Aleksander Molak

What should I buy next?

How to leverage word embeddings to build an efficient recommender system.

Photo: Karolina Grabowska @ Pexels – https://www.pexels.com/photo/tiny-shopping-cart-hanging-on-a-finger-5650028/

Overview

Recommender systems

- Intro
- Embeddings – refresher
- Recommendations with embeddings
- How to tune your embeddings?

About me

https://alxndr.io

Aleksander Molak

https://www.linkedin.com/in/aleksandermolak/

- Innovation Lead & Researcher at **Lingaro**
- **NLP**, **probabilistic modeling**, **causal inference**
- Author of **#SundayAiPapers**
- Complex systems, psychology, neuroscience
- Traveling with my wife, running, vegan food, languages

Intro

Intro

What is ***efficient****?*

Intro

*What is **efficient**?*

- Relatively easy to develop

https://developers.google.com/machine-learning/crash-course/embeddings/video-lecture

Intro

*What is **efficient**?*

- Relatively easy to develop
- Relatively easy to maintain

https://developers.google.com/machine-learning/crash-course/embeddings/video-lecture

Intro

*What is **efficient**?*

- Relatively easy to develop
- Relatively easy to maintain
- Fast inference time

https://developers.google.com/machine-learning/crash-course/embeddings/video-lecture

Word embeddings

Word embeddings

*An **embedding** is a relatively low-dimensional space into which you can translate high-dimensional vectors.*

https://developers.google.com/machine-learning/crash-course/embeddings/video-lecture

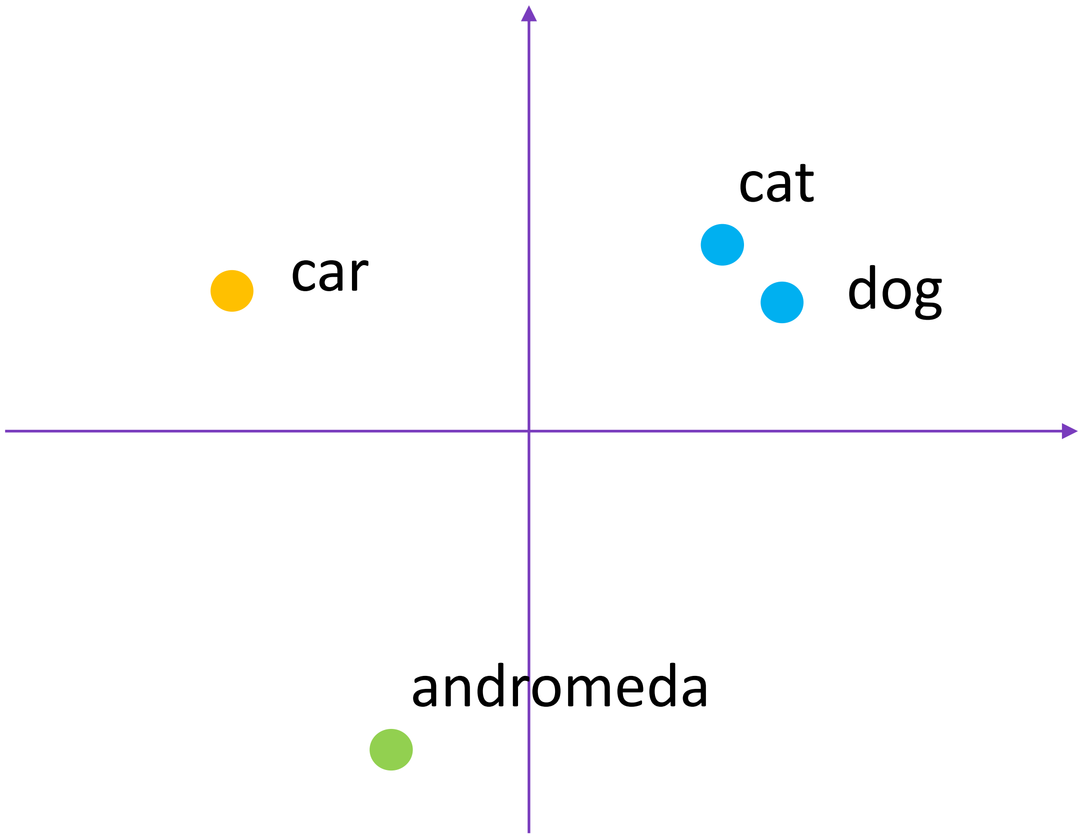
Word embeddings

*An **embedding** is a relatively low-dimensional space into which you can translate high-dimensional vectors.*

