New advances in Graph Representations for Ecommerce Search

Pedro Balage

Data Science Portugal Meetup - DSPT #47

Lisbon, 08th January

About me

FARFETCH

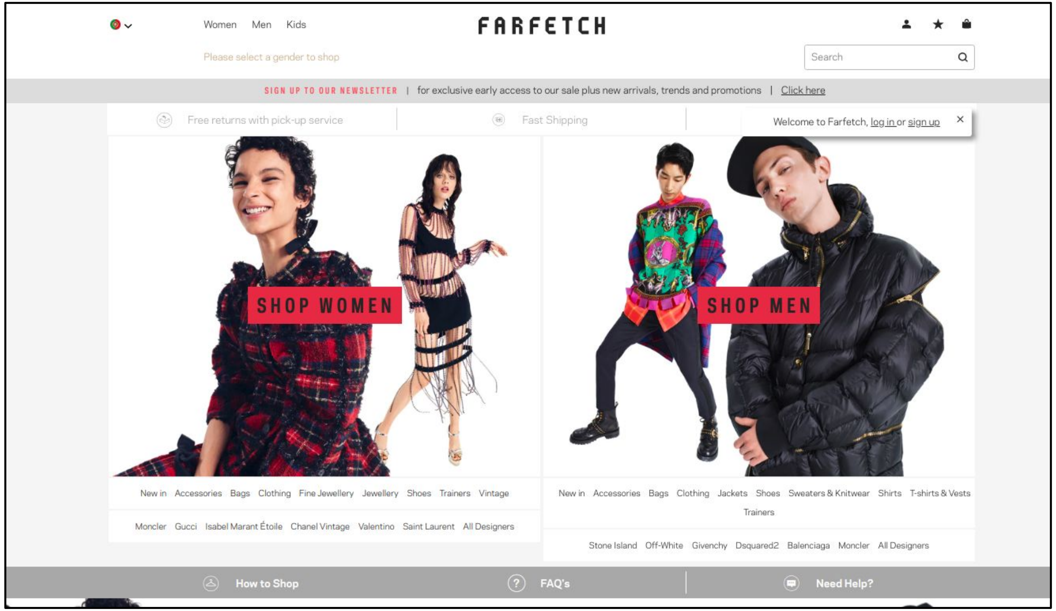
- Lead Data Scientist @ Farfetch - Search team
- Research and Industry experience in **NLP**
- Affiliated to Instituto de Telecomunicações
- Member of organization for the Lisbon Machine Learning School (**LxMLS)**

LXMLS
LISBON
MACHINE LEARNING
SCHOOL

instituto de
telecomunicações

FARFETCH context

- Online retail platform for fashion
- **+100k search queries** every day
- Search experience is key in ecommerce business



The problem: How to make a good search engine for ecommerce?

Steps to develop a good search engine

elasticsearch

Solr

Information Retrieval

- Indexing
- Retrieval
- Learning-to-rank

Relevance Search

- Measure
- Tweak relevance minimum
- Tweak boost field
- Manage exceptions

Natural Language Processing

- Language Analysers
- Synonyms/Acronyms
- Query understanding
- Auto-suggestion
- Did you mean?

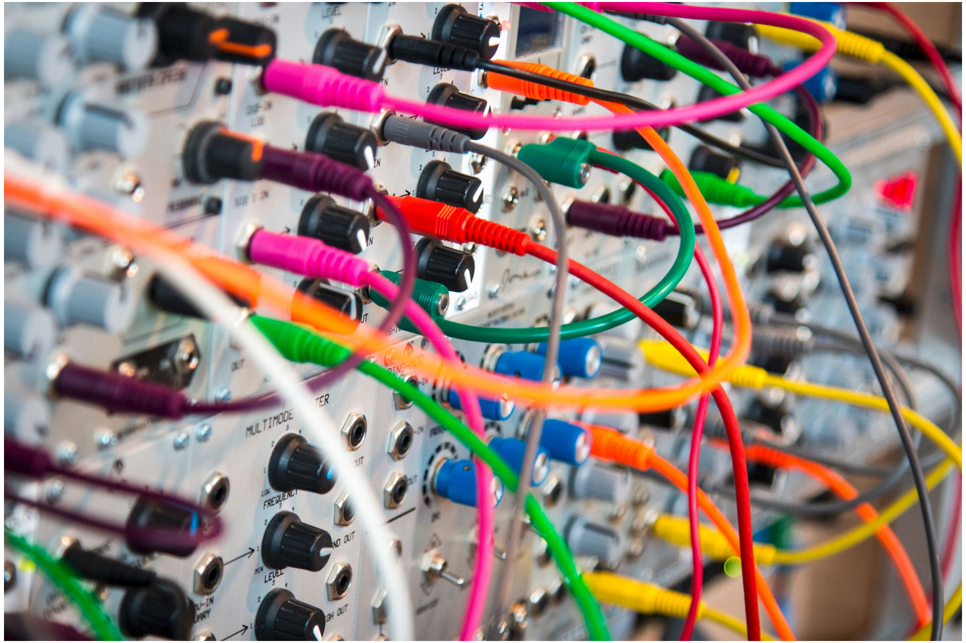
OPEN NLP™

spaCy

K Keras

Next steps...

- Why **XYZ** is not matching with **XZY**?
- What happens if I add this **synonym?**
- Let's add just one more **exception.**



Data Science

Click-through query logs

- Click-through logs provide **query-document relevance**!
- Incorporating **user feedback** is one of the most effective ways to improve a search engine.
- Lot of papers on the topic: SIGIR, KDD, RecSys, etc

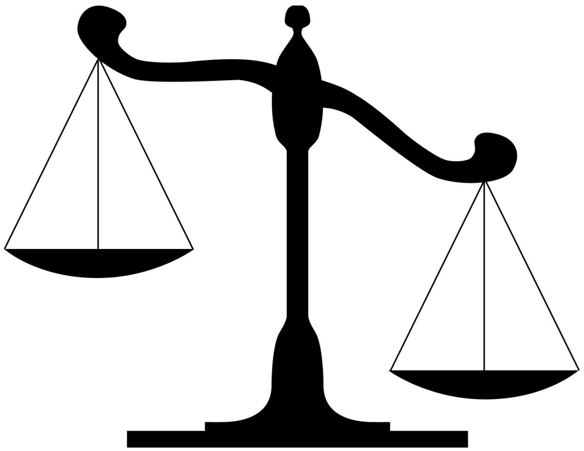
Pros and Cons of using clickthrough data

Pros

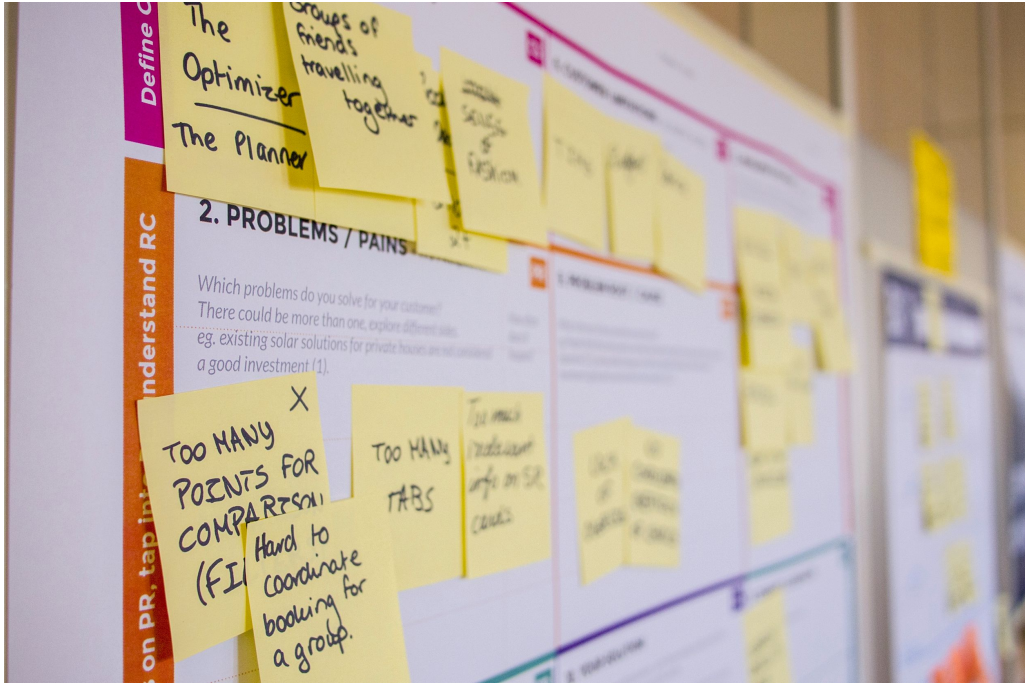
- Relies on user's feedback
- Link interactions among products

Cons

- Noisy and Sparse
- Only provides a true positive set for relevance
- Relevance vs popularity



The problem: How to use the clickthrough data?



An overview of literature papers on representation learning for click through logs

Click Graph (SIGIR 2016)

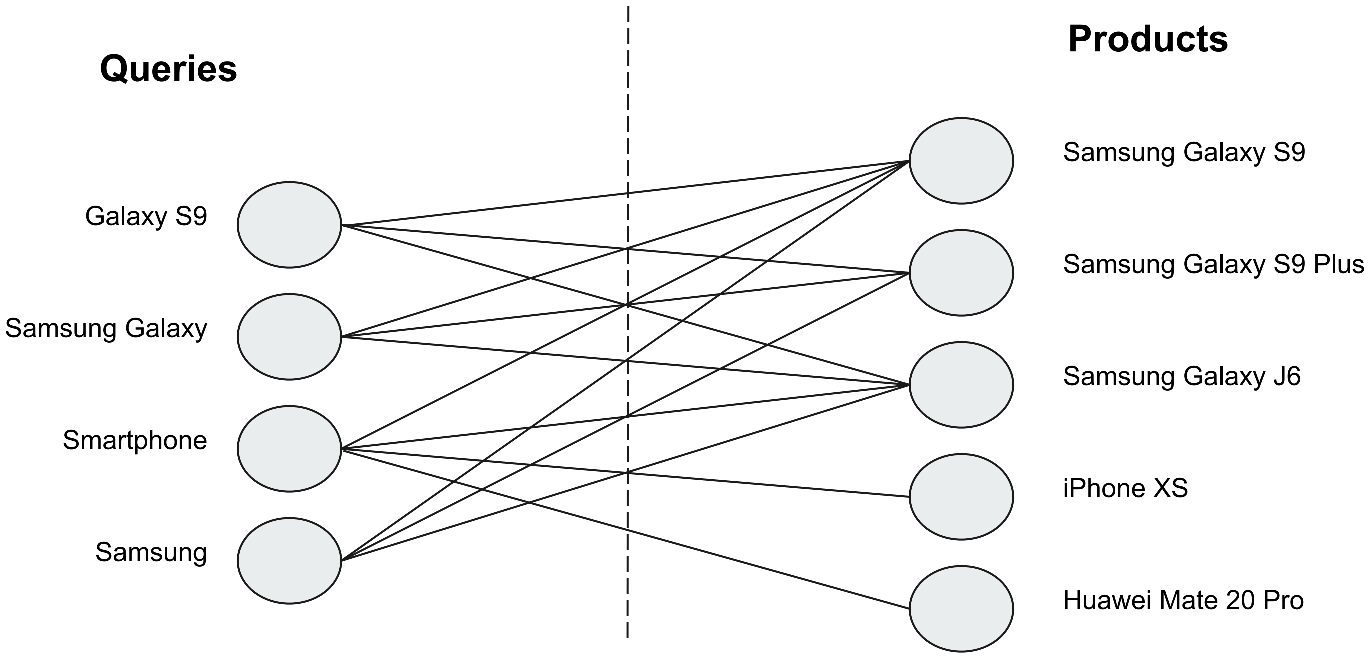
- Paper: **Learning Query and Document Relevance from a Web-scale Click Graph**

- **Motivation**
    - Improving **coverage** over purely click-based approaches
    - Efficient and **scalable** approach that can be easily applied to large scale click logs
        - Matrix factorization-based methods are not efficient for large graphs
        - Paper reports experiments with 25 billion query-document pairs
- **Approach**
    - Learn vector representation based on both content and click information

Shan Jiang, Yuening Hu, Changsung Kang, Tim Daly, Jr., Dawei Yin, Yi Chang, and Chengxiang Zhai. 2016. Learning Query and Document Relevance from a Web-scale Click Graph. In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '16). ACM, New York, NY, USA, 185-194.

