Content-Based Recommendation (Part A)

Topic 5A

Content-Based Recommendation

- Collaborative recommendation approaches are based on user ratings, without utilizing specific item descriptions.
- In contrast, content-based recommendations utilize item characteristics ("content") and user profiles to find recommendable items.
- A user profile describes the (past) interests of a user in terms of preferred item characteristics.
- Item descriptions may come from external sources as well as internal processing of item content (characteristics and attributes).

Content Representation and Content Similarity

- When an explicit list of features for each item and user preferences are maintained, the recommendation task is simply consists of matching item characteristics and a user profile of preferences.

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical

Item Profile and User's Preference Profile

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical

Title	Genre	Author	Type	Price	Keywords
...	Fiction, Suspense	Brunonia Barry, Ken Follett	Paperback	25.65	detective, murder, New York

Content Representation and Content Similarity

- User profile can be constructed in various ways
  - Direct response to questions
  - Rating of a set of items along different dimensions
  - Automatic derivation of a set of keywords
    - Dice coefficient measures the similarity between books *bi* and *bj* as:

$$\frac{2 \times |keywords(b_i) \cap keywords(b_j)|}{|keywords(b_i)| + |keywords(b_j)|}$$

Content-Based Recommendation Approaches

- The standard approach in content-based recommendation is to use a list of relevant keywords that appear within the document, generated automatically from the document content itself or from a free-text description thereof.
- Boolean vector approach
  - Problems and limitations
- Term frequency/inverse document frequency (TF-IDF) approach

TF-IDF

- Assume that *N* is the total number of documents that can be recommended to users and that keyword k_i appears in n_i of them. Also, assume that $f_{i,j}$ is the number of times keyword k_i appears in the document d_j.
  - $TF_{i,j} = f_{i,j} / max_z f_{z,j}$
    - Where $f_{z,j}$ of all keywords k_z that appear in the document d_j.
  - $IDF_i = \log(N/n_i)$
  - $W_{i,j} = TF_{i,j} \times IDF_i$

Improving the Vector Space Model

- Stop words and stemming
- Size cutoffs
- Phrases
- Limitations
  - Context
  - Feature extraction
  - Quality
  - Overspecialization

NLP – Overall Process

NLP process

Input Text

Input text: Article, news, blogs, SNS ..

Sentence Separation: Spliting text corpus into sentences

Tokenization: : Spliting sentences into tokens

Stemming and Lemmatization: Transferring to basic forms

Stop Word Removal: Removing unmeaningful, unnecessary tokens

POS tagging: : Identifying Part-of-speech

Term Selection: : Identifying relevant tokens only

A Set of representative tokens

NLP – Overall Process

NLP process	Working Example
Input Text — Input text: Article, news, blogs, SNS ..	
Sentence Separation: Spliting text corpus into sentences	Figure 1 shows the overall architecture of our framework…
Tokenization: : Spliting sentences into tokens	Figure/ 1 / shows / the / overall / architecture / of /our/ framework…/
Stemming and Lemmatization: Transferring to basic forms	Fig/ 1 / show / the / overall / architecture / of /our/ framework…/
Stop Word Removal: Removing unmeaningful, unnecessary tokens	Fig/ show / overall / architecture / framework…/
POS tagging: : Identifying Part-of-speech	Fig (NN)/ show(VB) / overall (ADV) / architecture (NN) / framework (NN)…/
Term Selection: : Identifying relevant tokens only	Fig (NN)/ architecture (NN) / framework (NN)…/
A Set of representative tokens	

Stemming and Lemmatization

- Lemmatization: Reduce inflectional/variant forms to base form
  - E.g.,

    am, are, is -> be

    car, cars, car's, *cars' -> car*
  - *the boy's cars are different colors -> the boy car be different color*
  - Lemmatization implies doing “proper” reduction to dictionary headword form
- Stemming: Reduce terms to their “roots” before indexing
  - “Stemming” suggest crude affix chopping
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.
- Note: Stemming would hurt search precision over recall:
  - Stemming returns ‘oper’ from ‘*operate operating operates operation operative operatives operational’*
  - since *operate* in its various forms is a common verb, we would expect to lose considerable precision on queries (i.e., operational and research, operating and system, operative and dentistry)

