Feature Engineering for Recommendations

Joshua Bernhard

Feature Engineering

- Sparse vs. Dense Matrix Representations
- Standardizing and Normalizing Numeric Data
- Encoding Categorical Data
- Encoding Text Data
- Missing Values
- Outliers

Sparse vs. Dense Matrix Representations

Sparse vs. Dense Matrix Representations

- For most recommendation engines the most common data format is either a sparse or dense representation of the user-item interactions.

- The next slide shows the same data in each of these representations.

- There are pros to each of these representations depending on your goal.

Sparse vs. Dense Matrix Representations

Sparse

		Item		
		Item 1	Item 2	Item 3
User	User 1		5	1
	User 2	3		
	User 3	4		
	User 4			2

Dense

User	Item	Rating
1	2	5
2	1	3
3	1	4
4	3	2
1	3	1

Sparse vs. Dense Matrix Representations

- When interactions are binary (the user interacted with an item or didn't), both the sparse and dense matrices can be simplified.

- Imagine we have the same situation as the previous slide, but we don't have ratings.

- The new representation of this situation is shown on the next slide.

Sparse vs. Dense Matrix Representations

Sparse

		Item		
		Item 1	Item 2	Item 3
User	User 1	0	1	1
	User 2	1	0	0
	User 3	1	0	0
	User 4	0	0	1

Dense

User	Item
1	2
2	1
3	1
4	3
1	3

Sparse vs. Dense Matrix Representations

- Notice even the sparse matrix in this case doesn't have the missing values that are associated with the sparse ratings matrix.

- Depending on the methods you use to build your recommendation system, dealing with the non-binary, sparse matrix can be complicated.

Feature Engineering

Feature Engineering

- Feature engineering is used to encode information about the world or a problem in a way that a machine can understand it.

- For recommendation systems, feature engineering might be used to encode information about users or items.

- The interactions are usually encoded as you saw in the previous slides, so no other feature engineering is needed for the interaction data.

Feature Engineering

- There aren't really special tips to feature engineering specifically for recommendation systems.

- The same tactics used for feature engineering in other data science applications hold for recommendation systems.

- Let's review some of these tactics.

Standardizing & Normalizing Data

Standardizing and Normalizing Data

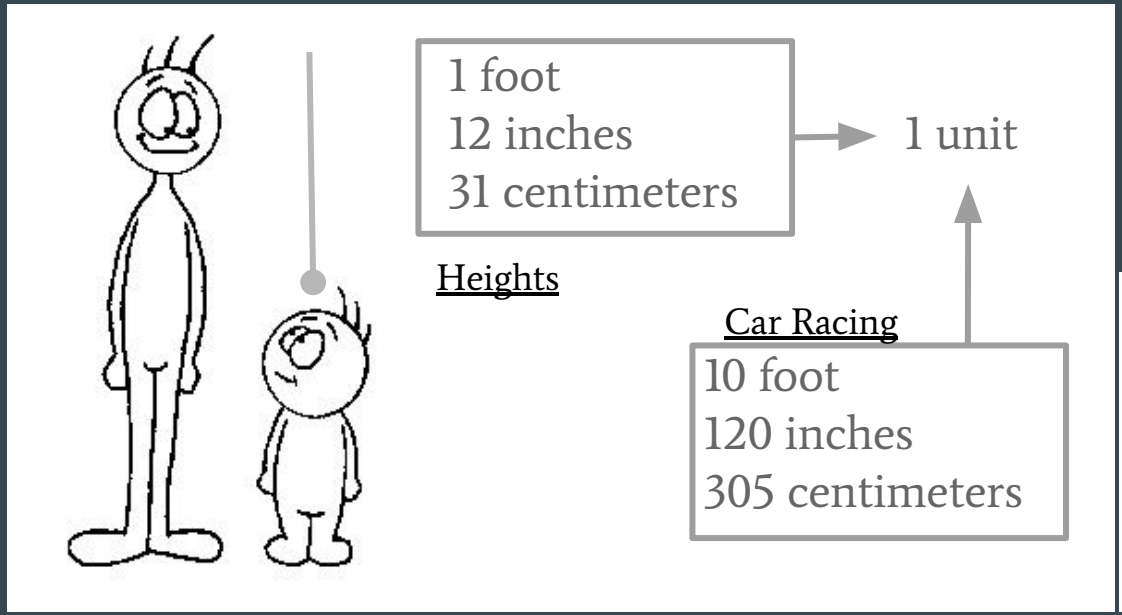
- Many recommendation engines are based on algorithms that focus on distance to determine a similar item or user and make a corresponding recommendation.