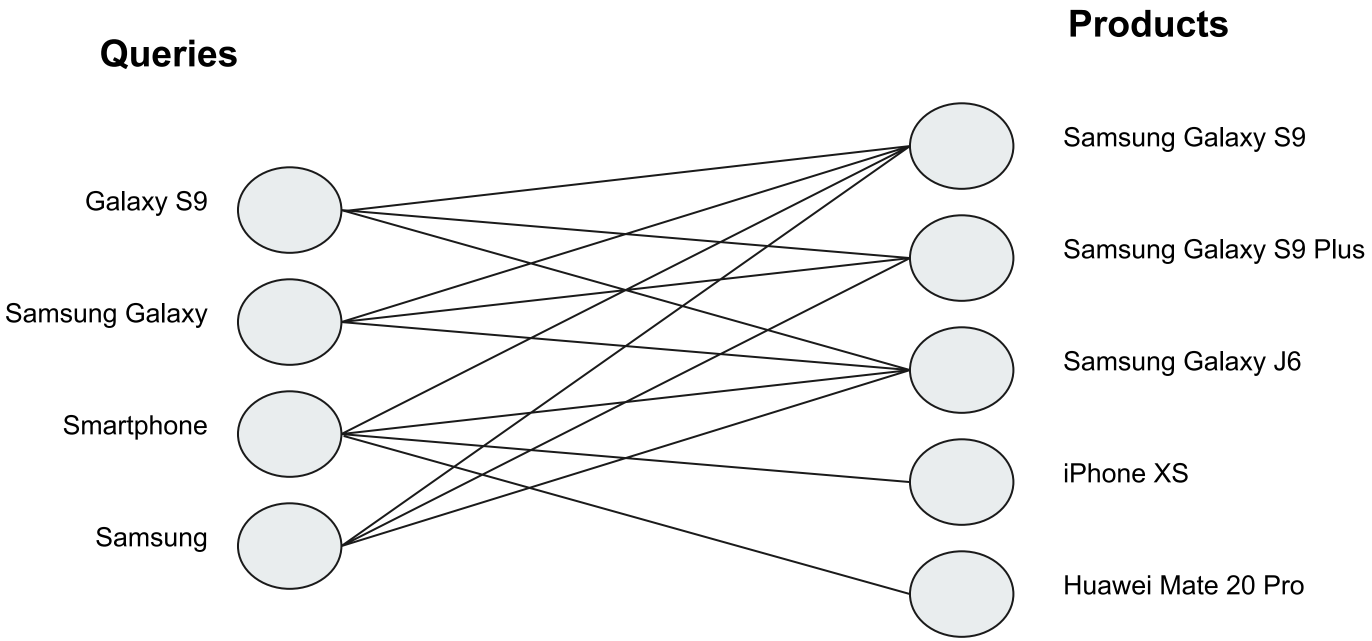
Click Graph (SIGIR 2016)

Click Graph (SIGIR 2016)



Bipartite graph

Click Graph (SIGIR 2016)

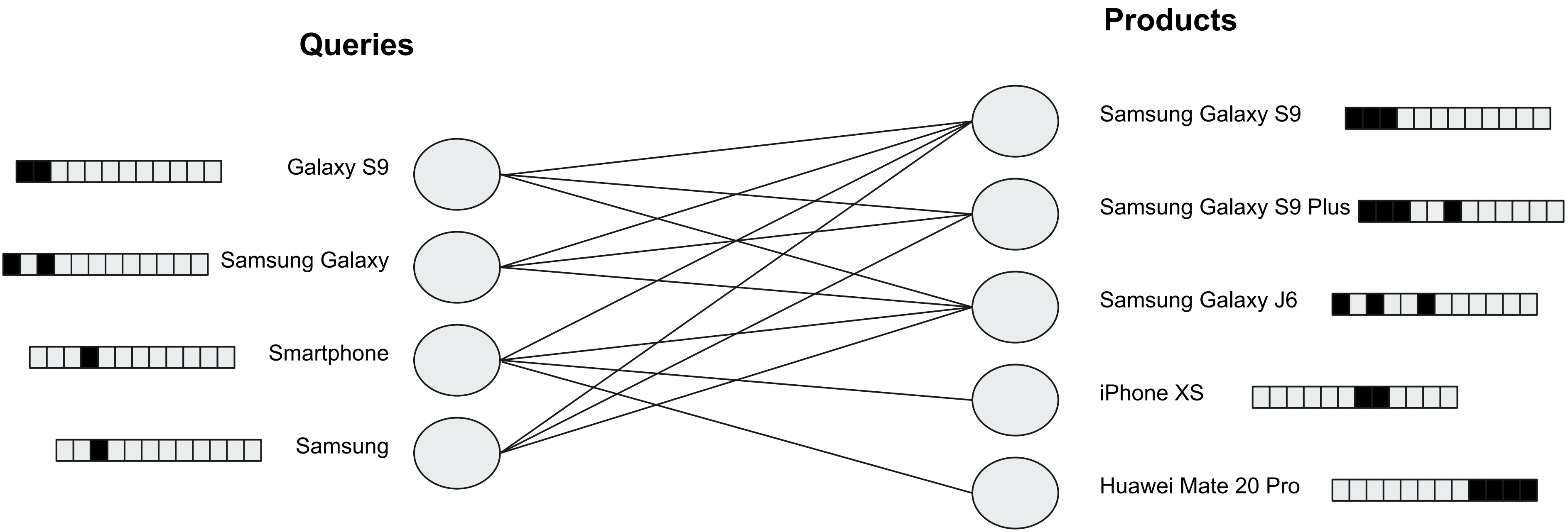


Feature vector

['Galaxy', 'S9', 'Samsung', 'Smartphone', 'Plus', 'J6', 'iPhone', 'XS', 'Huawei', 'Mate', '20', 'Pro']

|V| = 12

Click Graph (SIGIR 2016)

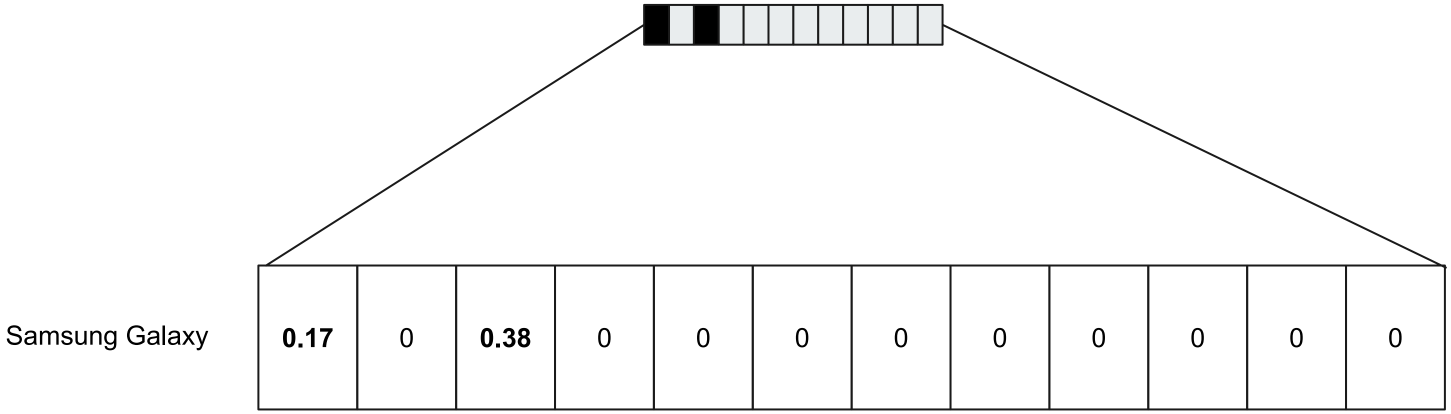


Feature vector

['Galaxy', 'S9', 'Samsung', 'Smartphone', 'Plus', 'J6', 'iPhone', 'XS', 'Huawei', 'Mate', '20', 'Pro']

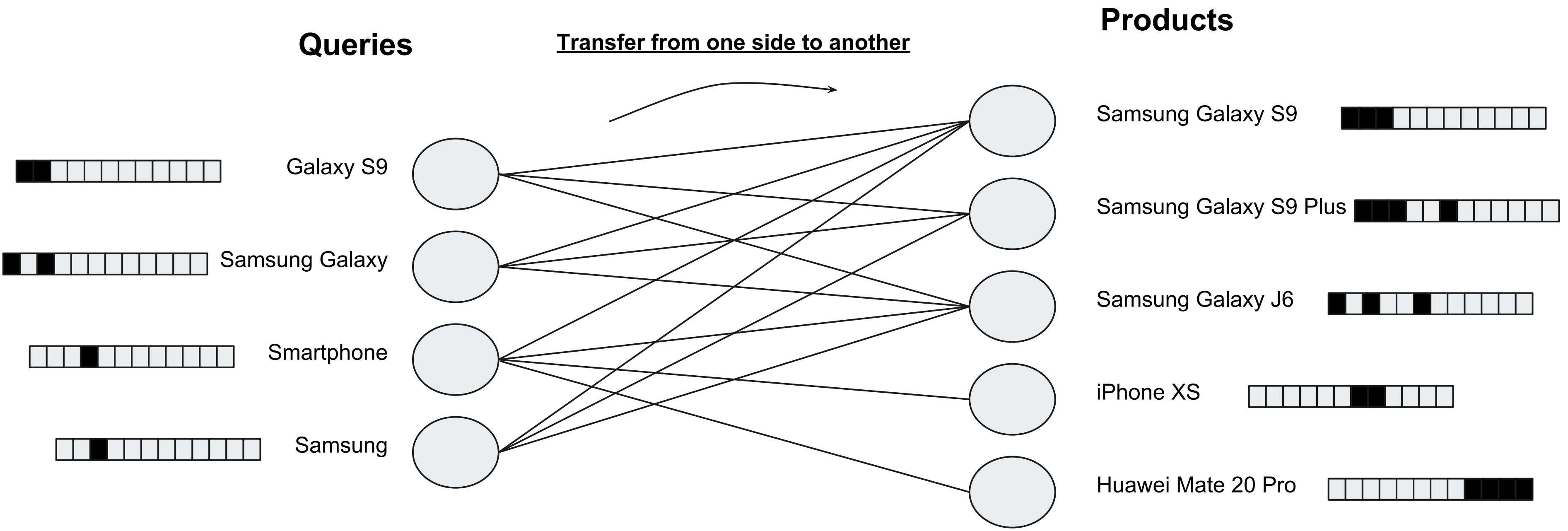
|V| = 12

Click Graph (SIGIR 2016)



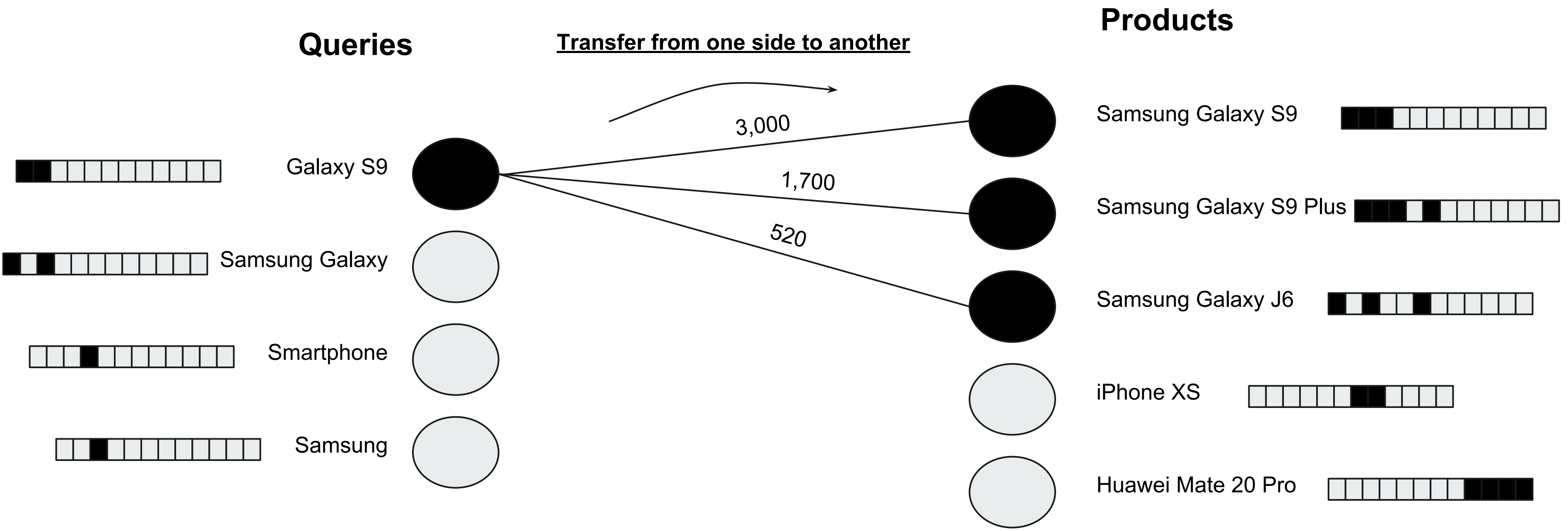
Tf-IDF Bag-of-words vector

Click Graph (SIGIR 2016)



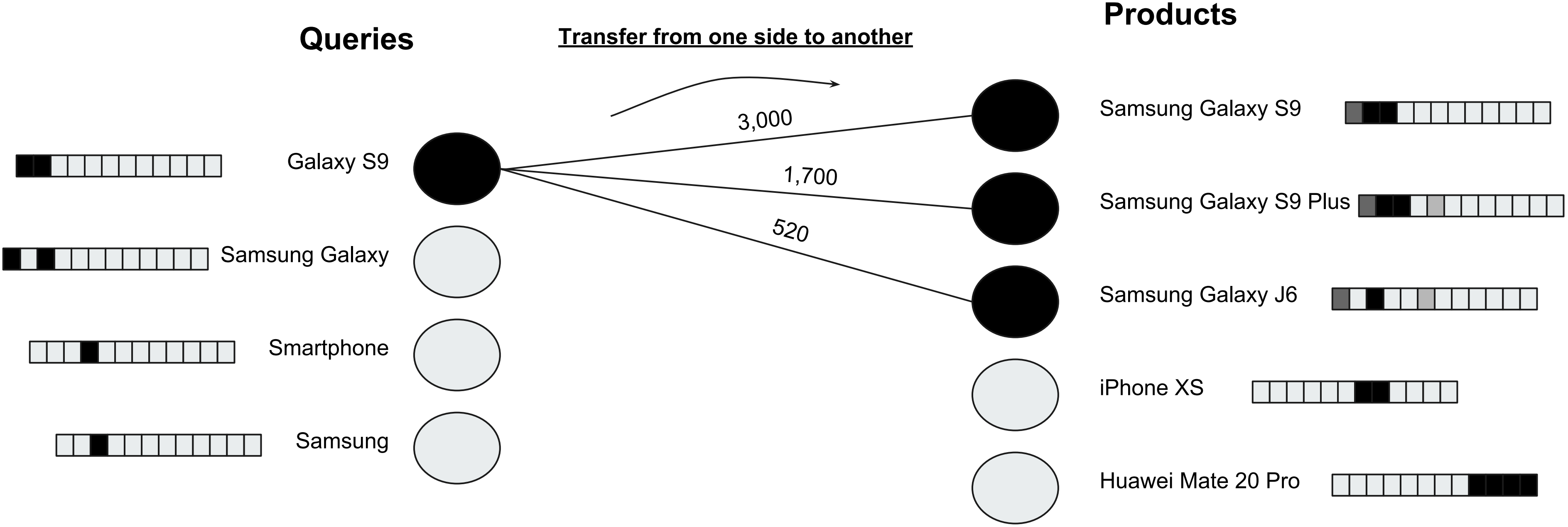
$$D_j^{(n)} = \frac{1}{||\sum_{i=1}^{|Query|} C_{i,j} \cdot Q_i^{(n-1)}||_2} \sum_{i=1}^{|Query|} C_{i,j} \cdot Q_i^{(n-1)}$$

Click Graph (SIGIR 2016)



