**Notes for word2vec content based filtering:**

Input: movies.csv and ratings.csv

Movies.csv (movie\_id, title, genres)

Ratings.csv (user\_id, movie\_id, rating, timestamp)

**Steps summary:**

This approach works on the principle based on users’ previous preferences it recommends the movies

Overview:

Data preprocessing:

Merge two datasets together and drop missing values, convert the movie\_id to string data type and check the number of unique users in our dataset by creating a list.

Split data into train and validation datasets – 90/10:

Use 90% of data to create word2vec embeddings and rest 10% as validation dataset

Create sequences in both train and validation datasets

Build word2vec embeddings:

Parameters:

Window = 8 (maximum distance between the current and predicted word within a sentence)

sg = 1 (1 for skip-gram)

hs = 0 (negative sampling will be used)

negative = 15 (negative sampling used, 15 represents the number of noise words to be drawn)

alpha = 0.001 (initial learning rate)

min\_alpha = 0.0001 (learning rate will linearly drop to this value as training progresses)

seed = 34 (random number generator)

build\_vocab:

build vocabulary from a sequence of sentences

progress\_per = 250 (how many words to process before updating the progress)

train model:

parameters:

total\_examples (count of sentences)

epochs = 20 (number of iterations over the corpus)

report\_delay = 1 (seconds to wait before reporting progress)

The model has a vocabulary of 3399 unique words and their vectors of size 100 each.

Extract all vectors:

Extract vectors of all words in our vocab

Two visualization approaches:

Visualize word2vec embeddings by reducing dimensions using tsne

Visualize word2vec embeddings by reducing dimensions using UMAP

Recommend movies:

Create a dictionary of movie id and title

Generate a function called similar\_watch, which will return top 6 similar movies (n=6).

Generate a function called aggregate\_vectors to take the average of all the vectors and find similar movies

The recommendation results are very relevant.



