In [1]:
```
import numpy as np
import scipy
import pandas as pd
import math
import random
import sklearn
from nltk.corpus import stopwords
from scipy.sparse import csr_matrix
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from scipy.sparse.linalg import svds
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
```

In [2]:
```
articles_df = pd.read_csv('../input/shared_articles.csv')
articles_df = articles_df[articles_df['eventType'] == 'CONTENT SHARED']
articles_df.head(10)
```

Out[2]:

authorSessionId	authorUserAgent	authorRegion	authorCountry	contentType	
341205206233829	NaN	NaN	NaN	HTML	http://www.nytimes.com/
341205206233829	NaN	NaN	NaN	HTML	http://cointelegraph.c
532940883382585	NaN	NaN	NaN	HTML	https://cloudplatform.gc
341205206233829	NaN	NaN	NaN	HTML	https://bitcoinmagazin
341205206233829	NaN	NaN	NaN	HTML	http://www.coindesk.co
341205206233829	NaN	NaN	NaN	HTML	http://www.newsbtc.co
341205206233829	NaN	NaN	NaN	HTML	https://bitcoinmagazi
341205206233829	NaN	NaN	NaN	HTML	https://news.bitcoin.c
341205206233829	NaN	NaN	NaN	HTML	https://www.cryptocoins
441319395545950	NaN	NaN	NaN	HTML	http://economia.ig.com

In [3]:
```
interactions_df = pd.read_csv('../input/users_interactions.csv')
interactions_df.head(10)
```

Out[3]:

	timestamp	eventType	contentId	personId	sessionId	
0	1465413032	VIEW	-3499919498720038879	-8845298781299428018	1264196770339959068	
1	1465412560	VIEW	8890720798209849691	-1032019229384696495	3621737643587579081	(M
2	1465416190	VIEW	310515487419366995	-1130272294246983140	2631864456530402479	
3	1465413895	FOLLOW	310515487419366995	344280948527967603	-3167637573980064150	
4	1465412290	VIEW	-7820640624231356730	-445337111692715325	5611481178424124714	
5	1465413742	VIEW	310515487419366995	-8763398617720485024	1395789369402380392	
6	1465415950	VIEW	-8864073373672512525	3609194402293569455	1143207167886864524	
7	1465415066	VIEW	-1492913151930215984	4254153380739593270	8743229464706506141	M Ap
8	1465413762	VIEW	310515487419366995	344280948527967603	-3167637573980064150	
9	1465413771	VIEW	3064370296170038610	3609194402293569455	1143207167886864524	

In [4]:
```
event_type_strength = {
    'VIEW': 1.0,
    'LIKE': 2.0,
    'BOOKMARK': 2.5,
    'FOLLOW': 3.0,
    'COMMENT CREATED': 4.0,
}

interactions_df['eventStrength'] = interactions_df['eventType'].apply(lambd
```

In [5]:
```
users_interactions_count_df = interactions_df.groupby(['personId', 'content
print('# users: %d' % len(users_interactions_count_df))
users_with_enough_interactions_df = users_interactions_count_df[users_inter
print('# users with at least 5 interactions: %d' % len(users_with_enough_in
```

```
# users: 1895
# users with at least 5 interactions: 1140
```

In [6]:
```
print('# of interactions: %d' % len(interactions_df))
interactions_from_selected_users_df = interactions_df.merge(users_with_enou
               how = 'right',
               left_on = 'personId',
               right_on = 'personId')
print('# of interactions from users with at least 5 interactions: %d' % len
```

```
# of interactions: 72312
# of interactions from users with at least 5 interactions: 69868
```

In [7]:
```
def smooth_user_preference(x):
    return math.log(1+x, 2)

interactions_full_df = interactions_from_selected_users_df \
                    .groupby(['personId', 'contentId'])['eventStrength'].su
                    .apply(smooth_user_preference).reset_index()
print('# of unique user/item interactions: %d' % len(interactions_full_df))
interactions_full_df.head(10)
```

```
# of unique user/item interactions: 39106
```

Out[7]:

	personId	contentId	eventStrength
0	-9223121837663643404	-8949113594875411859	1.000000
1	-9223121837663643404	-8377626164558006982	1.000000
2	-9223121837663643404	-8208801367848627943	1.000000
3	-9223121837663643404	-8187220755213888616	1.000000
4	-9223121837663643404	-7423191370472335463	3.169925
5	-9223121837663643404	-7331393944609614247	1.000000
6	-9223121837663643404	-6872546942144599345	1.000000
7	-9223121837663643404	-6728844082024523434	1.000000
8	-9223121837663643404	-6590819806697898649	1.000000
9	-9223121837663643404	-6558712014192834002	1.584963

