DATA SCIENCE SUMMIT ML EDITION

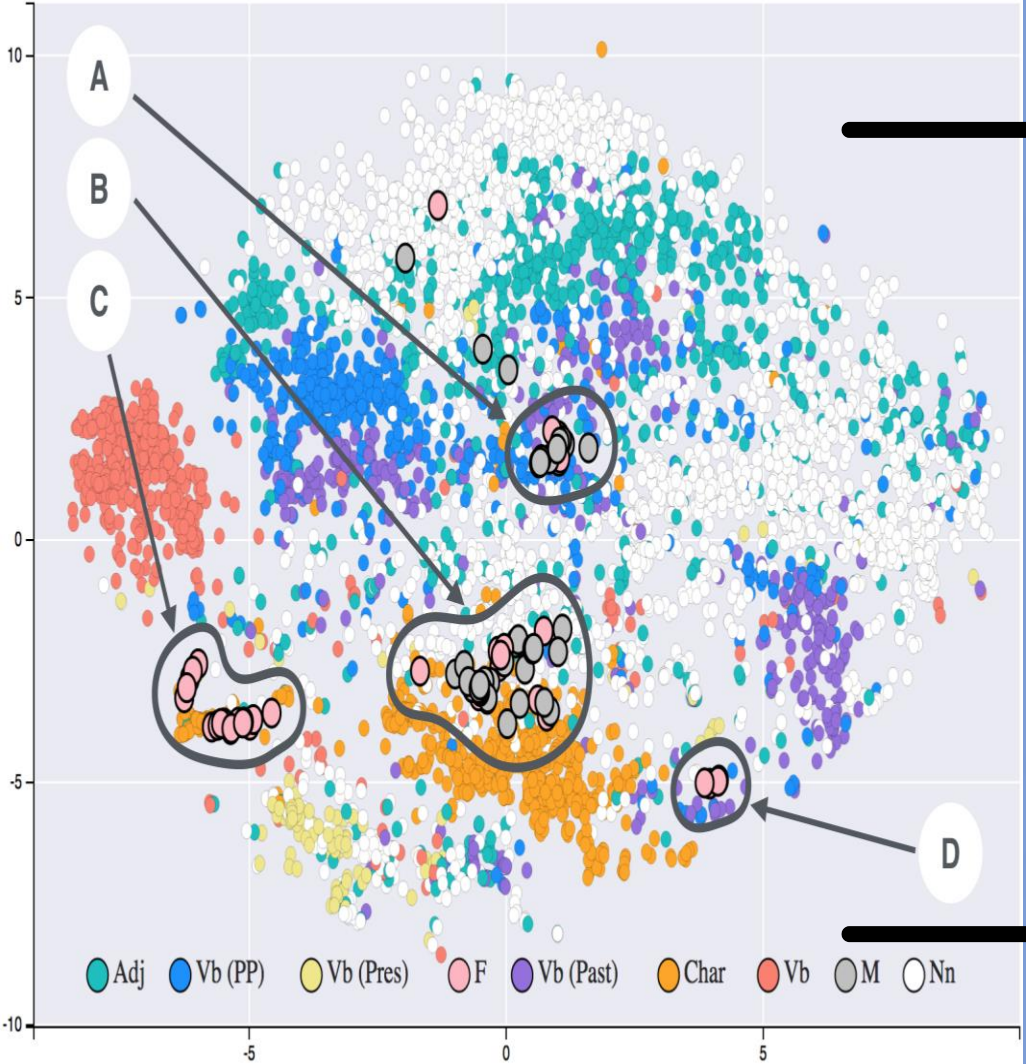
ONE CONCEPT to rule them all:

Neural network embeddings use cases beyond NLP

Filip Wójcik
Senior Data Scientist at Capgemini
PhD Candidate

dr Filip Wójcik
Wroclaw University
of Economics and Business

1. **Embeddings:** what they are?
2. **NLP:** a typical use cases
3. **Training:** algorithms for embeddings
4. **Recommendations:** embeddings in a collaborative filtering
5. **Tabular data:** embeddings helping in classification problems
6. **Frequent items:** embeddings as a tool for market basket problems



01

Embeddings

What they are?

Embeddings: what are they?

1. In general – embedding is a **mapping from a discrete variable to a vector of continuous numbers**.

2. More importantly – the dimensionality of these vectors is **much lower** than e.g. traditional one-hot encoding.

3. Weights used to build such a vector are then used to train the network.

4. Following one of the definitions from the literature, we can say that:

“

> *each core feature is embedded into a d dimensional space, and represented as a vector in that space. The dimension d is usually much smaller than the number of features, i.e., each item in a vocabulary (…). the embeddings (the vector representation of each core feature) are treated as parameters of the network, and are trained like the other parameters of the function f.*

Goldberg, Y., 2017. Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, *10*(1), p. 90

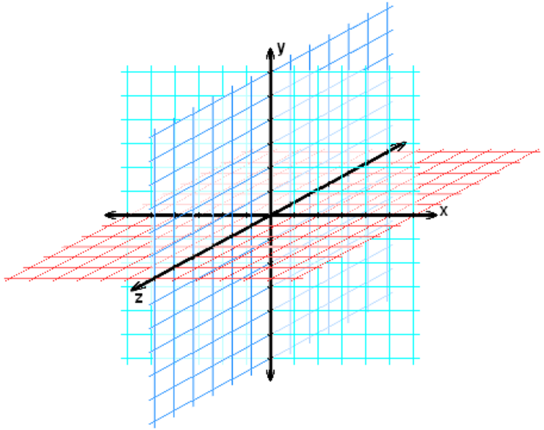
dr Filip Wójcik
Wroclaw University
of Economics and Business

Embeddings: what are they?

5. Additionally, an important property of embeddings is the fact, that each of the newly created dimensions can **carry and emphasize different aspect of the original feature**.

6. Therefore – visualizations can potentially capture such a meaning and allow **easier interpretation** of embeddings.

7. From the computational perspective – embeddings reduce the dimensionality, which is good for almost any machine learning model.



dr Filip Wójcik
Wroclaw University
of Economics and Business

Embeddings: what they are?

dr Filip Wójcik
Wroclaw University
of Economics and Business

One-hot encoding

- Dimensionality is the same as original features
- Features are independent from each other
- No way to judge which features appear in a similar context
- No easy way to visualize features

