Elden ring steam reviews

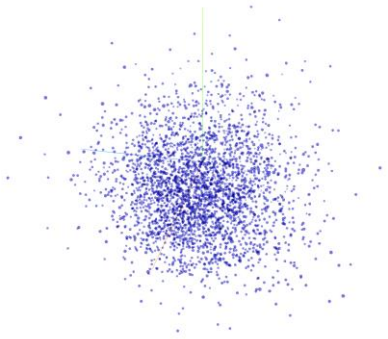
About the dataset

Elden Ring is an action role-playing video game that has been skillfully developed by From Software and adeptly published by Bandai Namco Entertainment. The game is being masterfully directed by Hidetaka Miyazaki, renowned for his exceptional work on the highly acclaimed video games Dark Souls and Bloodborne. Set in a dark fantasy world, created by Miyazaki and the esteemed author George R.R. Martin, Elden Ring offers open-world exploration and intense combat akin to other renowned games from From Software. The dataset comprises several columns, including user ID, review language, review content, review creation date, vote count, comment count, purchase history on Steam, game acquisition status, and author's total playtime on Steam. It includes English reviews of Elden Ring posted on the Steam platform from January 29th, 2023 to March 7th, 2023. (https://www.kaggle.com/datasets/noahx1/elden-ring-steam-reviews)

About the model

Based on the lab codes provided, I made several modifications to the preprocess definition. For example, as the raw data contains some floats, I opted to first remove all punctuation and numeric characters, followed by converting all text to lowercase. Additionally, to circumvent any potential errors that may arise when processing the raw data, I converted all input text to string format.[1] Furthermore, I tailored certain parameters to enhance the information representation. Specifically, I increased the vector size to 300 and the number of epochs to 100. Additionally, to capture a more comprehensive contextual understanding, I augmented the window size to 10.[2] Since all the reviews are about Elden Ring, I also removed task specific stopwords: “elden” and “ring”.

Visualization



Findings

As Elden Ring is a soulslike game, players are bound to experience multiple failures, numerous attempts, and even spend several hours “attempting” to defeat a challenging boss in the game. Consequently, the term "killed" is often used in conjunction with characters possessing high HP and attack power, such as "rats" and "dragons". Upon each death, players are "respawned" at designated locations called Sites of Grace.[3] When discussing the game's purchase, the majority of players believe the game to be a “worthy” investment, while a fraction of players consider it a “waste” of money.[4]

Furthermore, a noteworthy observation is the strong association of the term "ending" with the word "frenzied". Elden Ring boasts a total of 6 possible endings, with the "frenzied" ending being among them. It is surprising to note that the aforementioned ending is only achieved by a mere 13.9% of players, as indicated by Steam statistics. This could potentially be attributed to the fact that the comments in the dataset were collected approximately a year after the game's initial release, thus limiting the number of new players, with veteran players having already achieved the more common endings and seeking out the more obscure ones, such as the "frenzied" ending. Additionally, it is worth mentioning that the "Age of the Stars" ending, which centers around an NPC named "Ranni", is particularly popular among players. This is likely due to the alluring character design of Ranni, making her a favored non-playable character among players.[5]

Based on the analysis, we can see what a successful game looks like: “entertained” plot,[6] “movie”-like scene,[7] “spectacular” music,[8] and “pleasant” surprise.[9] And Elden Ring is like a delicious late afternoon wine and cracker platter under a sweltering sun, cooled by a spring breeze and relaxing backyard music.

