باسمه تعالی

درس: بازیابی اطلاعات

نام و نام خانوادگی: تینا توکلی - هادی امینی

شماره دانشجویی: 9912762370 - 9922762220

شماره تمرین گروهی:04

توضیح وتحلیل بخش ها:

بخش اول:

پس از crawlکردن داده ها ،خروجی در 2 فایل json (newest و relevance)ذخیره شده است که برای استفاده از آنها در بخش بعدی ،مقادیر عددی باید به صورت int باشد که این مورد را در فاز قبلی رعایت کرده ایم و همچنین تاریخ ها باید به صورتtimestamp باشد که این مورد در بخش بعدی در فایل insert.py پوشش داده میشود(قبل از اضافه کردن داده ها).

خروجی:


```
ta > {} newest.json > [ ] papers > {} 5
{
    "papers": [
        {
            "title": "Enhanced Perimeter Intrusion Detection System (PIDS) - Resilient to Environmental Variations for
            "Page(s)": null,
            "Cites in Papers": null,
            "Cites in Patent": null,
            "Full Text Views": null,
            "Publisher": "IEEE",
            "DOI": "10.1109/SysCon61195.2024.10553451",
            "Date of Publication": "15-18 April 2024",
            "Abstract": "Perimeter intrusion detection systems (PIDS) often face challenges in accurately detecting bre
            "Published in": {
                "name": "2024 IEEE International Systems Conference (SysCon)",
                "link": "https://ieeexplore.ieee.org/xpl/conhome/10553374/proceeding"
            },
            "Authors": [
                {
                    "name": "Ravindranath KV",
                    "from": "ComputerScience Department, Data Science, Texas A&M University, College Station, USA"
                },
                {
                    "name": "Irfan Ahmad Khan",
                    "from": "Marine Engineering Technology Department in a Joint Affiliate Appointment With Electrical
                }
            ],
            "IEEE Keywords": [
                "Optical fibers",
                "Optical fiber sensors",
                "Transducers",
                "Intrusion detection",
                "Real-time systems",
                "Sensors"
```


شکل 1بخشی از خروجی crawl

بخش دوم:

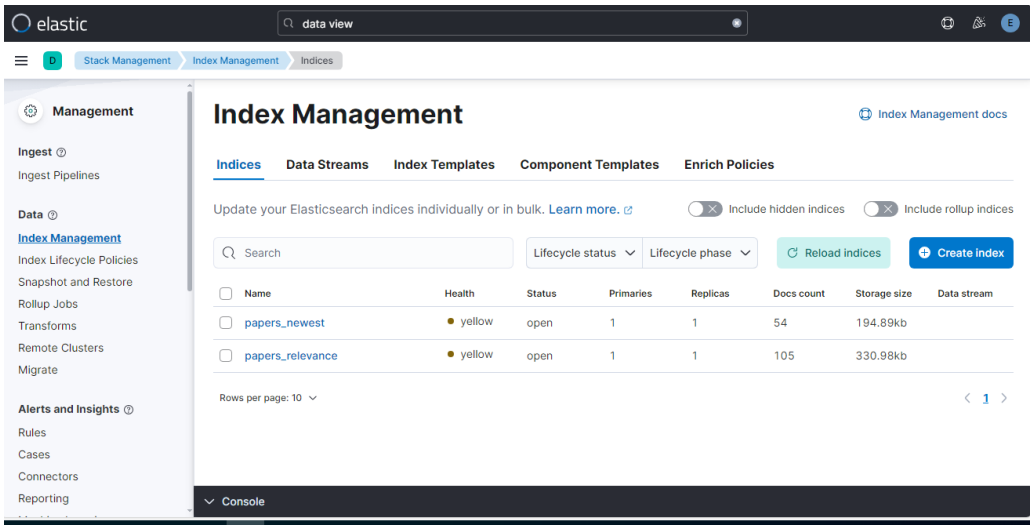
در این بخش از فایل insert.py استفاده می‌کنیم تا دو ایندکس برای دیتاهای newest و relevance ایجاد و داده ها را به elastic اضافه کنیم.

سپس چند visualization میسازیم تا با استفاده از آن ها اطلاعات آماری داده ها را در داشبورد نمایش دهیم:

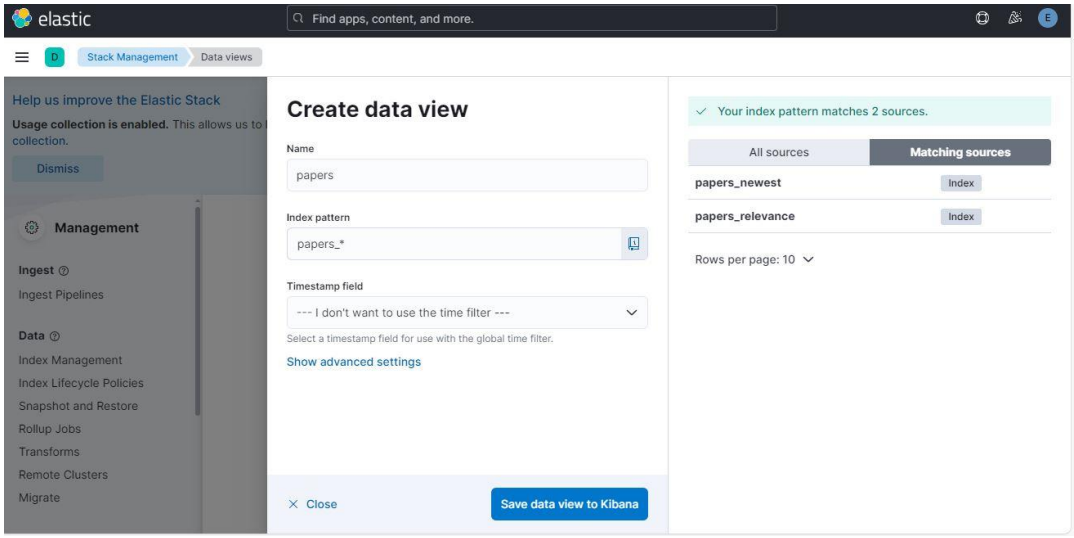
- نمودار tag cloud برای مشاهده کلمات کلیدی پرتکرار
- نمودار دایره ای برای مقایسه تعداد مقالات در کنفرانس های مختلف
- نمودار میله ای برای بررسی سال های انتشار مقالات
- نمودار میله‌ای افقی برای نمایش میزان citation ها
- تعداد کل رکورد ها

