Patent Recommendation System

Team #3

- Ruchika Hazariwal
- Prachi Chouksey
- Pavani Somarouthu
- Rachana Bumb

Recap

- Overview
- Dataset
- Data Analysis
- Data Preprocess

Continue

- System Architecture
- Recommender system approaches
- Results
- Demo

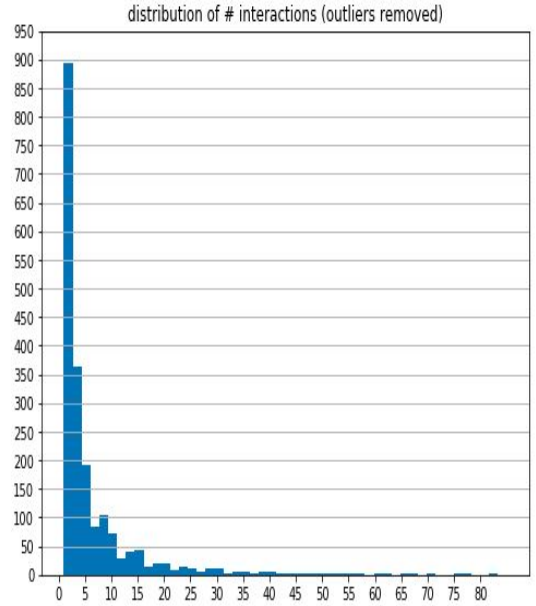
Introduction

The goal of project is to build a patent recommendation system for user.

Patent search is increasingly important to companies and users. An efficient recommender system will provide relevant results to user based on user history/keyword match/category match etc.

User based Collaborative Recommendation System

1- Exploratory data analysis: distribution of how many patents a user interacts with.
Explore and remove duplicate patents.
Most viewed patent id
Rank based- return n-top patents
User based collaborative filtering: user-patent matrix.
Because the results for each user here are binary, it (perhaps) makes sense to compute similarity as the dot product of two users.
Most similar users
Recommend patents



In [6]:
```
user_interaction_counts.describe()
```

Out[6]:
```
count    1993.000000
mean        6.464626
std        11.040400
min         1.000000
25%         1.000000
50%         3.000000
75%         7.000000
max       214.000000
Name: title, dtype: float64
```

Out[44]:
```
1330.0    299
1429.0    287
1314.0    215
1431.0    206
1364.0    198
Name: patent_id, dtype: int64
```

In [48]:
```
print(get_top_patents(10))
print(get_top_patent_ids(10))
```

```
['Method and apparatus for managing utilization of wireless resources via use of a reference signal to reduce distort
ion'
 'Access control method and apparatus for use in mobile communication'
 'Antenna device and electronic device including the same'
 'Rotation variant object detection in Deep Learning'
 'Sensing total current of distributed load circuits independent of current distribution using distributed voltage av
eraging'
 'Disabling onboard input devices in an autonomous vehicle'
 'Passive radiator assembly'
 'Monitoring and controlling the movement of mobile robots'
 'Message processing' 'Method and apparatus for promotional programming']
['1330.0', '1429.0', '1314.0', '1431.0', '1364.0', '1398.0', '1170.0', '1293.0', '1436.0', '1351.0']
```

The 10 most similar users to user 1 are: [23, 290, 204, 64, 395, 131, 334, 21, 203, 665]
The 3 most similar users to user 46 are: [23, 21, 98]

In [61]:
```
# Check Results
get_patents_names(user_user_recs(1, 10)) # Return 10 recommendations for user 1
```

```
['43.0', '109.0', '151.0', '310.0', '329.0', '585.0', '732.0', '1052.0', '1232.0', '1293.0', '1305.0', '1406.0', '142
9.0', '1430.0']
```

Out[61]:
```
['Method and apparatus for managing utilization of wireless resources via use of a reference signal to reduce distort
ion',
 'Safe illumination for computerized facial recognition',
 'Data storage system and data storage method',
 'Imaging device having pan/tilt control for object tracking, imaging method, and computer-readable medium',
 'Speaker position identification with respect to a user based on timing information for enhanced sound adjustment',
 'Multi-part navigation process by an unmanned aerial vehicle for navigation',
 'Methods, systems, and devices for obscuring entities depicted in captured images',
 'High-bandwidth underwater data communication system',
 'Beat alignment and selection for cardiac mapping',
 'Method and apparatus for route selection based on recorded and calculated routes']
```

Out[31]:

	neighbor_id	similarity	num_interactions
0	23	7	214
2	204	6	61
1	290	6	60
5	131	5	93
3	64	5	37

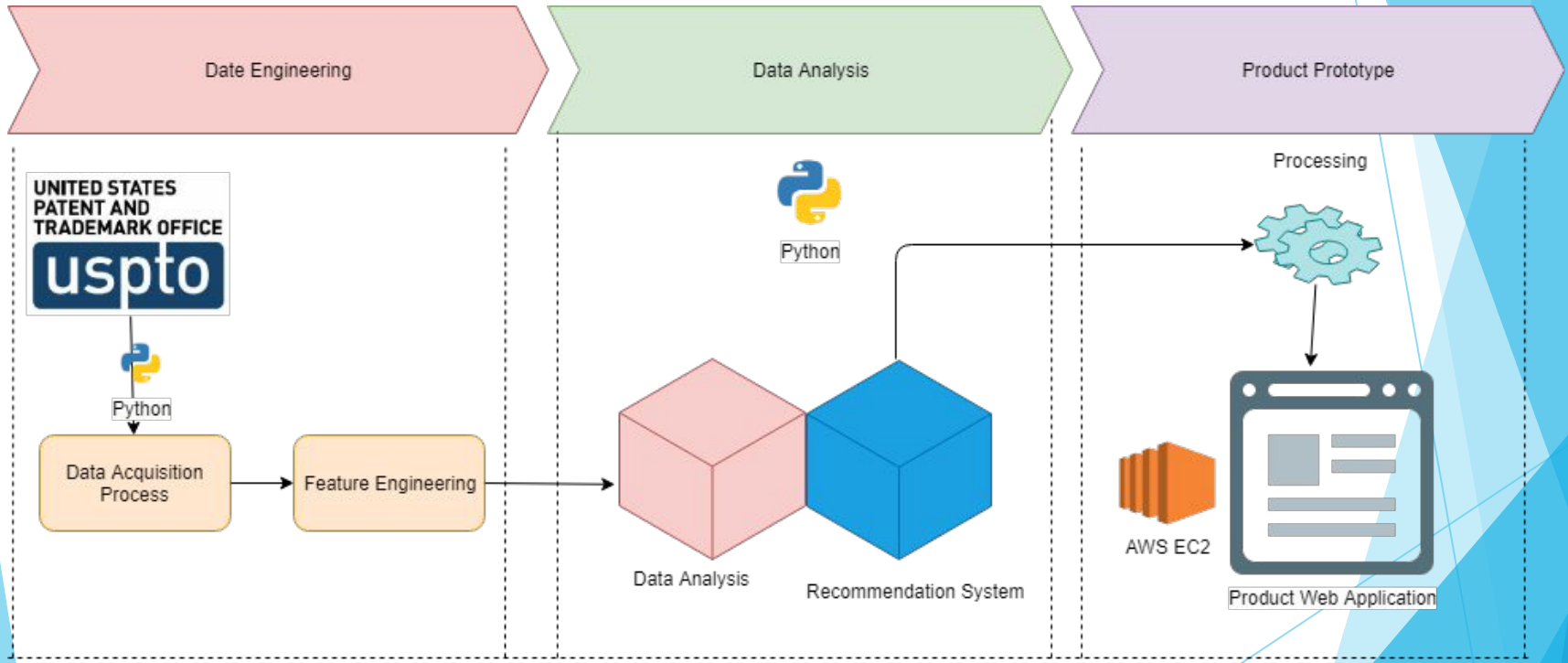
The top 10 recommendations for user 20 are the following patent ids:
['1314.0', '124.0', '1172.0', '1162.0', '1393.0', '1160.0', '1274.0', '14.0', '130.0', '116.0']

The top 10 recommendations for user 20 are the following patent names:
['Method and apparatus for managing utilization of wireless resources via use of a reference signal to reduce distort ion', 'Data storage system and data storage method', 'Multi-part navigation process by an unmanned aerial vehicle for navigation', 'Methods, systems, and devices for obscuring entities depicted in captured images', 'Methods for agronom ic and agricultural monitoring using unmanned aerial systems', 'Set up of direct mapped routers located across indepe ndently managed compute and storage networks', 'In-call command control', 'Over the air acquisition of radio frequenc y impairment information', 'Text processing method, system and computer program', 'System and method for streaming au dio of a visual feed']

7. Using your existing functions, provide the top 10 recommended patent you would provide for the a new user below. You can test your function against our thoughts to make sure we are all on the same page with how we might make a recommendation.