In [8]:
```
interactions_train_df, interactions_test_df = train_test_split(interactions
                                   stratify=interactions_full_df['personId'
                                   test_size=0.20,
                                   random_state=42)

print('# interactions on Train set: %d' % len(interactions_train_df))
print('# interactions on Test set: %d' % len(interactions_test_df))
```

```
# interactions on Train set: 31284
# interactions on Test set: 7822
```

In [9]:
```
#Indexing by personId to speed up the searches during evaluation
interactions_full_indexed_df = interactions_full_df.set_index('personId')
interactions_train_indexed_df = interactions_train_df.set_index('personId')
interactions_test_indexed_df = interactions_test_df.set_index('personId')
```

In [10]:
```
def get_items_interacted(person_id, interactions_df):
    # Get the user's data and merge in the movie information.
    interacted_items = interactions_df.loc[person_id]['contentId']
    return set(interacted_items if type(interacted_items) == pd.Series else
```

In [11]:
```
EVAL_RANDOM_SAMPLE_NON_INTERACTED_ITEMS = 100

class ModelEvaluator:


    def get_not_interacted_items_sample(self, person_id, sample_size, seed=
        interacted_items = get_items_interacted(person_id, interactions_ful
        all_items = set(articles_df['contentId'])
        non_interacted_items = all_items - interacted_items

        random.seed(seed)
        non_interacted_items_sample = random.sample(non_interacted_items, s
        return set(non_interacted_items_sample)

    def _verify_hit_top_n(self, item_id, recommended_items, topn):
            try:
                index = next(i for i, c in enumerate(recommended_items) if
            except:
                index = -1
            hit = int(index in range(0, topn))
            return hit, index

    def evaluate_model_for_user(self, model, person_id):
        #Getting the items in test set
        interacted_values_testset = interactions_test_indexed_df.loc[person
        if type(interacted_values_testset['contentId']) == pd.Series:
            person_interacted_items_testset = set(interacted_values_testset
        else:
            person_interacted_items_testset = set([int(interacted_values_te
        interacted_items_count_testset = len(person_interacted_items_testse

        #Getting a ranked recommendation list from a model for a given user
        person_recs_df = model.recommend_items(person_id,
                                               items_to_ignore=get_items_in

                                               topn=10000000000)

        hits_at_5_count = 0
        hits_at_10_count = 0
        #For each item the user has interacted in test set
        for item_id in person_interacted_items_testset:
            #Getting a random sample (100) items the user has not interacte
            #(to represent items that are assumed to be no relevant to the
            non_interacted_items_sample = self.get_not_interacted_items_sam
                                                                          s
                                                                          s

            #Combining the current interacted item with the 100 random item
            items_to_filter_recs = non_interacted_items_sample.union(set([i

            #Filtering only recommendations that are either the interacted
            valid_recs_df = person_recs_df[person_recs_df['contentId'].isin
            valid_recs = valid_recs_df['contentId'].values
            #Verifying if the current interacted item is among the Top-N re
            hit_at_5, index_at_5 = self._verify_hit_top_n(item_id, valid_re
            hits_at_5_count += hit_at_5
```

```
            hit_at_10, index_at_10 = self._verify_hit_top_n(item_id, valid_
            hits_at_10_count += hit_at_10

        #Recall is the rate of the interacted items that are ranked among t
        #when mixed with a set of non-relevant items
        recall_at_5 = hits_at_5_count / float(interacted_items_count_testse
        recall_at_10 = hits_at_10_count / float(interacted_items_count_test

        person_metrics = {'hits@5_count':hits_at_5_count,
                          'hits@10_count':hits_at_10_count,
                          'interacted_count': interacted_items_count_testse
                          'recall@5': recall_at_5,
                          'recall@10': recall_at_10}
        return person_metrics

    def evaluate_model(self, model):
        #print('Running evaluation for users')
        people_metrics = []
        for idx, person_id in enumerate(list(interactions_test_indexed_df.i
            #if idx % 100 == 0 and idx > 0:
            #    print('%d users processed' % idx)
            person_metrics = self.evaluate_model_for_user(model, person_id)
            person_metrics['_person_id'] = person_id
            people_metrics.append(person_metrics)
        print('%d users processed' % idx)

        detailed_results_df = pd.DataFrame(people_metrics) \
                            .sort_values('interacted_count', ascending=Fals

        global_recall_at_5 = detailed_results_df['hits@5_count'].sum() / fl
        global_recall_at_10 = detailed_results_df['hits@10_count'].sum() /

        global_metrics = {'modelName': model.get_model_name(),
                          'recall@5': global_recall_at_5,
                          'recall@10': global_recall_at_10}
        return global_metrics, detailed_results_df

model_evaluator = ModelEvaluator()
```

In [12]:
```
item_popularity_df = interactions_full_df.groupby('contentId')['eventStreng
item_popularity_df.head(10)
```

