



THE
DEVELOPER'S
CONFERENCE

2016

Track: Data Science

Gabriel Moreira - @gspmoreira
Gilmar Souza - @gilmarsouza

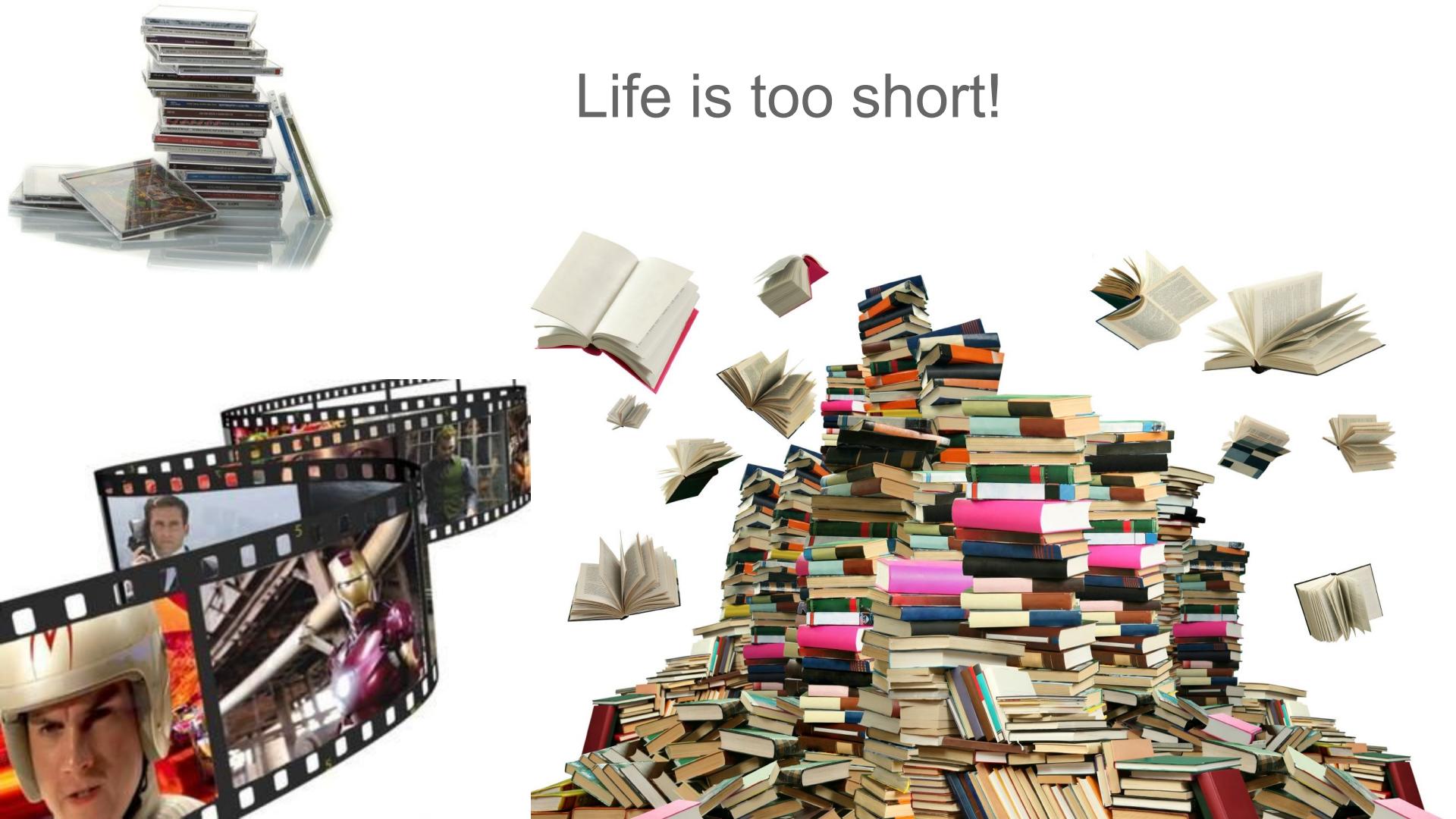
Discovering Users' Topics of Interest in Recommender Systems

Agenda

- Recommender Systems
- Topic Modeling
- Case: Smart Canvas - Corporative Collaboration
- Wrap up

Recommender Systems

An introduction



Life is too short!

Social recommendations



Recommendations by interaction



"A lot of times, people don't know what they want until you show it to them."

Steve Jobs

"We are leaving the Information Age and entering the Recommendation Age.".

Cris Anderson, "The long tail"

Recommendations are responsible for...



38% of sales



38% of top news
visualization

2/3 movie rentals

What else may I recommend?

products
tags
professionals
courses
musics movies
jobs books
papers girlfriends
investments restaurants
 videos
dressing



What can a Recommender Systems do?

Recommendation

Given a user, produce an ordered list matching the user needs

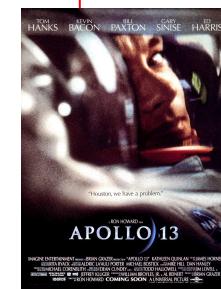
Prediction

Given an item, what is its relevance for each user?

How it works

Content-Based Filtering

Similar content (e.g. actor)



Likes

Recommends



Content-Based Filtering

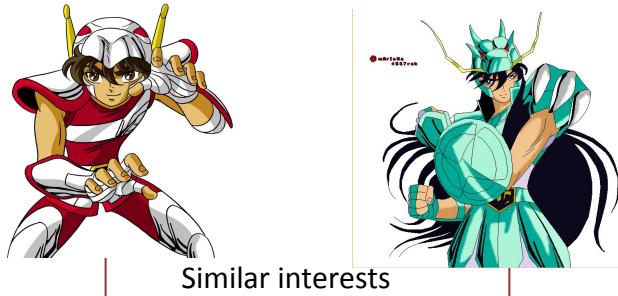
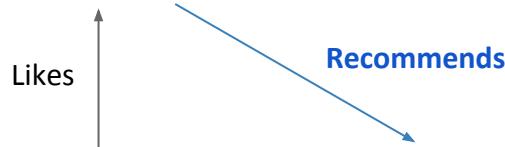
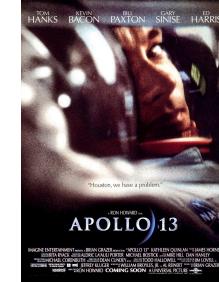
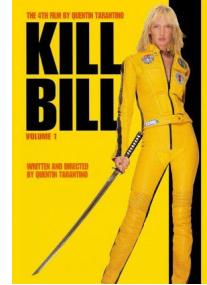
Advantages

- Does not depend upon other users
- May recommend new and unpopular items
- Recommendations can be easily explained

Drawbacks

- Overspecialization
- May not recommend to new users
 - May be difficult to extract attributes from audio, movies or images

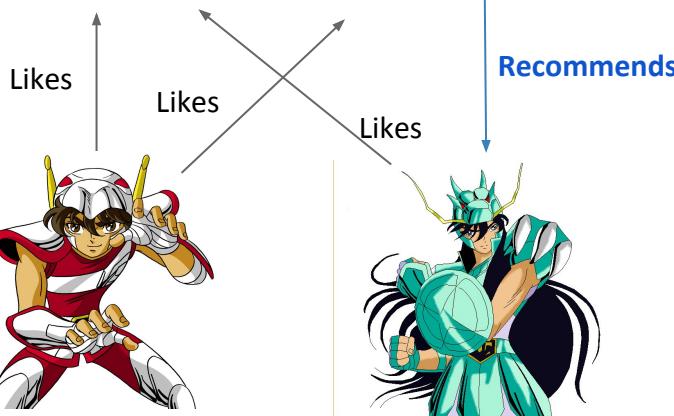
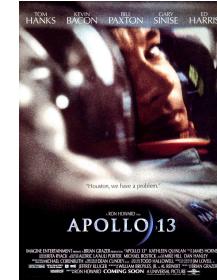
User-Based Collaborative Filtering



Similar interests

Item-Based Collaborative Filtering

Who likes A also likes B



Collaborative Filtering

Advantages

- Works to any item kind (ignore attributes)

Drawbacks

- Cannot recommend items not already rated/consumed
- Usually recommends more popular items
- Needs a minimum amount of users to match similar users (*cold start*)

Hybrid Recommender Systems

Some approaches

Composite

Iterates by a chain of algorithm, aggregating recommendations.

