Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Διοίκηση της Ψηφιακής Επιχείρισης – 8ο Εξάμηνο
Εργασία 4 – Ακ. Έτος 2022-2023

Κυριακόπουλος Γεώργιος – 03118153

Σημείωση: Για μερικά από τα ερωτήματα της εργασίας έχει χρησιμοποιηθεί κώδικας σε Python για υπολογισμούς. Ο σχετικός κώδικας και τα αποτελέσματα του βρίσκονται στο τέλος της εργασίας.

Αναζήτηση – TFIDF

Έχουμε το query (*earth, moon, astro*). Αρχικά, για να βρούμε το *Term Frequency (TF)*, σημειώνουμε τη συχνότητα κάθε όρου σε κάθε ταινία (τίτλος και περιγραφή), λαμβάνοντας υπόψη και τις παράγωγες/σύνθετες λέξεις (*stemming*).

2001: A Space Odyssey

At the "Dawn of Man," a group of hominids encounters a mysterious black monolith alien to their surroundings. To the strains of Strauss's 1896 Also sprach Zarathustra, a hominid invents the first weapon, using a bone to kill prey. As the hominid tosses the bone in the air, Kubrick cuts to a 21st century spacecraft hovering over the **Earth**, skipping ahead millions of years in technological development. U.S. scientist Dr. Heywood Floyd (William Sylvester) travels to the **moon** to check out the discovery of a strange object on the **moon's** surface: a black monolith. As the sun's rays strike the stone, however, it emits a piercing, deafening sound that fills the investigators' headphones and stops them in their path. Cutting ahead 18 months, impassive **astronauts** David Bowman (Keir Dullea) and Frank Poole (Gary Lockwood) head toward Jupiter on the spaceship Discovery, their only company three hibernating **astronauts** and the vocal, man-made HAL 9000 computer running the entire ship. When the all-too-human HAL malfunctions, however, he tries to murder the **astronauts** to cover his error, forcing Bowman to defend himself the only way he can.

The Hitchhiker's Guide to the Galaxy

Earthman Arthur Dent is having a very bad day. His house is about to be bulldozed, he discovers that his best friend is an alien and to top things off, Planet **Earth** is about to be demolished to make way for a hyperspace bypass. Arthur's only chance for survival: hitch a ride on a passing spacecraft. For the novice space traveler, the greatest adventure in the universe begins when the world ends. Arthur sets out on a journey in which he finds that nothing is as it seems: he learns that a towel is just the most useful thing in the universe, finds the meaning of life, and discovers that everything he needs to know can be found in one book: The Hitchhiker's Guide to the Galaxy.

The Martian

During a manned mission to Mars, passing **moon**, **Astronaut** Mark Watney (Matt Damon) is presumed dead after a fierce storm and left behind by his crew. But Watney has survived and finds himself stranded and alone on the hostile planet. With only meager supplies, he must draw upon his ingenuity, wit and spirit to subsist and find a way to signal to the base that he is alive. Millions of miles away, NASA and a team of international scientists work tirelessly to bring "the Martian" home, while his crewmates concurrently plot a daring, if not impossible rescue mission.

Interstellar

In the near future around the American Midwest, Cooper an ex-science engineer and pilot, is tied to his farming land with his daughter Murph and son Tom. As devastating sandstorms ravage **earths** crops, the people of **Earth** realize their life here is coming to an end as food begins to run out. Eventually stumbling upon a NASA base near Cooper's home, he is asked to go on a daring mission with a few other scientists into a wormhole because of Cooper's scientific intellect and ability to pilot aircraft unlike the other crew members. In order to find a new home while **Earth** decays, Cooper must decide to either stay, or risk never seeing his children again in order to save the human race by finding another habitable planet.

Elysium

In the year 2154, two classes of people exist: the very wealthy, who live on a pristine man-made space station called Elysium, and the rest, who live on an overpopulated, ruined **Earth**. The people of **Earth** are desperate to escape the planet's **Earth** crime and poverty, and they critically need the state-of-the-art medical care available on Elysium - but some in Elysium will stop at nothing to enforce anti-immigration laws and preserve their citizens' luxurious lifestyle. The only man with the chance bring equality to these worlds is Max (Matt Damon), an ordinary guy in desperate need to get to Elysium. With his life hanging in the balance, he reluctantly takes on a dangerous mission - one that pits him against Elysium's **Astro**-Secretary Delacourt (Jodie Foster) and her hard-line forces - but if he succeeds, he could save not only his own life, but millions of people on **Earth** as well.