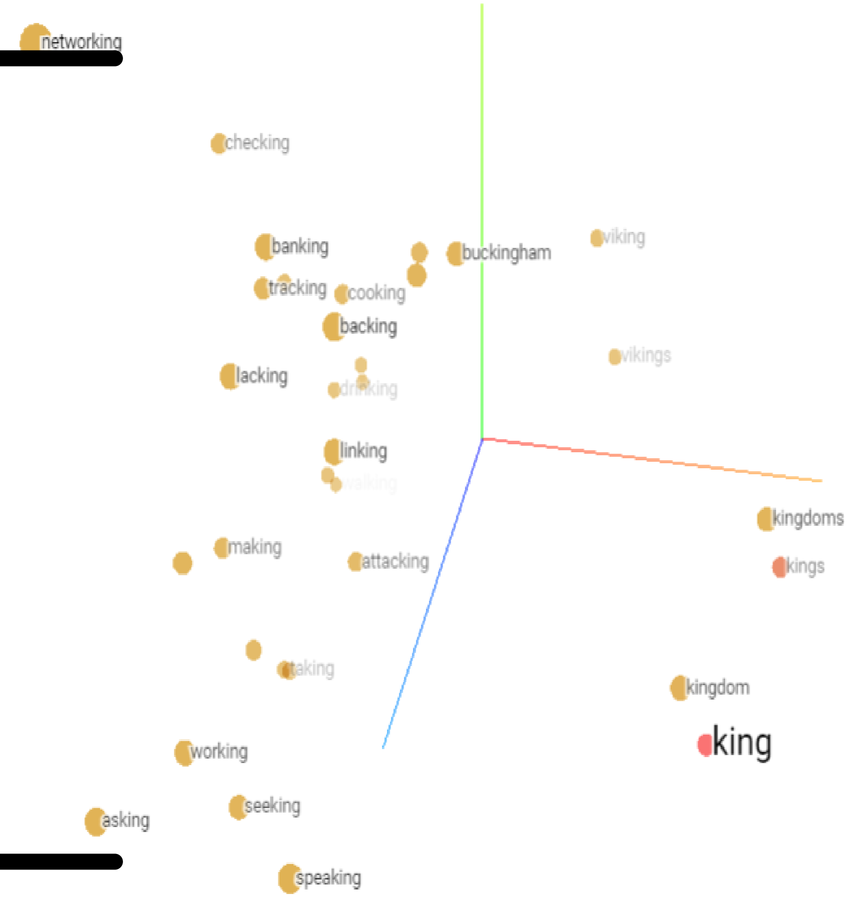
Embeddings

- Dimensionality of each feature representation is d where $d \ll \dim(feature)$
- Features sharing similar properties will be represented in a similar way
- Maintains the distance and the notion of context between features
- Can be visualized as points in low-dimensional space

02

NLP

A typical use cases



NLP: a typical use cases

1. Typical NLP use cases include **low-dimensionality representation of word vocabulary.**

2. The procedure when working with a text (classification/topic recognition/etc.) usually looks as follows:

   a) Limit allowed words (vocabulary) to some number

   b) Filter stop-words (specific for a given language) and keep the remaining ones

   c) Perform stemming/lemmatization

   d) Reduce the size of vocabulary by performing embeddings in the context

3. Depending on the specific task – resulting embeddings might help to **find words sharing similar properties**:

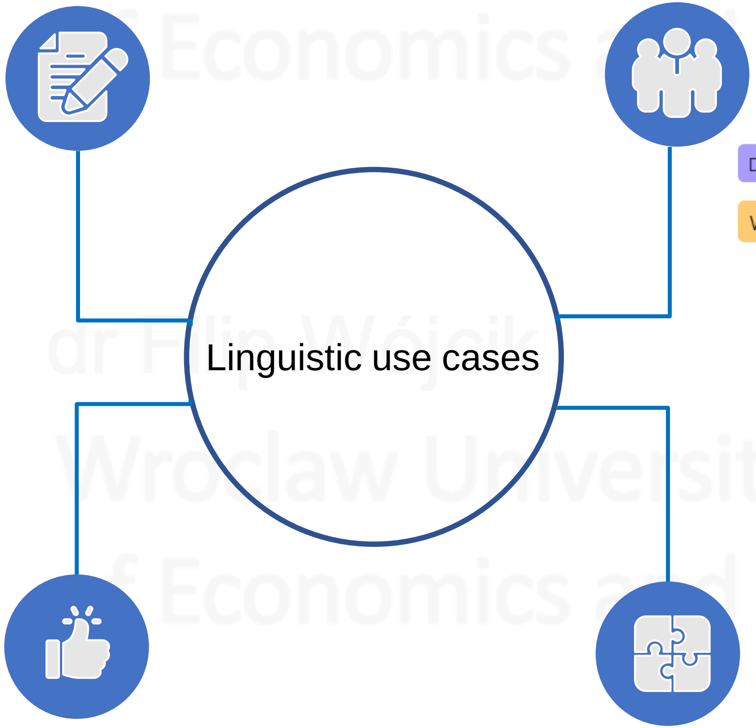
   a) For classification – words belonging to the same class will be close together

   b) For entity recognition – words describing entity will be close together

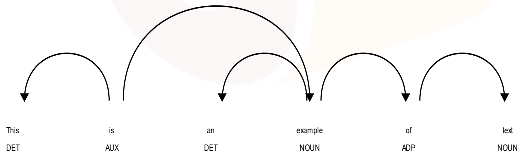
dr Filip Wójcik
Wroclaw University
of Economics and Business

NLP: a typical use cases

dr Filip Wójcik
Wroclaw University of Economics and Business



Part of speech (POS) tagging

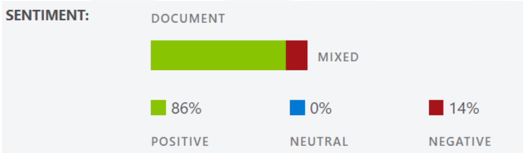


Named entity recognition (NER)

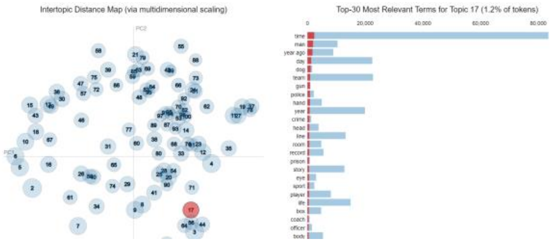
Donald Trump **PERSON** used to be a president of United States **GPE** .

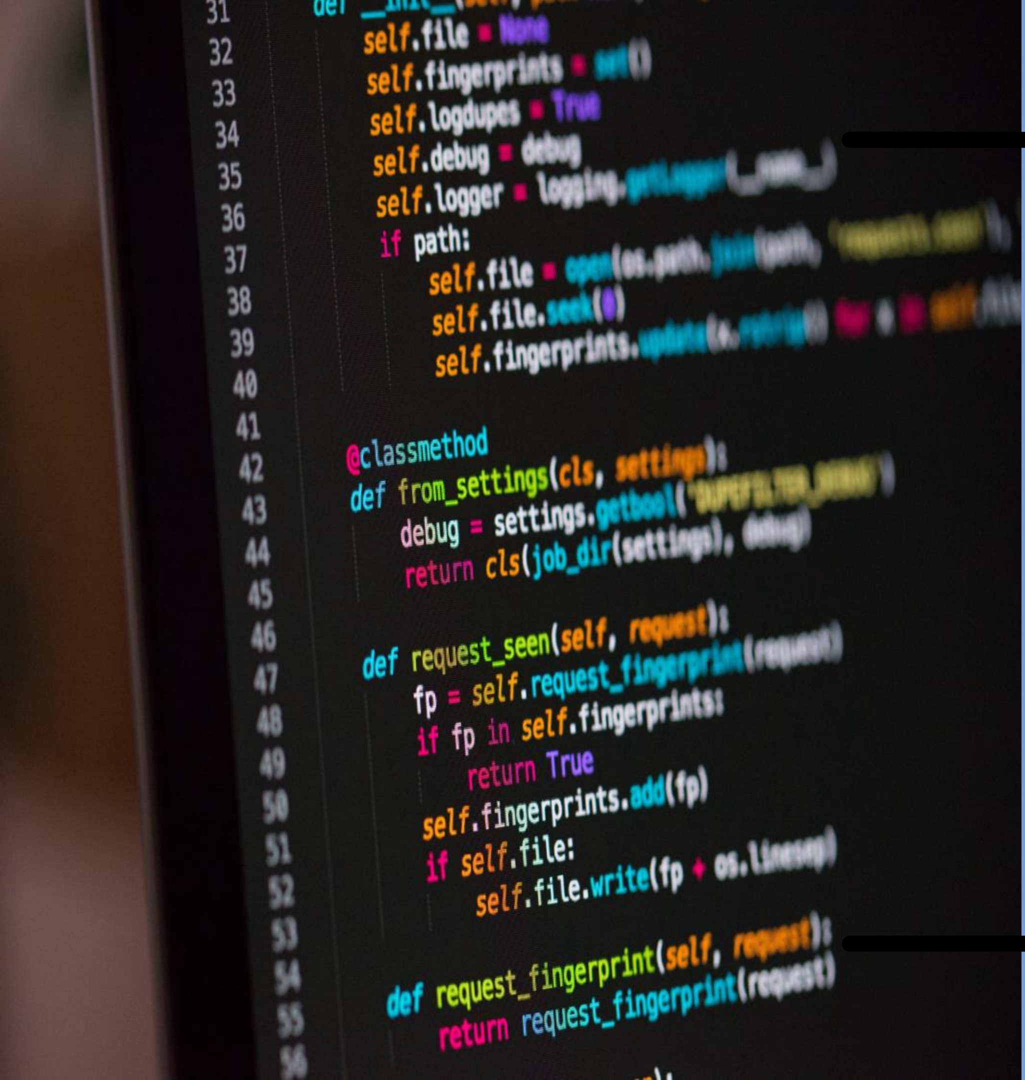
Washington **GPE** is the capital city.

