# Recommendation Systems

**Unclassical embeddings / W2V**

Eugeny Malyutin / Sergey Dudorov

# Vector embeddings

Eugeny Malyutin

# Previously:

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

$$TF-IDF(w,d,C) = \frac{count(w,d)}{count(d)} * log(\frac{\sum_{d' \in C} 1(w,d')}{|C|})$$

# One-hot encoding drawback:

- “monkeys eat bananas” and “apes consume fruits” - similarity equals to 0
- «Pouteria is widespread throughout the tropical regions of the world and monkeys eat their fruits»(c). What is Pouteria? Is it a tree?
- «a word is characterized by the company it keeps» – John Rupert Firth
- Ideally, we want vector representations where ***similar words*** end up with ***similar vectors***. **Dense** vectors. And when I say similar a mention some **similarity measure** (cosine).
- Even better, we’d want more similar representations when the words share some properties such as if they’re both plural or singular, verbs or adjectives or if they both reference to a male.

# Word2vec scheme:

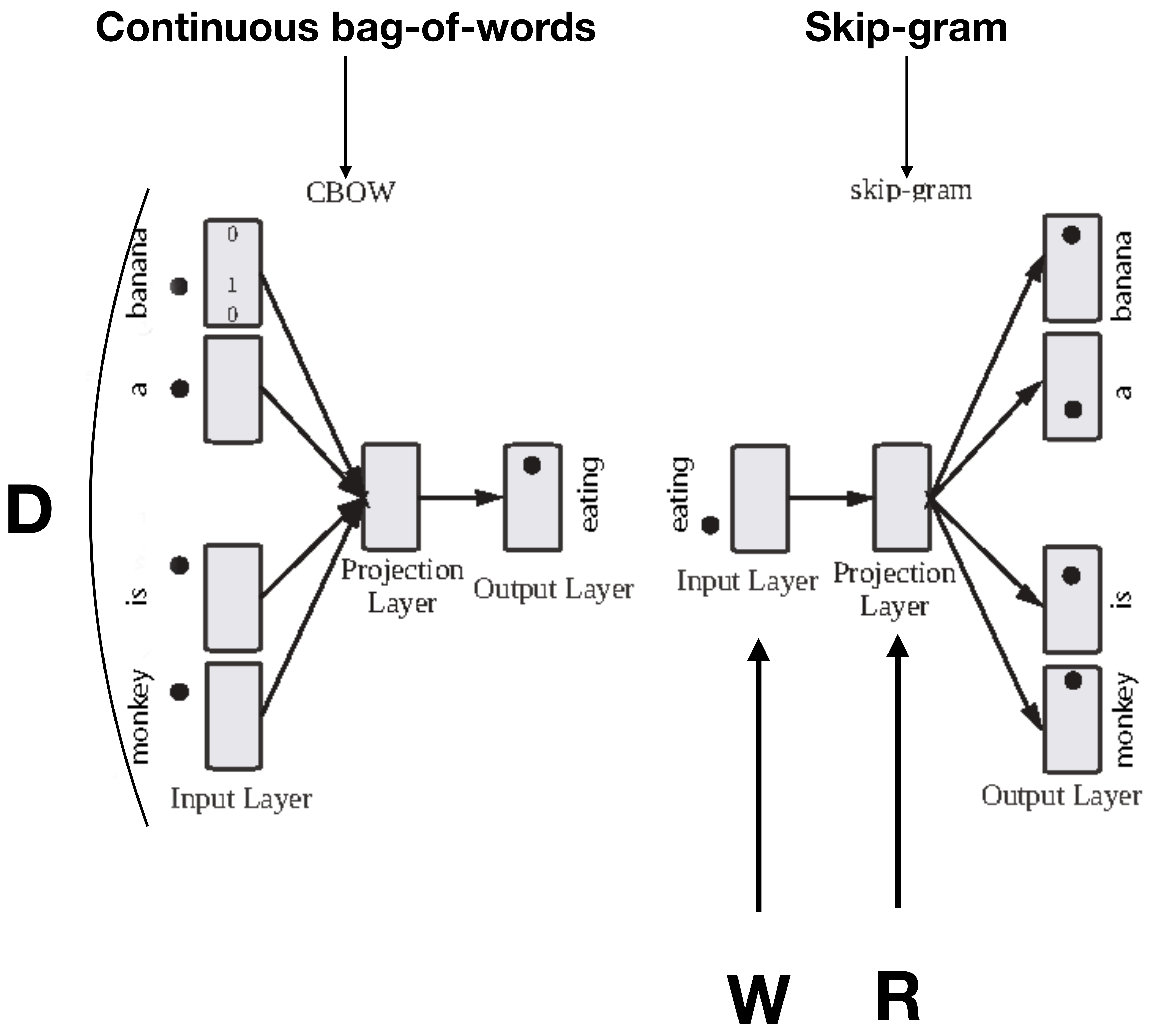
- It has an input layer that receives **D** one-hot encoded words which are of dimension **V** (the size of the vocabulary).

