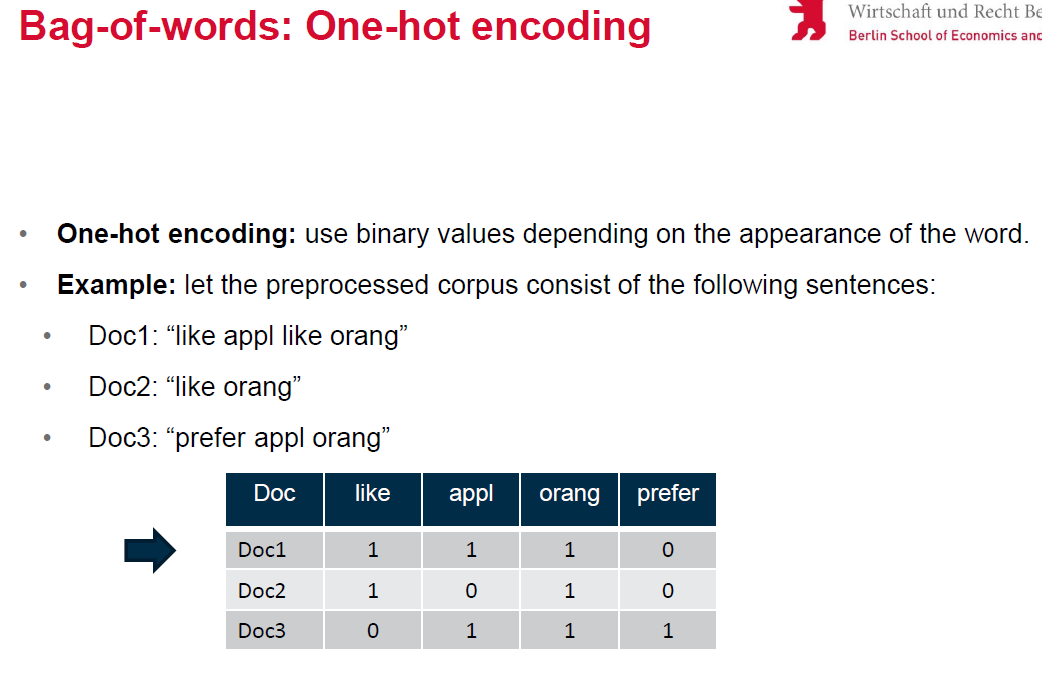
2nd Lesson

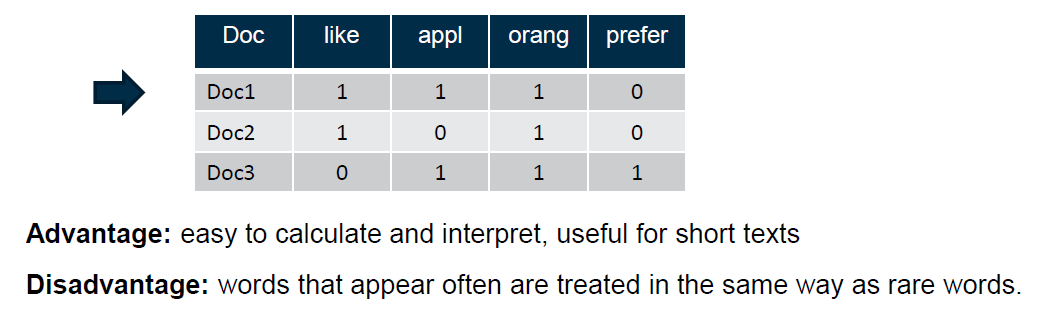
Weakness of Sentiment analysis: negations

* One hot encoding
* Absolut Term Frequencies
* Relative Term Frequencies
* Weighted Term Frequencies