Sentiment analysis



Topic recognition





03

Algorithms

for training embeddings

Algorithms for embeddings

1. There are several approaches on how to train embedding weights - most of the algorithms are designed primarily to **train embeddings on their own.**

2. There are different conceptual approaches to embedding training, e.g.:

   a) **Based on classification -** words appearing in the same context are treated as "positive" class, other words - as "negative" class

   b) **Based on the frequency** of word co-occurence

3. In each case, an important issue is to define a "context", or in other words: "closeness".

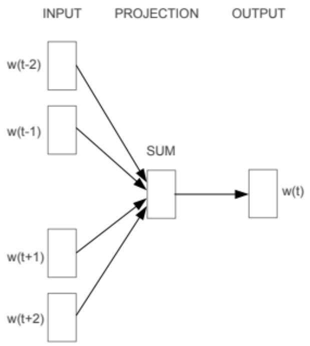
4. Especially problematic is a sampling of "negative" (not-related) words

dr Filip Wójcik
Wroclaw University
of Economics and Business

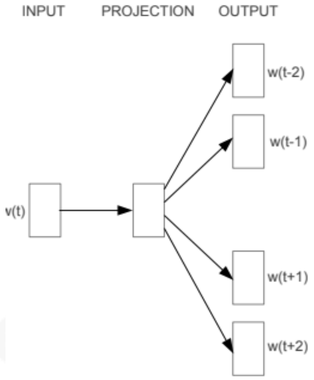
Algorithms for embeddings

Word2Vec

1. Based on the **classification** approach.
2. **Predict** a given word, from its context.
3. Learned weights are embeddings.
4. First - requires **to one-hot encode (can be sparse)** of all vocabulary
5. Two main prediction directions:
   a) **CBOW** (continuous bag of words) - predict "central" word from surrounding
   a) **Skip-gram model** - predict surrounding words from "central" word



CBOW



Skip-gram

Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

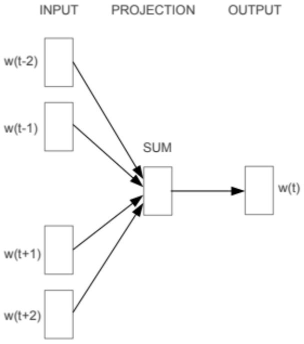
dr Filip Wójcik
Wroclaw University of Economics and Business

Algorithms for embeddings

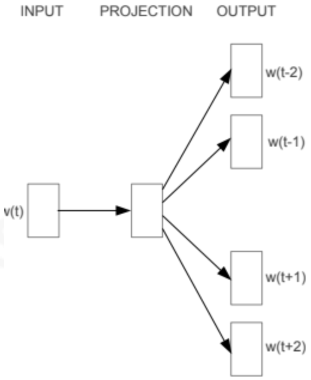
Word2Vec

The quick brown fox **jumps** over the lazy dog

The quick brown fox **jumps** over the lazy dog



CBOW



Skip-gram

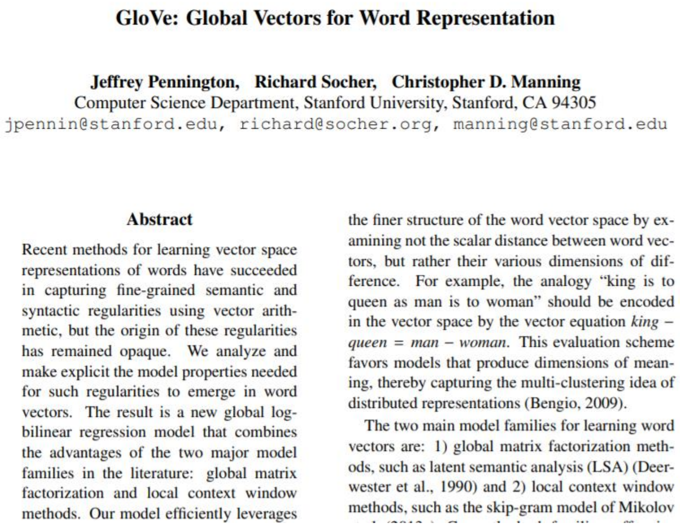
Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

dr Filip Wójcik
Wroclaw University
of Economics and Business

Algorithms for embeddings

GloVe

1. Glove is based on **the frequency of co-occurrence** matrix decomposition.
2. It requires first to build VxV **matrix of vocabulary words co-occurrence**, where each cell v_{ij} represents a number of co-occurrences of the word i and j in the defined context.
3. Then a co-occurrence matrix **should be transformed** to get a matrix of occurence probability ratio.
4. In reality - this **matrix is approximated by a neural network** performing decomposition and reconstruction (like in autoencoders).



GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning
Computer Science Department, Stanford University, Stanford, CA 94305
jpennin@stanford.edu, richard@socher.org, manning@stanford.edu

Abstract

Recent methods for learning vector space representations of words have succeeded in capturing fine-grained semantic and syntactic regularities using vector arithmetic, but the origin of these regularities has remained opaque. We analyze and make explicit the model properties needed for such regularities to emerge in word vectors. The result is a new global log-bilinear regression model that combines the advantages of the two major model families in the literature: global matrix factorization and local context window methods. Our model efficiently leverages

the finer structure of the word vector space by examining not the scalar distance between word vectors, but rather their various dimensions of difference. For example, the analogy "king is to queen as man is to woman" should be encoded in the vector space by the vector equation *king – queen = man – woman*. This evaluation scheme favors models that produce dimensions of meaning, thereby capturing the multi-clustering idea of distributed representations (Bengio, 2009).