- It «averages» them, creating a single input vector.

- That input vector is multiplied by a weights matrix ***W*** (that has size VxD, being D nothing less than the dimension of the vectors that you want to create). That gives you as a result a D-dimensional vector.

- The vector is then multiplied by another matrix (**R** - reverse W), this one of size DxV. The result will be a new V-dimensional vector.

- That V-dimensional vector is normalized to make all the entries a number between 0 and 1, and that all of them sum 1, using the softmax function, and that's the output. It has in the i-th position the predicted probability of the i-th word in the vocabulary of being the one in the middle for the given context.



# The skipgram model

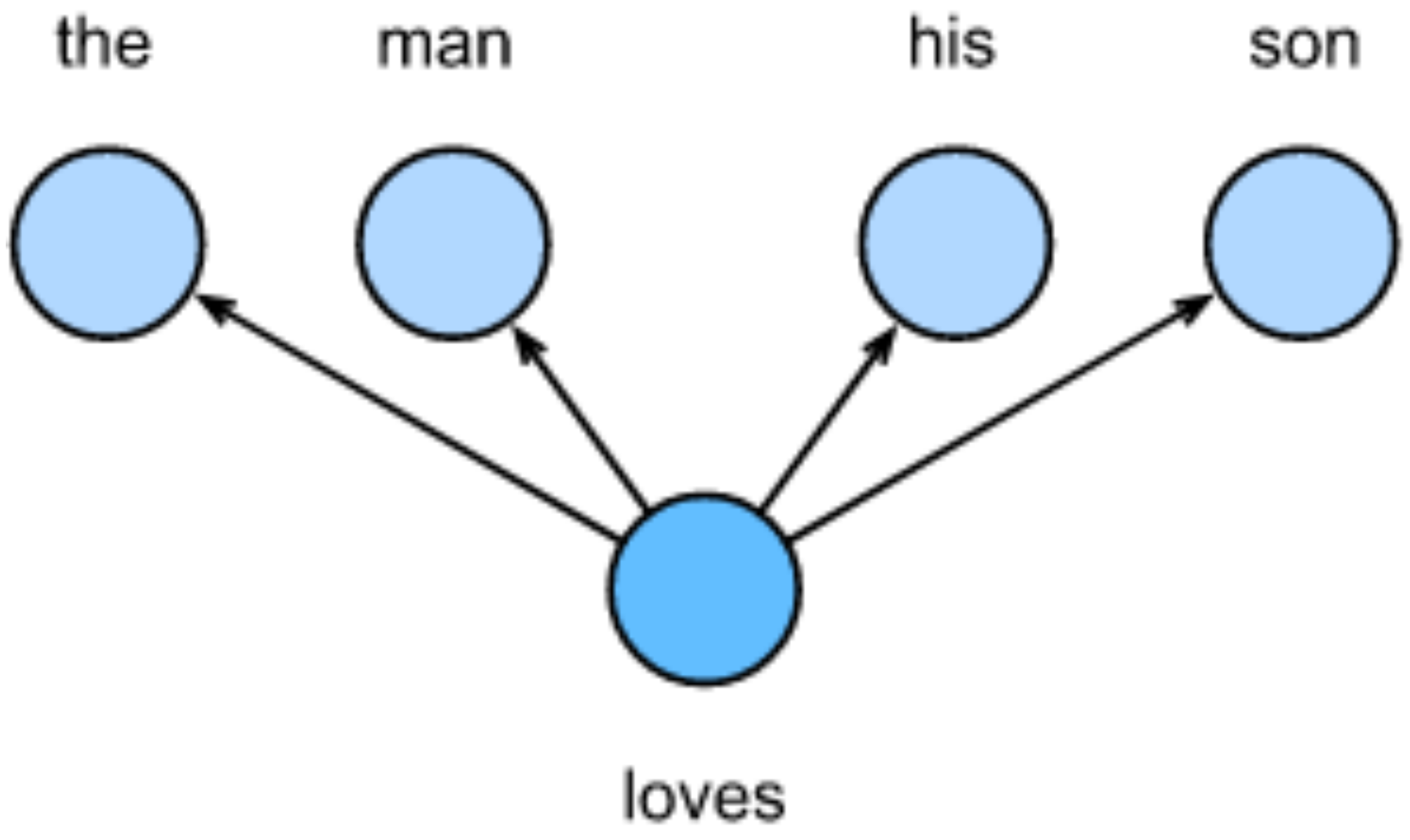
- We assume that, given the central target word, the context words are generated independently of each other.
  P(the, man, his, son | loves) = P(the | loves) * P(man | loves) * P(his | loves) * P(son | loves)