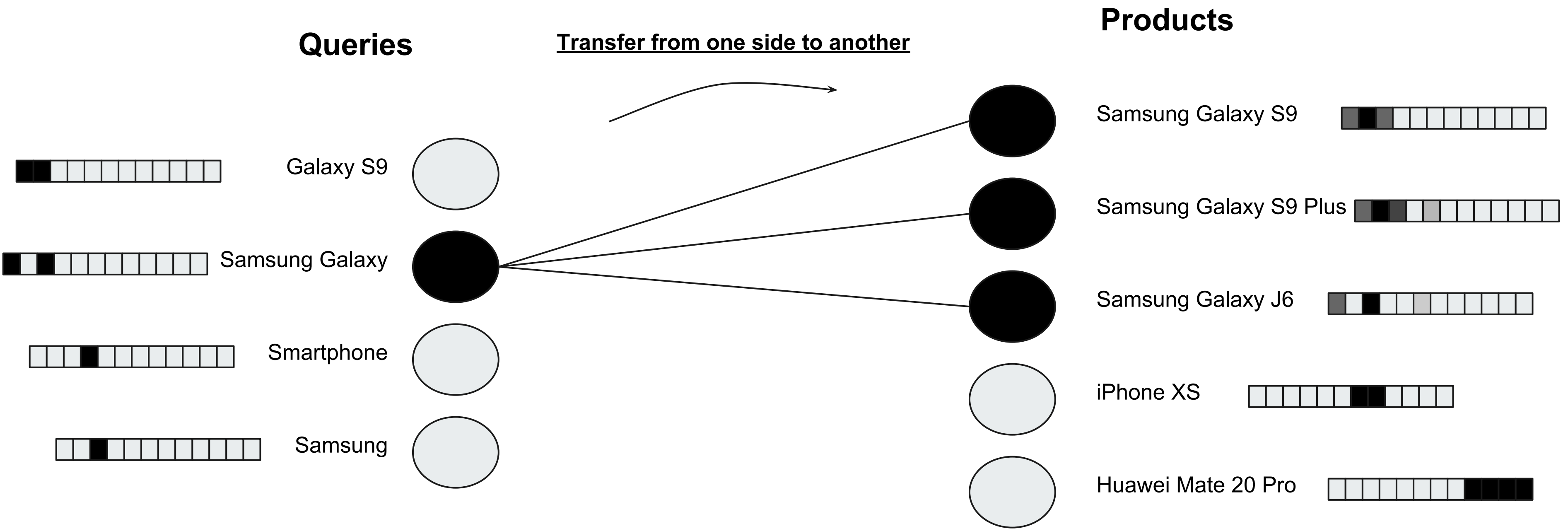
$$D_j^{(n)} = \frac{1}{||\sum_{i=1}^{|Query|} C_{i,j} \cdot Q_i^{(n-1)}||_2} \sum_{i=1}^{|Query|} C_{i,j} \cdot Q_i^{(n-1)}$$

Click Graph (SIGIR 2016)



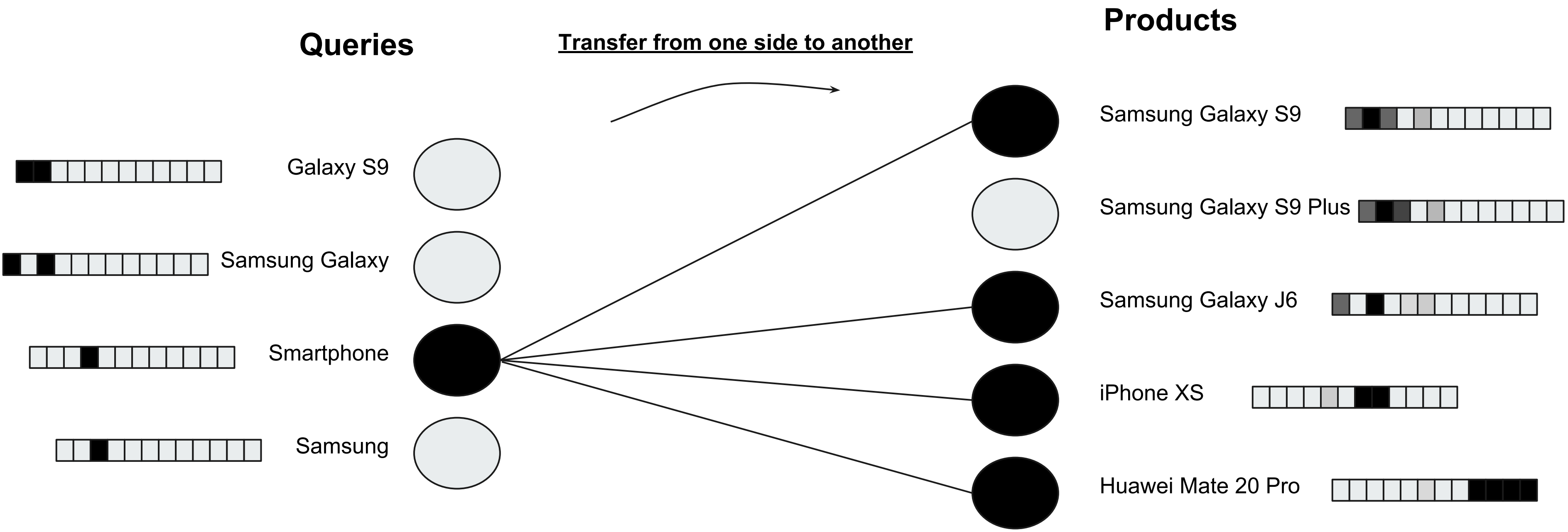
$$D_j^{(n)} = \frac{1}{||\sum_{i=1}^{|Query|} C_{i,j} \cdot Q_i^{(n-1)}||_2} \sum_{i=1}^{|Query|} C_{i,j} \cdot Q_i^{(n-1)}$$

Click Graph (SIGIR 2016)



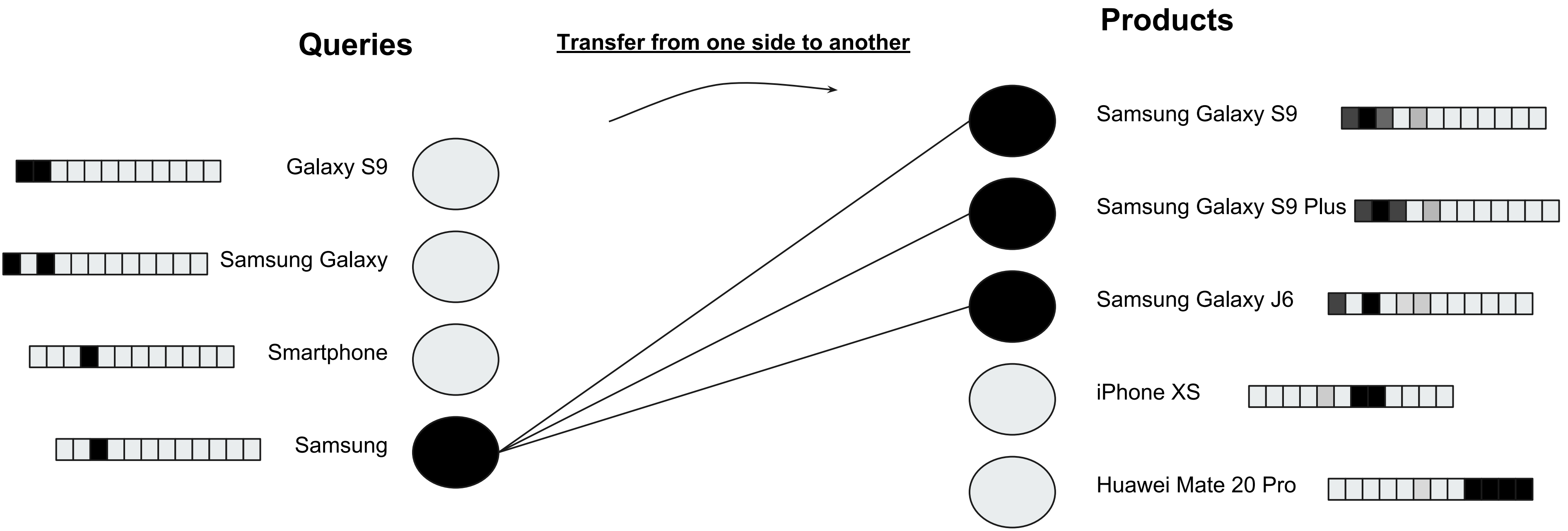
$$D_j^{(n)} = \frac{1}{||\sum_{i=1}^{|Query|} C_{i,j} \cdot Q_i^{(n-1)}||_2} \sum_{i=1}^{|Query|} C_{i,j} \cdot Q_i^{(n-1)}$$

Click Graph (SIGIR 2016)



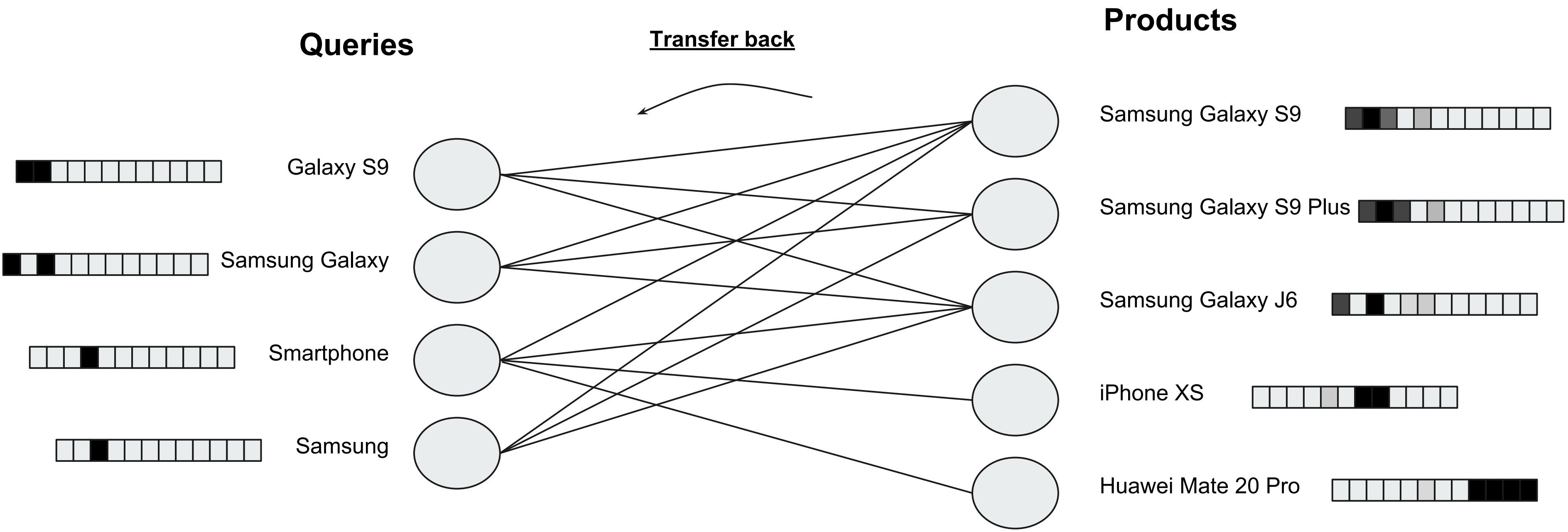
$$D_j^{(n)} = \frac{1}{||\sum_{i=1}^{|Query|} C_{i,j} \cdot Q_i^{(n-1)}||_2} \sum_{i=1}^{|Query|} C_{i,j} \cdot Q_i^{(n-1)}$$

Click Graph (SIGIR 2016)



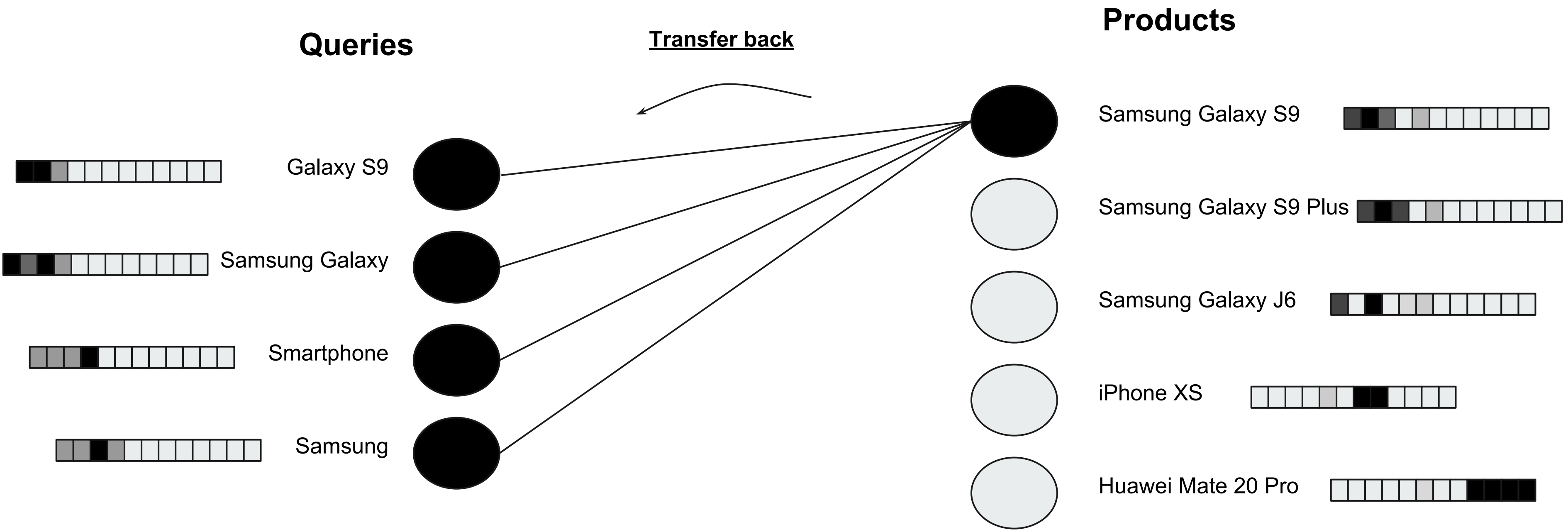
$$D_j^{(n)} = \frac{1}{||\sum_{i=1}^{|Query|} C_{i,j} \cdot Q_i^{(n-1)}||_2} \sum_{i=1}^{|Query|} C_{i,j} \cdot Q_i^{(n-1)}$$