REPRESENTATION

https://developers.google.com/machine-learning/crash-course/embeddings/video-lecture

Word embeddings



https://developers.google.com/machine-learning/crash-course/embeddings/video-lecture

Word embeddings

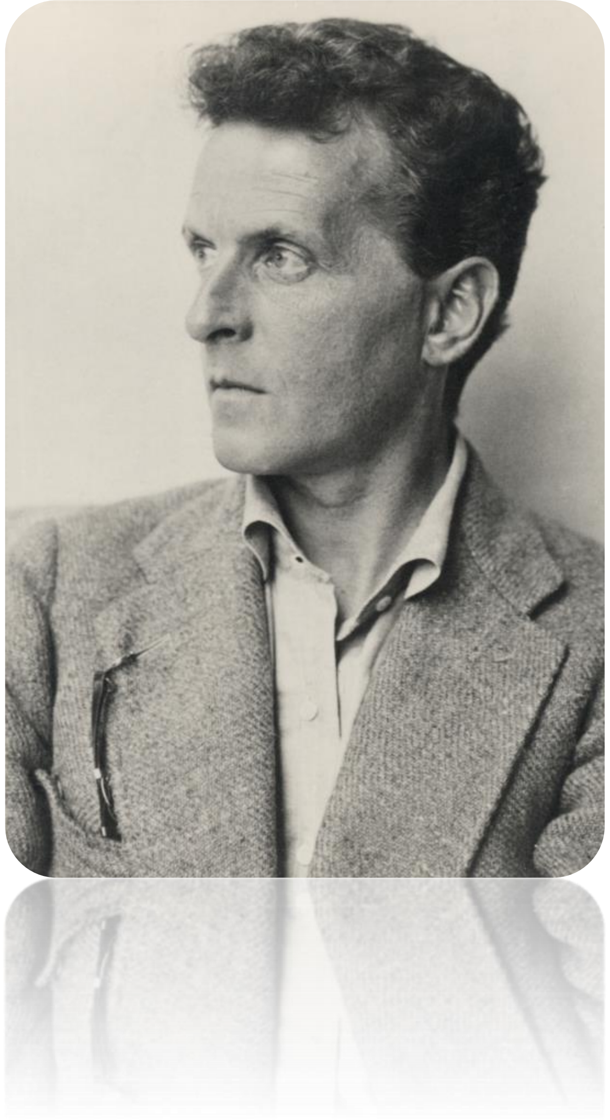
You shall know a word by the company it keeps.