Weighted

Each algorithm has as a weight and the final recommendations are defined by weighted averages.



UBCF Example (Java / [Mahout](#))

```
User,Item,Rating1,  
15,4.0  
1,16,5.0  
1,17,1.0  
1,18,5.0  
2,10,1.0  
2,11,2.0  
2,15,5.0  
2,16,4.5  
2,17,1.0  
2,18,5.0  
3,11,2.5
```

input.csv

```
// Loads user-item ratings  
1 DataModel model = new FileDataModel(new File("input.csv"));  
// Defines a similarity metric to compare users (Person's correlation coefficient)  
2 UserSimilarity similarity = new PearsonCorrelationSimilarity(model);  
// Threshold the minimum similarity to consider two users similar  
3 UserNeighborhood neighborhood = new ThresholdUserNeighborhood(0.1, similarity,  
model);  
// Create a User-Based Collaborative Filtering recommender  
4 UserBasedRecommender recommender = new GenericUserBasedRecommender  
5 (model, neighborhood, similarity);  
// Return the top 3 recommendations for userId=2  
6 List recommendations = recommender.recommend(2, 3);
```

Frameworks - Recommender Systems



Java



Python / Scala



Python

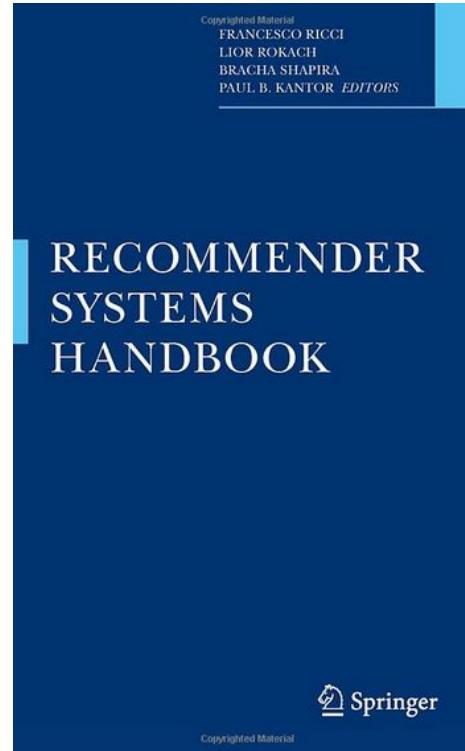
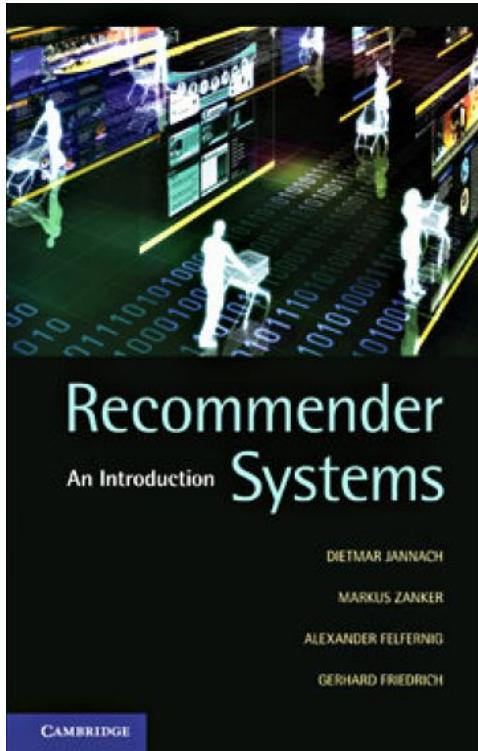


Java



.NET

Books

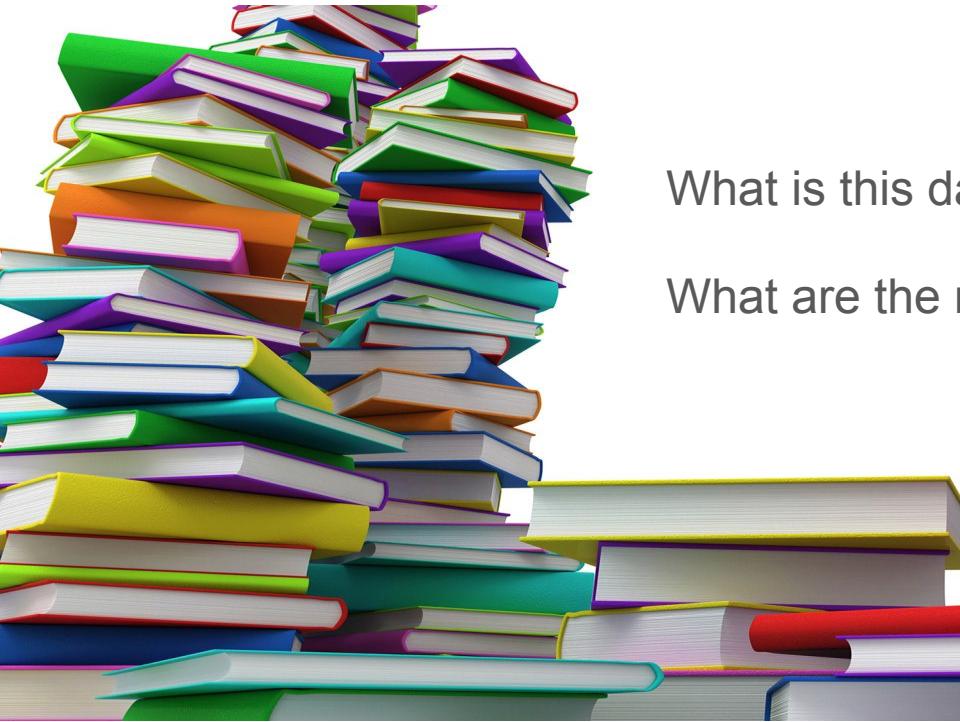


Topic Modeling

An introduction

Topic Modeling

A simple way to analyze topics of large text collections (corpus).



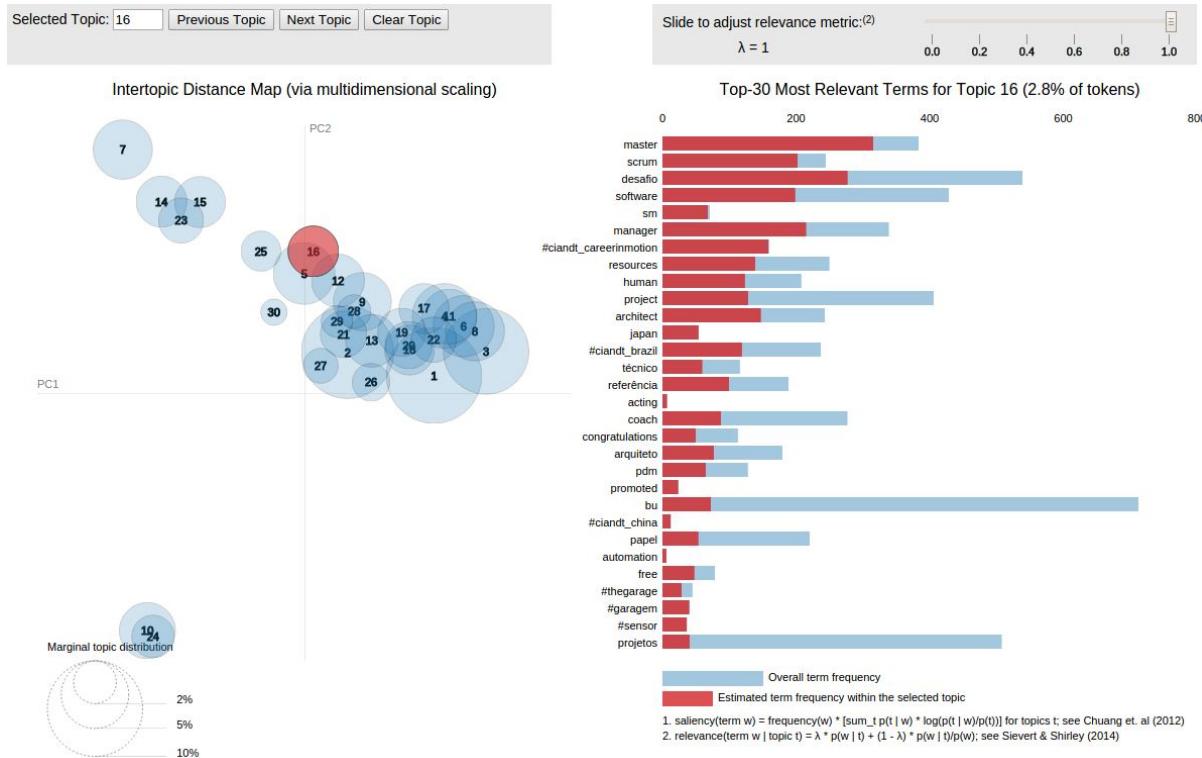
What is this data about?