Ταινία	**earth**	**moon**	**astro**
2001: A Space Odyssey	1	2	3
The Hitchhiker’s Guide to the Galaxy	2	0	0
The Martian	0	1	1
Interstellar	3	0	0
Elysium	4	0	1

Στη συνέχεια, για να βρούμε το *Inverse Document Frequency (IDF*), υπολογίζουμε τη συχνότητα των ταινιών που περιέχουν κάθε όρο, σύμφωνα με τον παρακάτω τύπο.

$$IDF = \log \frac{|D|}{|d : t_i \in d|},$$

όπου *D* το σύνολο των ταινιών και επομένως το |*D*| είναι ίσο με 5. Ακολουθούν οι υπολογισμοί για το *IDF*.

Όρος	**IDF**
earth	0.09691
moon	0.39794
astro	0.22185

Τώρα μπορούμε να υπολογίσουμε το *TFIDF*, σύμφωνα με τον παρακάτω τύπο.

$$TFIDF = TF \cdot IDF .$$

Ακολουθούν τα αποτελέσματα των υπολογισμών για κάθε ταινία.

Ταινία	**TFIDF διάνυσμα (earth, moon, astro)**		
2001: A Space Odyssey	0.09691	0.79588	0.66555
The Hitchhiker’s Guide to the Galaxy	0.19382	0	0
The Martian	0	0.39794	0.22185
Interstellar	0.29073	0	0
Elysium	0.38764	0	0.22185

Άρα, έχουμε τα παρακάτω *TFIDF* διανύσματα για κάθε ταινία. Τώρα, υπολογίζουμε το *TFIDF* διάνυσμα του query (*earth, moon, astro*), όπως φαίνεται παρακάτω.

$$TFIDF = TF \cdot IDF = (1, 1, 1) \cdot (0.09691, 0.39794, 0.22185) = (0.09691, 0.39794, 0.22185)$$

Θα χρησιμοποιήσουμε τώρα το *Cosine Similarity* μεταξύ του *TFIDF* διανύσματος του query και των ταινιών για να βρούμε την κατάταξη των αποτελεσμάτων.

Ταινία	Cosine Similarity
2001: A Space Odyssey	0.97609
The Hitchhiker’s Guide to the Galaxy	0.20805
The Martian	0.97812
Interstellar	0.20805
Elysium	0.41715

Με βάση τους υπολογισμούς αυτούς, διαμορφώνεται η παρακάτω κατάταξη των αποτελεσμάτων.

1. *The Martian*
2. *2001: A Space Odyssey*
3. *Elysium*
4. *The Hitchhiker's Guide to the Galaxy / Interstellar*
5. *The Hitchhiker's Guide to the Galaxy / Interstellar*

Στην 4η και 5η θέση έχουμε ισοβαθμία της ταινίας *The Hitchhiker's Guide to the Galaxy με την ταινία Interstellar*, έχοντας και οι δύο το ίδιο *Cosine Similarity* με το query (*earth, moon, astro*).

Αναζήτηση – Precision/Recall

Από τα 34 αποτελέσματα που έβγαλε συνολικά η μηχανή αναζήτησης, τα 18 από αυτά σχετίζονται με την ταινία *2001: A Space Odyssey*. Συγκεκριμένα, τα σχετικά αποτελέσματα είναι τα παρακάτω.

- *1η σελίδα – μουσική, εμπόρευμα, άρθρο*

  12. Prom 36: A Space Odyssey
  13. 2001: A SPACE ODYSSEY (SHIPS) ΚΟΥΠΑ
  14. 2001: A Space Odyssey still leaves an indelible mark on …

- *2η σελίδα – άρθρο, κριτική, άρθρο, υπηρεσία streaming, κριτική, πλατφόρμα ψυχαγωγίας, βιβλίο*

  12. 2001: A Space Odyssey
  13. 2001: A Space Odyssey
  14. “2001: A Space Odyssey”: What It Means, and How It Was …
  15. 2001: A Space Odyssey (1968)
  16. 2001: A Space Odyssey movie review (1968)
  17. 2001: A Space Odyssey (film)
  18. Space Odyssey: Stanley Kubrick, Arthur C. Clarke, and the …

