گزارش فاز 3 پروژه بازیابی اطلاعات

زهرا فاضل
96102053

بخش 1

منبع1 منبع2 منبع3 منبع4

کلاس PaperSpider را که از کلاس Spider کتابخانه scrapy ارث می‌برد، ساخته و توابع start_request و parse را override کردم. وقتی در ترمینال دستور scrapy crawl paper_spider را وارد کنیم که paper_spider مقدار فیلد name در کلاس است و برای شناسایی spider از آن استفاده می‌شود، تابع start_request را صدا می‌زند و این تابع برای تمام لینک‌های موجود در لیست seed، request می‌فرستد. هر وقت پاسخ یک request دریافت شود، تابع parse صدا زده می‌شود. در تابع parse، اطلاعات مورد نیاز با استفاده از توابع xpath و css selector که برای کلاس Response هستند، استخراج می‌شود و اگر در عمق مجاز باشیم، برای هر ارجاع مقاله یک request فرستاده می‌شود. این روند تا جایی ادامه پیدا می‌کند که به تعداد مقاله مورد نیاز برسیم. عمق مجاز با DEPTH_LIMIT مشخص می‌شود و معلوم می‌کند با شروع از یک لینک تا چند لینک جلو برویم. برای آنکه نیاز به استفاده از ترمینال نباشد، از CrawlerRunner استفاده می‌کنیم. در آخر اطلاعات در یک فایل json که مسیر آن توسط data_path مشخص می‌شود، ذخیره می‌شود.

برای اجرا از طریق رابط کاربری، فایل main را اجرا کرده و یکی از دو دستور زیر را وارد کنید:

'crawl seed:[%link 1%, %link 2%, ..., %link n%] %number of pages%'

'crawl seed:[%link 1%, %link 2%, ..., %link n%]'

```
from scrapy import Request, Spider
from twisted.internet import reactor
from scrapy.crawler import CrawlerRunner
from re import findall
from json import dump
from math import ceil, log10


data_path = 'D:\\University\\Modern Information Retrieval\\Project\\Phase 3\\data\\data.json'
```

شکل 1. کتابخانه‌های مورد نیاز و مسیر ذخیره فایل json

```python
class PaperSpider(Spider):
    name = 'paper_spider'
    seed = []
    number_of_papers = 0
    papers = []
    custom_settings = {'DEPTH_LIMIT': 2}

    def start_requests(self):
        for url in self.seed:
            yield Request(url=url, callback=self.parse)

    def parse(self, response):
        if len(self.papers) < self.number_of_papers:
            info = response.css('pre.bibtex-citation::text').get()
            title = findall('title={.*}', info)[0].replace('title={', '').replace('}', '')
            authors = findall('author={.*}', info)[0].replace('author={', '').replace('}', '').split(' and ')
            year = findall('year={.*}', info)[0].replace('year={', '').replace('}', '') \
                if len(findall('year={.*}', info)) > 0 else ''
            abstract = response.xpath("//meta[@name='description']/@content")[0].extract() \
                if len(response.xpath("//meta[@name='description']/@content")) > 0 else ''
            references = ['https://www.semanticscholar.org' + reference.css('h2.citation__title a::attr(href)').extract()[0]
                          for reference in response.css('div.references').css('div.paper-citation')]
            self.papers.append({'id': response.url.split("/")[-1], 'title': title, 'authors': authors, 'date': year,
                                'abstract': abstract, 'references': references})
            for reference in references:
                yield Request(url=reference, callback=self.parse)
```

شکل 2. کلاس PaperSpider

```python
def run(seed, number_of_papers=2000):
    PaperSpider.seed = seed
    PaperSpider.number_of_papers = number_of_papers
    PaperSpider.custom_settings['DEPTH_LIMIT'] = ceil(log10(number_of_papers / len(seed))) + 2
    runner = CrawlerRunner()
    d = runner.crawl(PaperSpider)
    d.addBoth(lambda _: reactor.stop())  # @UndefinedVariable
    reactor.run()  # @UndefinedVariable
    with open(data_path, 'w', encoding='utf8') as outfile:
        dump(PaperSpider.papers, outfile, ensure_ascii=False)
```

شکل 3. تابع اجراکننده خزنده

```
Run:   main
"C:\Users\Zahra Fazel\AppData\Local\Programs\Python\Python37\python.exe" "D:/University/Modern Information Retrieval/Project/Phase 3/main.py"
Help:
    * Pay attention to whitespaces!
    To run:
        part 1 enter: 'crawl seed:[%link 1%, %link 2%, ..., %link n%] %number of pages%' or 'crawl seed:[%link 1%, %link 2%, ..., %link n%]'
        part 2 to load data in elasticsearch enter: 'index %data_path% %elasticsearch_host% %elasticsearch_port%'
        part 2 to delete index enter: 'delete index %elasticsearch_host% %elasticsearch_port%'
        part 3 to delete index enter: 'rank pages %alpha% %elasticsearch_host% %elasticsearch_port%'
        part 4 enter: 'search title:{'phrase': %'phrase'%, 'weight': %weight%} abstract:{'phrase': %'phrase'%, 'weight': %weight%} date:{'from': %'year'%, 'weight': %weight%} %True if page rar
        part 5 enter: 'run HITS %number of authors% %elasticsearch_host% %elasticsearch_port%
    Enter 'exit' to exit
crawl seed:['https://www.semanticscholar.org/paper/Attention-is-All-you-Need-Vaswani-Shazeer/204e3073870fae3d05bcbc2f6a8e263d9b72e776','https://www.semanticscholar.org/paper/The-Lottery-Ticket
Crawler finished. Data stored in 'D:\University\Modern Information Retrieval\Project\Phase 3\data\data.json'
```