Improved Solution

System Architecture



Data Acquisition Process:

Our recommender system will follow prescriptive data analytics which involves high volume of data and advanced/complex analytical techniques to make correct recommendation.

```
base_url = 'http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&u=%2Fneta
url = base_url + searchstring + '&d=PTXT'
print(url)
r = requests.get(url, headers=Constants.request_header).text
s = BeautifulSoup(r, 'html.parser')
total_results = int(s.find(string=re.compile('out of')).find_next().text.strip())
patents = self.get_patents_from_results_url(url, limit=results_limit)
num_results_fetched = len(patents)
list_num = 2
base_url_nextpgs = 'http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&u
url_pre = base_url_nextpgs + '&OS=' + searchstring + '&RS=' + searchstring + '&Query='
url_post = '=Next+50+Hits'
while (num_results_fetched < total_results) and (num_results_fetched < results_limit):
    this_url = url_pre + str(list_num) + url_post
    thispatents = self.get_patents_from_results_url(this_url)
    patents.extend(thispatents)
    num_results_fetched = len(patents)
    if num_results_fetched >= results_limit:
        patents = patents[:results_limit]
    list_num += 1
self.patents = patents
```

Data Preprocess

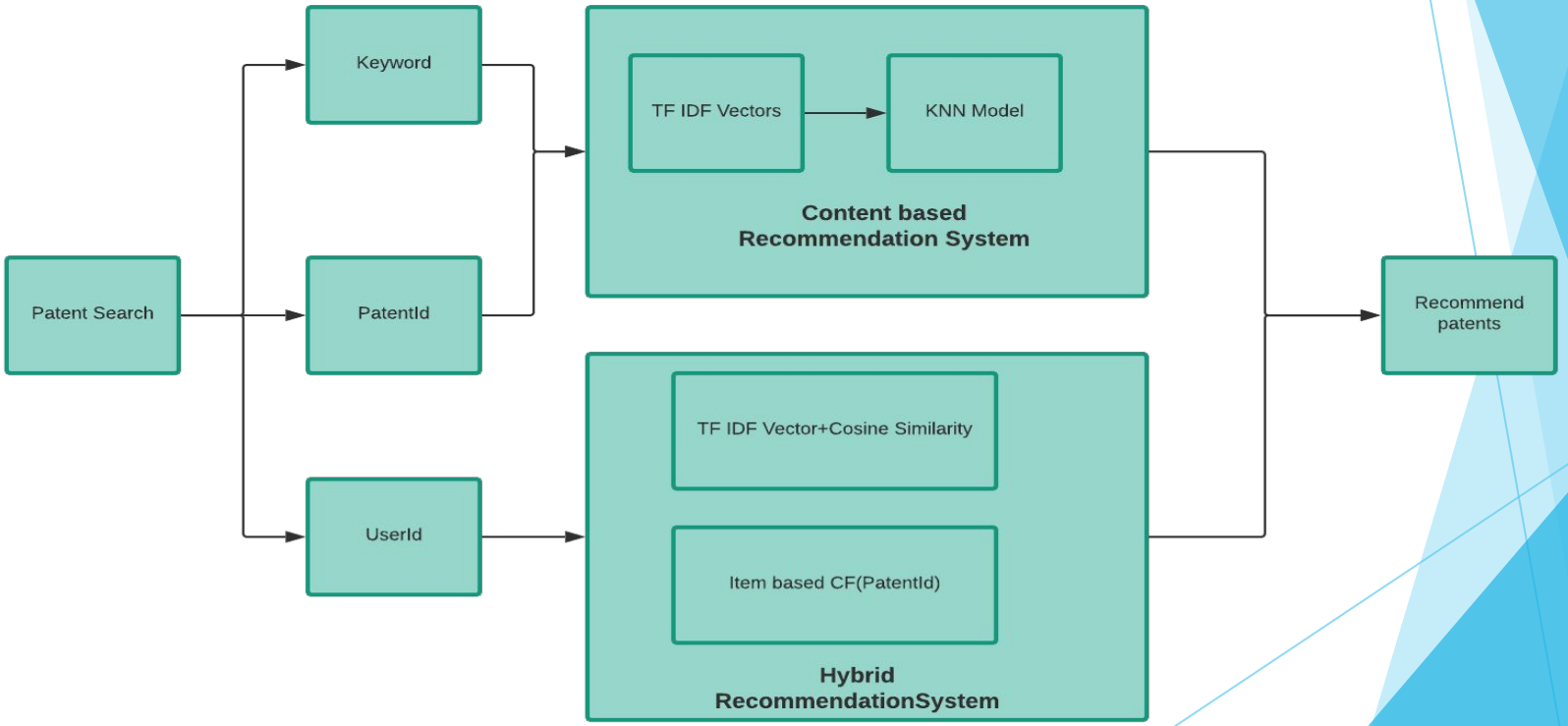
- **Redundancy-** We make sure redundant data is not present in our dataset while web scraping unique patents from the website.
- **Missing Values-** Few documents have missing abstracts, we fill in title in the abstract attribute to make sure data is complete.
- **Attribute Selection-** We have selected the following columns after preprocessing of data.
- **Stemming-** Generating variants of root/base words.

```
for item in patentlist['abstract_org']:
    text=[]
    strtext=''
    for word in item.lower().split():
        word=word.replace(",", "").replace(".", "").replace("(", "").replace(")", "")
        if word not in cachedStopWords:
            word = ps.stem(word)
            text.append(word)
        strtext=' '.join(text)
    abstract.append(strtext)
patentlist['abstract_org']=abstract
freq_count = []
for item in patentlist['abstract_org']:
    count = Counter(str(item).split())
    freq_count.append(count)
patentlist['word_count'] = freq_count
```

Dataset:

	A	B	C	D	E
1	patent_nur	abstract	cpc	file_date	url
2	10349422	In one embodiment, a ser	['H04W', 'H04B', 'H04L'	March 10, 2017	http://patft.uspto.go
3	10349418	Aspects of the subject disc	['H04W', 'H04L']	June 8, 2016	http://patft.uspto.go
4	10349318	The present disclosure rela	['H04W']	July 13, 2017	http://patft.uspto.go
5	10349295	Communication network a	['H04W', 'H04L']	July 7, 2017	http://patft.uspto.go
6	10349285	Wireless devices may be e	['H04W']	August 8, 2017	http://patft.uspto.go
7	10349227	Various systems and meth	['H04W']	November 2, 20	http://patft.uspto.go
8	10349218	Systems and methods for	['H04W']	March 6, 2018	http://patft.uspto.go
9	10349197	According to an aspect of	['H04S', 'G10L']	August 13, 2015	http://patft.uspto.go
10	10348985	A turbulence-free CCD can	['H04N', 'G01N', 'G02B'	September 20, 2	http://patft.uspto.go
11	10348514	A lighting system includes	['H04L', 'G05B', 'H04W'	October 26, 201	http://patft.uspto.go
12	10348394	A wireless communication	['H04B', 'H04W', 'H04L'	December 14, 2(	http://patft.uspto.go

