

Content-based recommendations and analysis of text data

Basic Python analysis of text data

- text data (documents, Web pages, short texts (e.g., social media))
- Bag of Words
- TF-IDF model



Recommender systems - approaches

- **Content-based recommendation**: items (content) are described by a feature set and we use only the target user's history
- **Collaborative filtering**: exclusively the ratings are utilised
 - user-based: prediction is based on similar users' preferences
 - item-based: prediction is based on the ratings of similar items
- **Hybrid** methods: the combinations of the two main approaches



Content-based recommendations

- Main idea: recommend items to customer C similar to previous items rated highly by C
- Movie recommendations
 - recommend movies with the same actor(s), director, genre, ...
- Websites, blogs, news
 - recommend other sites with “similar” content



Content-based recommendation systems

- Use exclusively the history of the target user
- Recommendations are based on information on the **content** of items rather than on other users' opinions.
- Uses a machine learning algorithm to induce a profile of the users preferences from examples based on a featural description of content.
- Items are described by features
e.g.: actors, director, category, words in the description



Advantages of Content-Based Approach

- No need for data on other users.
 - No cold-start or sparsity problems.
- Can provide explanations of recommended items by listing content-features that caused an item to be recommended.
- **No cold start:** As opposed to collaborative filtering, new items can be suggested before being rated by a substantial number of users.



Disadvantages of Content-Based Method

- Requires content that can be encoded as meaningful features.
- Users' tastes must be represented as a learnable function of these content features.
- Unable to exploit quality judgments of other users.
 - Unless these are somehow included in the content features.



Item profile


- For each item, create an **item profile**
- Profile is a set of features
 - **Movies:** author, title, actor, director,...
 - **Images, videos:** metadata and tags
 - **People:** Set of friends
- Convenient to think of the item profile as a vector
 - One entry per feature (e.g., each actor, director,...
 - Vector might be boolean or real-valued

Most CB-recommendation techniques were applied to recommending text documents.

- Like web pages or newsgroup messages for example.

Content of items can also be represented as text documents.

- With textual descriptions of their basic characteristics.
- Structured: Each item is described by the same set of attributes



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
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism

- Unstructured: free-text description.



Examples

- product description,
- recipe instructions,
- movie plot
- songs

Pandora music recommendation service

How It Works:

- Base its recommendation on data from Music Genome Project
- Assigns 400 attributes for each song, done by musicians, takes half an hour per song
- Use this method to find songs which is similar to user's favorite songs



Benefits:

- Accurate method, don't need lots of users information, needs little to get started

Drawback:

- Doesn't scale very well and often feels that Pandora's library is somewhat limited

Picture from: pandora.com



Similarity: Keywords

- general similarity approach based on keywords
- two sets of keywords A; B (description of two items or
description of item and user)
- how to measure similarity of A and B?



Similarity: Keywords Example

video 1: machine learning, education, visualization, math

video 2: late night, comedy, politics

video 3: football, goal, funny, Messi, trick, fail



Similarity: Keywords

sets of keywords A , B

- Dice coefficient: $\frac{2 \cdot |A \cap B|}{|A| + |B|}$
- Jaccard coefficient: $\frac{|A \cap B|}{|A \cup B|}$

many other coefficients available, see e.g. “A Survey of Binary Similarity and Distance Metrics”



Similarity: Text Descriptions

- Examples: product description, recipe instructions, movie plot
- basic approach: bag-of-words representation (words + counts of occurrences)



What is a Bag-of-Words?

- A bag-of-words model, or BoW for short, is a way of extracting features from text for use in modeling, such as with machine learning algorithms.
- It is called a “bag” of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.
- A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:
 - A vocabulary of known words.
 - A measure of the presence of known words.



Example: movie review

- Review 1: This movie is very scary and long
- Review 2: This movie is not scary and is slow
- Review 3: This movie is spooky and good

We will first build a vocabulary from all the unique words in the above three reviews. The vocabulary consists of these 11 words: 'This', 'movie', 'is', 'very', 'scary', 'and', 'long', 'not', 'slow', 'spooky', 'good'.

- We can now take each of these words and mark their occurrence in the three movie reviews above with 1s and 0s. This will give us 3 vectors for 3 reviews:

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

Vector of Review 1: [1 1 1 1 1 1 1 0 0 0 0]

Vector of Review 2: [1 1 2 0 0 1 1 0 1 0 0]

Vector of Review 3: [1 1 1 0 0 0 1 0 0 1 1]

And that's the core idea behind a Bag of Words (BoW) model.



Term Frequency --Inverse Document Frequency (TF-IDF)

Term frequency-inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus

- **Term Frequency:** is a scoring of the frequency of the word in the current document.
- **Inverse Document Frequency:** is a scoring of how rare the word is across documents.



