Philipps Universität Marburg

Step 2: Integration

TODO

https://github.com/Paralian/umr-data-integration-project-the-TODO-team.git

Juan Fernando Maya

Ilya Zykov

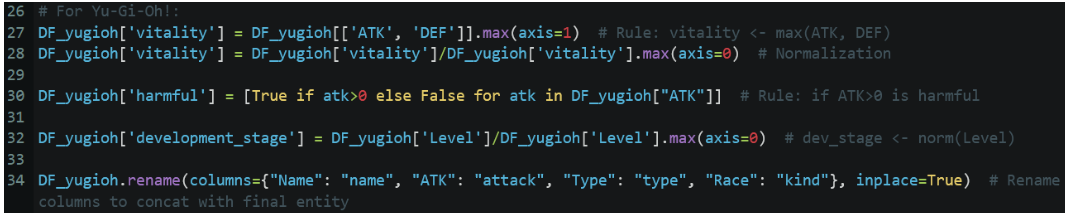
Miron Brandeis

Integration of RPG Entities
Scope

- Attributes in the integrated entity are supposed to be:
  - Essential:
    - ID: name, type, kind
    - Metrics: development, vitality,
  - Comparable/Classifiable
  - Aligned with the general idea of an entity in an RPG
    - They all have common way of functioning
    - Features are mostly similar, their functionality is very different
    - Dice from D&D, Card rules from YuGiOh, HP from Skyrim need conversion into universal system

Pipeline

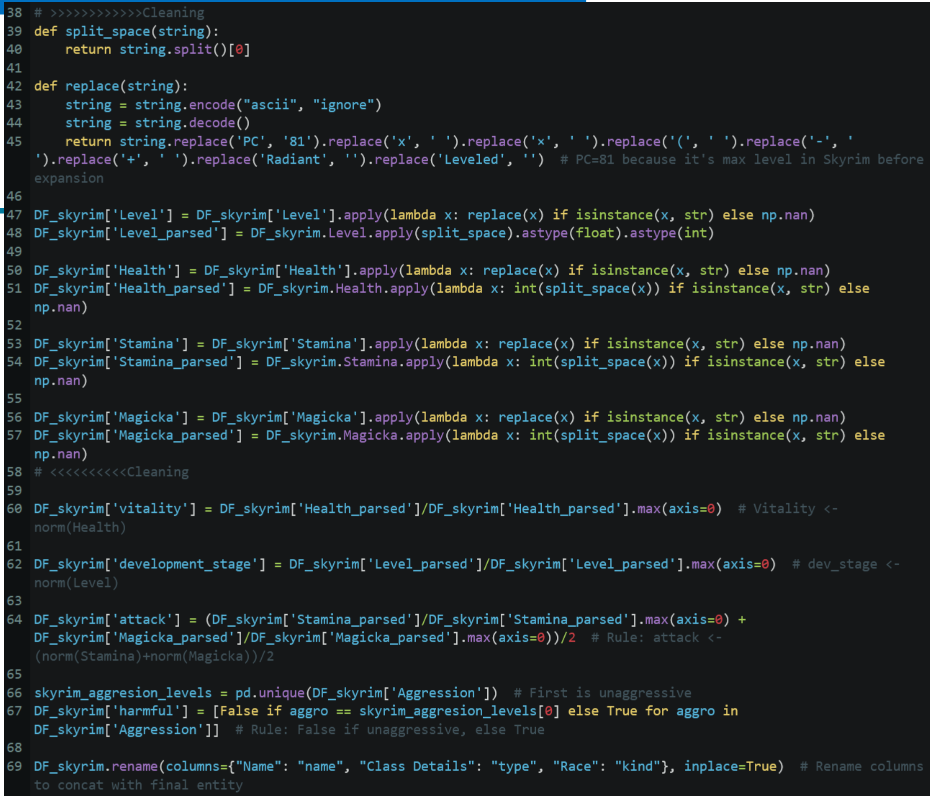
- Source (YuGiOh) → target:
  - Name → name
  - Type → type
  - Race → kind
  - Level → development stage
  - max(ATK, DEF) → vitality
  - ATK → attack
  - Monster → harmful
    - (boolean to boolean transfer, treat null as false)



```python
# For Yu-Gi-Oh!:
DF_yugioh['vitality'] = DF_yugioh[['ATK', 'DEF']].max(axis=1)  # Rule: vitality <- max(ATK, DEF)
DF_yugioh['vitality'] = DF_yugioh['vitality']/DF_yugioh['vitality'].max(axis=0)  # Normalization

DF_yugioh['harmful'] = [True if atk>0 else False for atk in DF_yugioh["ATK"]]  # Rule: if ATK>0 is harmful

DF_yugioh['development_stage'] = DF_yugioh['Level']/DF_yugioh['Level'].max(axis=0)  # dev_stage <- norm(Level)

DF_yugioh.rename(columns={"Name": "name", "ATK": "attack", "Type": "type", "Race": "kind"}, inplace=True)  # Rename columns to concat with final entity
```