Out[12]:

	contentId	eventStrength
0	-4029704725707465084	307.733799
1	-6783772548752091658	233.762157
2	-133139342397538859	228.024567
3	-8208801367848627943	197.107608
4	-6843047699859121724	193.825208
5	8224860111193157980	189.044680
6	-2358756719610361882	183.110951
7	2581138407738454418	180.282876
8	7507067965574797372	179.094002
9	1469580151036142903	170.548969

In [13]:
```
class PopularityRecommender:

    MODEL_NAME = 'Popularity'

    def __init__(self, popularity_df, items_df=None):
        self.popularity_df = popularity_df
        self.items_df = items_df

    def get_model_name(self):
        return self.MODEL_NAME

    def recommend_items(self, user_id, items_to_ignore=[], topn=10, verbose
        # Recommend the more popular items that the user hasn't seen yet.
        recommendations_df = self.popularity_df[~self.popularity_df['conten
                               .sort_values('eventStrength', ascending = Fa
                               .head(topn)

        if verbose:
            if self.items_df is None:
                raise Exception('"items_df" is required in verbose mode')

            recommendations_df = recommendations_df.merge(self.items_df, ho
                                                          left_on = 'conten
                                                          right_on = 'conte


        return recommendations_df

popularity_model = PopularityRecommender(item_popularity_df, articles_df)
```

In [14]:
```
print('Evaluating Popularity recommendation model...')
pop_global_metrics, pop_detailed_results_df = model_evaluator.evaluate_mode
print('\nGlobal metrics:\n%s' % pop_global_metrics)
pop_detailed_results_df.head(10)
```

```
Evaluating Popularity recommendation model...
1139 users processed

Global metrics:
{'modelName': 'Popularity', 'recall@5': 0.2418818716440808, 'recall@10':
0.3725389925850166}
```

Out[14]:

	hits@5_count	hits@10_count	interacted_count	recall@5	recall@10	_person_id
76	28	50	192	0.145833	0.260417	3609194402293569455
17	12	25	134	0.089552	0.186567	-2626634673110551643
16	13	23	130	0.100000	0.176923	-1032019229384696495
10	5	9	117	0.042735	0.076923	-1443636648652872475
82	26	40	88	0.295455	0.454545	-2979881261169775358
161	12	18	80	0.150000	0.225000	-3596626804281480007
65	20	34	73	0.273973	0.465753	1116121227607581999
81	17	23	69	0.246377	0.333333	692689608292948411
106	14	18	69	0.202899	0.260870	-9016528795238256703
52	21	28	68	0.308824	0.411765	3636910968448833585

In [25]:
```
#Ignoring stopwords (words with no semantics) from English and Portuguese (
stopwords_list = stopwords.words('english') + stopwords.words('portuguese')

#Trains a model whose vectors size is 5000, composed by the main unigrams a
vectorizer = TfidfVectorizer(analyzer='word',
                     ngram_range=(1, 2),
                     min_df=0.003,
                     max_df=0.5,
                     max_features=5000,
                     stop_words=stopwords_list)

item_ids = articles_df['contentId'].tolist()
tfidf_matrix = vectorizer.fit_transform(articles_df['title'] + "" + article
tfidf_feature_names = vectorizer.get_feature_names()
tfidf_matrix
```

Out[25]:
```
<3047x5000 sparse matrix of type '<class 'numpy.float64'>'
	with 638928 stored elements in Compressed Sparse Row format>
```

In [26]:
```
def get_item_profile(item_id):
    idx = item_ids.index(item_id)
    item_profile = tfidf_matrix[idx:idx+1]
    return item_profile

def get_item_profiles(ids):
    item_profiles_list = [get_item_profile(x) for x in ids]
    item_profiles = scipy.sparse.vstack(item_profiles_list)
    return item_profiles

def build_users_profile(person_id, interactions_indexed_df):
    interactions_person_df = interactions_indexed_df.loc[person_id]
    user_item_profiles = get_item_profiles(interactions_person_df['contentI

    user_item_strengths = np.array(interactions_person_df['eventStrength'])
    #Weighted average of item profiles by the interactions strength
    user_item_strengths_weighted_avg = np.sum(user_item_profiles.multiply(u
    user_profile_norm = sklearn.preprocessing.normalize(user_item_strengths
    return user_profile_norm

def build_users_profiles():
    interactions_indexed_df = interactions_train_df[interactions_train_df['
                                                    .isin(articles_df['conte
    user_profiles = {}
    for person_id in interactions_indexed_df.index.unique():
        user_profiles[person_id] = build_users_profile(person_id, interacti
    return user_profiles
```