Appendix

1.

```
def preprocess(text):
  text = re.sub(r'[^\w\s]', '', str(text)) #remove punctuations
  text = re.sub(r'\d+', '', str(text)) #remove numbers
  text = text.lower() #lowercase
  text = " ".join(text.split()) #stripWhitespace
  text = text.split()
  text = [x for x in text if x not in stop_words] #remove stopwords
  text = [x for x in text if x not in ["elden", "ring"]] #remove task specific stopwords
  text = " ".join(text)
  # stemmer_ps = PorterStemmer()
  # text = [stemmer_ps.stem(word) for word in text.split()] #stemming
  # text = " ".join(text)
  # lemmatizer = WordNetLemmatizer()
  # text = [lemmatizer.lemmatize(word) for word in text.split()]  #lemmatization
  # text = " ".join(text)
  return(text)
```

2.

```
model = Word2Vec(sentences=data['review_processed'].tolist(), vector_size=300,sg=1,min_count=5,window=10,workers=50,seed=10,epochs=100)
```

3.

```
model.wv.most_similar('killed', topn=10)
```

```
[('exploit', 0.33646878600120544),
 ('elevator', 0.29332464933395386),
 ('rats', 0.2752256393432617),
 ('dragon', 0.2722066044807434),
 ('ruins', 0.26664888858795166),
 ('respawning', 0.2624953091144562),
 ('rat', 0.2578144371509552),
 ('respawn', 0.2554304301738739),
 ('grace', 0.2504226565361023),
 ('attempts', 0.24889318645000458)]
```

4.

```
model.wv.most_similar('money', topn=10)
```

```
[('worth', 0.3450014591217041),
 ('gang', 0.3420043885707855),
 ('dollars', 0.3371058404445648),
 ('infinity', 0.3294423818588257),
 ('spare', 0.3290877342224121),
 ('ark', 0.32248052954673767),
 ('fortnite', 0.3201828896999359),
 ('deaf', 0.30601823329925537),
 ('burn', 0.2964648902416229),
 ('evolved', 0.28911903500556946)]
```

5.
```
model.wv.most_similar('ending', topn=10)
```

```
[('frenzied', 0.38064053654670715),
 ('endings', 0.3343939483165741),
 ('rannis', 0.3159030079841614),
 ('hug', 0.3082789480686188),
 ('azur', 0.30606716871261597),
 ('achievement', 0.30452626943588257),
 ('age', 0.2967379093170166),
 ('accomplished', 0.2804076671600342),
 ('wildly', 0.27869394421577454),
 ('completed', 0.27568483352661133)]
```

6.
```
model.wv.most_similar('plot', topn=10)
```

```
[('gather', 0.30340901017189026),
 ('grasp', 0.3000735640525818),
 ('entertained', 0.29185399413108826),
 ('repeating', 0.29004380106925964),
 ('themes', 0.2753866910934448),
 ('noice', 0.27247852087020874),
 ('tone', 0.27215951681137085),
 ('consoles', 0.2704331576824188),
 ('typical', 0.2691148519515991),
 ('stellar', 0.26908132433891296)]
```

7.
```
model.wv.most_similar('scene', topn=10)
```

```
[('cutting', 0.4096072018146515),
 ('scenes', 0.3821605443954468),
 ('disappoint', 0.36313316226005554),
 ('noice', 0.35858845710754395),
 ('marika', 0.35744160413742065),
 ('occasional', 0.3539219796657562),
 ('movie', 0.34203866124153137),
 ('spell', 0.3334980010986328),
 ('ight', 0.3309401571750641),
 ('social', 0.3287423253059387)]
```

8.
```
model.wv.most_similar('music', topn=10)
```

```
[('spectacular', 0.345890611410141),
 ('seamlessly', 0.3309445083141327),
 ('ost', 0.3130759000778198),
 ('exception', 0.2837921679019928),
 ('beatiful', 0.2778931260108948),
 ('unusual', 0.2770550549030304),
 ('cutting', 0.26338258385658264),
 ('replayability', 0.26288795471191406),
 ('toptier', 0.2606656849384308),
 ('beats', 0.2584938406944275)]
```

9.
```
model.wv.most_similar('surprise', topn=10)
```

```
[('humans', 0.36773577332496643),
 ('company', 0.3245186507701874),
 ('purchased', 0.32205909490585327),
 ('moon', 0.31840193271636963),
 ('beautifully', 0.31718453764915466),
 ('syndrome', 0.30310356616973877),
 ('addiction', 0.2997865378856659),
 ('pleasant', 0.2981395423412323),
 ('veri', 0.2937401831150055),
 ('reminds', 0.29072442650794983)]
```