John Rupert Firth

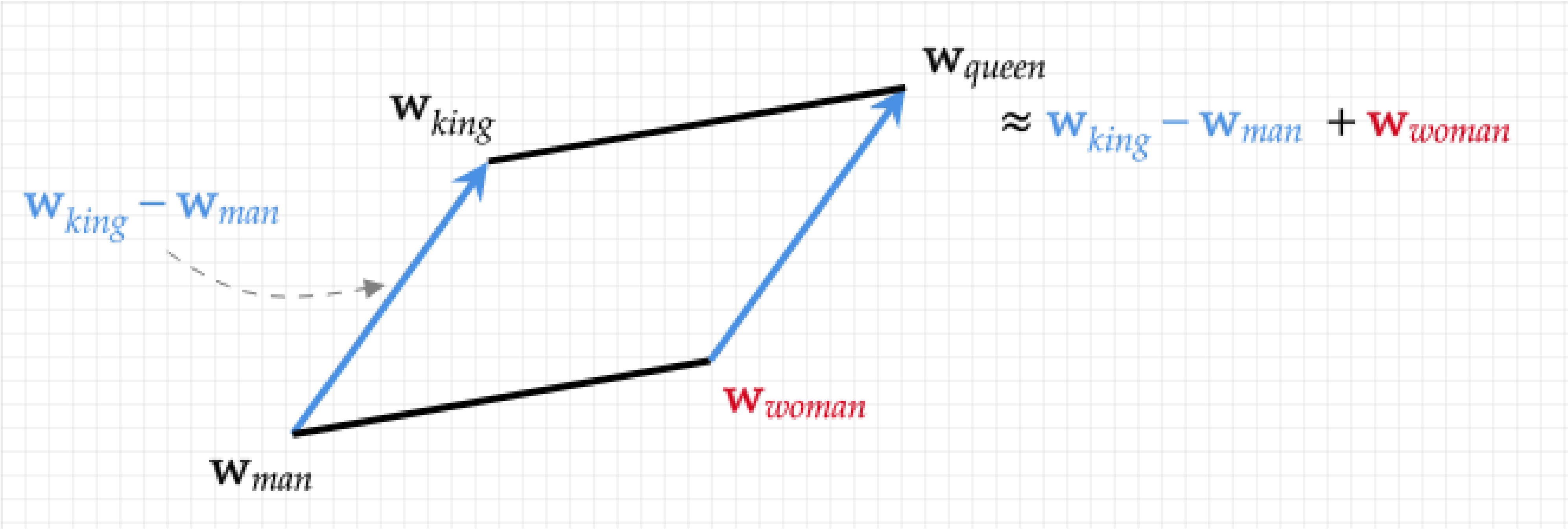
Word embeddings

For a large class of cases – though not for all – in which we employ the word "meaning" it can be defined thus: the meaning of a word is its use in the language.

Ludwig Wittgenstein



Word embeddings



Source: https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.ed.ac.uk%2Finformatics%2Fnews-events%2Fstories%2F2019%2Fking-man-woman-queen-the-hidden-algebraic-struct&psig=AOvVaw25ANF5cSKCjgQV6zk0rdf8&ust=1601706755751000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCID6h_uklewCFQAAAAAdAAAAABAK

Recommendations with embeddings

Recommendations with embeddings

Recommendations with embeddings

- *If we can run **semantic arithmetic** on word embeddings, why not to try the same thing for **products**?*

Recommendations with embeddings

- ***How*** *to do it?*

Recommendations with embeddings

- Model **inter-product relations**

Recommendations with embeddings

- Model **inter-product relations**

- Model **meta-relations**

Recommendations with embeddings

- Model **inter-product relations**
- Model **meta-relations**
- Encode **morphological** information

Recommendations with embeddings

Model **meta-relations**

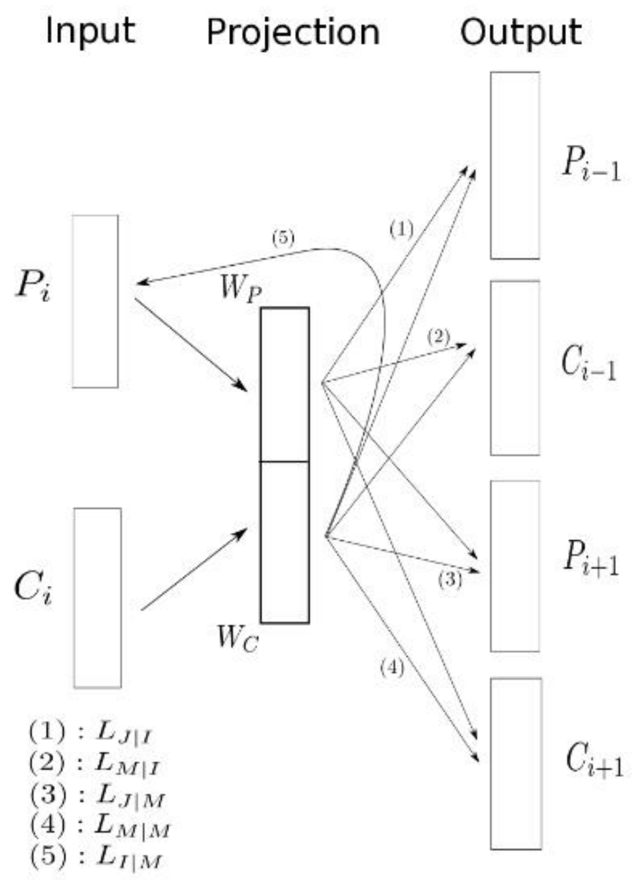


Figure 2: Meta-Prod2Vec Neural Net Architecture.