Pipeline

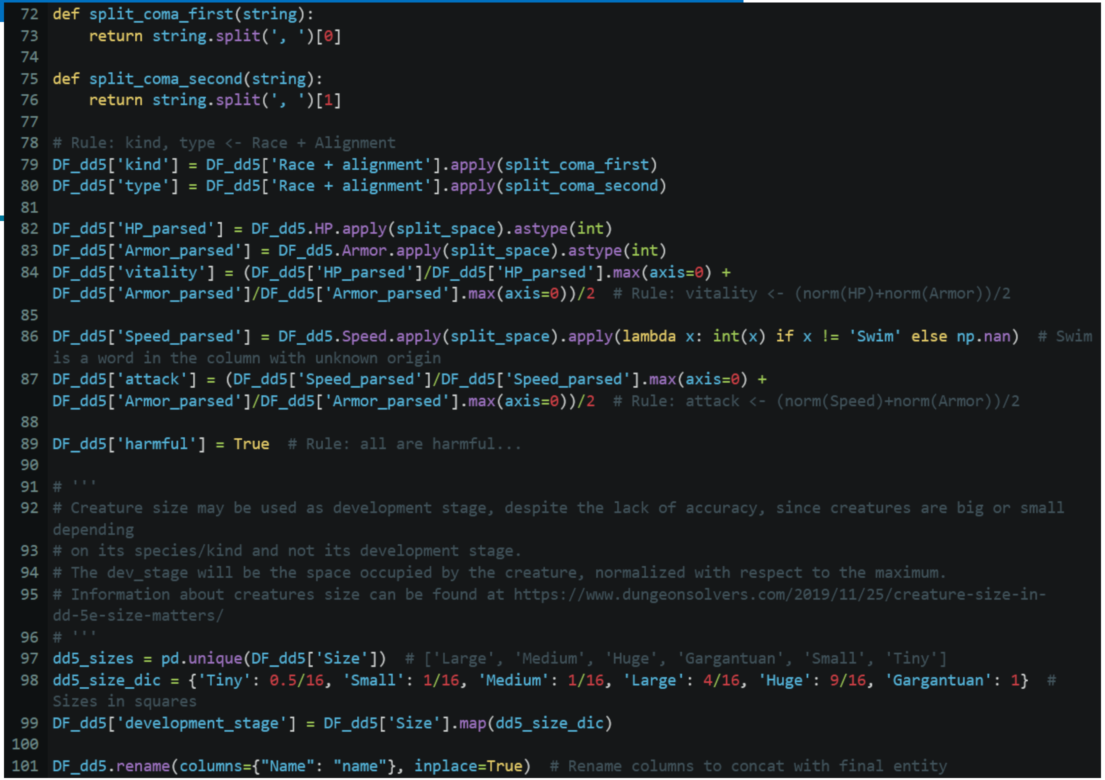
- Source (Skyrim) → target:
  - Name → name
  - Class Details → type
  - Race → kind
  - Level → development stage
  - Health → vitality
  - $\frac{norm(Stamina)+norm(Magicka)}{2}$ → attack
  - !unaggressive → harmful



```python
# >>>>>>>>>>>>Cleaning
def split_space(string):
    return string.split()[0]

def replace(string):
    string = string.encode("ascii", "ignore")
    string = string.decode()
    return string.replace('PC', '81').replace('x', ' ').replace('×', ' ').replace('(', ' ').replace('-', ' ').replace('+', ' ').replace('Radiant', '').replace('Leveled', '')  # PC=81 because it's max level in Skyrim before expansion

DF_skyrim['Level'] = DF_skyrim['Level'].apply(lambda x: replace(x) if isinstance(x, str) else np.nan)
DF_skyrim['Level_parsed'] = DF_skyrim.Level.apply(split_space).astype(float).astype(int)

DF_skyrim['Health'] = DF_skyrim['Health'].apply(lambda x: replace(x) if isinstance(x, str) else np.nan)
DF_skyrim['Health_parsed'] = DF_skyrim.Health.apply(lambda x: int(split_space(x)) if isinstance(x, str) else np.nan)

DF_skyrim['Stamina'] = DF_skyrim['Stamina'].apply(lambda x: replace(x) if isinstance(x, str) else np.nan)