Click Graph (SIGIR 2016)



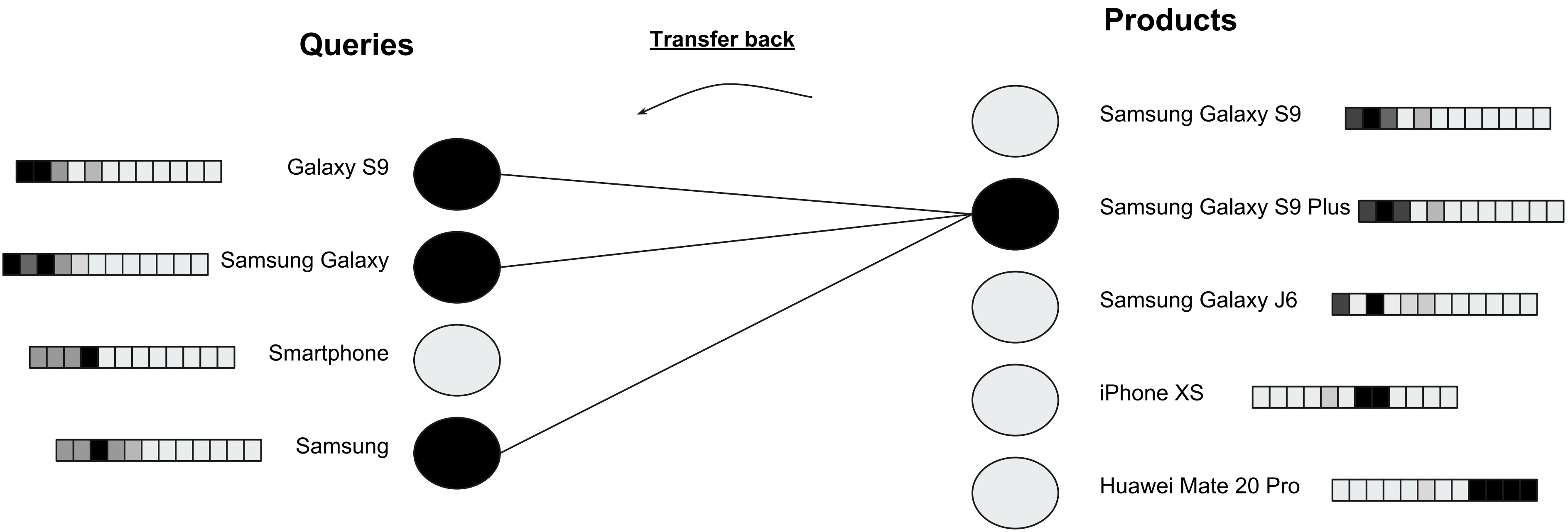
$$Q_i^{(n)} = \frac{1}{\|\sum_j^{|\mathcal{D}oc|} C_{i,j} \cdot D_j^{(n)}\|_2} \sum_{j=1}^{|\mathcal{D}oc|} C_{i,j} \cdot D_j^{(n)}$$

Click Graph (SIGIR 2016)



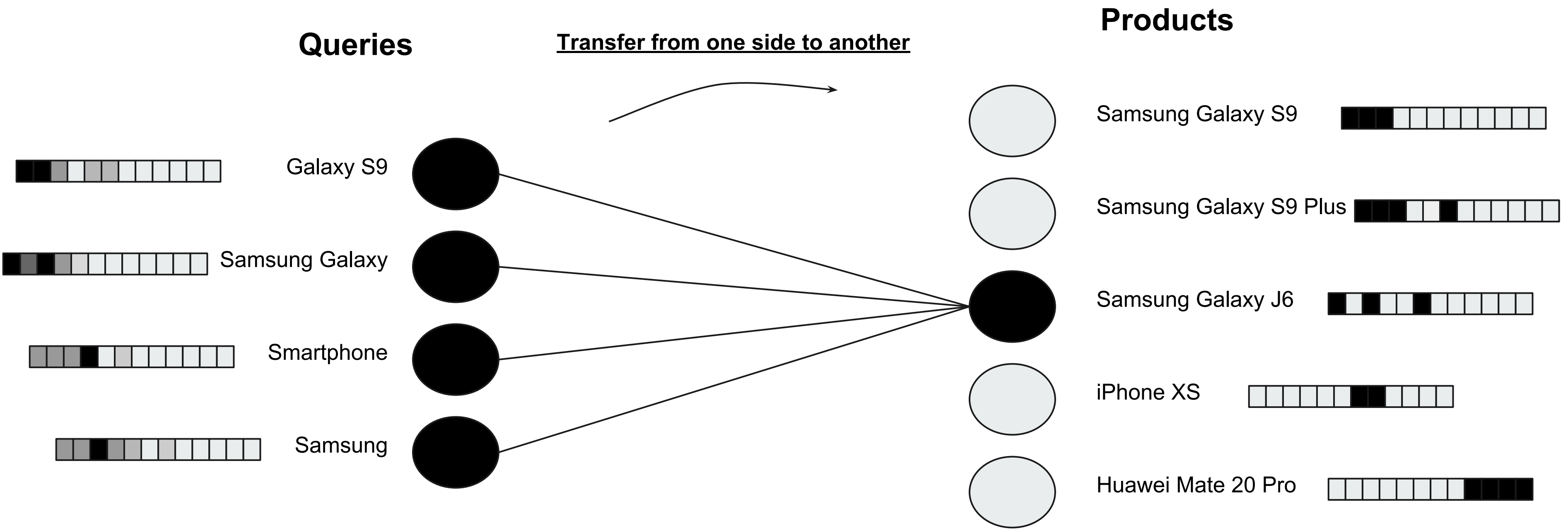
$$Q_i^{(n)} = \frac{1}{||\sum_j^{|\mathcal{D}oc|} C_{i,j} \cdot D_j^{(n)}||_2} \sum_{j=1}^{|\mathcal{D}oc|} C_{i,j} \cdot D_j^{(n)}$$

Click Graph (SIGIR 2016)



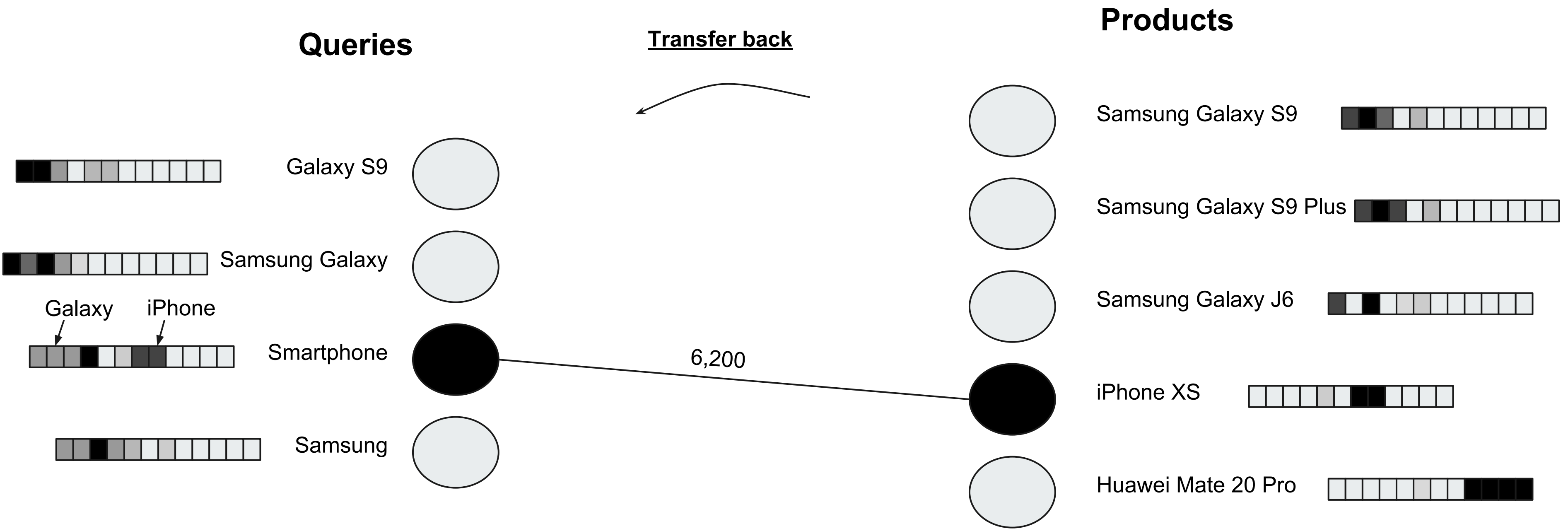
$$Q_i^{(n)} = \frac{1}{||\sum_j^{|\mathcal{D}oc|} C_{i,j} \cdot D_j^{(n)}||_2} \sum_{j=1}^{|\mathcal{D}oc|} C_{i,j} \cdot D_j^{(n)}$$

Click Graph (SIGIR 2016)



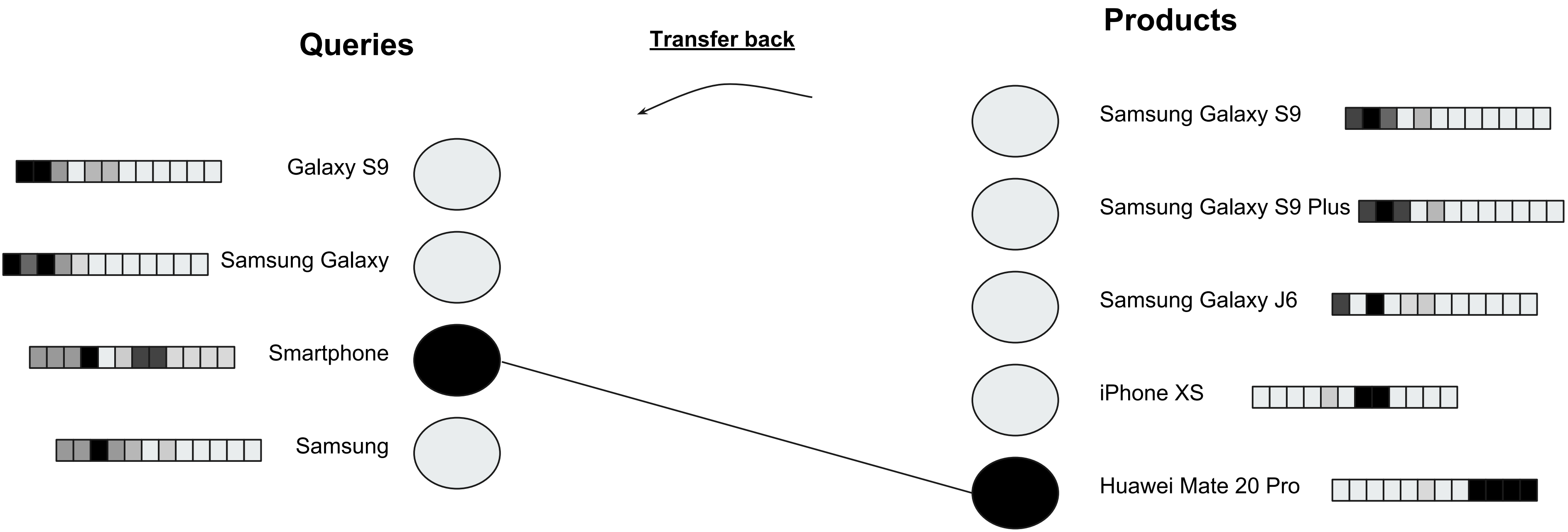
$$D_j^{(n)} = \frac{1}{||\sum_{i=1}^{|Query|} C_{i,j} \cdot Q_i^{(n-1)}||_2} \sum_{i=1}^{|Query|} C_{i,j} \cdot Q_i^{(n-1)}$$

Click Graph (SIGIR 2016)



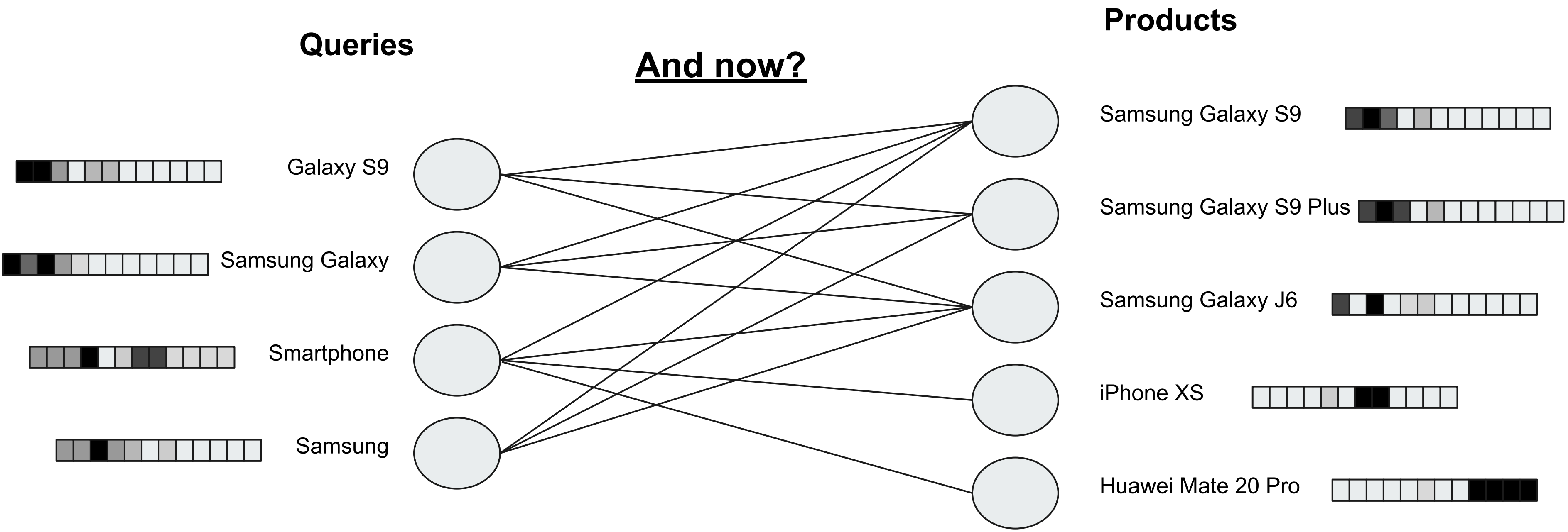
$$Q_i^{(n)} = \frac{1}{|| \sum_j^{|\mathcal{D}oc|} C_{i,j} \cdot D_j^{(n)} ||_2} \sum_{j=1}^{|\mathcal{D}oc|} C_{i,j} \cdot D_j^{(n)}$$