- And $p(w_o | w_c) = \frac{exp(u_o^T v_c)}{\sum_{i \in V} exp(u_i^T v_c)}$ **cond. probability**, u and v — vector representations.

  u_0 - context,
  v_c — central target.

- The **likelihood function** of the skip-gram model:

$$\prod_{i=1}^{T} \prod_{-m \leq j \leq m} P(w^{(t+j)} | w^t)$$



# Skipgram model training

- Loss function $-\sum_{t=1}^{T} \sum_{-m \leq j \leq m,\ j \neq 0} \mathbf{log}\, \mathbb{P}(w^{(t+j)} \mid w^{(t)})$

- If we want to SGD it - we need to compute gradient of conditional probability:

$$\log \mathbb{P}(w_o \mid w_c) = \mathbf{u}_o^\top \mathbf{v}_c - \log\left(\sum_{i \in \mathcal{V}} \mathbf{exp}(\mathbf{u}_i^\top \mathbf{v}_c)\right)$$

- Through differentiation, we can get the gradient from the formula above.

$$\begin{aligned}
\frac{\partial \mathbf{log}\, \mathbb{P}(w_o \mid w_c)}{\partial \mathbf{v}_c} &= \mathbf{u}_o - \frac{\sum_{j \in \mathcal{V}} \exp(\mathbf{u}_j^\top \mathbf{v}_c)\mathbf{u}_j}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)} \\
&= \mathbf{u}_o - \sum_{j \in \mathcal{V}} \left(\frac{\mathbf{exp}(\mathbf{u}_j^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \mathbf{exp}(\mathbf{u}_i^\top \mathbf{v}_c)}\right)\mathbf{u}_j \\
&= \mathbf{u}_o - \sum_{j \in \mathcal{V}} \mathbb{P}(w_j \mid w_c)\mathbf{u}_j.
\end{aligned}$$

- Any problems?

# Skipgram model training

- Loss function $$-\sum_{t=1}^{T} \sum_{-m \leq j \leq m,\ j \neq 0} \textbf{log}\, \mathbb{P}(w^{(t+j)} \mid w^{(t)})$$

- If we want to SGD it - we need to compute gradient of conditional probability:

$$\log \mathbb{P}(w_o \mid w_c) = \mathbf{u}_o^\top \mathbf{v}_c - \log\left(\sum_{i \in \mathcal{V}} \textbf{exp}(\mathbf{u}_i^\top \mathbf{v}_c)\right)$$

- Through differentiation, we can get the gradient from the formula above:

$$\begin{aligned}
\frac{\partial \textbf{log}\, \mathbb{P}(w_o \mid w_c)}{\partial \mathbf{v}_c} &= \mathbf{u}_o - \frac{\sum_{j \in \mathcal{V}} \exp(\mathbf{u}_j^\top \mathbf{v}_c)\mathbf{u}_j}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)} \\
&= \mathbf{u}_o - \sum_{j \in \mathcal{V}} \left(\frac{\textbf{exp}(\mathbf{u}_j^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \textbf{exp}(\mathbf{u}_i^\top \mathbf{v}_c)}\right)\mathbf{u}_j \\
&= \mathbf{u}_o - \sum_{j \in \mathcal{V}} \mathbb{P}(w_j \mid w_c)\mathbf{u}_j .
\end{aligned}$$

- Its computation obtains the conditional probability for **all the words in the dictionary** given the central target word w_c
We then use **the same method** to obtain the gradients **for other word vectors**.