TF 🡪 Document Frequencies

# Bag-of-words: One-hot encoding

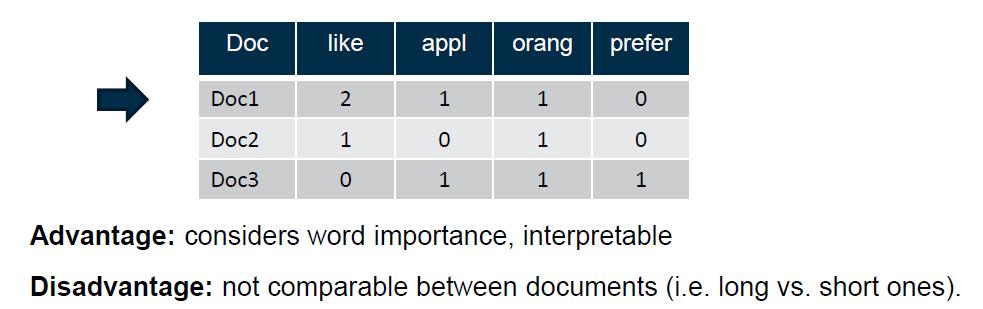




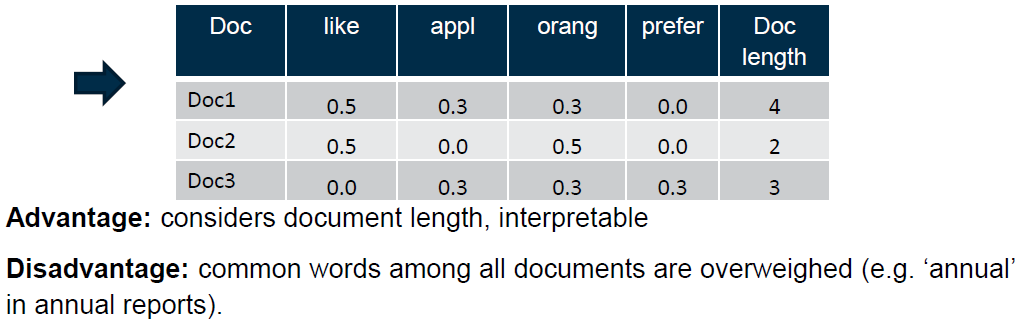
But one-hot-encoding has **disadvantages**:

1. Not possible to compare documents of different length
2. All words are considered equally important
3. Not useful for short texts

