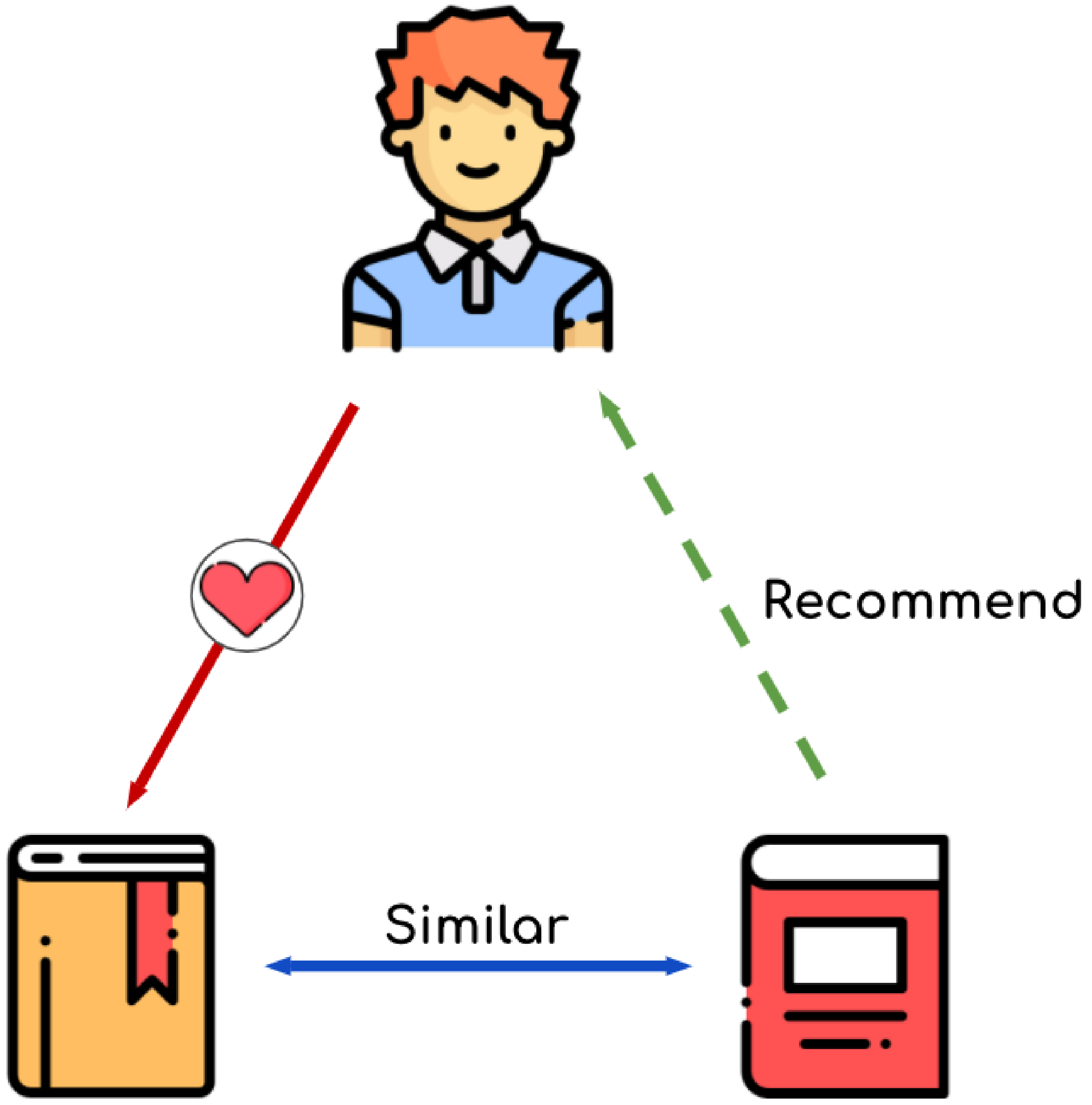
# Intro to content-based recommendations