- *3η σελίδα – κριτική*

  12. 2001: A Space Odyssey

- *4η σελίδα – κριτική, υπηρεσία streaming, μουσική, βιβλίο, άρθρο, υπηρεσία streaming, άρθρο*

    12. 2001: A Space Odyssey (1968)
    13. 2001: A Space Odyssey
    14. Proms 2023 Prom 36: A Space Odyssey
    15. 2001: A Space Odyssey by Arthur C. Clarke
    16. 2001: A Space Odyssey still leaves an indelible mark on …
    17. 2001: A Space Odyssey
    18. 10 Movies Like '2001: A Space Odyssey'

Επομένως, έχουμε συνολικά 18 σωστά αποτελέσματα και 16 λάθος.

Επιπλέον, ξέρουμε ότι έχουμε άλλα 350 αποτελέσματα που σχετίζονται με την ταινία και δεν βρέθηκαν.

- $True\,Positive\,(TP)=18$
- $False\,Positive\,(FP)=16$
- $False\,Negative\,(FN)=350$

Για τα *Precision*, *Recall* και *F-Measure* έχουμε τους παρακάτω υπολογισμούς.

- $Precision=\frac{TP}{TP+FP}=0.5294117647\Rightarrow \boldsymbol{Precision\simeq 0.529}$
- $Recall=\frac{TP}{TP+FN}=0.0489130435\Rightarrow \boldsymbol{Recall\simeq 0.049}$
- $\text{F-Measure}=2\frac{Precision\cdot Recall}{Precision+Recall}=0.0895522388\Rightarrow \textbf{F-Measure}\simeq \textbf{0.09}$

Παρατηρούμε ότι το *Precision*, δηλαδή η ακρίβεια των αποτελεσμάτων είναι λίγο πάνω από το 50%, δηλαδή περίπου μόνο τα μισά αποτελέσματα από αυτά που επιστράφηκαν ήταν χρήσιμα και σχετικά με την αναζήτηση μας, την ταινία.

Το *Recall*, δηλαδή η πληρότητα, είναι αρκετά χαμηλό, μικρότερο του 5%, το οποίο σημαίνει ότι απουσιάζουν από τα αποτελέσματα που μας δόθηκαν, ένα μεγάλο ποσοστό αυτών που σχετίζονται με την ταινία.

Τέλος, το *F-Measure*, δηλαδή ο σταθμισμένος αρμονικός μέσος όρος των *Precision* και *Recall* είναι, επίσης, σχετικά χαμηλός, κοντά στο 10%, όπως αναμενόταν, αφού παρότι το *Precision* είναι σε υψηλότερα επίπεδα, το *Recall* έχει χαμηλή τιμή. Το *F-Measure* είναι μία καλή μετρική για τη γενικότερη εκτίμηση της συμπεριφοράς και απόδοσης μίας μηχανής αναζήτησης. Ένα υψηλότερο σκορ υποδεικνύει μία καλή ισορροπία μεταξύ ακρίβειας και πληρότητας, δηλαδή ότι η μηχανή αναζήτησης επιστρέφει κυρίως σωστά αποτελέσματα, χωρίς να λείπουν πολλά σχετικά αποτελέσματα.

Recommender Systems

Έχουμε τις παρακάτω βαθμολογίες των 5 ταινιών που χρησιμοποιήθηκαν και νωρίτερα, από 6 διαφορετικούς χρήστες, χρησιμοποιώντας την κλίμακα από 1 (καθόλου καλή) έως 10 (εξαιρετική).

Χρήστης	2001: A Space Odyssey	The Hitchhiker's Guide to the Galaxy	The Martian	Interstellar	Elysium
1	4	3		7	7
2	7	1		2	5
3	8	2	4	2	4
4	3	3	2	7	3
5		3	4	7	6
6	9	4	5	4	9

Θα υπολογίσουμε την ομοιότητα (similarity) μεταξύ των 6 χρηστών με δύο μεθόδους, την Ευκλείδεια Απόσταση και την Pearson Correlation.