# Negative sampling:

- Given a context window for the central target word $w\_c$, we will treat it as an event for context word $w\_o$ to appear in the context window and compute the probability of this event from
$$\mathbb{P}(D = 1 \mid w_c, w_o) = \sigma(\mathbf{u}_o^\top \mathbf{v}_c),$$
- Now we consider maximizing the joint probability $\prod_{t=1}^{T} \prod_{-m \leq j \leq m,\ j \neq 0} \mathbb{P}(D = 1 \mid w^{(t)}, w^{(t+j)})$.
- However, the events included in the model only consider positive examples. We need to sample additional negative events (never occurred in the same context) and then:

$$\mathbb{P}(w^{(t+j)} \mid w^{(t)}) = \mathbb{P}(D = 1 \mid w^{(t)}, w^{(t+j)}) \prod_{k=1,\ w_k \sim \mathbb{P}(w)}^{K} \mathbb{P}(D = 0 \mid w^{(t)}, w_k).$$

# Negative sampling:

- Now we consider maximizing the joint probability $$\prod_{t=1}^{T} \prod_{-m\leq j\leq m,\ j\neq 0} \mathbb{P}(D=1 \mid w^{(t)}, w^{(t+j)})\,.$$
- However, the events included in the model only consider positive examples. We need to sample additional negative K events (never occurred in the same context) and then:
$$\mathbb{P}(w^{(t+j)} \mid w^{(t)}) = \mathbb{P}(D=1 \mid w^{(t)}, w^{(t+j)}) \prod_{k=1,\ w_k\sim\mathbb{P}(w)}^{K} \mathbb{P}(D=0 \mid w^{(t)}, w_k)\,.$$
- The logarithmic loss for the conditional probability above is $$-\log \mathbb{P}(w^{(t+j)} \mid w^{(t)}) = -\log \mathbb{P}(D=1 \mid w^{(t)}, w^{(t+j)}) - \sum_{k=1,\ w_k\sim\mathbb{P}(w)}^{K} \log \mathbb{P}(D=0 \mid w^{(t)}, w_k)$$
- Here, the gradient computation in each step of the training is no longer related to the dictionary size, but linearly related to $K$
$$= -\log\ \sigma\left(\mathbf{u}_{i_{t+j}}^{\top}\mathbf{v}_{i_t}\right) - \sum_{k=1,\ w_k\sim\mathbb{P}(w)}^{K} \log\left(1-\sigma\left(\mathbf{u}_{h_k}^{\top}\mathbf{v}_{i_t}\right)\right)$$
$$= -\log\ \sigma\left(\mathbf{u}_{i_{t+j}}^{\top}\mathbf{v}_{i_t}\right) - \sum_{k=1,\ w_k\sim\mathbb{P}(w)}^{K} \log\sigma\left(-\mathbf{u}_{h_k}^{\top}\mathbf{v}_{i_t}\right).$$

# Negative sampling:

- The logarithmic loss for the conditional probability above is

$$-\log \mathbb{P}(w^{(t+j)} \mid w^{(t)}) = -\log \mathbb{P}(D=1 \mid w^{(t)}, w^{(t+j)}) - \sum_{k=1,\ w_k \sim \mathbb{P}(w)}^{K} \log \mathbb{P}(D=0 \mid w^{(t)}, w_k)$$

$$= -\log\, \sigma\left(\mathbf{u}_{i_{t+j}}^{\top}\mathbf{v}_{i_t}\right) - \sum_{k=1,\ w_k \sim \mathbb{P}(w)}^{K} \log\left(1 - \sigma\left(\mathbf{u}_{h_k}^{\top}\mathbf{v}_{i_t}\right)\right)$$

$$= -\log\, \sigma\left(\mathbf{u}_{i_{t+j}}^{\top}\mathbf{v}_{i_t}\right) - \sum_{k=1,\ w_k \sim \mathbb{P}(w)}^{K} \log \sigma\left(-\mathbf{u}_{h_k}^{\top}\mathbf{v}_{i_t}\right).$$

- Here, the gradient computation in each step of the training is no longer related to the dictionary size, but linearly related to $K$

- **Key idea:** sample additional negatives and learn your positive probabilities «opposite to» them.

