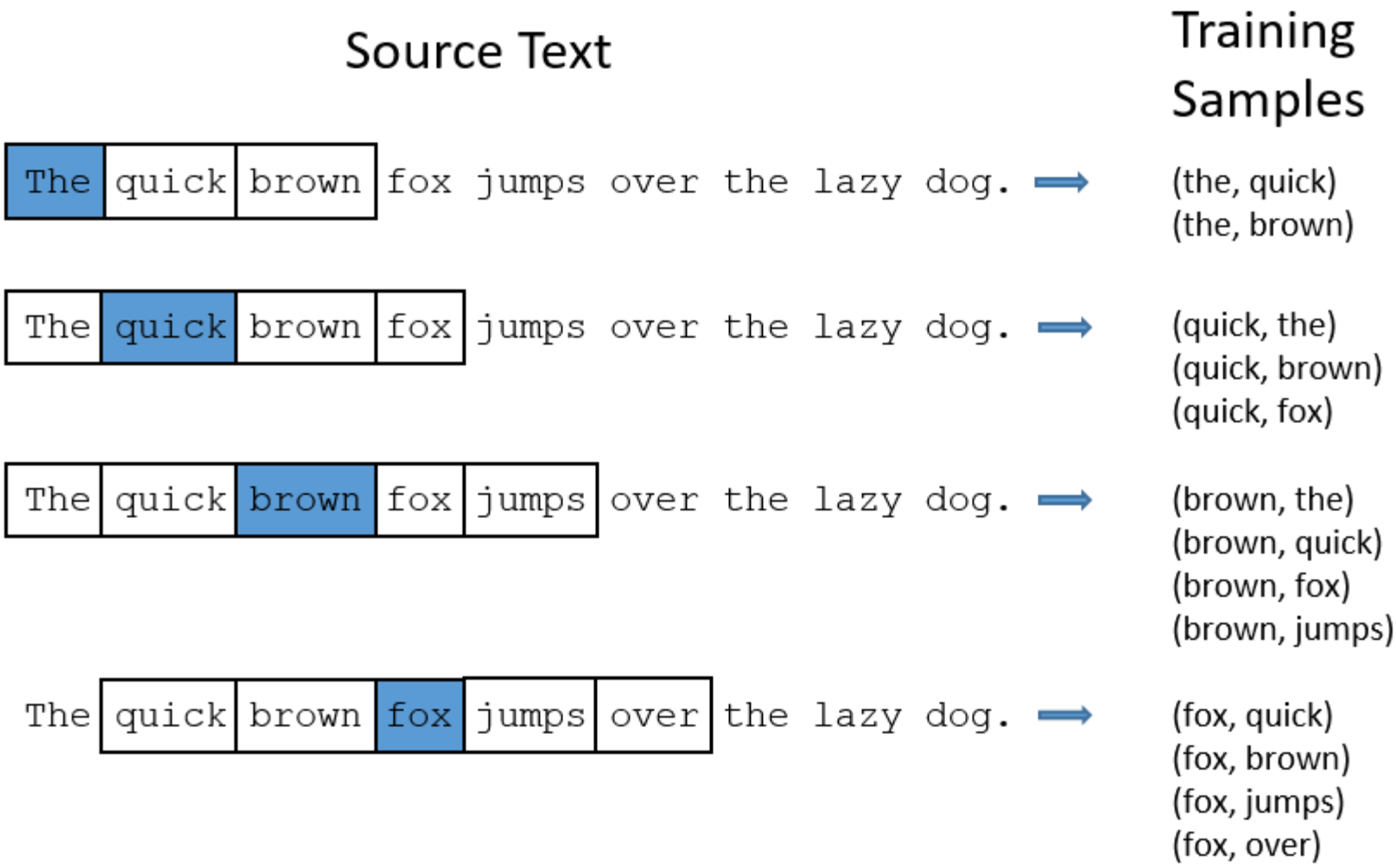
- Maximalizes the probability of the context, given the target word
- Model
  - Input: one-hot encoded word
  - Input to hidden matrix: embeddings
  - Hidden state
    - Item embedding of target
  - Softmax transformation
  - Output: likelihood of context words (given the input word)





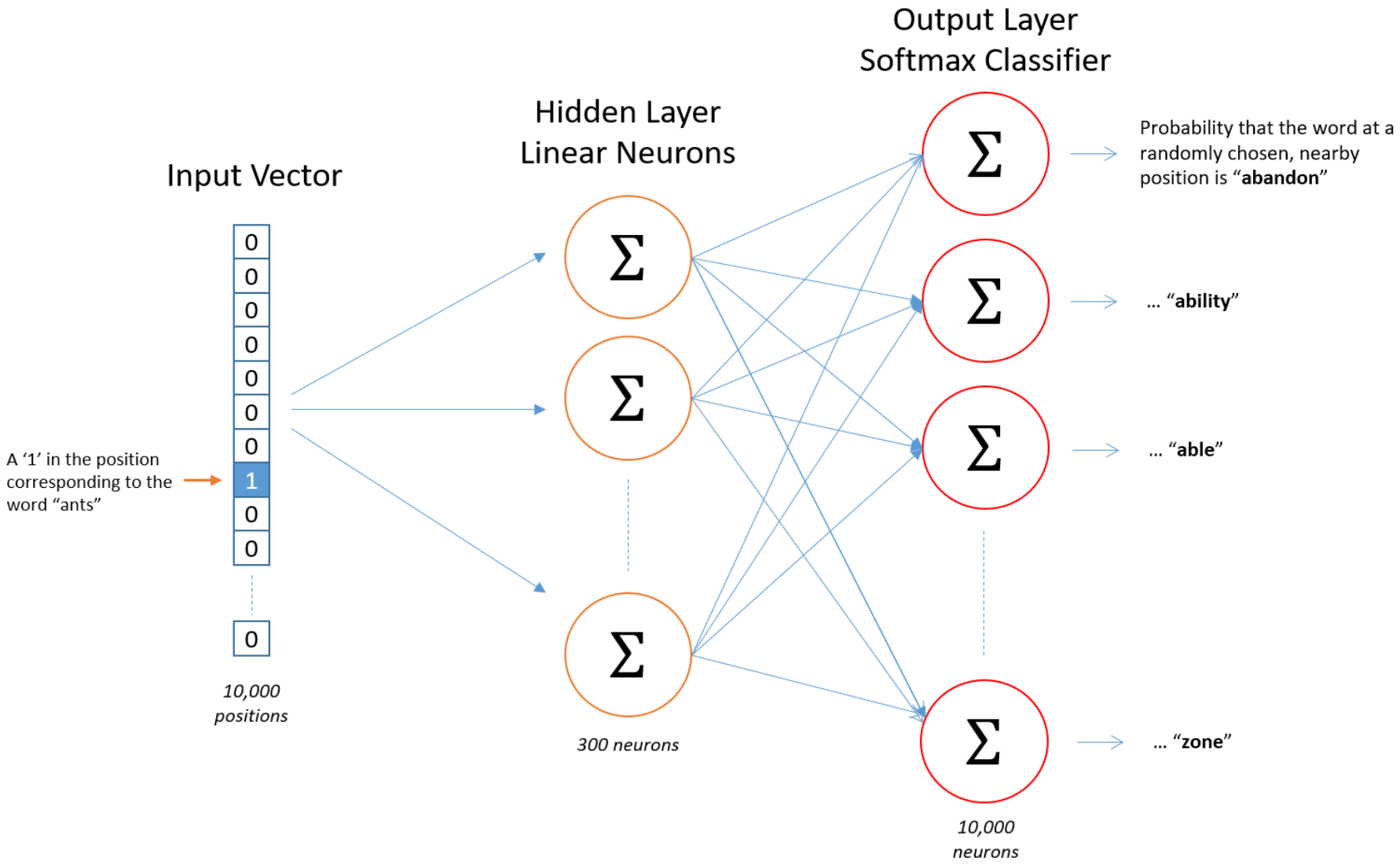
Shared input/output weights for the same words

Word2Vec – Skip-gram



http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

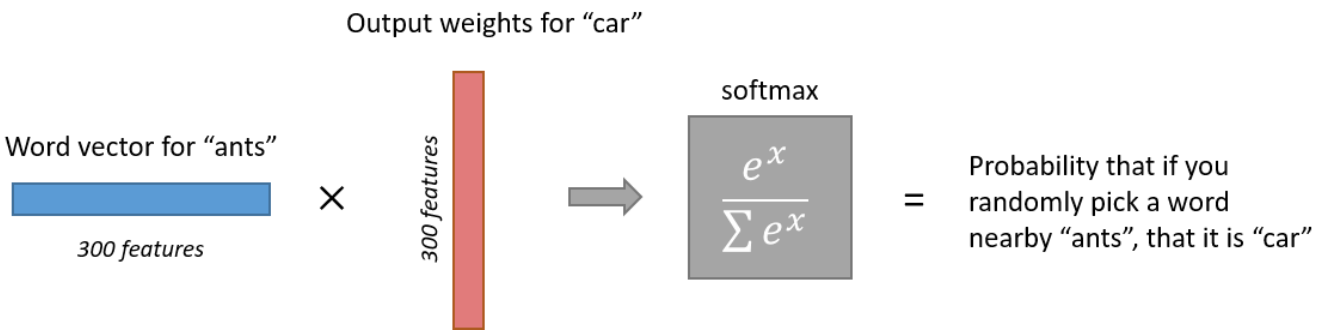
Word2Vec – Skip-gram



Word2Vec – Skip-gram

- Shared weights for the same input and output word
- Afterwards, apply softmax to get a probability distribution