The two main model families for learning word vectors are: 1) global matrix factorization methods, such as latent semantic analysis (LSA) (Deerwester et al., 1990) and 2) local context window methods, such as the skip-gram model of Mikolov

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).

dr Filip Wójcik Wroclaw University of Economics and Business

Algorithms for embeddings

GloVe

Sent. 1.: water changes into gas or steam
Sent. 2: ice changes into steam

	water	changes	into	gas	or	steam
water	1	1	1	1	1	1
changes	1	2	2	1	1	2
into	1	2	2	1	1	2
gas	1	1	1	1	1	1
or	1	1	1	1	1	1
steam	1	2	2	1	1	2

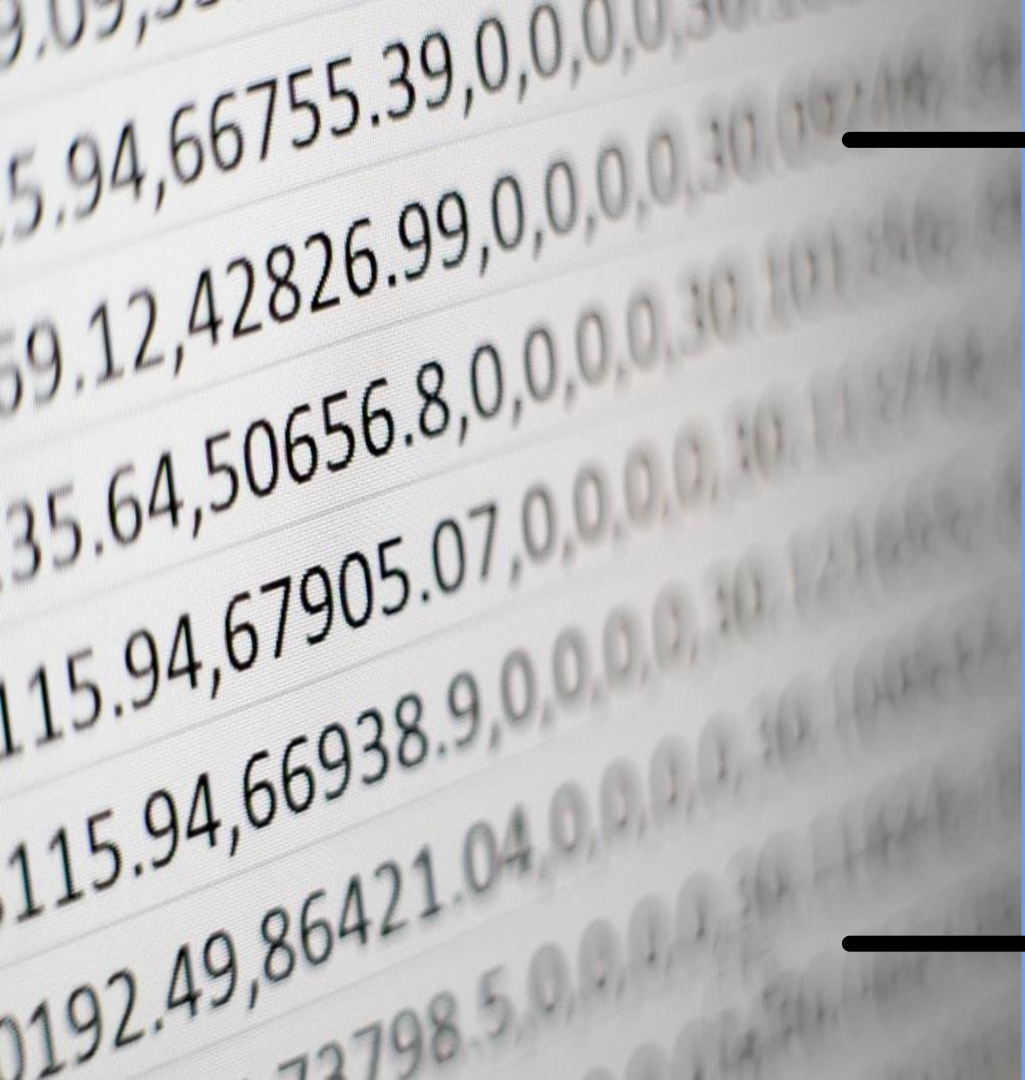
Matrix of raw co-occurence X

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

$\approx$

$$J = \sum_{i,j=1}^{V} f(X_{ij})\left(w_i^T \cdot w_j + b_i + b_j - \log X_{ij}\right)^2$$

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).



04

Tabular data

Embeddings use in tabular classification

Tabular data

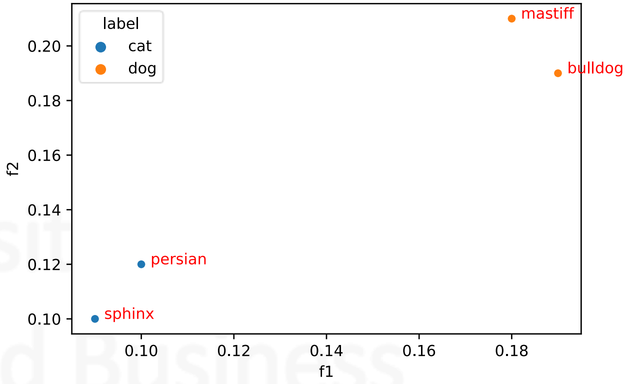
dr Filip Wójcik
Wroclaw University
of Economics and Business

1. Embeddings can be used instead of one-hot encoding as a part of the classification process.
2. They should be included **right after the input layer.**
3. After full classification training loop - embeddings will be a **supervised discrete data representation.**
4. Embedding **dimensionality** can be much lower than one-hot, with a possibility to visualize it.
5. Such embeddings can be interpreter directly, unlike the One-hot-encoding.

Weight	Breed	…	Class
3 kg	Persian		Cat
2 kg	Sphinx		Cat
8 kg	French bulldog		Dog
50 kg	Mastiff		Dog

embedding

Weight	Breed feature 1	Breed feature 2	Class
3 kg	0.1	0.12	Cat
2 kg	0.09	0.1	Cat
8 kg	0.19	0.19	Dog
50 kg	0.18	0.21	Dog



Tabular data: example

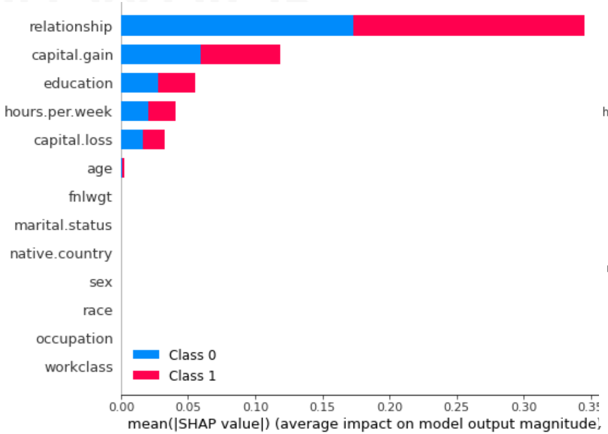
DATA — MODEL — RESULTS

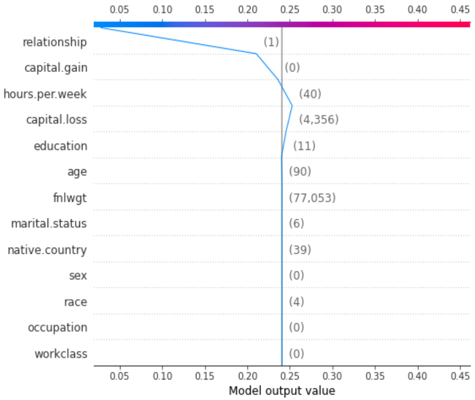
Adult census dataset – binary classification (earnings: >50K$, <=50K$) with demographic features

age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.cou
90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0	4356	40	United-S
82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18	United-S
66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0	4356	40	United-S
54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900	40	United-S
41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900	40	United-S

Using „classic ML” models we might suspect, that some attributes help in predicting classes.