# So what? (Synonyms)

```
get_similar_tokens('chip', 3, glove_6b50d)
```

```
cosine sim=0.856: chips
cosine sim=0.749: intel
cosine sim=0.749: electronics
```

```
get_similar_tokens('baby', 3, glove_6b50d)
```

```
cosine sim=0.839: babies
cosine sim=0.800: boy
cosine sim=0.792: girl
```

```
get_similar_tokens('beautiful', 3, glove_6b50d)
```

```
cosine sim=0.921: lovely
cosine sim=0.893: gorgeous
cosine sim=0.830: wonderful
```

- *get_similar_tokens* — top-K words by cosine measure to the target word;
- glove_6b50d — glove model on some common corpora (Wikipedia?) with 6B of words and vector dimension equals to 50;

# So what? (2) (Finding Analogies)

```
get_analogy('man', 'woman', 'son', glove_6b50d)
```

```
'daughter'
```

"Capital-country" analogy

```
get_analogy('bad', 'worst', 'big', glove_6b50d)
```

```
'biggest'
```

"Adjective-superlative adjective" analogy

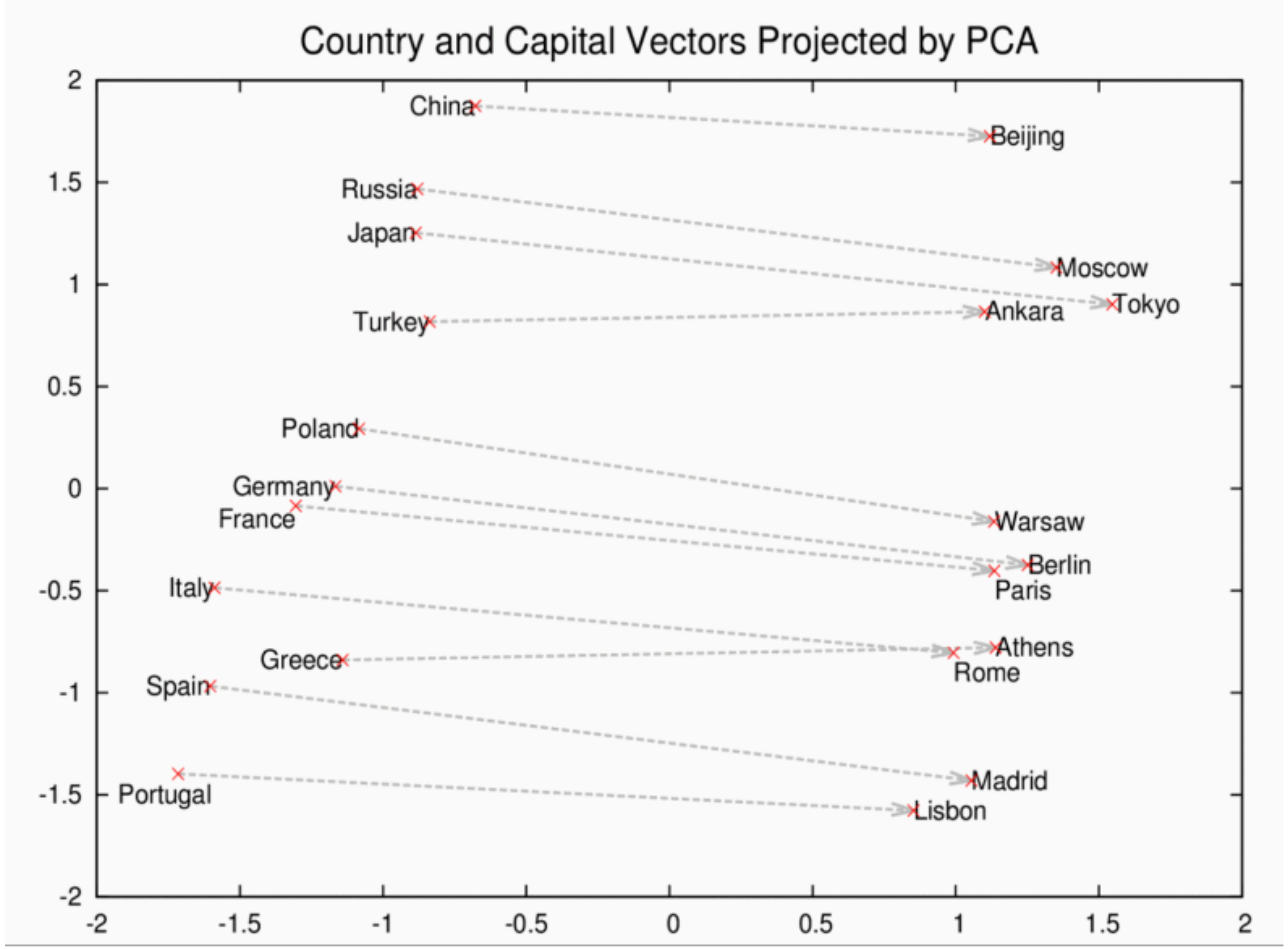
```
get_analogy('do', 'did', 'go', glove_6b50d)
```

```
'went'
```