Useful NLP Tools - Python

- NLTK (Natural Language Toolkit): http://www.nltk.org/
  - used for such tasks as tokenization, lemmatization, stemming, parsing, POS tagging, etc. This library has tools for almost all NLP tasks
- Spacy: https://spacy.io/
  - the main competitor of the NLTK. These two libraries can be used for the same tasks.
- Scikit-learn: https://scikit-learn.org/
  - provides a large library for machine learning. The tools for text preprocessing are also included.
- Gensim: https://radimrehurek.com/gensim/
  - the package for topic and vector space modeling, document similarity.
- KoNLPy: https://konlpy-ko.readthedocs.io/ko/v0.4.3/
  - a Python package for Korean natural language processing

	⊕ PROS	⊖ CONS
Natural Language ToolKit	+ The most well-known and full NLP library + Many third-party extensions + Plenty of approaches to each NLP task + Fast sentence tokenization + Supports the largest number of languages compared to other libraries	– Complicated to learn and use – Quite slow – In sentence tokenization, NLTK only splits text by sentences, without analyzing the semantic structure – Processes strings which is not very typical for object-oriented language Python – Doesn't provide neural network models – No integrated word vectors
spaCy	+ The fastest NLP framework + Easy to learn and use because it has one single highly optimized tool for each task + Processes objects; more object-oriented, comparing to other libs + Uses neural networks for training some models + Provides built-in word vectors + Active support and development	– Lacks flexibility, comparing to NLTK – Sentence tokenization is slower than in NLTK – Doesn't support many languages. There are models only for 7 languages and "multi-language" models
scikit learn NLP toolkit	+ Has functions which help to use the bag-of-words method of creating features for the text classification problems + Provides a wide variety of algorithms to build machine learning models + Has good documentation and intuitive classes' methods	– For more sophisticated preprocessing things (for example, pos-tagging), you should use some other NLP library and only after it you can use models from scikit-learn – Doesn't use neural networks for text preprocessing
gensim	+ Works with large datasets and processes data streams + Provides tf-idf vectorization, word2vec, document2vec, latent semantic analysis, latent Dirichlet allocation + Supports deep learning	– Designed primarily for unsupervised text modeling – Doesn't have enough tools to provide full NLP pipeline, so should be used with some other library (Spacy or NLTK)

From https://www.kdnuggets.com/2018/07/comparison-top-6-python-nlp-libraries.html

Text Preprocessing in Python

- Helpful Links
  - **Text Preprocessing in Python: Steps, Tools, and Examples**
    - https://medium.com/@datamonsters/text-preprocessing-in-python-steps-tools-and-examples-bf025f872908
  - **How to Clean Text for Machine Learning with Python**
    - https://machinelearningmastery.com/clean-text-machine-learning-python/
  - **How to Implement TF-IDF**
    - https://medium.freecodecamp.org/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3

Useful NLP Tools - Java

- Stanford NLP: https://nlp.stanford.edu/software
  - Well-known, widely used for English NLP tasks
  - Offers many features, but unreliable for Korean preprocessing
- Berkeley NLP: http://nlp.cs.berkeley.edu/software.shtml
  - Offers a smaller set of functionalities, but its parser works well
- Apache OpenNLP: http://opennlp.apache.org/
  - Supports common NLP tasks
  - Source code is offered as open source
- Komoran: https://www.shineware.co.kr/products/komoran/
  - Reliable performance for Korean
  - Can be easily used with open source search library such as Lucene, Indri
  - Delay for initial loading

Stanford Core NLP Suite

- *Input text: "Annie has a little lamb. She is very cute."*

Function	Output
ssplit	"Annie has a little lamb." "She is very cute."
token	Annie, has, a, little, lamb, She, is, very, cute
pos	Annie/NNP, has/VBZ, a/DT, little/JJ, lamb/NN, She/PRP, is/VBZ, very/RB, cute/JJ
lemma	Annie, have, a, little, lamb, she, be, very, cute
ner	She/PERSON, has/O, a/O, little/O, lamb/O, She/O, is/O, very/O, cute/O
sentiment	"Annie has a little lamb"/Negative "She is very cute"/Positive

Similarity-Based Retrieval

- Nearest neighbors
  - To recommend documents, we need
    - Some history of like/dislike statements made by the user about previous items
    - Similarity measure – cosine similarity
  - K-nearest-neighbor method (KNN)
    - Varies the neighborhood size k
    - More weight on keywords associated with recent ratings
    - Long-term vs. short-term interests

Relevance Feedback – Rocchio Method

- User provides feedback on the relevance of documents retrieved so as to improve retrieval results in the next round.
- The Rocchio algorithm splits the already rated documents into two groups, D^+ and D^-, and calculate a prototype (or average) vector for these categories. The current query Qi is then repeatedly refined to Q_{i+1} as follows:

$$Q_{i+1} = \alpha * Q_i + \beta \left(\frac{1}{|D^+|} \sum_{d^+ \in D^+} d^+ \right) - \gamma \left(\frac{1}{|D^-|} \sum_{d^- \in D^-} d^- \right) \qquad (3.5)$$

Rocchio Method Formula

- In the formula, the value ofα describes how strongly the last query should be weighted while β and γ are control parameters that are used to set the relative importance of positive and negative examples.
- For instance, if $\alpha= 1.5$, $\beta=2$,and $\gamma=1$, we don't want the negative examples to have as strong influence as the positive examples.