What are the main topics?



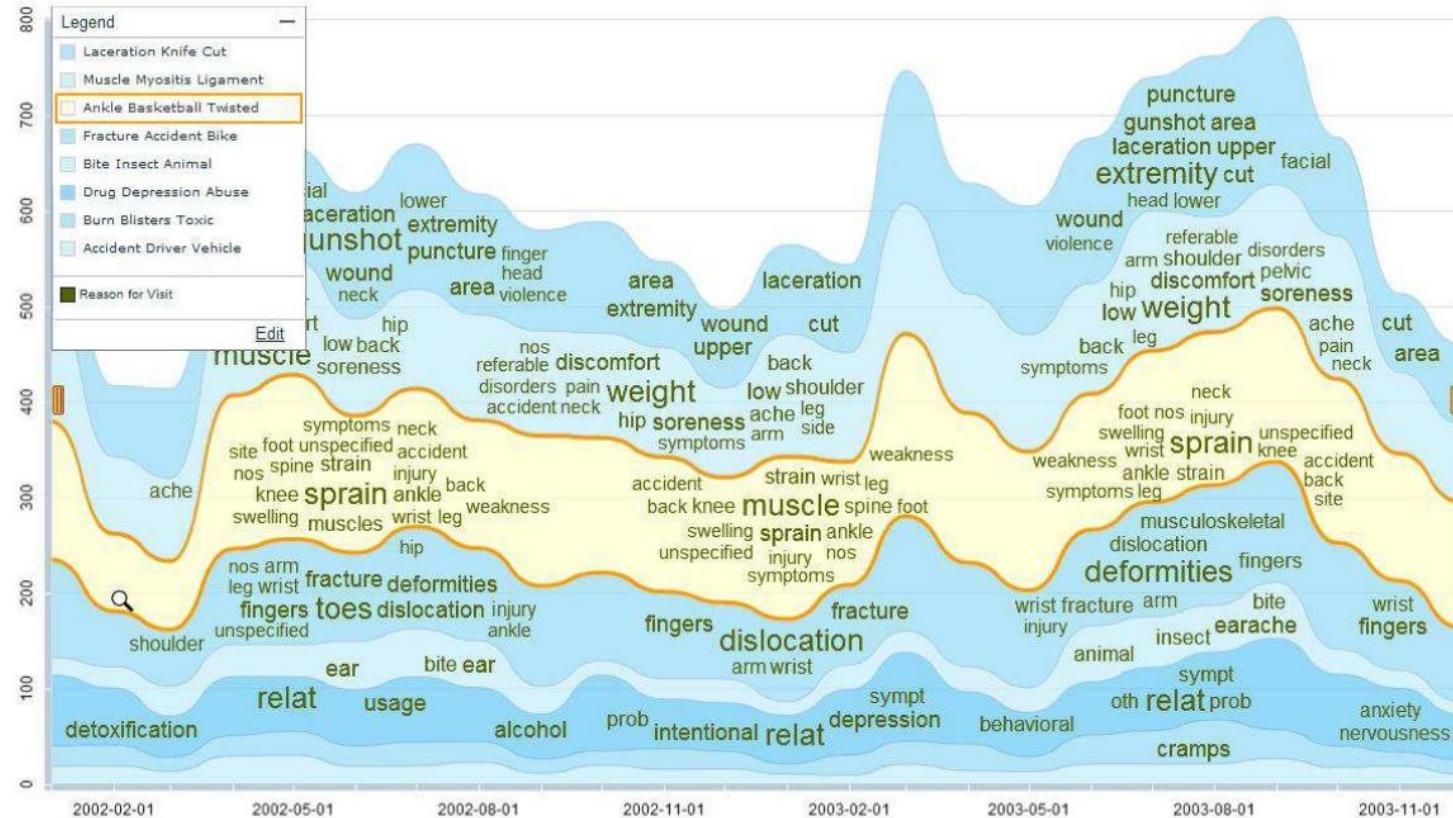
Topic

A Cluster of words that generally occur together and are related.



Example 2D topics visualization with [pyLDAviz](#) (topics are the circles in the left, main terms of selected topic are shown in the right)

Topics evolution



Visualization of topics evolution during across time

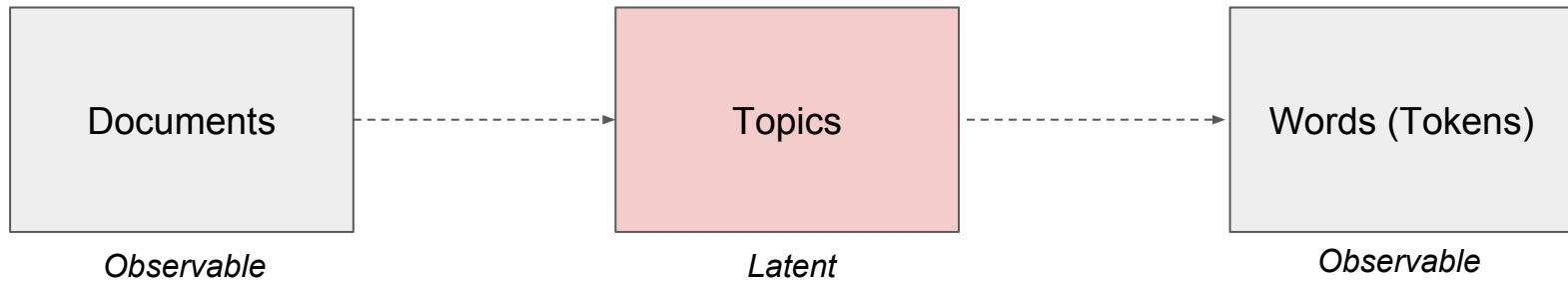
Topic Modeling applications

- Detect trends in the market and customer challenges of a industry from media and social networks.
- Marketing SEO: Optimization of search keywords to improve page ranking in searchers
- Identify users preferences to recommend relevant content, products, jobs, ...



Topic Modeling

Unsupervised learning technique, which does not require too much effort on pre-processing (usually just Text Vectorization). In general, the only parameter is the number of topics (k).



- Documents are composed by many topics
- Topics are composed by many words (tokens)

Topics in a document

Topics	
gene	0.84
dna	0.82
genetic	0.81
...	
life 0.82	
evolve	0.81
organism	0.81
...	
brain 0.84	
neuron	0.82
nerve	0.81
...	
data 0.82	
number	0.82
computer	0.81
...	