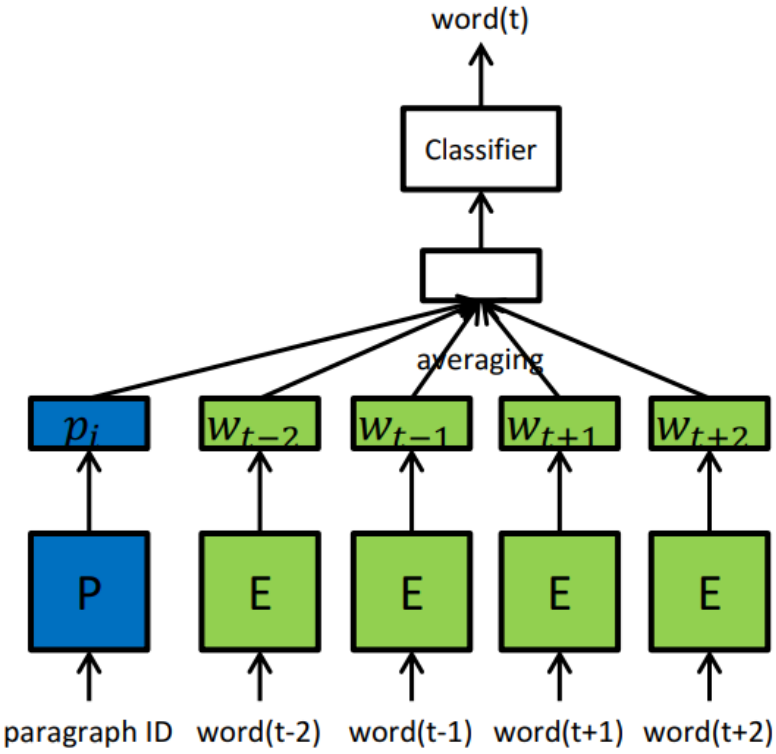
- Still, too many weights -> sample negative elements to be updated
  - Negative sampling, http://mccormickml.com/2017/01/11/word2vec-tutorial-part-2-negative-sampling/



http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

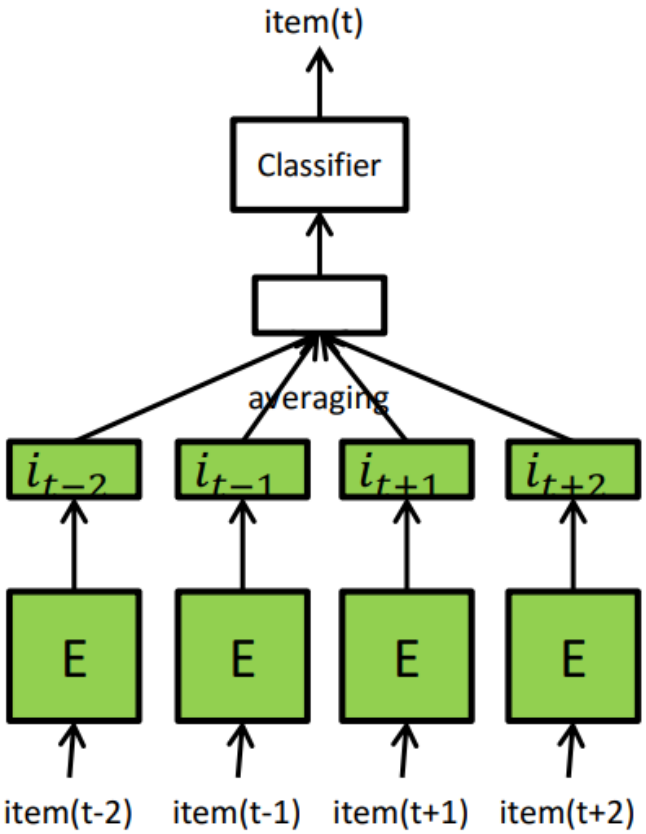
Paragraph2vec, doc2vec

- [Le & Mikolov, 2014]
- Learns representation of paragraph/document
- Based on CBOW model
- Paragraph/document embedding added to the model as global context



...2vec for Recommendations

Replace words with items in a session/user profile

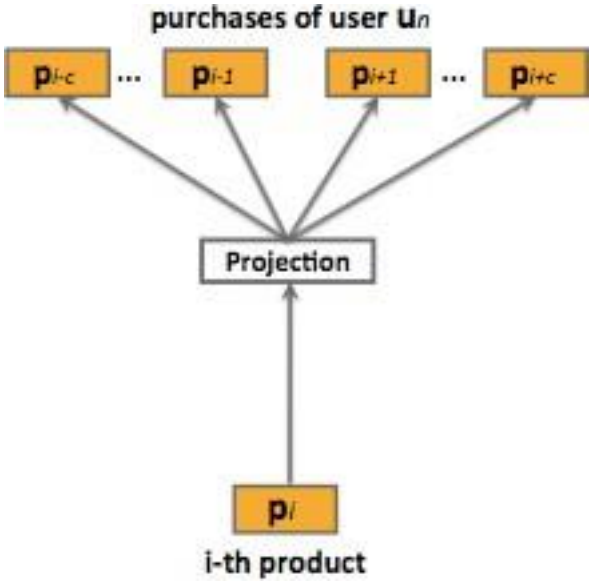


Prod2Vec

- [Grbovic et. al, 2015]
- Skip-gram model on products
  - Input: i-th product purchased by the user
  - Context: the other purchases of the user
- Bagged prod2vec model
  - Input: products purchased in one basket by the user
    - Basket: sum of product embeddings
  - Context: other baskets of the user
- Learning user representation
  - Follows paragraph2vec
  - User embedding added as global context
  - Input: user + products purchased except for the i-th
  - Target: i-th product purchased by the user
- [Barkan & Koenigstein, 2016] proposed the same model later as item2vec
  - Skip-gram with Negative Sampling (SGNS) is applied to event data

Prod2Vec

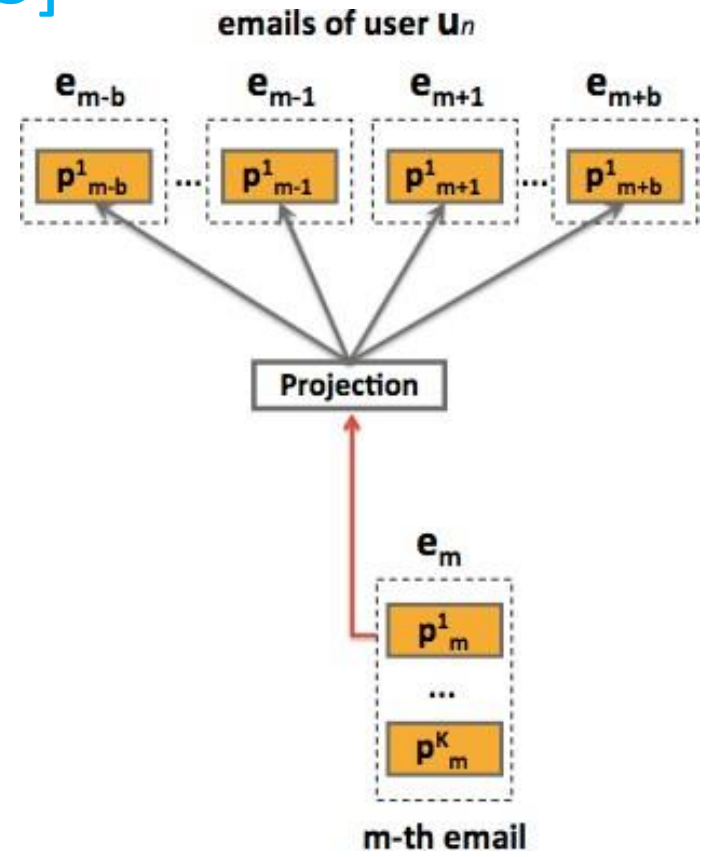
[Grbovic et. al, 2015]



pro2vec skip-gram model on products

Bagged Prod2Vec

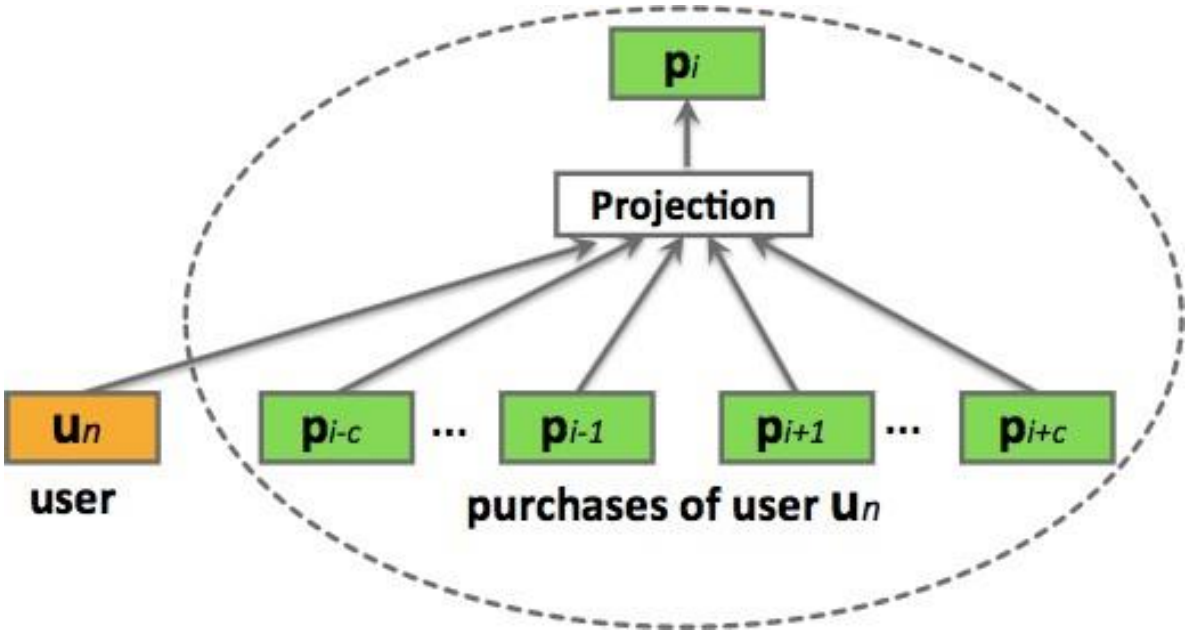
[Grbovic et. al, 2015]



bagged-prod2vec model updates

User-Prod2Vec

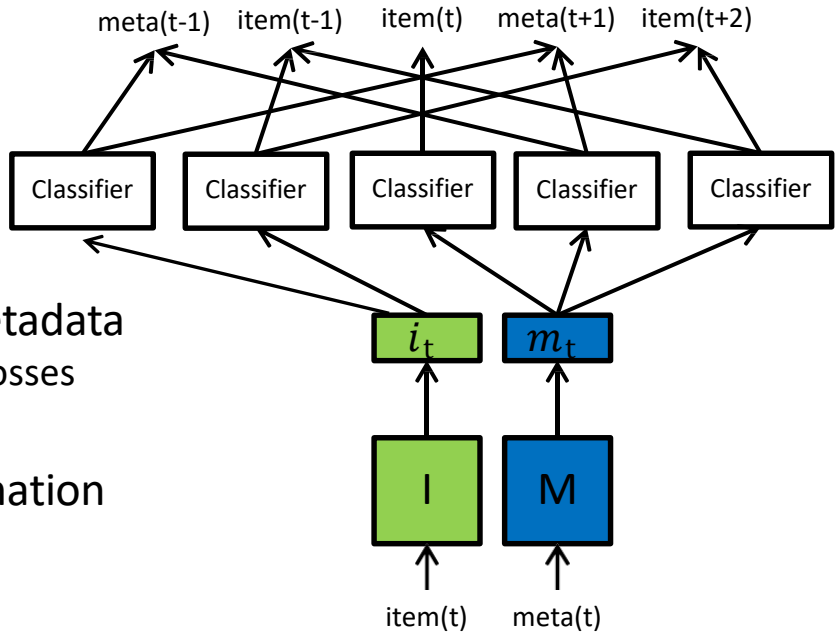
[Grbovic et. al, 2015]