DF_skyrim['Stamina_parsed'] = DF_skyrim.Stamina.apply(lambda x: int(split_space(x)) if isinstance(x, str) else np.nan)

DF_skyrim['Magicka'] = DF_skyrim['Magicka'].apply(lambda x: replace(x) if isinstance(x, str) else np.nan)
DF_skyrim['Magicka_parsed'] = DF_skyrim.Magicka.apply(lambda x: int(split_space(x)) if isinstance(x, str) else np.nan)
# <<<<<<<<<<Cleaning

DF_skyrim['vitality'] = DF_skyrim['Health_parsed']/DF_skyrim['Health_parsed'].max(axis=0)  # Vitality <- norm(Health)

DF_skyrim['development_stage'] = DF_skyrim['Level_parsed']/DF_skyrim['Level_parsed'].max(axis=0)  # dev_stage <- norm(Level)

DF_skyrim['attack'] = (DF_skyrim['Stamina_parsed']/DF_skyrim['Stamina_parsed'].max(axis=0) + DF_skyrim['Magicka_parsed']/DF_skyrim['Magicka_parsed'].max(axis=0))/2  # Rule: attack <- (norm(Stamina)+norm(Magicka))/2

skyrim_aggresion_levels = pd.unique(DF_skyrim['Aggression'])  # First is unaggressive
DF_skyrim['harmful'] = [False if aggro == skyrim_aggresion_levels[0] else True for aggro in DF_skyrim['Aggression']]  # Rule: False if unaggressive, else True

DF_skyrim.rename(columns={"Name": "name", "Class Details": "type", "Race": "kind"}, inplace=True)  # Rename columns to concat with final entity
```