Functional flow



Content based Recommendation (Keyword/Patent Id search)

- Step 1: Generate TF-IDF vectors and dump in a pickle file

```
def generateTFIDFMatrix():
    patentlist['abstract_org']=patentlist['abstract']
    abstract=[]
    for item in patentlist['abstract_org']:
        text=[]
        strtext=''
        for word in item.lower().split():
            word=word.replace(",", "").replace(".", "").replace("(", "").replace(")", "")
            if word not in cachedStopWords:
                word = ps.stem(word)
                text.append(word)
            strtext=' '.join(text)
        abstract.append(strtext)
    patentlist['abstract_org']=abstract
    freq_count = []
    for item in patentlist['abstract_org']:
        count = Counter(str(item).split())
        freq_count.append(count)
    patentlist['word_count'] = freq_count
    tfidfVect = TfidfVectorizer()
    tfidf = tfidfVect.fit_transform(patentlist['abstract_org'])
    pickle.dump(tfidf, open("tfidf_fit_transform.pickle", "wb"),protocol=2)
```

Content based Recommendation (Keyword/Patent Id search)

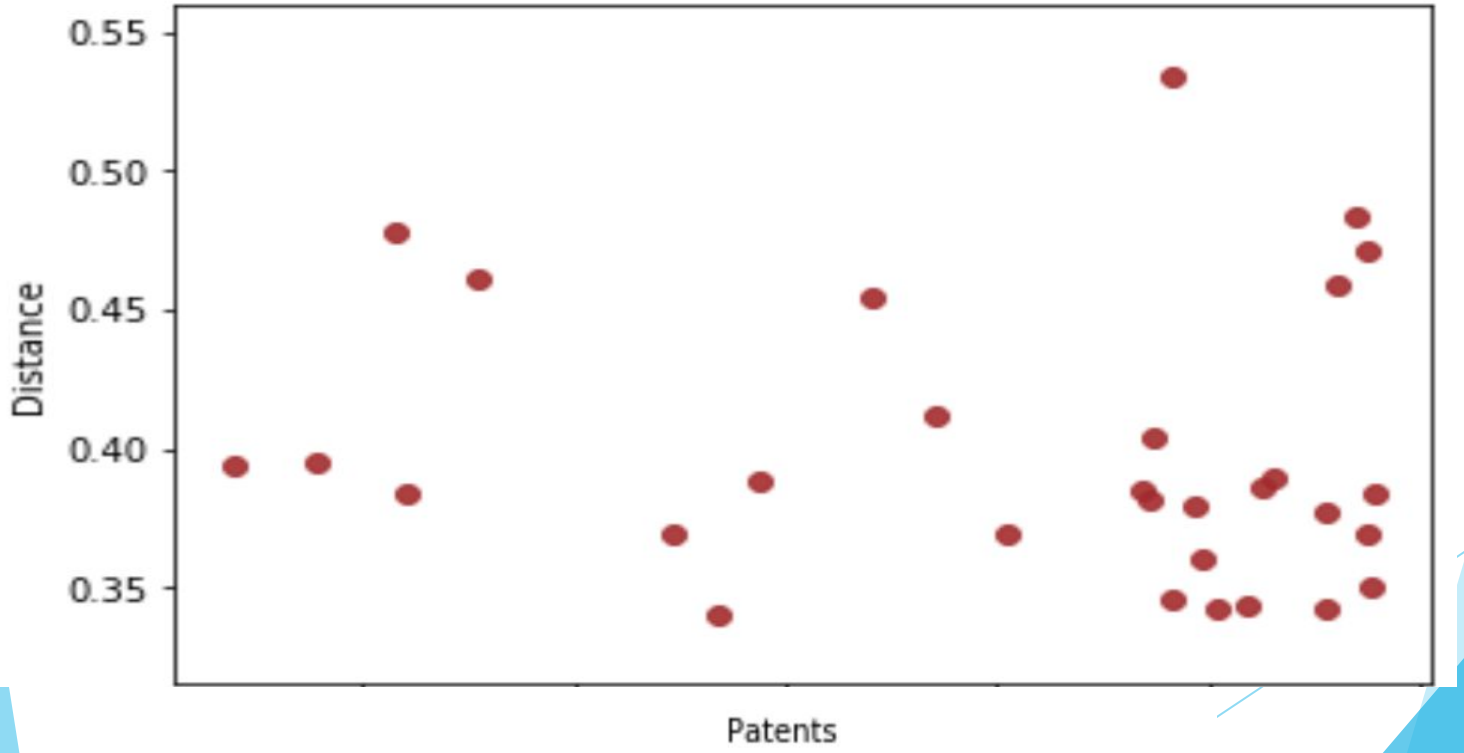
- Step 2: Fetch keyword/patent id and generate fit transform vectors

```
def generateFitVector():
    patent1_tfidfVect = TfidfVectorizer()
    patent1_tfidfVect = patent1_tfidfVect.fit(patentlist['abstract_org'])
    pickle.dump(patent1_tfidfVect, open("tfidf_fit.pickle", "wb"),protocol=2)
```

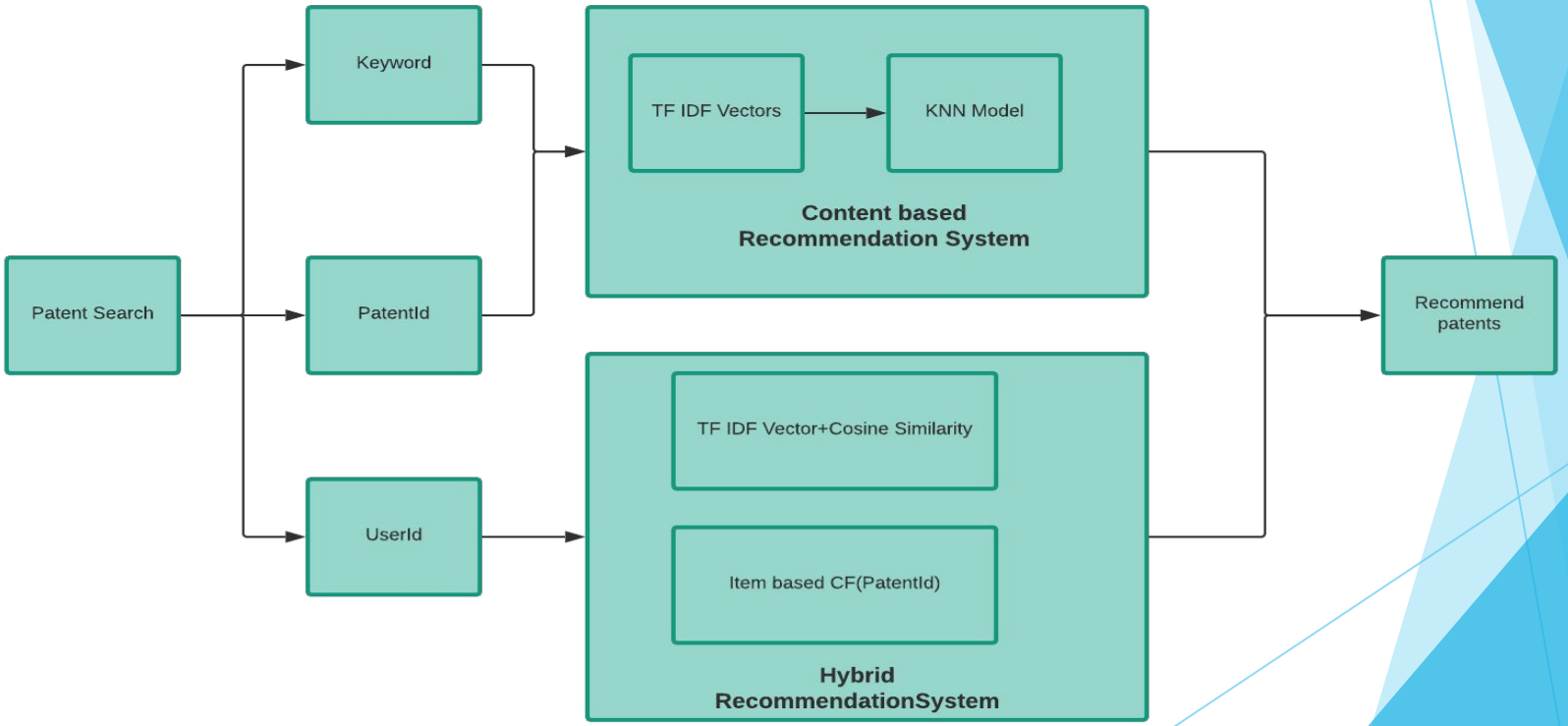
- Step 3: Feed to KNN and send dataframe with required fields in response

```
def patentKeywordMatch(keyword):
    search=[]
    text=[]
    strtext=[]
    for word in keyword.lower().split():
            word=word.replace(",", "").replace(".", "").replace("(", "").replace(")", "")
            if word not in cachedStopWords:
                word = ps.stem(word)
                text.append(word)
            strtext=' '.join(text)
            search.append(strtext)
    patent1 =search
    patent1_tfidfVect = pickle.load(open("tfidf_fit.pickle", "rb"))
    patent1_tfidf = patent1_tfidfVect.transform(patent1)
    new_tfidf=pickle.load(open("tfidf_fit_transform.pickle", "rb"))
    nbrs = NearestNeighbors(n_neighbors=5).fit(new_tfidf)
    distances, indices = nbrs.kneighbors(patent1_tfidf)
    names_similar = pd.Series(indices.flatten()).map(patentlist.reset_index()['patent_num'])
    abstract_similar = pd.Series(indices.flatten()).map(patentlist.reset_index()['abstract'])
    url_similar = pd.Series(indices.flatten()).map(patentlist.reset_index()['url'])
    title_similar = pd.Series(indices.flatten()).map(patentlist.reset_index()['title'])
    result = pd.DataFrame({'distance':distances.flatten(), 'patent_num':names_similar, 'abstract':abstrac
                           'title':title_similar})
    result=result.sort_values('distance')
    result.to_csv('result_keyword.csv')
    return result
```