به وسیله این داشبورد میتوانیم دید جامعی نسبت به نتایج جستجو به دست آوریم.

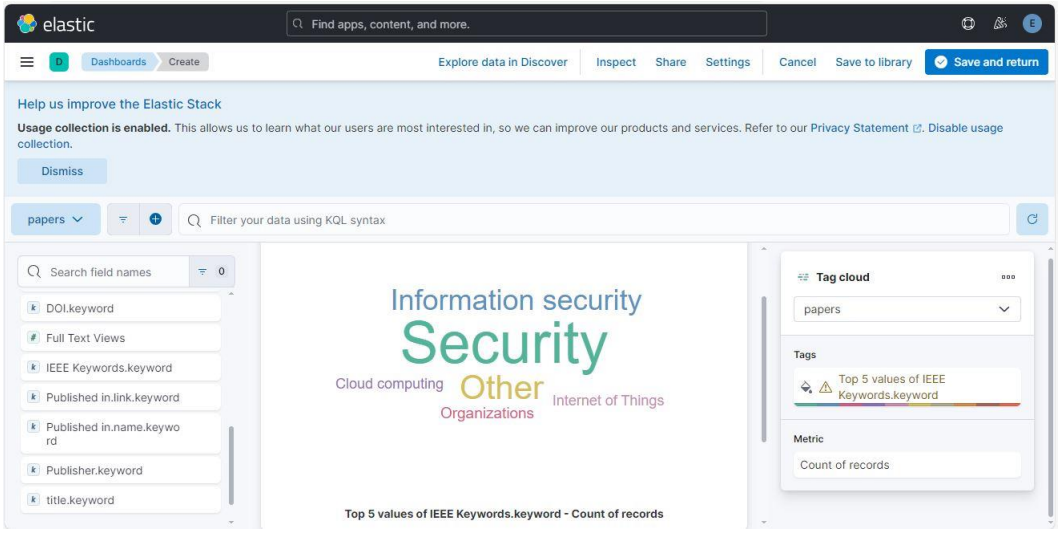
- **خروجی :**



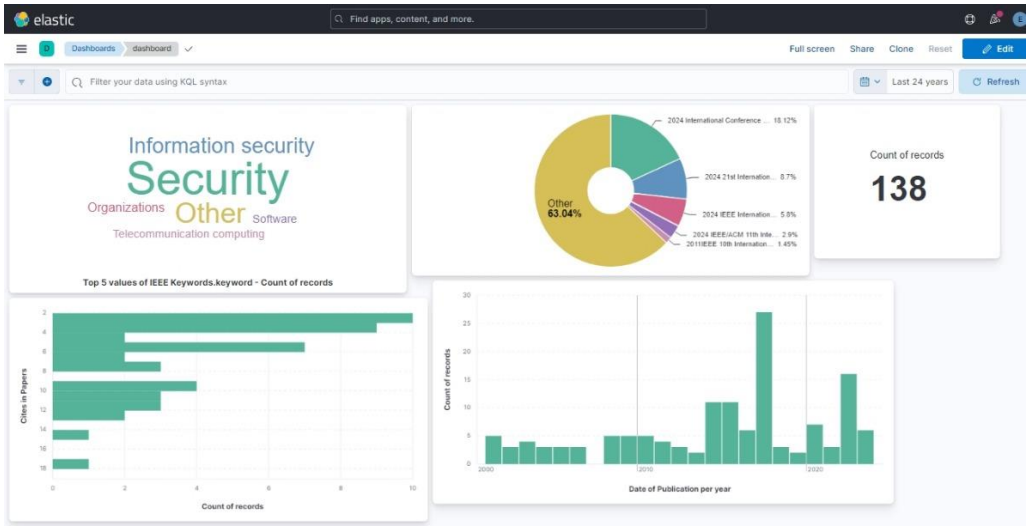
شکل 2 اضافه شدن ایندکس ها



شکل 3ساخت data view از indexs



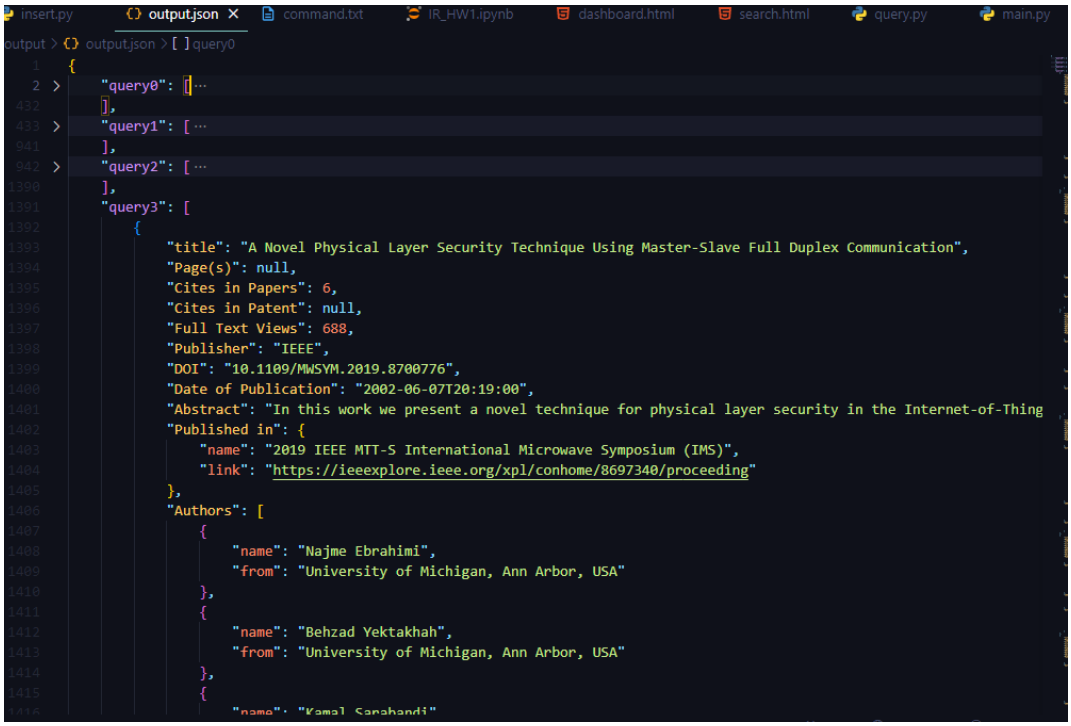
شکل 4-ایجاد یک داشبورد در ارتباط با keywords