- How exactly are feature values related to classes
- Can we group certain feture values together?
- Can we visualize it?





dr Filip Wójcik
Wrocław University

Tabular data: example

DATA — MODEL — RESULTS

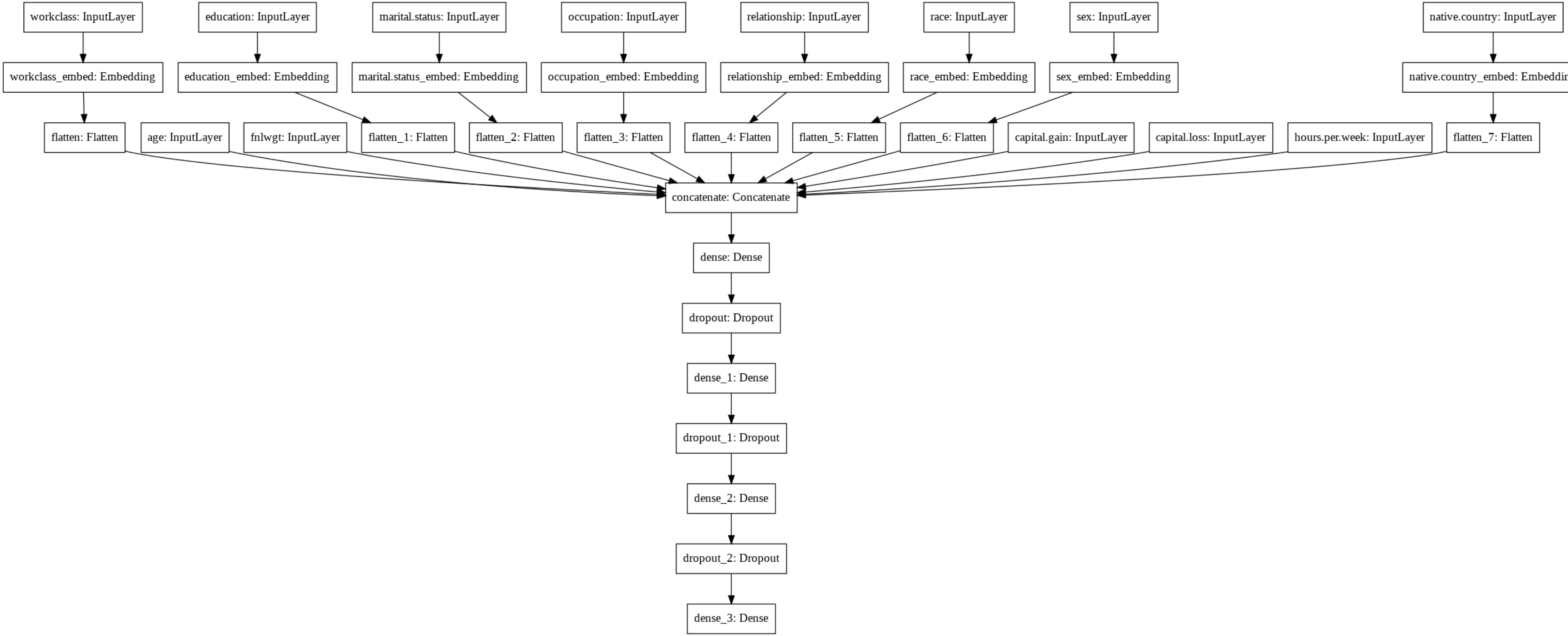
Encode categorical columns or use raw data

```
input_cols = []
encoded_cols = []

for col in X.columns:
    col_name = '.'.join(col.lower().split(" "))
    input_cols.append(krs.layers.Input(shape=(1,), name=col_name))

    # embed discrete column
    if col in translators.keys():
        e = krs.layers.Flatten()(krs.layers.Embedding(
          name=f"{col_name}_embed",
          input_dim=X_train[col].max()+1,
          output_dim=3,
          input_length=1)(input_cols[-1]))

    # use raw input
    else:
        e = input_cols[-1]
    encoded_cols.append(e)

# dense layers
encoded_cols = krs.layers.concatenate(encoded_cols)
layer1 = krs.layers.Dropout(0.1)(krs.layers.Dense(32, activation="tanh")(encoded_cols))
layer2 = krs.layers.Dropout(0.1)(krs.layers.Dense(32, activation="tanh")(layer1))
layer3 = krs.layers.Dropout(0.1)(krs.layers.Dense(16, activation="tanh")(layer2))
out = krs.layers.Dense(1, activation='sigmoid')(layer3)
nn = krs.models.Model(inputs=input_cols, outputs=[out])
```

Dense layers

Tabular data: example

DATA | MODEL | RESULTS

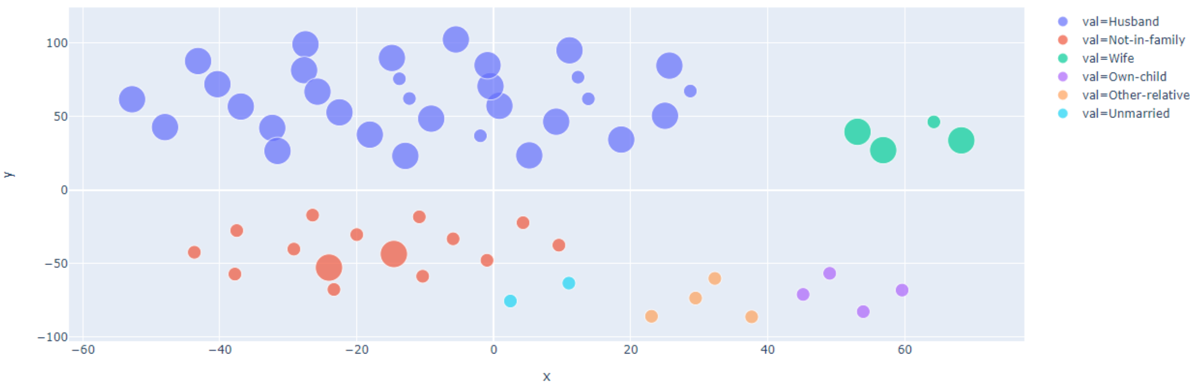


Tabular data: example

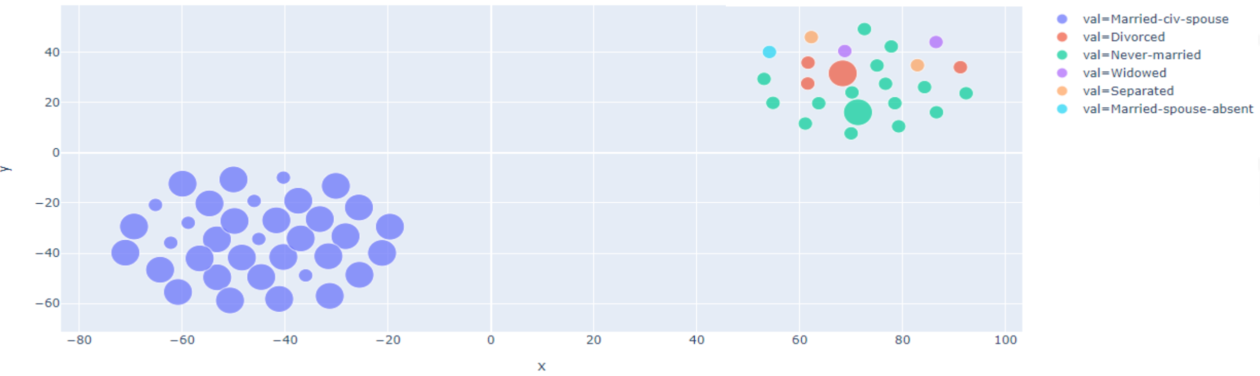
DATA — MODEL — RESULTS

Big bubble: >50K USD
Small bubble: <=50K USD

Visualization of relationship embedding (color) against class (size of bubble)