# Bag-of-words: Absolute frequency

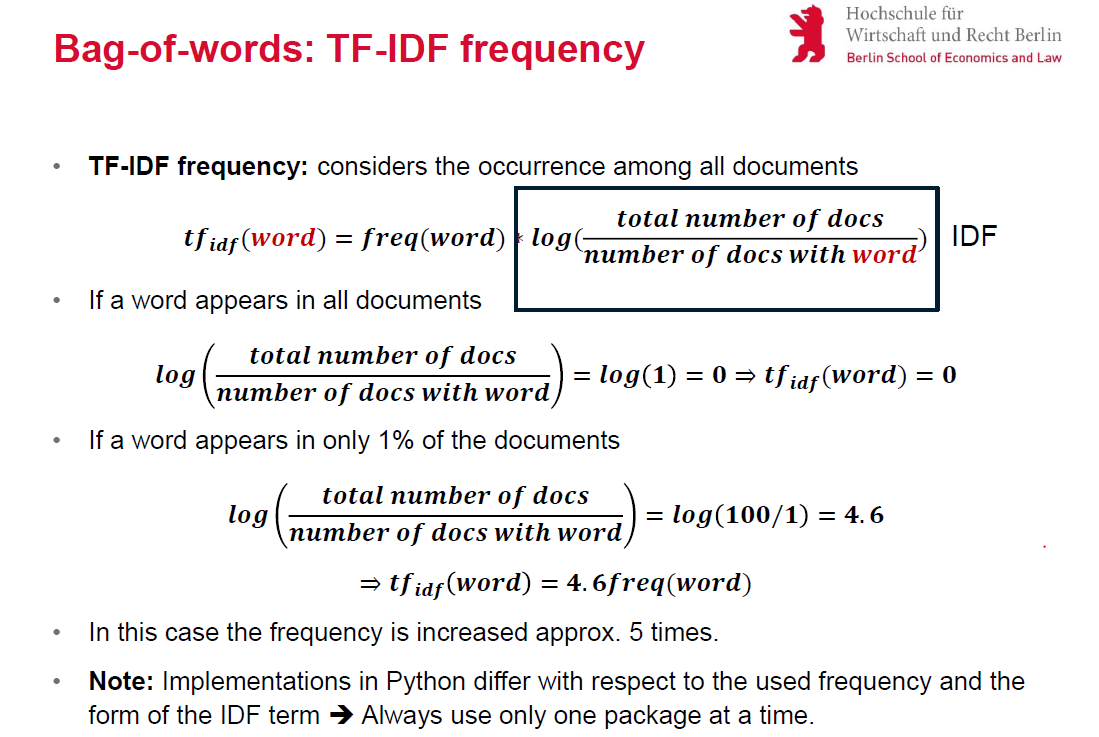


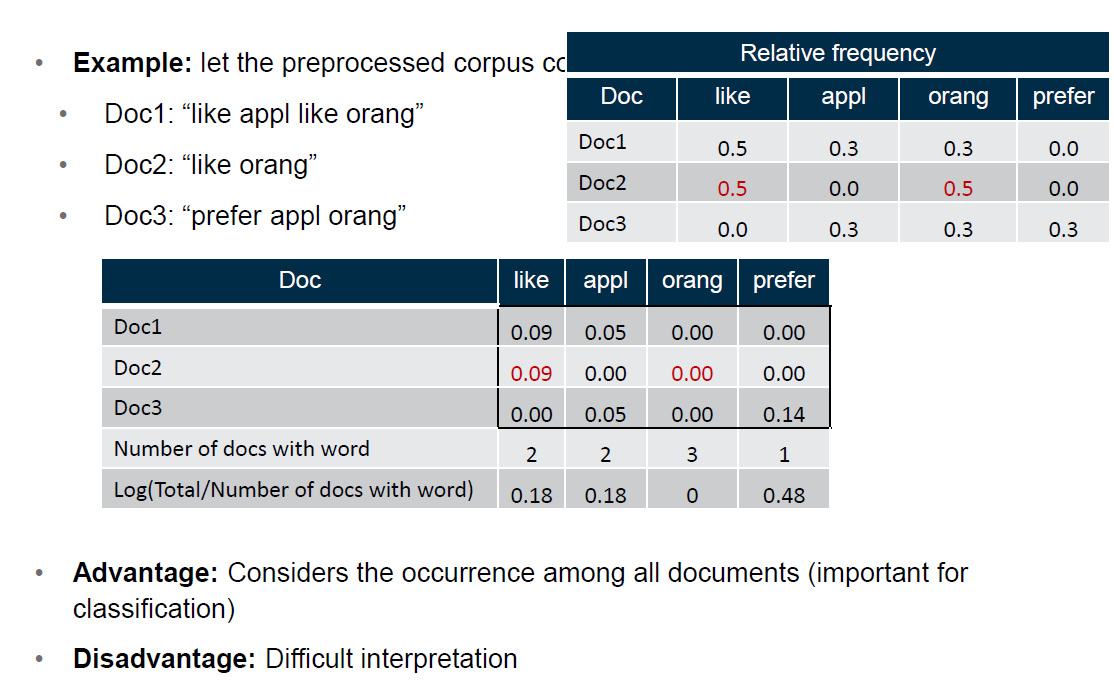
# Bag-of-words: Relative frequency



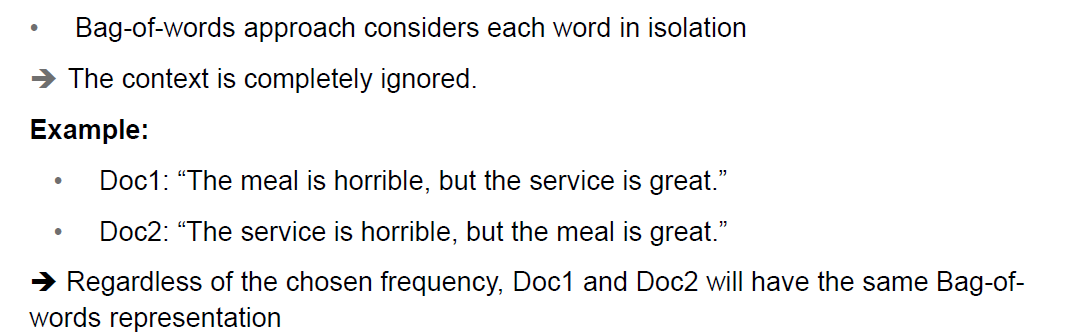
It’s important to not overweight common words:

# Bag-of-words: TF-IDF frequency

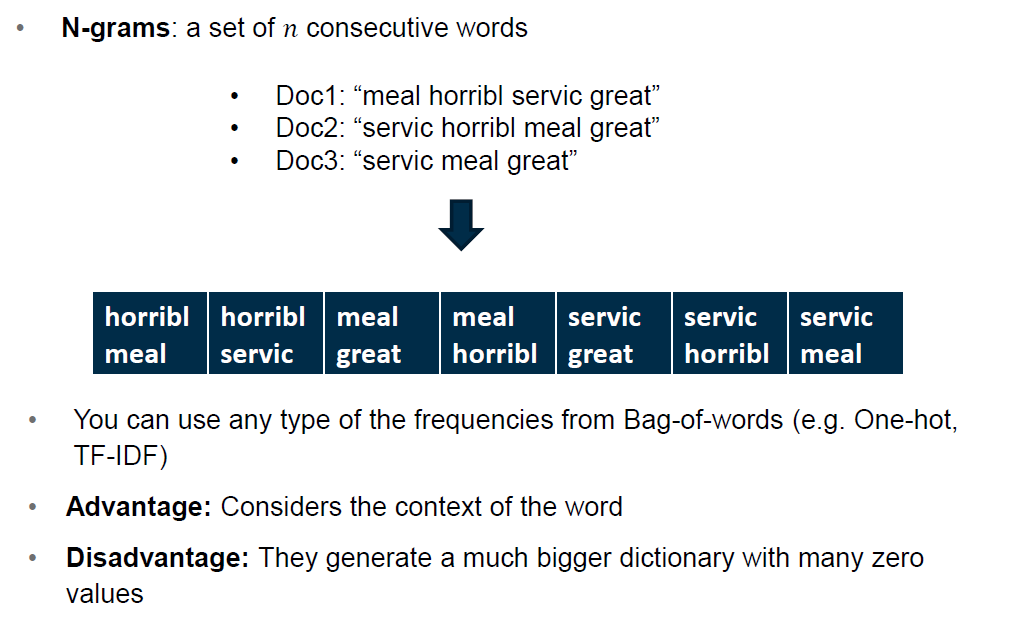




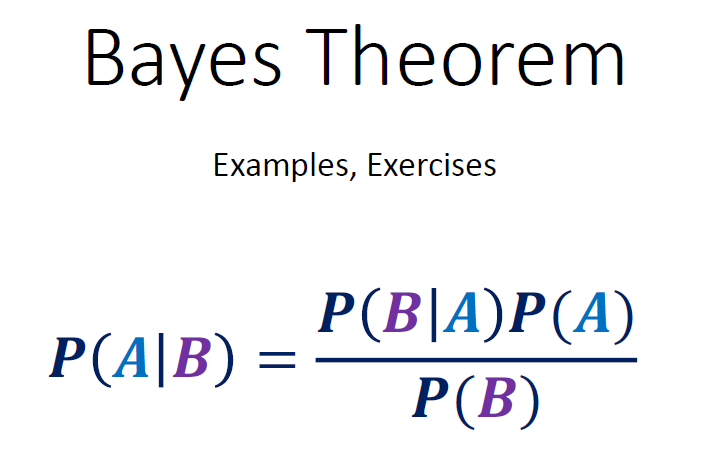
# Disadvantages of Bag-of-words:



# N-grams



3rd Lesson



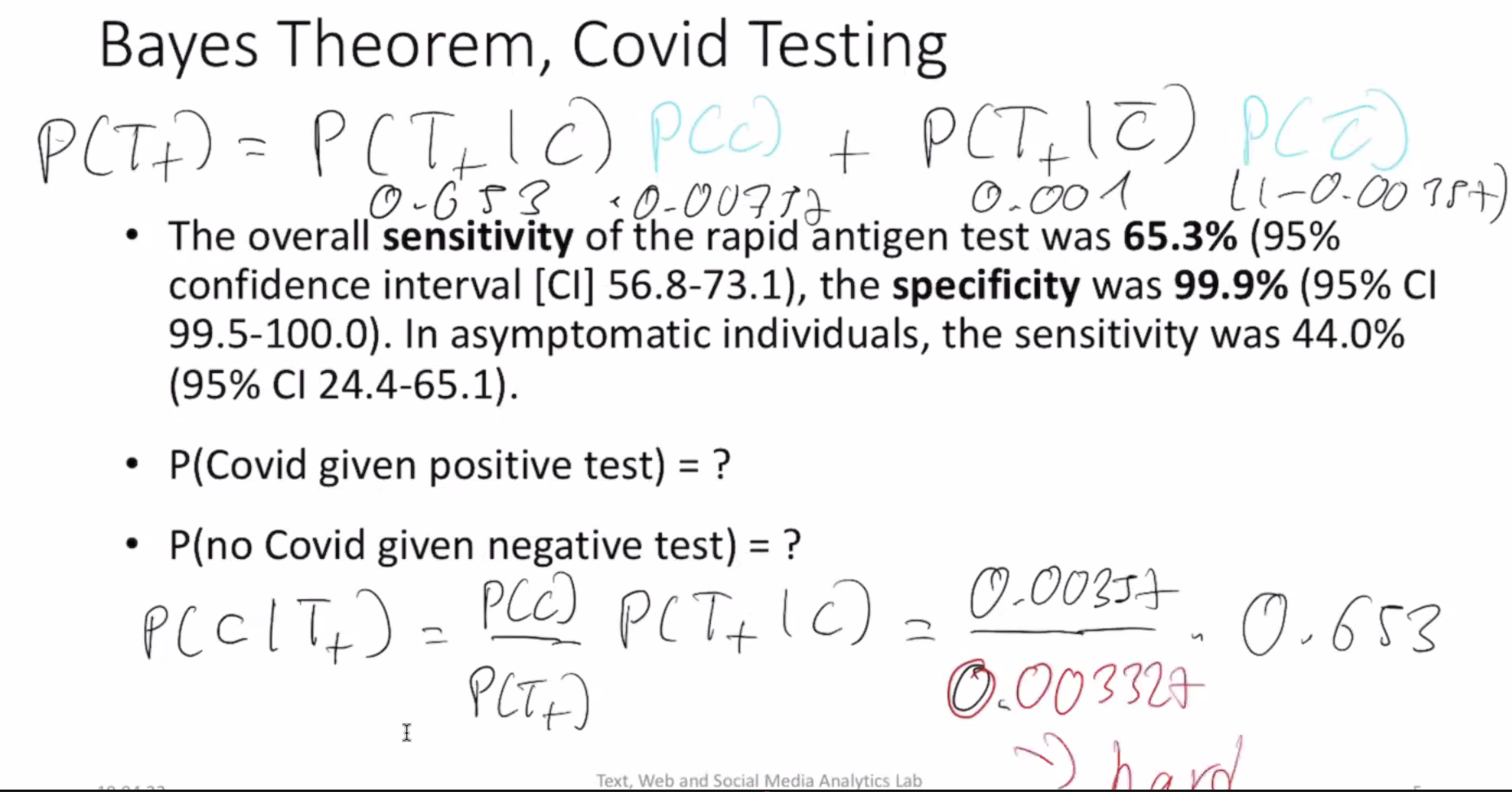
Baseline fallacy:

P(G|A) = 0.001

P(A|G) = P(A) / 0.001 x 0.001 = P(A)

Law of total probability:



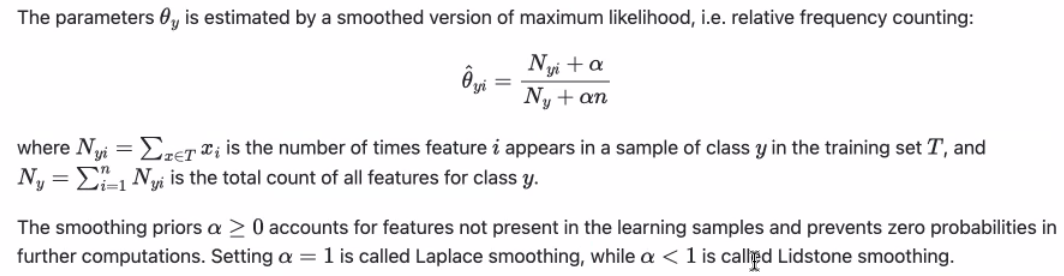


**Naïve bayes**: you never get the true probability out.

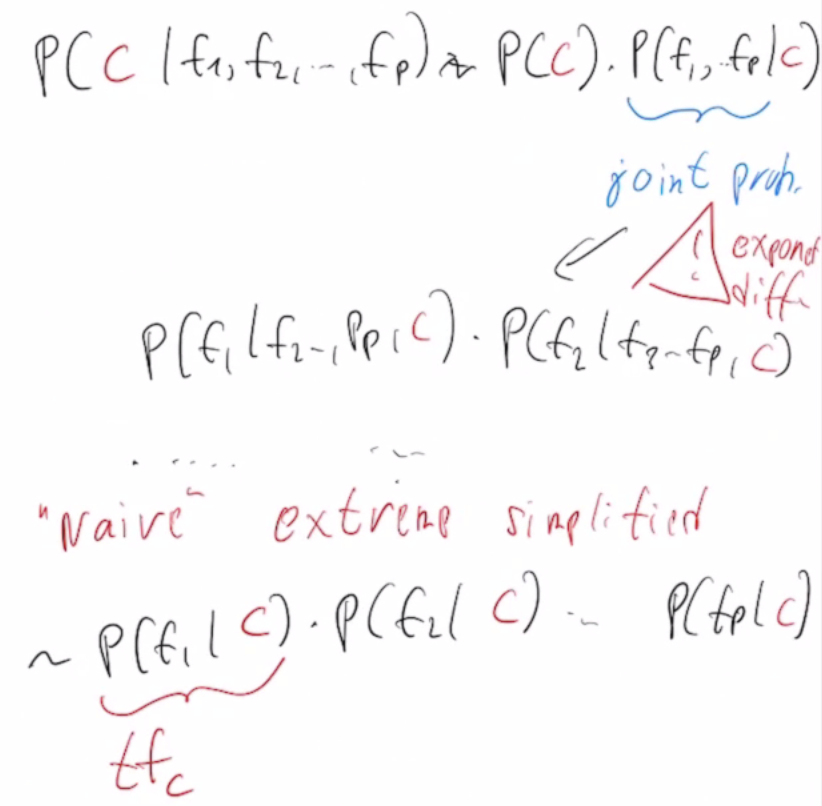
Simplifications:

1. It’s useful to leave the denominator outside because it’s shared between all choices.
2. You have multiple features.

Multinomial Naïve Bayes:



**Low count smoothers:** prior knowledge



**Neural Networks**

Regression = only 1 output node

Multinomial = different output nodes, one for each class

It’s done with the **keras** library

Number of weights 1st step: 10.000 x 16 features + 16 bias

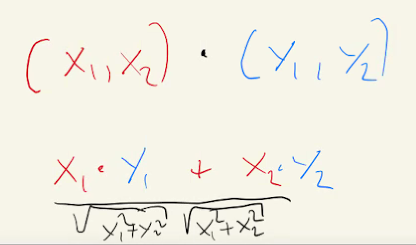
Number of weights 2nd step: 16 x (16+1)

4th Lesson. Lesson\_LatentFactors.pdf

Objective: **reduce dimensions**.

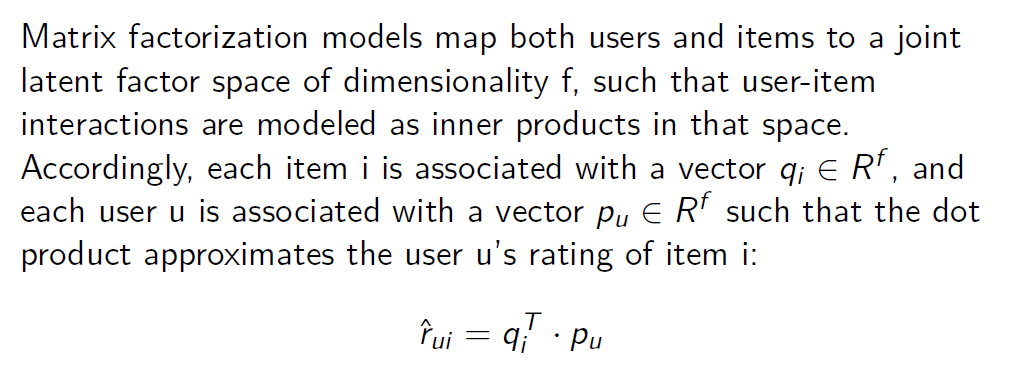
Unsupervised methods (no Y). Ex: k-means.

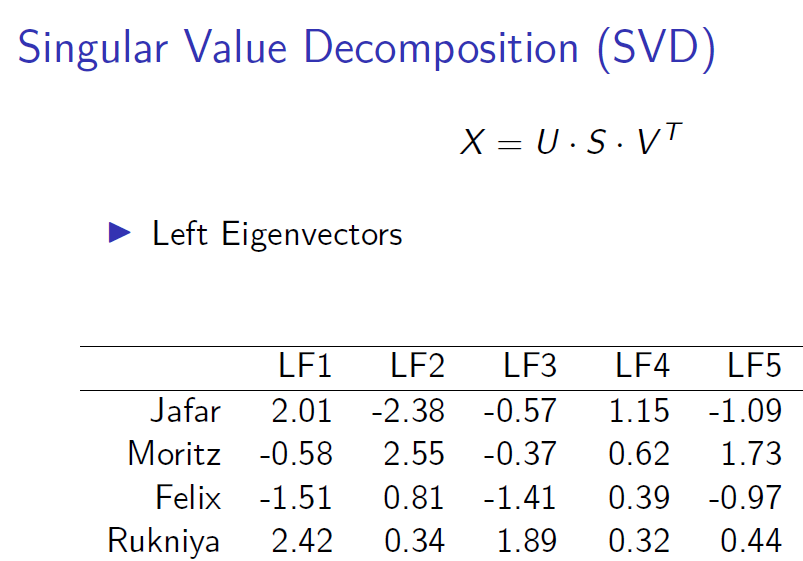
Combine vectors:



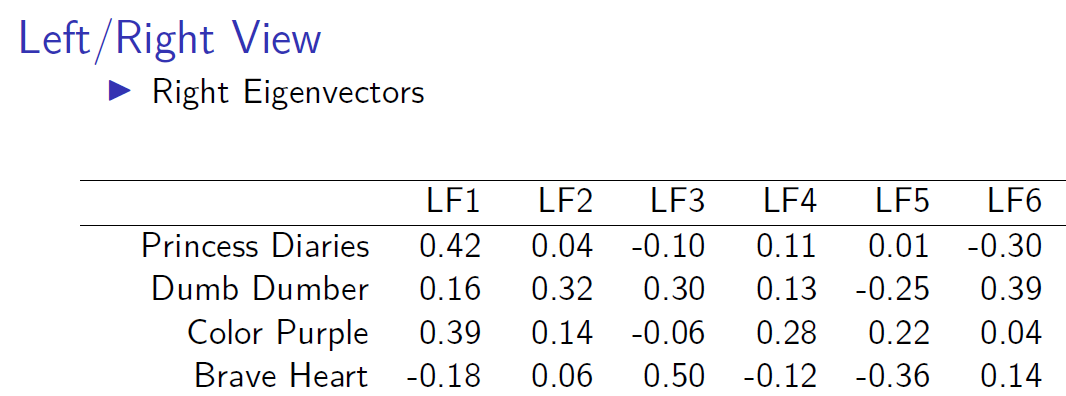
Critique: why only two vectors? Who chooses the features and why?