Για την Ευκλείδεια Απόσταση, θα υπολογίσουμε, αρχικά, για κάθε ζεύγος χρηστών, το άθροισμα των τετραγώνων των διαφορών ανάμεσα στις βαθμολογίες, σε όσες ταινίες έχουν βαθμολογήσει και οι δύο. Έχοντας αυτό, το άθροισμα, εύκολα βρίσκουμε μετά την ομοιότητα. Οι τύποι που θα χρησιμοποιήσουμε ακολουθούν παρακάτω, μαζί με τα αποτελέσματα.

- $\text{sum} = \text{sum} + (rating(user\,1, movie) - rating(user\,2, movie))^2$
- $similarity = \frac{1}{1 + \sqrt{(\text{sum})}}$

	Χρήστης 1	Χρήστης 2	Χρήστης 3	Χρήστης 4	Χρήστης 5	Χρήστης 6
Χρήστης 1		0.13368	0.12283	0.19519	0.5	0.13803
Χρήστης 2	0.13368		0.36603	0.125	0.15439	0.14827
Χρήστης 3	0.12283	0.36603		0.11788	0.15439	0.14459
Χρήστης 4	0.19519	0.125	0.11788		0.21713	0.09488
Χρήστης 5	0.5	0.15439	0.15439	0.21713		0.18274
Χρήστης 6	0.13803	0.14827	0.14459	0.09488	0.18274	

Για την Pearson Correlation, θα υπολογίσουμε, αρχικά, τα αθροίσματα των βαθμολογίων και των τετραγώνων των βαθμολογίων για κάθε χρήστη σε ένα ζεύγος, σε όσες ταινίες έχουν βαθμολογήσει και

οι δύο. Στη συνέχεια, υπολογίζουμε το άθροισμα των γινομένων των βαθμολογίων τους, σε όσες ταινίες έχουν βαθμολογήσει και οι δύο. Τέλος, υπολογίζουμε τον αριθμητή και τον παρονομαστή, ώστε τέλος να υπολογίσουμε τη συσχέτιση και την ομοιότητα. Οι τύποι που θα χρησιμοποιήσουμε ακολουθούν παρακάτω, μαζί με τα αποτελέσματα.

- $\text{sum1} = \text{sum}\left(ratings\left(user\,1\right)\right)$
- $\text{sum2} = \text{sum}\left(ratings\left(user\,2\right)\right)$
- $\text{sum1sq} = \text{sum}\left(ratings\left(user\,1\right)^2\right)$
- $\text{sum2sq} = \text{sum}\left(ratings\left(user\,2\right)^2\right)$
- $psum = \text{sum}\left(rating\left(user\,1, movie\right) \cdot rating\left(user\,2, movie\right)\right)$
- $num = psum - \frac{\text{sum1} \cdot \text{sum2}}{n}$
- $den = \sqrt{\left(\text{sum1sq} - \frac{\text{sum1}^2}{n}\right) \cdot \left(\text{sum2sq} - \frac{\text{sum2}^2}{n}\right)}$
- $correlation = \frac{num}{den}$
- $similarity = \frac{1 + correlation}{2}$

	Χρήστης 1	**Χρήστης 2**	**Χρήστης 3**	**Χρήστης 4**	**Χρήστης 5**	**Χρήστης 6**
Χρήστης 1		0.5367	0.38567	0.78296	0.98536	0.57001
Χρήστης 2	0.5367		0.97076	0.28817	0.75	0.97173
Χρήστης 3	0.38567	0.97076		0.29057	0.5	0.8943
Χρήστης 4	0.78296	0.28817	0.29057		0.87052	0.31172
Χρήστης 5	0.98536	0.75	0.5	0.87052		0.65339
Χρήστης 6	0.57001	0.97173	0.8943	0.31172	0.65339	

Στη συνέχεια, θα χρησιμοποιήσουμε *K-Nearest Neighbors* με *k=2* και *weighted average* και με τις δύο μετρικές για να υπολογίσουμε πως περιμένουμε να αξιολογήσει την ταινία *The Martian* ο Χρήστης 2.

Αρχικά, για την Ευκλείδεια Απόσταση μετρική, παρατηρούμε ότι ο Χρήστης 2 έχει τη μεγαλύτερη ομοιότητα με τον Χρήστη 3 (0.36603) και με τον Χρήστη 5 (0.15439). Επομένως, έχουμε την πρόβλεψη που ακολουθεί.