Term Frequency (TF)

Let's first understand Term Frequency (TF). It is a measure of how frequently a term, t , appears in a document, d :

$$tf_{t,d} = \frac{n_{t,d}}{\text{Number of terms in the document}}$$

Here, in the numerator, n is the number of times the term “ t ” appears in the document “ d ”. Thus, each document and term would have its own TF value.




We will again use the same vocabulary we had built in the Bag-of-Words model to show how to calculate the TF for Review #2:

Review 2: This movie is not scary and is slow

Here,

- Vocabulary: 'This', 'movie', 'is', 'very', 'scary', 'and', 'long', 'not', 'slow', 'spooky', 'good'
- Number of words in Review 2 = 8
- TF for the word 'this' = (number of times 'this' appears in review 2)/(number of terms in review 2) = 1/8



Term	Review 1	Review 2	Review 3	TF (Review 1)	TF (Review 2)	TF (Review 3)
This	1	1	1	1/7	1/8	1/6
movie	1	1	1	1/7	1/8	1/6
is	1	2	1	1/7	1/4	1/6
very	1	0	0	1/7	0	0
scary	1	1	0	1/7	1/8	0
and	1	1	1	1/7	1/8	1/6
long	1	0	0	1/7	0	0
not	0	1	0	0	1/8	0
slow	0	1	0	0	1/8	0
spooky	0	0	1	0	0	1/6
good	0	0	1	0	0	1/6



Inverse Document Frequency (IDF)

IDF is a measure of how important a term is. We need the IDF value because computing just the TF alone is not sufficient to understand the importance of words:

$$idf_t = \log \frac{\text{number of documents}}{\text{number of documents with term 't'}}$$




We can calculate the IDF values for all the words in Review 2:

$$\text{IDF('this')} = \log(\text{number of documents} / \text{number of documents containing the word 'this'}) = \log(3/3) = \log(1) = 0$$


Similarly,

- $\text{IDF('movie')} = \log(3/3) = 0$
- $\text{IDF('is')} = \log(3/3) = 0$
- $\text{IDF('not')} = \log(3/1) = \log(3) = 0.48$
- $\text{IDF('scary')} = \log(3/2) = 0.18$
- $\text{IDF('and')} = \log(3/3) = 0$
- $\text{IDF('slow')} = \log(3/1) = 0.48$




Term	Review 1	Review 2	Review 3	IDF
This	1	1	1	0.00
movie	1	1	1	0.00
is	1	2	1	0.00
very	1	0	0	0.48
scary	1	1	0	0.18
and	1	1	1	0.00
long	1	0	0	0.48
not	0	1	0	0.48
slow	0	1	0	0.48
spooky	0	0	1	0.48
good	0	0	1	0.48

Hence, we see that words like “is”, “this”, “and”, etc., are reduced to 0 and have little importance; while words like “scary”, “long”, “good”, etc. are words with more importance and thus have a higher value.



We can now compute the TF-IDF score for each word in the corpus. Words with a higher score are more important, and those with a lower score are less important:

$$(tf_idf)_{t,d} = tf_{t,d} * idf_t$$




We can now calculate the TF-IDF score for every word in Review 2:

$$\text{TF-IDF}(\text{'this'}, \text{Review 2}) = \text{TF}(\text{'this'}, \text{Review 2}) * \text{IDF}(\text{'this'}) = 1/8 * 0 = 0$$

Similarly,

- $\text{TF-IDF}(\text{'movie'}, \text{Review 2}) = 1/8 * 0 = 0$
- $\text{TF-IDF}(\text{'is'}, \text{Review 2}) = 1/4 * 0 = 0$
- $\text{TF-IDF}(\text{'not'}, \text{Review 2}) = 1/8 * 0.48 = 0.06$
- $\text{TF-IDF}(\text{'scary'}, \text{Review 2}) = 1/8 * 0.18 = 0.023$
- $\text{TF-IDF}(\text{'and'}, \text{Review 2}) = 1/8 * 0 = 0$
- $\text{TF-IDF}(\text{'slow'}, \text{Review 2}) = 1/8 * 0.48 = 0.06$




Term	Review 1	Review 2	Review 3	IDF	TF-IDF (Review 1)	TF-IDF (Review 2)	TF-IDF (Review 3)
This	1	1	1	0.00	0.000	0.000	0.000
movie	1	1	1	0.00	0.000	0.000	0.000
is	1	2	1	0.00	0.000	0.000	0.000
very	1	0	0	0.48	0.068	0.000	0.000
scary	1	1	0	0.18	0.025	0.022	0.000
and	1	1	1	0.00	0.000	0.000	0.000
long	1	0	0	0.48	0.068	0.000	0.000
not	0	1	0	0.48	0.000	0.060	0.000
slow	0	1	0	0.48	0.000	0.060	0.000
spooky	0	0	1	0.48	0.000	0.000	0.080
good	0	0	1	0.48	0.000	0.000	0.080

We have now obtained the TF-IDF scores for our vocabulary. TF-IDF also gives larger values for less frequent words and is high when both IDF and TF values are high i.e the word is rare in all the documents combined but frequent in a single document.



