Embeddings

Between Dimensionality Reduction and Feature Engineering

Tomasz Chabinka, Senior Data Scientist @ Alior Bank

PyData Track, PyCon LT 2019, 25.05.2019

Plan

1. Dimensionality Reduction for Sentiment Analysis
2. Feature Engineernig for Recommender Systems
3. How to train embeddings with Word2Vec, GloVe and Recommender Systems?
4. Why all the fuss?
5. TensorFlow

1* Bonus: Gensim Word2Vec

2* Bonus: Bias in Word Embeddings

1

Dimensionality Reduction for Sentiment Analysis

Let’s try to asses sentiment of Harari’s “21 Lessons for the 21st Century” reviews:

“*There's a few good chapters but this is not a great read and a long way from his better work*”
☆☆

“*Would make a good light hearted documentary with Alan Partridge*” ☆☆☆

“*I enjoyed the reading though, especially the sections where he is looking at today's issues, but the last chapter about meditation seemed to be rather bolted on and disconnected to the rest of the book*” ☆☆☆☆

Sentiment Analysis: First Try

We can use bag-of-word representation for our sentences:

few	field	...	goal	good	government	great	...	not
1	0		0	1	0	1		1

Number of features: 400K - 2.2M

Other problems: it’s lacking context, not all the words in training dataset

Sentiment Analysis: Second Try (a)

We can use bag-of-word representation (hot-encode) each word :

few	field	...	goal	good	government	great	...	not
1	0	...	0	0	0	0	...	0
0	0	...	0	1	0	0	...	0
...	...	...	...	...	...	...	...	...
0	0	...	0	0	0	0	...	1
0	0	...	0	0	0	1	...	0

Number of features: 400K * 5 (= 2M) - 2.2M * 30 (= 66 M)

Sentiment Analysis: Second Try (b)

We could try to reduce dimensionality with:

- subset selection,
- shrinkage methods (regularization),
- dimension reduction methods (e.g. Principal Component Analysis) [1]

but it won’t work 😩

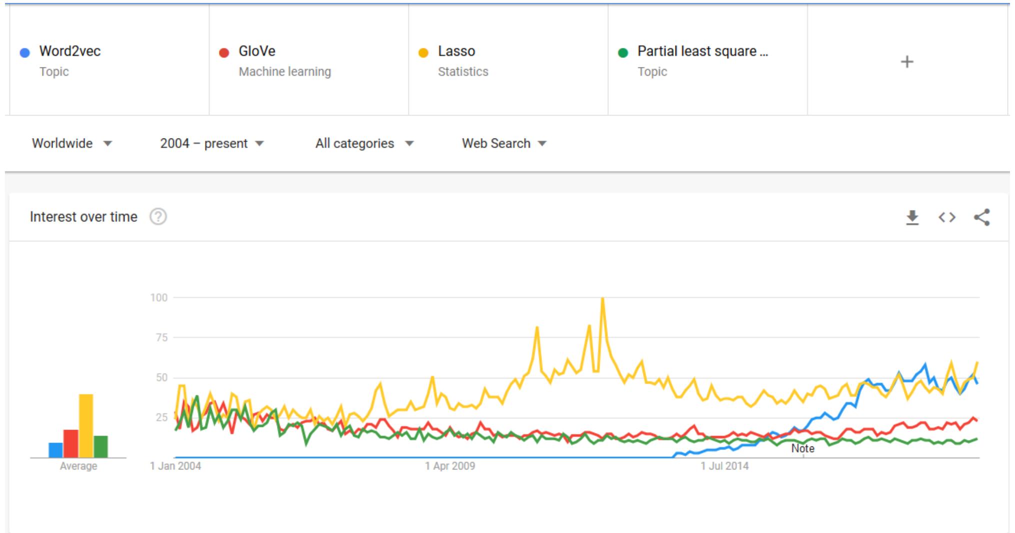
Sentiment Analysis: Embeddings to the Rescue