In [27]:
```
user_profiles = build_users_profiles()
len(user_profiles)
```

Out[27]: 1140

In [28]:
```
myprofile = user_profiles[-1479311724257856983]
print(myprofile.shape)
pd.DataFrame(sorted(zip(tfidf_feature_names,
                        user_profiles[-1479311724257856983].flatten().tolis
             columns=['token', 'relevance'])
```

(1, 5000)

Out[28]:

	token	relevance
0	learning	0.298732
1	machine learning	0.245992
2	machine	0.237843
3	google	0.202839
4	data	0.169776
5	ai	0.156203
6	algorithms	0.115666
7	like	0.097744
8	language	0.087609
9	people	0.082024
10	deep	0.081542
11	deep learning	0.080979
12	research	0.076020
13	algorithm	0.074905
14	apple	0.074050
15	intelligence	0.072663
16	use	0.072597
17	human	0.072494
18	models	0.072388
19	artificial	0.072062

In [29]:
```
edRecommender:

 'Content-Based'

(self, items_df=None):
m_ids = item_ids
ms_df = items_df

l_name(self):
elf.MODEL_NAME

ilar_items_to_user_profile(self, person_id, topn=1000):
s the cosine similarity between the user profile and all item profiles
imilarities = cosine_similarity(user_profiles[person_id], tfidf_matrix)
e top similar items
indices = cosine_similarities.argsort().flatten()[-topn:]
e similar items by similarity
items = sorted([(item_ids[i], cosine_similarities[0,i]) for i in similar_inc
imilar_items

d_items(self, user_id, items_to_ignore=[], topn=10, verbose=False):
items = self._get_similar_items_to_user_profile(user_id)
 items the user has already interacted
items_filtered = list(filter(lambda x: x[0] not in items_to_ignore, similar_

dations_df = pd.DataFrame(similar_items_filtered, columns=['contentId', 'rec
                    .head(topn)

se:
elf.items_df is None:
raise Exception('"items_df" is required in verbose mode')

mmendations_df = recommendations_df.merge(self.items_df, how = 'left',
                                          left_on = 'contentId',
                                          right_on = 'contentId')[['recStren


ecommendations_df

commender_model = ContentBasedRecommender(articles_df)
```

In [30]:
```
print('Evaluating Content-Based Filtering model...')
cb_global_metrics, cb_detailed_results_df = model_evaluator.evaluate_model(
print('\nGlobal metrics:\n%s' % cb_global_metrics)
cb_detailed_results_df.head(10)
```

```
Evaluating Content-Based Filtering model...
1139 users processed

Global metrics:
{'modelName': 'Content-Based', 'recall@5': 0.16287394528253643, 'recall@1
0': 0.2614420864229097}
```

Out[30]:

	hits@5_count	hits@10_count	interacted_count	recall@5	recall@10	_person_id
76	15	24	192	0.078125	0.125000	3609194402293569455
17	18	29	134	0.134328	0.216418	-2626634673110551643
16	20	33	130	0.153846	0.253846	-1032019229384696495
10	32	47	117	0.273504	0.401709	-1443636648652872475
82	6	15	88	0.068182	0.170455	-2979881261169775358
161	11	23	80	0.137500	0.287500	-3596626804281480007
65	8	13	73	0.109589	0.178082	1116121227607581999
81	8	19	69	0.115942	0.275362	692689608292948411
106	3	9	69	0.043478	0.130435	-9016528795238256703
52	3	8	68	0.044118	0.117647	3636910968448833585

In [31]:
```
#Creating a sparse pivot table with users in rows and items in columns
users_items_pivot_matrix_df = interactions_train_df.pivot(index='personId',
                                                          columns='contentI
                                                          values='eventStre

users_items_pivot_matrix_df.head(10)
```

Out[31]:

contentId	-9222795471790223670	-9216926795620865886	-9194572880052200111	-91925
personId				
-9223121837663643404	0.0	0.0	0.0	
-9212075797126931087	0.0	0.0	0.0	
-9207251133131336884	0.0	2.0	0.0	
-9199575329909162940	0.0	0.0	0.0	
-9196668942822132778	0.0	0.0	0.0	
-9188188261933657343	0.0	0.0	0.0	
-9172914609055320039	0.0	0.0	0.0	
-9156344805277471150	0.0	0.0	0.0	
-9120685872592674274	0.0	0.0	0.0	
-9109785559521267180	0.0	0.0	0.0	

10 rows × 2926 columns

In [32]:
```
users_items_pivot_matrix = users_items_pivot_matrix_df.to_numpy()

users_items_pivot_matrix[:10]
```

Out[32]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 2., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [33]:
```
users_ids = list(users_items_pivot_matrix_df.index)
users_ids[:10]
```