Recommendations with embeddings

Model **meta-relations**

Meta-Prod2Vec - Product Embeddings Using Side-Information for Recommendation

Flavian Vasile
Criteo
Paris
f.vasile@criteo.com

Elena Smirnova
Criteo
Paris
e.smirnova@criteo.com

Alexis Conneau *
Facebook AI Research
Paris
aconneau@fb.com

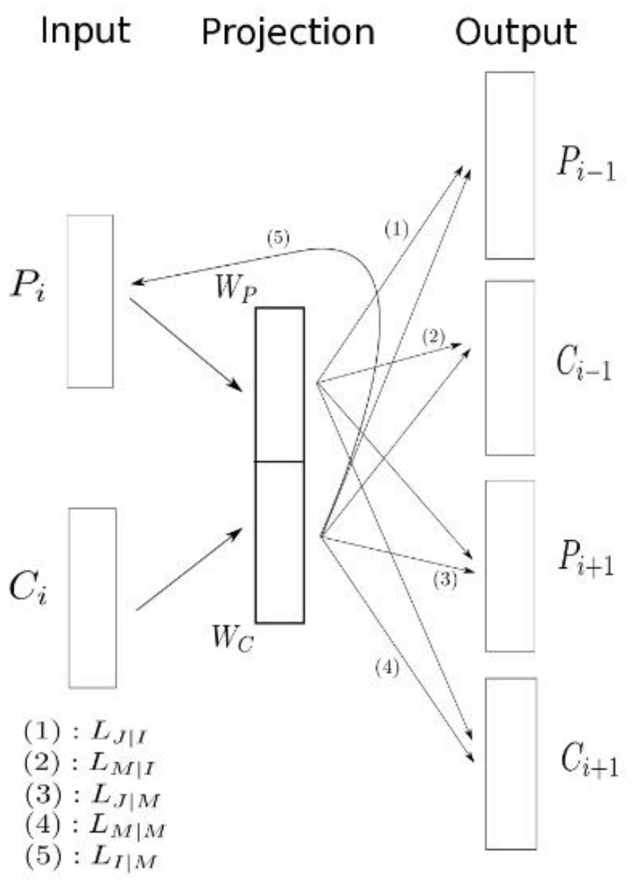


Figure 2: Meta-Prod2Vec Neural Net Architecture.

Recommendations with embeddings

Encode **morphological** information

Recommendations with embeddings

Encode **morphological** information

fastText

Library for efficient text classification and representation learning

GET STARTED DOWNLOAD MODELS

Implementation

Recommendations with embeddings

```
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
```

Recommendations with embeddings

```
# Train the model
model = Word2Vec(train_data, **model_params, callbacks=callbacks)
```