Pipeline

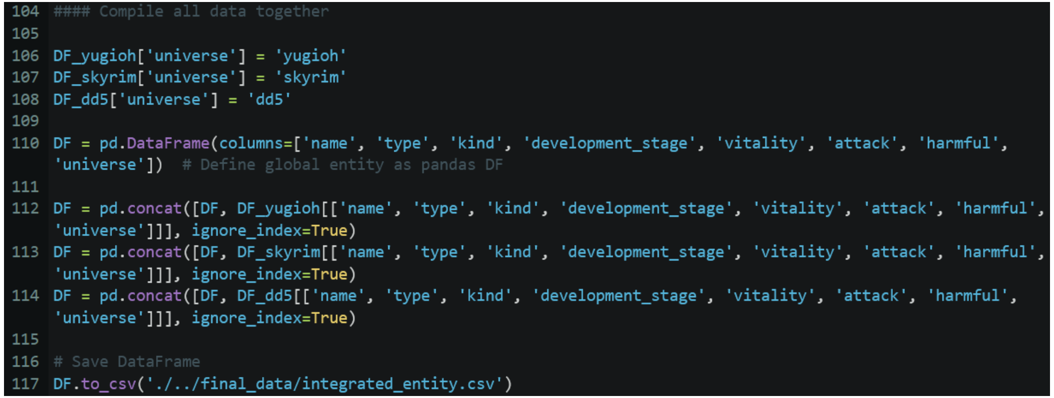
- Source (D&D) → target:
  - Name → name
  - Race → type
  - Alignment → kind
  - Size → development stage
  - $\frac{norm(HP)+norm(Armor)}{2}$ → vitality
  - $\frac{norm(Speed)+norm(Armor)}{2}$ → attack
  - harmful:=true (for all)



```python
def split_coma_first(string):
    return string.split(', ')[0]

def split_coma_second(string):
    return string.split(', ')[1]

# Rule: kind, type <- Race + Alignment
DF_dd5['kind'] = DF_dd5['Race + alignment'].apply(split_coma_first)
DF_dd5['type'] = DF_dd5['Race + alignment'].apply(split_coma_second)

DF_dd5['HP_parsed'] = DF_dd5.HP.apply(split_space).astype(int)
DF_dd5['Armor_parsed'] = DF_dd5.Armor.apply(split_space).astype(int)
DF_dd5['vitality'] = (DF_dd5['HP_parsed']/DF_dd5['HP_parsed'].max(axis=0) + DF_dd5['Armor_parsed']/DF_dd5['Armor_parsed'].max(axis=0))/2  # Rule: vitality <- (norm(HP)+norm(Armor))/2

DF_dd5['Speed_parsed'] = DF_dd5.Speed.apply(split_space).apply(lambda x: int(x) if x != 'Swim' else np.nan)  # Swim is a word in the column with unknown origin
DF_dd5['attack'] = (DF_dd5['Speed_parsed']/DF_dd5['Speed_parsed'].max(axis=0) + DF_dd5['Armor_parsed']/DF_dd5['Armor_parsed'].max(axis=0))/2  # Rule: attack <- (norm(Speed)+norm(Armor))/2

DF_dd5['harmful'] = True  # Rule: all are harmful...

# '''
# Creature size may be used as development stage, despite the lack of accuracy, since creatures are big or small depending
# on its species/kind and not its development stage.
# The dev_stage will be the space occupied by the creature, normalized with respect to the maximum.
# Information about creatures size can be found at https://www.dungeonsolvers.com/2019/11/25/creature-size-in-dd-5e-size-matters/
# '''
dd5_sizes = pd.unique(DF_dd5['Size'])  # ['Large', 'Medium', 'Huge', 'Gargantuan', 'Small', 'Tiny']
dd5_size_dic = {'Tiny': 0.5/16, 'Small': 1/16, 'Medium': 1/16, 'Large': 4/16, 'Huge': 9/16, 'Gargantuan': 1}  # Sizes in squares
DF_dd5['development_stage'] = DF_dd5['Size'].map(dd5_size_dic)

DF_dd5.rename(columns={"Name": "name"}, inplace=True)  # Rename columns to concat with final entity
```