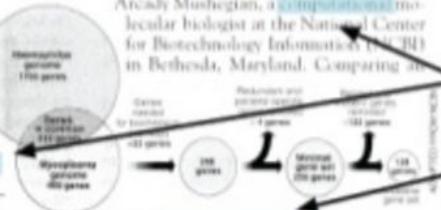
Documents

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions

"are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Svante Andersson, a genetics professor at the Royal Institute of Technology in Stockholm who arrived at the 800-gene mark. But coming up with a consensus answer may be more than just a numbers game. "Surprisingly, more and more genomes are completely sequenced and analyzed," says Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing the



Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

Topic proportions and assignments

Topics example from NYT

music
band
songs
rock
album
jazz
pop
song
singer
night

book
life
novel
story
books
man
stories
love
children
family

art
museum
show
exhibition
artist
artists
paintings
painting
century
works

game
Knicks
nets
points
team
season
play
games
night
coach

show
film
television
movie
series
says
life
man
character
know

theater
play
production
show
stage
street
broadway
director
musical
directed

clinton
bush
campaign
gore
political
republican
dole
presidential
senator
house

stock
market
percent
fund
investors
funds
companies
stocks
investment
trading

restaurant
sauce
menu
food
dishes
street
dining
dinner
chicken
served

budget
tax
governor
county
mayor
billion
taxes
plan
legislature
fiscal

Topic analysis (LDA) from 1.8 millions of New York Times articles

How it works

Text vectorization

Represent each document as a feature vector in the vector space, where each position represents a word (token) and the contained value is its relevance in the document.

- *BoW (Bag of words)*
- *TF-IDF (Term Frequency - Inverse Document Frequency)*

	D1	D2	D3	D4
linux	3	4	1	0
modem	4	3	0	1
the	3	4	4	3
clutch	0	1	4	3
steering	2	0	3	3
petrol	0	1	3	4

Document Term Matrix - Bag of Words

Text vectorization

TF-IDF - More “relevant” terms in a document are frequent terms in the document and rare in other documents

f_{ij} = frequency of term (feature) i in doc (item) j

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$$

n_i = number of docs that mention term i

N = total number of docs

$$IDF_i = \log \frac{N}{n_i}$$

TF-IDF score: $w_{ij} = TF_{ij} \times IDF_i$

Doc profile = set of words with highest **TF-IDF** scores, together with their scores

TF-IDF example with scikit-learn

```
from sklearn.feature_extraction.text import TfidfVectorizer  
vectorizer = TfidfVectorizer(max_df=0.5, max_features=1000,  
                           min_df=2, stop_words='english')  
tfidf_corpus = vectorizer.fit_transform(text_corpus)
```

		<i>tokens</i>									
		face	person	guide	lock	cat	dog	sleep	micro	pool	gym
		0	1	2	3	4	5	6	7	8	9
D1			0.05					0.25			
D2	0.02				0.32					0.45	
...											

TF-IDF sparse matrix example

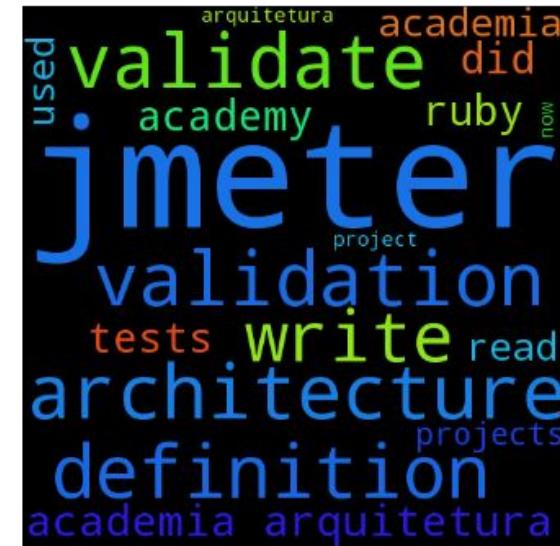
Text vectorization

Example

“Did you ever wonder how great it would be if you could **write** your **jmeter tests** in **ruby** ? This projects aims to do so. If you use it on your project just let me now. On the Architecture Academy you can read how jmeter can be used to **validate** your Architecture. **definition** | architecture **validation** | academia de **arquitetura**”

Relevant keywords (TF-IDF)

Tokens (unigrams and bigrams)	Weight
jmeter	0.466
architecture	0.380
validate	0.243
validation	0.242
definition	0.239
write	0.225
academia arquitetura	0.218
academy	0.216
ruby	0.213
tests	0.209



Text vectorization

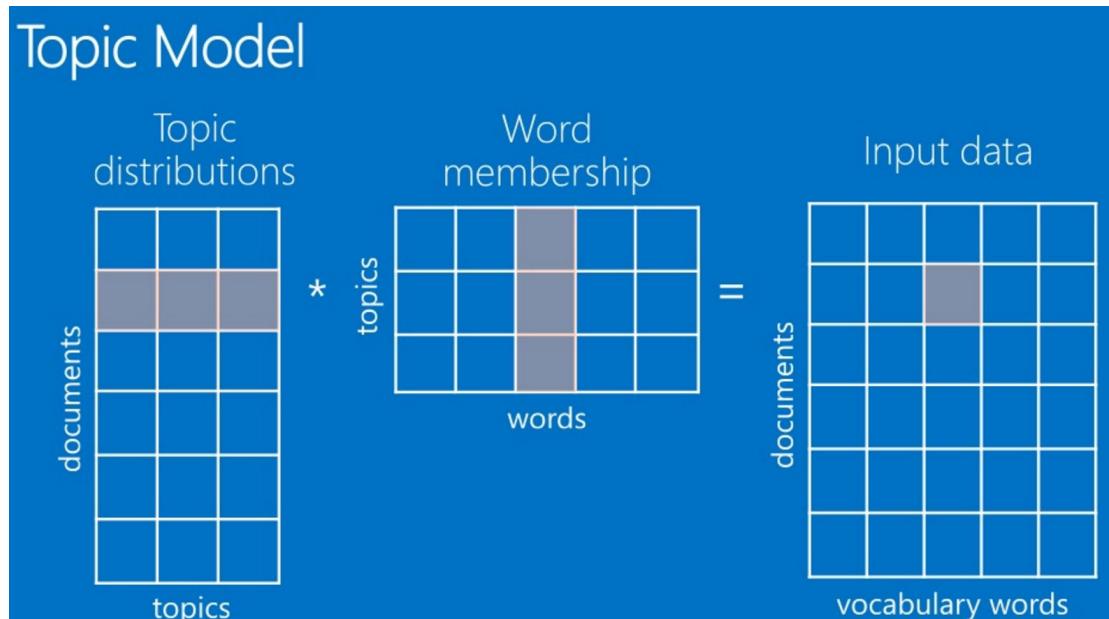
Visualization of the average TF-IDF vectors of posts at Google+ for Work (CI&T)



See the more details about this social and text analytics at http://bit.ly/python4ds_nb

Main techniques

- ***Latent Dirichlet Allocation (LDA)*** -> Probabilistic
- ***Latent Semantic Indexing / Analysis (LSI / LSA)*** -> Matrix Factorization
- ***Non-Negative Matrix Factorization (NMF)*** -> Matrix Factorization



Topic Modeling example (Python / [gensim](#))

```
1 from gensim import corpora, models, similarities  
2 documents = ["Human machine interface for lab abc computer applications",  
3               "A survey of user opinion of computer system response time",  
4               "The EPS user interface management system", ...]  
5 stoplist = set('for a of the and to in'.split())  
6 texts = [[word for word in document.lower().split() if word not in stoplist] for document in documents]  
7 dictionary = corpora.Dictionary(texts)  
8 corpus = [dictionary.doc2bow(text) for text in texts]  
9 tfidf = models.TfidfModel(corpus)  
10 corpus_tfidf = tfidf[corpus]  
11 lsi = models.LsiModel(corpus_tfidf, id2word=dictionary, num_topics=2)  
12 corpus_lsi = lsi[corpus_tfidf]  
13 lsi.print_topics(2)
```

Example of topic modeling using LSI technique

```
topic #0(1.594): 0.703*"trees" + 0.538*"graph" + 0.402*"minors" + 0.187*"survey" + ...  
topic #1(1.476): 0.460*"system" + 0.373*"user" + 0.332*"eps" + 0.328*"interface" + 0.320*"response" + ...
```

Example of the 2 topics discovered in the corpus

Topic Modeling example (PySpark MLLib LDA)