User embeddings for user to product predictions

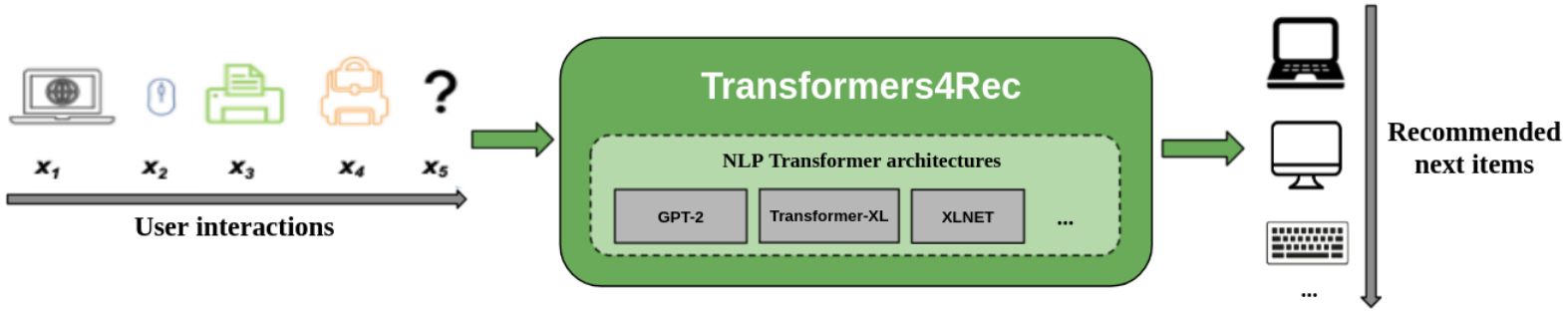
Utilizing more information

- Meta-Prod2vec [Vasile et. al, 2016]
  - Based on the prod2vec model
  - Uses item metadata
    - Embedded metadata
    - Added to both the input and the context
  - Losses between: target/context item/metadata
    - Final loss is the combination of 5 of these losses
- Content2vec [Nedelec et. al, 2017]
  - Separate modules for multimodel information
    - CF: Prod2vec
    - Image: AlexNet (a type of CNN)
    - Text: Word2Vec and TextCNN
  - Learns pairwise similarities
    - Likelihood of two items being bought together



Follow-up Development

- Usually, 1-2 years after a new network is adopted for NPL, it is tested in RecSys domain
  - https://github.com/NVIDIA-Merlin/Transformers4Rec



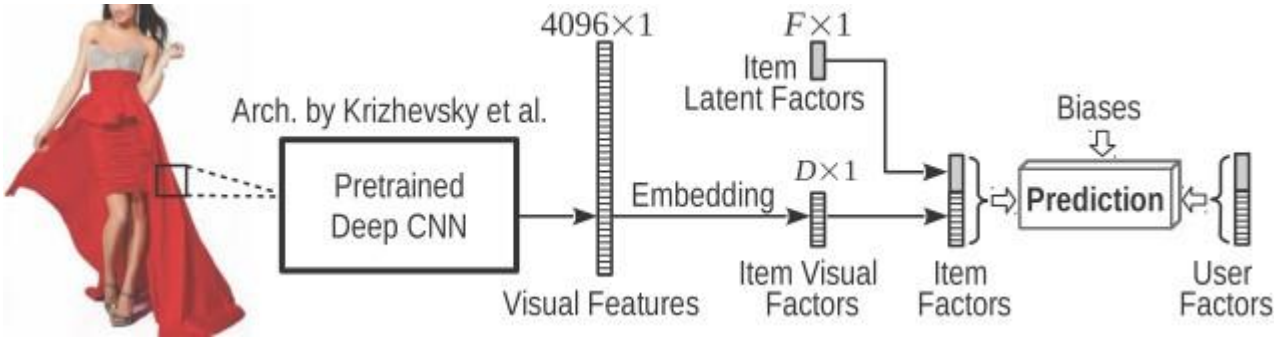
Feature extraction from content

- Deep learning is capable of direct feature extraction
  - Work with content directly
  - Instead (or beside) metadata
- Images
  - E.g.: product pictures, video thumbnails/frames
  - Extraction: convolutional networks
  - Applications (e.g.):
    - Fashion
    - Video
- Text
  - E.g.: product description, content of the product, reviews
  - Extraction
    - RNNs
    - 1D convolution networks
    - Weighted word embeddings
    - Paragraph vectors
    - BERT, CLIP, LLMs
  - Applications (e.g.):
    - News
    - Books
    - Publications
- Music/audio
  - Extraction: convolutional networks (or RNNs)

Images in recommenders

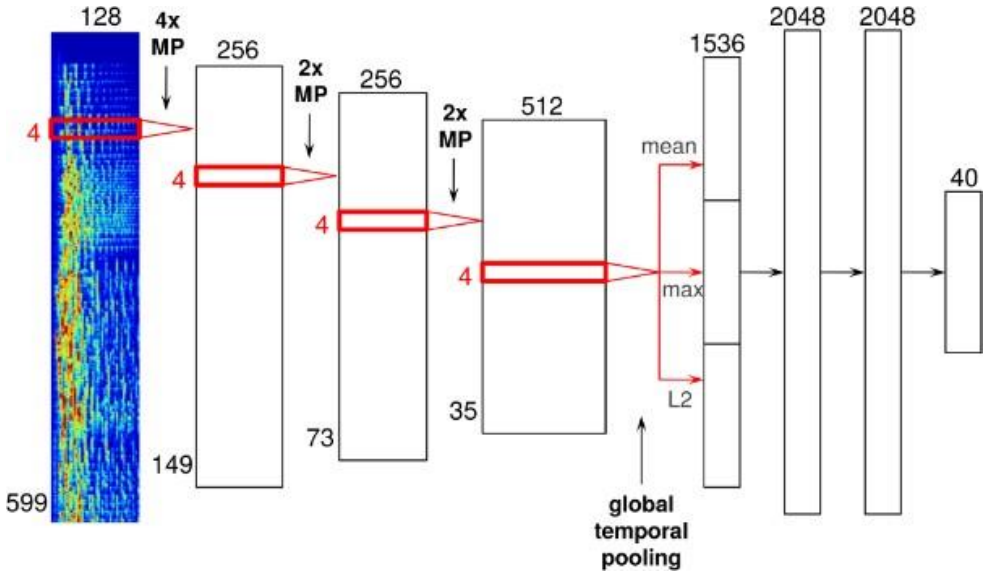
- Visual BPR [He & McAuley, 2016]
  - Model composed of
    - Bias terms
    - MF model
    - Visual part
      - Pretrained CNN features
      - Dimension reduction through „embedding”
      - The product of this visual item feature and a learned user feature vector is used in the model
    - Visual bias
      - Product of the pretrained CNN features and a global bias vector over its features
  - BPR loss
  - Tested on clothing datasets (9-25% improvement)

Whole new fashion RecSys research emerged after this. E.g. finding outfits (i.e. Products matching together)



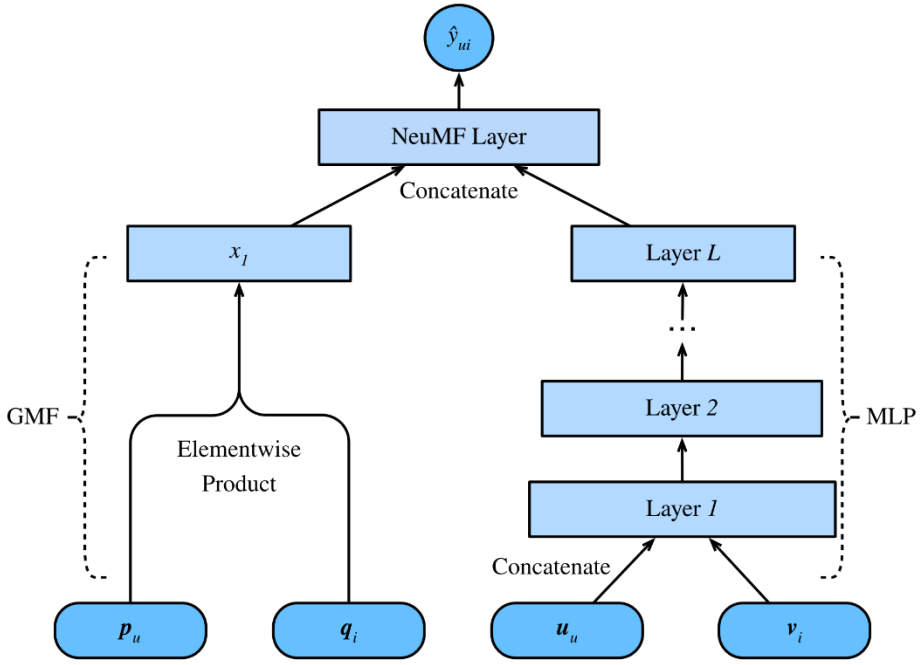
Music representations

- [Oord et. al, 2013]
  - Extends iALS/WMF with audio features
    - To overcome cold-start
  - Music feature extraction
    - Time-frequency representation
    - Applied CNN on 3 second samples
    - Latent factor of the clip: average predictions on consecutive windows of the clip
  - Integration with MF
    - (a) Minimize distance between music features and the MF's feature vectors
    - (b) Replace the item features with the music features (minimize original loss)



Neural Collaborative Filtering

- NeuMF (He *et al.*, 2017)
- Instead of a simple dot product, try MLP
  - Theoretically, learn more complex dependencies
  - Reported improvements, but…



Neural Collaborative Filtering

- NeuMF (He *et al.*, 2017)
- Instead of a simple dot product, try MLP
  - Reported improvements, but

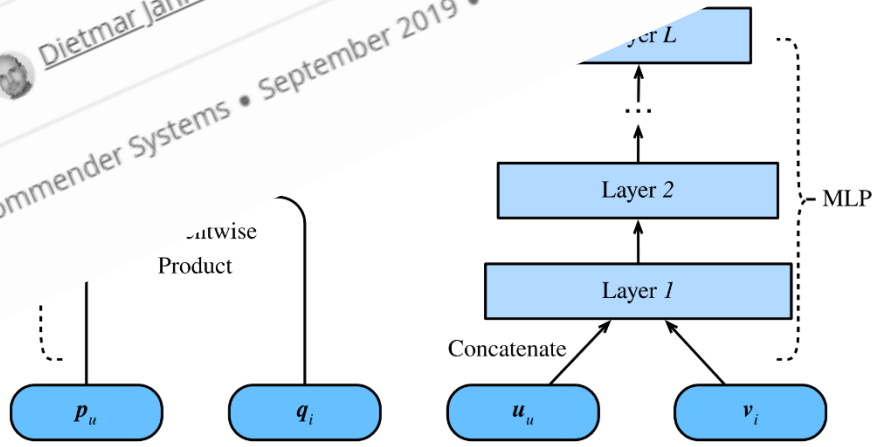
RESEARCH-ARTICLE

Are we really making much progress? A worrying analysis of recent neural recommendation approaches

Authors: Maurizio Ferrari Dacrema, Paolo Cremonesi, Dietmar Jannach Authors Info & Claims

RecSys '19: Proceedings of the 13th ACM Conference on Recommender Systems • September 2019 • Pages 101–109

• https://doi.org/10.1145/3298689.3347058



Neural Collaborative Filtering

- NeuMF / NCF (He *et al.*, 2017)
- Instead of a simple dot product, try MLP
  - Reported improvements, but…
  - … w.r.t. under-optimized baselines
  - Often, simple KNN approaches wins if tuned properly

Table 4: Experimental results for CVAE (CiteULike-a).