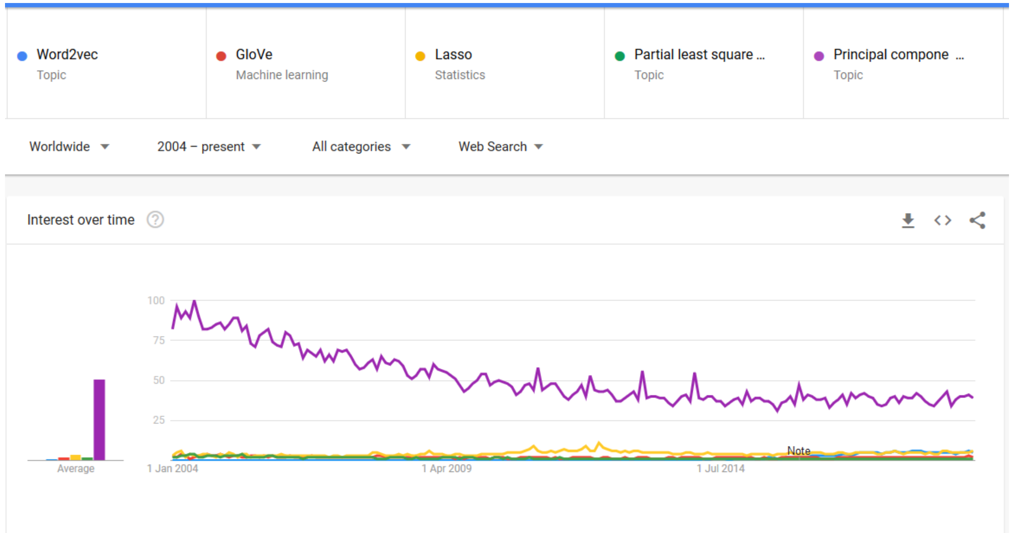
With word embeddings we can encode each token with ~300 features:

- 2M vs 1.5K (1333.3:1 ratio)
- 66M vs 9K (7333.3:1 ratio)

And it will help with rare words too!

Popularity of dimensionality reduction techniques*





Word2Vec / GloVe / Lasso / Partial Least Squares...

...Principal Component Analysis

* according to Google Trends

Plan

1. Dimensionality Reduction for Sentiment Analysis
2. Feature Engineernig for Recommender Systems
3. How to train embeddings with Word2Vec, GloVe and Recommender Systems?
4. Why all the fuss?
5. TensorFlow