شکل 5-نمایش داشبورد نهایی

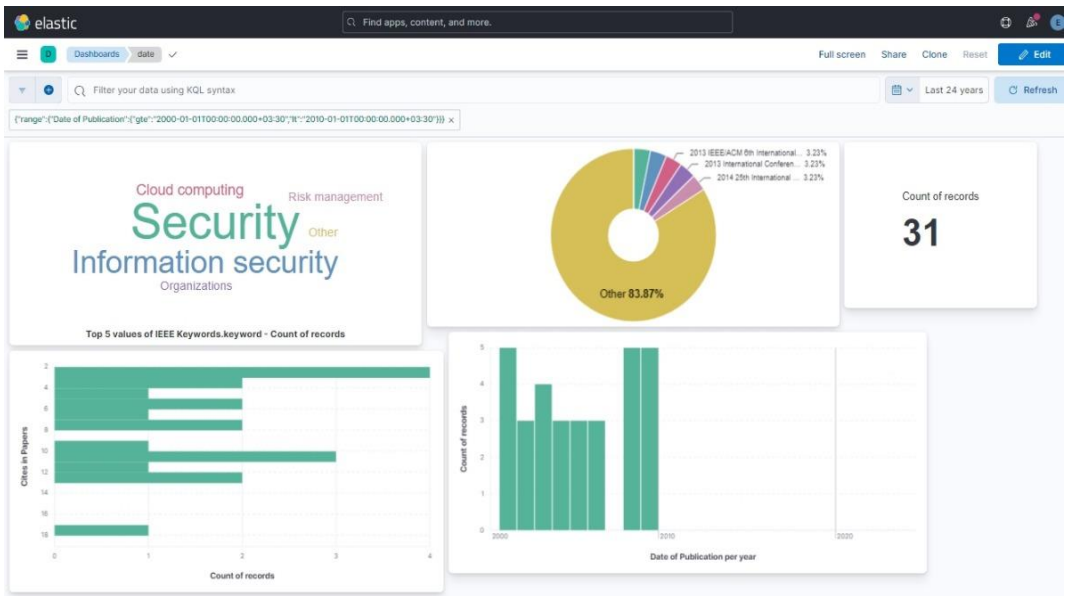
در 138تا مقاله ی موجود بیشترین keywordsمرتبط با امنیت وامنیت اطلاعات بوده وهمچنین اکثر مقالات در بازه 2015 تا 2018 منتشر شده و همچنین citationهای کمی دارند و بیشتر آنها نیز در کفرانس یکسانی منتشر نشده اند و پراکندگی زیادی وجود دارد.