Hyperparameters

Word embeddings

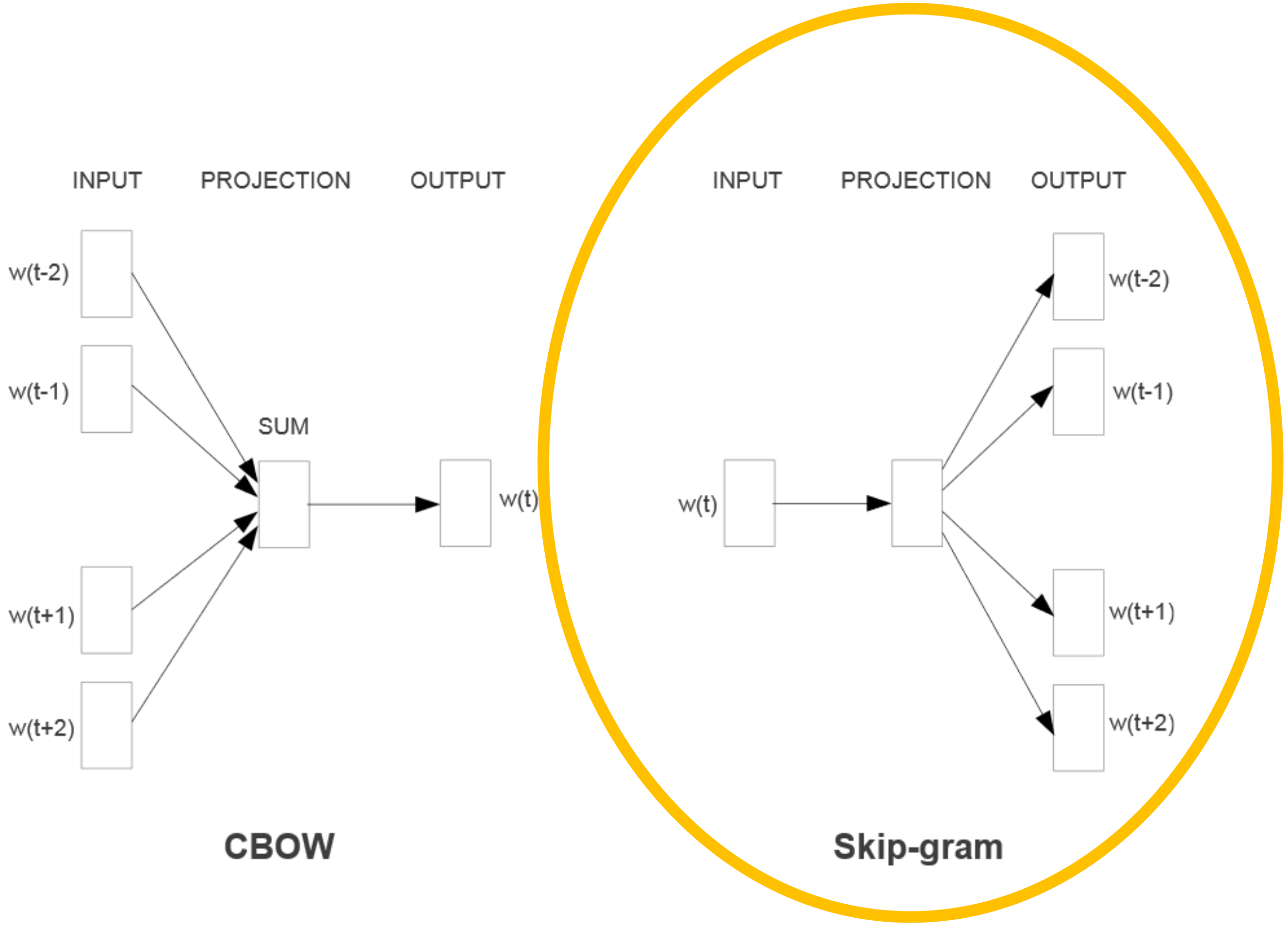


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

Recommendations with embeddings