شکل 4. نتیجه اجرا


```
[{"id": "f90720ed12e045ac84beb94c27271d6fb8ad48cf", "title": "The Lottery Ticket Hypothesis: Training Pruned Neural Networks", "authors": ["Jonathan Frankle", "Michael Carbin"], "date": "2018", "abstract": "Recent work on neural network pruning indicates that, at training time, neural networks need to be significantly larger in size than is necessary to represent the eventual functions that they learn. This paper articulates a new hypothesis to explain this phenomenon. This conjecture, which we term the \"lottery ticket hypothesis,\" proposes that successful training depends on lucky random initialization of a smaller subcomponent of the network. Larger networks have more of these \"lottery tickets,\" meaning they are more likely to luck out with a subcomponent initialized in a configuration amenable to successful optimization. \nThis paper conducts a series of experiments with XOR and MNIST that support the lottery ticket hypothesis. In particular, we identify these fortuitously-initialized subcomponents by pruning low-magnitude weights from trained networks. We then demonstrate that these subcomponents can be successfully retrained in isolation so long as the subnetworks are given the same initializations as they had at the beginning of the training process. Initialized as such, these small networks reliably converge successfully, often faster than the original network at the same level of accuracy. However, when these subcomponents are randomly reinitialized or rearranged, they perform worse than the original network. In other words, large networks that train successfully contain small subnetworks with initializations conducive to optimization. \nThe lottery ticket hypothesis and its connection to pruning are a step toward developing architectures, initializations, and training strategies that make it possible to solve the same problems with much smaller networks.", "references": ["https://www.semanticscholar.org/paper/Dropout%3A-a-simple-way-to-prevent-neural-networks-Srivastava-Hinton/34f25a8704614163c4095b3ee2fc969b60de4698", "https://www.semanticscholar.org/paper/Learning-both-Weights-and-Connections-for-Efficient-Han-Pool/1ff9a37d766e3a4f39757f5e1b235a42dacf18ff", "https://www.semanticscholar.org/paper/Data-free-Parameter-Pruning-for-Deep-Neural-Srinivas-Babu/b0bd441a0cc04cdd0d0e469fe4c5184ee148a97d", "https://www.semanticscholar.org/paper/Understanding-Dropout-Baldi-Sadowski/cc46229a7c47f485e090857cbab6e6bf68c09811", "https://www.semanticscholar.org/paper/Deep-Compression%3A-Compressing-Deep-Neural-Network-Han-Mao/642d0f49b7826adcf986616f4af77e736229990f", "https://www.semanticscholar.org/paper/ThiNet%3A-A-Filter-Level-Pruning-Method-for-Deep-Luo-Wu/049fd80f52c0b1fa4d532945d95a24734b62bdf3", "https://www.semanticscholar.org/paper/Diversity-Networks%3A-Neural-Network-Compression-Mariet-Sra/2dfef5635c8c44431ca3576081e6cfe6d65d4862", "https://www.semanticscholar.org/paper/A-Deep-Neural-Network-Compression-Pipeline%3A-Huffman-Han-Mao/397de65a9a815ec39b3704a79341d687205bc80a", "https://www.semanticscholar.org/paper/Pruning-Filters-for-Efficient-ConvNets-Li-Kadav/c2a1cb1612ba21e067a5c3ba478a8d73b796b77a", "https://www.semanticscholar.org/paper/Optimal-Brain-Surgeon-and-general-network-pruning-Hassibi-Stork/e8eaf8aedb495b6ae0e174eea11e3cfcdf4a3724"]}, {"id": "204e3073870fae3d05bcbc2f6a8e263d9b72e776", "title": "Attention is All you Need", "authors": ["Ashish Vaswani", "Noam Shazeer", "Niki Parmar", "Jakob Uszkoreit", "Llion Jones", "Aidan N. Gomez", "Lukasz Kaiser", "Illia Polosukhin"], "date": "2017", "abstract": "The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder-decoder configuration. The best performing models also connect the encoder and
```


شکل 5. قسمتی از فایل json خروجی

بخش 2

منبع 1 منبع 2 منبع 3 منبع 4

با استفاده از host و port ای که الستیک‌سرچ روی آن در حال اجرا است، در پایتون به آن دسترسی پیدا کرده و با تابع create، شاخص را می‌سازیم. اطلاعات را از فایل json خوانده و آن‌ها را وارد شاخص می‌کنیم. برای حذف شاخص از تابع delete استفاده می‌کنیم.

برای اجرا از طریق رابط کاربری، فایل main را اجرا کرده و از دستورات زیر استفاده کنید:

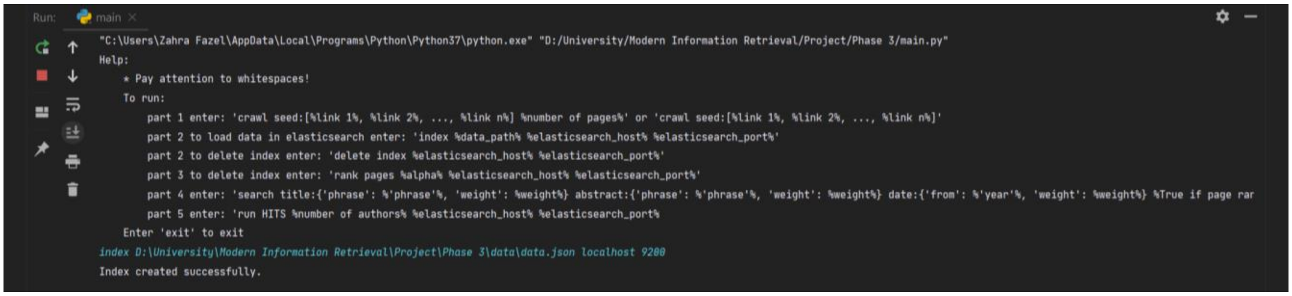
'index %data_path% %elasticsearch_host% %elasticsearch_port%'