1* Bonus: Gensim Word2Vec

2* Bonus: Bias in Word Embeddings

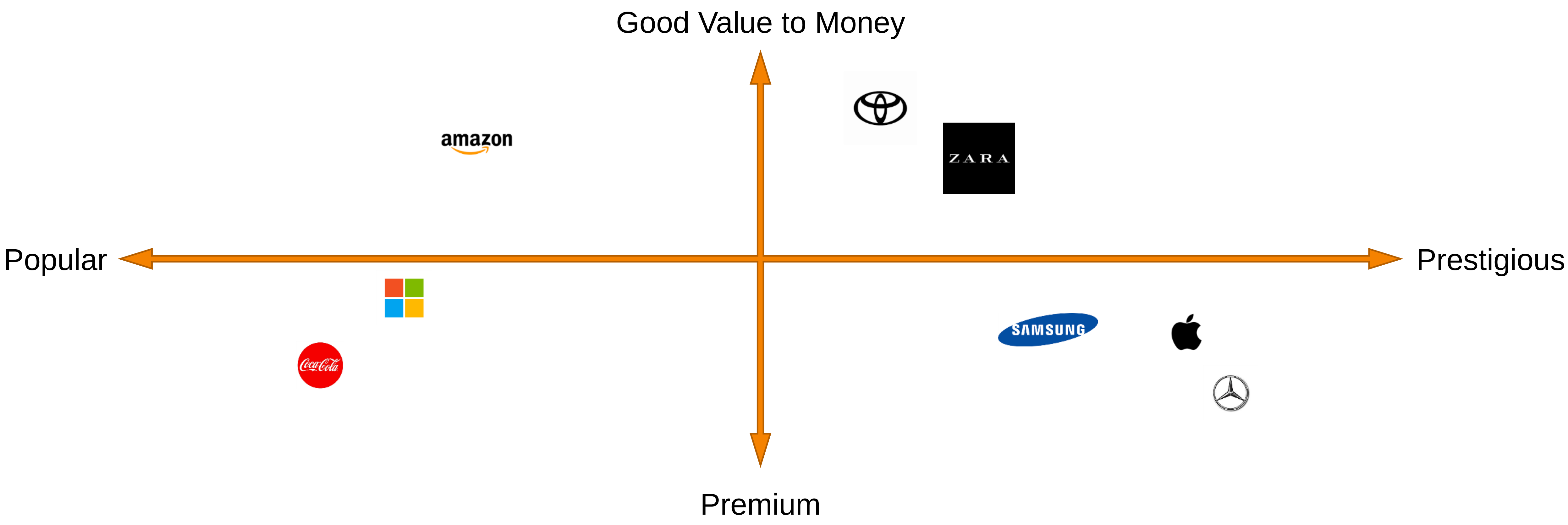
2

Feature Engineernig for Recommender Systems

Let’s say we are building discount coupons app and we want to predict to which shops we should grant coupons to each user based on shops the user visited in the past

One approach would be to try and describe each shop, so we're able to find similar ones.

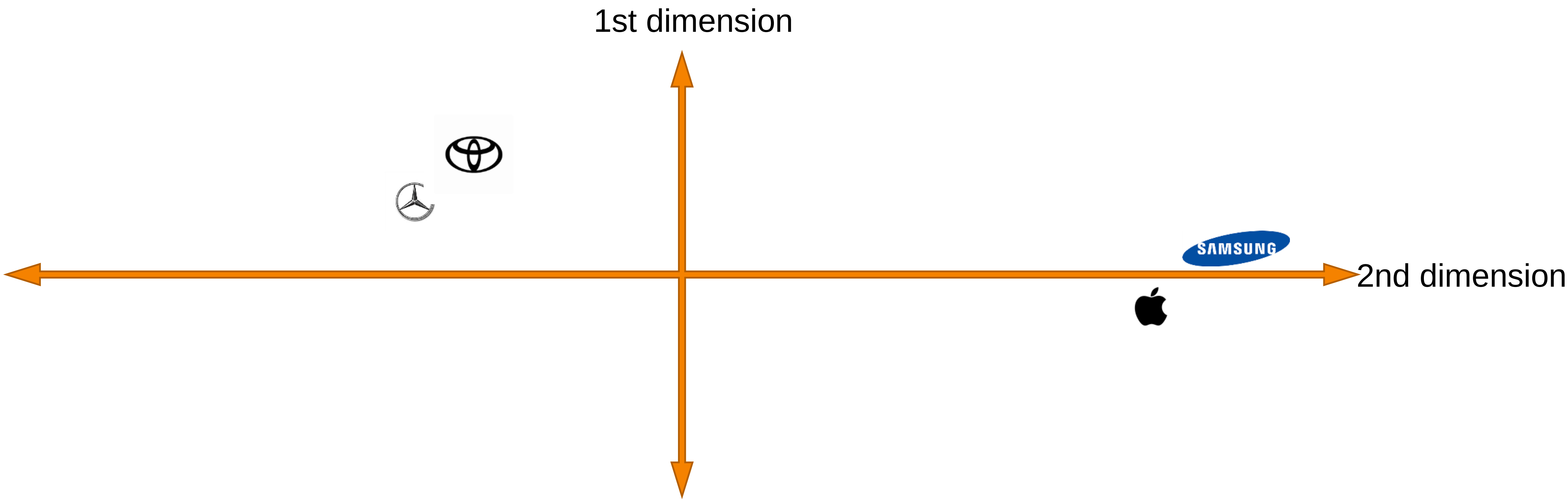
Drawbacks: it may be hard to describe thousands of shops; it requires a lot of expertise



Feature Engineering: Embeddings to the Rescue

Embeddings will learn useful features from categorical data you provide.