- **بخش اول ساخت پرس و جوها:**



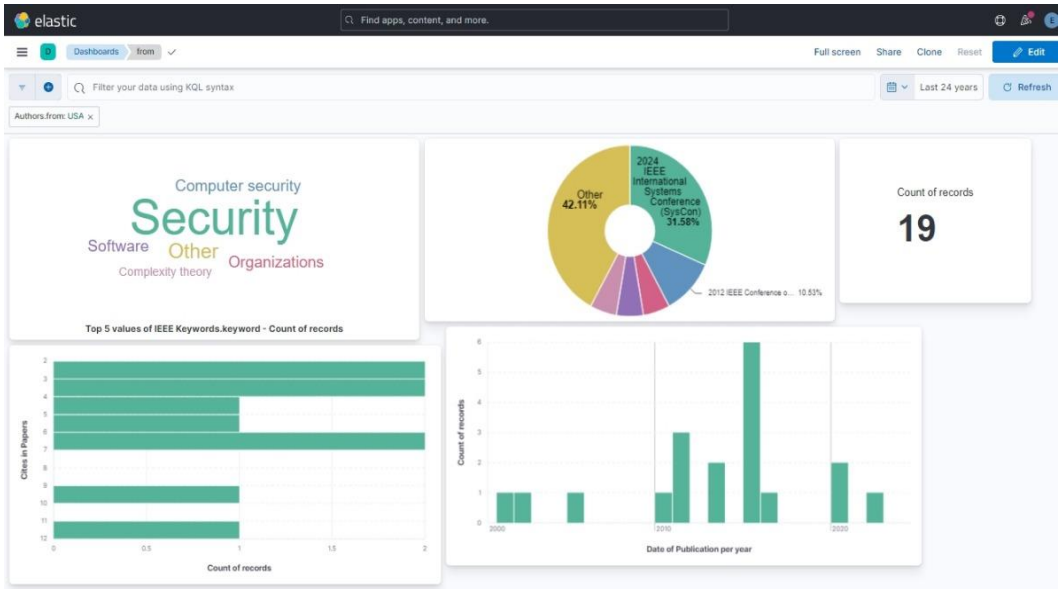
شکل 6-خروجی ذخیره شده در فایل

- **بخش دوم ساخت پرس و جوها:**



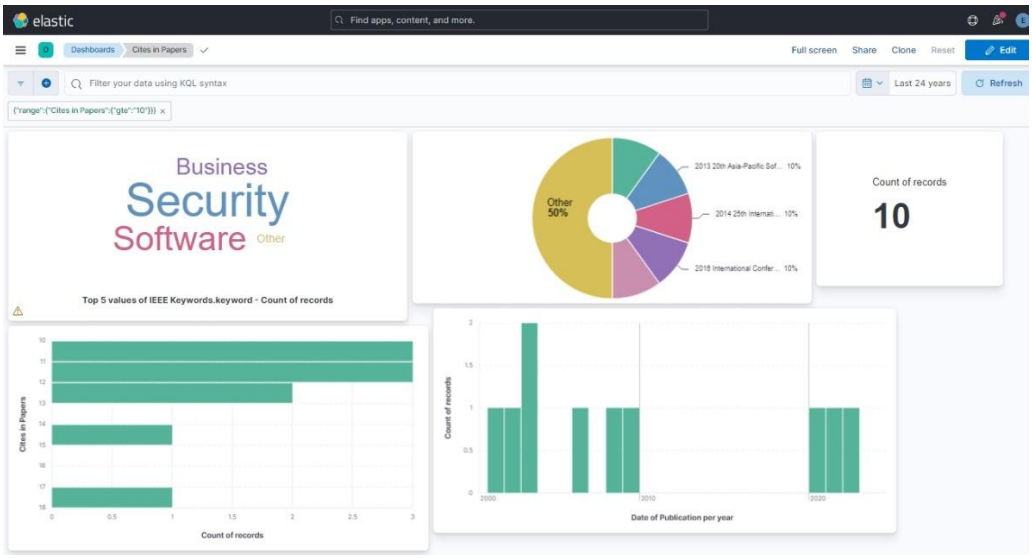
شکل 7-کوئری مربوط به date

این کوئری مقالات در بازه سال 2000 الی 2010 را نشان میدهد. بر اساس کلمات کلیدی، مبحث امنیت از پردازش ابری اهمیت بیشتری داشته است. اکثر این مقالات در ابتدای بازه یا سال های انتهایی بازه منتشر شده اند.



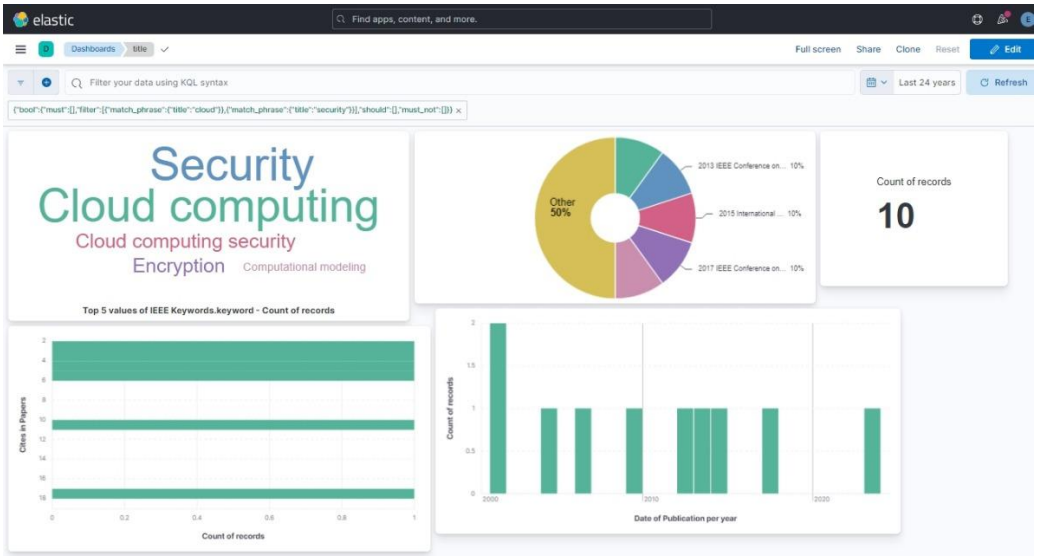
شکل 8کوئری مربوط به author.from

این کوئری، مقاله های مربوط به نویسندگان آمریکایی را نشان میدهد. 30٪ این مقالات در کنفرانس IEEE International Systems 2024 منتشر شده اند. همچنین نمودار تاریخ انتشار نشان میدهد که در سال 2015 نسبت به سایر سال ها مقالات بیشتری منتشر شده است.



شکل 9کوئری مربوط بهcites in paper

این کوئری مقالات با حداقل 10 citiation را بازیابی میکند. اکثر این مقالات در حوزه های امنیت و نرم افزار و بیزینس بوده‌اند.