'delete index %elasticsearch_host% %elasticsearch_port%'

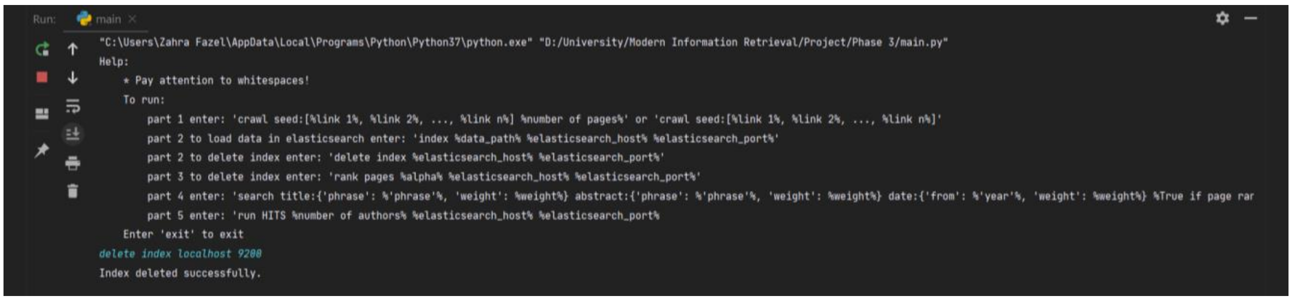
پارامتر date_path مسیری است که فایل json در آن قرار دارد.

```
from elasticsearch import Elasticsearch
from json import load
import numpy


def index(data_path, elasticsearch_host, elasticsearch_port):
    es = Elasticsearch([{'host': elasticsearch_host, 'port': elasticsearch_port}])
    es.indices.create(index='paper-index', ignore=400)
    with open(data_path, 'r', encoding='utf8') as infile:
        papers_data = load(infile)
    for i in range(len(papers_data)):
        es.index(index='paper_index', id=i, body={'paper': papers_data[i]})


def delete_index(elasticsearch_host, elasticsearch_port):
    es = Elasticsearch([{'host': elasticsearch_host, 'port': elasticsearch_port}])
    es.indices.delete(index='paper_index', ignore=[400, 404])
```

شکل 6. ایجاد و حذف شاخص



شکل 7. نتیجه اجرای ایجاد شاخص



شکل8 . نتیجه اجرای حذف شاخص

بخش 3

با استفاده از host و port ای که الستیک‌سرچ روی آن در حال اجرا است، در پایتون به آن دسترسی پیدا می‌کنیم. با پرسمان همه داده‌ها را از الستیک‌سرچ می‌گیریم. مراجع هر صفحه دو حالت دارند: خزیده شده‌اند که در این صورت اطلاعات آن‌ها در شاخص وجود دارد، یا خزیده نشده‌اند که در این صورت می‌توان آن‌ها را از گراف ارجاع صفحات حذف کرد. بنابراین رأس‌های گراف مجموعه صفحاتی‌اند که در الستیک‌سرچ اطلاعاتشان وجود دارد. رأس‌های مجاور هر رأس برابر است با مجموعه ارجاعاتی از آن که در رأس‌های گراف وجود دارند. به این شکل گراف ساخته می‌شود. مطابق با آنچه در اسلایدها بود، ماتریس احتمالات را ساخته و با استفاده از روش توانی، بردار ویژه‌ی آن را که همان page ranks است، پیدا می‌کنیم.

برای اجرا از طریق رابط کاربری، فایل main را اجرا کرده و از دستور زیر استفاده کنید:

' rank pages %alpha% %elasticsearch_host% %elasticsearch_port%'

```
def calculate_page_rank(alpha, elasticsearch_host, elasticsearch_port):
    es = Elasticsearch([{'host': elasticsearch_host, 'port': elasticsearch_port}])
    total = es.count(index='paper_index', body={'query': {'match_all': {}}})['count']
    responses = es.search(index='paper_index', body={'size': total, 'query': {'match_all': {}}})['hits']['hits']
    web_graph = {}
    pages = {}
    for response in responses:
        page_id = response['_source']['paper']['id']
        pages[page_id] = int(response['_id'])
    for response in responses:
        page_id = response['_source']['paper']['id']
        web_graph[pages[page_id]] = []
        for reference in response['_source']['paper']['references']:
            reference_id = reference.split('/')[-1]
            if reference_id in pages.keys():
                web_graph[pages[page_id]].append(pages[reference_id])
    probability_matrix = [[alpha / len(pages) for _ in range(len(pages))] for __ in range(len(pages))]
    for page in web_graph.keys():
        for reference in web_graph[page]:
            probability_matrix[page][reference] += (1 - alpha) * (1 / len(web_graph[page]))
    probability_matrix = numpy.asarray(probability_matrix)
```