- If you have numeric data about users or items, it can be important to standardize or normalize this data.

- Using standardizing or normalizing is a way to make sure no matter the units of your data, the observed difference between points is easy and fair to compare to all other differences you might observe in your dataset.

Standardizing and Normalizing Data

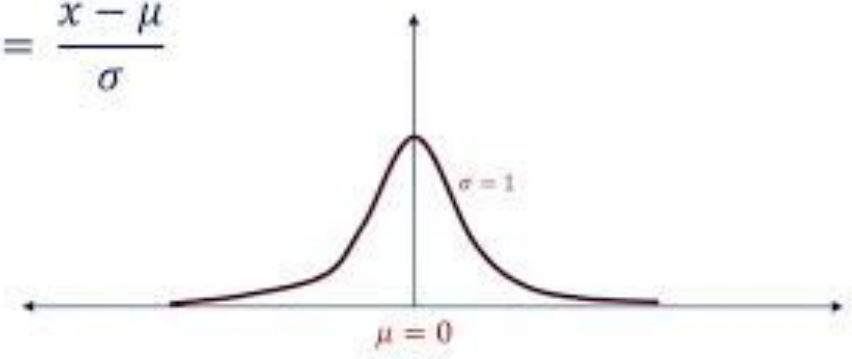
Example:



Standardizing and Normalizing Data

Standardizing

$$z = \frac{x - \mu}{\sigma}$$



Normalizing [0, 1]

$$z = \frac{x \text{ - } \min(x)}{\max(x) \text{ - } \min(x)}$$

Encoding Categorical Data

Encoding Categorical Data

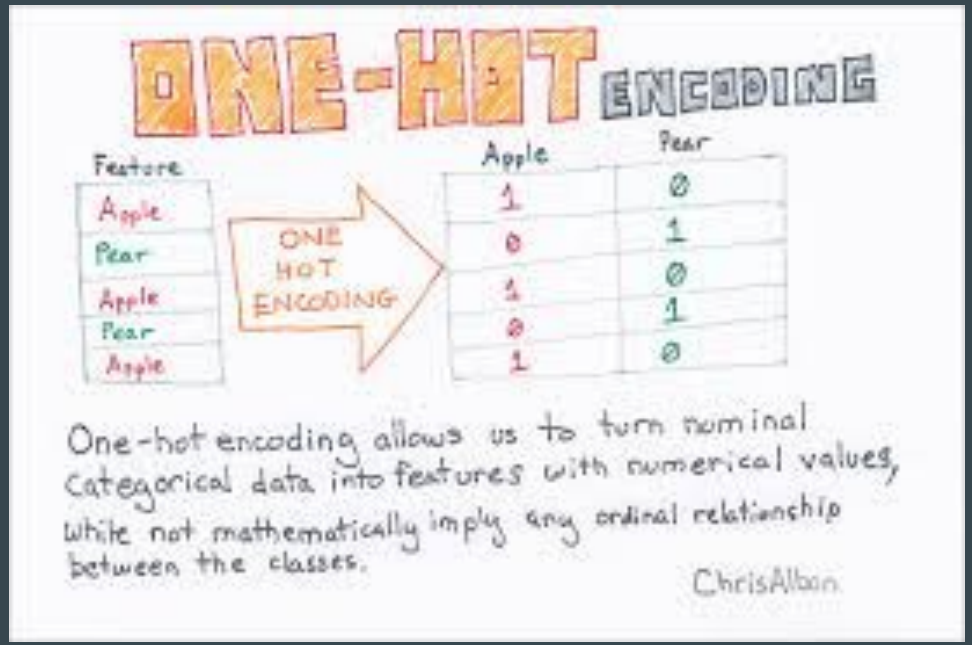
- In order to make use of categorical data, we need to encode it in some way.