	REC@50	REC@100	REC@300
TopPopular	0.0044	0.0081	0.0258
UserKNN	0.0683	0.1016	0.1685
ItemKNN	**0.0788**	0.1153	0.1823
$P^3\alpha$	**0.0788**	0.1151	0.1784
$RP^3\beta$	**0.0811**	0.1184	0.1799
ItemKNN-CFCBF	**0.1837**	**0.2777**	**0.4486**
CVAE	0.0772	0.1548	0.3602

Table 6: Experimental results for NCF.

	Pinterest			
	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.1663	0.1065	0.2744	0.1412
UserKNN	0.7001	**0.5033**	0.8610	**0.5557**
ItemKNN	**0.7100**	**0.5092**	**0.8744**	**0.5629**
$P^3\alpha$	0.7008	**0.5018**	0.8667	**0.5559**
$RP^3\beta$	**0.7105**	**0.5116**	**0.8740**	**0.5650**
NeuMF	0.7024	0.4983	0.8719	0.5536
	Movielens 1M			
	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.3043	0.2062	0.4531	0.2542
UserKNN	0.4916	0.3328	0.6705	0.3908
ItemKNN	0.4829	0.3328	0.6596	0.3900
$P^3\alpha$	0.4811	0.3331	0.6464	0.3867
$RP^3\beta$	0.4922	0.3409	0.6715	0.3991
NeuMF	0.5486	0.3840	0.7120	0.4369
SLIM	**0.5589**	**0.3961**	**0.7161**	**0.4470**

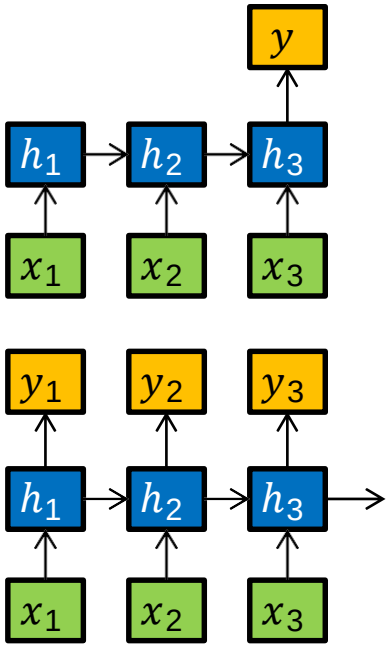
Session-based Recommendations with RNNs

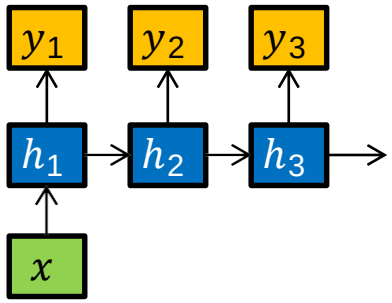
Recurrent Neural Networks

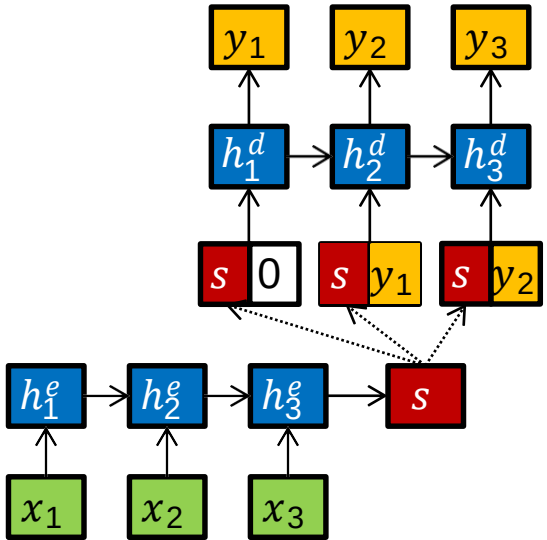
- Input: sequential information $\{x_t\}_{t=1}^{T}$
- Hidden state (h_t):
  - representation of the sequence so far
  - influenced by every element of the sequence up to t
- $h_t = f(W x_t + U h_{t-1} + b)$

RNN-based machine learning

- Sequence to value
  - Encoding, labeling
  - E.g.: time series classification
- Value to sequence
  - Decoding, generation
  - E.g.: sequence generation
- Sequence to sequence
  - Simultaneous
    - E.g.: next-click prediction
  - Encoder-decoder architecture
    - E.g.: machine translation
    - Two RNNs (encoder & decoder)
      - Encoder produces a vector describing the sequence
        - » Last hidden state
        - » Combination of hidden states (e.g. mean pooling)
        - » Learned combination of hidden states
      - Decoder receives the summary and generates a new sequence
        - » The generated symbol is usually fed back to the decoder
        - » The summary vector can be used to initialize the decoder
        - » Or can be given as a global context
    - Attention mechanism (optionally)







Exploding/Vanishing gradients

- $h_t = f(W x_t + U h_{t-1} + b)$
- Gradient of h_t wrt. x_1
  - Simplification: linear activations
    - In reality: bounded
  - $$\frac{\partial h_t}{\partial x_1} = \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \cdots \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial x_1} = U^{t-1} W$$

  - $\|U\|_2 < 1$ → vanishing gradients
    - The effect of values further in the past is neglected
    - The network forgets
  - $\|U\|_2 > 1$ → exploding gradients
    - Gradients become very large on longer sequences
    - The network becomes unstable

Handling exploding gradients

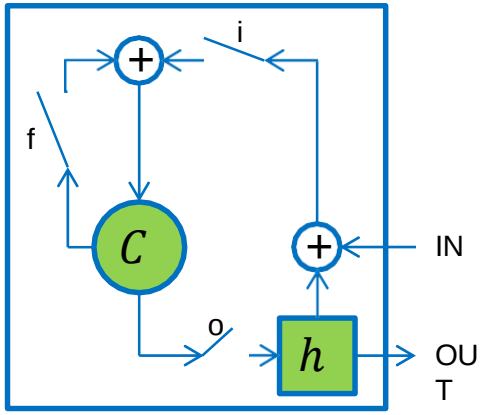
- Gradient clipping
  - If the gradient is larger than a threshold, scale it back to the threshold
  - Updates are not accurate
  - Vanishing gradients are not solved
- Enforce $\|U\|_2 = 1$
  - Unitary RNN
  - Unable to forget
- Gated networks
  - Long-Short Term Memory (LSTM)
  - Gated Recurrent Unit (GRU)
  - (and a some other variants)

Long-Short Term Memory (LSTM)

- [Hochreiter & Schmidhuber, 1999]
- Instead of rewriting the hidden state during update, add a delta
  - $s_t = s_{t-1} + \Delta s_t$
  - Keeps the contribution of earlier inputs relevant
- Information flow is controlled by gates
  - Gates depend on input and the hidden state
  - Between 0 and 1
  - Forget gate (f): 0/1 → reset/keep hidden state
  - Input gate (i): 0/1 → don't/do consider the contribution of the input
  - Output gate (o): how much of the memory is written to the hidden state
- Hidden state is separated into two (read before you write)
  - Memory cell (c): internal state of the LSTM cell
  - Hidden state (h): influences gates, updated from the memory cell

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$\tilde{c}_t = \tanh(W x_t + U h_{t-1} + b)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$
$$h_t = o_t \circ \tanh(c_t)$$

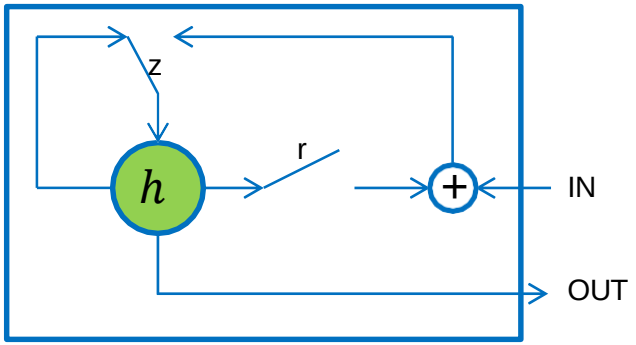


Gated Recurrent Unit (GRU)

- [Cho et. al, 2014]
- Simplified information flow
  - Single hidden state
  - Input and forget gate merged → update gate (z)
  - No output gate
  - Reset gate (r) to break information flow from previous hidden state
- Similar performance to LSTM

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$

$$\tilde{h}_t = \tanh(W x_t + r_t \circ U h_{t-1} + b)$$
$$h_t = z_t \circ h_t + (1 - z_t) \circ \tilde{h}_t$$

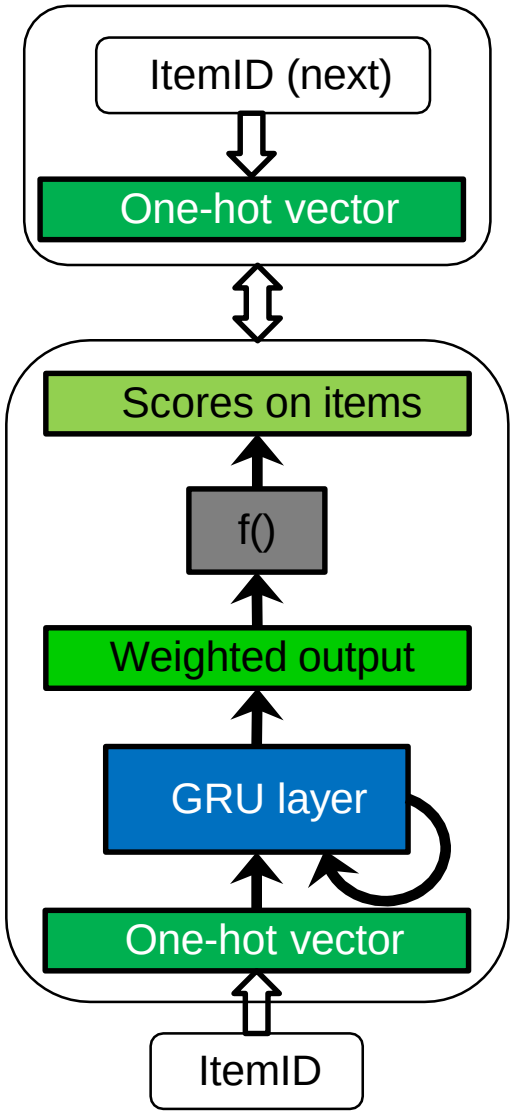


Session-based recommendations

- Sequence of events
  - User identification problem
  - Disjoint sessions (instead of consistent user history)
- Tasks
  - Next click prediction
  - Predicting intent
- Classic algorithms can’t cope with it well
  - Item-to-item recommendations as approximation in live systems
- Area revitalized by RNNs