شکل 9 . ساخت گراف صفحات و محاسبه ماتریس احتمالات

```python
probability_matrix = [[alpha / len(pages) for _ in range(len(pages))] for __ in range(len(pages))]
for page in web_graph.keys():
    for reference in web_graph[page]:
        probability_matrix[page][reference] += (1 - alpha) * (1 / len(web_graph[page]))
probability_matrix = numpy.asarray(probability_matrix)
page_rank = numpy.transpose([1 for _ in range(len(pages))])
eigenvalue = numpy.dot(numpy.transpose(page_rank), probability_matrix.dot(page_rank)) / \
             numpy.dot(numpy.transpose(page_rank), page_rank)
converge = False
iteration = 0
while not converge and iteration < 200000:
    page_rank_new = probability_matrix.dot(page_rank)
    page_rank_new /= numpy.linalg.norm(page_rank_new)
    eigenvalue_new = numpy.dot(numpy.transpose(page_rank_new), probability_matrix.dot(page_rank_new)) / \
                     numpy.dot(numpy.transpose(page_rank_new), page_rank_new)
    converge = abs(eigenvalue_new - eigenvalue) / eigenvalue_new <= 10 ** (-9)
    iteration += 1
    eigenvalue = eigenvalue_new
    page_rank = page_rank_new
for id in pages.values():
    es.update(index='paper_index', id=id, body={'doc': {'paper': {'page_rank': page_rank[id]}}})
```

شکل10 . محاسبه page rank و درج آن در شاخص

```
Run:  main
"C:\Users\Zahra Fazel\AppData\Local\Programs\Python\Python37\python.exe" "D:/University/Modern Information Retrieval/Project/Phase 3/main.py"
Help:
    * Pay attention to whitespaces!
    To run:
        part 1 enter: 'crawl seed:[%link 1%, %link 2%, ..., %link n%] %number of pages%' or 'crawl seed:[%link 1%, %link 2%, ..., %link n%]'
        part 2 to load data in elasticsearch enter: 'index %data_path% %elasticsearch_host% %elasticsearch_port%'
        part 2 to delete index enter: 'delete index %elasticsearch_host% %elasticsearch_port%'
        part 3 to delete index enter: 'rank pages %alpha% %elasticsearch_host% %elasticsearch_port%'
        part 4 enter: 'search title:{'phrase': %'phrase'%, 'weight': %weight%} abstract:{'phrase': %'phrase'%, 'weight': %weight%} date:{'from': %'year'%, 'weight': %weight%} %True if page rar
        part 5 enter: 'run HITS %number of authors% %elasticsearch_host% %elasticsearch_port%
    Enter 'exit' to exit
rank pages 0.1 localhost 9200
Calculation of page rank completed.
```

شکل11 . نتیجه اجرای page rank

بخش 4

منبع 1 منبع 2 منبع 3

با استفاده از host و port ای که الستیک‌سرچ روی آن در حال اجرا است، در پایتون به آن دسترسی پیدا می‌کنیم. از boost برای تعیین وزن استفاده می‌شود. از field_value_factor برای آنکه از یک یا چند فیلد داده‌ها را در امتیاز تأثیر دهیم، که در اینجا هدف تأثیر دادن page rank است. boost_mode مشخص می‌کند که امتیاز پرسمان و امتیاز تابع چگونه با هم ترکیب شوند. تأثیر آن در صورت استفاده در نتایج به شکل زیر خواهد بود:

_score = query_score + ln(1 + 10000 * page_rank)

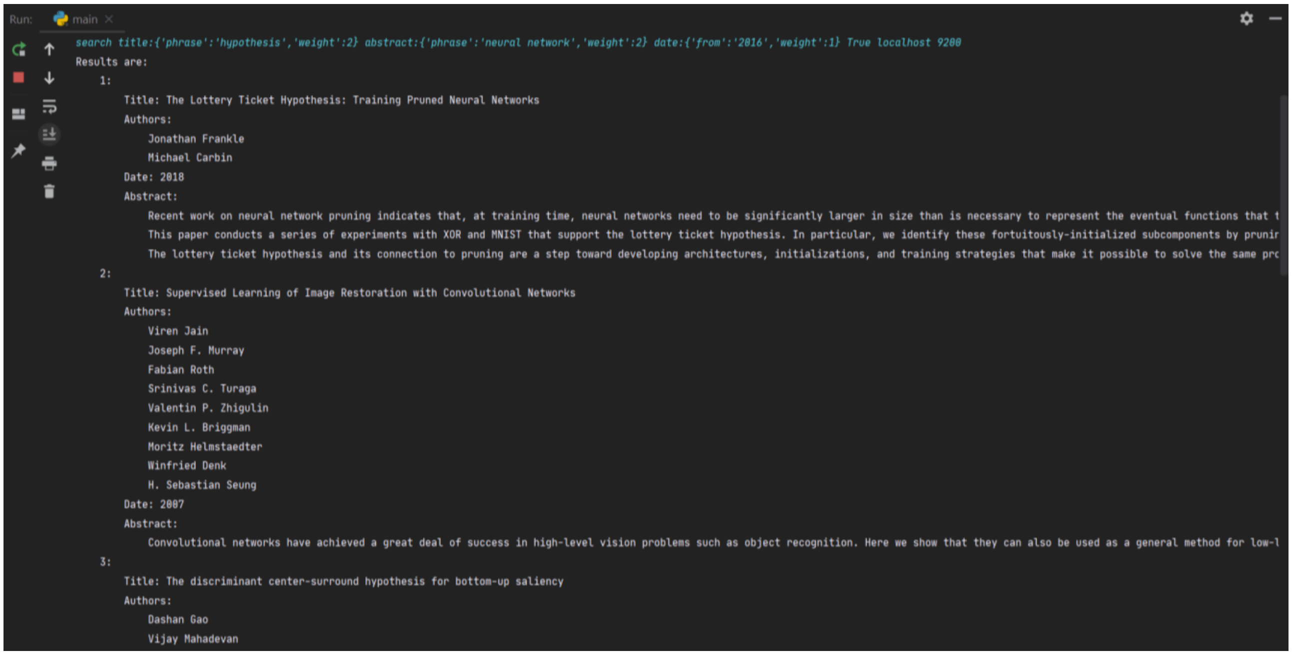
در نهایت جواب‌ها به ترتیب نزولی امتیاز برمی‌گردند.