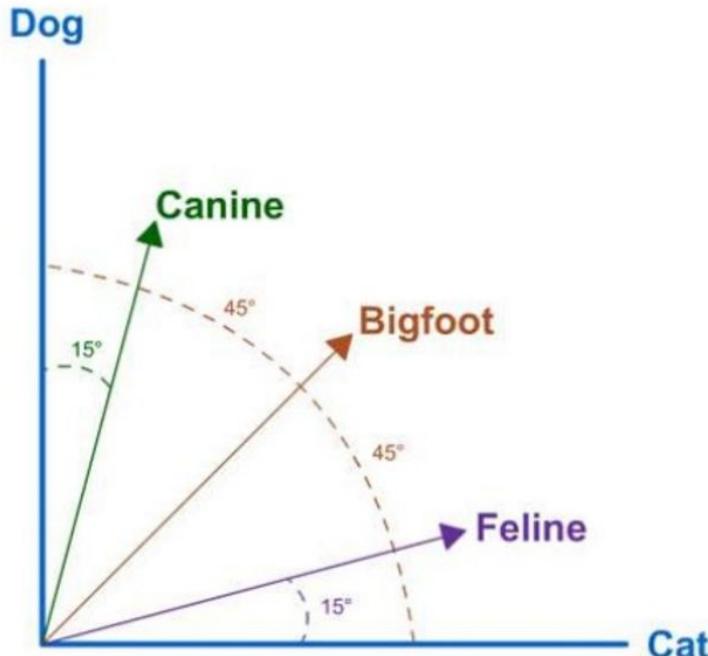
```
1 from pyspark.mllib.clustering import LDA, LDAModel
2 from pyspark.mllib.linalg import Vectors
3 # Load and parse the data
4 data = sc.textFile("data/mllib/sample_lda_data.txt")
5 parsedData = data.map(lambda line: Vectors.dense([float(x) for x in line.strip().split(' ')]))
6 # Index documents with unique IDs
7 corpus = parsedData.zipWithIndex().map(lambda x: [x[1], x[0]]).cache()
8 # Cluster the documents into three topics using LDA
9 ldaModel = LDA.train(corpus, k=3)
```

Sometimes simpler is better!

Scikit-learn topic models worked better for us than Spark MLlib LDA distributed implementation because we have thousands of users with hundreds of posts, and running scikit-learn on workers for each person was much quicker than running distributed Spark LDA for each person.

Cosine similarity

Similarity metric between two vectors is cosine among the angle between them



$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos \theta$$

$$\cos \theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

```
from sklearn.metrics.pairwise import cosine_similarity  
cosine_similarity(tfidf_matrix[0:1], tfidf_matrix)
```

Example with scikit-learn

People similarity

GABRIEL MOREIRA INTERESTS



Gabriel Moreira

Product Engineer Master
gabrielpm@ciantt.com

Activities on Google+

Analysed Posts: **42**

Average interactions on Gabriel Moreira posts: **3.5**

Main terms* in G+ (posts, comments and +1s)



* Click in the terms to search for people interested in them.

10 people with similar interests



Gilmar Souza (40%) - Posts



Henrique Souza (28%) - Posts



Mars Cyrillo (26%) - Posts



Johann Vivot (24%) - Posts



Fabio Fogliarini Brolesi (24%) - Posts



Rubens Barreto (23%) - Posts



Fulvio P. Parmejani (21%) - Posts



Fabio da Silva Santos (20%) - Posts



Lucas Arruda (20%) - Posts



Mauricio Pedroso (19%) - Posts



Felipe Antonio Souza Dewulf (19%) - Posts

Example of relevant keywords for a person and people with similar interests

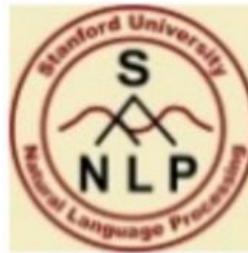
Frameworks - Topic Modeling



[Python](#)



[Java](#)



Stanford Topic Modeling Toolbox (Excel plugin)
<http://nlp.stanford.edu/software/tmt/tmt-0.4/>



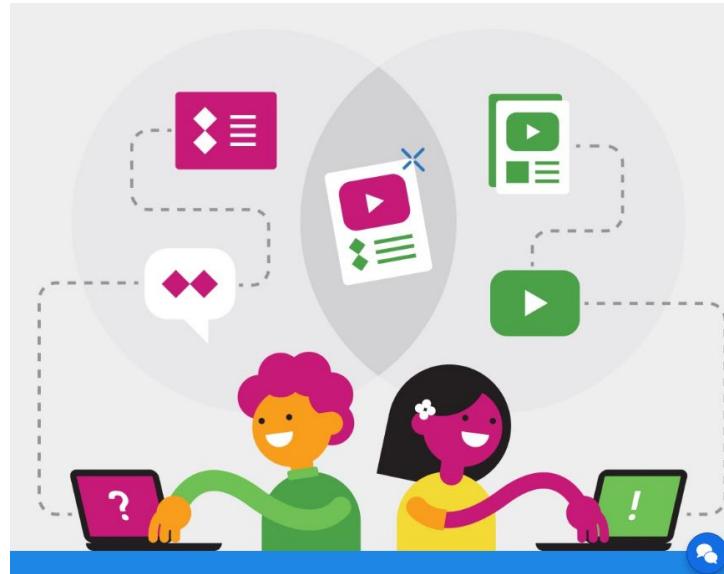
[Python / Scala](#)

Case

Smart Canvas[©]

Corporate Collaboration

Powered by Recommender Systems and Topic Modeling techniques



<http://www.smartcanvas.com/>

Content recommendations

The screenshot shows the 'Discover' channel in the Smart Canvas interface. The left sidebar includes links for 'For you', 'Discover' (which is selected), 'Everything', 'Boards', 'People', 'Teams', 'Bookmarks', 'Your profile', 'Settings', 'Help & Feedback', and 'Log out'. The main area is divided into three sections of recommendations:

- Suggestions based on what people like you posted or read:**
 - [OKR] D1 Q2/2016 (edited 8 days ago) by Google Sheets
 - A group of young people playing soccer by Google
 - How Google's AI paved the way for the next generation of bots by VentureBeat
 - SpaceX Falcon 9 - Successful Drone Ship Landing - 8th April 2016 by YouTube
- Suggestions based on your reads about machine learning, google, ci&t:**
 - Google takes Cloud Machine Learning service mainstream by Google Cloud Platform
 - Google: Machine Learning For Spam Detection & Search Quality Is Coming by seroundtable.com
 - Google launches new machine learning platform by TechCrunch
 - 'Machine learning' is a revolution as big as the internet or personal computers by Tech Insider
- Suggestions based on your reads about smart canvas:**
 - TN 2016 Goals: SAAS users by a user icon
 - Smart Canvas Backlog and Ideas (edited on Feb 28) by a user icon
 - Improving the Google Drive for Work experience - Accept and by a user icon
 - Smart Canvas Release Notes by Mars (edited on Mar 9) with a yellow '+' button

Discover channel - Content recommendations with personalized explanations, based on user's topics of interest (discovered from their contributed content and reads)

Person topics of interest



Gabriel Moreira (gabrielpm)

#algorithm #computer #python #machine_learning

✉ gabrielpm@ciandt.com

LESS ^

User profile - Topics of interest of users are from the content that they contribute and are presented as tags in their profile.

Searching people interested in topics / experts...

A screenshot of a search interface. At the top left is a back arrow icon. Next to it is the search term "machine learning". To the right are a microphone icon and a magnifying glass icon. Below the search bar is a list of results. Each result item has a small icon on the left: a grid icon for categories, a person icon for users, and a document icon for articles. The results are:

- Machine Learning
- Language Learning
- Bayesian machine learning
- Add insights with machine learning
- The Machine Learning GCP Spectrum
- Gabriel Moreira (gabrielpm) - gabrielpm@ciandt.com
- Gilmar Souza (gilmarj) - Head of Machine Learning - +55 19 33333333 - gilmarj@ciandt.com
- Machine Learning Practitioners

User discovered tags are searchable, allowing to find experts or people with specific interests.