Click Graph (SIGIR 2016)



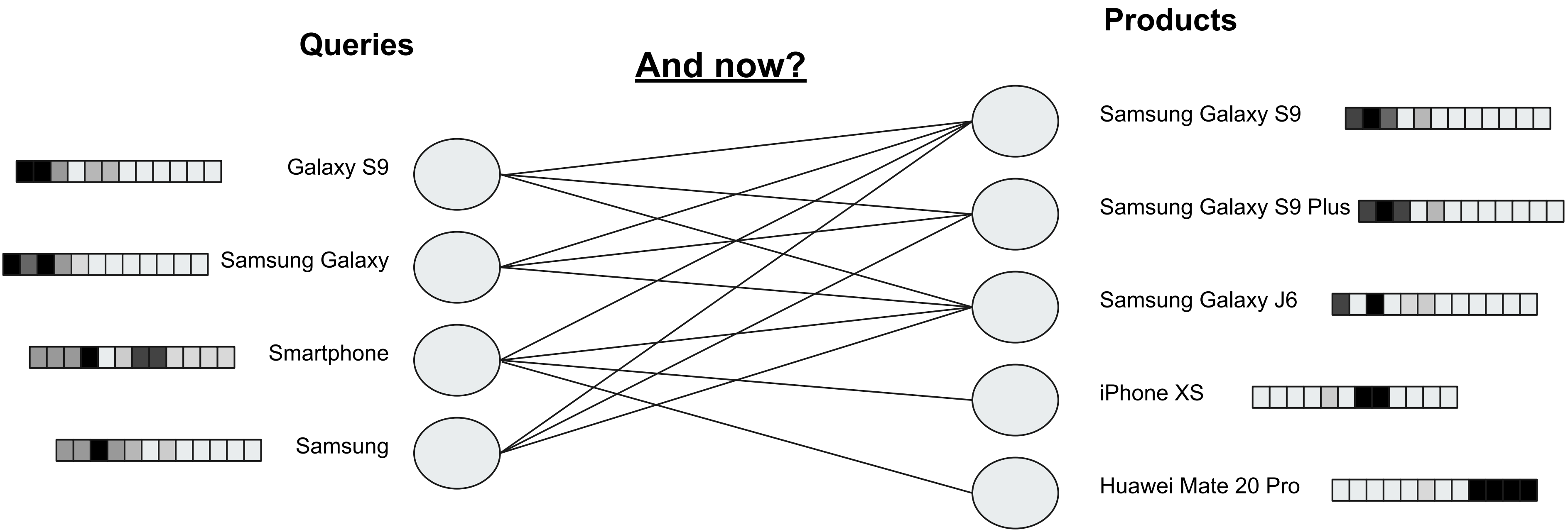
$$Q_i^{(n)} = \frac{1}{||\sum_j^{|\mathcal{D}oc|} C_{i,j} \cdot D_j^{(n)}||_2} \sum_{j=1}^{|\mathcal{D}oc|} C_{i,j} \cdot D_j^{(n)}$$

Click Graph (SIGIR 2016)



Click Graph (SIGIR 2016)

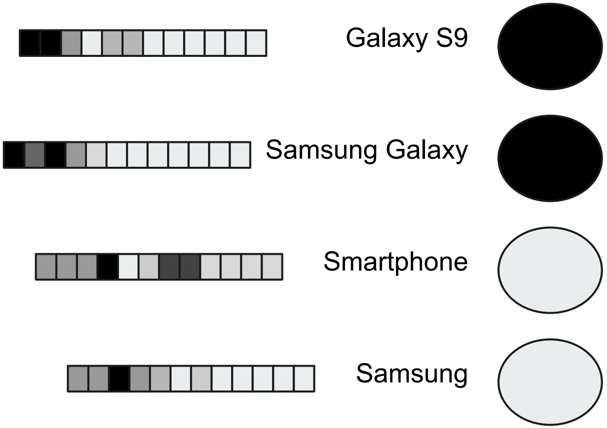
And now?



Repeat both transfers again up to convergence!

How to include new queries?

Queries



Galaxy S10 ?

Split the new query into word units and calculate a linear combination of known queries with these words

$$q_v = \sum_{u_i \in \mathcal{U}_q} W_i U_i$$

Feature vector

['Galaxy', 'S9', 'Samsung', 'Smartphone', 'Plus', 'J6', 'iPhone', 'XS', 'Huawei', 'Mate', '20', 'Pro']

|V| = 12

How to retrieve products given a query?

- After **convergence**, both **queries** and **products** vectors are in the **same vector space**.

- For retrieving products, just compute the cosine similarity between vectors.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}},$$

Click Graph (SIGIR 2016)

Table 1: Performance as an individual ranking model.

Feature	NDCG@1	NDCG@3	NDCG@5	NDCG@10
BM25SD	0.4373	0.4937	0.5460	0.6542
BM25SD_MULT	0.6132	0.6346	0.6668	0.7464
CTR	0.5769	0.5941	0.6238	0.7064
WMD	0.4585	0.5145	0.5641	0.6674
VPCG_QUERY	0.6268	0.6498	0.6797	0.7509
VPCG&VG_QUERY	**0.6344***	**0.6618***	**0.6948***	**0.7687***
VPCG_DOC	0.5648	0.6209	0.6623	0.7382
VPCG&VG_DOC	0.5668	0.6268	0.6717	0.7509

Two-tailed t-test is done for paired data where each pair is VPCG&VG_QUERY and any of the other methods, and * indicates *p-value* < 0.01 for all tests.

Shan Jiang, Yuening Hu, Changsung Kang, Tim Daly, Jr., Dawei Yin, Yi Chang, and Chengxiang Zhai. 2016. Learning Query and Document Relevance from a Web-scale Click Graph. In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '16). ACM, New York, NY, USA, 185-194.

Click Graph (SIGIR 2016)

Pros

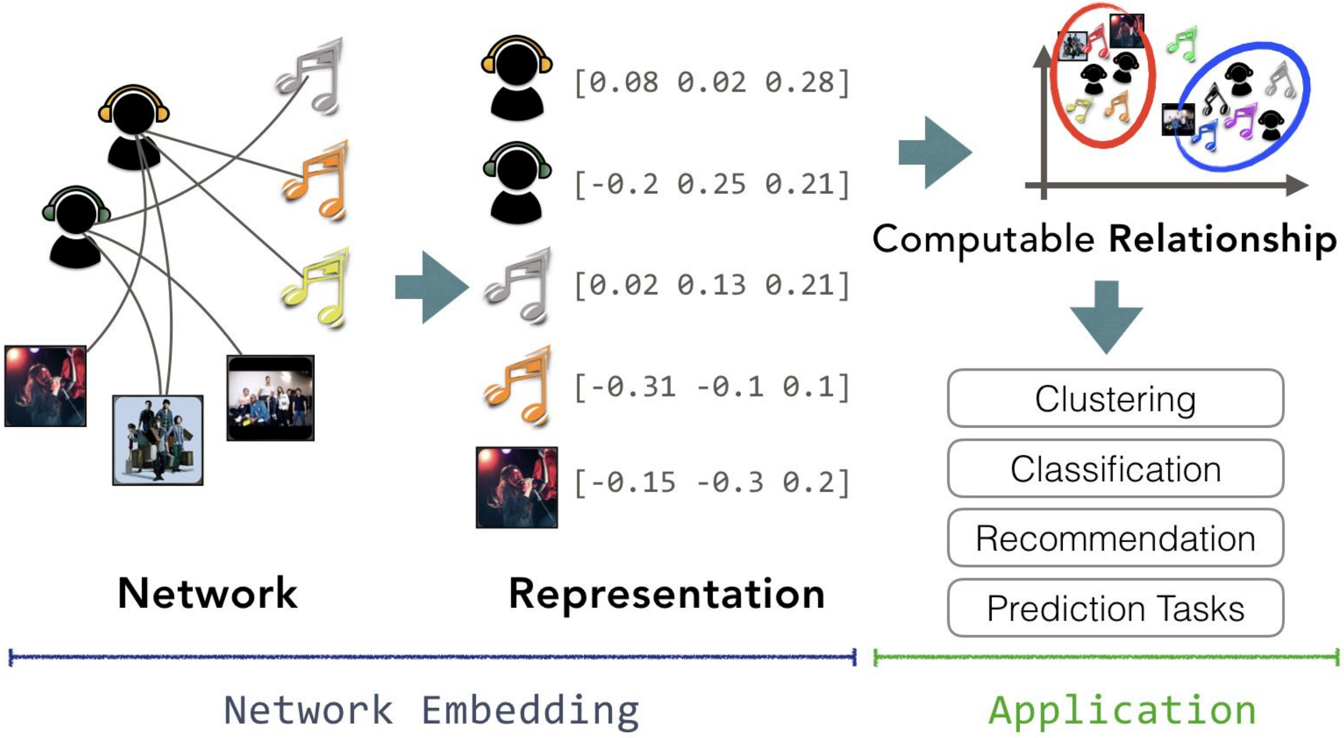
- Maximization for **CTR** (Click Through Rate)
- **Scalable** approach
- Improve **coverage**

Cons

- Sparse representation (based on bag-of-words)
- Too much focus on CTR

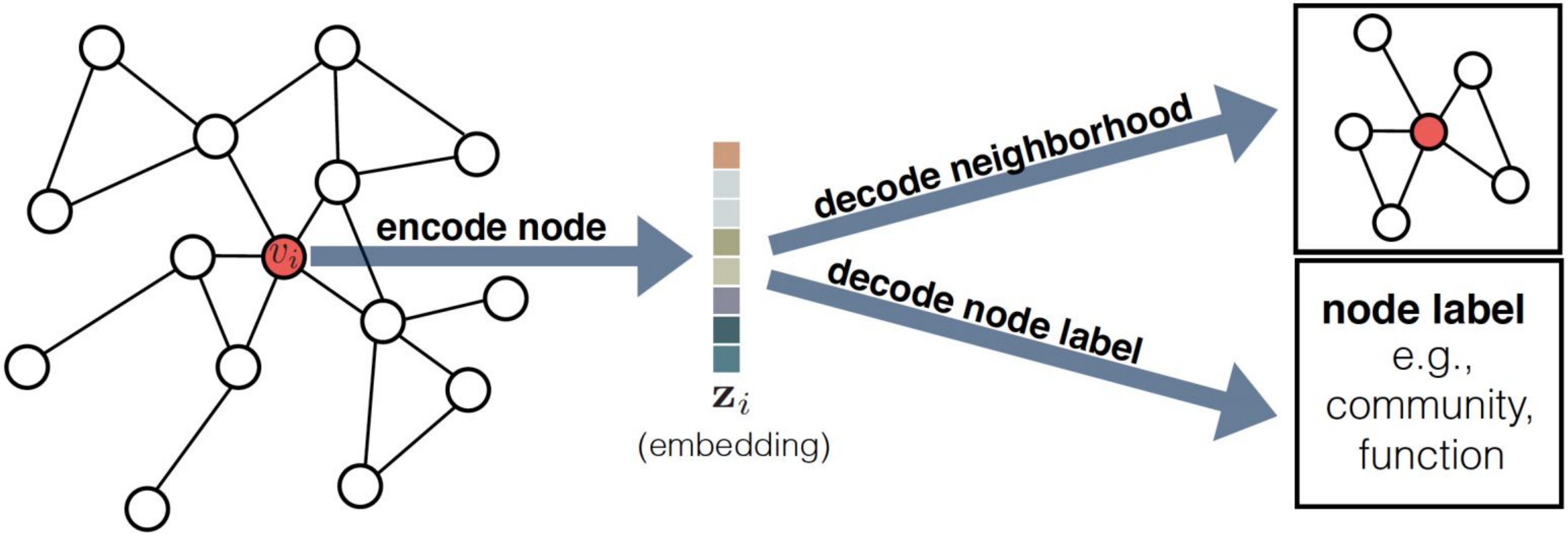
A new era: Neural Approaches

Why Neural Graphs?



Source: https://github.com/chihming/awesome-network-embedding

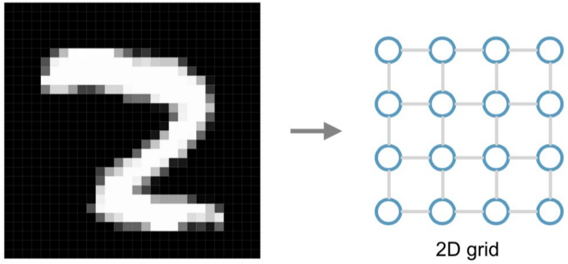
Representation Learning for Graphs

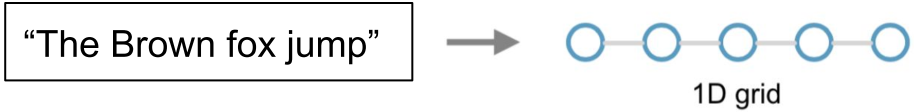


Source: Hamilton, William L., Rex Ying, and Jure Leskovec. "Representation learning on graphs: Methods and applications." *arXiv preprint arXiv:1709.05584* (2017).

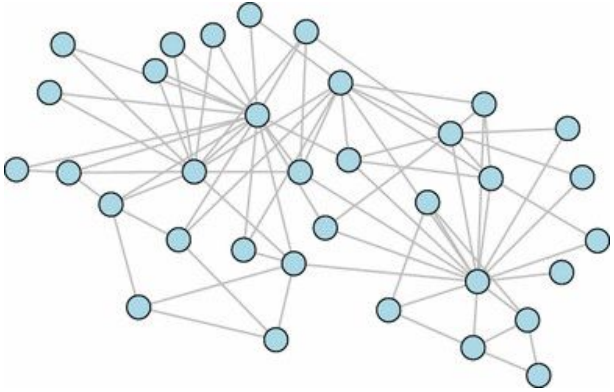
Why Representation Learning for Graphs is hard?