"Present tense verb-past tense verb" analogy

- And it's only x = $\textbf{vec}(c) + \textbf{vec}(b) - \textbf{vec}(a)$
- And then top word for x

# So what? (country-capital)




**Based on Wikipedia training corpora**

# Any problems?

- Out-of-vocabulary

- How we can train it? How big our doc’s collection should be?

- Stop, firstly we talk about **text** and word2vec is about **words**

# Any problems?

- Out-of-vocabulary

  Yeah, it’s true. But there are few extensions; (fastText)

- Stop, firstly we talk about **text** and word2vec is about **words**

  Ok, average it; Or average with weights; Or do not average and learn some averaging embedding; (look to BERT model)

- How we can train it? How big our doc’s collection should be?

  Really big; Starting from 10+M of symbols; Use pertained vectors;

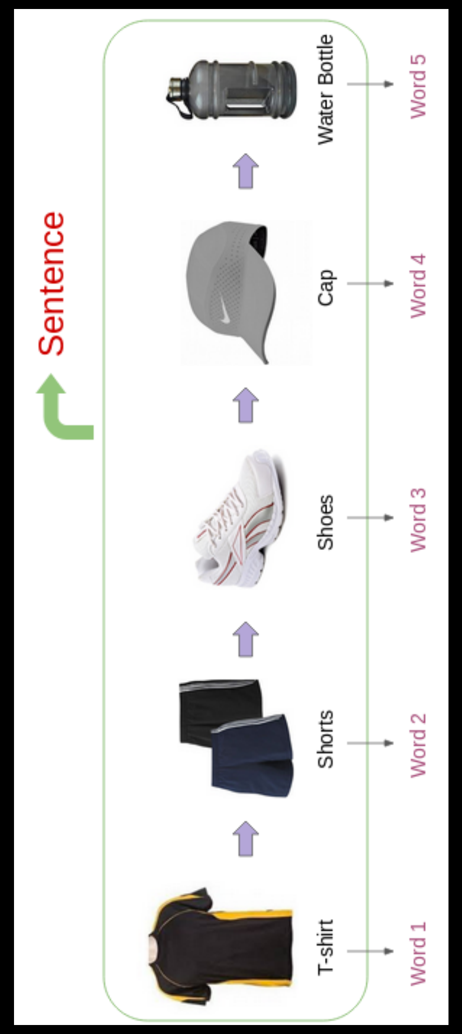
# Word2Vec in RecSys

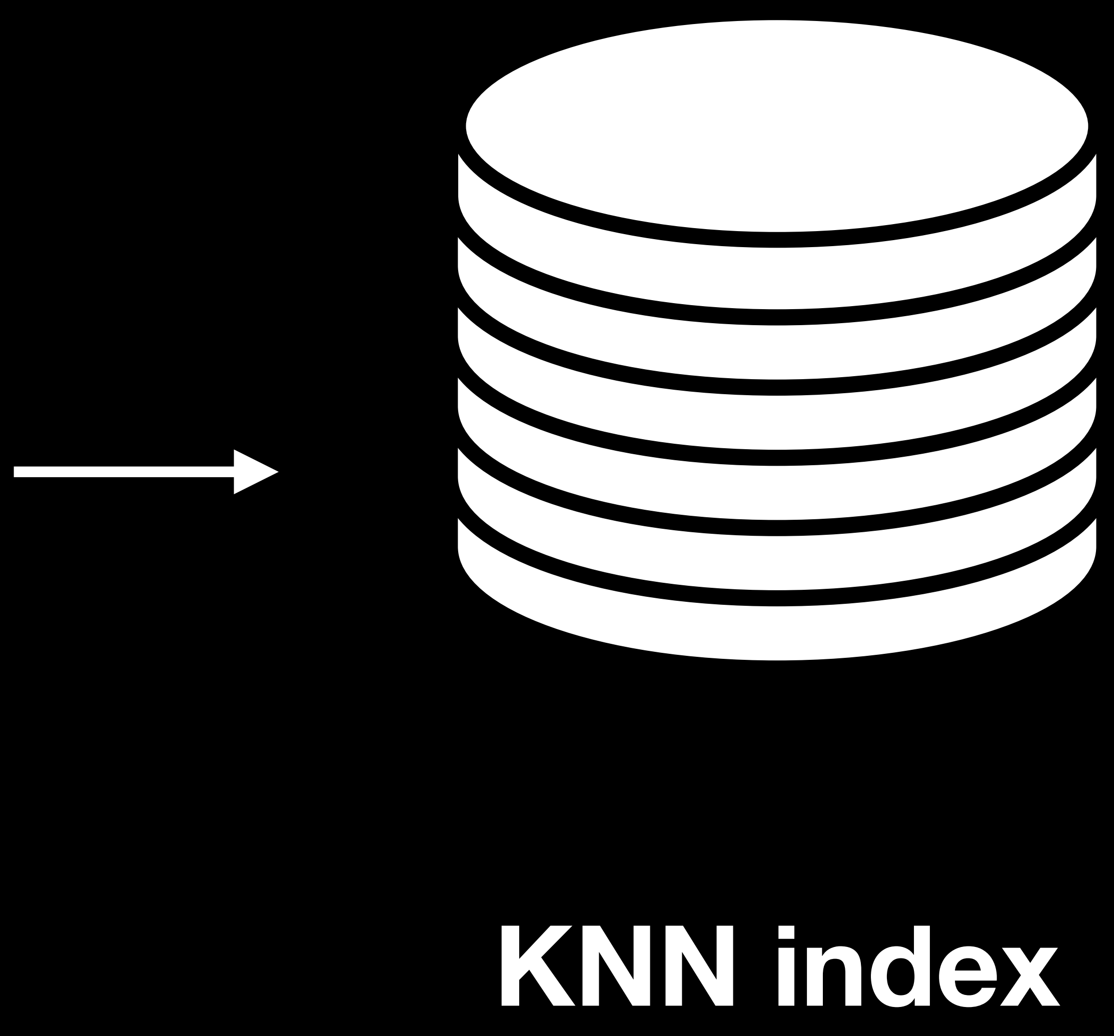
- Negative sampling
- Next item in sequence as a task
- Good initialisation for «Embedding layer»
- Filter your datasets, please