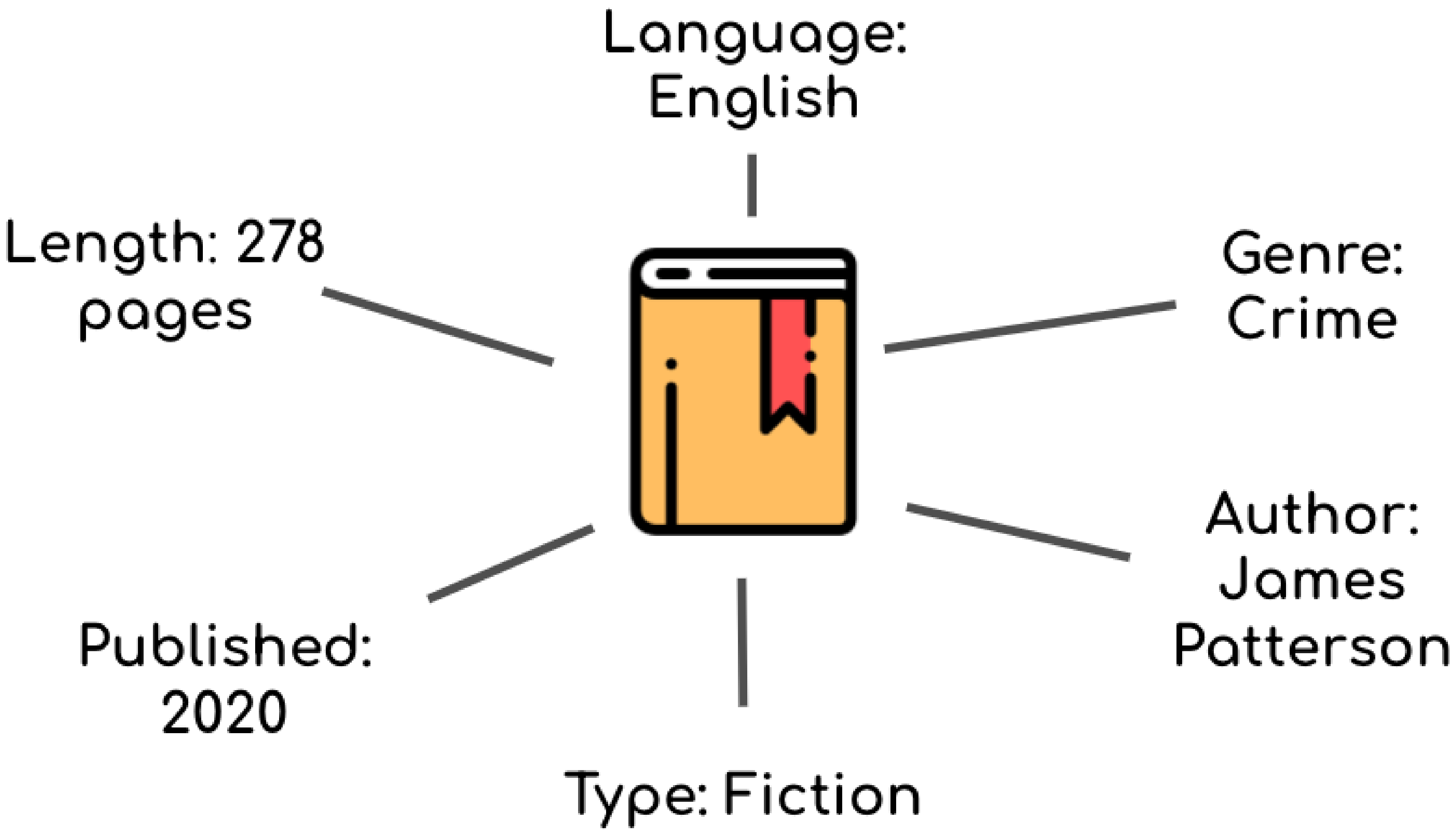
BUILDING RECOMMENDATION ENGINES IN PYTHON