Similar people

[← People related](#)



People that are also interested in **artificial intelligence, google**

Mars Cyrillo
VP Products and Cognitive Computing

[Call](#) [Message](#) [More](#)

Eduardo Sangjon
Product Planner @D1

[Call](#) [Message](#) [More](#)

Lucas Persona
Chief Digital Evangelist

[Call](#) [Message](#) [More](#)

Daniel Viveiros
Head of Product Engineering

[Call](#) [Message](#) [More](#)

People that are also interested in **machine learning**

Gabriel Moreira

[Call](#) [Message](#) [More](#)

Mars Cyrillo
VP Products and Cognitive Computing

[Call](#) [Message](#) [More](#)

Lucas Arruda
Software Architect

[Call](#) [Message](#) [More](#)

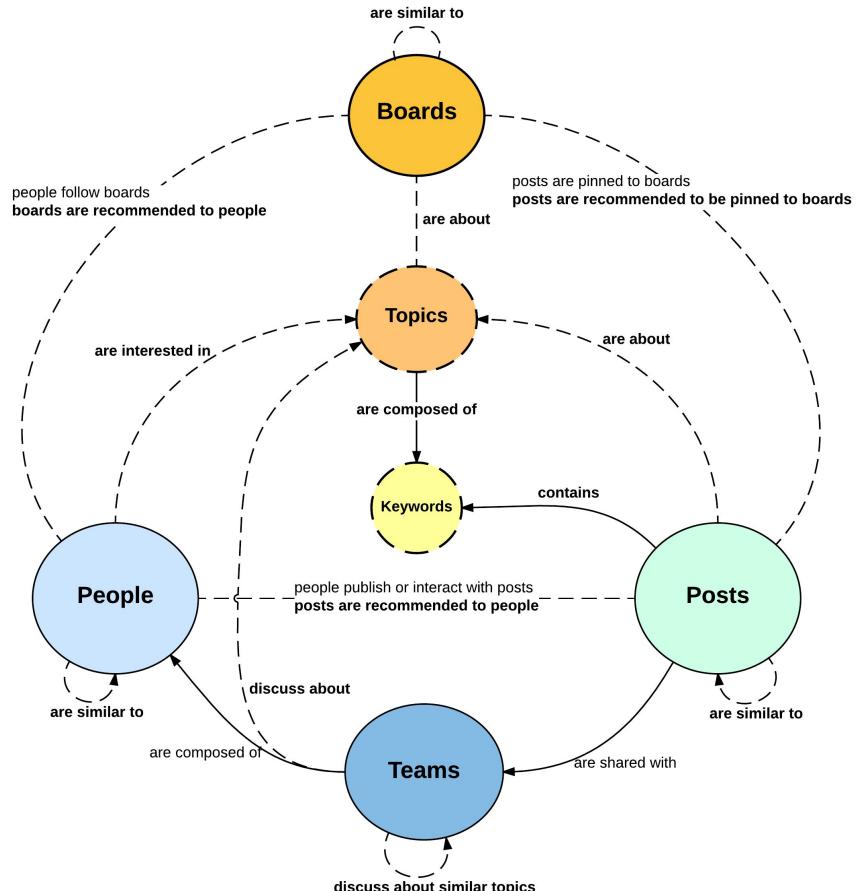
Cesar Gon
CEO

[Call](#) [Message](#) [More](#)

People recommendation - Recommends people with similar interests, explaining which topics are shared.