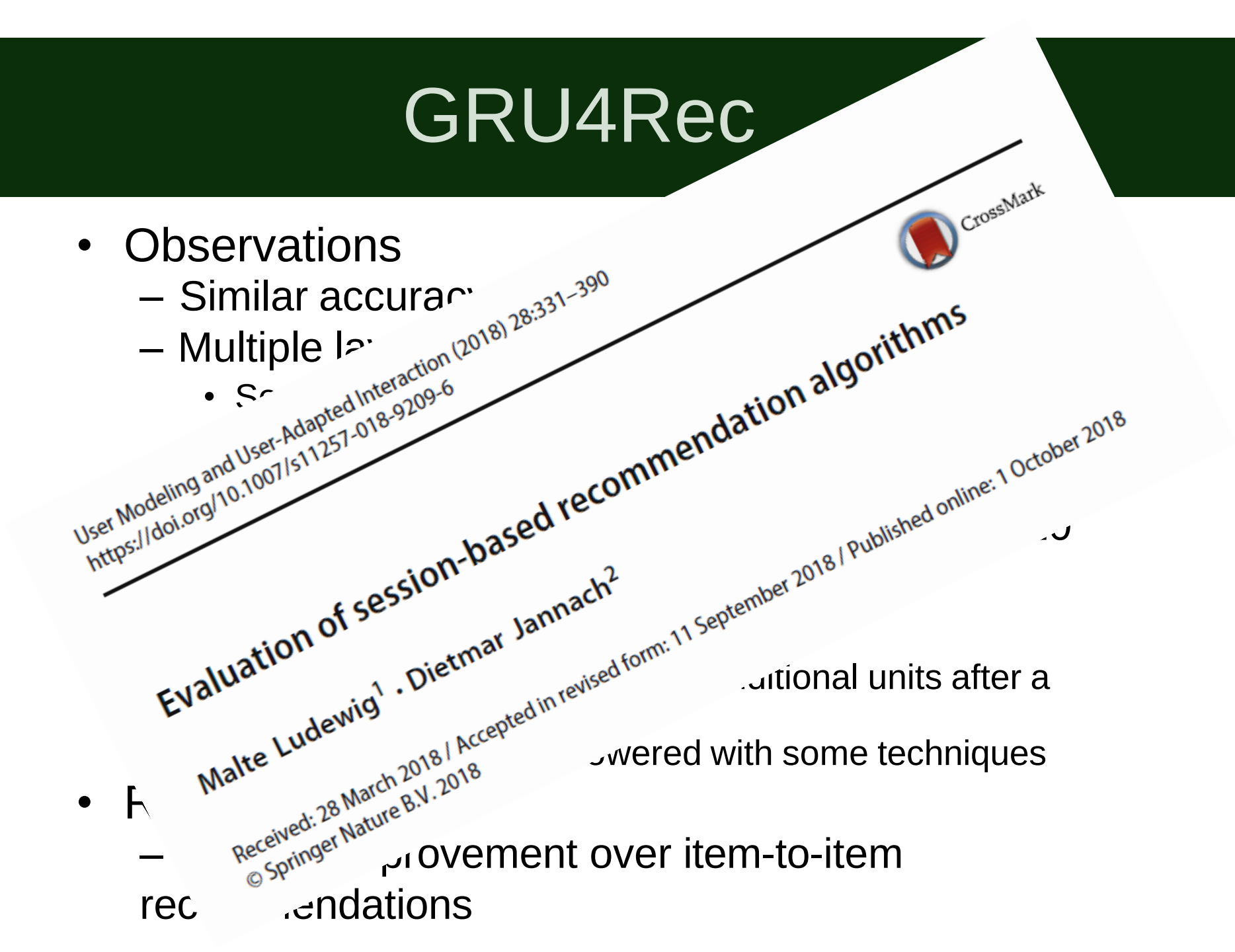
GRU4Rec

- [Hidasi et. al, 2015]
- Network structure
  - Input: one hot encoded item ID
  - Optional embedding layer
  - GRU layer(s)
  - Output: scores over all items
  - Target: the next item in the session
- Adapting GRU to session-based recommendations
  - Sessions of (very) different length & lots of short sessions: session-parallel mini-batching
  - Lots of items (inputs, outputs): sampling on the output
  - The goal is ranking: listwise loss functions on pointwise/pairwise scores



GRU4Rec

- Observations
  - Similar accura
  - Multiple la
    - S



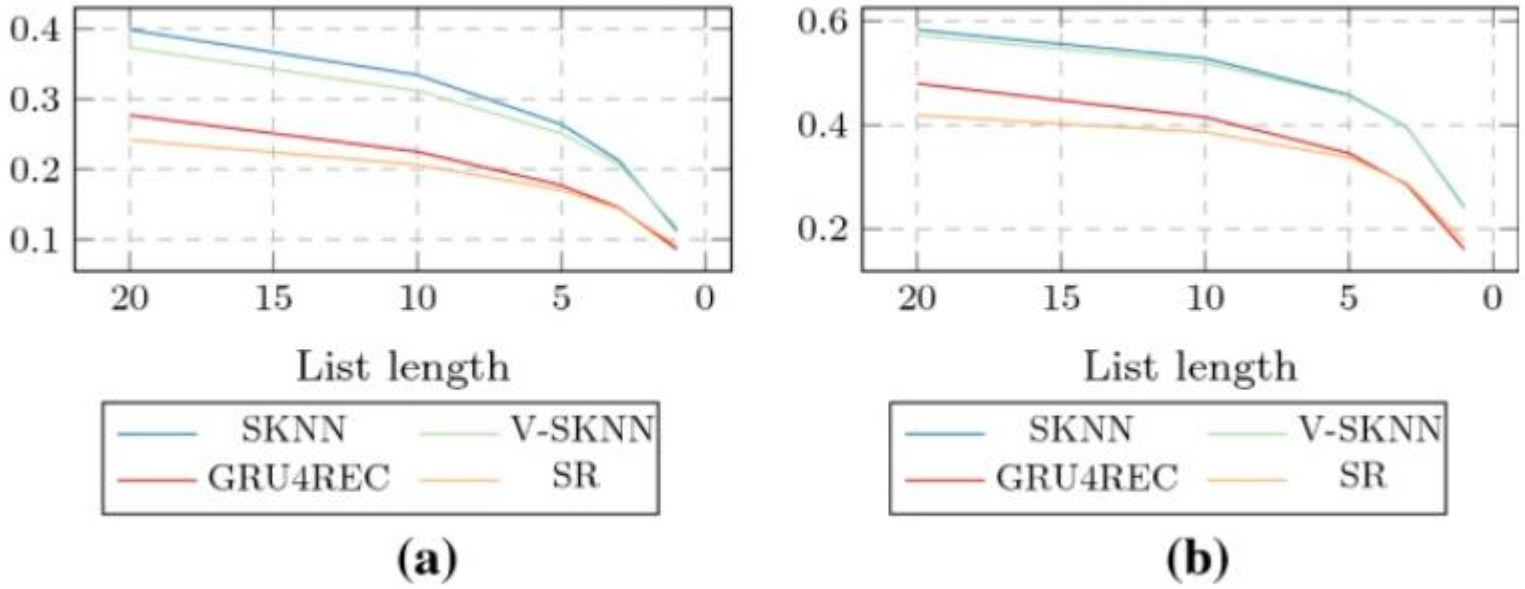
ditional units after a

owered with some techniques

- R
  - provement over item-to-item rec endations

GRU4Rec

Fig. 4



Hit rate (HR) for two of the e-commerce datasets when reducing the recommendation list length from 20 to 1. **a** TMALL, **b** RETAILR

Session KNN variants may outperform deep learning approaches

Additional topics

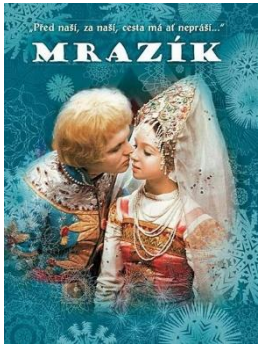
Context-awareness
Explanations and interface
Multi-stakeholder and multi-objective
Fairness and biases

Context-aware RecSys

Context-awareness

Eating alone vs. With friends vs. On a conference

Usual tase vs. The end of December



While in homepage vs. in headset category

RecSys: Context-aware

Key design choice: which contextual axes and values / granularity to consider?

Context-awareness

Simple method for context incorporation into matrix factorization

$$\hat{r}_{uic_1 \ldots c_k} = \boldsymbol{v}_u \cdot \boldsymbol{q}_i + \bar{\imath} + b_u + \sum_{j=1}^{k} b_{g_i j c_j}$$

- $\boldsymbol{v_u}$ and $\boldsymbol{q_i}$ are d dimensional real valued vectors representing the user *u* and the item *i*
- $\bar{\imath}$ is the average of the item *i* ratings
- b_u is a baseline parameter for user u
- b_{gjc} is the baseline of the **contextual condition** c_j (factor *j*) and **genre** g_i of item *i*
  - *We assume that context influences uniformly all the tracks with a given genre*
- If a contextual factor is unknown, i.e., $c_j = 0$, then the corresponding baseline b_{gjc} is set to 0.

Nowadays, more convenient to model through DL

Explanations in RecSys

This item was recommended to you because…

- Improve Transparency / Validity checking / Trust in the system / Persuasiveness / Simplify decision-making / Understand domain,…

- Many different strategies, interfaces, use-cases and source data tried so far

Transparent, Scrutable and Explainable User Models for Personalized Recommendation

Krisztian Balog
Google
London, UK
krisztianb@google.com

Filip Radlinski
Google
London, UK
filiprad@google.com

Shushan Arakelyan*
USC Information Sciences Institute
Marina Del Rey, CA, USA
shushan@isi.edu

- You like movies that are tagged as 'action', especially those that are tagged as 'violent', such as Aliens.
- You like movies that are tagged as 'twist ending', such as A Pure Formality.
- You don't like movies that are tagged as 'adventure', unless they are tagged as 'thriller', such as Twister.
- You like movies that are tagged as 'cheesy', such as Who Framed Roger Rabbit?
- You like movies that are tagged as 'australia', such as Crocodile Dundee II.

Figure 1: Example summary of a user's preferences.

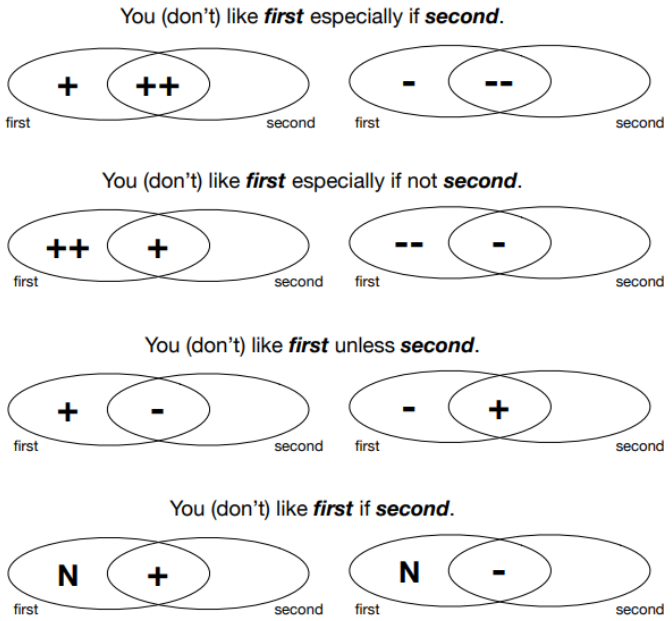


Figure 3: Pairwise set interactions allowed in the user model. *First* is the tag of the left set and *second* is that of the right set. +, - and N indicate positive, negative and neutral average user appeal, with double symbols indicating stronger signals.