Visualization of marital.status embedding (color) against class (size of bubble



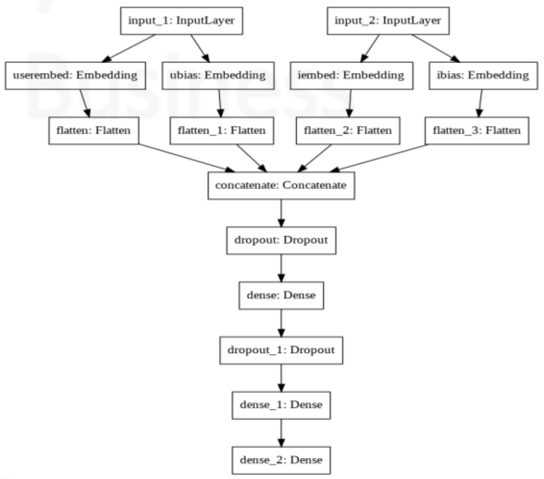
05

Recommendations

Embeddings in a collaborative filtering

Recommendations

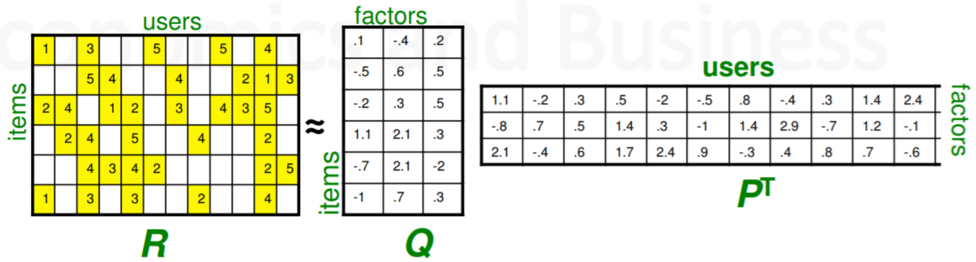
1. Embeddings can be used in recommendation engines - they can replace the Collaborative Filtering algorithm.
2. Given **user id, item id** it can be used to perform matrix decomposition:
   a) **User to latent features** mapping - describe users' "taste"
   b) **Item to latent features** mapping - describe items features
3. The neural network performs a rating **matrix reconstruction.**
4. Then each embedding can be interpreted as a **mapping from a discrete space** of users & items to a common space of **latent features.**
5. **Example algorithms:**
   a) **AutoRec –** based on autoencoders as recommendation engines
   b) **DeepAutoRec –** guess what ☺
   c) **Extreme Deep Factorization Machines –** variation of the above with additional feature interactions

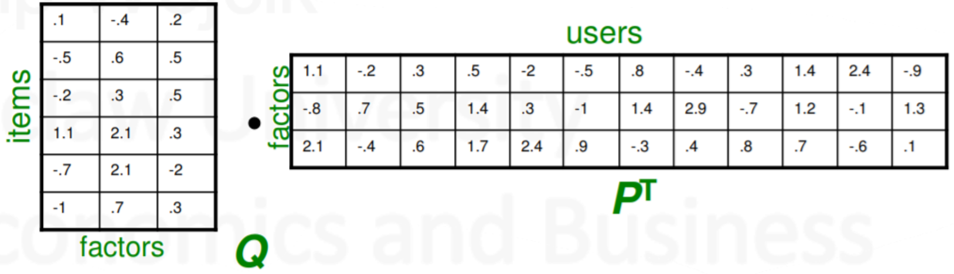


dr Filip Wójcik Wroclaw University of Economics and Business

Recommendations

Classic ratings matrix decomposition approach





Rajaraman, A. and Ullman, J.D., 2011. *Mining of massive datasets*. Cambridge University Press.

dr Filip Wójcik Wroclaw University of Economics and Business

Recommendations

dr Filip Wójcik
Wroclaw University

User / item embedding

Movie\ohe				
Matrix	1	0	0	0
Blade Runner	0	1	0	0
The Ring	0	0	1	0
Dracula	0	0	0	1

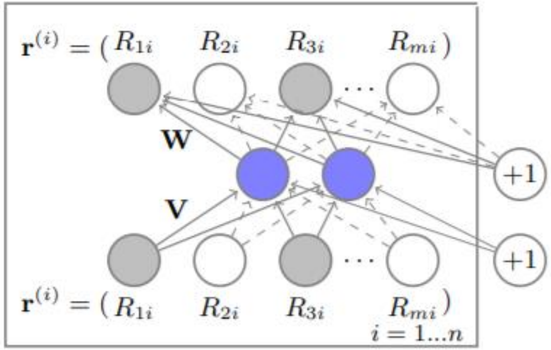
User\Movie	Matrix	Blade Runner	The Ring	Dracula
U1	5	5	3	2
U2	4	5	1	1
U3	1	2	4	4
U4	2	2	5	5

embedding →

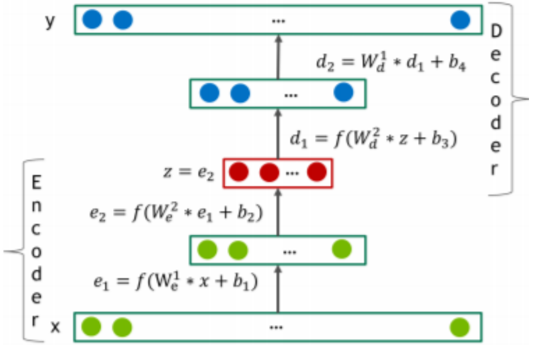
Movie\latent feature	Science fiction	Horror
Matrix	2.98	0.001
Blade Runner	1.23	0.02
The Ring	0.1	1.11
Dracula	0.5	1.32

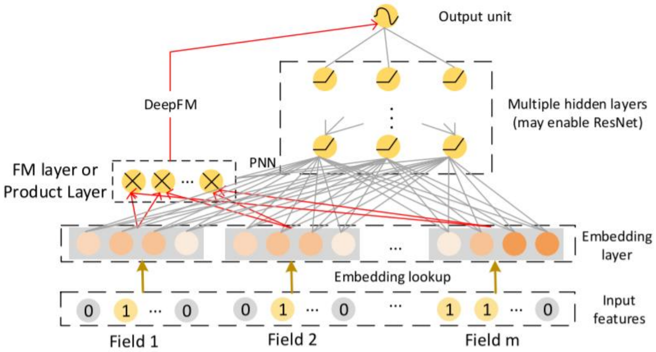
User\latent feature	Science fiction	Horror
U1	0.9	0.1
U2	0.85	0.0
U3	0.11	0.8
U4	0.2	1.0

Recommendations



$$h(\mathbf{r};\theta) = f\left(\mathbf{W} \cdot g(\mathbf{V}\mathbf{r} + \boldsymbol{\mu}) + \mathbf{b}\right)$$





AutoRec
Sedhain, S., Menon, A. K., Sanner, S., Xie, L. (2015). Autorec: Autoencoders meet collaborative filtering. Proceedings of the 24th International Conference on World Wide Web, 111–112.