Out[33]:
```
[-9223121837663643404,
 -9212075797126931087,
 -9207251133131336884,
 -9199575329909162940,
 -9196668942822132778,
 -9188188261933657343,
 -9172914609055320039,
 -9156344805277471150,
 -9120685872592674274,
 -9109785559521267180]
```

In [34]:
```
users_items_pivot_sparse_matrix = csr_matrix(users_items_pivot_matrix)
users_items_pivot_sparse_matrix
```

Out[34]:
```
<1140x2926 sparse matrix of type '<class 'numpy.float64'>'
	with 31284 stored elements in Compressed Sparse Row format>
```

In [35]:
```
#The number of factors to factor the user-item matrix.
NUMBER_OF_FACTORS_MF = 15
#Performs matrix factorization of the original user item matrix
#U, sigma, Vt = svds(users_items_pivot_matrix, k = NUMBER_OF_FACTORS_MF)
U, sigma, Vt = svds(users_items_pivot_sparse_matrix, k = NUMBER_OF_FACTORS_
```

In [36]:
```
U.shape
```

Out[36]:
```
(1140, 15)
```

In [37]:
```
Vt.shape
```

Out[37]:
```
(15, 2926)
```

In [38]:
```
sigma = np.diag(sigma)
sigma.shape
```

Out[38]:
```
(15, 15)
```

In [39]:
```
all_user_predicted_ratings = np.dot(np.dot(U, sigma), Vt)
all_user_predicted_ratings
```

Out[39]:
```
array([[ 0.01039915,  0.00081872, -0.01725263, ...,  0.00140708,
         0.0110647 ,  0.00226063],
       [-0.00019285, -0.00031318, -0.00264624, ...,  0.00251658,
         0.00017609, -0.00189488],
       [-0.01254721,  0.0065947 , -0.00590676, ...,  0.00698975,
        -0.01015696,  0.01154572],
       ...,
       [-0.02995379,  0.00805715, -0.01846307, ..., -0.01083078,
        -0.00118591,  0.0096798 ],
       [-0.01845505,  0.00467019,  0.01219602, ...,  0.00409507,
         0.00019482, -0.00752562],
       [-0.01506374,  0.00327732,  0.13391269, ..., -0.01191815,
         0.06422074,  0.01303244]])
```

In [40]:
```
all_user_predicted_ratings_norm = (all_user_predicted_ratings - all_user_pr
```

In [41]:
```
#Converting the reconstructed matrix back to a Pandas dataframe
cf_preds_df = pd.DataFrame(all_user_predicted_ratings_norm, columns = users
cf_preds_df.head(10)
```

Out[41]:

	-9223121837663643404	-9212075797126931087	-9207251133131336884	-91995
contentId				
-9222795471790223670	0.139129	0.137930	0.136531	
-9216926795620865886	0.138044	0.137916	0.138698	
-9194572880052200111	0.135998	0.137652	0.137283	
-9192549002213406534	0.141924	0.137996	0.134663	
-9190737901804729417	0.140209	0.137408	0.138708	
-9189659052158407108	0.138932	0.138699	0.138117	
-9176143510534135851	0.143208	0.138673	0.139514	
-9172673334835262304	0.138527	0.138021	0.138274	
-9171475473795142532	0.140720	0.137865	0.138061	
-9166778629773133902	0.138989	0.137725	0.136520	

10 rows × 1140 columns

In [42]:
```
len(cf_preds_df.columns)
```

Out[42]:
```
1140
```

In [43]:
```
der:

 'Collaborative Filtering'

(self, cf_predictions_df, items_df=None):
predictions_df = cf_predictions_df
ms_df = items_df

l_name(self):
elf.MODEL_NAME

d_items(self, user_id, items_to_ignore=[], topn=10, verbose=False):
d sort the user's predictions
ser_predictions = self.cf_predictions_df[user_id].sort_values(ascending=Fals
                    .reset_index().rename(columns={user_id: 'recStrength'})

end the highest predicted rating movies that the user hasn't seen yet.
dations_df = sorted_user_predictions[~sorted_user_predictions['contentId'].i
               .sort_values('recStrength', ascending = False) \
               .head(topn)

se:
elf.items_df is None:
raise Exception('"items_df" is required in verbose mode')

mmendations_df = recommendations_df.merge(self.items_df, how = 'left',
                                          left_on = 'contentId',
                                          right_on = 'contentId')[['recStren


ecommendations_df

odel = CFRecommender(cf_preds_df, articles_df)
```

In [44]:
```
print('Evaluating Collaborative Filtering (SVD Matrix Factorization) model.
cf_global_metrics, cf_detailed_results_df = model_evaluator.evaluate_model(
print('\nGlobal metrics:\n%s' % cf_global_metrics)
cf_detailed_results_df.head(10)
```

```
Evaluating Collaborative Filtering (SVD Matrix Factorization) model...
1139 users processed

Global metrics:
{'modelName': 'Collaborative Filtering', 'recall@5': 0.33392994119151115,
'recall@10': 0.46803886474047557}
```

