Content-based filtering

Key concept

- Building a vector of attribute
- Or a vector of keyword preferences

How to

- 사용자가 직접 제시 (or 수정)
- Infer profile from user actions
- Infer profile from explicit user ratings
- Merge user actions and explicit user ratings

Preferences?

- Keywords that users have (or not have) opinion on
- Count
- Or more sophisticated other methods

How to use preferences

- Vector of keyword preferences
  - Simply add up likes and dislikes
  - 더 relevant한 것에 weight 주기

- TFIDF(Term Frequency Inverse Document Frequency)
  - How frequent/important is this description in the current product?
  - Ex) blue + long sleeve + shirt + 목사 collar

Case-based recommendation

- Products described in a set of relevant attributes
- Preference -> recommendations -> further preferences
- 일시적
- Ex) 어떤 상품? -> 무슨 용도? -> 사용자의 전문성? -> price range? ->…other feature attribute

Knowledge based recommendation

- 어떤 item을 선호했는지에 대한 정보에서 출발
- 그 item을 기준으로
  - Cheaper, nicer, more traditional…
- Ex) Entrée

Case-based recommendation

- 일시적
  - 현재 보는 것과 비슷한 상품

- Easier to explain to the user

Content-based recommendations

- 장기적인 선호도 반영
- Harder to explain to users
- Not so clear on all domains
  - Ex) paintings- more blue? more oil?
  - Need to come up with attributes that seem to correlate with preference
- Harder to find 보완재 than 대체제
- Work without large set of user data

Content(product attribute)-based

- Short term : database of cases(co-purchased, co-browsed), navigate

- Longer term : build profile (on content preferences)

TFIDF

Term Frequency Inverse Document Frequency

The search problem

- Search “civil war”

- How often does the term occur in the document?
  - ‘civil’- 100, ‘war’- 100
  - ‘civil war’- 50
  - In terms of importance: ‘civil’ > ‘war’

TFIDF weighting

- Term Frequency * Inverse Document Frequency = weight

- Term Frequency
  - Number of occurrences of a term in the document
  - Or tags

- Inverse document frequency
  - How few documents contain this term
  - 일반적 공식
    - Log (#documents / #documents with term)
  - 얼마나 rare한가? Ex) ‘the’ vs ‘civil’

But TFIDF fails when…

- Core term/concept not used much in document
  - Legal contracts – 이름 1회 노출 후 '갑' 등으로 대체

- 사람들이 검색을 '못'할 때

How does TFIDF apply to CBF?

- Use TFIDF to create a profile of a document/object

- Can be combined with ratings, etc

Variants on TF(Term Frequency)

- 0 or 1 Boolean frequencies (above certain threshold)
- Logarithmic frequencies
- Normalized frequency (ex) divide by document length

More things to consider

- Phrases and n-grams
  - ‘computer science’ != ‘computer’ and ‘science’
- 문서에서의 중요도 (or 문서 제공자의 신뢰도)
  - 제목, 부제, …
- General document authority
- Implied content
  - 키워드가 한번도 포함되지 않는다면? Ex) 뉴욕 양키스 스케줄
  - Co-purchasing as an alternative to direct atrrtibute

Keyword vector

Keyword vector

- Keywords define a content space
  - Each keyword = dimension
  - Each item has a position in that space; vector
  - Each user has taste profile in that space; vector
  - User preference & item – 벡터가 얼마나 가까이 있는가
  - May limit/collapse space – stem and stop

Representing item through a keyword vector

- 0 or 1?
- Occurrence count?
- TFIDF (most common)
- Normalize vector

Representing item through a keyword vector

- 0 or 1
  - Actor – ‘톰 크루즈‘
  - Descriptive – ‘스릴 있는‘
- IDF (IDF 값이 낮을 수록-많이 나온 단어일수록-중요한 단어가 아니다)
  - Actor – ‘톰 크루즈 ' vs ‘엑스트라1’
  - Descriptive – ‘재판장 씬 ' vs ‘로맨스’/‘스릴러’

Limitation of the vector space model

- No difference between liking and importance
  - 바비큐 소스를 좋아하지만, 있어도 그만 없어도 그만이라면?

User profiles

- Add together the item vectors?
  - Normalize?
    - All vectors should be the same weight?
  - Weigh the vector?
    - Ratings for weight … etc