شکل 10کوئری مربوط بهtitle

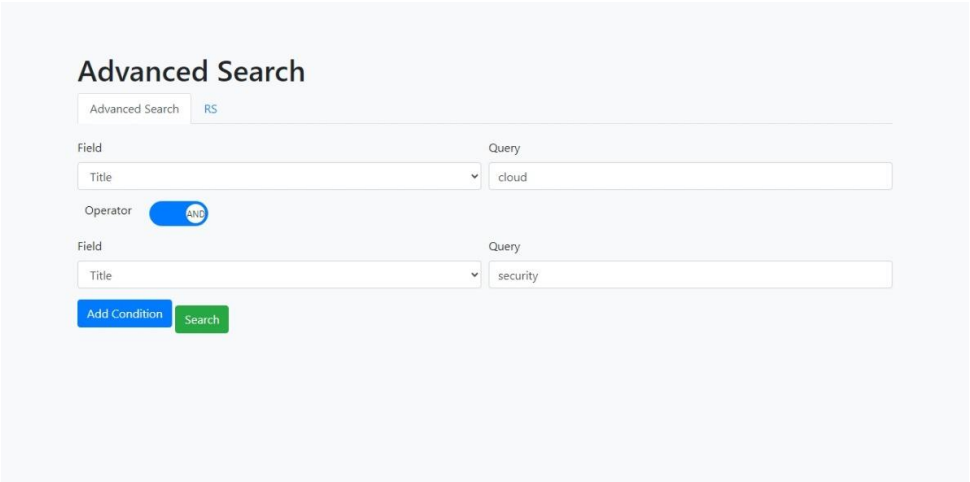
این کوئری مقالات شامل کلمه security و cloud در عنوان هستند. مطابق انتظار کلمات Cloud Computing و Security بیشتر در کلمات کلیدی حضور دارند و ترکیب این دو معادل Cloud computing security در رتبه سوم قرار دارد . همچنین رمزنگاری(encryption) نیز اهمیت داشته است.

بخش سوم:

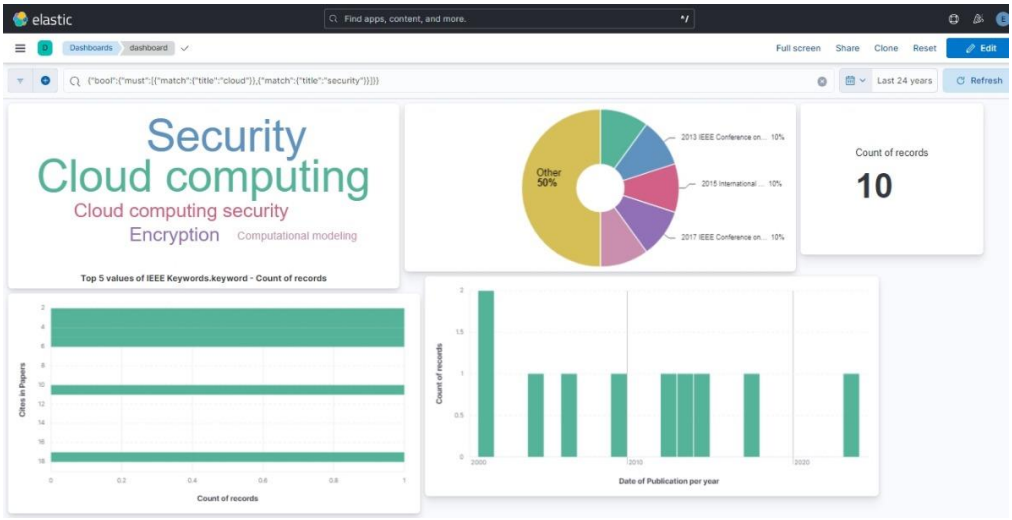
در این قسمت با استفاده از flask یک رابط وب برای جستجوی مقالات پیاده سازی میکنیم. با استفاده از ورودی کاربر، کوئری ساخته میشود و لینک داشبورد برای مشاهده نتایج جستجو نمایش داده میشود.

(کاربر امکان ترکیب فیلدهای مختلف با استفاده از عملگرهای and ,or دارد)

- **خروجی:**



شکل 11 – advanced search



شکل 12-نمایش داشبورد براساس کوئری وارد شده در صفحه advanced search

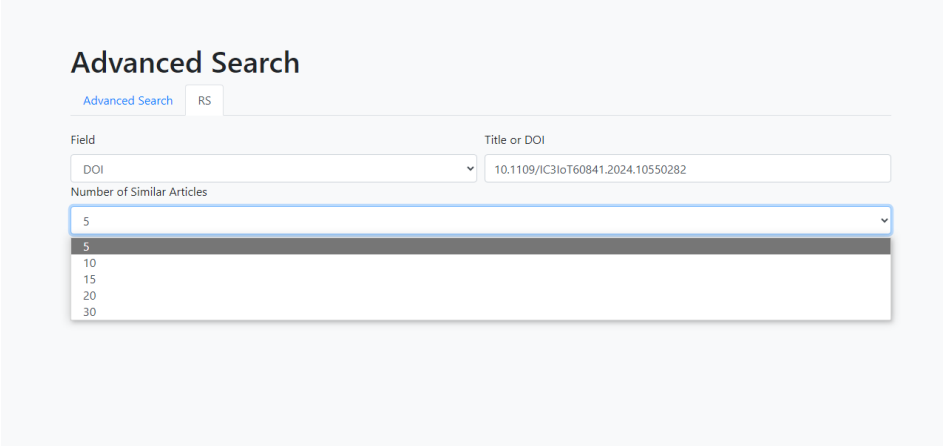
بخش چهارم:

در این بخش کاربر نام مقاله یا DOI آن را به همراه تعداد مقالات مشابه درخواستی اعلام میکند. مقاله مورد نظر از elastic دریافت میشود و شباهت کسینوسی آن با سایر مقالات محاسبه میشود. میزان شباهت از طریق محاسبه score به صورت زیر به دست می‌آید:

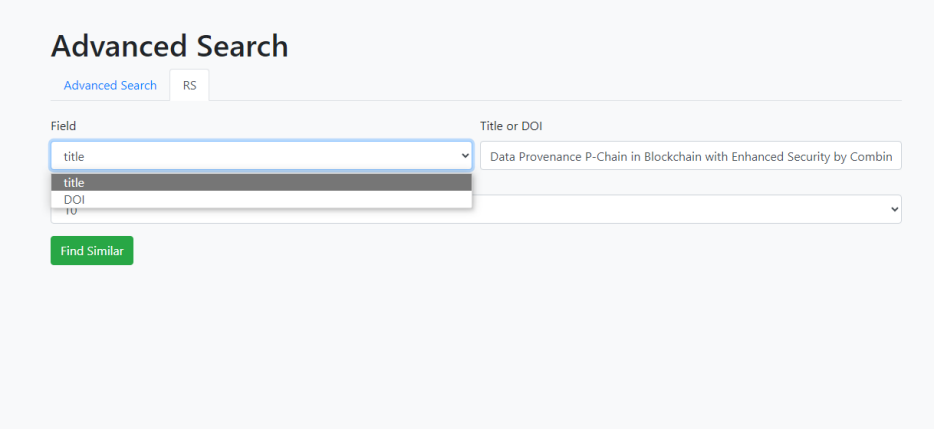
$$Score_i = 0.5 * Similarity_{title}\ (paper_{user}, paper_i) + Similarity_{author\_keywords}\ (paper_{user}, paper_i) + Similarity_{IEEE\_keywords}\ (paper_{user}, paper_i) + Similarity_{abstract}\ (paper_{user}, paper_i)$$

سپس امتیاز ها sort میشوند و بالاترین ها به تعداد درخواستی انتخاب میشوند.

- **خروجی :**



شکل 13-RS براساس DOI



شکل 14 RSبراساس title

Recommended Papers

Rank	Title	DOI
1	Prov-IoT: A Security-Aware IoT Provenance Model	10.1109/TrustCom50675.2020.00183
2	The Research Study on Identification of Threats and Security Techniques in Cloud Environment	10.1109/IDICAIEI58380.2023.10407003
3	Review of Data Governance Approaches in the Field of Transportation Domain	10.1109/SCSP61506.2024.10552682
4	Cloud Security: Challenges and Strategies for Ensuring Data Protection	10.1109/ICTACS59847.2023.10390302
5	Research of Security Situational Awareness and Visualization Approach in Cloud Computing	10.1109/NANA.2018.8648738
6	Cloud Storage Security Risks, Practices and Measures: A Review	10.1109/INOCON50539.2020.9298281
7	Ethereum Powered E-Tendering System	10.1109/IC3IoT60841.2024.10550251
8	Security threat probability computation using Markov Chain and Common Vulnerability Scoring System	10.1109/ATNAC.2018.8615386
9	Public Key Infrastructure Approaches Based on Blockchain	10.1109/SSD61670.2024.10549485
10	IDPS based framework for security in green cloud computing and comprehensive review on existing frameworks and security issues	10.1109/CCCS.2015.7374153

شکل 15خروجی مرتبط با لیست مقالات پیشنهاد داده شده براساس ورودی کاربر

مقاله مرجع در ارتباط با blockchain و security و data protection است که با توجه به امتیازی که به سایر مقالات تعلق گرفته براساس شباهت چکیده وکلمات کلیدی ،این موضوعات در مقالات پیشنهاد شده نیز مشاهده میشود .

توضیح کدها:

- **فایل insert.py:**

برای ایندکس کردن داده های JSON از دو فایل جداگانه به Elasticsearch استفاده می شود.

- مراحل:

اتصال به Elasticsearch:

به Elasticsearch با استفاده از URL، احراز هویت API و غیرفعال کردن تأیید گواهی SSL متصل می شود.

- تابع load_json:

برای خواندن داده های JSON از یک فایل مشخص شده استفاده می شود.

فایل را با استفاده از open() در حالت خواندن ('r') و رمزگذاری UTF-8 باز می کند.

داده های JSON را با استفاده از load.json() بارگیری می کند و آنها را برمی گرداند.

- تابع convert_to_date:

برای تبدیل رشته تاریخ در هر مقاله به datetime استفاده می شود.

رشته تاریخ را با استفاده از paper.("Date of Publication")get بازیابی می کند.

اگر رشته تاریخ وجود داشته باشد:

رشته را با استفاده از parser.dateutil به datetime تبدیل کند.

در صورت موفقیت datetime را به عنوان "Date of Publication" در مقاله ذخیره می کند و تاریخ را چاپ می کند.

مقاله را با تاریخ تبدیل شده برمی گرداند.

- بارگذاری داده ها:

کد داده ها را از دو فایل JSON جداگانه با استفاده از load_json() بارگیری می کند.

- تابع insert:

برای ایندکس کردن داده ها در Elasticsearch استفاده می شود.

داده ها را به عنوان ورودی دریافت می کند و نام شاخص را به عنوان arg می گیرد.

تاریخ های رشته را در داده ها با استفاده از convert_to_date() به datetime تبدیل می کند.

لیستی از actions را ایجاد می کند که هر کدام شامل _index، _source (داده های مقاله) و _id (شناسه منحصر به فرد برای هر مقاله) است.