- Text and Images have a **fixed grid structure**





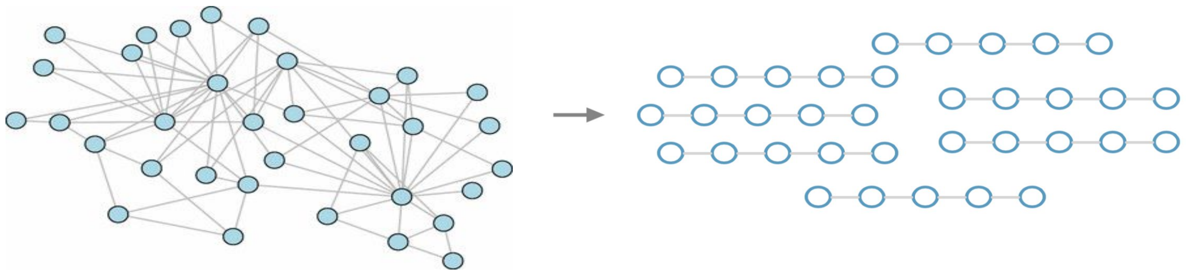
- Graphs are **non-euclidian**!



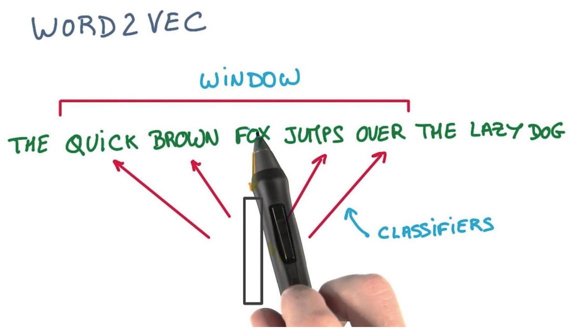
Source: Hamilton, William L., Rex Ying, and Jure Leskovec. “Tutorial Representation Learning on Networks”. The Web Conference, 2018

DeepWalk (KDD 2014)

- **Solution**: **Linearization** of graph network by extracting a corpus of “sentences” (walks)
  - Random Walks!



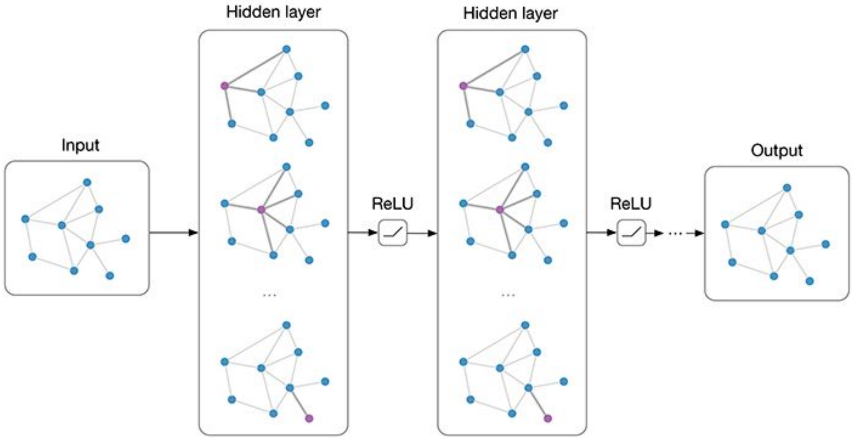
- DeepWalk: apply **Skip-Gram** over the “sentences” to extract node embeddings



Source: "Deep Learning". Udacity https://www.udacity.com/course/ud730

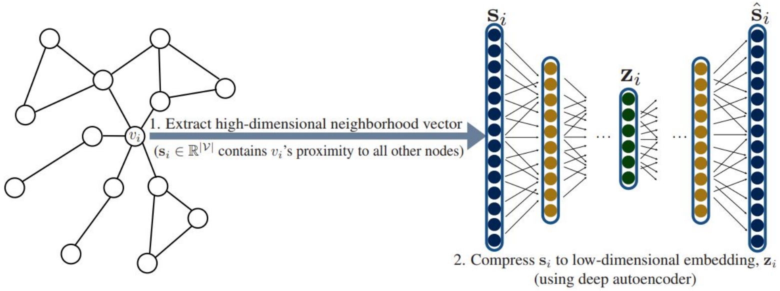
Other approaches

- **Graph Convolutional Networks**



Source: Thomas Kipf. Graph Convolutional Networks. https://tkipf.github.io/graph-convolutional-networks/

- **Autoencoders**



Source: Hamilton, William L., Rex Ying, and Jure Leskovec. "Representation learning on graphs: Methods and applications." *arXiv preprint arXiv:1709.05584* (2017).

Other approaches

- 60+ different methods
  - 25+ only in 2018!
  - Trending topic

- More Info:
  - WWW-18 Tutorial Representation Learning on Networks
  - Awesome Network Embedding github repo

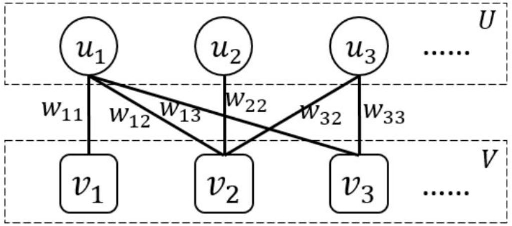
BiNE (SIGIR 2018)

- Paper: **BiNE: Bipartite Network Embedding**

- **Motivation**
  - No previous work focused on **bipartite graphs** (recommendations, search logs, etc)
  - Model both **explicit** (observed links) and **implicit information** (unobserved but transitive links)
- **Approach**
  - Learn vector representation based on both content and click information

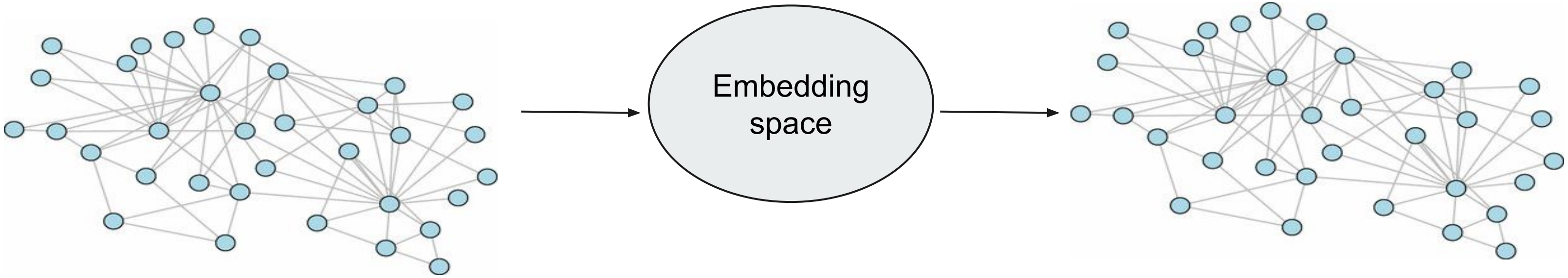
Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. 2018. BiNE: Bipartite Network Embedding. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18). ACM, New York, NY, USA, 715-724.

BiNE (SIGIR 2018)



Explicit relations

- A good network embedding should be capable of **reconstructing** the original network!

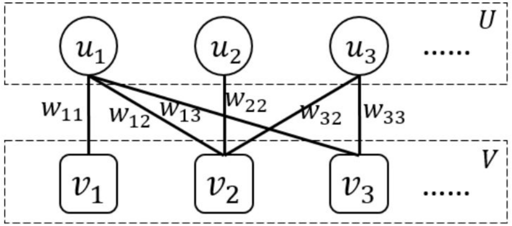


$$P(i, j) = \frac{w_{ij}}{\sum_{e_{ij} \in E} w_{ij}}.$$

$$\hat{P}(i, j) = \frac{1}{1 + exp(-\vec{\mathbf{u}}_i{}^T \vec{\mathbf{v}}_j)}$$

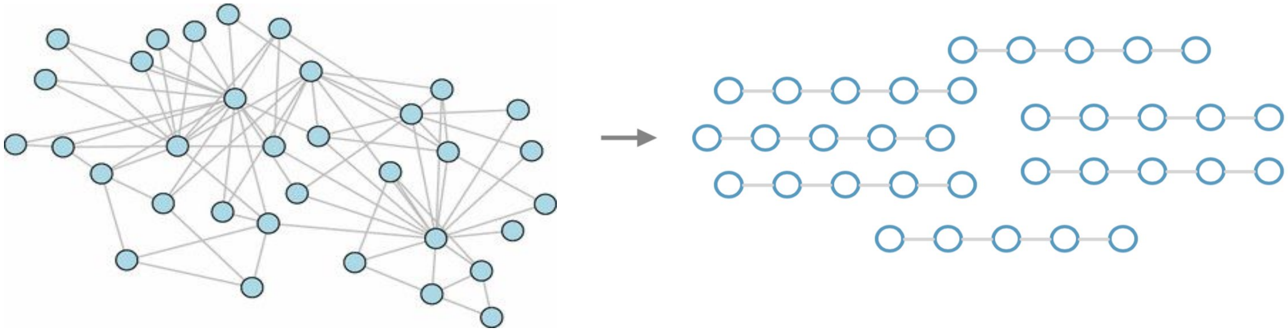
$$minimize \quad O_1 = KL(P||\hat{P}) = \sum_{e_{ij} \in E} P(i, j) \log\left(\frac{P(i, j)}{\hat{P}(i, j)}\right)$$

BiNE (SIGIR 2018)

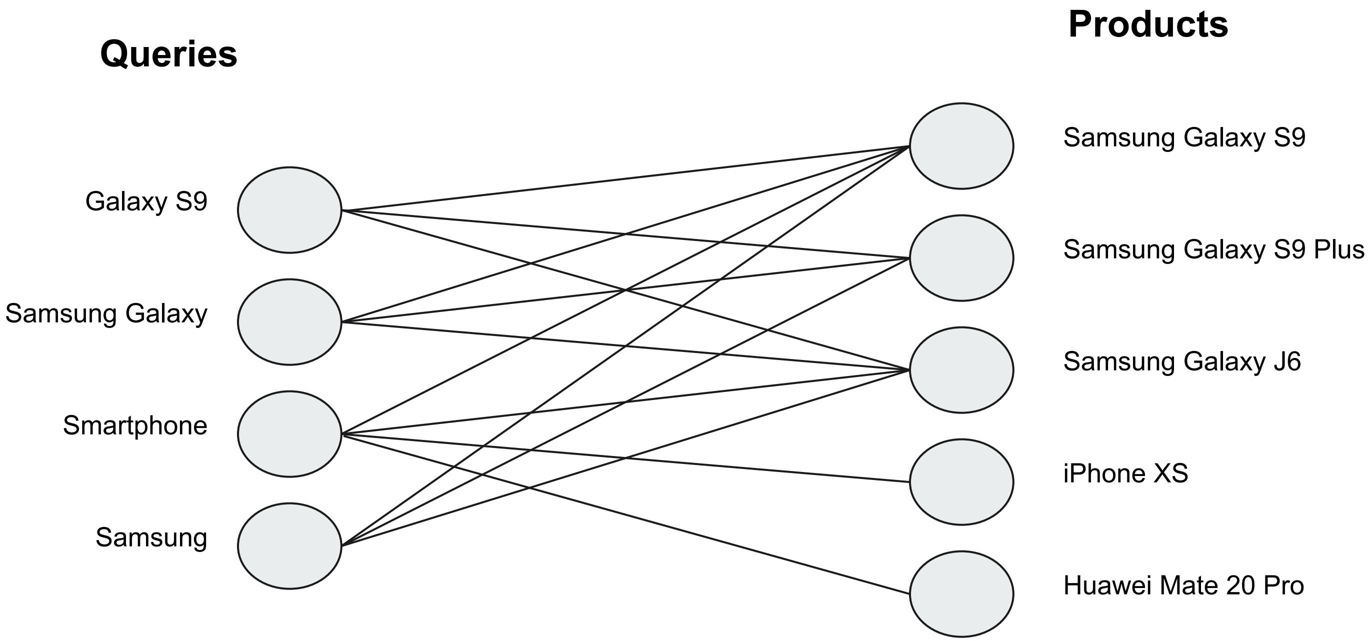


Implicit relations

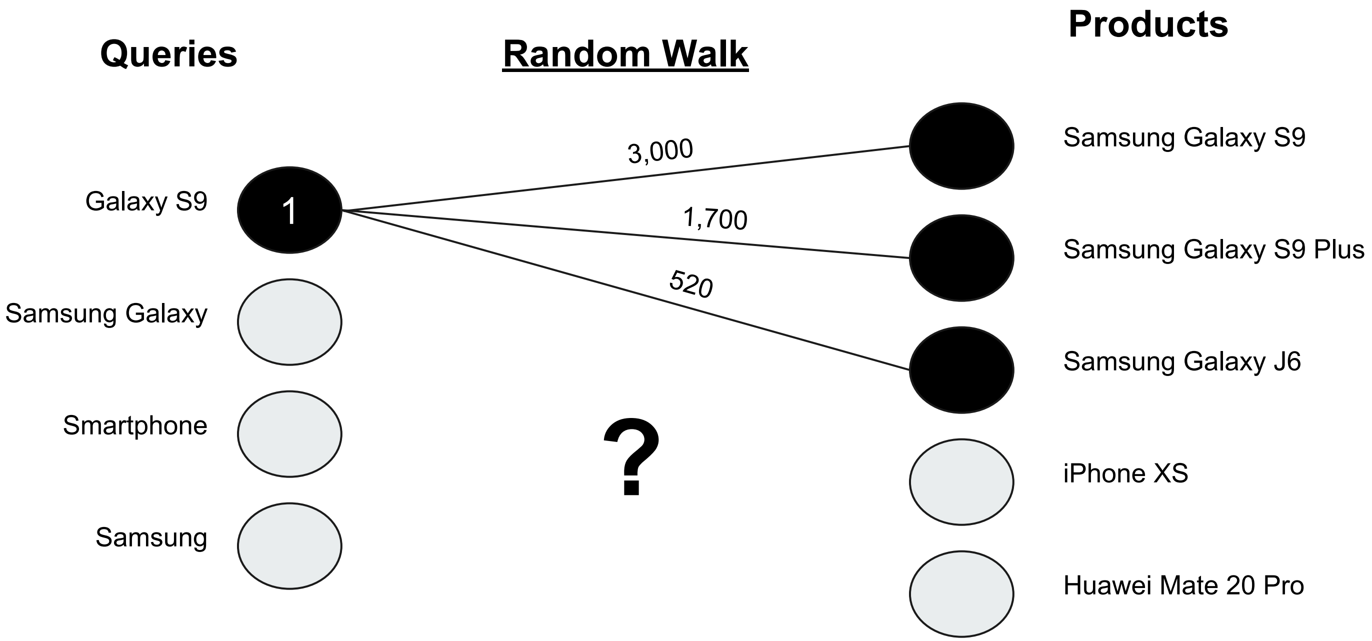
- Too many possible latent combinations
- Let's use random walks