برای اجرا از طریق رابط کاربری، فایل main را اجرا کرده و از دستورات زیر استفاده کنید:

'search title:{'phrase': %'phrase'%, 'weight': %weight%} abstract:{'phrase': %'phrase'%, 'weight': %weight%} date:{'from': %'year'%, 'weight': %weight%} %True if page rank should be used else False% %elasticsearch_host% %elasticsearch_port%'

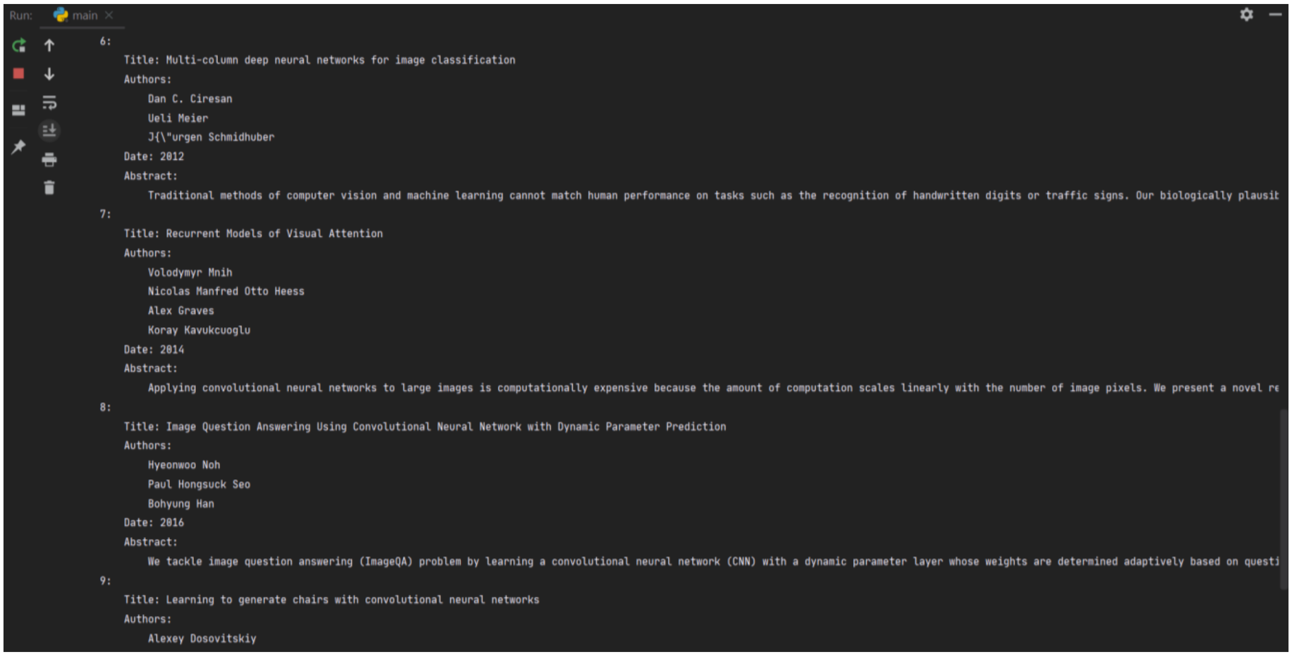
```python
def search(title, abstract, date, elasticsearch_host, elasticsearch_port, use_page_rank):
    es = Elasticsearch([{'host': elasticsearch_host, 'port': elasticsearch_port}])
    query = {'query': {
        'function_score': {'query': {'bool': {'should': [
                {'match': {"paper.title": {"query": title['phrase'], "boost": title['weight']}}}},
                {'match': {"paper.abstract": {"query": abstract['phrase'], "boost": abstract['weight']}}}},
                {'range': {"paper.date": {"gte": date['from'], "boost": date['weight']}}}
            ]}},
            'functions': [{"field_value_factor": {
                            "field": "paper.page_rank",
                            "factor": 10000 if use_page_rank else 0,
                            "modifier": "ln1p",
                            "missing": 0
            }}],
            "boost_mode": "sum"
        }
    }}
    response = es.search(index='paper_index', body=query)['hits']['hits']
    return response
```