How to factor in rating

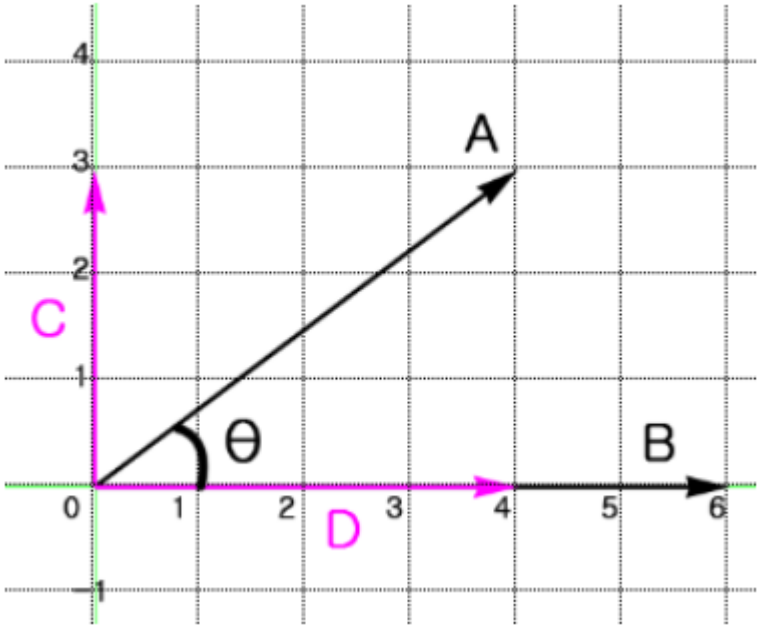
- Simple unary – without weights
- Simple binary – add positives and subtract negatives
- Unary with threshold – 3.5 and above as equal
- Weight, but positive only
- Weight, and include negative also – negative weight

How to update profiles (new ratings)

- Recompute each time – 비효율적
- Weight new/old similarly
  - For changed ratings, subtract old
- Decay old profile and mix in new
  - 0.95*old + 0.05*new
  - More weight to new

Computing predictions

- Prediction = cosine of the angle between two vectors(profile, item)
- Normalized vector들의 내적
- Ranges between -1 and 1 (0 and 1 if all positive)
  - 1에 가까울수록 match
- …more on later lectures



A·B (A와 B의 내적)

= B 크기 × D의 크기

= | B | × | A | × cosθ

= 6 × 4 = 24

(벡터 D의 크기는 A벡터의 크기에 cosθ를 곱한 것이다. | D | = | A | × cosθ)

여기서 A의 크기가 아니라 A를 분해한 벡터 D의 크기를 곱해주는 이유는

B 벡터에 실제적으로 영향을 주는 벡터가 D이기 때문이다.

(벡터 C는 벡터 B의 방향으로 어떠한 영향도 주지 못하기 때문에, 내적 계산에서 무시한다.)

Keyword 사용 장점

- Entirely content based
- Understandable profile
- Easy computation
- Flexibility – can integrate with other approaches
  - Case-based
  - Query-based

단점

- Factors?
- Weights?
- Simplified – can’t handle interdependencies
  - 톰 크루즈의 액션영화와 르네 젤웨거의 로맨틱 코미디,
    not 르네 젤웨거의 액션영화와 톰 크루즈의 로맨틱 코미디
  - Comedies with violence and historical documentaries, but not historical comedies or violent documentaries.
  - Not unless infinite combinations of attributes
- 이미 좋아한다고 한 것만 추천해준다

Entrée style recommenders

Knowledge-based recommenders - The FindMe systems

I would like to eat at a restaurant that has:

Cuisine | Price

Style | Atmosphere | Occasion

I would like to eat at a restaurant just like:

Chinois on Main | Los Angeles

New Query | Submit

Entrée style recommenders

- Similar reassessment – finding similarity between objects
- Criteria arranged hierarchically
  - Cuisine -> price -> quality of the experience
- Similarity can be asymmetric
  - 비슷하지만 더 비싼 음식점 – not so similar

Recommending restaurants

- Manual knowledge engineering => similarity matrix
  - multiple cuisines in a restaurant
  - same cuisine may not be the same food depending on the location(city)

- 사용한 유저 정보
  - Critique-based RS <= 사용 안함
    - Cheaper recommendations – options get small very fast
    - Always need to give recommendations even if not so similar
    - This encouraged users to explore – keep hitting cheaper
  - '비슷함' <- 사람마다 주관적인 기준
    - Is the person rich? Is the person feeling rich that day?
      - Price may be in the lower hierarchy of the similarity matrix
    - Start with default matric (- then maybe let the users choose their own similarity metric)

Type retrieval

- ‘Just like this Chinese restaurant but Italian’
  - Lots of queries like this
  - Make up pretend restaurant, find the closest one
- Let people explore the space of possibilities

Case-based reasoning

Memory-based reasoning

Case-based reasoning

- 과거의 문제와 해결방법을 새로운 문제 해결에 사용한다
- Problems and solutions as cases
  - Problem – solution (- outcome)
- Retrieve a case similar to the new problem
- Most useful in domains where there’s no strong domain models

Case-based recommenders

- Single-shot case-based reasoning system(/case-based RS)
  - Query represented in a case-like representation
  - Suggests a set of similar cases
- Conversational recommendations
  - Conversational interactions between user and the case-based system
  - User provides feedback at each recommendation step
    - Recommends Chinese restaurant -> ‘more cheaper’ -> 반영된 추천
    - Critiques on multiple features
- Relevance and diversity

Opinion mining

- Amazon
  - Catalog features - info about the product
  - Reviews - user generated info about the product
    - Opinion mining to extract out new features
    - Ex) 맥북 에어 – 디자인, 화면 크기, 배터리
    - Sentiment anlaysis
    - Ex) 맥북 디자인 – 긍정, 맥북 가격 - 부정