از bulk.helpers() برای ایندکس کردن batch داده ها در Elasticsearch به طور همزمان استفاده می کند.

- **فایل app.py**:

app = Flask(name): یک نمونه برنامه Flask ایجاد می کند.

- اتصال به Elasticsearch:

به Elasticsearch با استفاده از URL، احراز هویت API و غیرفعال کردن تأیید گواهی SSL متصل می شود.

- تابع create_kibana_query:

لیستی از نتایج جستجو (results) را از Elasticsearch دریافت می کند.

اگر نتایجی وجود داشته باشد، یک پرس و جو Kibana را با استفاده از یک عبارت "should" بولین با یک پرس و جو "match" برای عنوان هر نتیجه (با فرض وجود title در اسناد Elasticsearch) ایجاد می کند.

دیکشنری پرس و جو Kibana یا None را در صورت عدم وجود نتیجه برمی گرداند.

- تابع get_dashboard_url:

رشته پرس و جو Elasticsearch رمزگذاری شده JSON (query) را دریافت می کند.

URL برای نمایش داشبورد Kibana را با درج رشته پرس و جو ارائه شده ایجاد می کند.

URL کامل نمایش داشبورد Kibana را برمی گرداند.

- home Route (GET Request):

قالب search.html را render می کند.

- search Route (POST Request):

پرس و جوهای جستجوی کاربر را مدیریت می کند.

داده های فرم را برای فیلدها، پرس و جوها و عملگرها بازیابی می کند.

پرس و جوهای Elasticsearch را با استفاده از یک عبارت "must" یا "should" بولین بسته به عملگرها ایجاد می کند.

جستجو را در شاخص papers_relevance Elasticsearch و papers_newestاجرا می کند.

بر اساس نتایج جستجو، یک پرس و جو Kibana و یک URL قابل جاسازی ایجاد می کند.

قالب dashboard.html را با URL جاسازی Kibana render می کند.

- rs_search Route (POST Request):

درخواست های کاربر برای مقالات مشابه را با استفاده از عملکرد سیستم توصیه (در rs.py پیاده سازی شده است) مدیریت می کند.

داده های فرم را برای فیلد، عنوان یا DOI و تعداد نتایج بازیابی می کند.

برای یافتن مقالات مشابه، تابع get_similar_papers را از rs.py فراخوانی می کند.

قالب rs.html را با لیست مقالات مشابهrender می کند.

- Main:
- app.run(debug=True): برنامه Flask را در حالت debug اجرا می کند، که برای توسعه مفید است اما در محیط های تولیدی توصیه نمی شود.
- **rs.py (Recommender System Module):**
- تابع get_similar_papers:
- ورودی ها:

es: شیء کلاینت Elasticsearch که برای تعامل با پایگاه داده Elasticsearch استفاده می شود.

target_field: نام فیلدی در اسناد Elasticsearch که برای جستجوی مقالات مشابه بر اساس آن استفاده می شود (مانند "title" یا "DOI").

target_value: مقداری که باید در فیلد target_field جستجو شود.

count: تعداد مقالات مشابهی که باید به عنوان نتایج برگردانده شوند.

- مراحل:

ایجاد پرس و جو Elasticsearch: تابع create_query را فراخوانی می کند تا یک پرس و جو جستجو ایجاد کند که مقالات مربوطه را بر اساس target_field و target_value بازیابی می کند.

جستجوی مقالات: از شیء es برای اجرای پرس و جوی جستجو در شاخص های papers_relevance و papers_newest استفاده می کند. نتایج جستجو در متغیری به نام res ذخیره می شوند.

استخراج مقاله هدف: اولین مقاله بازیابی شده در res['hits']['hits'] مقاله هدفی است که برای آن مقالات مشابه باید یافت شوند.

محاسبه شباهت: تابع calculate_similarity را با پارامترهای data، target_paper و count فراخوانی می کند تا مقالهها مشابه را بر اساس TF-IDF و شباهت کسینوس پیدا کند.

بازگشت نتایج: لیستی حاوی title و DOI های count مقاله مشابه برتر را برمی گرداند.

- تابع calculate_similarity:
  - ورودی ها:

    data: لیستی از تمام مقالات موجود در مجموعه داده، که به طور بالقوه از فایل های JSON بارگیری می شوند.

    target_paper: مقاله ای که برای آن مقالات مشابه باید یافت شوند.

    count: تعداد مقالات مشابهی که باید به عنوان نتایج برگردانده شوند.

  - مراحل:

    پیش پردازش کلمات کلیدی: کلمات کلیدی را از فیلدهای "IEEE keywords" و "Author Keywords " در target_paper استخراج و به یک رشته واحد تبدیل میکند.

    تکرار در فیلدهای متنی: در فیلدهای متنی مرتبط (title، IEEE Keywords، Author Keywords، Abstract) در هر مقاله از لیست data تکرار می کند:

    محتوای متنی را از فیلد فعلی برای هر مقاله استخراج می کند.

    یک نمونه از TfidfVectorizer را برای یادگیری اهمیت کلمات در کل مجموعه داده ایجاد می کند.

    vectorizerرا بر روی محتوای متنی استخراج شده آموزش می دهد و یک نمایش TF-IDF از آن ایجاد می کند.

    محتوای متنی target_paper را برای فیلد فعلی با استفاده از vectorizerرا آموزش دیده به یک بردار TF-IDF تبدیل می کند.

    شباهت کسینوس بین بردار TF-IDF target_paper و بردارهای TF-IDF تمام مقالات دیگر را برای فیلد فعلی محاسبه می کند.