KNN Plot: Distance vs Patents



Functional flow



Item based Collaborative Filtering (User Id search)

Step 1: Create similarity matrix for all the patents.

```
def createTfIdfSimMatrix():
    patentlist = pd.read_csv('../data/Dataset1.csv')
    patentlist['abstract_org']=patentlist['abstract']
    abstract=[]
    for item in patentlist['abstract_org']:
        text=[]
        strtext=''
        for word in item.lower().split():
            word=word.replace(",", "").replace(".", "").replace("(", "").replace(")", "")
            if word not in cachedStopWords:
                word = ps.stem(word)
                text.append(word)
            strtext=' '.join(text)
        abstract.append(strtext)
    patentlist['abstract_org']=abstract
    tf_idf = TfidfVectorizer().fit_transform(patentlist['abstract_org'])
    print(tf_idf.shape[1])
    cosine = linear_kernel(tf_idf, tf_idf)
    df=pd.DataFrame(cosine,columns=patentlist["patent_num"], index=patentlist["patent_num"])
    df.to_csv("../data/tf_idf.csv")
    print(cosine)
```

Similarity Matrix

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1		10349422	10349418	10349318	10349295	10349285	10349227	10349218	10349197	10348985	10348514	10348394	10348341	10348329	10348108	10347985	10347579	10347565	10347344	10347329	10347290
2	10349422	1	0.20611042	0.13420136	0.16896043	0.11372347	0.11297744	0.08209348	0.0303789	0.00422363	0.02082916	0.18004246	0.05114671	0.01348129	0.01054402	0.04733973	0.00675404	0.02520546	0.01404876	0.01172937	0.04270574
3	10349418	0.20611042	1	0.11721271	0.17904791	0.15421714	0.14600898	0.12422746	0.17871952	0.00762415	0.02299453	0.18209461	0.08667758	0.02958937	0.08096596	0.08115517	0.01681503	0.17961375	0.02313877	0.02639476	0.02433532
4	10349318	0.13420136	0.11721271	1	0.12835952	0.12243282	0.08640746	0.13524956	0.03788909	0.02231292	0.01085263	0.04396496	0.05770066	0.05258314	0.00230623	0.01967352	0.00583325	0.02356336	0.01741118	0.01147762	0.03159763
5	10349295	0.16896043	0.17904791	0.12835952	1	0.06939229	0.08706631	0.17525095	0.01156507	0.0095217	0.02472869	0.26652819	0.04818038	0.0331501	0.02254918	0.02957931	0.01437011	0	0.03370668	0.0209357	0.03964243
6	10349285	0.11372347	0.15421714	0.12243282	0.06939229	1	0.09025328	0.1762558	0.03391568	0.01582374	0.02867733	0.0939894	0.03991006	0.03187945	0.01621297	0.03312049	0.00830385	0.02942052	0.02902309	0.04104166	0.04494421
7	10349227	0.11297744	0.14600898	0.08640746	0.08706631	0.09025328	1	0.06254838	0.05991076	0.01405549	0.07853336	0.11317867	0.07538113	0	0.13690744	0.06403481	0.01286155	0.00159933	0.01495141	0.03565626	0.07605407
8	10349218	0.08209348	0.12422746	0.13524956	0.17525095	0.1762558	0.06254838	1	0.02341759	0.00599154	0.00918427	0.05958797	0.06008215	0.01145911	0.12163681	0.00556273	0.01779946	0.07576264	0.01438136	0.0292924	0.0209192
9	10349197	0.0303789	0.17871952	0.03788909	0.01156507	0.03391568	0.05991076	0.02341759	1	0.00124897	0.00661131	0.01508077	0.06931527	0.00925394	0.16854591	0.01892586	0.01667943	0.00783484	0.00748817	0.00530781	0.01630215
10	10348985	0.00422363	0.00762415	0.02231292	0.0095217	0.01582374	0.01405549	0.00599154	0.00124897	1	0.12078328	0.01365149	0.0045989	0.00866153	0.00574075	0.00204793	0.01747805	0.00854662	0.0082204	0	0.0013351
11	10348514	0.02082916	0.02299453	0.01085263	0.02472869	0.02867733	0.07853336	0.00918427	0.00661131	0.12078328	1	0.02468624	0.05387367	0	0.03049589	0.03223664	0.03706067	0.00350058	0.01639975	0.02128688	0.04943545
12	10348394	0.18004246	0.18209461	0.04396496	0.26652819	0.0939894	0.11317867	0.05958797	0.01508077	0.01365149	0.02468624	1	0.0225564	0.00466243	0.00599671	0.10293621	0.03386388	0.01836188	0.02169711	0.0234121	0.02859425
13	10348341	0.05114671	0.08667758	0.05770066	0.04818038	0.03991006	0.07538113	0.06008215	0.06931527	0.0045989	0.05387367	0.0225564	1	0.0086878	0.02206972	0.01803727	0.00910571	0.03241788	0.0298872	0.04409845	0.02811958
14	10348329	0.01348129	0.02958937	0.05258314	0.0331501	0.03187945	0	0.01145911	0.00925394	0.00866153	0	0.00466243	0.0086878	1	0.00953297	0	0.0117466	0	0.03037885	0	0.00242653
15	10348108	0.01054402	0.08096596	0.00230623	0.02254918	0.01621297	0.13690744	0.12163681	0.16854591	0.00574075	0.03049589	0.00599671	0.02206972	0.00953297	1	0.0657661	0.00867746	0.11211251	0.00587565	0.01683799	0.03290464
16	10347985	0.04733973	0.08115517	0.01967352	0.02957931	0.03312049	0.06403481	0.00556273	0.01892586	0.00204793	0.03223664	0.10293621	0.01803727	0	0.0657661	1	0.12793784	0.02600836	0.00871658	0.01255097	0.02512806
17	10347579	0.00675404	0.01681503	0.00583325	0.01437011	0.00830385	0.01286155	0.01779946	0.01667943	0.01747805	0.03706067	0.03386388	0.00910571	0.0117466	0.00867746	0.12793784	1	0.01447656	0.0019529	0.0176369	0.00581798