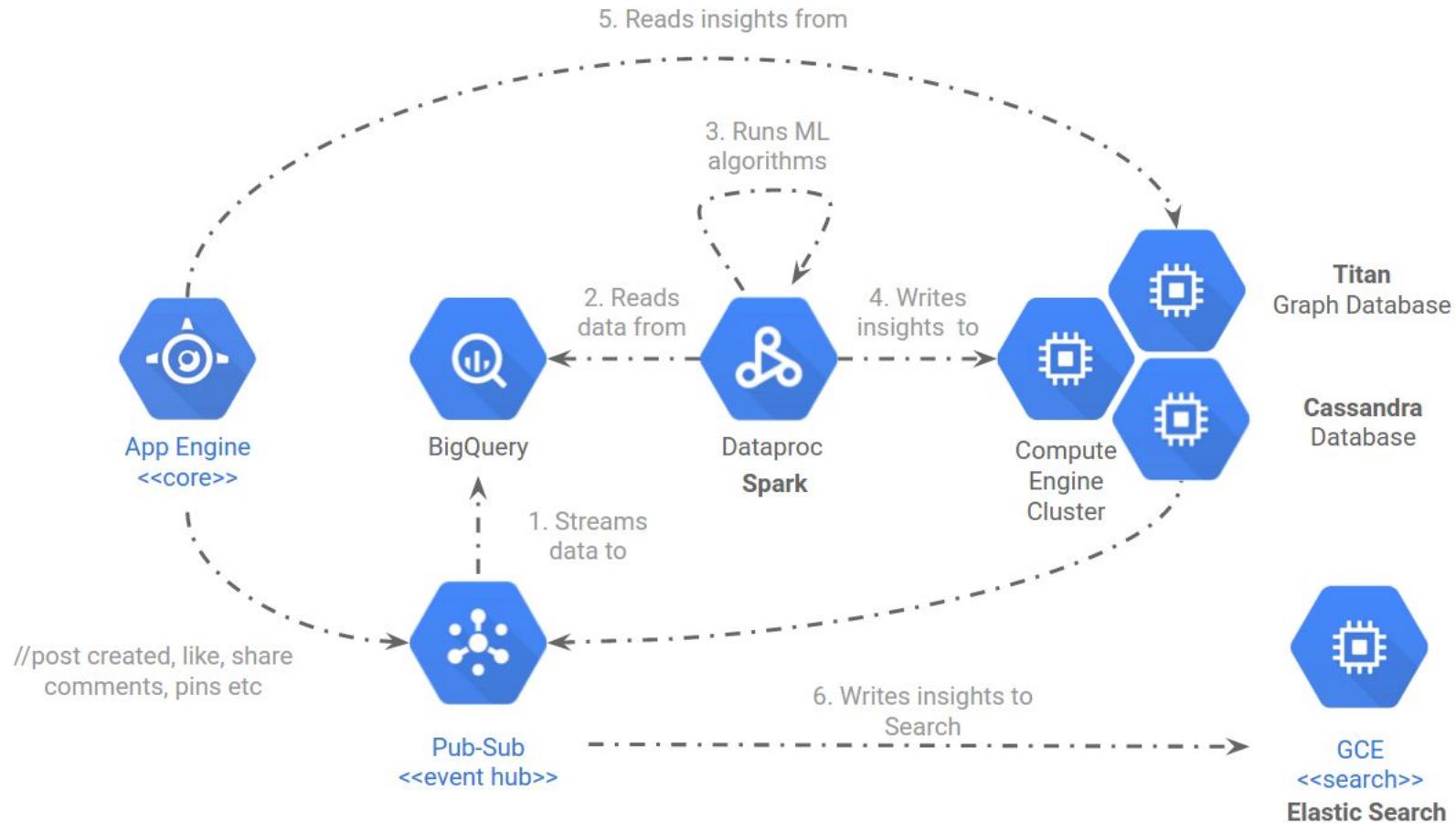
How it works

Collaboration Graph Model

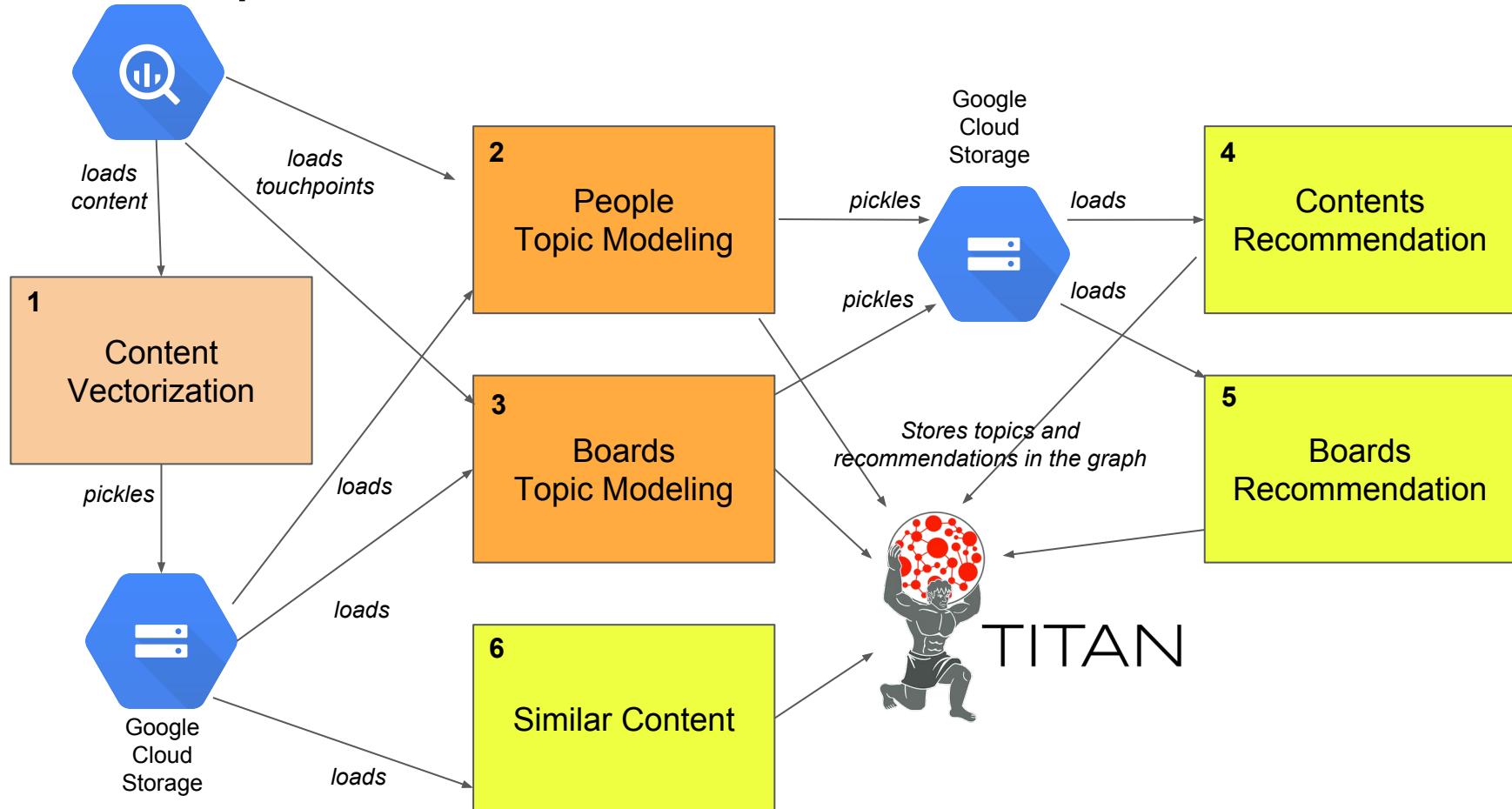


Smart Canvas © Graph Model: Dashed lines and bold labels are the relationships inferred by usage of RecSys and Topic Modeling techniques

Architecture Overview



Jobs Pipeline



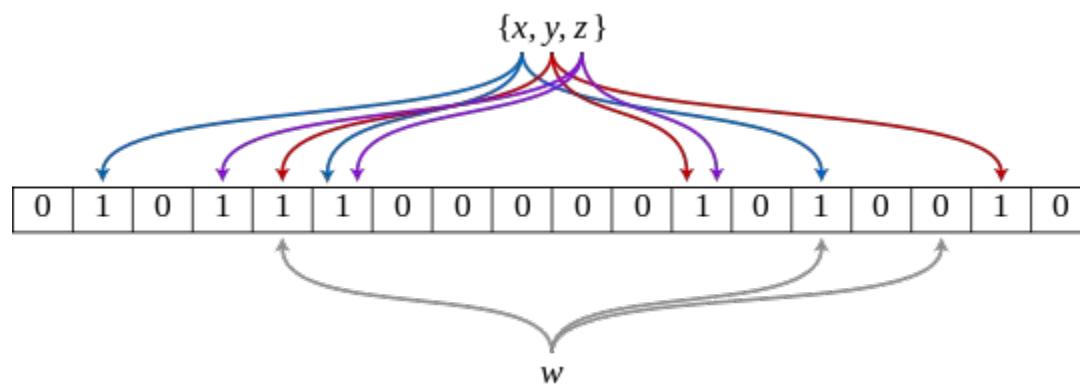
People Topic Modeling and Content Recommendations algorithm



1. Groups all touchpoints (user interactions) by person
2. For each person
 - a. Selects contents user has interacted
 - b. Clusters contents TF-IDF vectors to model (e.g. LDA, NMF, Kmeans) person' topics of interest (varying the k from a range (1-5) and selecting the model whose clusters best describe cohesive topics, penalizing large k)
 - c. Weights clusters relevance (to the user) by summing touchpoints strength (view, like, bookmark, ...) of each cluster, with a time decay on interaction age (older interactions are less relevant to current person interest)
 - d. **Returns highly relevant topics** vectors for the person,labeled by its three top keywords
3. For each people topic
 - a. Calculate the cosine similarity (optionally applying Pivoted Unique Pivoted Normalization) among the topic vector and content TF-IDF vectors
 - b. **Recommends more similar contents** to person user topics, which user has not yet

Bloom Filter

- A space-efficient Probabilistic Data Structure used to test whether an element is a member of a set.
- False positive matches are possible, but false negatives are not.
- An empty Bloom filter is a bit array of m bits, all set to 0.
- Each inserted key is mapped to bits (which are all set to 1) by k different hash functions.



The set $\{x, y, z\}$ was inserted in Bloom Filter. w is not in the set, because it hashes to one bit equal to 0

Bloom Filter - Example

Using **cross_bloomfilter** - Pure Python and Java compatible implementation
GitHub: http://bit.ly/cross_bf

```
In [4]: #Max number of entries
BLOOM_CAPACITY = 100000;
#False positive error rate
BLOOM_ERROR_RATE = 0.005;

bloom_filter = BloomFilter(BLOOM_CAPACITY, BLOOM_ERROR_RATE)
```

```
In [5]: all_keys = get_random_keys(BLOOM_CAPACITY)
for key in all_keys:
    bloom_filter.add(key)
```

```
In [8]: #Checks for keys existing in the set
errors = 0
for key in all_keys:
    if key not in bloom_filter:
        errors += 1

if errors > 0:
    print("Some included keys were not found in the Bloom Filter (false negatives)")
```

```
In [11]: #Checks for keys not existing in the set
errors = 0
for key in all_keys:
    if (key+"DOES NOT EXISTS") in bloom_filter:
        errors += 1

error_rate = float(errors) / BLOOM_CAPACITY;
print(error_rate)
```

0.00511

Bloom Filter - Complexity Analysis

Ordered List

Space Complexity: $O(n)$
where n is the number of items in the set

Check for key presence

Time Complexity (binary search): $O(\log(n))$
assuming an ordered list

Bloom Filter

Space Complexity: $O(t * \ln(p))$
where t is the maximum number of items to be inserted and p the accepted false positive rate

Check for key presence

Time Complexity: $O(k)$
where k is a constant representing the number of hash functions to be applied to a string key

Example: For a list of 100,000 string keys, with an average length of 18 characters

Key list size: 1,788,900 bytes

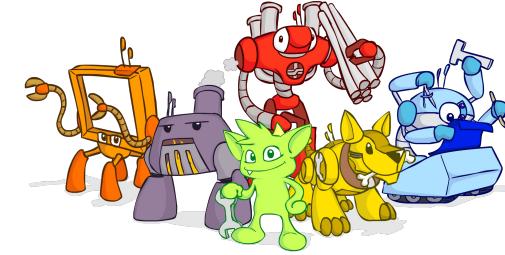
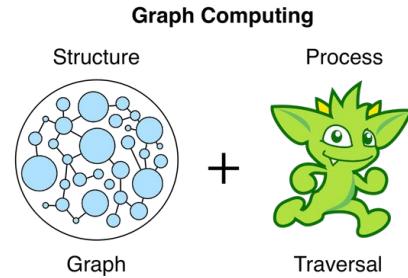
Bloom filter size: 137,919 bytes (false positive rate = 0.5%)

Bloom filter size: 18,043 bytes (false positive rate = 5%)

(and you can gzip it!)