- If you don't have too many categories, one of the most common encodings is the use of one-hot encoding or dummy variables.

- If you do have a lot of categories, you might use a more advanced technique like embeddings, principal component analysis, or partial least squares.

Encoding Categorical Data

- One-hot or dummy coding is the most popular way to encode categorical variables and in practice looks like the following:



Chris Albon's work is fantastic - check it out here!

Encoding Text Data

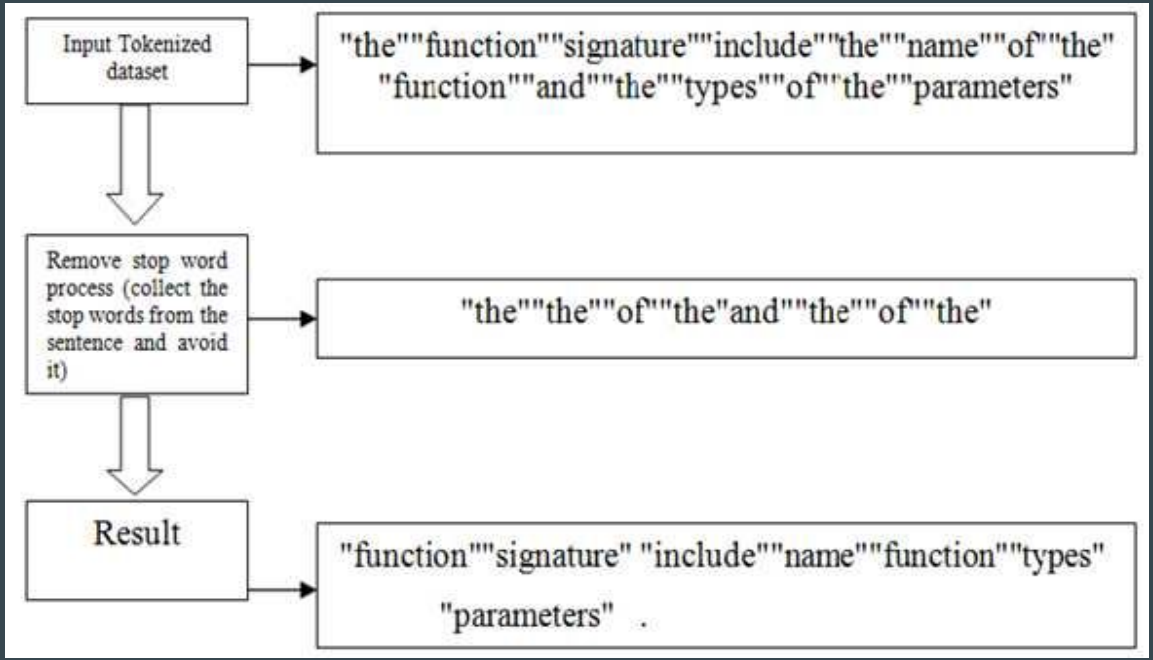
Encoding Text Data

- There is a common process for feature engineering text data, which follows as:
  - Remove stop words
  - Stemming/Lemmatization
  - Term-frequency inverse document-frequency (tf-idf)/Word Embeddings

- Let’s review this process in the next slides.

Encoding Text Data

- Removing stop words.



Encoding Text Data

- Stemming and Lemmatization

Stemming

Stemming is the process of eliminating affixes (suffixed, prefixes, infixes, circumfixes) from a word in order to obtain a word stem.

running → run

Lemmatization

Lemmatization is related to stemming, differing in that lemmatization is able to capture canonical forms based on a word's lemma.