18	10347565	0.02520546	0.17961375	0.02356336	0	0.02942052	0.00159933	0.07576264	0.00783484	0.00854662	0.00350058	0.01836188	0.03241788	0	0.11211251	0.02600836	0.01447656	1	0.01577895	0.00141341	0
19	10347344	0.01404876	0.02313877	0.01741118	0.03370668	0.02902309	0.01495141	0.01438136	0.00748817	0.0082204	0.01639975	0.02169711	0.0298872	0.03037885	0.00587565	0.00871658	0.0019529	0.01577895	1	0.14788842	0.01781601
20	10347329	0.01172937	0.02639476	0.01147762	0.0209357	0.04104166	0.03565626	0.0292924	0.00530781	0	0.02128688	0.0234121	0.04409845	0	0.01683799	0.01255097	0.0176369	0.00141341	0.14788842	1	0.02457856
21	10347290	0.04270574	0.02433532	0.03159763	0.03964243	0.04494421	0.07605407	0.0209192	0.01630215	0.0013351	0.04943545	0.02859425	0.02811958	0.00242653	0.03290464	0.02512806	0.00581798	0	0.01781601	0.02457856	1
22	10347265	0.0205777	0.14824748	0.0136764	0.01023096	0.02158576	0.01152105	0.0187648	0.30496721	0.01007927	0.01673953	0.0153761	0.18862547	0.00464859	0.0096976	0.01278912	0.00446068	0.23663903	0.00683468	0.00320046	0.00627896
23	10347139	0.01307438	0.01289892	0.01240582	0.01928937	0.07044482	0.00724179	0	0.00728541	0.00939737	0.00687649	0.02200794	0.01328375	0.02683578	0.00194143	0.00188258	0.02786016	0	0.0330902	0.00663044	0.00477143
24	10347063	0.02070449	0.01570261	0.02397164	0.00546015	0.01423842	0.02764528	0.0266742	0.01340768	0.00317599	0.01208507	0.00126278	0.03849544	0.00392151	0.01192603	0.01898595	0.00625053	0.0004633	0.01519412	0.02856505	0.01249318
25	10347026	0	0.00974688	0.02566673	0	0.00971276	0.0338277	0	0.01873854	0.08474821	0.06632967	0.01164441	0	0.02319027	0	0.00229109	0.0022795	0.00234502	0.00798496	0.00302422	0.00385917
26	10347007	0.0293877	0.01477173	0.02661293	0.02808504	0.02149615	0.07431017	0.04472225	0.10124495	0.10588735	0.03457298	0.03589472	0.02432972	0.01816431	0.1107952	0.03270913	0.03069069	0.04364835	0.01478142	0.00529376	0.04852209
27	10346958	0.00926839	0.00283108	0.0107438	0.01232694	0.05215606	0.01327857	0.01653624	0.0037172	0.11529602	0.07071882	0.03585555	0.00696977	0	0.01310642	0.00152149	0.00823911	0.0015573	0.02093107	0.01445755	0.00135534
28	10346888	0	0.02233747	0.07329897	0.05145757	0.18637096	0.11565764	0.03204313	0.00546966	0.0049403	0.00435218	0.00413139	0.01200126	0.02346539	0.00257633	0.00249822	0.00290154	0	0.00824797	0.00439938	0.04645631
29	10346794	0.0042526	0.00774589	0.01382488	0.01658444	0.02204425	0.01294138	0.02680625	0.01096104	0.00155049	0.01619688	0.0171686	0.00622959	0.01398237	0.01569446	0.00355816	0.0083198	0	0.03050059	0.02732239	0.0517482
30	10346765	0.01759332	0.03007469	0.12965967	0.03192231	0.24170313	0.24870097	0.04474047	0.0089593	0.01009073	0.01334518	0.04579451	0.03342406	0.02983103	0.01611628	0.00588383	0.00524279	0.00293065	0.03577206	0.05704874	0.00542334
31	10346720	0.00609044	0.00894573	0.00701435	0.02514617	0.00383552	0.02105171	0.01910766	0.00153	0.03418965	0.02443818	0.006589	0.00293693	0.0161962	0	0.00958632	0.0428697	0	0.0170348	0.00489972	0.00791528
32	10346649	0.02729346	0.06770838	0.03188811	0.03480981	0.0374226	0.03865551	0.07814734	0.14329488	0.00622301	0.01014506	0.04561806	0.04120666	0.05034419	0.06505252	0.01034635	0.00158687	0	0.09254542	0.07795427	0.07833113
33	10346624	0.00570329	0.01875116	0.02832185	0.02603392	0.01367073	0.05106401	0.01327453	0.02999517	0.055806	0.08068889	0.01336585	0.06611113	0.01299922	0.01860145	0.02297618	0.01561781	0.00285389	0.00817343	0.0215921	0.0099481
34	10346617	0.02453976	0.01377475	0.0352609	0.00783193	0.03643179	0.02125904	0.01656674	0.01145822	0.00644435	0.0096211	0.05286468	0.01877346	0.01793129	0.01281816	0.01245345	0.06043835	0	0.018258	0.04344426	0.02136027

Item based Collaborative Filtering (User Id search)

Step 2: Check whether user exists or not. If user exists, get the similar patents.

```
def patentUserIdMatch(id):
    df= pd.read_excel('../data/Item_based_patents.xlsx')
    df =df.dropna()
    df
    user_list = df["UID"].tolist()
    user_list
    if id in user_list:
        print("USERID is found")
        k = get_patentnumbers(id,df).values.tolist()
        res = get_similarity_matrix(k)
        return res
    else:
        print("USERID not there")
```

Item based Collaborative Filtering (User Id search)

Step 3: Find the similar patents and get their related details.

```
def get_similarity_matrix(k):
    numbers = k[0]
    for i in range(0, len(numbers)):
        numbers[i] = int(numbers[i])
    print(numbers)
    data= pd.read_csv('../data/tf_idf.csv')
    data.rename( columns={'Unnamed: 0':'patent_no'}, inplace=True)
    data.set_index('patent_no', inplace=True)
    results = list(data.columns.values)
    pp= data.loc[numbers]
    pp= pp.dropna()
    bigdata1 = pd.concat([patent1, patent2,patent3,patent4,patent5,patent6, patent7,
    bigdata=bigdata1[['patent_num','abstract','similarity','title','url']]
    bigdata.to_csv('result_user.csv')
    return bigdata
```