Explanations in RecSys

Home › MultiMedia Modeling › Conference paper

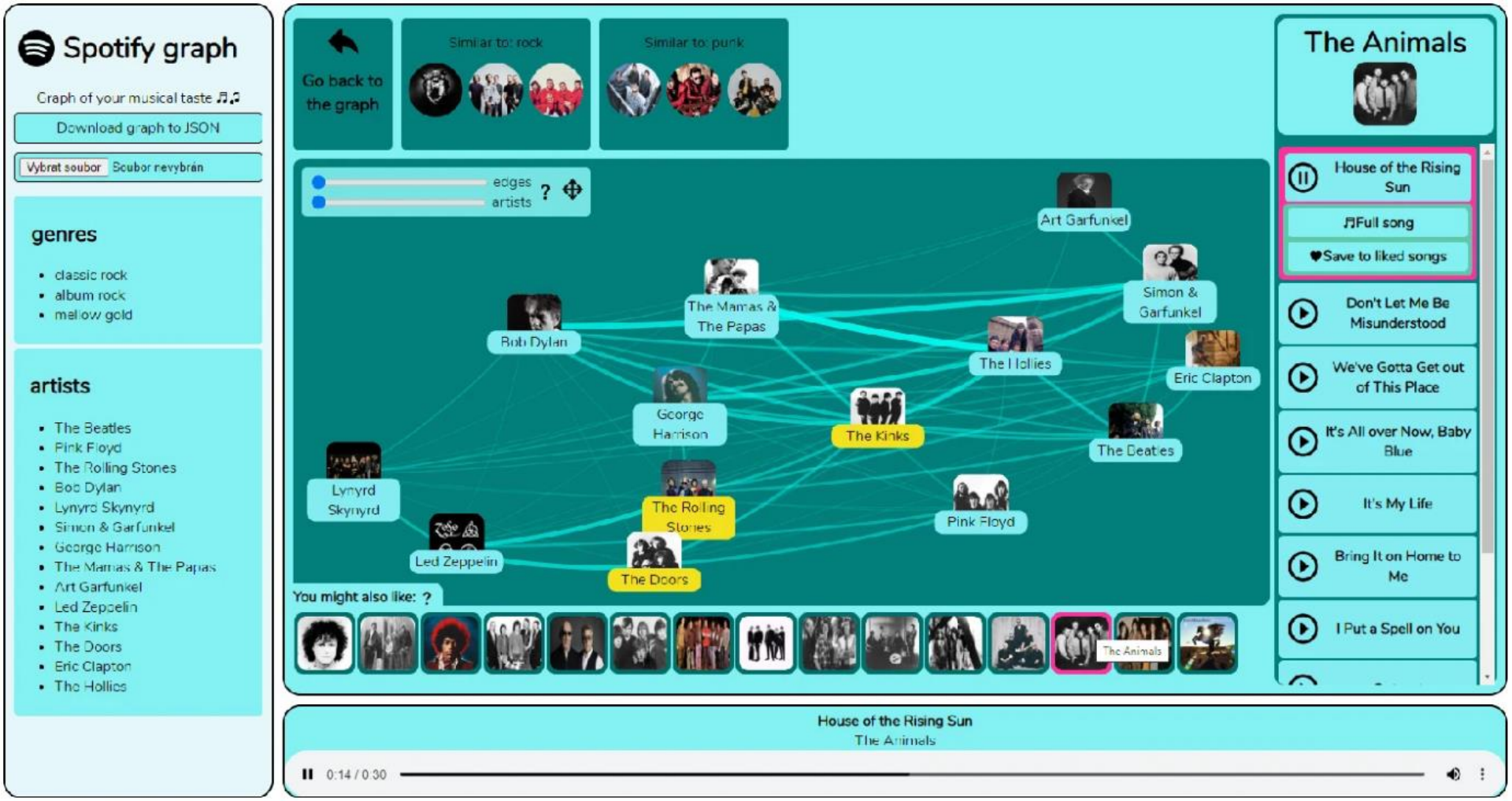
SpotifyGraph: Visualisation of User's Preferences in Music

Conference paper | First Online: 21 January 2021
pp 379–384 | Cite this conference paper

Access provided by National Library of Technology

Download book PDF Download book EPUB

Pavel Gajdusek & Ladislav Peska



Explanations in RecSys

It's Time to Let Go: Stopping Criteria Recommendations in Content-rich Domains

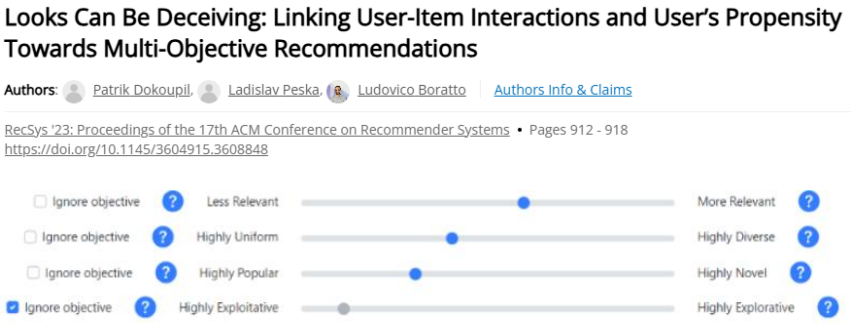
Matej Scerba
m.scerba@seznam.cz
Faculty of Mathematics and Physics, Charles University
Prague, Czech Republic

Ladislav Peska
ladislav.peska@matfyz.cuni.cz
Faculty of Mathematics and Physics, Charles University
Prague, Czech Republic

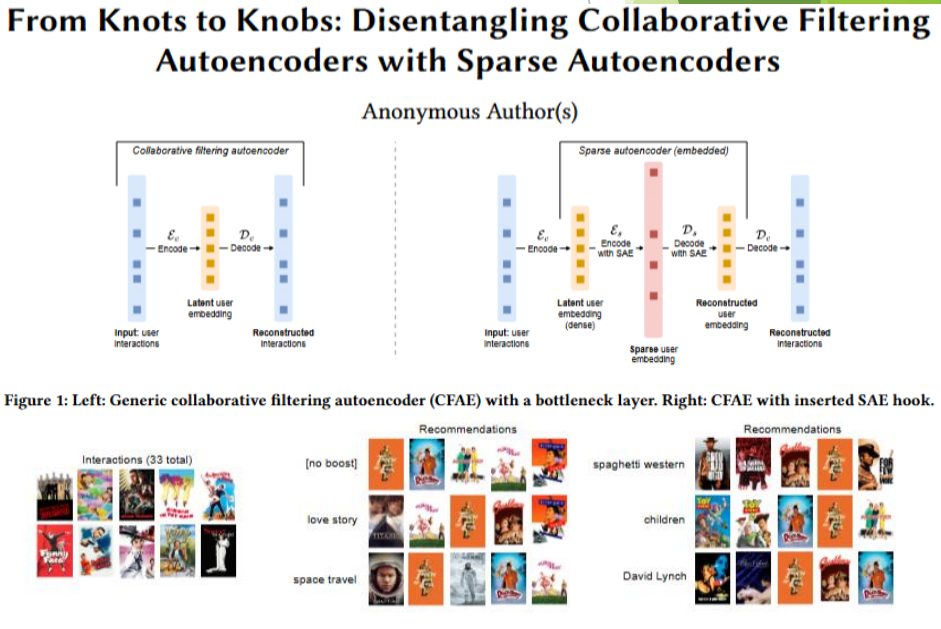
User Control in RecSys

Normally, it is enough if the system provides the user with recommendations

- But sometimes, the user has certain specific needs / wishes
  - Joint models for search and recommendation
  - Conversational recommender systems
  - Adding controlability/steering for recommendations



- Set high level objectives; re-ranking

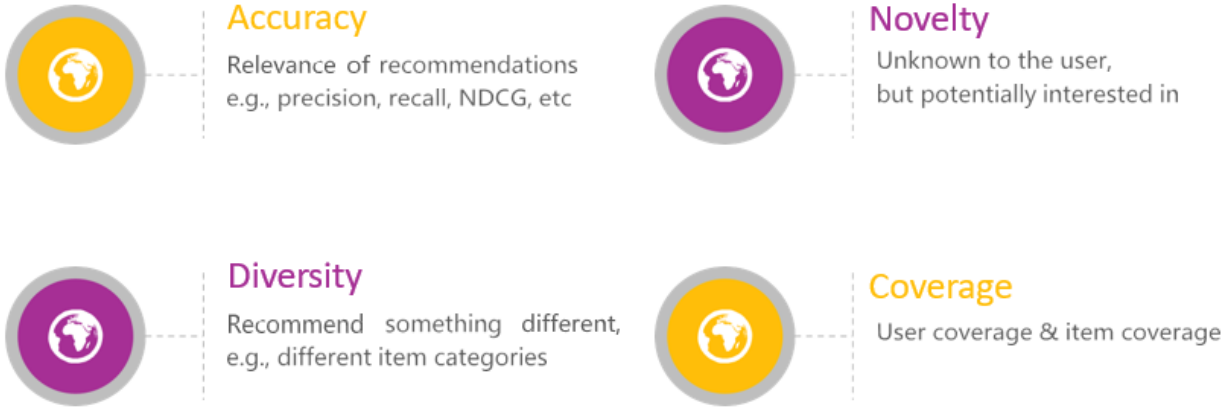


- Finer-grained interests („levers“), in-process

Multi-stakeholder RecSys
Multi-objective RecSys

- You are not alone in the system, others have their needs too
- You may have multiple needs hard to sum into a single objective

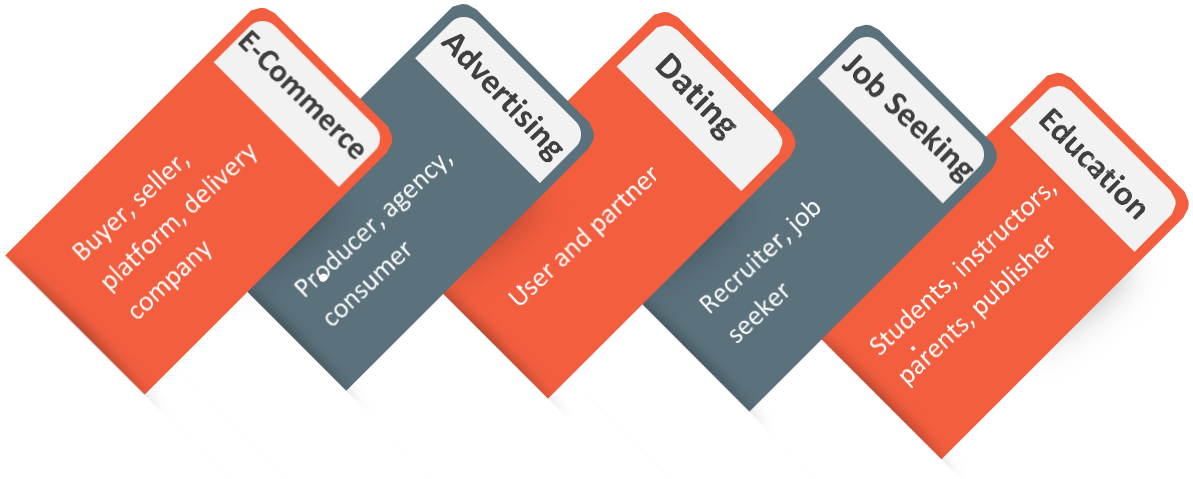
- Objective definitions in this context
  - Optimize more metrics in addition to accuracy



Kaminskas, M., & Bridge, D. (2016). Diversity, serendipity, novelty, and coverage: a survey and empirical analysis of beyond-accuracy objectives in recommender systems. ACM TIIS, 7(1), 1-42.