For example, stemming the word "better" would fail to return its citation form (another word for lemma); however, lemmatization would result in the following:

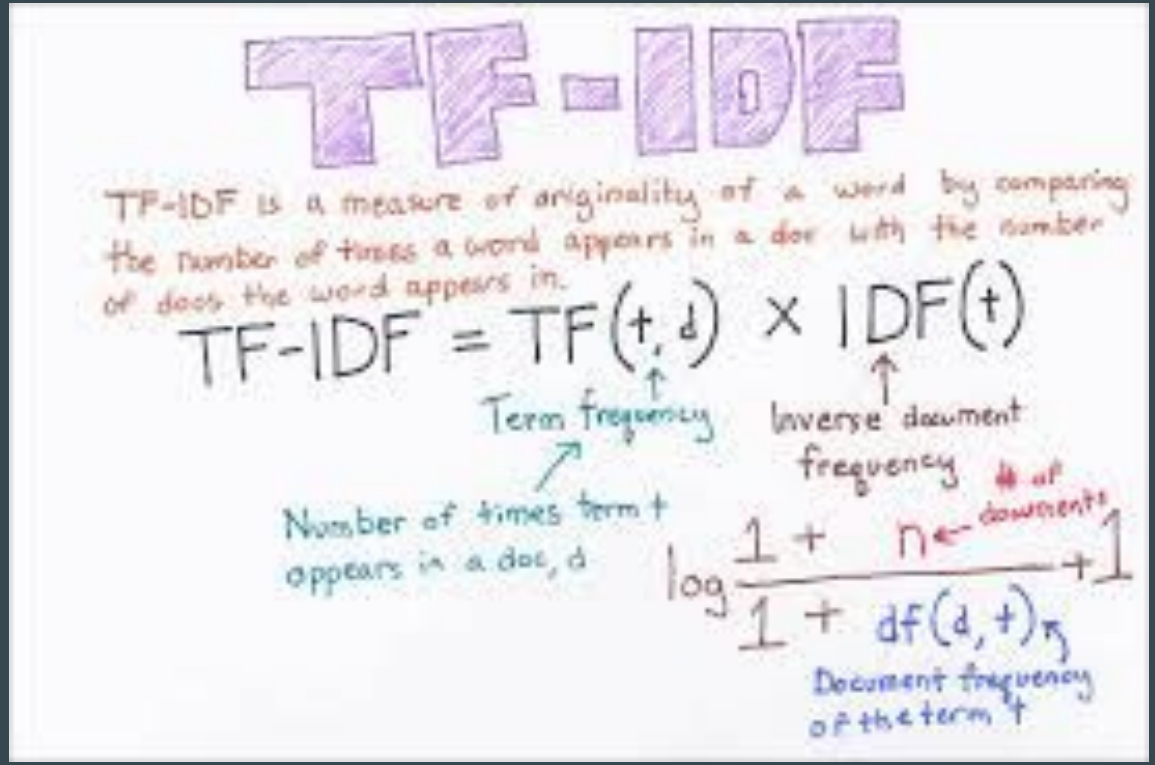
better → good

Encoding Text Data

- Did I mention: Chris Albon's work is fantastic - check it out here!

- Tf-idf is one of the most popular techniques for encoding text.

- You can also use Embeddings like word2vec.



Missing Values

Missing Values

- Missing values in recommendation systems are usually due to the fact that a particular user-item interaction doesn't exist.

- Due to this, you generally do not want to fill these values in.

- In the next section, we will see recommendation methods that handle this missing values kindly.

Outliers

Outliers

- Imagine you are collecting ratings on a 1-5 scale.
- You find a rating of 8, what do you do?
- How about a rating of 0?
- In many cases, engineering will control the process to ensure this doesn't happen, but you never know.
- If you can't be sure of what the entry should truly be, it is likely best to simply remove the data point rather than try to impute some value.

Recap

- For collaborative filtering (the most common recommendation method), your data will fall into either sparse or dense matrix dataframes with user-item ratings stored.

- If you are interested in performing other methods of creating recommendations, it is important to consider how to create features of the user and item data available.
  - Normalizing or standardizing
  - Tf-idf/Embeddings (for text)
  - Removing or Handling Outliers
  - Dummy (One Hot Encoding)
  - Filling missing values