Sub-sampling

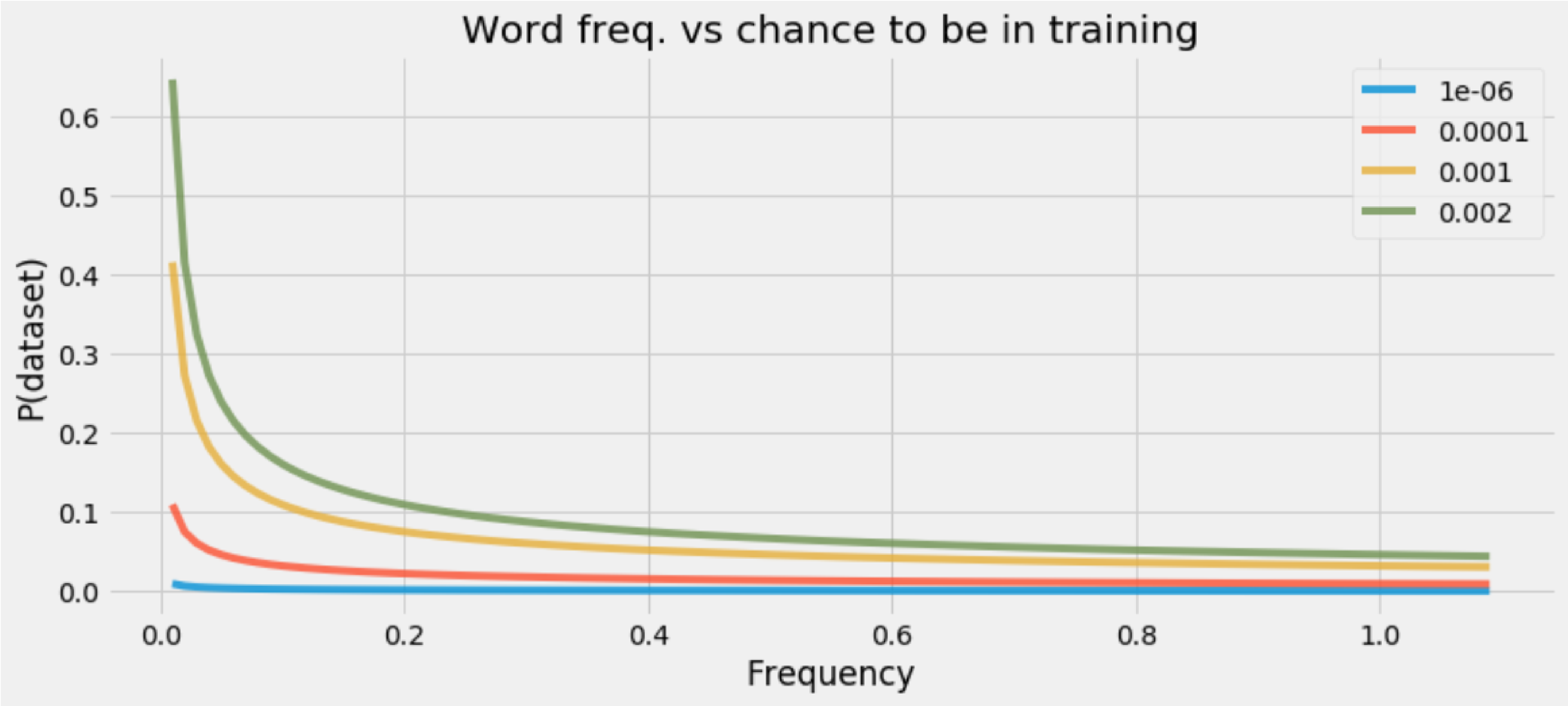
Sub-sampling reduces probability of using the most frequent (or the least frequent) words in the training.