شکل 12 . ساخت پرسمان و جست‌وجو

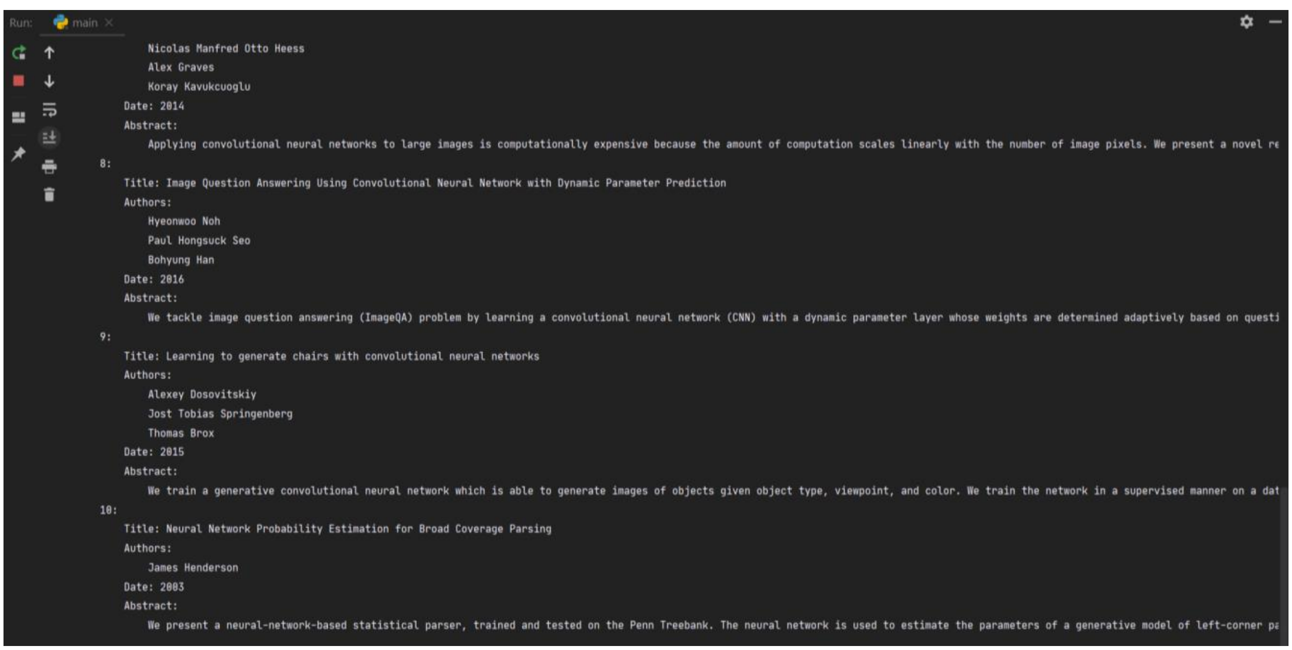


شکل 13. نتیجه اجرای جست‌وجو با تأثیر دادن page rank

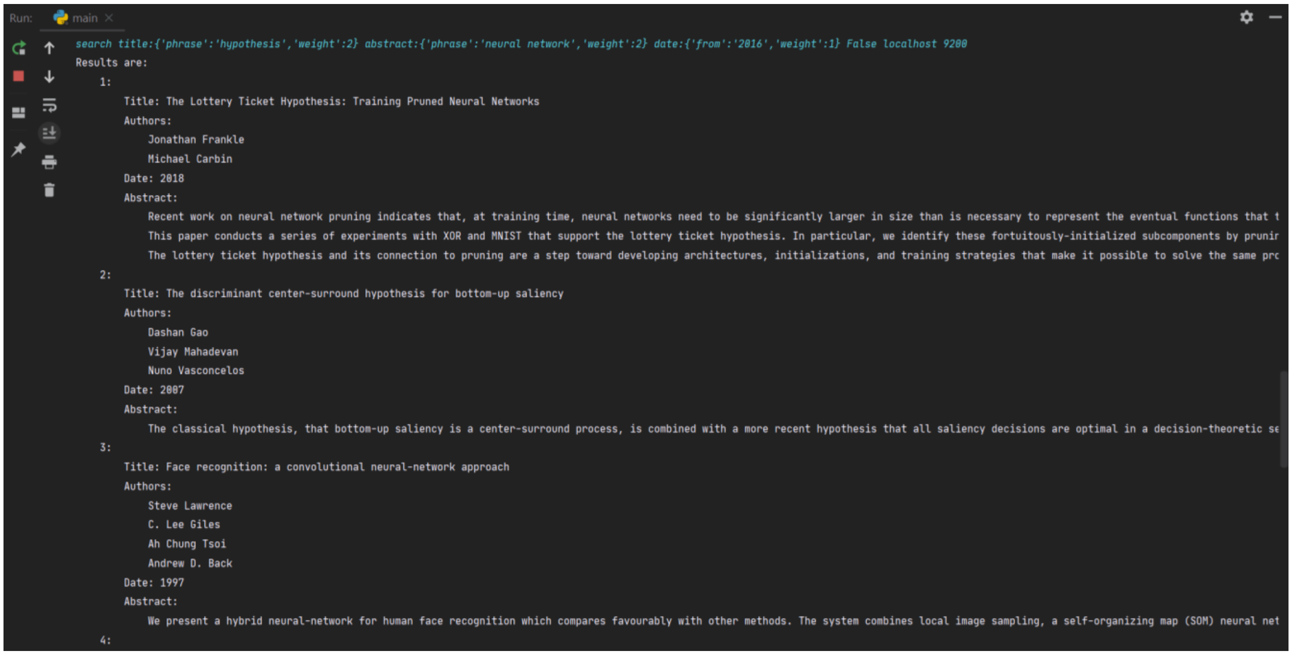
شکل 14. نتیجه اجرای جست‌وجو با تأثیر دادن page rank