	Χρήστης 1	**Χρήστης 2**	**Χρήστης 3**	**Χρήστης 4**	**Χρήστης 5**	**Χρήστης 6**
Χρήστης 2	0.13368		0.36603	0.125	0.15439	0.14827

$$prediction_{euclidean} = \frac{similarity(user\,2, user\,3)\cdot rating(user\,3) + similarity(user\,2, user\,5)\cdot rating(user\,5)}{similarity(user\,2, user\,3) + similarity(user\,2, user\,5)} \Rightarrow$$

$$prediction_{euclidean} = \frac{0.36603\cdot 4 + 0.15439\cdot 4}{0.36603 + 0.15439} \Rightarrow \boldsymbol{prediction_{euclidean} = 4}$$

Έπειτα, για την Pearson Correlation μετρική, παρατηρούμε ότι ο Χρήστης 2 έχει τη μεγαλύτερη ομοιότητα με τον Χρήστη 6 (0.97173) και με τον Χρήστη 3 (0.97076). Επομένως, έχουμε την πρόβλεψη που ακολουθεί.

	Χρήστης 1	**Χρήστης 2**	**Χρήστης 3**	**Χρήστης 4**	**Χρήστης 5**	**Χρήστης 6**
Χρήστης 2	0.5367		0.97076	0.28817	0.75	0.97173

$$prediction_{pearson} = \frac{similarity(user\,2, user\,6)\cdot rating(user\,6) + similarity(user\,2, user\,3)\cdot rating(user\,3)}{similarity(user\,2, user\,6) + similarity(user\,2, user\,3)} \Rightarrow$$

$$prediction_{pearson} = \frac{0.97173\cdot 5 + 0.97076\cdot 4}{0.97173 + 0.97076} = 4.50024968 \Rightarrow \boldsymbol{prediction_{pearson} \simeq 5}$$

Παρατηρούμε ότι στην περίπτωση της Ευκλείδειας Απόστασης, οι δύο πιο όμοιοι γείτονες του Χρήστη 2 είχαν και οι δύο την ίδια βαθμολογία, 4, στην ταινία *The Martian*, επομένως όπως ήταν αναμενόμενο, και ο Χρήστης 2 έχει πρόβλεψη βαθμολογίας ίση με 4. Στην περίπτωση της Pearson Correlation, ωστόσο, ο ένας γείτονας μένει ίδιος, ωστόσο πιο κοντινός ορίζεται ένας άλλος γείτονας, ο Χρήστης 6, ο οποίος έχει βαθμολογήσει την ταινία με 5. Επομένως, με βαθμολογία 5 από τον ομοιότερο γείτονα και βαθμολογία 4 από τον δεύτερο, η πρόβλεψη ανεβαίνει ελάχιστα πάνω από το 4.5 και επομένως πηγαίνει στο 5. Βλέπουμε, δηλαδή, ότι οι δύο διαφορετικές μετρικές δίνουν διαφορετικά αποτελέσματα, ακόμα και αν αυτή η διαφορά είναι οριακή.

Εάν, τώρα, χρησιμοποιήσουμε τις προτιμήσεις των χρηστών στις ταινίες για να προτείνουμε φίλους, τότε οι παρακάτω σχέσεις μοιάζουν αρκετά πιθανές, με βάση την κάθε μετρική.

Ευκλείδεια Απόσταση	**Pearson Correlation**
Χρήστες 1 και 5 – 0.5	Χρήστες 1 και 5 – 0.98536
Χρήστες 2 και 3 – 0.36603	Χρήστες 2 και 3 – 0.97076
	Χρήστες 2 και 6 – 0.97173
	Χρήστες 3 και 6 – 0.8943
Χρήστες 4 και 5 – 0.21713	Χρήστες 4 και 5 – 0.87052
Χρήστες 5 και 6 – 0.18274	Χρήστες 5 και 6 – 0.65339

Επομένως, τελικά, θα προτείναμε τις παραπάνω 4 προτάσεις φίλων, οι οποίες υποστηρίζονται ισχυρά και από τις δύο μετρικές που χρησιμοποιήσαμε.