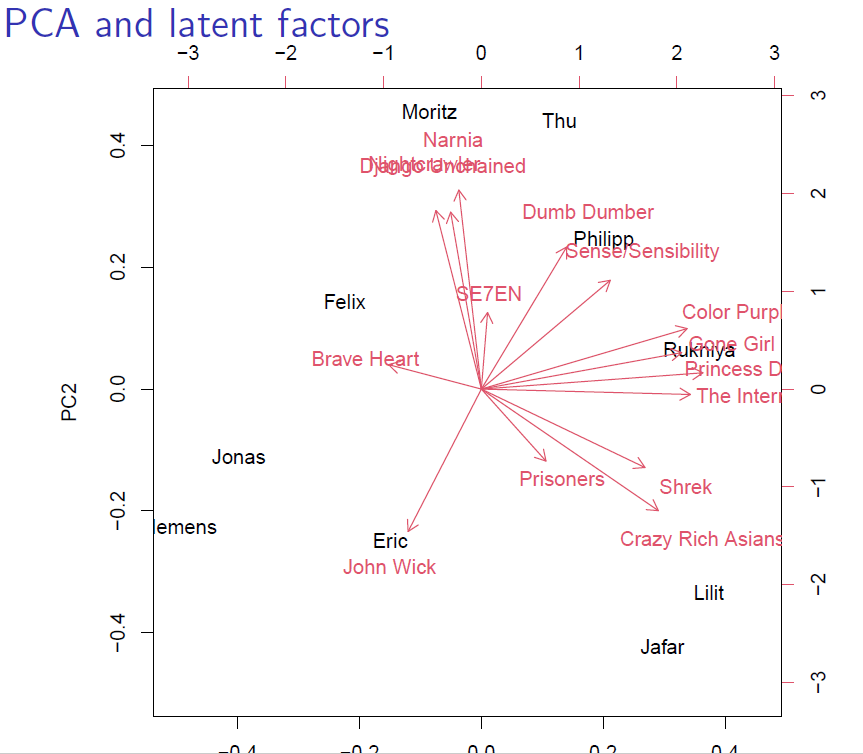
**Matrix factorization**





LF1 is a feature and the numbers define how affine are the persons to it.





Recommendation engines:

* Content filtering
* Collaborative filtering – relies on past behaviors

Text Representation 2. Embedding-based approaches

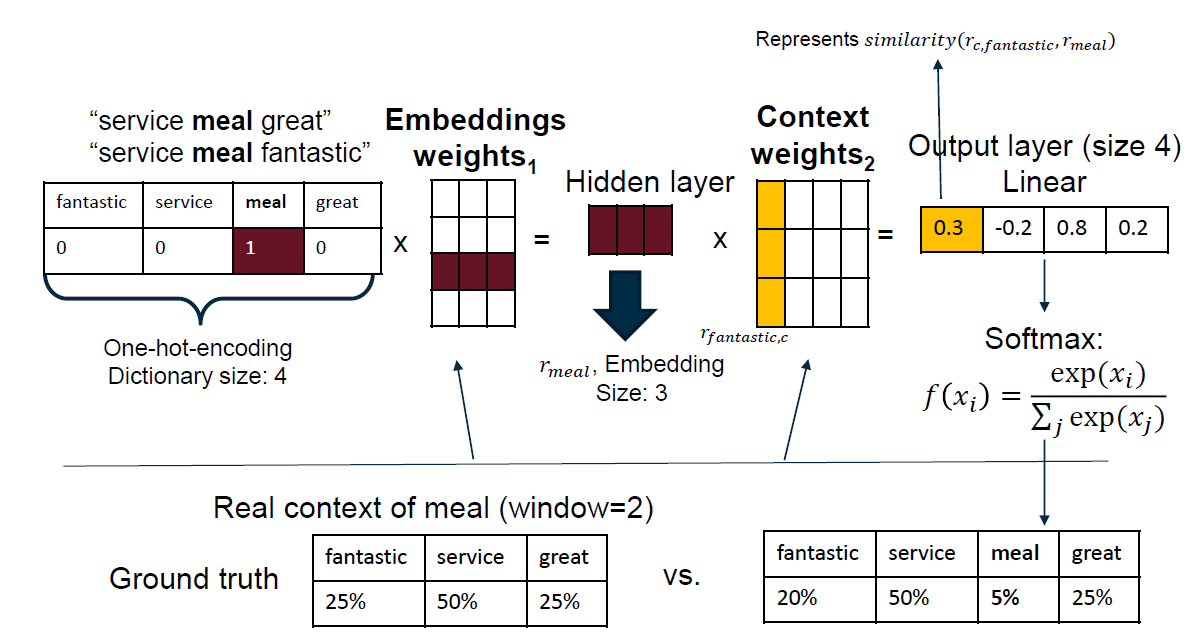
Bag-of-words approaches don’t consider the general meaning of the word in relation to others and thus do not allow for the comparison of documents that do not share words.