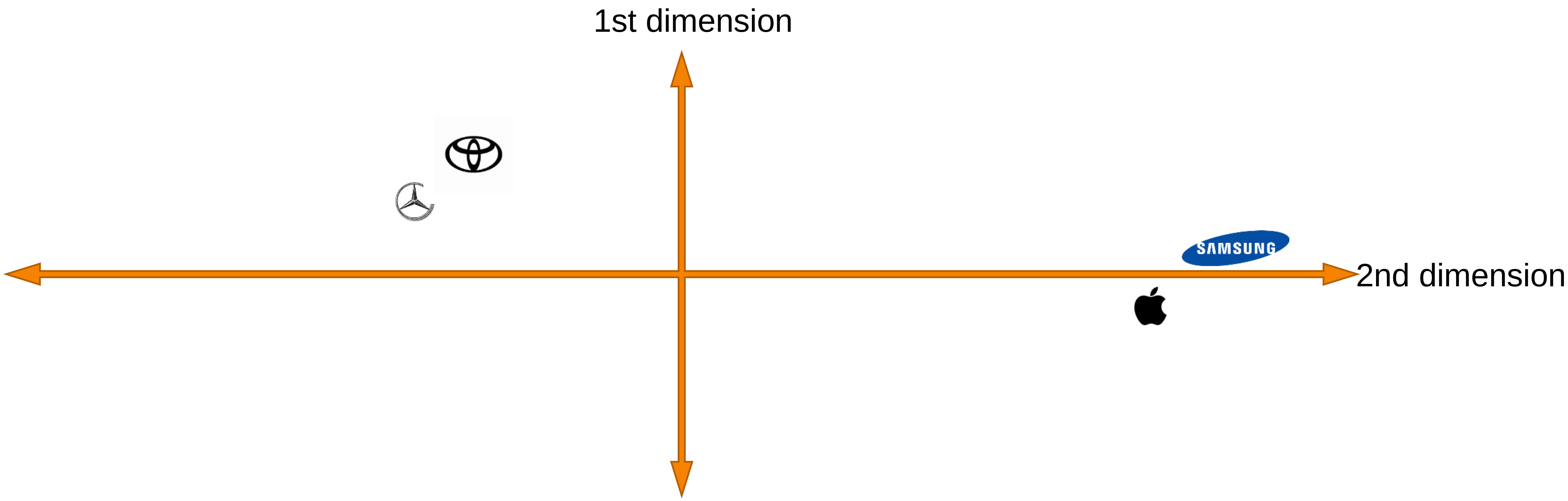
Usually embeddings will learn around useful $\sqrt[4]{n}$ features, where n is a number of items in your problem (e.g. words, shops)



Feature Engineering: Embeddings to the Rescue

We are not able to interpret dimensions, but:

- similar items are close to each other
- we can do math at vectors we have, e.g.: toyota - samsung + apple = mercedes



Embeddings Formal Definition

> embeddings: a categorical feature represented as a continuous-valued feature. Typically, an embedding is a translation of a high-dimensional vector into a low-dimensional space

Google Developer's Machine
Learning Glossary

Embedding Space Formal Definition

> embedding space: the d-dimensional vector space that features from a higher-dimensional vector space are mapped to. Ideally, the embedding space contains a structure that yields meaningful mathematical results; for example, in an ideal embedding space, addition and subtraction of embeddings can solve word analogy tasks

Google Developer's Machine
Learning Glossary

Plan

3

How to train embeddings with Word2Vec, GloVe and Recommender Systems?