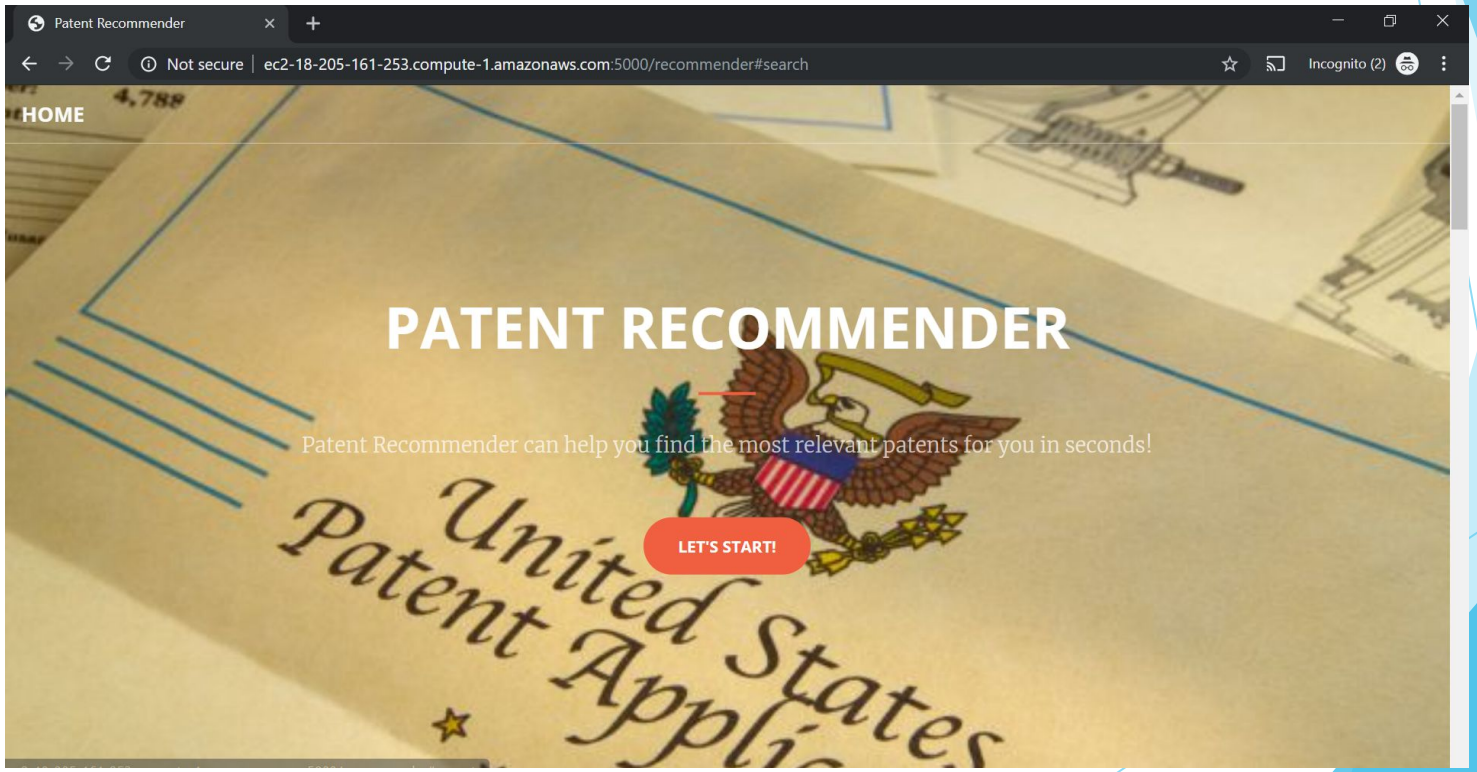
Item based Collaborative Filtering (User Id search)

Step 4: Find the details of read patents

```
def getReadPatents(id):
    df= pd.read_excel('../data/Item_based_patents.xlsx')
    df =df.dropna()
    k = get_patentnumbers(id,df).values.tolist()
    patentno=k[0]
    print(patentno)
    patentlist = pd.read_csv('../data/Dataset1.csv')
    df=pd.DataFrame()
    for i in range(len(patentno)):
        patent=patentlist[patentlist['patent_num'] == str(patentno[i])]
        df = df.append(pd.DataFrame(patent), ignore_index=True)
    return df[["patent_num","abstract","title","url"]]
```