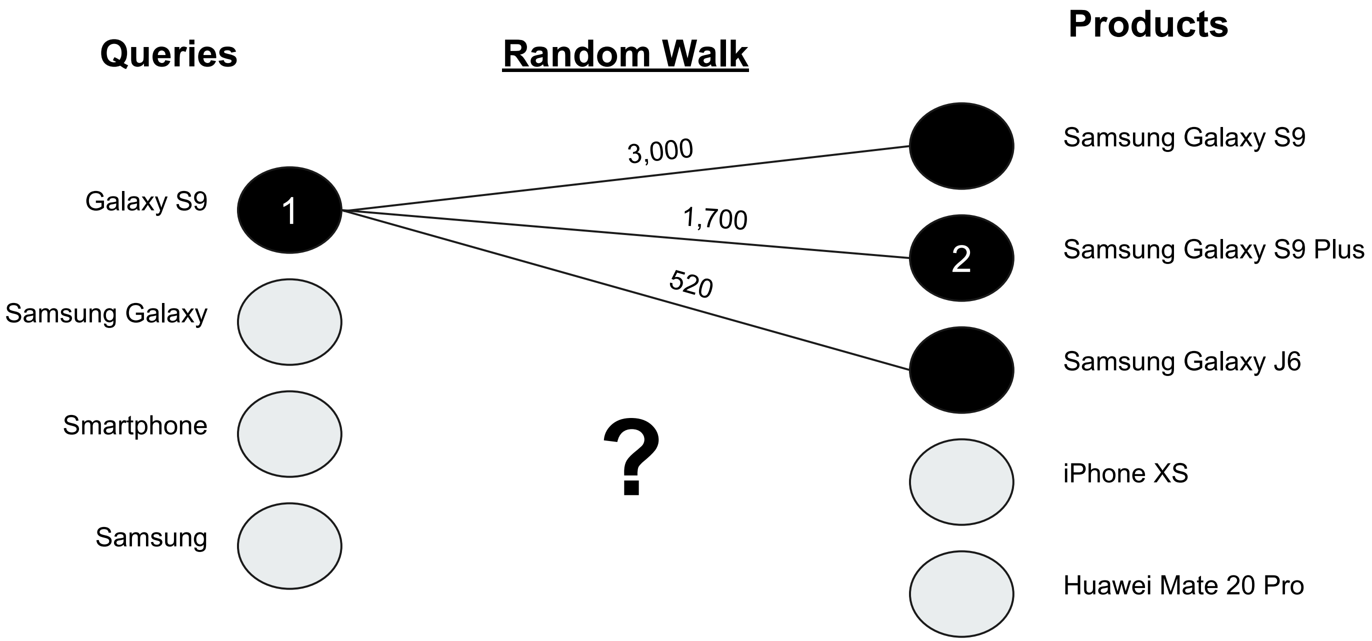
BiNE (SIGIR 2018)



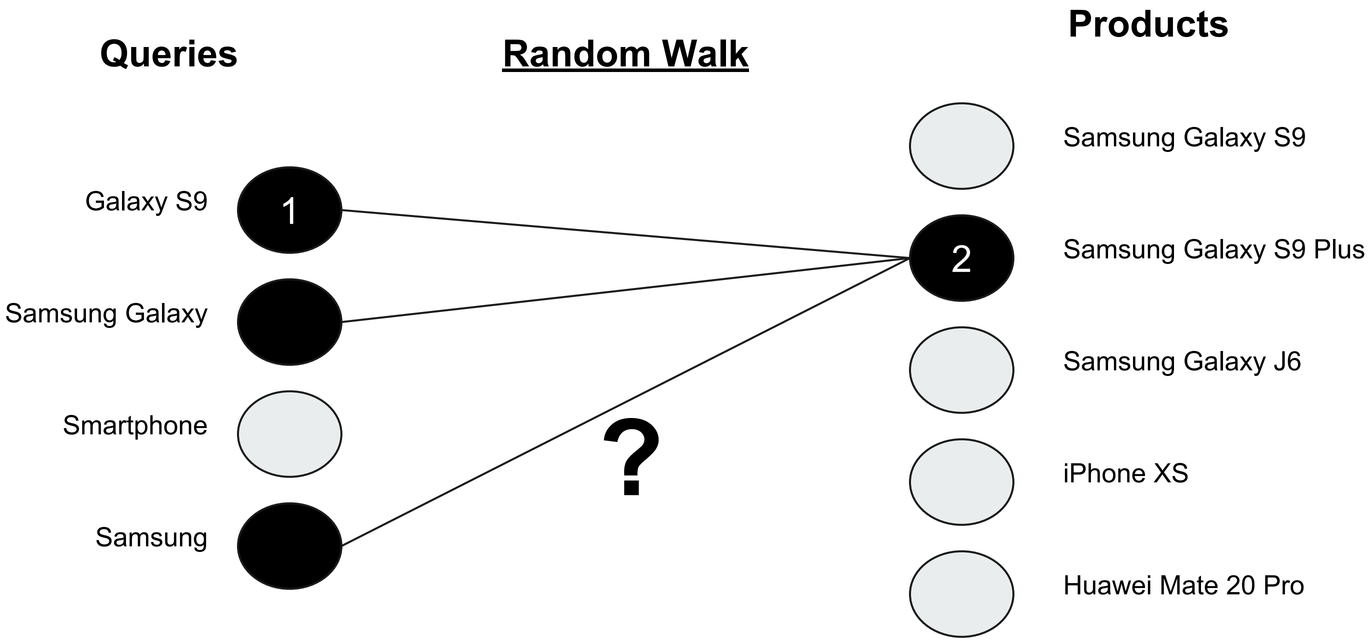
BiNE (SIGIR 2018)



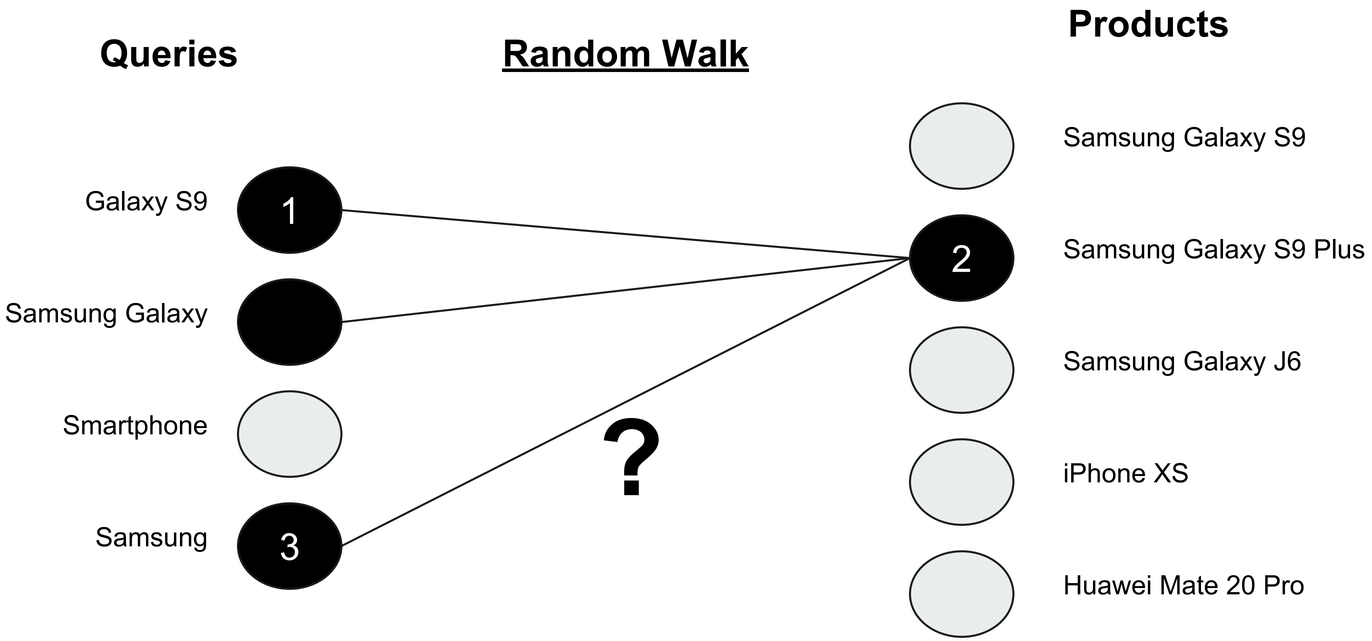
BiNE (SIGIR 2018)



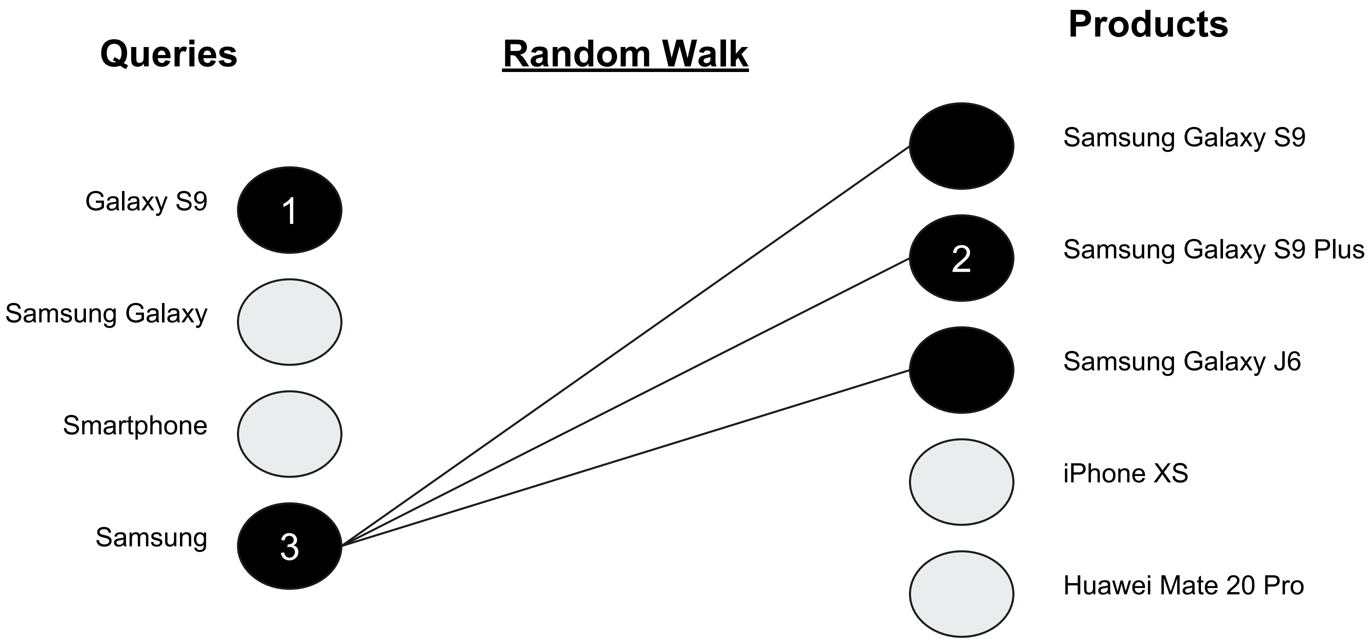
BiNE (SIGIR 2018)



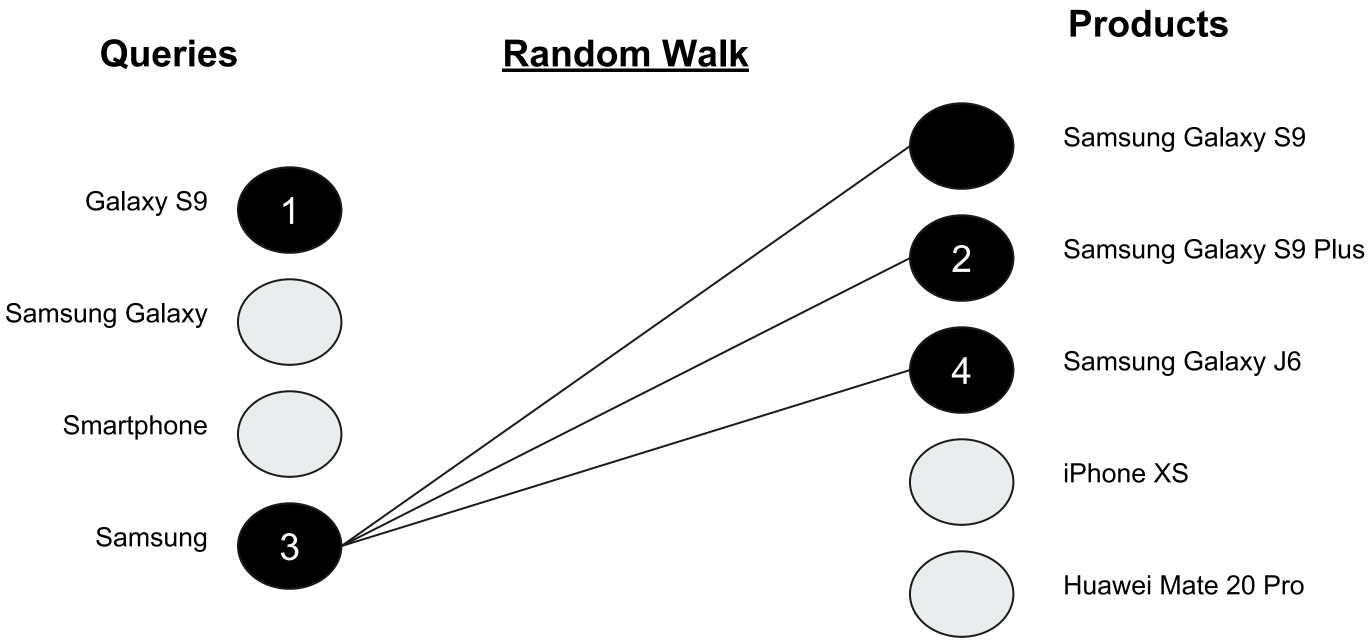
BiNE (SIGIR 2018)