Word2Vec

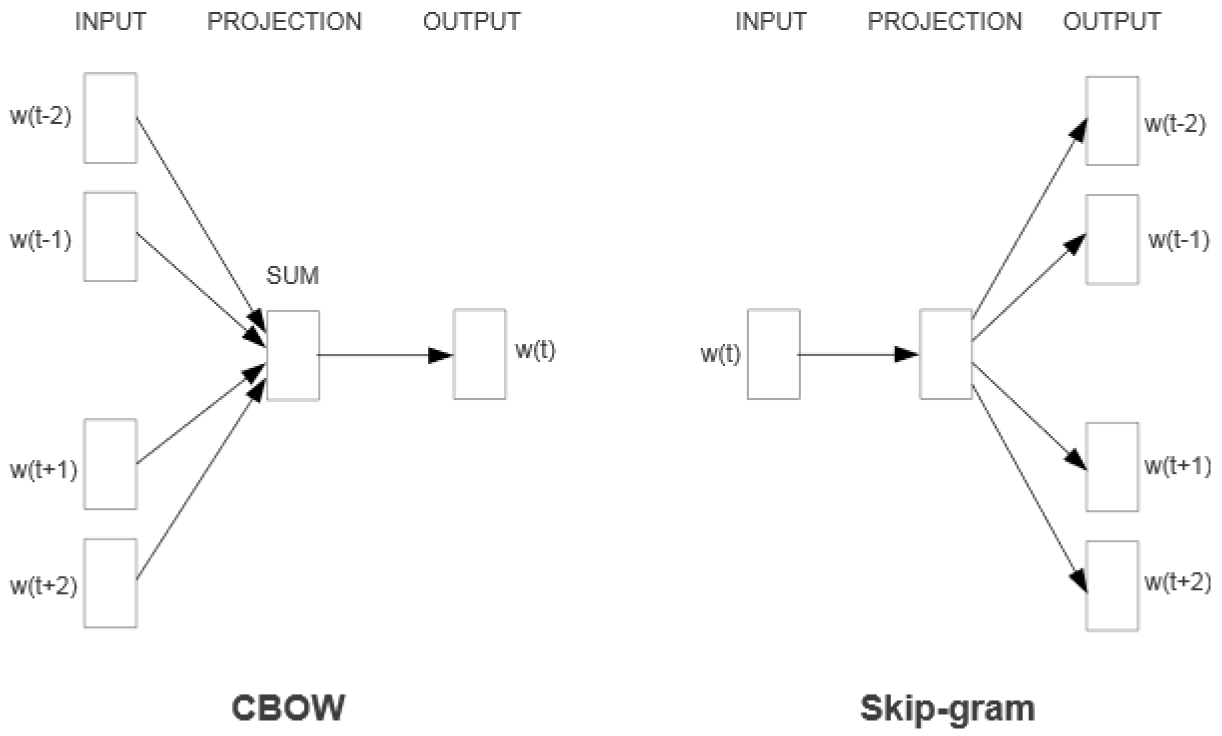