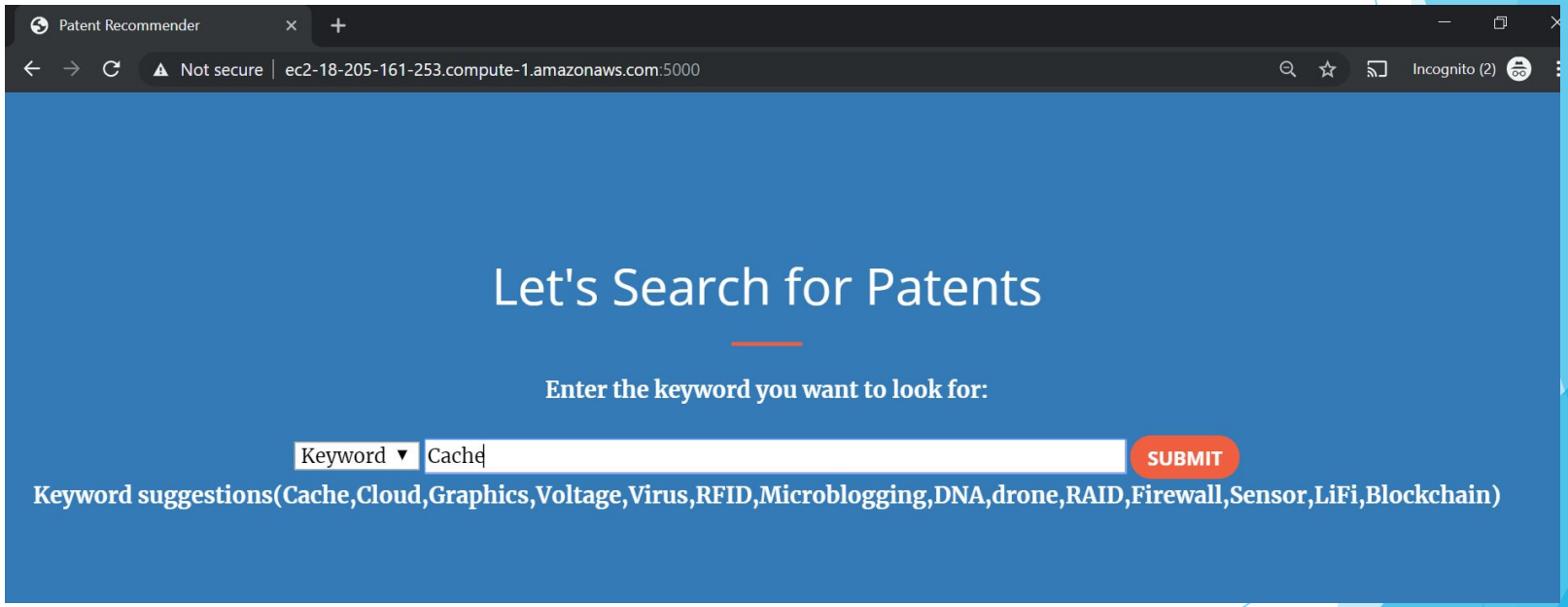
Our System Prototype

http://ec2-18-205-161-253.compute-1.amazonaws.com:5000/recommender#search



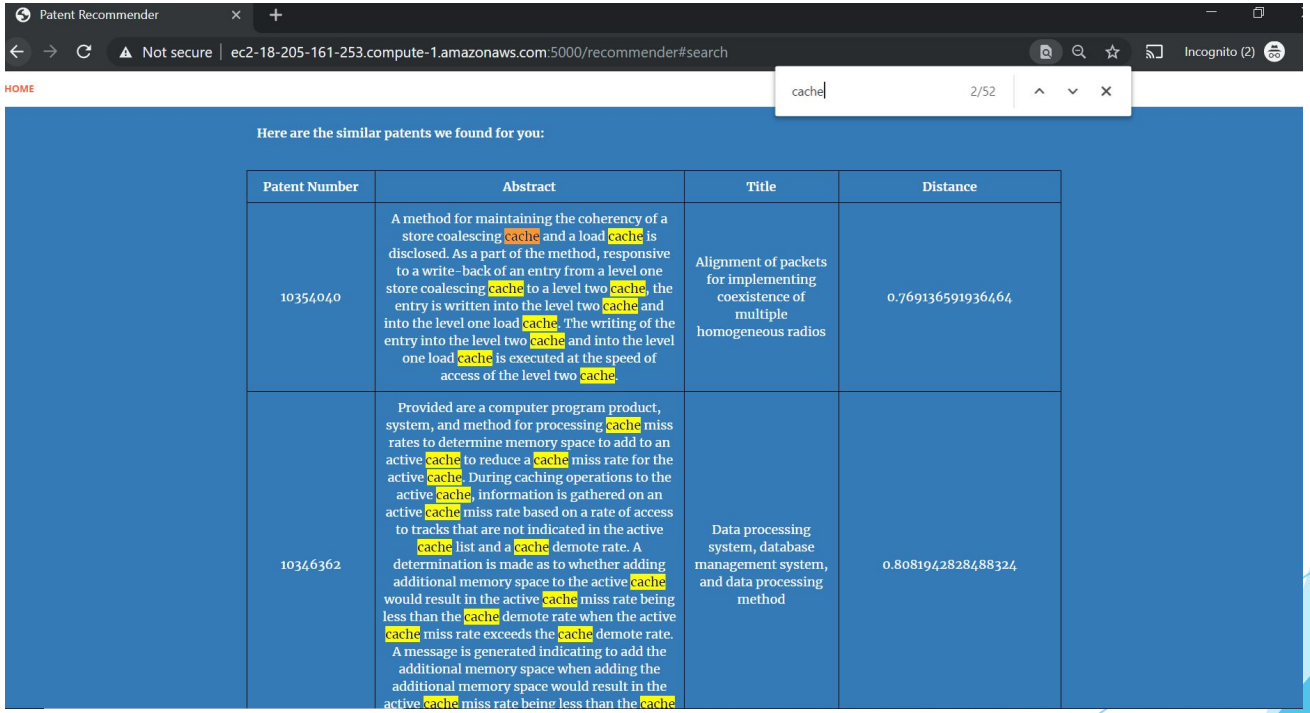
Our System Prototype

Input :Keyword



Our System Prototype

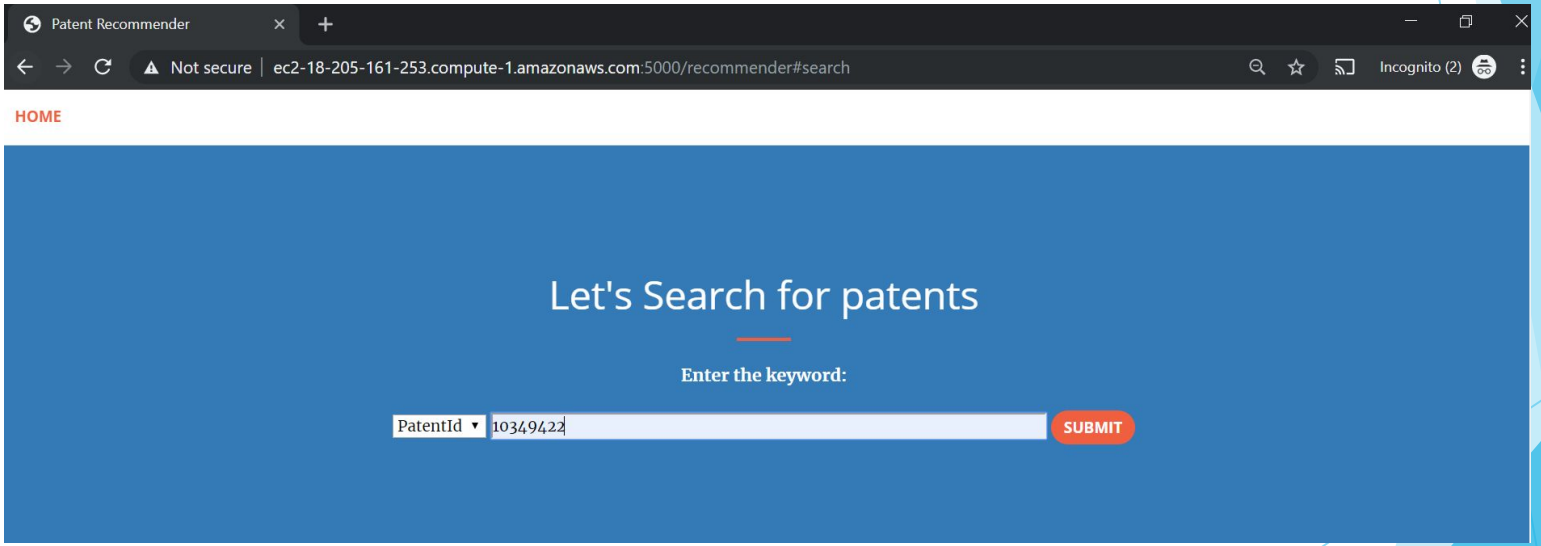
Output: Patent Recommendations



Patent Number	Abstract	Title	Distance
10354040	A method for maintaining the coherency of a store coalescing cache and a load cache is disclosed. As a part of the method, responsive to a write-back of an entry from a level one store coalescing cache to a level two cache, the entry is written into the level two cache and into the level one load cache. The writing of the entry into the level two cache and into the level one load cache is executed at the speed of access of the level two cache.	Alignment of packets for implementing coexistence of multiple homogeneous radios	0.7691365919364644
10346362	Provided are a computer program product, system, and method for processing cache miss rates to determine memory space to add to an active cache to reduce a cache miss rate for the active cache. During caching operations to the active cache, information is gathered on an active cache miss rate based on a rate of access to tracks that are not indicated in the active cache list and a cache demote rate. A determination is made as to whether adding additional memory space to the active cache would result in the active cache miss rate being less than the cache demote rate when the active cache miss rate exceeds the cache demote rate. A message is generated indicating to add the additional memory space when adding the additional memory space would result in the active cache miss rate being less than the cache	Data processing system, database management system, and data processing method	0.8081942828488324