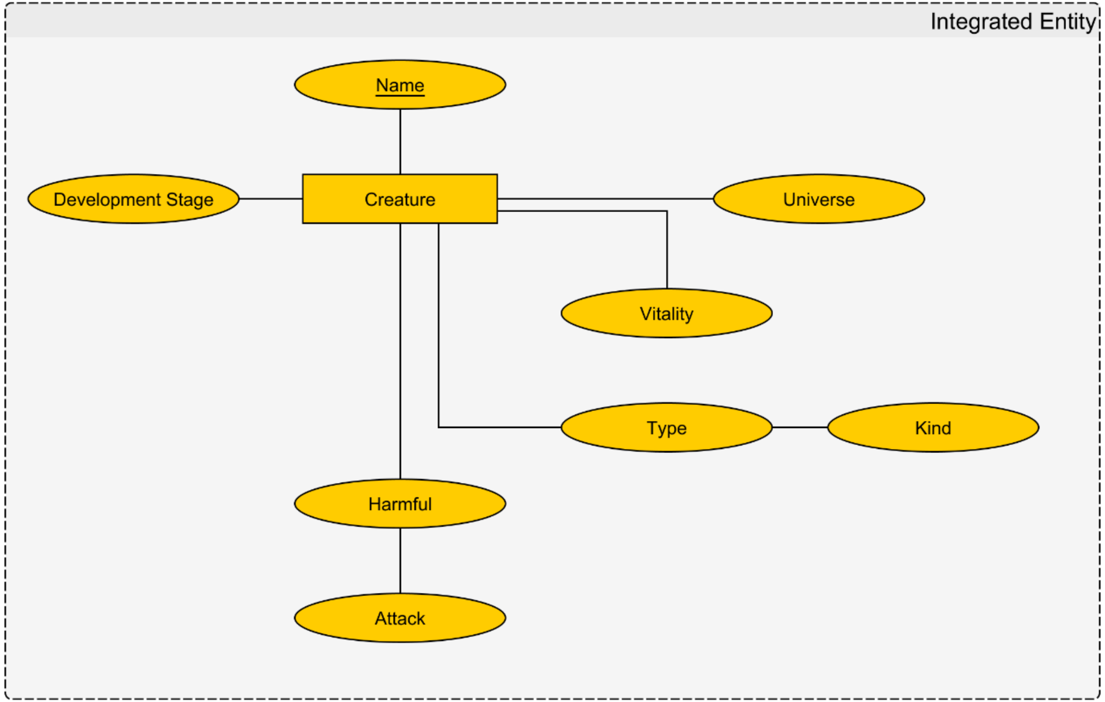
Pipeline

- Putting all dataframes together
- Saving the results



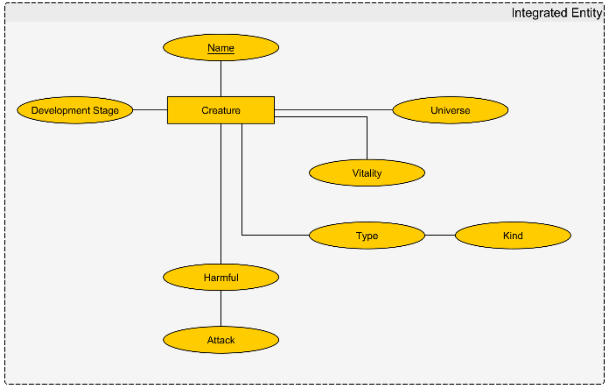
```python
#### Compile all data together

DF_yugioh['universe'] = 'yugioh'
DF_skyrim['universe'] = 'skyrim'
DF_dd5['universe'] = 'dd5'

DF = pd.DataFrame(columns=['name', 'type', 'kind', 'development_stage', 'vitality', 'attack', 'harmful', 'universe'])  # Define global entity as pandas DF

DF = pd.concat([DF, DF_yugioh[['name', 'type', 'kind', 'development_stage', 'vitality', 'attack', 'harmful', 'universe']]], ignore_index=True)
DF = pd.concat([DF, DF_skyrim[['name', 'type', 'kind', 'development_stage', 'vitality', 'attack', 'harmful', 'universe']]], ignore_index=True)
DF = pd.concat([DF, DF_dd5[['name', 'type', 'kind', 'development_stage', 'vitality', 'attack', 'harmful', 'universe']]], ignore_index=True)

# Save DataFrame
DF.to_csv('./../final_data/integrated_entity.csv')
```