شکل 15. نتیجه اجرای جستوجو با تأثیر دادن page rank

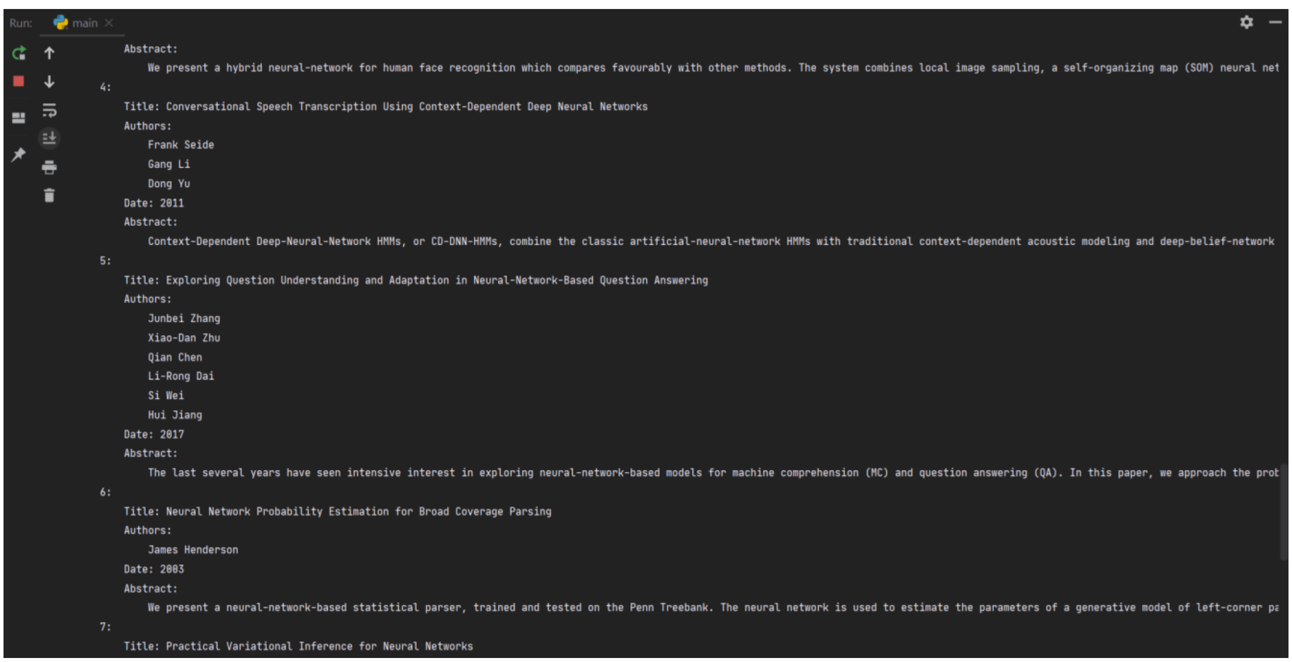


شکل 16. نتیجه اجرای جستوجو با تأثیر دادن page rank



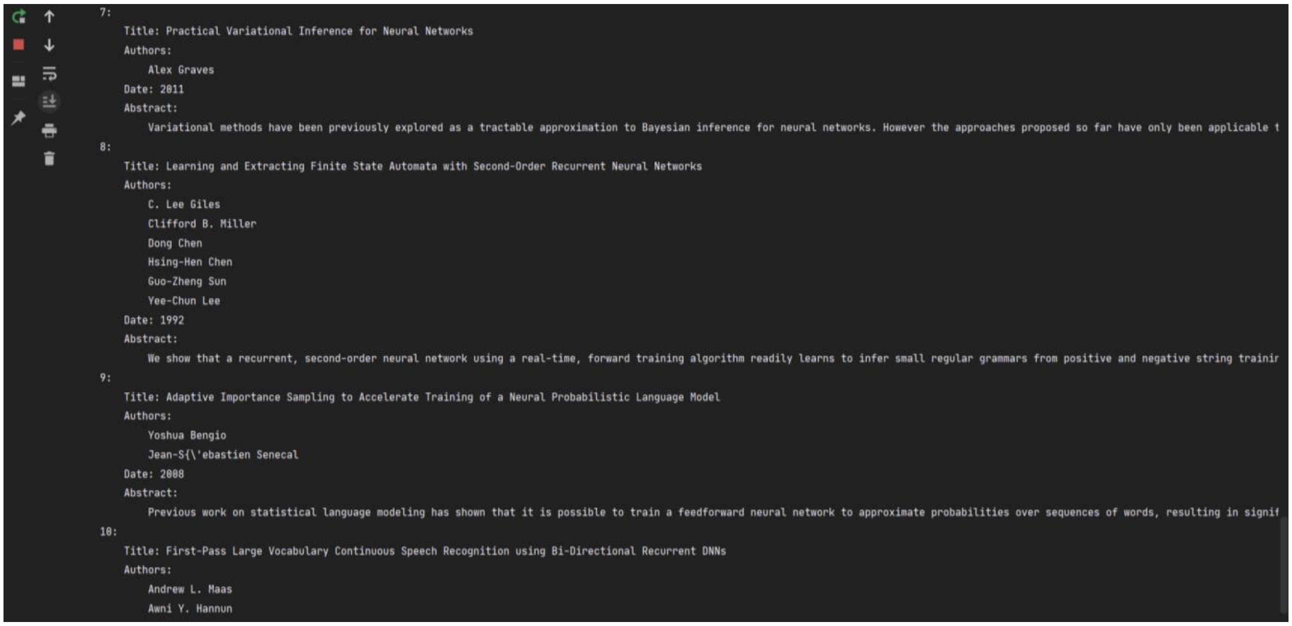
```
Run:    main
search title:{'phrase':'hypothesis','weight':2} abstract:{'phrase':'neural network','weight':2} date:{'from':'2016','weight':1} False localhost 9200
Results are:
    1:
        Title: The Lottery Ticket Hypothesis: Training Pruned Neural Networks
        Authors:
            Jonathan Frankle
            Michael Carbin
        Date: 2018
        Abstract:
            Recent work on neural network pruning indicates that, at training time, neural networks need to be significantly larger in size than is necessary to represent the eventual functions that t
            This paper conducts a series of experiments with XOR and MNIST that support the lottery ticket hypothesis. In particular, we identify these fortuitously-initialized subcomponents by prunir
            The lottery ticket hypothesis and its connection to pruning are a step toward developing architectures, initializations, and training strategies that make it possible to solve the same prc
    2:
        Title: The discriminant center-surround hypothesis for bottom-up saliency
        Authors:
            Dashan Gao
            Vijay Mahadevan
            Nuno Vasconcelos
        Date: 2007
        Abstract:
            The classical hypothesis, that bottom-up saliency is a center-surround process, is combined with a more recent hypothesis that all saliency decisions are optimal in a decision-theoretic se
    3:
        Title: Face recognition: a convolutional neural-network approach
        Authors:
            Steve Lawrence
            C. Lee Giles
            Ah Chung Tsoi
            Andrew D. Back
        Date: 1997
        Abstract:
            We present a hybrid neural-network for human face recognition which compares favourably with other methods. The system combines local image sampling, a self-organizing map (SOM) neural net
    4:
```