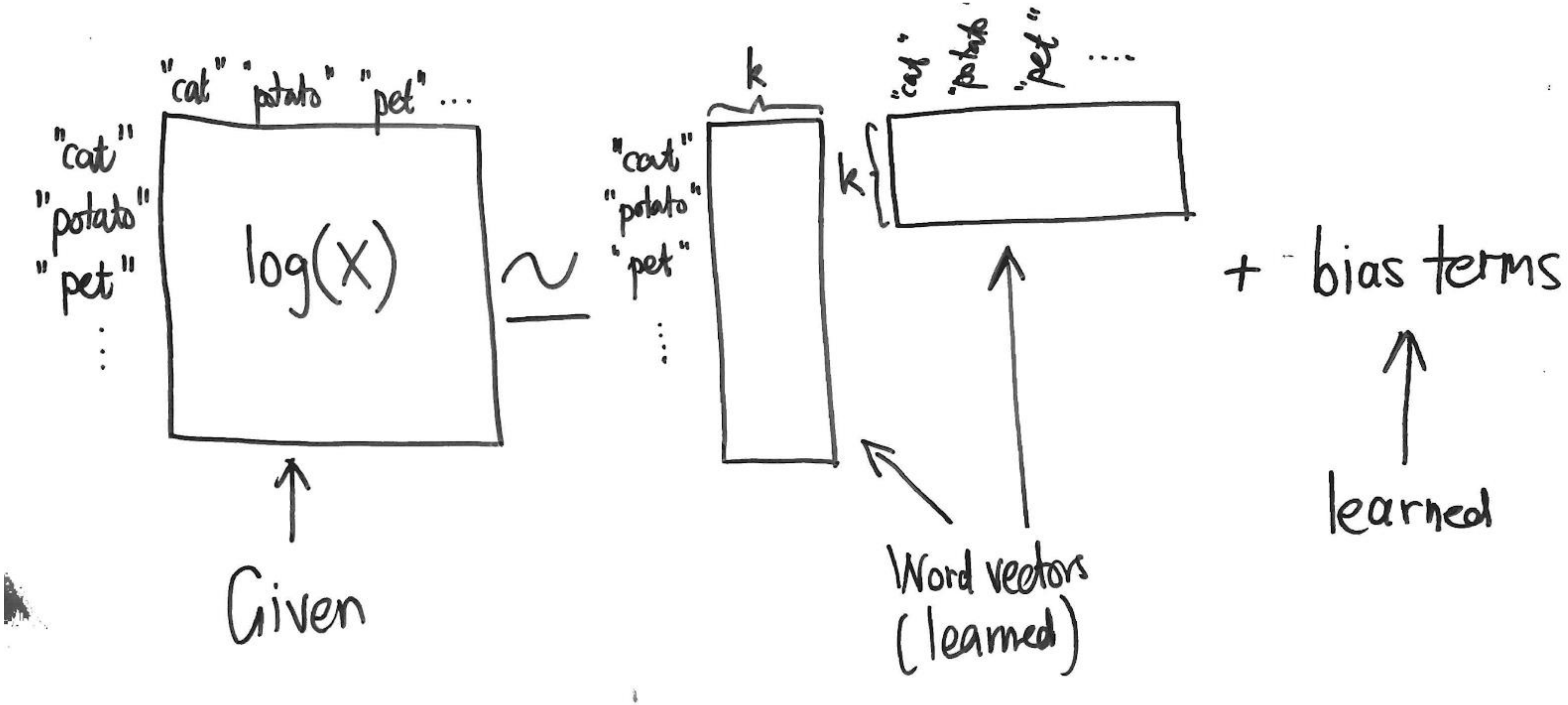