Out[44]:

	hits@5_count	hits@10_count	interacted_count	recall@5	recall@10	_person_id
76	21	46	192	0.109375	0.239583	3609194402293569455
17	30	56	134	0.223881	0.417910	-2626634673110551643
16	16	34	130	0.123077	0.261538	-1032019229384696495
10	38	51	117	0.324786	0.435897	-1443636648652872475
82	39	48	88	0.443182	0.545455	-2979881261169775358
161	22	34	80	0.275000	0.425000	-3596626804281480007
65	24	32	73	0.328767	0.438356	1116121227607581999
81	16	21	69	0.231884	0.304348	692689608292948411
106	20	28	69	0.289855	0.405797	-9016528795238256703
52	23	30	68	0.338235	0.441176	3636910968448833585

In [45]:
```
HybridRecommender:

)DEL_NAME = 'Hybrid'

ef __init__(self, cb_rec_model, cf_rec_model, items_df, cb_ensemble_weight=1
   self.cb_rec_model = cb_rec_model
   self.cf_rec_model = cf_rec_model
   self.cb_ensemble_weight = cb_ensemble_weight
   self.cf_ensemble_weight = cf_ensemble_weight
   self.items_df = items_df

ef get_model_name(self):
   return self.MODEL_NAME

ef recommend_items(self, user_id, items_to_ignore=[], topn=10, verbose=False
   #Getting the top-1000 Content-based filtering recommendations
   cb_recs_df = self.cb_rec_model.recommend_items(user_id, items_to_ignore=i
                                                  topn=1000).rename(colu

   #Getting the top-1000 Collaborative filtering recommendations
   cf_recs_df = self.cf_rec_model.recommend_items(user_id, items_to_ignore=i
                                                  topn=1000).rename(colu

   #Combining the results by contentId
   recs_df = cb_recs_df.merge(cf_recs_df,
                              how = 'outer',
                              left_on = 'contentId',
                              right_on = 'contentId').fillna(0.0)

   #Computing a hybrid recommendation score based on CF and CB scores
   #recs_df['recStrengthHybrid'] = recs_df['recStrengthCB'] * recs_df['recSt
   recs_df['recStrengthHybrid'] = (recs_df['recStrengthCB'] * self.cb_ensemb
                                + (recs_df['recStrengthCF'] * self.cf_ensemb

   #Sorting recommendations by hybrid score
   recommendations_df = recs_df.sort_values('recStrengthHybrid', ascending=F

   if verbose:
       if self.items_df is None:
           raise Exception('"items_df" is required in verbose mode')

       recommendations_df = recommendations_df.merge(self.items_df, how = 'l
                                                     left_on = 'contentId',
                                                     right_on = 'contentId')


   return recommendations_df

d_recommender_model = HybridRecommender(content_based_recommender_model, cf_
                                        cb_ensemble_weight=1.0, cf_ensemble_
```

In [46]:
```
print('Evaluating Hybrid model...')
hybrid_global_metrics, hybrid_detailed_results_df = model_evaluator.evaluat
print('\nGlobal metrics:\n%s' % hybrid_global_metrics)
hybrid_detailed_results_df.head(10)
```

```
Evaluating Hybrid model...
1139 users processed

Global metrics:
{'modelName': 'Hybrid', 'recall@5': 0.34262336998210174, 'recall@10': 0.4
796727179749425}
```

Out[46]:

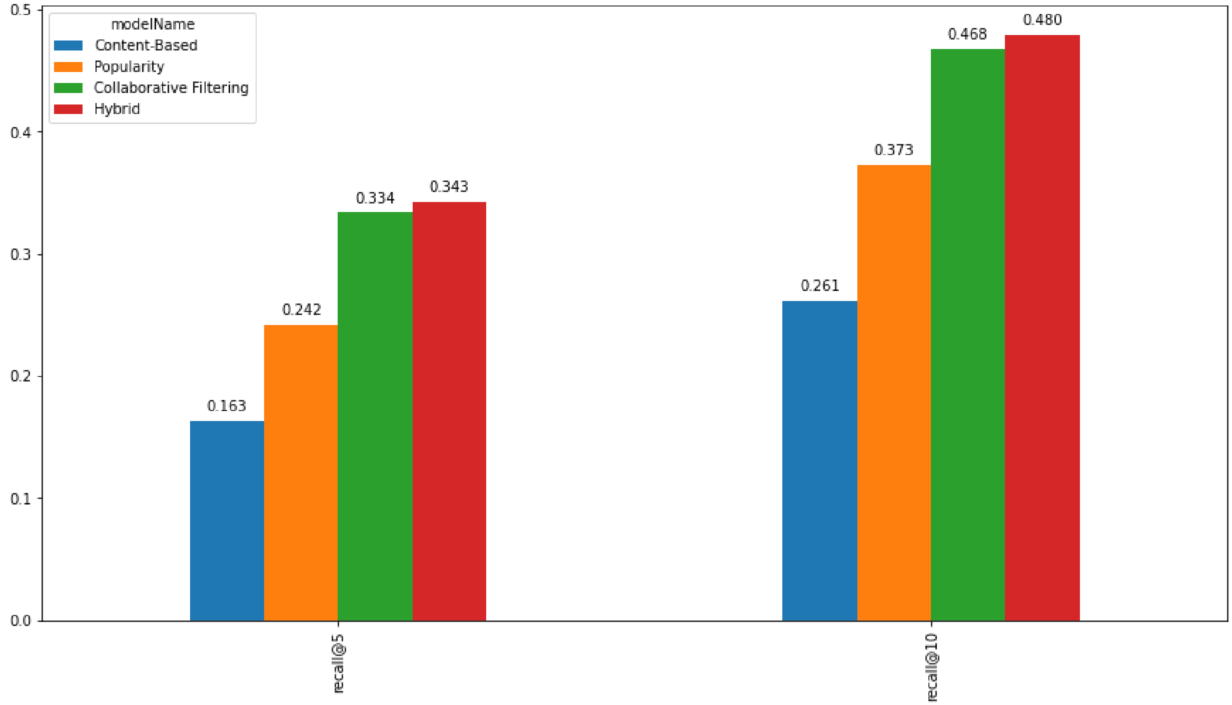
	hits@5_count	hits@10_count	interacted_count	recall@5	recall@10	_person_id
76	22	46	192	0.114583	0.239583	3609194402293569455
17	31	58	134	0.231343	0.432836	-2626634673110551643
16	21	37	130	0.161538	0.284615	-1032019229384696495
10	40	51	117	0.341880	0.435897	-1443636648652872475
82	38	50	88	0.431818	0.568182	-2979881261169775358
161	23	35	80	0.287500	0.437500	-3596626804281480007
65	23	32	73	0.315068	0.438356	1116121227607581999
81	16	21	69	0.231884	0.304348	692689608292948411
106	20	27	69	0.289855	0.391304	-9016528795238256703
52	22	29	68	0.323529	0.426471	3636910968448833585

In [47]:
```
global_metrics_df = pd.DataFrame([cb_global_metrics, pop_global_metrics, cf
                        .set_index('modelName')
global_metrics_df
```

Out[47]:

	recall@5	recall@10
modelName		
Content-Based	0.162874	0.261442
Popularity	0.241882	0.372539
Collaborative Filtering	0.333930	0.468039
Hybrid	0.342623	0.479673

In [48]:
```
%matplotlib inline
ax = global_metrics_df.transpose().plot(kind='bar', figsize=(15,8))
for p in ax.patches:
    ax.annotate("%.3f" % p.get_height(), (p.get_x() + p.get_width() / 2., p
```



In [49]:
```
def inspect_interactions(person_id, test_set=True):
    if test_set:
        interactions_df = interactions_test_indexed_df
    else:
        interactions_df = interactions_train_indexed_df
    return interactions_df.loc[person_id].merge(articles_df, how = 'left',
                                                    left_on = 'contentId'
                                                    right_on = 'contentId
                        .sort_values('eventStrength', ascending = False)[
                                                                         '
                                                                         '
```

In [50]:
```
inspect_interactions(-1479311724257856983, test_set=False).head(20)
```

Out[50]:

	eventStrength	contentId	title	
115	4.285402	7342707578347442862	At eBay, Machine Learning is Driving Innovativ...	https://www.ebayinc.com/stories/news/at-el
38	4.129283	621816023396605502	AI Is Here to Help You Write Emails People Wil...	http://www.wired.com/2016/08/boomerang-us
8	4.044394	-4460374799273064357	Deep Learning for Chatbots, Part 1 - Introduction	http://www.wildml.com/2016/04/deep-learning-
116	3.954196	-7959318068735027467	Auto-scaling scikit-learn with Spark	https://databricks.com/blog/2016/02/08/auto-
10	3.906891	2589533162305407436	6 reasons why I like KeystoneML	http://radar.oreilly.com/2015/07/6-reasons-w
28	3.700440	5258604889412591249	Machine Learning Is No Longer Just for Experts	https://hbr.org/2016/10/machine-learning-is-r
6	3.700440	-398780385766545248	10 Stats About Artificial Intelligence That Wi...	http://www.fool.com/investing/2016/06/19/10-
113	3.643856	-6467708104873171151	5 reasons your employees aren't sharing their ...	http://justcuriousblog.com/2016/04/5-reasons
42	3.523562	-4944551138301474550	Algorithms and architecture for job recommenda...	https://www.oreilly.com/ideas/algorithms-and
43	3.459432	-8377626164558006982	Bad Writing Is Destroying Your Company's Produ...	https://hbr.org/2016/09/bad-writing-is-destr
41	3.459432	444378495316508239	How to choose algorithms for Microsoft Azure M...	https://azure.microsoft.com/en-us/documenta
3	3.321928	2468005329717107277	How Netflix does A/B Testing - uxdesign.cc - U...	https://uxdesign.cc/how-netflix-does-a-b-te