    ترکیب شباهت ها: یک دیکشنری به نام similarities ایجاد می کند تا نمرات شباهت کسینوس را برای هر فیلد متنی ذخیره کند.

    به شباهت عنوان (فیلد title) وزنی معادل 0.5 نسبت به سایر فیلدها اختصاص می دهد(3مورد دیگر وزنی برابر با 1 دارند).

    نمرات شباهت را از تمام فیلدها در یک لیست نمره واحد به نام score_list ترکیب می کند.

    رتبه بندی مقالات: از argsort برای یافتن شاخص های مقالهها با بالاترین نمرات شباهت ترکیبی در score_list استفاده می کند. وهمچنین شاخص target_paper را از لیست نهایی حذف می کند.

استخراج نتایج برتر: لیستی به نام top_docs حاوی title و DOI های مقاله ها مشابه برتر بر اساس شاخص های رتبه بندی شده در top_indices ایجاد می کند.

- تابع create_query:

- بر اساس نام فیلد (target_field) و مقدار (target_value) ارائه شده، یک پرس و جوی Elasticsearch ایجاد می کند.

  - مراحل:

یک دیکشنری به نام query ایجاد می کند.

در داخل دیکشنری query، یک دیکشنری دیگر به نام "bool" تعریف می کند. این دیکشنری نحوه ترکیب چندین شرط در پرس و جو را مشخص می کند.

در داخل دیکشنری "bool"، یک لیست به نام "must" ایجاد می کند. این لیست شرایطی را مشخص می کند که همه اسناد باید برای تطابق با آن ها برآورده شوند.

در داخل لیست "must"، یک دیکشنری به نام "match" ایجاد می کند. این دیکشنری یک شرط تطابق فیلد و مقدار را تعریف می کند.

دیکشنری "match" دارای:

target_field: این کلید به طور پویا بر اساس آرگومان ورودی تابع (target_field) تنظیم می شود. فیلد را در اسناد Elasticsearch مشخص می کند که باید با آن مطابقت داشته باشد.

target_value: این value نیز به طور پویا بر اساس آرگومان ورودی تابع (target_value) تنظیم می شود. مقداری را که فیلد در اسناد باید با آن مطابقت داشته باشد مشخص می کند.

  - مقدار برگشتی:

دیکشنری(query) را که بر اساس آرگومان های ورودی (target_field,target_value)ساخته شده است، برمی گرداند.

- بارگذاری داده ها:

تابع load_json و محتوای این فایل های JSON را می خواند.

لیست مقالات را از هر فایل JSON استخراج می کند و به ترتیب آنها را به متغیرهای data_newest و data_relevance اختصاص می دهد.

  - ترکیب داده ها:

data = data_newest + data_relevance: لیست مقالات را از هر دو فایل JSON در یک لیست واحد به نام data ترکیب می کند. این یک مجموعه داده یکپارچه برای سیستم توصیه ارائه می دهد.

- **Query.py**:

برای جستجو در یک خوشه Elasticsearch و ذخیره نتایج در قالب JSON است.

- اتصال به Elasticsearch:

به Elasticsearch با استفاده از URL، احراز هویت API و غیرفعال کردن تأیید گواهی SSL متصل می شود.

- تابع save_json:

داده ها (نتایج جستجو) را در قالب JSON ذخیره می کند.

data: داده ای که باید ذخیره شود.

filename : مسیر و نام فایل خروجی JSON. به طور پیش فرض "json.output/output" است.

این تابع فایل را در حالت نوشتن ("w") باز می کند، داده ها را با استفاده از json.dump ذخیره می کند .

- تابع search:

جستجوهای Elasticsearch را بر اساس لیستی از کوئری ها اجرا می کند.

queries: لیستی از دیکشنری ها، که هر کدام یک کوئری Elasticsearch را نشان می دهد.

یک دیکشنری خالی به نام json_result را برای ذخیره نتایج جستجو برای هر کوئری ایجاد می کند.

حلقه for هر کوئری را در لیست queries پیمایش می کند:

جستجوی Elasticsearch را با استفاده از es.search با query فعلی و شاخص های مشخص شده ('papers_relevance,papers_newest') انجام می دهد.

اسناد منبع را از نتایج جستجو با استفاده از لیست ([hit["_source"] for hit in res['hits']['hits']]) استخراج می کند و آنها را در یک لیست ذخیره می کند.

لیست اسناد منبع را برای کوئری فعلی به دیکشنری json_result با استفاده از فرمت 'f'query{i} (به عنوان مثال query0، query1 و غیره) به عنوان کلید اضافه می کند.

این تابع دیکشنری json_result حاوی اسناد بازیابی شده را برای هر کوئری برمی گرداند.

- تابع main:

قطعه کد اصلی که هر نتیجه جستجو را به طور مستقیم چاپ می کرد، نظر داده شده است. در عوض، از تابع search برای جمع آوری نتایج و ذخیره آنها در یک فایل JSON استفاده می شود.

کوئری های نمونه: لیستی از چهار query ایجاد می کند:

کوئری اول مقالاتی را با عبارت دقیق "cloud" و "security" در عنوان با استفاده از match_phrase با اپراتورهای bool جستجو می کند.

کوئری دوم مقالاتی را که بین 1 ژانویه 2000 و 1 ژانویه 2010 با استفاده از کوئری range در فیلد " Date of Publication" منتشر شده اند، جستجو می کند.

کوئری سوم مقالاتی با حداقل 10 استناد در فیلد "Cites in Papers" را با استفاده از کوئری range با gte (بزرگتر یا مساوی) جستجو می کند.

کوئری چهارم مقالاتی را که حداقل یک نویسنده از USAباشد.

مشارکت:

تینا توکلی	هادی امینی
10	10