**Rob O'Callaghan**
Director of Data

# What are content-based recommendations?



# Items' attributes or characteristics



# Vectorizing your attributes

| ITEM | Attribute 1 | Attribute 2 | Attribute 3 | Attribute 4 |
|---|---|---|---|---|
| Item_001 | 0 | 1 | 1 | 0 |
| Item_002 | 1 | 0 | 1 | 0 |
| Item_003 | 0 | 1 | 0 | 1 |

# One to many relationships

| Book | Genre |
| --- | --- |
| The Hobbit | Adventure |
| The Hobbit | Fantasy |
| The Great Gatsby | Tragedy |
| ... | ... |

| Book | Adventure | Fantasy | Tragedy | ... |
| --- | --- | --- | --- | --- |
| The Hobbit | 1 | 1 | 0 | ... |
| The Great Gatsby | 0 | 0 | 1 | ... |
| ... | ... | ... | ... | ... |

# Crosstabulation

```
pd.crosstab(                              ,                                 )
```

# Crosstabulation

```
pd.crosstab(book_genre_df['Book'], book_genre_df['Genre'])
```

| Book | Adventure | Fantasy | Tragedy | Social commentary |
|---|---|---|---|---|
| The Hobbit | 1 | 1 | 0 | 0 |
| The Great Gatsby | 0 | 0 | 1 | 1 |
| A Game of Thrones | 0 | 1 | 0 | 0 |
| Macbeth | 0 | 0 | 1 | 0 |
| ... | ... | ... | ... | ... |

# Let's practice!

BUILDING RECOMMENDATION ENGINES IN PYTHON

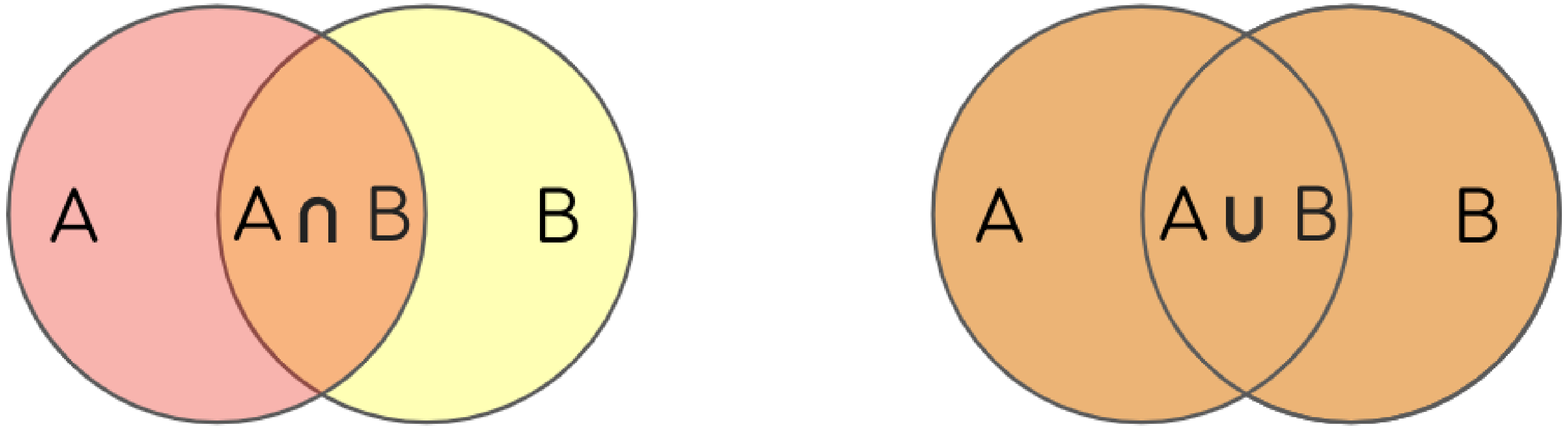
# Making content-based recommendations

BUILDING RECOMMENDATION ENGINES IN PYTHON

**Rob O'Callaghan**
Director of Data

# Introducing the Jaccard similarity

Jaccard similarity:




The number of attributes that two items have in common

The total number of their combined attributes

$$J(A,B) = \frac{A \cap B}{A \cup B}$$

# Calculating Jaccard similarity between books

`genres_array_df` :

| Book | Adventure | Fantasy | Tragedy | Social commentary | ... |
|---|---|---|---|---|---|
| The Hobbit | 1 | 1 | 0 | 0 | ... |
| The Great Gatsby | 0 | 0 | 1 | 1 | ... |
| A Game of Thrones | 0 | 1 | 0 | 0 | ... |
| Macbeth | 0 | 0 | 1 | 0 | ... |
| ... | ... | ... | ... | ... | ... |

# Calculating Jaccard similarity between books

```python
from sklearn.metrics import jaccard_score

hobbit_row = book_genre_df.loc['The Hobbit']
GOT_row = book_genre_df.loc['A Game of Thrones']

print(jaccard_score(hobbit_row, GOT_row))
```

```
0.5
```

# Finding the distance between all items

```
from scipy.spatial.distance import pdist, squareform

jaccard_distances = pdist(book_genre_df.values, metric='jaccard')
print(jaccard_distances)
```

```
[1.  0.5 1.  1.  0.5 1. ]
```

```
square_jaccard_distances = squareform(jaccard_distances)
print(square_jaccard_distances)
```

```
[[0.  1.  0.5 1. ]
 [1.  0.  1.  0.5]
 [0.5 1.  0.  1. ]
 [1.  0.5 1.  0. ]]
```

# Finding the distance between all items

```
print(square_jaccard_distances)
```

```
[[0.  1.  0.5 1. ]
 [1.  0.  1.  0.5]
 [0.5 1.  0.  1. ]
 [1.  0.5 1.  0. ]]
```

```
jaccard_similarity_array = 1 -  square_jaccard_distances
print(jaccard_similarity_array)
```

```
[[1.  0.  0.5 0. ]
 [0.  1.  0.  0.5]
 [0.5 0.  1.  0. ]
 [0.  0.5 0.  1. ]]
```

# Creating a usable distance table

```
distance_df = pd.DataFrame(jaccard_similarity_array,
                           index=genres_array_df['Book'],
                           columns=genres_array_df['Book'])
distance_df.head()
```

```
                The Hobbit The Great Gatsby  A Game of Thrones          Macbeth      ...
The Hobbit            1.00             0.15               0.75             0.01      ...
The Great Gatsby  0.15                 1.00               0.01             0.43      ...
...
```

# Comparing books

```
print(distance_df['The Hobbit']['A Game of Thrones'])
```

```
0.75
```

```
print(distance_df['The Hobbit']['The Great Gatsby'])
```

```
0.15
```

# Finding the most similar books

```
print(distance_df['The Hobbit'].sort_values(ascending=False))
```

```
title
The Hobbit                                                    1.00
The Two Towers                                                0.91
A Game of Thrones                                             0.50
...
```

# Let's practice!

BUILDING RECOMMENDATION ENGINES IN PYTHON

# Text-based similarities

BUILDING RECOMMENDATION ENGINES IN PYTHON

**Rob O'Callaghan**
Director of Data

# Working without clear attributes


Roll over image to zoom in

Play-Doh Modeling Compound 10 Pack Case of Colors, Non-Toxic, Assorted Colors, 2 Oz Cans, Ages 2 & Up, (Amazon Exclusive), Multicolor

Visit the Play-Doh Store

11,347 ratings | 108 answered questions

Amazon's Choice for "play doh"

Price: $7.99 & FREE Returns

Get $50 off instantly: Pay $0.00 ~~$7.99~~ upon approval for the Amazon Rewards Visa Card. No annual fee.

Available at a lower price from other sellers that may not offer free Prime shipping.