Deep Auto Rec
Kuchaiev, O., Ginsburg, B. (2017). Training deep autoencoders for collaborative filtering. ArXiv Preprint ArXiv:1708.01715.

Extreme Deep Factorization Machines
Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., Sun, G. (2018). xdeepfm: Combining explicit and implicit feature interactions for recommender systems. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 1754–1763.

Recommendations: example

DATA — MODEL — RESULTS

Movielens: 100’000 ratings, 9’000 movies, 6’000 users

	userId	uid	movieId	mid	rating
0	1	0	1	0	4.0
1	1	0	3	2	4.0
2	1	0	6	5	4.0
3	1	0	47	43	5.0
4	1	0	50	46	5.0
5	1	0	70	62	3.0
6	1	0	101	89	5.0
7	1	0	110	97	4.0
8	1	0	151	124	5.0
9	1	0	157	130	5.0

	movieId	mid	title	genres
0	1	0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	1	Jumanji (1995)	Adventure|Children|Fantasy
2	3	2	Grumpier Old Men (1995)	Comedy|Romance
3	4	3	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	4	Father of the Bride Part II (1995)	Comedy
5	6	5	Heat (1995)	Action|Crime|Thriller
6	7	6	Sabrina (1995)	Comedy|Romance
7	8	7	Tom and Huck (1995)	Adventure|Children
8	9	8	Sudden Death (1995)	Action
9	10	9	GoldenEye (1995)	Action|Adventure|Thriller

Ids need to be consecutive numbers

dr Filip Wójcik Wroclaw University of Economics and Business

dr Filip Wójcik

Recommendations: example

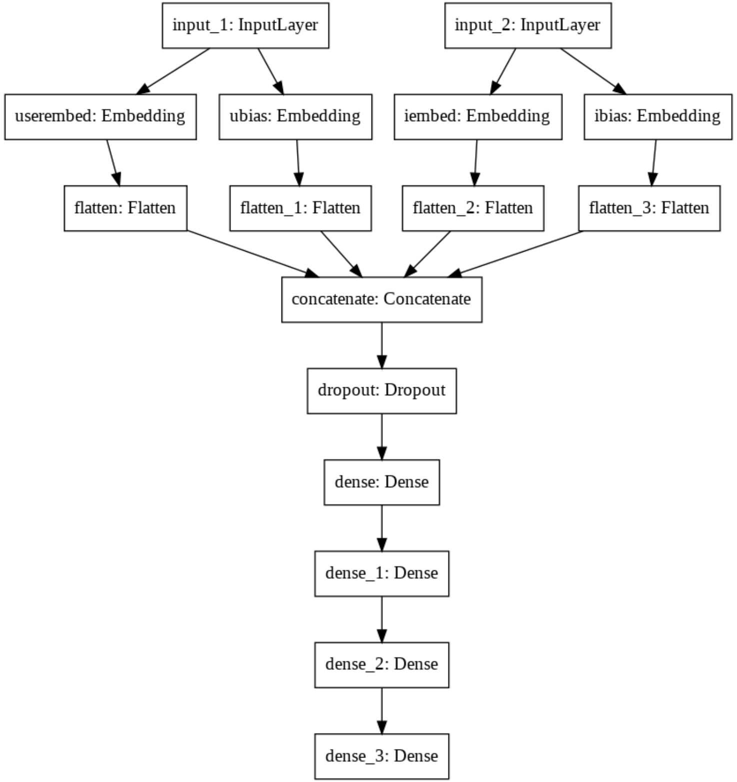
DATA — **MODEL** — RESULTS

User embedding

Movie embedding

Dense layers

```
# User part
uinput = krs.layers.Input(shape=(1,))
uembed = krs.layers.Embedding(
    output_dim=uembed,
    input_dim=nusers+1,
    input_length=1,
    name='userembed',
    embeddings_regularizer=krs.regularizers.l2(1e-6),
            embeddings_initializer=krs.initializers.he_normal()

    )(uinput)
uembed = krs.layers.Flatten()(uembed)
ubias = krs.layers.Flatten()(krs.layers.Embedding(
    output_dim=1,
    input_dim=nusers+1,
    input_length=1,
    name='ubias')(uinput))

# Movie part
iinput = krs.layers.Input(shape=(1,))
iembed = krs.layers.Embedding(
    output_dim=iembed,
    input_dim=nitems+1,
    input_length=1,
    name='iembed',
    embeddings_regularizer=krs.regularizers.l2(1e-6),
    embeddings_initializer=krs.initializers.he_normal()
    )(iinput)
iembed = krs.layers.Flatten()(iembed)
ibias = krs.layers.Flatten()(krs.layers.Embedding(
    output_dim=1,
    input_dim=nitems+1,
    input_length=1,
    name='ibias')(iinput))

# Dense layers

concat = krs.layers.Concatenate()([uembed, iembed, ubias, ibias])
drop1 = krs.layers.Dropout(0.2)(concat)
d1 = krs.layers.Dense(128, activation='relu')(drop1)
d2 = krs.layers.Dense(64, activation='relu')(d1)
d3 = krs.layers.Dense(32, activation='relu')(d2)

out = krs.layers.Dense(1, activation='relu')(d3)

model = krs.models.Model(inputs=[uinput, iinput], outputs=[out])
```



Recommendations: example

DATA — MODEL — RESULTS

dr Filip Wójcik
Wroclaw University
of Economics and Business
dr Filip Wójcik
Wroclaw University
of Economics and Business

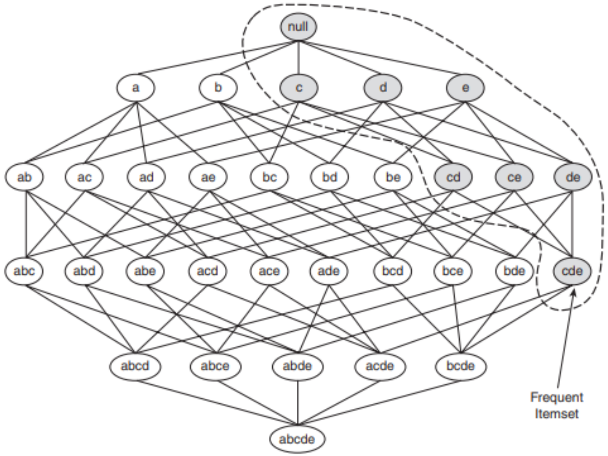
06

Frequent items

Embeddings as a tool for market basket problems.

Frequent items

1. Frequent items problems are the primary tool for market basket analysis:
   a) Which goods do customers **buy often**?
   b) Which goods are **bought together** - in pairs/triplets/etc.?
   c) Are there any rare connections between products?
2. Typically, this kind of problems was solved by algorithms like:
   a) A priori - historically the first algorithm
   b) FpGrowth - much more effective implementation
3. Embeddings can be used in a similar way as in recommendation engine
   a) Items bought together will show **up close to each other.**
   b) One can investigate a space to find its neighbors. **surrounding a single item.**
   c) We can use algorithms used in language processing, like **Word2Vec in gensim**



Tan, P.N., Steinbach, M. and Kumar, V., 2016. *Introduction to data mining*. Pearson Education India.