TITAN



TinkerPop

- Distributed Graph Database
- Elastic and linear scalability for a growing data and user base
- Support for ACID and eventual consistency
- Backended by Cassandra or HBase
- Support for global graph data analytics, reporting, and ETL through integration with Spark, Hadoop, Giraph
- Support for geo and full text search (ElasticSearch, Lucene, Solr)
- Native integration with TinkerPop (Gremlin query language, Gremlin Server, ...)

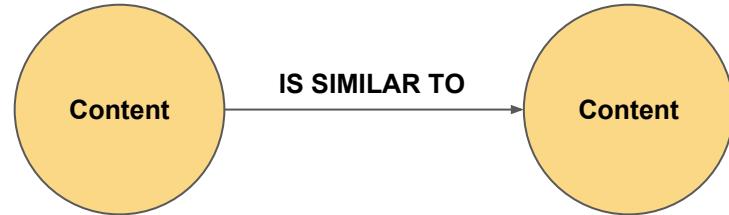


APACHE
HBASE

Gremlin traversals

Querying similar contents

```
1 g.V().has('CONTENT','id', 'Post:123')
2 .outE('IS SIMILAR TO')
3 .has('strength', gte(0.1)).as('e').inV().as('v')
4 .select('e','v')
5 .map{['id': it.get().v.values('id').next(),
6       'strength': it.get().e.values('strength').next()]}
7 .limit(10)
```



Example output:

```
[{'id': 'Post:456', 'strength': 0.672},
 {'id': 'Post:789', 'strength': 0.453},
 {'id': 'Post:333', 'strength': 0.235},
 ...]
```

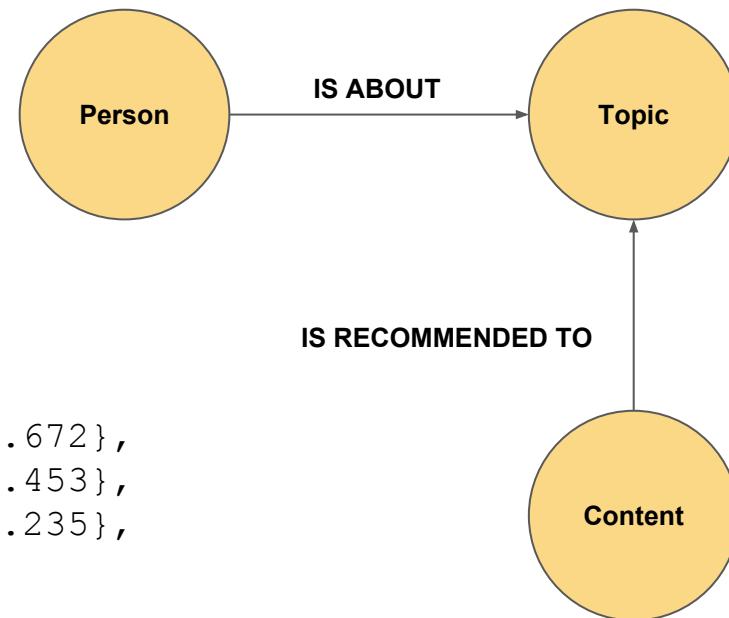
Gremlin traversals

Querying recommended contents for a person

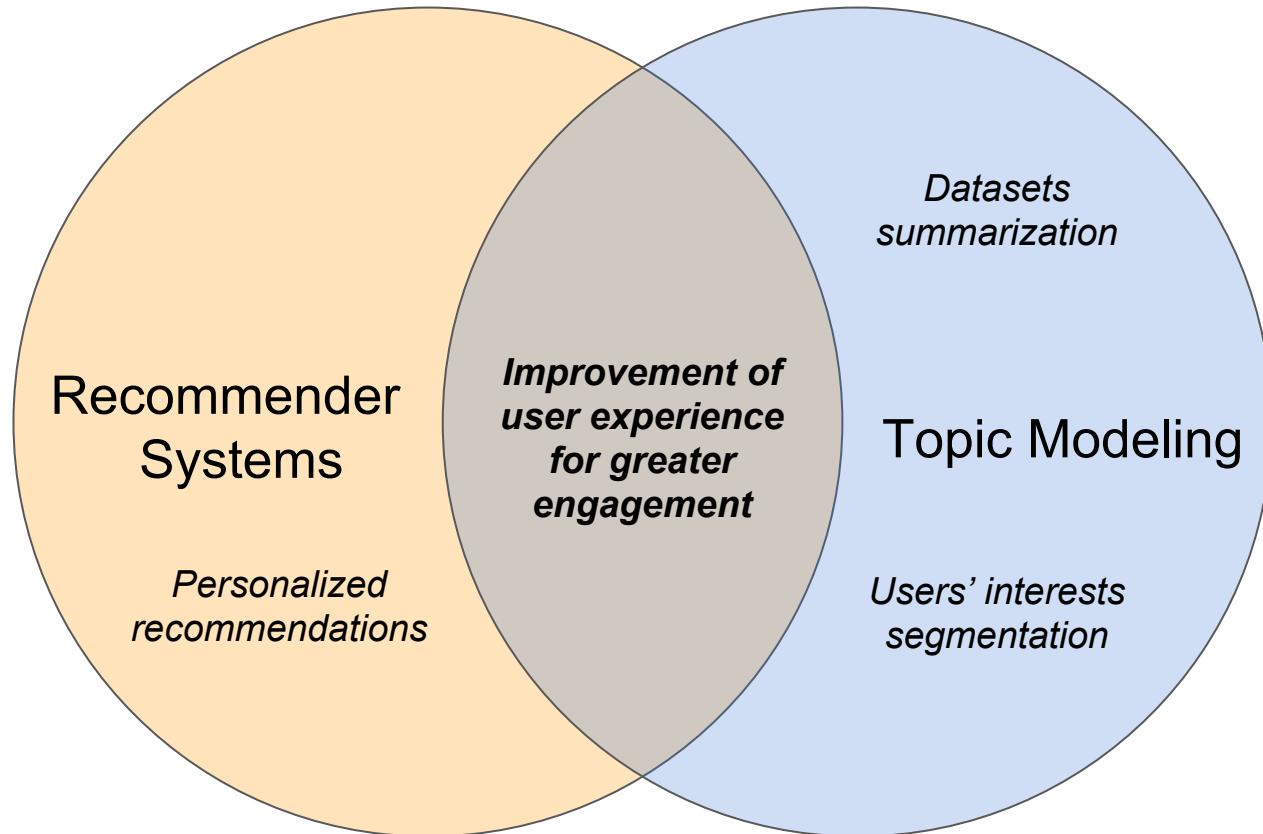
```
1 g.V().has('PERSON','id','Person:gabrielpm@ciandt.com')
2 .out('IS ABOUT').has('number', 1)
3 .inE('IS RECOMMENDED TO').filter{ it.get().value('strength') >= 0.1}
4 .order().by('strength',decr).as('tc').outV().as('c')
5 .select('tc','c')
6 .map{ ['contentId': it.get().c.value('id'),
7       'recommendationStrength': it.get().tc.value('strength') *
8           getTimeDecayFactor(it.get().c.value('updatedOn')) ] }
```

Example output:

```
[ { 'contentId': 'Post:456', 'strength': 0.672},
  { 'contentId': 'Post:789', 'strength': 0.453},
  { 'contentId': 'Post:333', 'strength': 0.235},
  ... ]
```



Wrap up



Start building smarter, deeper connections
and discover what you are missing inside
your own organization, for free.



Smart Canvas

<http://www.smartcanvas.com/>



Thanks!

Discovering Users' Topics of Interest in Recommender Systems

Gabriel Moreira - @gspmoreira
Gilmar Souza - @gilmarsouza