	eventStrength	contentId	title	
101	3.321928	-8085935119790093311	Graph Capabilities with the Elastic Stack	https://www.elastic.co/webinars/sneak-peek-c
107	3.169925	-1429167743746492970	Building with Watson Technical Web Series	https://w 304.ibm.com/partnerworld/wps/se
16	3.169925	6340108943344143104	Text summarization with TensorFlow	https://research.googleblog.com/2016/08/text
49	3.169925	1525777409079968377	Probabilistic Programming	http://probabilistic-programming.org/wiki/H
44	3.169925	-5756697018315640725	Being A Developer After 40 - Free Code Camp	https://medium.freecodecamp.com/bein deve
97	3.087463	2623290164732957912	Creative Applications of Deep Learning with Te...	https://www.kadenze.com/courses/creative-ap
32	3.000000	279771472506428952	5 Unique Features Of Google Compute Engine Tha...	http://www.forbes.com/sites/janakirammsv/201
78	2.906891	-3920124114454832425	Worldwide Ops in Minutes with DataStax & Cloud	http://www.datastax.com/2016/01/datastax-en

In [51]:
```
hybrid_recommender_model.recommend_items(-1479311724257856983, topn=20, ver
```

Out[51]:

	recStrengthHybrid	contentId	title	
0	25.436876	3269302169678465882	The barbell effect of machine learning.	http://techcrunch.com/2016/06/02/the-barl
1	25.369932	-8085935119790093311	Graph Capabilities with the Elastic Stack	https://www.elastic.co/webinars/sneak-pee
2	24.493428	1005751836898964351	Seria Stranger Things uma obra de arte do algo...	https://www.linkedin.com/pulse/seria-stra
3	24.382997	-8377626164558006982	Bad Writing Is Destroying Your Company's Produ...	https://hbr.org/2016/09/bad-writing-is-de
4	24.362064	-6727357771678896471	This Super Accurate Portrait Selection Tech Us...	http://petapixel.com/2016/06/29/super-acc
5	24.190327	-8190931845319543363	Machine Learning Is At The Very Peak Of Its Hy...	https://arc.applause.com/2016/08/17/gartr
6	24.172285	7395435905985567130	The AI business landscape	https://www.oreilly.com/ideas/the-ai-busi
7	23.932289	5092635400707338872	Power to the People: How One Unknown Group of ...	https://medium.com/@atduskgreg/power-
8	23.865716	-5253644367331262405	Hello, TensorFlow!	https://www.oreilly.com/learning/hello-ten
9	23.811519	1549650080907932816	Spark comparison: AWS vs. GCP	https://www.oreilly.com/ideas/spark-compa
10	23.537832	621816023396605502	AI Is Here to Help You Write Emails People Wil...	http://www.wired.com/2016/08/boomerang
11	23.195716	-1901742495252324928	Designing smart notifications	https://medium.com/@intercom/designing-
12	23.101347	882422233694040097	Infográfico: Algoritmos para Aprendizado de Má...	https://www.infoq.com/br/news/2016/07/inf

	recStrengthHybrid	contentId	title	
13	22.725769	2468005329717107277	How Netflix does A/B Testing - uxdesign.cc - U...	https://uxdesign.cc/how-netflix-does-a-b
14	22.561032	-5756697018315640725	Being A Developer After 40 - Free Code Camp	https://medium.freecodecamp.com/b d
15	22.448418	-4944551138301474550	Algorithms and architecture for job recommenda...	https://www.oreilly.com/ideas/algorithms-a
16	22.342822	1415230502586719648	Machine Learning Is Redefining The Enterprise ...	http://www.forbes.com/sites/louiscolumbus/
17	22.311658	-8771338872124599367	Funcionários do mês no CoolHow: os Slackbots -...	https://medium.com/coolhow-creative-lab/fu
18	22.278853	5258604889412591249	Machine Learning Is No Longer Just for Experts	https://hbr.org/2016/10/machine-learning-
19	22.239822	-5027816744653977347	Apple acquires Turi, a machine learning company	https://techcrunch.com/2016/08/05/apple-a

In [ ]:

```
```

In [ ]:

```
```