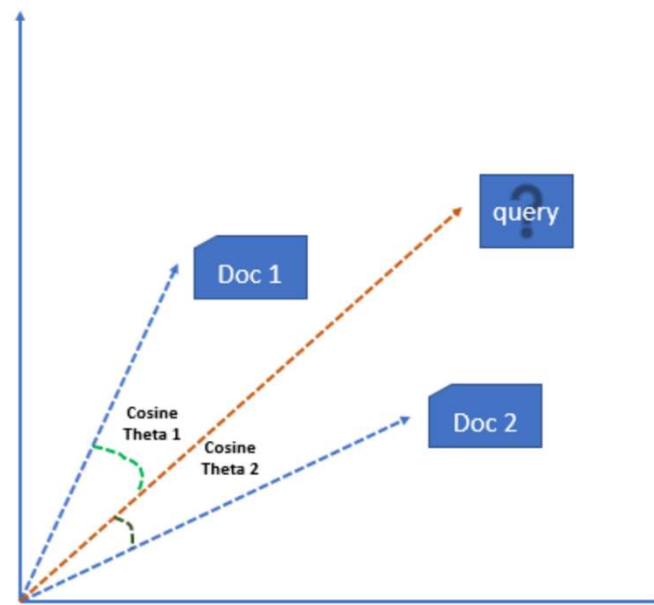
More on TF-IDF

- The intuition behind TF-IDF is based on two characteristics of text documents. First, the more times a term appears in a document, the more relevant it is to the topic of the document. Second, the more times a term occurs in all documents in the collection, the more poorly it discriminates between documents.
- Google has already been using $TF*IDF$ (or $TF-IDF$, $TFIDF$, $TF.IDF$) as a ranking factor for your content for a long time, as the search engine seems to focus more on *term frequency* rather than on *counting keywords*.

- 
- Bag of Words just creates a set of vectors containing the count of word occurrences in the document (reviews), while the TF-IDF model contains information on the more important words and the less important ones as well.
 - Bag of Words vectors are easy to interpret. However, TF-IDF usually performs better in machine learning models.

Measuring Similarity

- Each item is stored as a vector of its attributes (which are also vectors) in an n-dimensional space, and the angles between the vectors are calculated to determine the similarity between the vectors.



Documents represented as vectors



Vector-space model

- A representation that is often used for text documents is the vector space model.
- In the vector space model a document D is represented as an m -dimensional vector, where each dimension corresponds to a distinct term and m is the total number of terms used in the collection of documents. The entry in the vector corresponds to the term's tf-idf score.