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

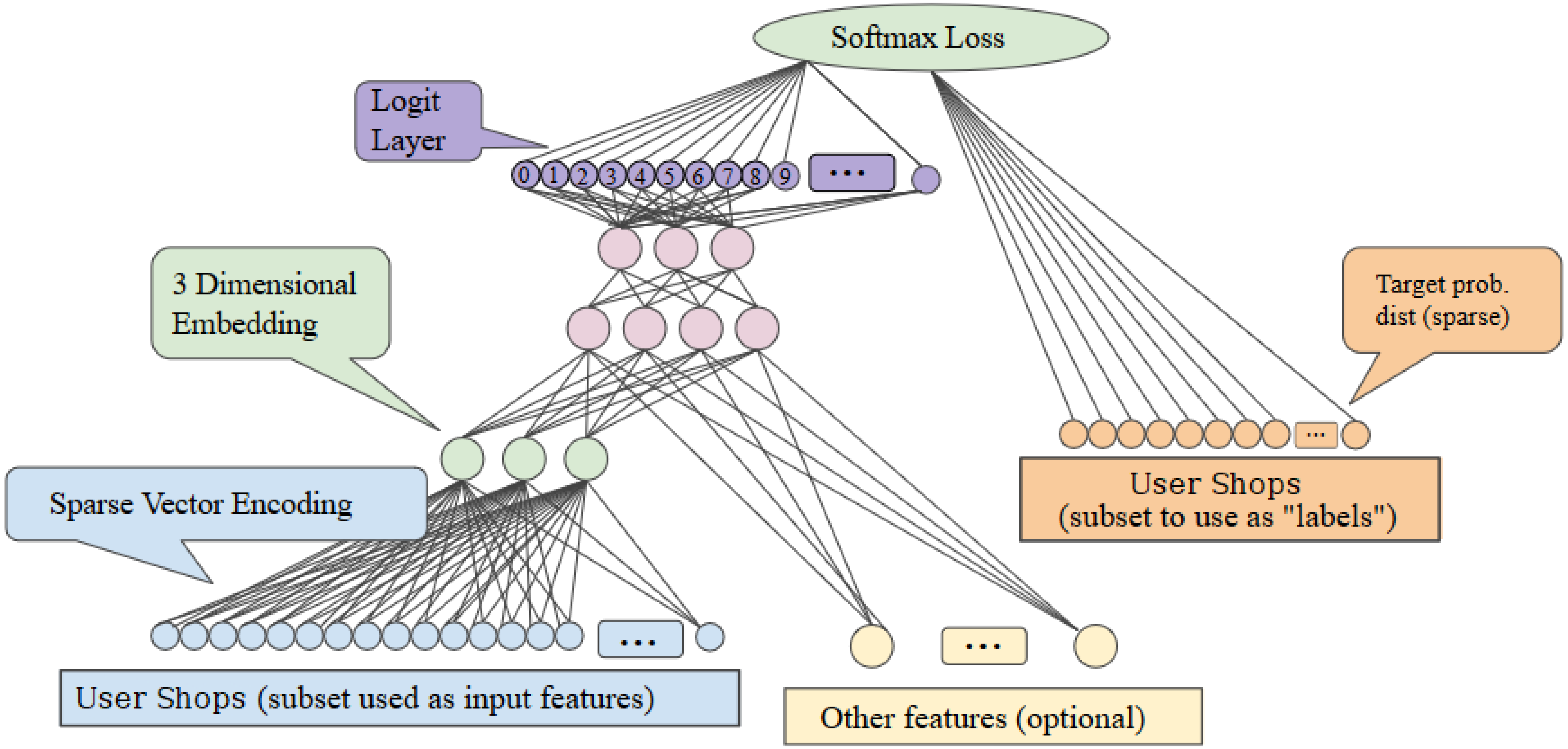
Source: Mikolov et. al., 2013. Efficient estimation of word representations in vector space (https://arxiv.org/pdf/1301.3781.pdf)

GloVe



Source: http://building-babylon.net/2015/07/29/glove-global-vectors-for-word-representations/

Recommender System



Source: https://developers.google.com/machine-learning/crash-course/embeddings/obtaining-embeddings (modified)

Plan

4

Why all the fuss?

Why to use embeddings?

- To find similar items
- To calculate relationships between items
- For transfer learning

Source: https://towardsdatascience.com/collaborative-embeddings-for-lipstick-recommendations-98eccfa816bd

Plan

5

TensorFlow

tf.feature_column.embedding_column

```
terms_embedding_column = tf.feature_column.embedding_column(terms_feature_column, dimension=2)
feature_columns = [ terms_embedding_column ]

my_optimizer = tf.train.AdagradOptimizer(learning_rate=0.1)
my_optimizer = tf.contrib.estimator.clip_gradients_by_norm(my_optimizer, 5.0)

classifier = tf.estimator.DNNClassifier(
  feature_columns=feature_columns,
  hidden_units=[20,20],
  optimizer=my_optimizer
)
```

```
terms_embedding_column = tf.feature_column.embedding_column(
    terms_feature_column,
    ckpt_to_load_from='model.ckpt-1000',
    tensor_name_in_ckpt='dnn/input_from_feature_columns/input_layer/terms_embedding/embedding_weights',
    trainable=False,
    dimension=2)
feature_columns = [ terms_embedding_column ]
```

Source:https://colab.research.google.com/notebooks/mlcc/intro_to_sparse_data_and_embeddings.ipynb (modified)

Q&A

Check job openings @ Alior Bank [PL]:
https://tinyurl.com/alior-it

Apply to our acceleration program:
https://www.accelerator.aliorbank.pl

Please fill in short feedback form:
https://tinyurl.com/pyconlt-embeddings

Stay in touch:
tomaszchabinka@gmail.com
https://www.linkedin.com/in/tomaszchabinka/