Multi-Stakeholder RecSys

- Why we need MOO in this context
  - The end user is not the only stakeholder



  - RecSys should be built by considering the item utility from the perspective of different stakeholders

Multi-Stakeholder RecSys

- Objective definitions
  - It varies from domains to domains
  - For each stakeholder, there's at least one objective
    - E-Commerce or Marketplace
      - Buyer: user preferences on items, budget
      - Seller: profits
      - Platform: commission fees
      - Delivery company: costs and profits
    - Job seeking
      - Job seeker: user preferences
      - Recruiter: talent requirements

Bias and Fairness Issues

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Tutorial on Fairness of Machine Learning in Recommender Systems

Yunqi Li
Rutgers University
yunqi.li@rutgers.edu

Yingqiang Ge
Rutgers University
yingqiang.ge@rutgers.edu

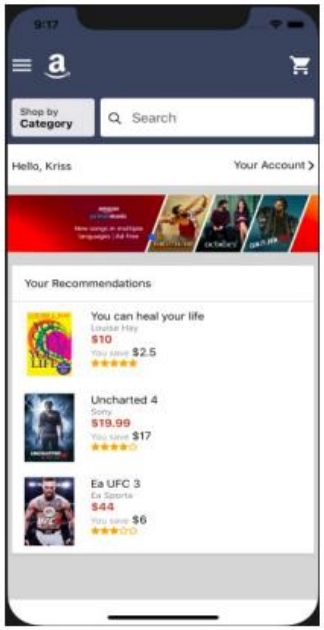
Yongfeng Zhang
Rutgers University
yongfeng.zhang@rutgers.edu

Tutorial from: https://fairness-tutorial.github.io/

Fairness Issues

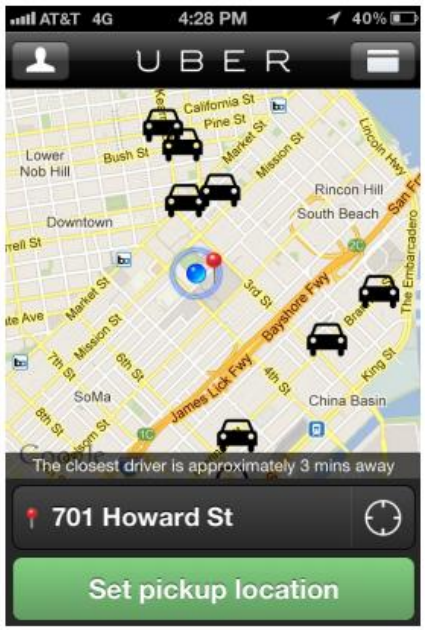
RUTGERS

Why Fairness in RecSys? Resources Could be Limited

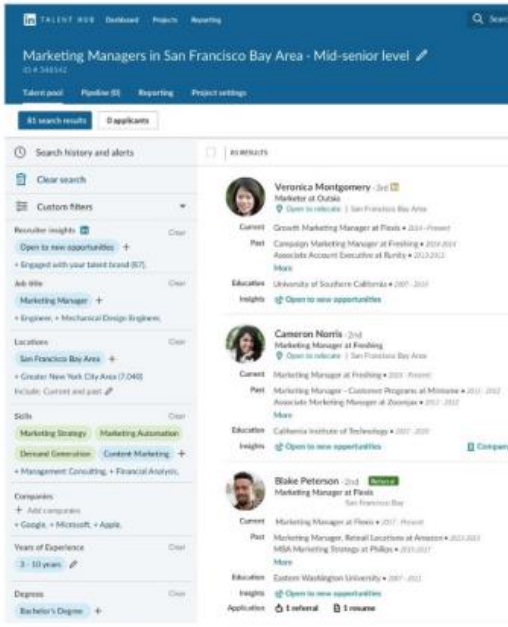


Recommendation slot positions are limited, which producers' items should be recommended and get the exposure opportunity to users?

User attention is a limited resource, whose twite should get exposure on the timeline?



Passengers are limited, which driver should get the task and make money?



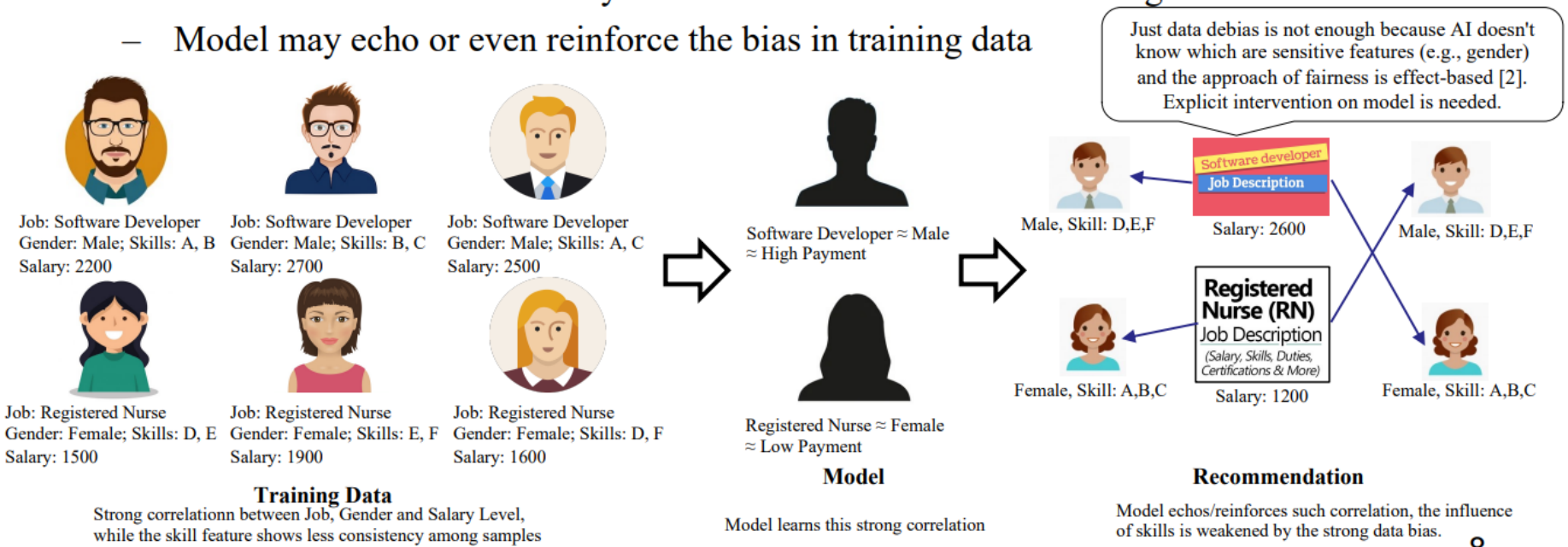
Interview opportunities are limited, which candidate(s) should get an interview opportunity?

Fairness Issues

RUTGERS

Why Fairness in RecSys? Data Could be Biased

- Most RecSys models are ML models trained on some training data
  - Training data may encode social bias
  - Recommendation models may learn "shotcuts" for decision making
  - Model may echo or even reinforce the bias in training data



Fairness Issues

RUTGERS

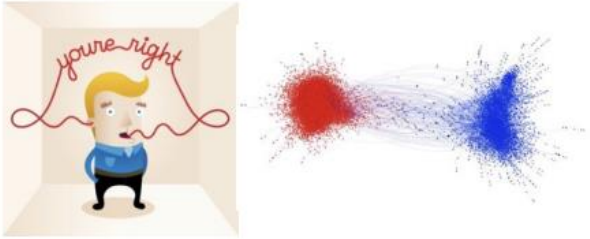
Potential Consequences of Unfairness in RecSys

Information Asymmetry

Knowing a piece of valuable information (e.g., a job opportunity) could change one's life

Matthew Effect

Advantaged users, items, or groups get further propagated by recommendations, sometimes not because of their good quality but because the recommendation model is dominated by their data



Echo Chambers

Unfair, undiversified exposure of news, messages, tweets, etc. may create echo chamber. Makes it difficult to explore new ideas and opinions different from one's own. Makes people feel like the whole world thinks the same way as they think. May even reinforce someone's extremist ideas

Fairness Issues

RUTGERS

Fairness in RecSys: Beyond Ethics, a Utilitarian Perspective

- RecSys platforms should consider fairness for the sake of themselves
  - Not only for legal regulations, but for the sustainable/long-term development of the platform

An e-commerce example

Big retailors vs. Small retailors

If products from small retailors (e.g., family workshops) do not have fair exposure opportunity by e-commerce recommender system, they may eventually leave since they cannot survive in the platform, making the platform unsustainable.

A social network example

Star accounts vs. Grassroot accounts

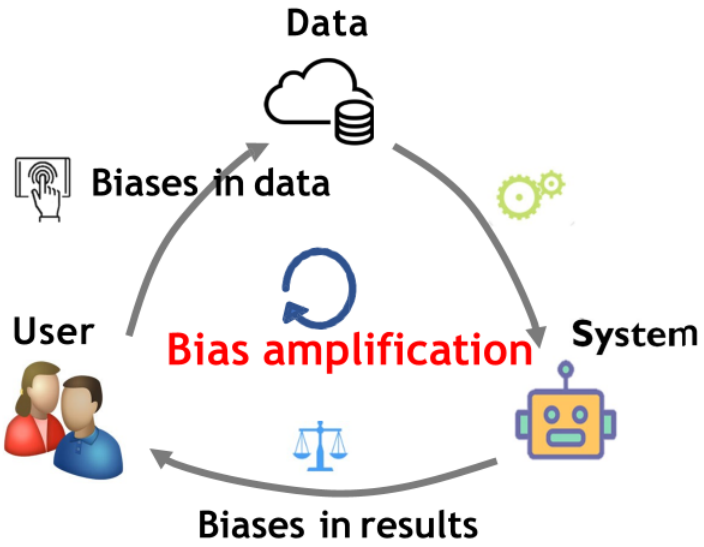
Videos from famous accounts (e.g., a film star) usually get more attention, but if videos created by grassroot accounts do not have any exposure opportunity to users, they may leave the platform, making the platform's contents less diversified and even boring.