- Χρήστης 1 και Χρήστης 5
- Χρήστης 2 και Χρήστης 3
- Χρήστης 4 και Χρήστης 5
- Χρήστης 5 και Χρήστης 6

Κώδικας Python και Αποτελέσματα

Αναζήτηση – TFIDF

tfidf.py > ...

```python
import numpy as np

from numpy.linalg import norm


# query, term frequencies and movies
query = ['earth', 'moon', 'astro']
tfs = {
    "2001: A Space Odyssey": [1, 2, 3],
    "The Hitchhiker's Guide to the Galaxy": [2, 0, 0],
    "The Martian": [0, 1, 1],
    "Interstellar": [3, 0, 0],
    "Elysium": [4, 0, 1]
}
movies = list(tfs.keys())


# calculate idf for a query term
def calculate_idf(term_freqs):
    return np.log10(len(movies) / np.count_nonzero(term_freqs))

idfs = {term: calculate_idf([tfs[movie][i] for movie in movies]) for i, term in enumerate(query)}


# calculate tfidf for a movie
def calculate_tfidf(movie_freqs):
    return [tf * idfs[term] for tf, term in zip(movie_freqs, query)]

tfidfs = {movie: calculate_tfidf(tfs[movie]) for movie in movies}
tfidfs = {"Query": calculate_tfidf([1, 1, 1]), **tfidfs}


# cosine similarity between two tfidf vectors
def calculate_cosine_similarity(movie_tfidf):
    query_tfidf = calculate_tfidf([1, 1, 1])
    return np.dot(query_tfidf, movie_tfidf)/(norm(query_tfidf) * norm(movie_tfidf))

cosine_similarities = {movie: calculate_cosine_similarity(tfidfs[movie]) for movie in movies}


# # function to print the results, with option for single results and list of results
def print_results(identifier, dictionary, is_list):
    print(identifier)
    for key, result in dictionary.items():
        if is_list:
            print(f"{key} - {[round(element, 5) for element in result]}")
        else:
            print(f"{key} - {round(result, 5)}")

identifiers = ["- TFs", "\n- IDFs", "\n- TFIDFs", "\n- Cosine Similarities"]
dictionaries = [tfs, idfs, tfidfs, cosine_similarities]
is_lists = [True, False, True, False]

for identifier, dictionary, is_list in zip(identifiers, dictionaries, is_lists):
    print_results(identifier, dictionary, is_list)
```

```
george@geeko:~/Desktop> python3 tfidf.py
 - TFs
 2001: A Space Odyssey - [1, 2, 3]
 The Hitchhiker's Guide to the Galaxy - [2, 0, 0]
 The Martian - [0, 1, 1]
 Interstellar - [3, 0, 0]
 Elysium - [4, 0, 1]

 - IDFs
 earth - 0.09691
 moon - 0.39794
 astro - 0.22185

 - TFIDFs
 Query - [0.09691, 0.39794, 0.22185]
 2001: A Space Odyssey - [0.09691, 0.79588, 0.66555]
 The Hitchhiker's Guide to the Galaxy - [0.19382, 0.0, 0.0]
 The Martian - [0.0, 0.39794, 0.22185]
 Interstellar - [0.29073, 0.0, 0.0]
 Elysium - [0.38764, 0.0, 0.22185]

 - Cosine Similarities
 2001: A Space Odyssey - 0.97609
 The Hitchhiker's Guide to the Galaxy - 0.20805
 The Martian - 0.97812
 Interstellar - 0.20805
 Elysium - 0.41715
```

Αναζήτηση – Precision/Recall

precision_recall.py > ...

```python
# function to print the results
def print_results(identifier, float_precision):
    print(identifier)
    print(f"- Precision: {round(precision, float_precision)}")
    print(f"- Recall: {round(recall, float_precision)}")
    print(f"- F-Measure: {round(fmeasure, float_precision)}")

# search results statistics
tp = 18
fp = 16
fn = 350

# calculate precision, recall and fmeasure
precision = tp / (tp + fp)
recall = tp / (tp + fn)
fmeasure = 2 * precision * recall / (precision + recall)

identifiers = ["Metrics", "\nRounded Metrics"]
float_precisions = [10, 3]

for identifier, float_precision in zip(identifiers, float_precisions):
    print_results(identifier, float_precision)
```

```
george@geeko:~/Desktop> python3 precision_recall.py
Metrics
- Precision: 0.5294117647
- Recall: 0.0489130435
- F-Measure: 0.0895522388

Rounded Metrics
- Precision: 0.529
- Recall: 0.049
- F-Measure: 0.09
```