Plan

Bonus #1

Gensim Word2Vec

Gensim Word2Vec

```
>>> from gensim.test.utils import common_texts, get_tmpfile
>>> from gensim.models import Word2Vec
>>>
>>> path = get_tmpfile("word2vec.model")
>>>
>>> model = Word2Vec(common_texts, size=100, window=5, min_count=1, workers=4)
>>> model.save("word2vec.model")
```

```
import gensim

# Load Google's pre-trained Word2Vec model.
model = gensim.models.KeyedVectors.load_word2vec_format('./GoogleNews-vectors-negative300.bin', binary=True)
```

```
model.most_similar(positive=['woman', 'king'], negative=['man'], topn=1)
[('queen', 0.50882536)]
model.doesnt_match("breakfast cereal dinner lunch";.split())
'cereal'
model.similarity('woman', 'man')
0.73723527
```

Sources: https://radimrehurek.com/gensim/models/word2vec.html https://rare-technologies.com/word2vec-tutorial/

Plan

1. Dimensionality Reduction for Sentiment Analysis
2. Feature Engineernig for Recommender Systems
3. How to train embeddings with Word2Vec, GloVe and Recommender Systems?
4. Why all the fuss?
5. TensorFlow

1* Bonus: Gensim Word2Vec

2* Bonus: Bias in Word Embeddings

Bonus #2

Bias in Word Embeddings

$$\overrightarrow{\mathrm{man}} - \overrightarrow{\mathrm{woman}} \approx \overrightarrow{\mathrm{king}} - \overrightarrow{\mathrm{queen}}$$

$$\overrightarrow{\mathrm{man}} - \overrightarrow{\mathrm{woman}} \approx \overrightarrow{\mathrm{computer\ programmer}} - \overrightarrow{\mathrm{homemaker}}.$$

Figure 1: The most extreme occupations as projected on to the *she*−*he* gender direction on g2vNEWS. Occupations such as *businesswoman*, where gender is suggested by the orthography, were excluded.

Gender stereotype *she*-*he* analogies.

sewing-carpentry	register-nurse-physician	housewife-shopkeeper
nurse-surgeon	interior designer-architect	softball-baseball
blond-burly	feminism-conservatism	cosmetics-pharmaceuticals
giggle-chuckle	vocalist-guitarist	petite-lanky
sassy-snappy	diva-superstar	charming-affable
volleyball-football	cupcakes-pizzas	hairdresser-barber

Gender appropriate *she*-*he* analogies.

queen-king	sister-brother	mother-father
waitress-waiter	ovarian cancer-prostate cancer	convent-monastery

Source: Bolukbasi et. al., 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings https://arxiv.org/abs/1607.06520

Goals when debiasing:

1. Reduce bias:
   a) Ensure that gender neutral words such as nurse are equidistant between gender pairs such as he and she.
   b) Reduce gender associations that pervade the embedding even among gender neutral words.
2. Maintain embedding utility:
   a) Maintain meaningful non-gender-related associations between gender neutral words, including associations within stereotypical categories of words such as fashion-related words or words associated with football.
   b) Correctly maintain definitional gender associations such as between man and father

Source: Bolukbasi et. al., 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings https://arxiv.org/abs/1607.06520

Steps:

1. Identifying the gender subspace (gender direction)
2. Neutralize - ensures that gender neutral words are zero in the gender subspace
3. Equalize - perfectly equalizes sets of words outside the subspace and thereby enforces the property that any neutral word is equidistant to all words in each equality set.

For instance, if {grandmother,grandfather} and {guy,gal} were two equality sets, then after equalization babysit would be equidistant to grandmother and grandfather and also equidistant to gal and guy, but presumably closer to the grandparents and further from the gal and guy. This is suitable for applications where one does not want any such pair to display any bias with respect to neutral words.

Source: Bolukbasi et. al., 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings https://arxiv.org/abs/1607.06520