**Embeddings** (distributional representation) represent words in a continuous vector space such that similar words are close together. They also allow for negative values.

**Embeddings: Word2vec**

**Word2vec** is a model for embeddings. Idea: **word that appear in similar context must have similar embedding representation**. It’s based on a **two-layer neural network**.

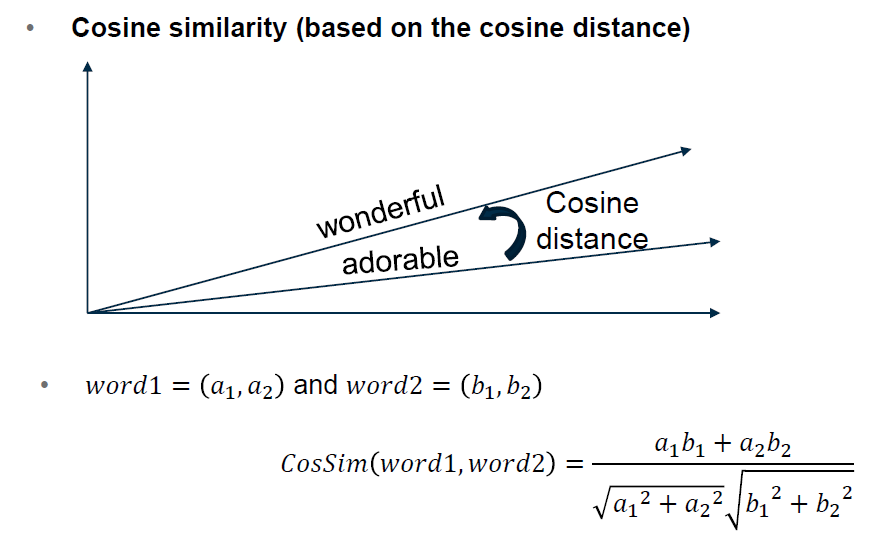
Two approaches: continues-bag-of-words and skip-gram



The resulting embeddings:

* Are derived based on the similarity between words
* Allow for vector arithmetic that mimic the semantic meaning of a word
* Are the same for all usages of the word

Calculate **similarity between two representations**:

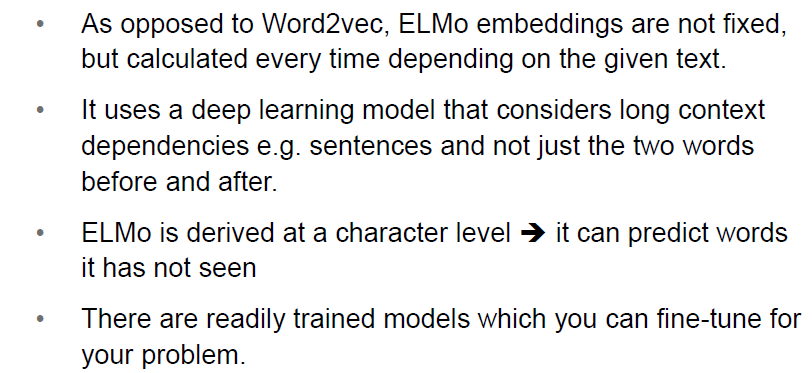


Ex. Slide 17

**Advantages Word2vec**: considers relationship of the word meaning to other words

**Disadvantage Word2vec**: vector representation is always the same regardless of particular use (bank office vs river bank vs blood bank). Solution: Contextual Embeddings

**Contextual Embeddings: Embedding for Language Models**



* ELMo uses a 2-layer-long short-term memory neural network: a NN capable of modelling long-term dependencies.
* The network is **bi-directional** such that both forward and backward language model are trained.
* The results from the last layer create **prediction** **probabilities**.

Disadvantage: training is difficult to parallelize due to recurrent nature.

**Transformers: Main Architecture**

Transformers follow an encoder-decoder transformer architecture stemming from translation systems. The original transformer paper uses six encoders and six decoders.

**Transformer: Inputs**