Size: **2 oz cans**

- 10 cans of creative fun kids can get creative with just the right colors They need in this Play Doh 10 pack of 2 ounce cans
- Shape your imagination squishy, bright, non toxic Play Doh compound sparks imaginations For kids 2 and up who love arts and crafts like Modeling clay
- Easy open, recyclable package ships in simple recyclable packaging that's easy to open and frustration free, and the Play Doh cans and lids are also recyclable to help build a more sustainable world
- Notice to parents: contains wheat
- Just the right colors to start shape, squish, mix, and make it all. Great for lots of uses like Play Doh refills, as a Play Doh starter set, or as an add on to any Play Doh toy (sold separately)

New & Used (10) from $7.35 & FREE Shipping on orders over $25.00

Item Description

# Term frequency inverse document frequency

$$\text{TF-IDF} = \frac{\frac{\text{Count of word occurrences}}{\text{Total words in document}}}{\log(\frac{\text{Number of docs word is in}}{\text{Total number of docs}})}$$

# Our data

`book_summary_df`:

| Book | Description |
|---|---|
| The Hobbit | "Bilbo Baggins lives a simple life with his fellow hobbits in the shire..." |
| The Great Gatsby | "Set in Jazz Age New York, the novel tells the tragic story of Jay ..." |
| A Game of Thrones | "15 years have passed since Robert's rebellion, with a nine-year-long ..." |
| Macbeth | "A brave Scottish general receives a prophecy from a trio of witches ..." |
| ... | ... |

# Instantiate the vectorizer

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfidfvec = TfidfVectorizer(          ,             )
```

# Filtering the data

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfidfvec = TfidfVectorizer(min_df=2,             )
```

# Filtering the data

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfidfvec = TfidfVectorizer(min_df=2, max_df=0.7)
```

# Vectorizing the data

```
vectorized_data = tfidfvec.fit_transform(book_summary_df['Descriptions'])
print(tfidfvec.get_feature_names)
```

```
['age', 'ancient', 'angry', 'brave', 'battle', 'fellow', 'game', 'general', ...]
```

```
print(vectorized_data.to_array())
```

```
[[0.21,      0.53,    0.41,    0.64,     0.01,     0.02,     ...
 [0.31,      0.00,    0.42,    0.03,     0.00,     0.73,     ...
 [...,        ...,     ...,     ...,      ...,      ...,     ...
```

# Formatting the data

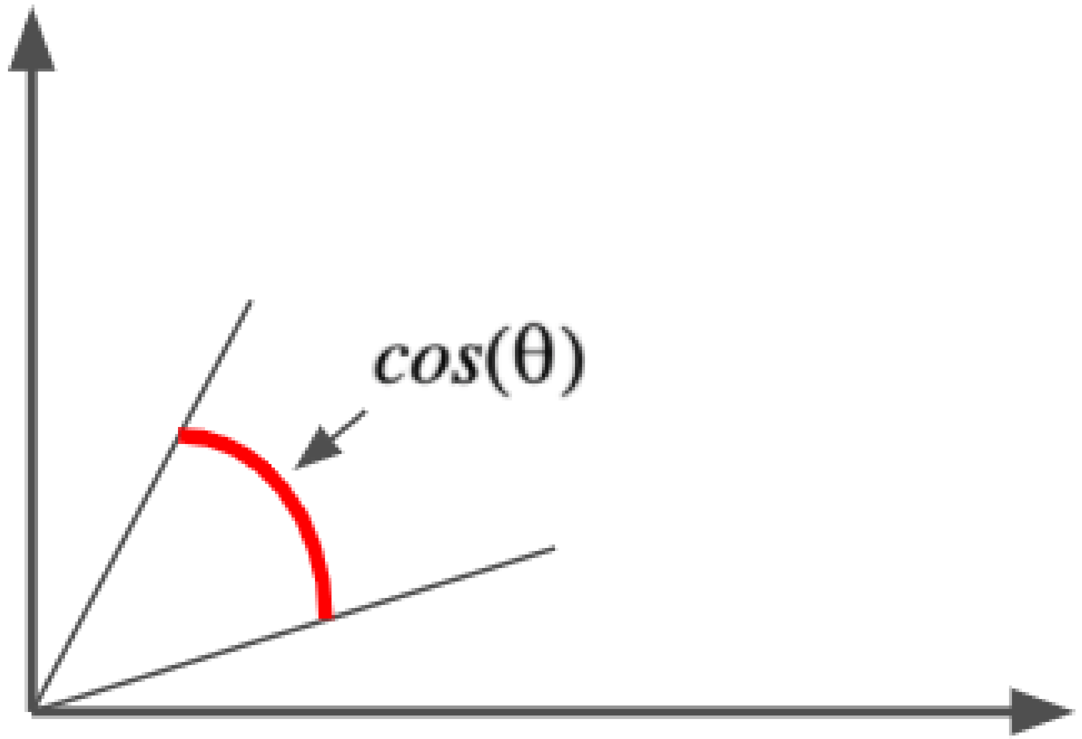
```
tfidf_df = pd.DataFrame(vectorized_data.toarray(),
                        columns=tfidfvec.get_feature_names())
tfidf_df.index = book_summary_df['Book']
print(tfidf_df)
```