Frequent items

DATA — MODEL — RESULTS

Online retail Kaggle Dataset, 3921 items, 4338 users

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0
536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0
536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0

Express data as „transactions":

- a list of items that user **buys in one go**
- each transaction is like **a „sentence" in NLP**

Transaction 1	User 1	Item 1, Item2, Item 3
Transaction 2	User 1	Item 2, Item 3
Transaction 3	User 2	Item 1, Item 3, Item 4

dr Filip Wójcik
Wroclaw University of Economics and Business

Frequent items

DATA — MODEL — RESULTS

1. Use **word2vec** from NLP world
2. „Vocabulary” are all items
3. „Context” – is a single transaction
4. Finding similar items - nearest neigbors in an embeddings space

Prepare „sentences” - transactions

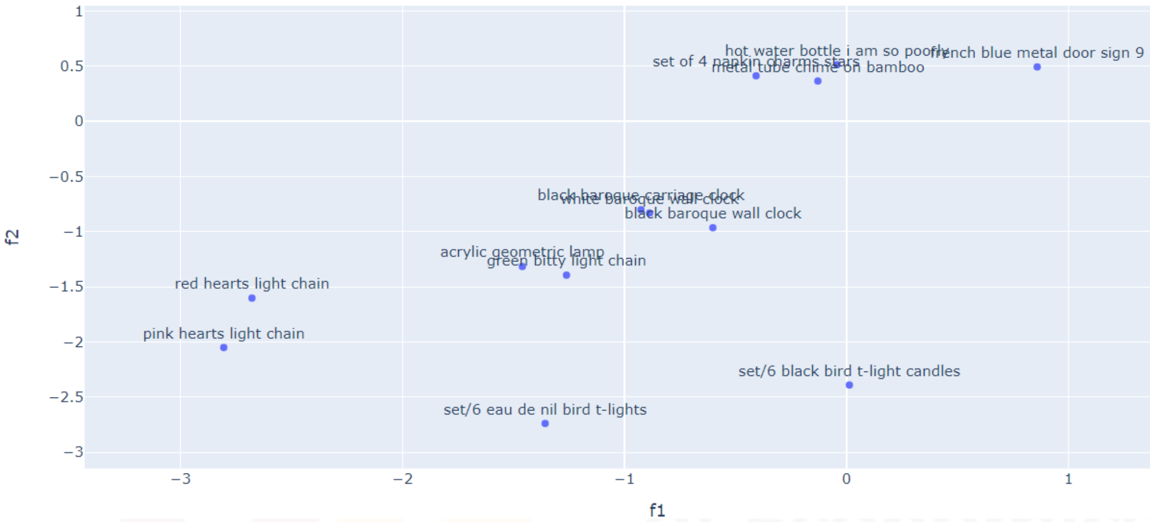
Word2Vec == Item2Vec

```
transactions = []
for grp, data in valid_market_data.groupby("InvoiceNo"):
    tran = data.iid.astype(str).to_list()
    transactions.append(tran)

model = Word2Vec(transactions, min_count=1)
X = model[model.wv.vocab]
```

dr Filip Wójcik
Wroclaw University
of Economics and Business

Frequent items

DATA — MODEL — **RESULTS**

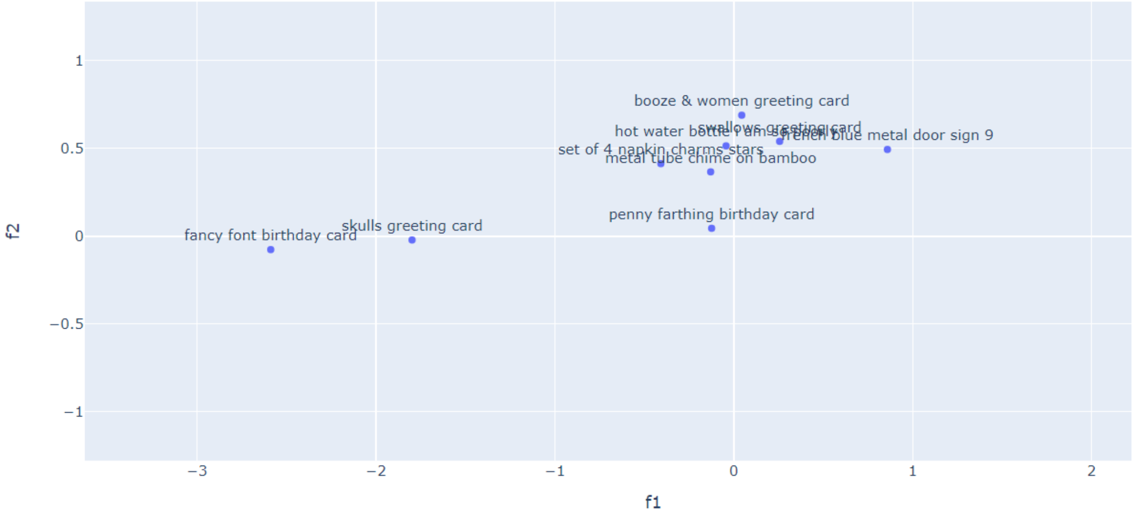



```
item = 'black baroque wall clock'
for item in find_most_similar(model, item):
  print(item)

> 'acrylic geometric lamp'
> 'black baroque carriage clock'
> 'white baroque wall clock'
> 'eau de nil love bird candle'
> 'red hearts light chain'
> 'pink love bird candle'
> 'pink hearts light chain'
> 'set/6 eau de nil bird t-lights'
> 'set/6 black bird t-light candles'
> 'green bitty light chain'
```

dr Filip Wójcik
Wroclaw University of Economics and Business

Frequent items

DATA — MODEL — **RESULTS**




```python
item = 'fancy font birthday card'
for item in find_most_similar(model, item):
  print(item)

>'elephant birthday card'
>'gin & tonic diet greeting card'
>'vintage kid dolly card'
>'cowboys and indians birthday card'
>'skulls greeting card'
>'robot birthday card'
>'booze & women greeting card'
>'swallows greeting card'
>'penny farthing birthday card'
>'ring of roses birthday card'
```

Literature

Chai, W., Ispir, M., others. (2016). Wide & deep learning for recommender systems. Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, 7–10.

Church, K. W. (2017). Word2Vec. *Natural Language Engineering*, *23*(1), 155-162.
Guo, H., Tang, R., Ye, Y., Li, Z., He, X. (2017). DeepFM: a factori]ation-machine based neural network for CTR prediction. ArXiv Preprint ArXiv:1703.04247

Kuchaiev, O., & Ginsburg, B. (2017). Training deep autoencoders for collaborative filtering. *arXiv preprint arXiv:1708.01715*.

Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., & Sun, G. (2018, July). xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 1754-1763).

Pennington, J., Socher, R., & 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).

Sedhain, S., Menon, A. K., Sanner, S., Xie, L. (2015). Autorec: Autoencoders meet collaborative filtering. Proceedings of the 24th International Conference on World Wide Web, 111–112.

Zhang, L., Zhang, S., & Balog, K. (2019, July). Table2vec: Neural word and entity embeddings for table population and retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 1029-1032).

dr Filip Wójcik
Wroclaw University
of Economics and Business

ML EDITION

Thank you for watching!