Dialog-based recommenders

Critiquing based

Critique-based recommender system

- Critiques
  - User initiated
    - I like this laptop, but I want something cheaper
  - Provided by the system
    - This item might be also interesting for you
    - This laptop satisfied your requirement, and gives better battery life, but more expensive

Difference between critique vs rating

- Rating based recommender systems
  - Preferences represented as a score (as a whole)

- Critique based recommender systems
  - Preferences expressed over the features of a product
  - Single shot high stakes decision – 꼼꼼히 조건 봄
  - User preference change overtime
    - 아이폰 16기가 –(1년 후)-> 아이폰 128기가
    - Preferences change based on the available options shown to us
    - Given today's market, today's tech, today's context…etc
  - Also catches stable and long term preferences
  - Compound critiquing – several features at once

asos WOMEN MEN Search for items, brands and inspiration

FREE CLICK & COLLECT NEXT DAY DELIVERY OVER £100*
Select a click & collect service at checkout and then enter code COLLECT100 to redeem.

STUDENTS: 10% OFF 24/7
+ More good stuff

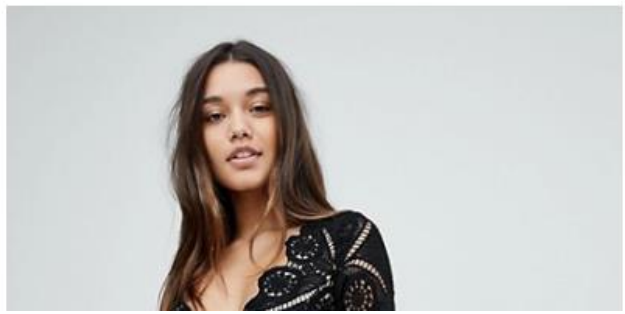
Home › Search results for party dress

Your search results for:

"Party Dress"

Sort | Style | Range | Colour | Size | Brand | Price Range

4,213 styles found

User preference model

- MAUT(multi attribute utility theory)
  - Linear combination weighted by the importance of each attribute
  - Direction of utility
    - Direction matters: more memory - better, more expensive – worse
  - Pro: Easy to compute – linear algorithm
    - Easy to compute and rank scores
  - Con: Assumes independence of different attributes
    - Screen size and weight
    - 역에서 집까지 거리 (500미터)

Critiques change

- 100만원 미만의 상품을 원한다 -> 상품을 보다 보니까
  ->200만원 이상이더라도 더 좋은 성능을 원한다

- Compound critiquing
  - Suggest items that satisfy some of the preferences and make compromises
  - Educate users what options are available
  - Giving people a tool to navigate and helping them learn what the space is.

Search, recommendation, and target audiences

Search vs Recommendation

- Search
  - User triggers the task
  - User knows what they are looking for

- Recommendation
  - Recommender gives suggestions
  - User does not know that item exists

Data for content-based recommendation

- User provided content
- Item content
- Tags
- Reviews
  - Great for understanding “why”

Special demographic populations of users

- Children
  - Harder to get user profile
  - Need to consider readability
- English as second language
  - Can get user profile
  - Need to consider comprehension level of English
  - Topic interests

Beyond TFIDF

The Netflix prize

- Matrix factorization

- “Recommending New Movies: Even a Few Ratings are More Valuable Than Metadata”
  - Content-based approach 는 쓰레기?

Content-based approach 재조명

- Social media nowadays – full of content
  - Both quality and quantity

- Explanation
  - 유저보다,
  - 추천 시스템에 투자하는 회사에게 더 필요

- Comprehension of the info conveyed by text or content > Method
  - To semantic representation

Semantic representation

- Explicit representation
  - Explicitly mapping features describing the item or the whole item with concepts contained in external knowledge sources
  - Wikipedia
  - DBpedia etc

- Implicit representation
  - Distributed hypothesis: “the meaning is it’s use”
  - Meaning of words determined by analyzing their usage (cooccurrence)
  - Word embedding, word2vec, …

Semantics-aware RS advantages

- Richer representations
  - Overcome limited content analysis
  - Better explanations
  - Foster unexpectedness

추가 내용

Damped means

- Problem: low confidence w/ few ratings
- Solution: assume that, without evidence, everything is average
- Ratings are evidence of non-averageness
- *k* controls strength of evidence required

$$\frac{\sum_u r_{ui} + k\mu}{n + k}$$

Scoring news stories

- Hacker News

$$\frac{(U - D - 1)^{\alpha}}{(t_{\text{now}} - t_{\text{post}})^{\gamma}} \times P$$

- Net upvotes, polynomially decayed by age
- Old items scored mostly by vote
- Multiplied by item penalty terms
  - incorporate community goals into score

Reddit algorithm (c. 2010)

$$\log_{10} \max(1, |U - D|) + \frac{\text{sign}(U - D) t_{\text{post}}}{45000}$$

- Log term applied to votes
  - decrease marginal value of later votes
- Time is seconds since Reddit epoch
- Buries items with negative votes
- Time vs. vote impact independent of age
- Scores news items, not comments