Rocchio Method: Example

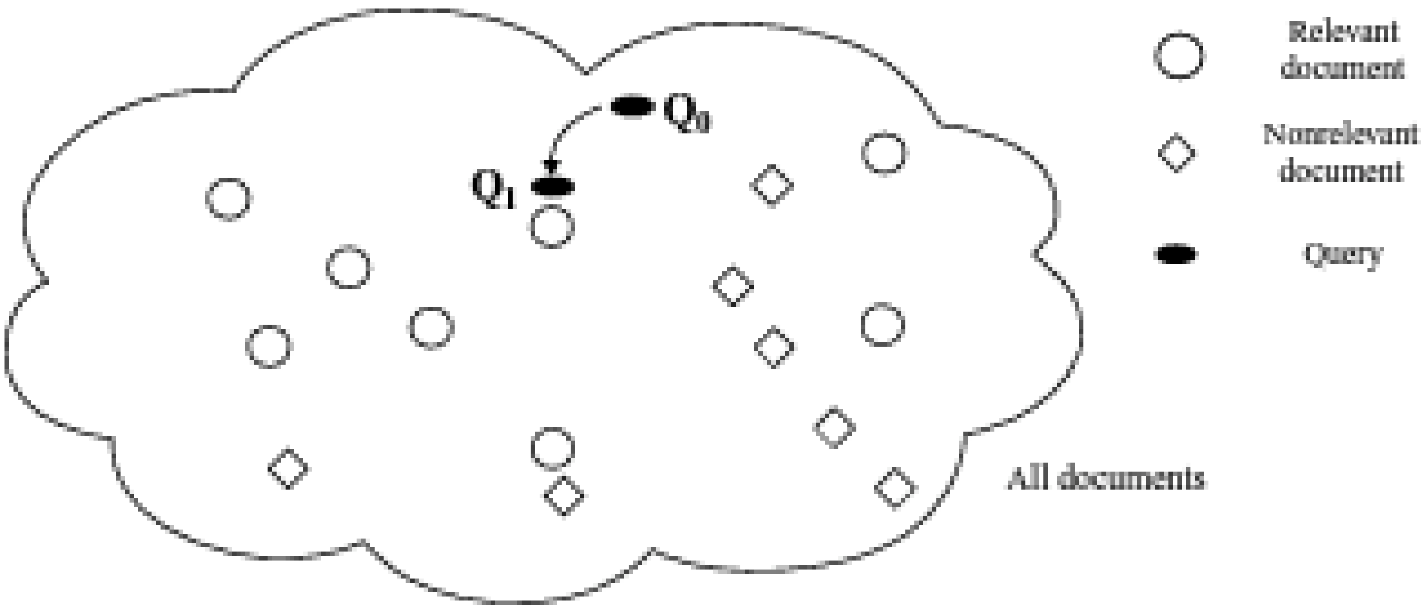
- Let's assume that we have identified 118 terms from the set of documents classified into the categories of medicine, energy, and environment.
  - The weight of each term represents the importance of the respective term for the category
  - What is the weight of term 'nuclear' in the category 'medicine'?
- $POS_{medicine}$ contains the documents Doc1-Doc4, and $NEG_{medicine}$ contains the documents Doc5-Doc10
  - $|D^+| = 4$ and $|D^-| = 6$.

Rocchio Method: Example

- Weights of term 'nuclear' in documents in $POS_{medicine}$
  - w_nuclear_doc1 = 0.5
  - w_nuclear_doc2 = 0
  - w_nuclear_doc3 = 0
  - w_nuclear_doc4 = 0.5
- Weight in documents in $NEG_{medicine}$
  - w_nuclear_doc6 = 0.5
- Weight of 'nuclear' in the category 'medicine':
  - $2 * (0.5 + 0.5)/4 - 1 * 0.5/6 = 0.5 - 0.08 = 0.42$

Relevance Feedback Effect

Figure 3.2.



After feedback, the original query is moved toward the cluster of the relevant documents.