Cascade of Biases

- **Matthew Effect: Bias + Loop**

- Biases amplification along the loop:
  - Biases would be circled back into the collected data
  - Resulting in "Matthew effect" issue: the rich gets richer
  - Damaging the ecosystem of RS

Managable through exploration promotion



- **Selection Bias**

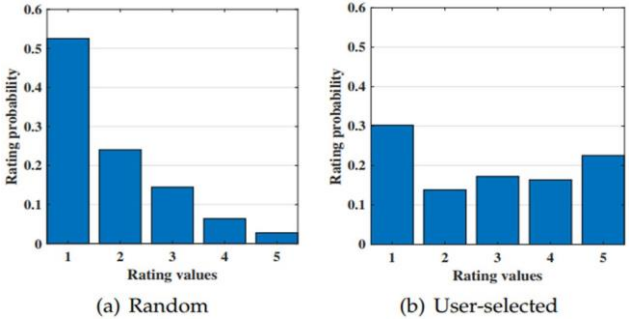
  - Definition: *Selection bias happens in explicit feedback data as users are free to choose which items to rate, so that the observed ratings are not a representative sample of all ratings.*

3	4	2	5
1	3	2	5
2	3	4	4

Selection bias

3	4		5
	3		5
	3	4	4

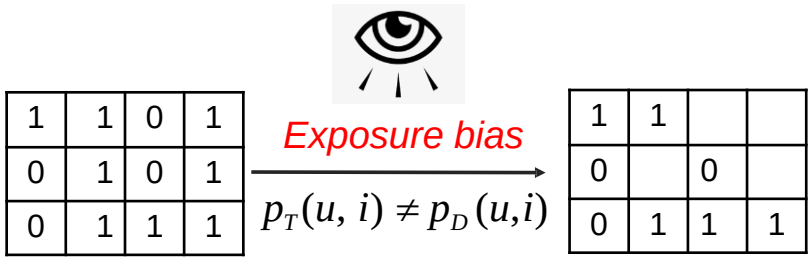
$$p_T(u, i) \neq p_D(u, i)$$



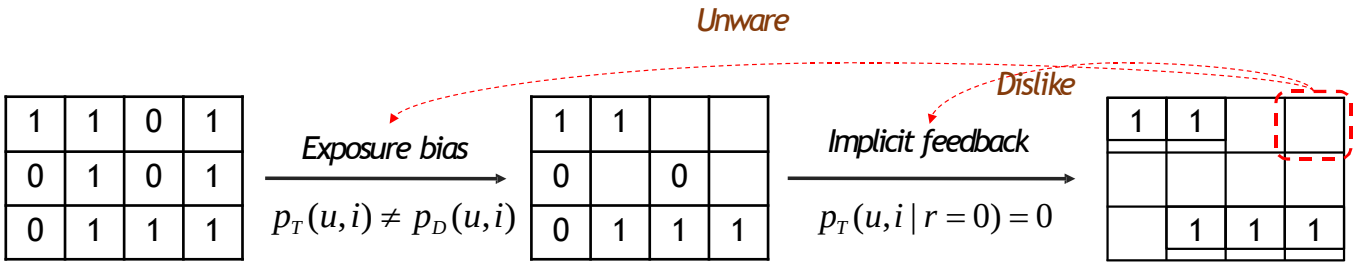
1Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as Treatments: Debiasing Learning and Evaluation. In ICML.
2B. M. Marlin, R. S. Zemel, S. Roweis, and M. Slaney, "Collaborative filtering and the missing at random assumption," in UAI, 2007

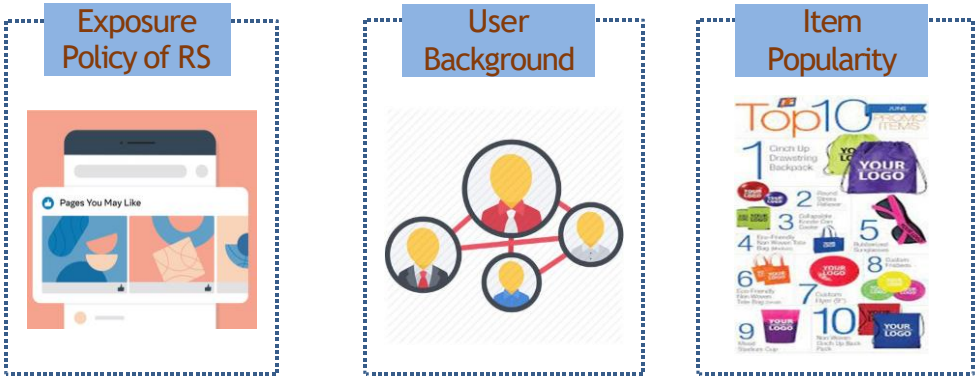
• Exposure Bias

- Definition: *Exposure bias happens in implicit feedback data as users are only exposed to a part of specific items.*
- Explanation: A user generates behaviors on exposed items, making the observed user-item distribution $p_T(u, i)$ deviate from the ideal one $p_D(u, i)$.



- **Exposure Bias**





- **Conformity Bias**

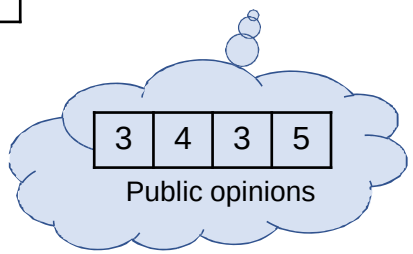
  - Definition: *Conformity bias happens as users tend to behave similarly to the others in a group, even if doing so goes against their own judgment.*

3	4		5
	3		4
	3	4	3

Conformity bias

$p_T(r|u,i) \neq p_D(r \mid u,i)$

3	4		5
	3		5
	3	3	4

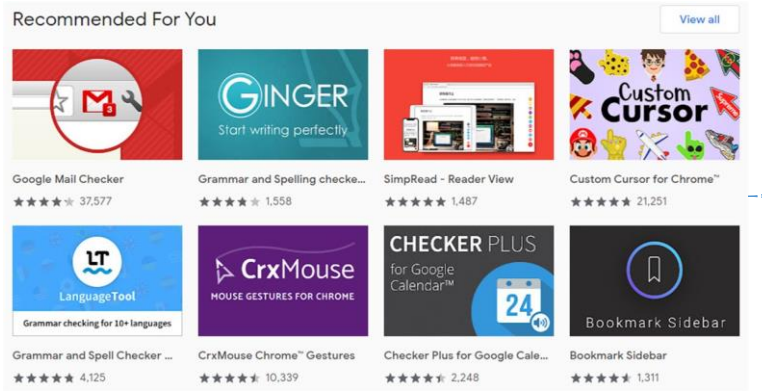


- **Position Bias (Presentation Bias more generally)**
  - Definition: *Position bias happens as users tend to interact with items in higher position of the recommendation list.*

1	1	0	1
0	1	0	1
0	1	1	1

Position bias →

1	1	1	
		1	
		1	1



$p_T(u, i) \neq p_D(u,i)$

User exposure will be affected by the position

$p_T(r \mid u,i) \neq p_D(r \mid u,i)$

User judgments also will be affected by the position

De-biasing Off-line Evaluation

- https://dl.acm.org/doi/pdf/10.1145/3240323.3240355

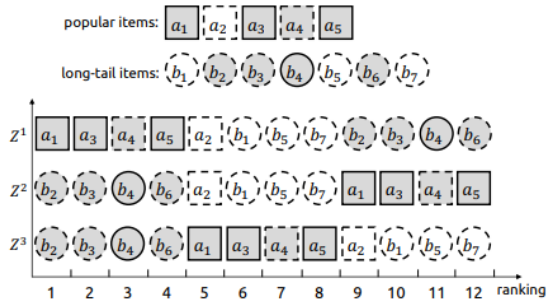


Figure 1: A hypothetical example to illustrate the evaluation bias that results from use of the AOA evaluator. Three recommenders generated distinct lists of recommendations, Z^1, Z^2 and Z^3, for the same user. Among the shaded items that were preferred by the user, the ones with a solid border were observed by recommenders. The performance was measured by DCG, and the results are presented in Table 1.

Table 1: The true and estimated DCG values for three recommenders in Fig. 1. $R(\hat{Z})$ denotes the ground truth, and $\hat{R}_{\text{AOA}}(\hat{Z})$ denotes the AOA estimations. The AOA estimator outputs larger values when popular items are ranked higher.

Estimator	Z^1	Z^2	Z^3
$R(\hat{Z})$	0.463	0.463	0.494
$\hat{R}_{\text{AOA}}(\hat{Z})$	0.585	0.340	0.390

3.1 Average-over-all (AOA) evaluator

In prior literature, $R(\hat{Z})$ was estimated by taking the average over all observed user feedback $\mathcal{S}_u^*$:

$$\hat{R}_{\text{AOA}}(\hat{Z}) = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{1}{|\mathcal{S}_u^*|}\sum_{i\in\mathcal{S}_u^*} c(\hat{Z}_{u,i}) = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{1}{\sum_{i\in\mathcal{S}_u} O_{u,i}}\sum_{i\in\mathcal{S}_u} c(\hat{Z}_{u,i})\cdot O_{u,i} \quad (6)$$

nDCG, AUC, MAP,...

3.2 Unbiased evaluator

To conduct unbiased evaluation of biased observations, we leverage the IPS framework [16, 22] that weights each observation with the inverse of its propensity, where the term *propensity* refers to the tendency or the likelihood of an event happening. The intuition is to down-weight the commonly observed interactions, while up-weighting the rare ones. In the context of this paper, the probability $P_{u,i}$ is treated as the pointwise propensity score. Therefore, the IPS unbiased evaluator is defined as follows:

$$\hat{R}_{\text{IPS}}(\hat{Z}|P) = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{1}{|\mathcal{S}_u|}\sum_{i\in\mathcal{S}_u^*}\frac{c(\hat{Z}_{u,i})}{P_{u,i}} = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{1}{|\mathcal{S}_u|}\sum_{i\in\mathcal{S}_u}\frac{c(\hat{Z}_{u,i})}{P_{u,i}}\cdot O_{u,i} \quad (7)$$

Propensity score

$$\hat{P}_{*,i} \propto (n_i^*)^\gamma \cdot n_i, \quad (13$$

where $n_i = \sum_{u\in\mathcal{U}} \mathbf{1}[i \in \mathcal{S}_u]$ and $n_i^* = \sum_{u\in\mathcal{U}, i\in\mathcal{S}_u^*} O_{*,i}$.

However, empirically, n_i is not directly observable. To addres this problem, we observe that n_i^* is sampled from a binomial distribution[4] parameterized by n_i, that is, $n_i^* \sim \mathcal{B}(n_i, P_{*,i})$. Therefore, relationship between n_i and n_i^* can be built by bridging the genera tive model (eqn. 13) with the following unbiased estimator:

$$\hat{P}_{*,i} = \frac{n_i^*}{n_i} \propto (n_i^*)^\gamma \cdot n_i \quad (14$$

Therefore, $n_i \propto (n_i^*)^{\frac{1-\gamma}{2}}$. We use this as a replacement for the unobserved n_i in eqn. 13, which results in an unbiased $\hat{P}_{*,i}$ estimato that is determined by only the empirical counts of items:

$$\hat{P}_{*,i} \propto (n_i^*)^{\left(\frac{\gamma+1}{2}\right)} \quad (15$$

Outlook

More complex task-specific architectures (bit boring)
Scalability issues (sparse EASE variants)
User controll + Large Language Models (Conversational RecSys)
Evaluation issues (evergreen)
System -> Ecosystem (normal in industry, not so much in papers)