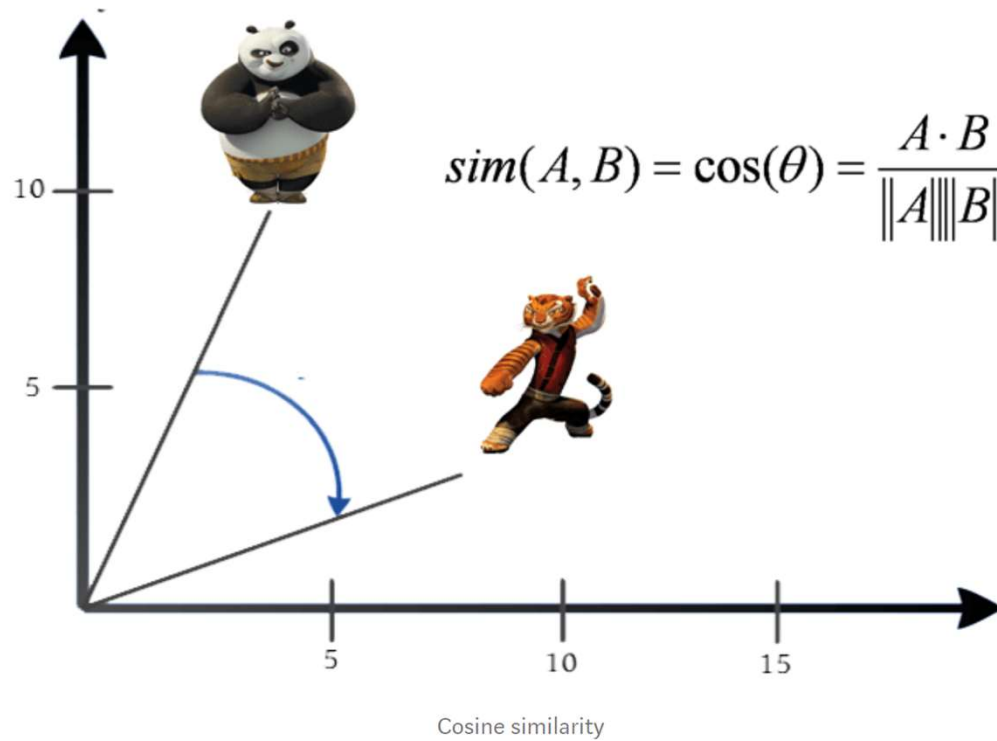


Cosine measure of similarity

- The method of calculating the user's likes / dislikes / measures is calculated by taking the cosine of the angle between the user profile vector and the document vector; or in our case, the angle between two document vectors.
- The ultimate reason behind using cosine is that the value of cosine will increase as the angle between vectors decreases, which signifies more similarity.
- The vectors are length-normalized, after which they become vectors of length 1.


Calculating Cosine Similarity

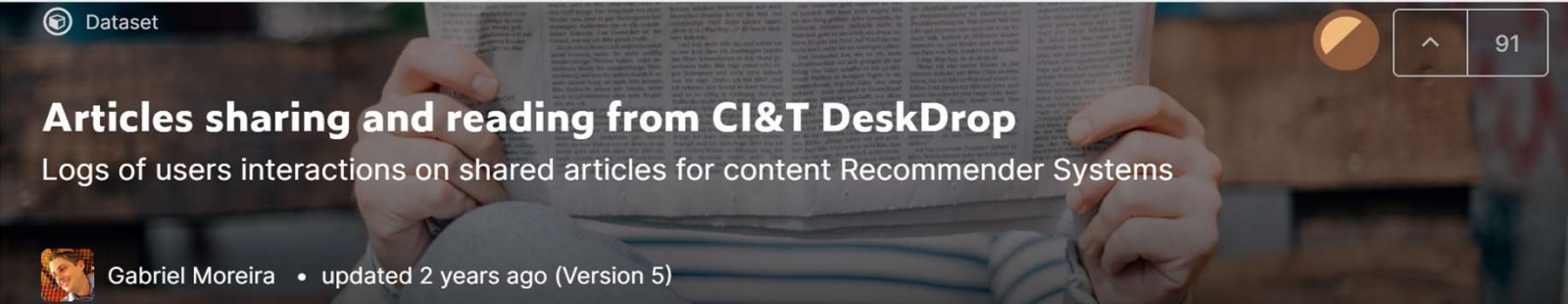
Cosine Similarity



kaggle.com/gspmoreira/articles-sharing-reading-from-cit-deskdrop#shared_articles.csv





 Dataset






Articles sharing and reading from CI&T DeskDrop

Logs of users interactions on shared articles for content Recommender Systems

 Gabriel Moreira • updated 2 years ago (Version 5)

[Data](#) [Tasks](#) [Kernels \(9\)](#) [Discussion](#) [Activity](#) [Metadata](#) [Download \(29 MB\)](#) [New Notebook](#) 

 Usability 8.2  License Database: Open Database, Contents: Database Contents  Tags computing, computer science, internet, web sites, human-computer interaction

This rich and rare dataset contains a real sample of 12 months logs (Mar. 2016 - Feb. 2017) from CI&T's Internal Communication platform (DeskDrop). It contains about 73k logged users interactions on more than 3k public articles shared in the platform.



This dataset features some distinctive characteristics:

- Item attributes: Articles' original URL, title, and content plain text are available in two languages (English and Portuguese).
- Contextual information: Context of the users visits, like date/time, client (mobile native app / browser) and geolocation.
- Logged users: All users are required to login in the platform, providing a long-term tracking of users preferences (not depending on cookies in devices).
- Rich implicit feedback: Different interaction types were logged, making it possible to infer the user's level of interest in the articles (eg. comments > likes > views).
- Multi-platform: Users interactions were tracked in different platforms (web browsers and mobile native apps)

Text analytics libraries for Python

- **NLTK: Natural Language toolkit**
 - Collection of NLP tools
- **TextBlob**
 - Built on top of NLTK, especially useful for beginners
- **spaCy**
 - Fast NLP implementation
- **Gensim**
 - For topic modeling and similarity detection
- **Pattern**
 - Web mining module for Python and more
- **Pyparsing**
 - For parsing text