```
                  | 'age'| 'ancient'| 'angry'| 'brave'| 'battle'| 'fellow'|...
|-----------------|------|----------|--------|--------|---------|---------|...
| The Hobbit      |  0.21|      0.53|    0.41|    0.64|     0.01|     0.02|...
| The Great Gatsby |  0.31|      0.00|    0.42|    0.03|     0.00|     0.73|...
| A Game of Thrones|  0.61|      0.42|    0.77|    0.31|     0.83|     0.03|...
|              ...|   ...|       ...|     ...|     ...|      ...|      ...|...
```

# Cosine similarity

Cosine Distance:

$$cos(\theta) = \frac{A.B}{||A|| \cdot ||B||}$$



# Cosine similarity

```
from sklearn.metrics.pairwise import cosine_similarity


# Find similarity between all items
cosine_similarity_array = cosine_similarity(tfidf_summary_df)
```

```
# Find similarity between two items
cosine_similarity(tfidf_df.loc['The Hobbit'].values.reshape(1, -1),
                  tfidf_df.loc['Macbeth'].values.reshape(1, -1))
```

# Let's practice!

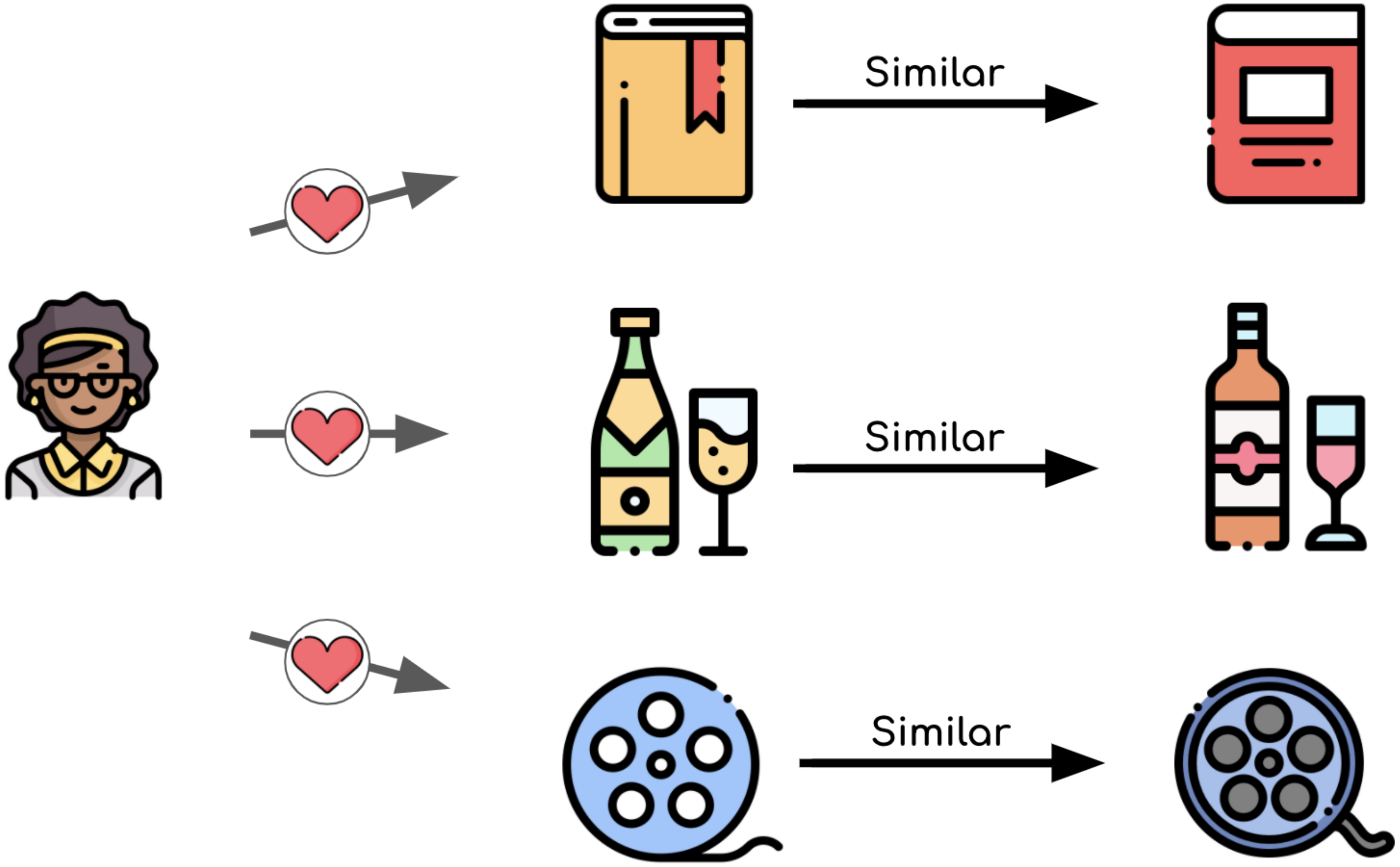
BUILDING RECOMMENDATION ENGINES IN PYTHON

# User profile recommendations

BUILDING RECOMMENDATION ENGINES IN PYTHON

**Rob O'Callaghan**
Director of Data

# Item to item recommendations



# User profiles

`tfidf_summary_df`:

| Book | Adventure | Fantasy | Tragedy | Social commentary |
|---|---|---|---|---|
| The Hobbit | 1 | 1 | 0 | 0 |
| Macbeth | 0 | 0 | 1 | 0 |
| ... | ... | ... | ... | ... |

User Profile:

| User Profile | Adventure | Fantasy | Tragedy | Social commentary |
|---|---|---|---|---|
| User_001 | ??? | ??? | ??? | ??? |

# Extract the user data