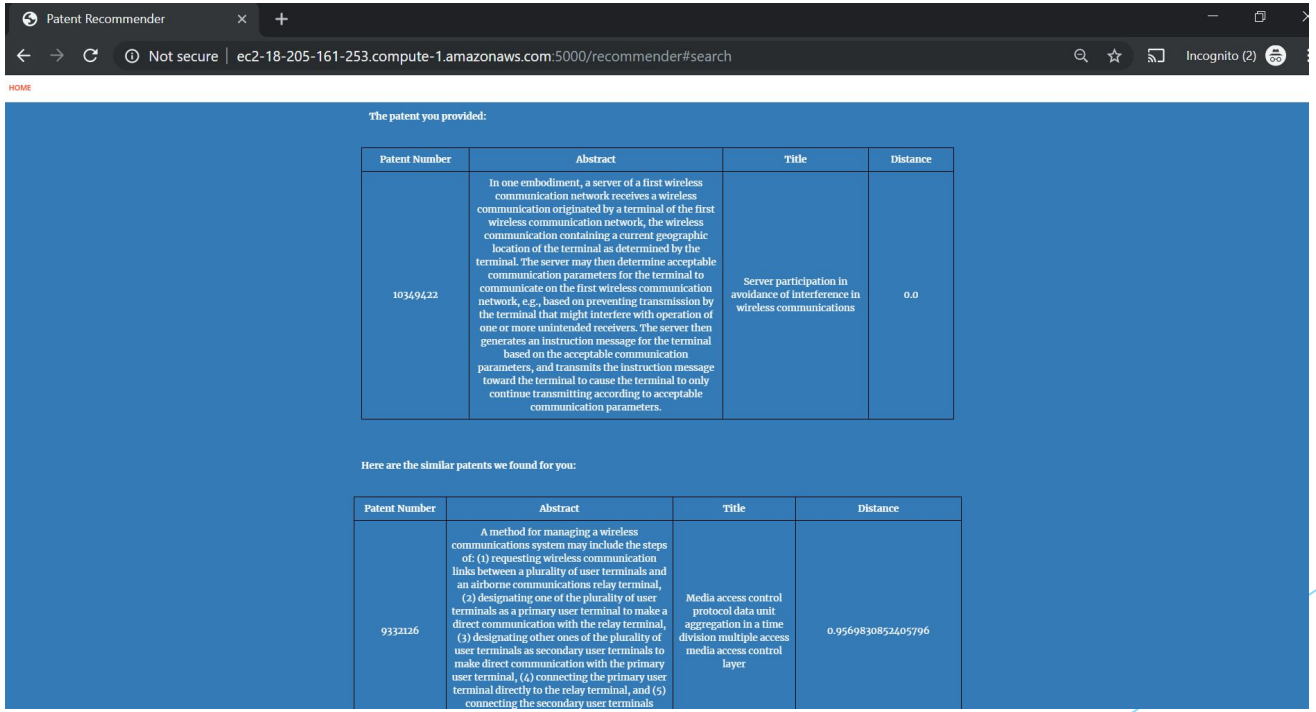
Our System Prototype

Input :PatentId



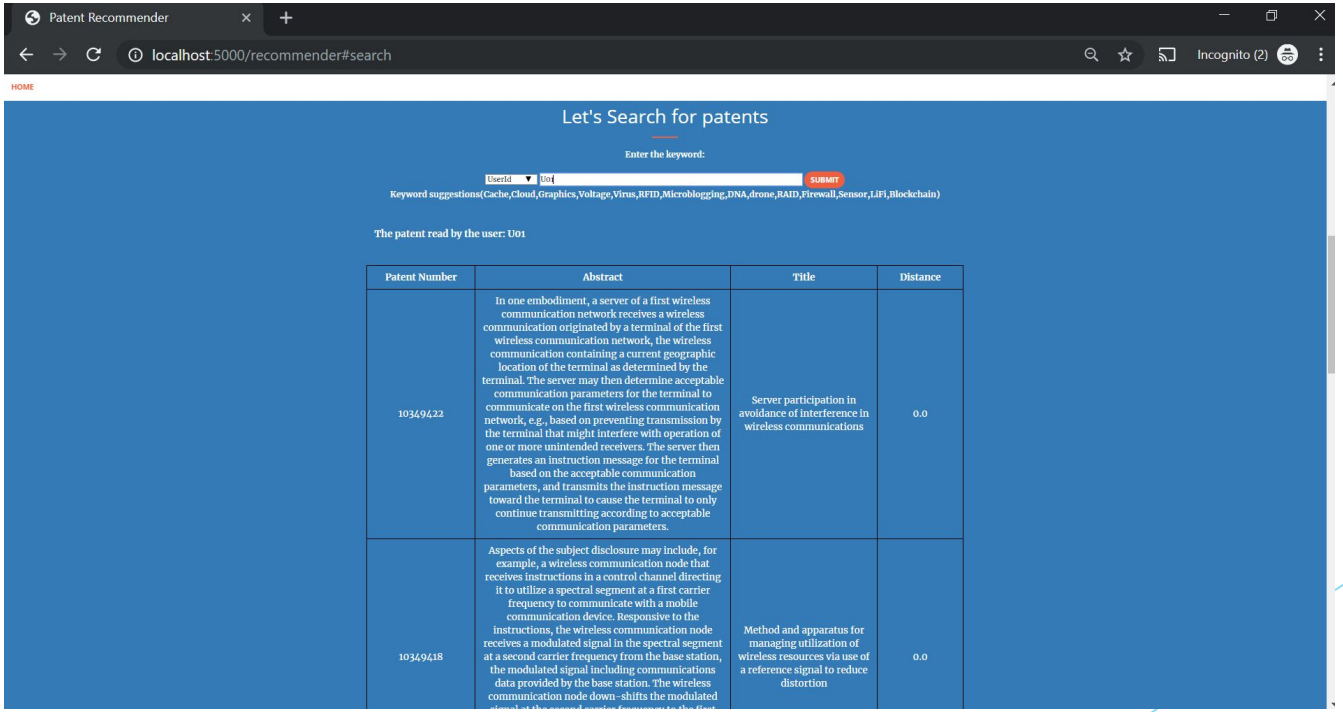
Our System Prototype

Output: Patent Recommendations



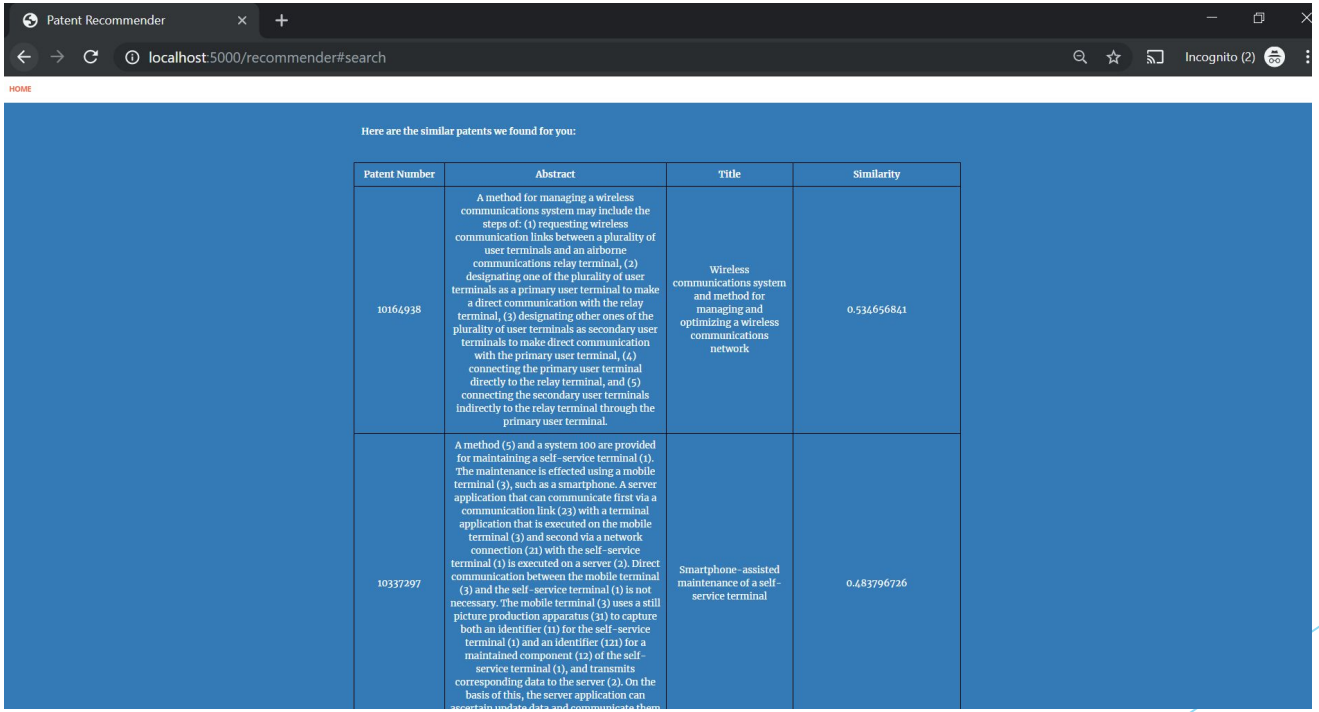
Our System Prototype

Input :UserId



Our System Prototype

Output: Patent Recommendations



References

- https://www.uspto.gov/patents-application-process/search-patents
- https://pandas.pydata.org
- https://en.wikipedia.org/wiki/Tf%E2%80%93idf
- https://en.wikipedia.org/wiki/Recommender_system

Thank You