Recommender Systems

similarity.py > ...

```python
import numpy as np

from itertools import combinations


# movie ratings and users
ratings = {
    "User 1": {"Space Odyssey": 4, "Hitchhiker": 3, "Interstellar": 7, "Elysium": 7},
    "User 2": {"Space Odyssey": 7, "Hitchhiker": 1, "Interstellar": 2, "Elysium": 5},
    "User 3": {"Space Odyssey": 8, "Hitchhiker": 2, "Martian": 4, "Interstellar": 2, "Elysium": 4},
    "User 4": {"Space Odyssey": 3, "Hitchhiker": 3, "Martian": 2, "Interstellar": 7, "Elysium": 3},
    "User 5": {"Hitchhiker": 3, "Martian": 4, "Interstellar": 7, "Elysium": 6},
    "User 6": {"Space Odyssey": 9, "Hitchhiker": 4, "Martian": 5, "Interstellar": 4, "Elysium": 9}
}
users = list(ratings.keys())


# calculate euclidean similarity between two users
def euclidean_similarity(user1, user2):
    common_movies = set(ratings[user1].keys()) & set(ratings[user2].keys())
    summary = sum((ratings[user1][movie] - ratings[user2][movie]) ** 2 for movie in common_movies)
    return 1 / (1 + np.sqrt(summary))

user_combinations = combinations(users, 2)
euclidean_similarities = {f"{user1} and {user2}": euclidean_similarity(user1, user2) for user1, user2 in user_combinations}


# calculate pearson correlation similarity between two users
def pearson_correlation_similarity(user1, user2):
    common_movies = set(ratings[user1].keys()) & set(ratings[user2].keys())
    n = len(common_movies)
    sum1 = sum([ratings[user1][movie] for movie in common_movies])
    sum2 = sum([ratings[user2][movie] for movie in common_movies])
    sum1sq = sum([ratings[user1][movie] ** 2 for movie in common_movies])
    sum2_sq = sum([ratings[user2][movie] ** 2 for movie in common_movies])
    psum = sum([ratings[user1][movie] * ratings[user2][movie] for movie in common_movies])

    num = psum - (sum1 * sum2) / n
    den = np.sqrt((sum1sq - sum1 ** 2 / n) * (sum2_sq - sum2 ** 2 / n))
    correlation = num / den

    return (1 + correlation) / 2

user_combinations = combinations(users, 2)
pearson_correlation_similarities = {f"{user1} and {user2}": pearson_correlation_similarity(user1, user2) for user1, user2 in user_combinations}


# function to print the results
def print_results(identifier, dictionary):
    print(identifier)
    for key, result in dictionary.items():
        print(f"{key} - {round(result, 5)}")

identifiers = ["- Euclidean Similarities", "\n- Pearson Correlation Similarities"]
dictionaries = [euclidean_similarities, pearson_correlation_similarities]

for identifier, dictionary in zip(identifiers, dictionaries):
    print_results(identifier, dictionary)
```

```
george@geeko:~/Desktop> python3 similarity.py
 - Euclidean Similarities
 User 1 and User 2 - 0.13368
 User 1 and User 3 - 0.12283
 User 1 and User 4 - 0.19519
 User 1 and User 5 - 0.5
 User 1 and User 6 - 0.13803
 User 2 and User 3 - 0.36603
 User 2 and User 4 - 0.125
 User 2 and User 5 - 0.15439
 User 2 and User 6 - 0.14827
 User 3 and User 4 - 0.11788
 User 3 and User 5 - 0.15439
 User 3 and User 6 - 0.14459
 User 4 and User 5 - 0.21713
 User 4 and User 6 - 0.09488
 User 5 and User 6 - 0.18274

 - Pearson Correlation Similarities
 User 1 and User 2 - 0.5367
 User 1 and User 3 - 0.38567
 User 1 and User 4 - 0.78296
 User 1 and User 5 - 0.98536
 User 1 and User 6 - 0.57001
 User 2 and User 3 - 0.97076
 User 2 and User 4 - 0.28817
 User 2 and User 5 - 0.75
 User 2 and User 6 - 0.97173
 User 3 and User 4 - 0.29057
 User 3 and User 5 - 0.5
 User 3 and User 6 - 0.8943
 User 4 and User 5 - 0.87052
 User 4 and User 6 - 0.31172
 User 5 and User 6 - 0.65339
```