```
list_of_books_read = ['The Hobbit', 'Foundation', 'Nudge']
user_books = tfidf_summary_df.reindex(list_of_books_read)
print(user_books)
```

```
              age    ancient    angry    brave    battle    fellow    ...
The Hobbit   0.21       0.53     0.41     0.64      0.01      0.02    ...
Foundation   0.31       0.90     0.42     0.33      0.64      0.04    ...
     Nudge   0.61       0.01     0.45     0.31      0.12      0.74    ...
```

# Build the user profile

```
user_prof = user_movies.mean()
print(user_prof)
```

```
age        0.376667
ancient    0.480000
angry      0.426667
brave      0.256667
               ...
```

```
print(user_prof.values.reshape(1,-1))
```

```
[0.376667, .480000, 0.426667, 0.256667, ...]
```

# Finding recommendations for a user

```
# Create a subset of only the non read books
non_user_movies = tfidf_summary_df.drop(list_of_movies_seen, axis=0)

# Calculate the cosine similarity between all rows
user_prof_similarities = cosine_similarity(user_prof.values.reshape(1, -1),
                                           non_user_movies)
# Wrap in a DataFrame for ease of use
user_prof_similarities_df = pd.DataFrame(user_prof_similarities.T,
                                         index=tfidf_summary_df.index,
                                         columns=["similarity_score"])
```

# Getting the top recommendations

```
sorted_similarity_df = user_prof_similarities.sort_values(by="similarity_score",
                                                           ascending=False)
print(sorted_similarity_df)
```

```
                        similarity_score
Title
The Two Towers                  0.422488
Dune                            0.363540
The Magicians Nephew            0.316075
...                             ...
```

# Let's practice!

BUILDING RECOMMENDATION ENGINES IN PYTHON