COMP 474/6741 Intelligent Systems (Winter 2024)

Worksheet #4: Recommender Systems

Task 1. Let's take some movies that have been #tagged (or categorized) as follows:

	Action	Comedy	Sci-Fi	Horror	Drama	Romance	*length*
Movie 1	4	8	6	3	0	0	
Movie 2	0	5	0	8	5	0	
Movie 3	1	4	0	3	0	10	

So, each movie becomes a 6-dimensional vector of tags t_i, e.g., $\overrightarrow{\text{Movie}_1} = \langle 4, 8, 6, 3, 0, 0 \rangle$. Compute the *length* of each movie vector, which is defined as $\|\vec{m}\| = \sqrt{t_1^2 + \ldots + t_n^2}$ (rounded to two significant digits).

Task 2. Now you can *normalize* the vectors, by dividing the raw count of each tag t_i by the length $\frac{t_i}{||\vec{m}||}$:

	Action	Comedy	Sci-Fi	Horror	Drama	Romance
Movie 1						
Movie 2						
Movie 3						

Use 4 significant digits for this table (protip: the *length* of each movie vector must now be 1).

Task 3. We can now compute how *similar* the movies are, by computing their *cosine similarity*. Since the vectors are normalized, this is simply their dot product: $\text{sim}(\vec{m}, \vec{n}) = \cos(\vec{m}, \vec{n}) = \vec{m} \cdot \vec{n} = \sum_i m_i \cdot n_i$:

	Movie 1	Movie 2	Movie 3
Movie 1	1		
Movie 2		1	
Movie 3			1

This is the information we need for an *item-to-item recommendation engine:* Now we can answer the question, which movie is interesting to (buy, watch) for a customer who (bought, watched) Movie 1? ....

Task 4. Now we want to *personalize* the recommendations. We collected the following profiles about the movies watched (bought) by our users in the past:

	Action	Comedy	Sci-Fi	Horror	Drama	Romance	*length*
Jane	1	2	1	1	1	0	
Joe	0	1	0	1	0	1	

Compute the length of each *user vector* and normalize it like before:

	Action	Comedy	Sci-Fi	Horror	Drama	Romance
Jane						
Joe						

Task 5. Now we can answer the question which movie a user is interested in. Compute the cosine similarities between the *user vectors* and the *movie vectors:*

	Movie 1	Movie 2	Movie 3
Jane			
Joe			

Task 6. Consider the results from three different recommender systems below: Here, X1–X5 are the items (movies, photos, songs, ...) that the systems should have recommended as relevant for a specific user. The remaining 495 instances are not relevant for the user. A checkmark indicates that a system recommended this item to the user (the first *Target* column is the ground truth):

	Target	*system 1*	*system 2*	*system 3*
	X1 ✓	X1 ×	X1 ✓	X1 ✓
	X2 ✓	X2 ×	X2 ×	X2 ✓
	X3 ✓	X3 ×	X3 ✓	X3 ✓
	X4 ✓	X4 ×	X4 ✓	X4 ✓
	X5 ✓	X5 ×	X5 ×	X5 ✓
	X6 ×	X6 ×	X6 ×	X6 ✓
	X7 ×	X7 ×	X7 ×	X7 ✓
	... ×	... ×	... ×	... ×
	... ×	... ×	... ×	... ×
	X500 ×	X500 ×	X500 ×	X500 ×

Evaluate the performance of the three systems using the measures *Precision* and *Recall*:

	Precision	Recall
system 1		
system 2		
system 3		

$$\text{precision} = \frac{\#\text{correct system recommendations}}{\#\text{all system recommendations}}$$

$$\text{recall} = \frac{\#\text{correct system recommendations}}{\#\text{all correct recommendations}}$$

Task 7. Now we're looking at *ranked* results. Based on the output below, compute precision@k $= \frac{1}{k} \cdot \sum_{c=1}^{k} \text{rel}(c)$ for the three recommender systems (for $k = 1, 2, 3$):

	rel(k)			precision@k			
	1	2	3	1	2	3	AP@3
system 1	1	0	0				
system 2	0	1	0				
system 3	0	0	1				

That is, here each system got exactly one recommendation right, but in a different position.

Task 8. Moving on to the *average precision*, AP $@N = \frac{1}{m} \sum_{k=1}^{N}$ precision@k $\cdot\text{rel}(k)$. Compute the AP@3 and add it to the table above. Here, assume $m = 3$ (i.e., there could have been 3 correct recommendations in the top-3). Note the difference in the AP@3 for the three systems!

Task 9. Create a *content vector* for the movie description $m_1 =$ *"A comedy with zombies."* Start by filling in the tf values below. Then compute idf $= \log_{10} \frac{N}{\text{df}}$ (assume $N =$ 10,000,000) and tf-idf $= (1 + \log \text{tf}_{t,d}) \times \text{idf}$. Finally, compute the normalized vector $\vec{q}$ as before (in Tasks 1&2) from the tf-idf vector and its length:

	m_1				
token	tf	df	idf	tf-idf	q_i
action		50,000			
comedy		10,000			
zombies		100,000			
romantic		10,000			

You can now use these vectors for cosine similarity calculations to find recommendations as before, but this time based on the *content* of an item (like a movie description).