Note that the formula comes from Google implementation of W2V and it's slightly different from the one in he paper.

$$P(w_j) = (\sqrt{\frac{z(w_j)}{k}} + 1)\frac{k}{z(w_j)}$$

** Default value for $k = .001$

Recommendations with embeddings



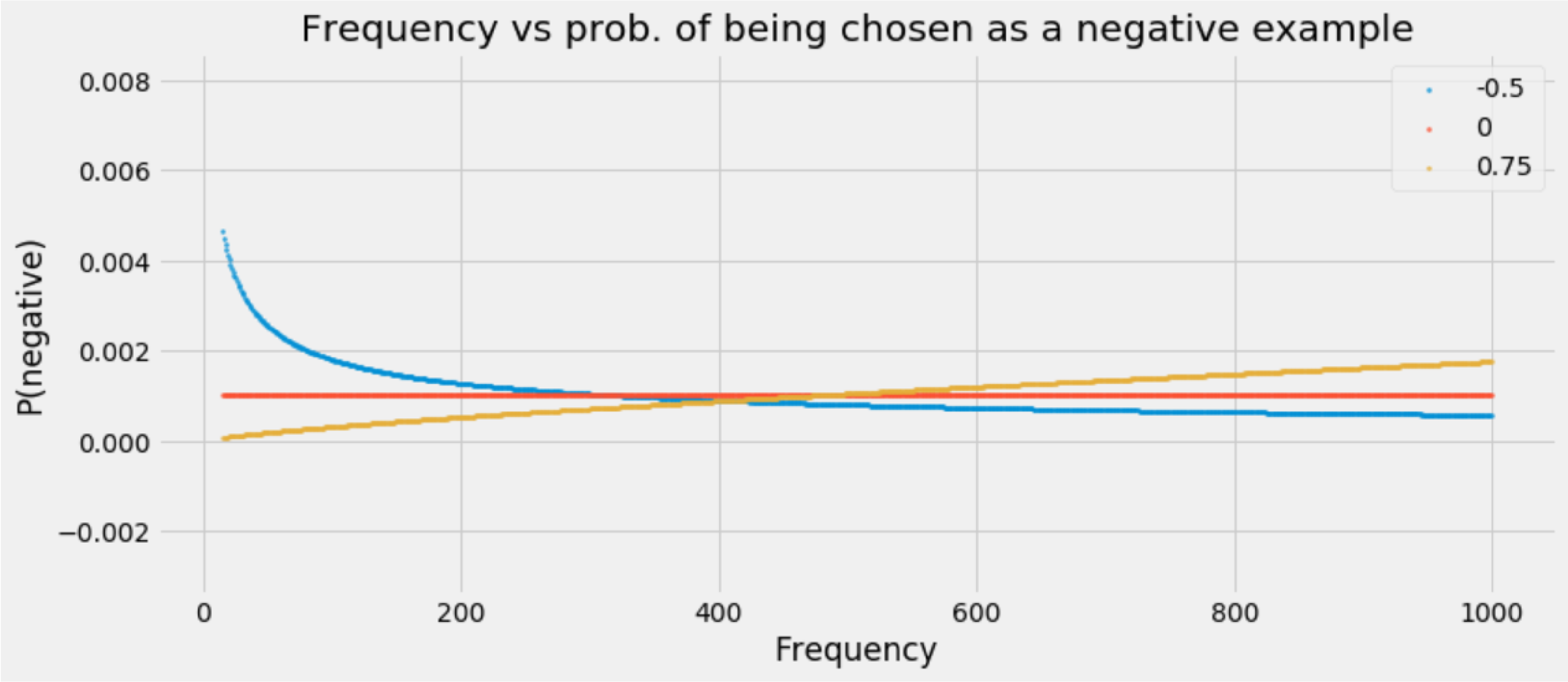
Recommendations with embeddings

Negative sampling

Instead of all negative examples (all words not present in the context), we will just update weights for a couple of randomly selected words.

$$P(w_i) = \frac{f(w_i)^k}{\sum_{j=0}^{N} \left(f(w_j)^k \right)}$$

Recommendations with embeddings



Recommendations with embeddings

Word2vec applied to Recommendation: Hyperparameters Matter

Hugo Caselles-Dupré[1,2*]
[1] Flowers Laboratory (ENSTA ParisTech & INRIA)
[2] Softbank Robotics Europe
Paris, France
caselles@ensta.fr

Florian Lesaint
Deezer SA
Paris, France
flesaint@deezer.com

Jimena Royo-Letelier
Deezer SA
Paris, France
jroyo@deezer.com

Recommendations with embeddings

```
# Initialize callbacks
callbacks = [EpochLogger()]

# Set model params
EMB_DIM = 300
N_CORES = 4
WINDOW = 9   # For traditional w2v you might want to try the 90th percentile of len(rows)
N_EPOCHS = 200
NEG_EXP = -.5
MIN_COUNT = 5
SAMPLE = 0

model_params = dict(size = EMB_DIM,
                    sg = 1,
                    negative = 5,
                    iter = N_EPOCHS,
                    ns_exponent = NEG_EXP,
                    sample = SAMPLE,
                    workers = N_CORES,
                    window = WINDOW,
                    min_count = MIN_COUNT)
```

```
# Train the model
model = Word2Vec(train_data, **model_params, callbacks=callbacks)
```

Recommendations with embeddings

Search

Recommendations with embeddings

- **Large dataset? HNSW!** Hierarchical navigable small worlds

Recommendations with embeddings

- **Large dataset? HNSW!** Hierarchical navigable small worlds
- **Multi-entity-type scenarios? Space partitioning!**

Insights

Recommendations with embeddings

- **Shuffling** either **helps** or is neutral

Recommendations with embeddings

- **Shuffling** either **helps** or is neutral
- **Skip-gram** (usually) performs **better** than CBOW

Recommendations with embeddings

- **Shuffling** either **helps** or is neutral
- **Skip-gram** (usually) performs **better** than CBOW
- **Euclidean** distance **outperforms angular** distance in some scenarios

Recommendations with embeddings

- **Shuffling** either **helps** or is neutral
- **Skip-gram** (usually) performs **better** than CBOW
- **Euclidean** distance **outperforms angular** distance in some scenarios
- **Space partitioning** might be **useful** for multi-entity-type scenarios

Pros and cons

Pros 🥰

- **Easy to maintain**
- **Can model complex interactions**
- **Supports continuous training**
- **(Relatively) fast to train**
- **Ultra fast retrieval** (e.g. with HNSW*)
- **Can be easily served in a serverless fashion** (depending on the no. of tokens)

* Paper: https://arxiv.org/ftp/arxiv/papers/1603/1603.09320.pdf

C++ / Python implementation: https://github.com/nmslib/hnswlib

Cons 😥

- **Not suitable for cold-start scenarios**
- **Might be noisy** (depending on the dataset and training regime)

Thank you!

Thank you!

Aleksander Molak

LinkedIn

https://www.linkedin.com/in/aleksandermolak/

Twitter

@AleksanderMolak