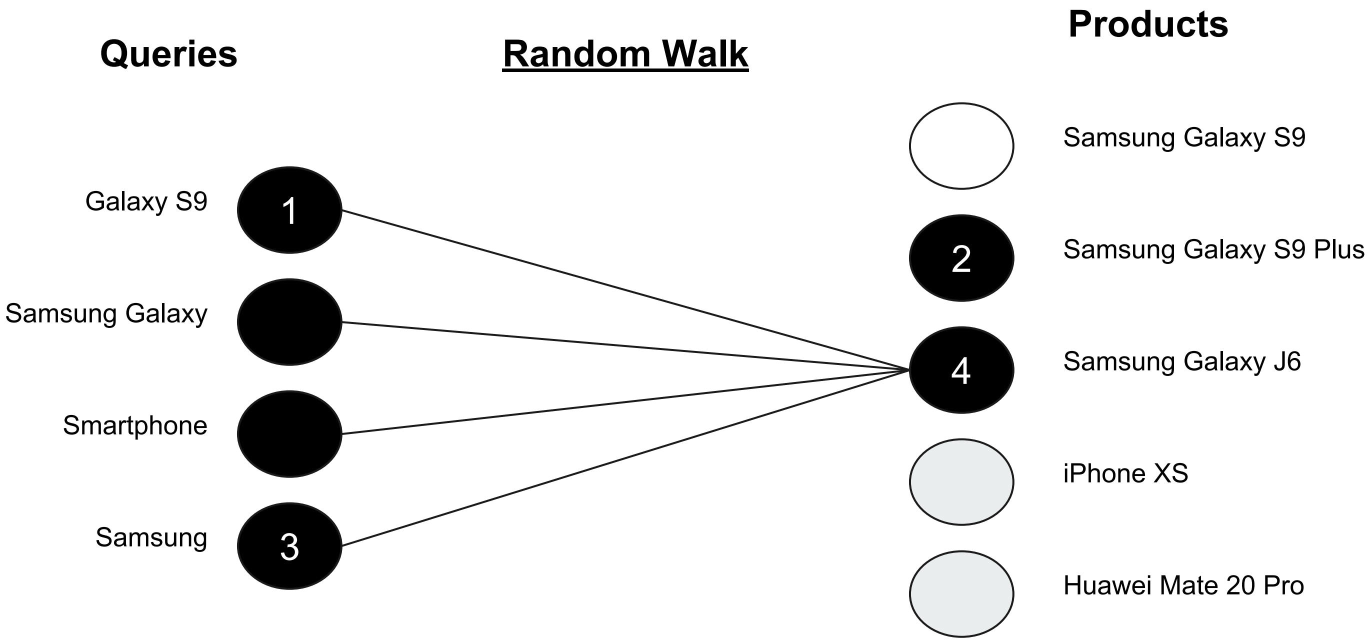
BiNE (SIGIR 2018)



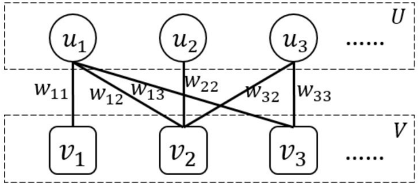
BiNE (SIGIR 2018)



BiNE (SIGIR 2018)



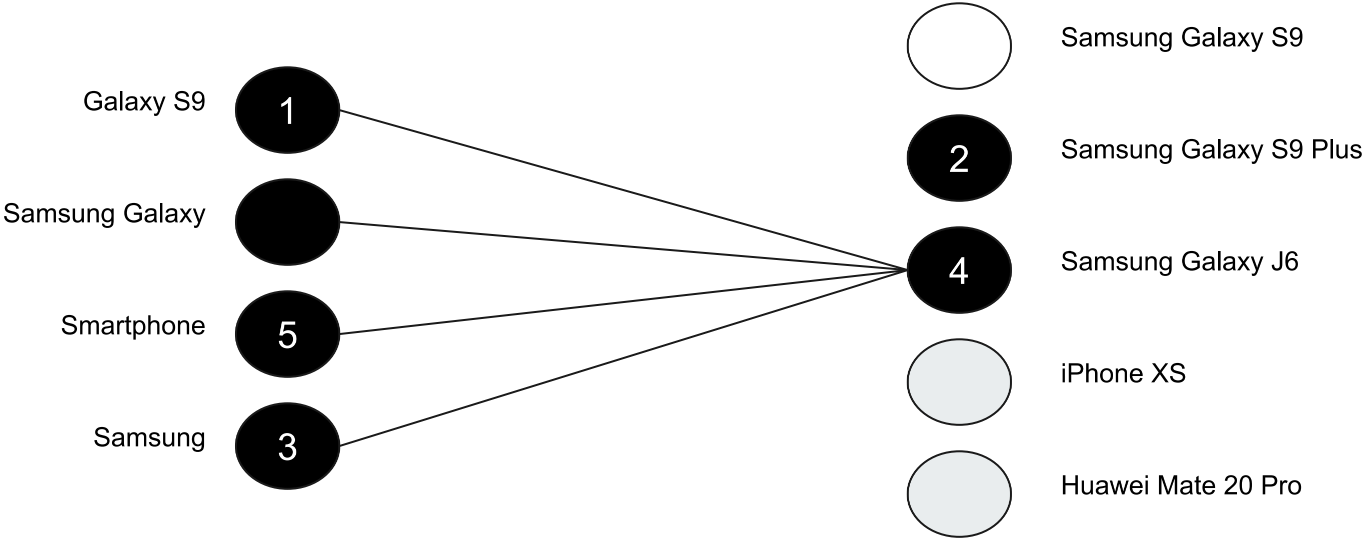
BiNE (SIGIR 2018)



Products

Queries

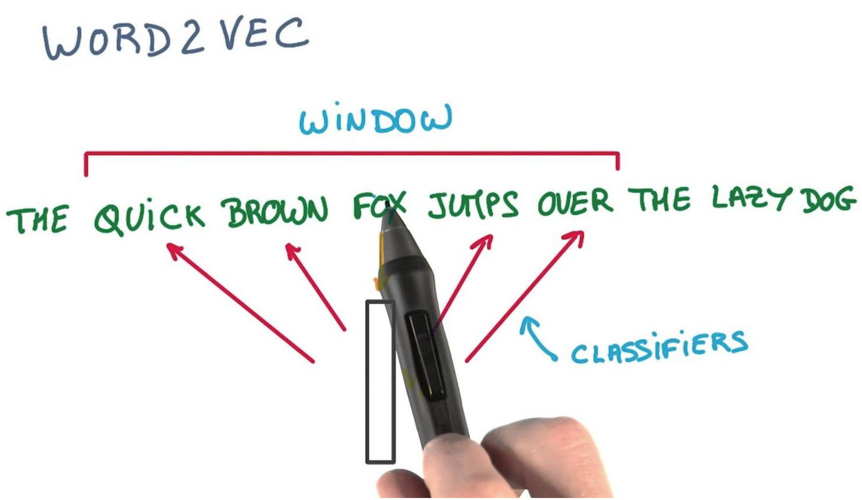
<u>Random Walk</u>

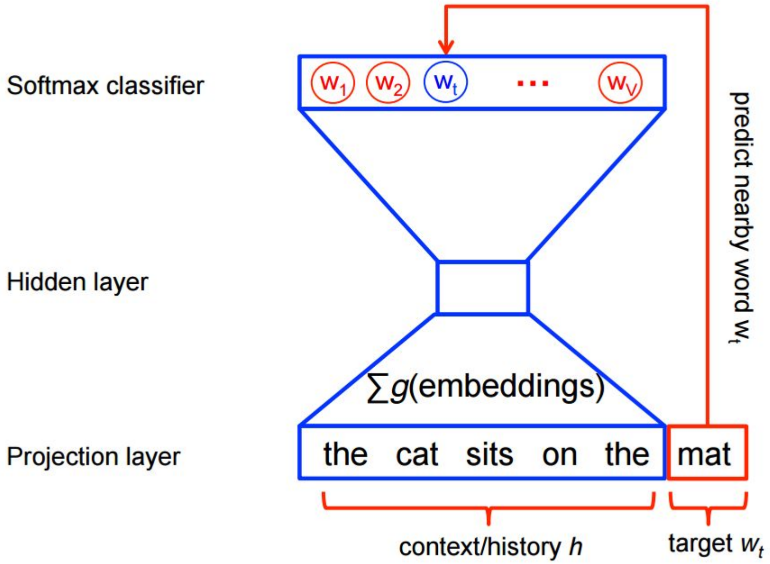


Walk for <u>Queries</u> = ['Galaxy S9', 'Samsung', 'Smartphone']

BiNE (SIGIR 2018)

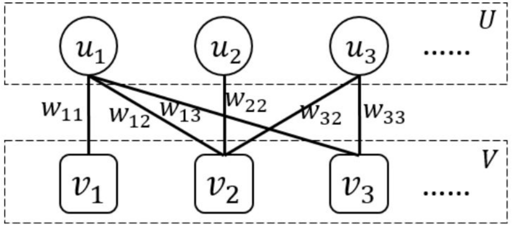
- DeepWalk
  - Collect a corpus of 'sentences' using deep walk
  - **Compute Skip-Gram**





Source: Vector Representations of Words. TensorFlow.
https://www.tensorflow.org/tutorials/representation/word2vec

BiNE (SIGIR 2018)



Implicit relations

- Too many possible latent combinations
- Let's use random walks

Objective function for **implicit** relations in **U**

$$P(u_c|u_i) = \frac{\exp(\vec{\mathbf{u}}_i{}^T \vec{\boldsymbol{\theta}}_c)}{\sum_{k=1}^{|U|} \exp(\vec{\mathbf{u}}_i{}^T \vec{\boldsymbol{\theta}}_k)}$$

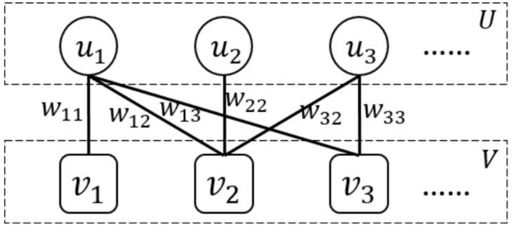
$$maximize\ O_2 = \prod_{u_i \in S \wedge S \in D^U} \prod_{u_c \in C_S(u_i)} P(u_c|u_i).$$

Objective function for **implicit** relations in **V**

$$P(v_c|v_j) = \frac{\exp(\vec{\mathbf{v}}_j{}^T \vec{\boldsymbol{\vartheta}}_c)}{\sum_{k=1}^{|V|} \exp(\vec{\mathbf{v}}_j{}^T \vec{\boldsymbol{\vartheta}}_k)}$$

$$maximize\ O_3 = \prod_{v_j \in S \wedge S \in D^V} \prod_{v_c \in C_S(v_j)} P(v_c|v_j)$$

BiNE (SIGIR 2018)



Global objective function

- Allow to model both **explicit** and **implicit** relations
- α, β and γ allow a linear regularization between the explicit and implicit components.

$$minimize \quad O_1 = KL(P||\hat{P}) = \sum_{e_{ij} \in E} P(i,j) \log\left(\frac{P(i,j)}{\hat{P}(i,j)}\right)$$

$$maximize\ O_2 = \prod_{u_i \in S \wedge S \in D^U} \prod_{u_c \in C_S(u_i)} P(u_c|u_i).$$

$$maximize\ O_3 = \prod_{v_j \in S \wedge S \in D^V} \prod_{v_c \in C_S(v_j)} P(v_c|v_j)$$

$$\boxed{maximize\ L = \alpha \log O_2 + \beta \log O_3 - \gamma O_1}$$

BiNE (SIGIR 2018)

Table 4: Performance comparison of Top-10 Recommendation on VisualizeUs, DBLP, and MovieLens.

Algorithm	VisualizeUs				DBLP				Movielens			
	F1@10	NDCG@10	MAP@10	MRR@10	F1@10	NDCG@10	MAP@10	MRR@10	F1@10	NDCG@10	MAP@10	MRR@10
BPR	6.22%	9.52%	5.51%	13.71%	8.95%	18.38%	13.55%	22.25%	8.03%	7.58%	2.23%	40.81%
RankALS	2.72%	3.29%	1.50%	3.81%	7.62%	11.50%	7.52%	14.87%	8.48%	7.95%	2.66%	38.93%
FISMauc	10.25%	15.46%	8.86%	16.67%	9.81%	13.77%	7.38%	14.51%	6.77%	6.13%	1.63%	34.04%
DeepWalk	5.82%	8.83%	4.28%	12.12%	8.50%	24.14%	19.71%	31.53%	3.73%	3.21%	0.90%	15.40%
LINE	9.62%	13.76%	7.81%	14.99%	8.99%	14.41%	9.62%	17.13%	6.91%	6.50%	1.74%	38.12%
Node2vec	6.73%	9.71%	6.25%	13.95%	8.54%	23.89%	19.44%	31.11%	4.16%	3.68%	1.05%	18.33%
Metapath2vec++	5.92%	8.96%	5.35%	13.54%	8.65%	25.14%	19.06%	31.97%	4.65%	4.39%	1.91%	16.60%
BiNE	**13.63%****	**24.50%****	**16.46%****	**34.23%****	**11.37%****	**26.19%****	**20.47%****	**33.36%****	**9.14%****	**9.02%****	**3.01%****	**45.95%****

** indicates that the improvements are statistically significant for $p < 0.01$ judged by paired t-test.

Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. 2018. BiNE: Bipartite Network Embedding. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18). ACM, New York, NY, USA, 715-724.

Next steps

- Include side information about product and query (taxonomy, knowledge graph)
- Improve representations for the “cold start” problem
- Deal with the position bias introduced by the results
- Use as a feature for learning-to-rank approaches
- Neural Information Retrieval

New advances in Graph Representations for Ecommerce Search

Pedro Balage

Data Science Portugal Meetup - DSPT #47

Lisbon, 08th January