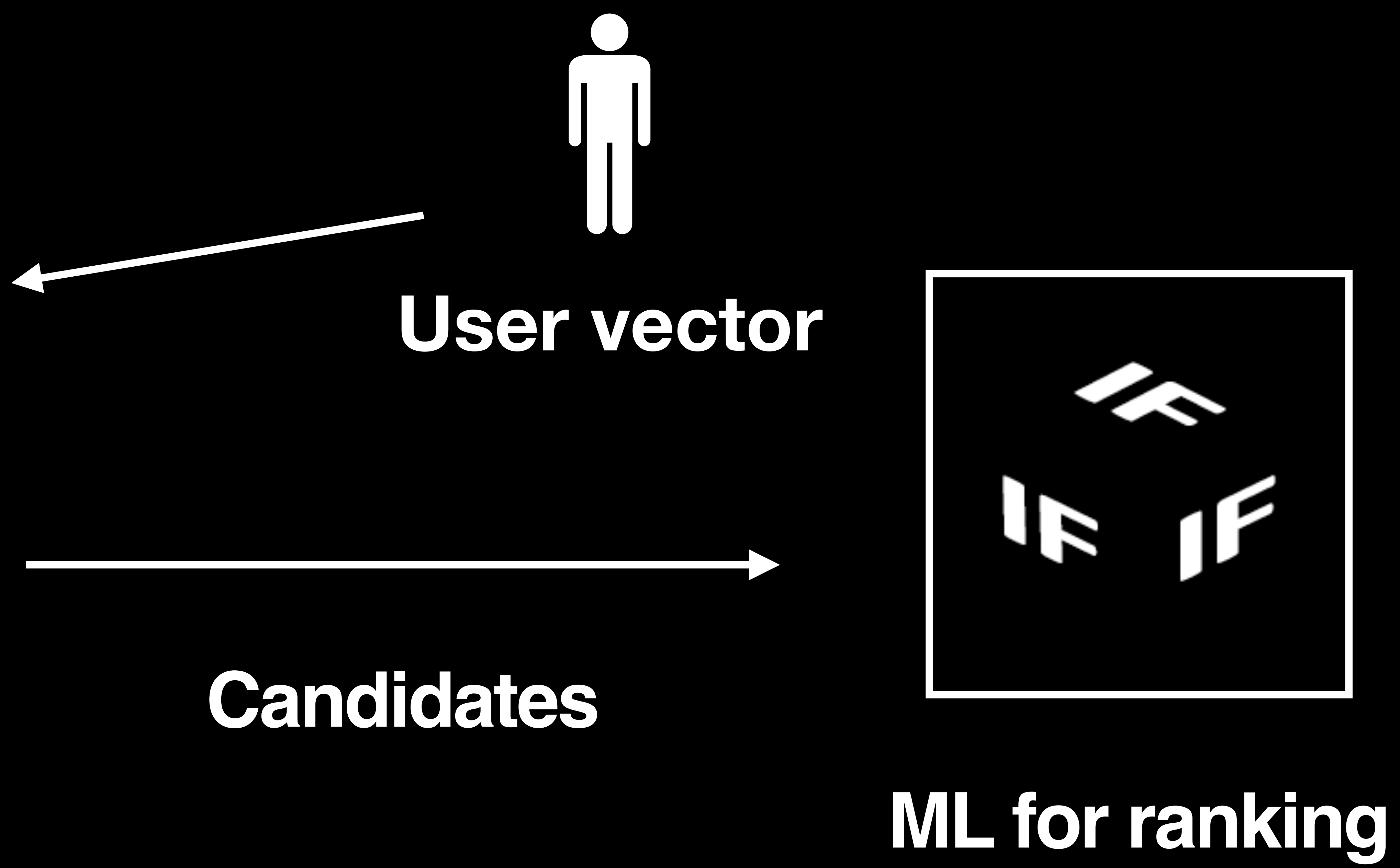
# Word2Vec in RecSys

- Negative sampling
- Next item in sequence as a task
- Good initialisation for «Embedding layer»
- Filter your datasets, please

**Problem?**




**Word2vec offline model with embeddings**

# W2V:

- **No user** vector:
  - Sum all items user interacted with
  - Sum with weights ~ activity
  - Sum with TF-IDF
  - Cluster and select medoid…
  - Feed to an attent…

- **Pros:**
  - Easy to realise
  - No need to limit users interaction
  - Good at similars
  - Session-based
  - Consume CPU , not GPU
- **Cons:**
  - Nothing except interactions
  - Cold start with items
  - No user vector

# W2V

## Discussion notes

- Items are limited by number of interactions — how to overcome it?
- Intrinsic evaluation. What to do?

- Rare items lost their similars quality, why?