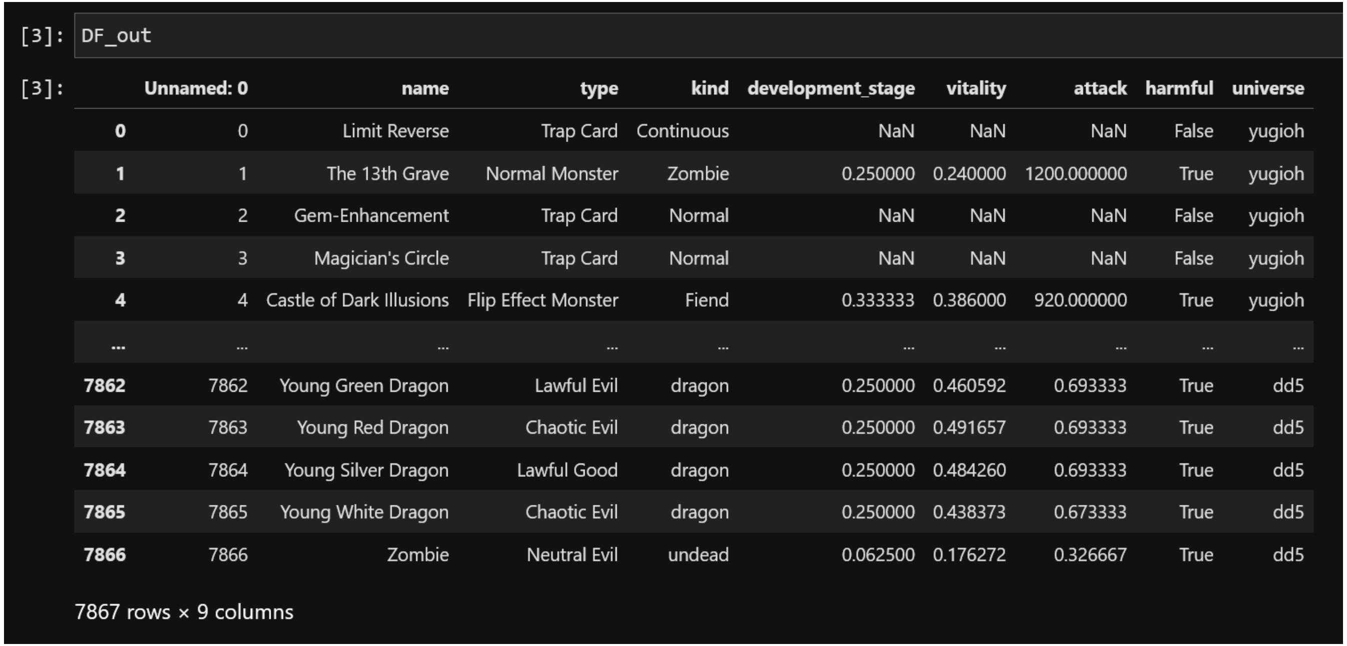
Resulting Entity-Relation model



Resulting dataframe

- Resulting integrated Entity + metadata





[3]:
```
DF_out
```

[3]:

	Unnamed: 0	name	type	kind	development_stage	vitality	attack	harmful	universe
0	0	Limit Reverse	Trap Card	Continuous	NaN	NaN	NaN	False	yugioh
1	1	The 13th Grave	Normal Monster	Zombie	0.250000	0.240000	1200.000000	True	yugioh
2	2	Gem-Enhancement	Trap Card	Normal	NaN	NaN	NaN	False	yugioh
3	3	Magician's Circle	Trap Card	Normal	NaN	NaN	NaN	False	yugioh
4	4	Castle of Dark Illusions	Flip Effect Monster	Fiend	0.333333	0.386000	920.000000	True	yugioh
...	...	...	...	...	...	...	...	...	...
7862	7862	Young Green Dragon	Lawful Evil	dragon	0.250000	0.460592	0.693333	True	dd5
7863	7863	Young Red Dragon	Chaotic Evil	dragon	0.250000	0.491657	0.693333	True	dd5
7864	7864	Young Silver Dragon	Lawful Good	dragon	0.250000	0.484260	0.693333	True	dd5
7865	7865	Young White Dragon	Chaotic Evil	dragon	0.250000	0.438373	0.673333	True	dd5
7866	7866	Zombie	Neutral Evil	undead	0.062500	0.176272	0.326667	True	dd5

7867 rows × 9 columns