شکل 17. نتیجه اجرای همان جستوجو بدون تأثیر دادن page rank

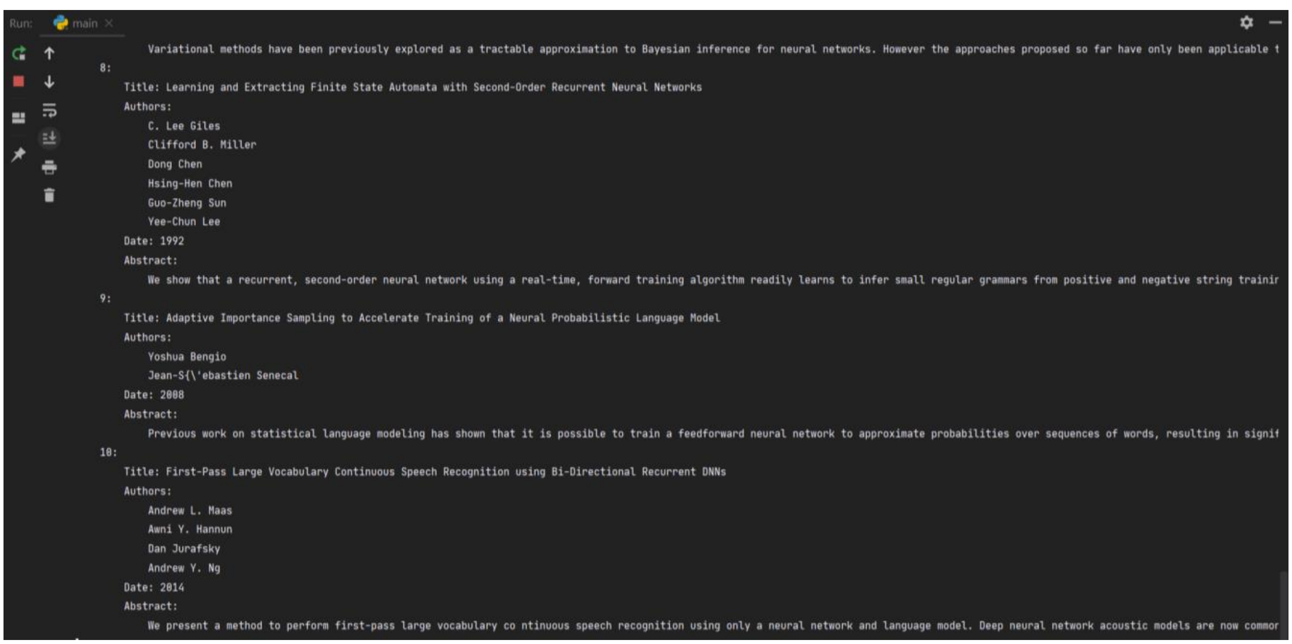


```
Run:    main
        Abstract:
            We present a hybrid neural-network for human face recognition which compares favourably with other methods. The system combines local image sampling, a self-organizing map (SOM) neural net
    4:
        Title: Conversational Speech Transcription Using Context-Dependent Deep Neural Networks
        Authors:
            Frank Seide
            Gang Li
            Dong Yu
        Date: 2011
        Abstract:
            Context-Dependent Deep-Neural-Network HMMs, or CD-DNN-HMMs, combine the classic artificial-neural-network HMMs with traditional context-dependent acoustic modeling and deep-belief-network
    5:
        Title: Exploring Question Understanding and Adaptation in Neural-Network-Based Question Answering
        Authors:
            Junbei Zhang
            Xiao-Dan Zhu
            Qian Chen
            Li-Rong Dai
            Si Wei
            Hui Jiang
        Date: 2017
        Abstract:
            The last several years have seen intensive interest in exploring neural-network-based models for machine comprehension (MC) and question answering (QA). In this paper, we approach the prot
    6:
        Title: Neural Network Probability Estimation for Broad Coverage Parsing
        Authors:
            James Henderson
        Date: 2003
        Abstract:
            We present a neural-network-based statistical parser, trained and tested on the Penn Treebank. The neural network is used to estimate the parameters of a generative model of left-corner pa
    7:
        Title: Practical Variational Inference for Neural Networks
```

شکل 18. نتیجه اجرای همان جستوجو بدون تأثیر دادن page rank



شکل 19. نتیجه اجرای همان جستوجو بدون تأثیر دادن page rank



شکل 20. نتیجه اجرای همان جستوجو بدون تأثیر دادن page rank

بخش 5

با استفاده از host و port ای که الستیک‌سرچ روی آن در حال اجرا است، در پایتون به آن دسترسی پیدا می‌کنیم. با پرسمان همه داده‌ها را از الستیک‌سرچ می‌گیریم. به ازای هر مقاله، مراجع و نویسندگان آن را در دیکشنری‌های id_authors و id_references ذخیره می‌کنیم. همچنین نام همه نویسندگان را گرفته و به هرکدام یک شماره اختصاص می‌دهیم. با استفاده از این سه دیکشنری، گراف ارجاع یک نویسنده به یک نویسنده دیگر را ساخته (reference_graph) و از روی آن با الگوریتم اسلایدها، مقدار hub و authority را محاسبه کرده و n نویسنده اول را برمی‌گردانیم.

برای اجرا از طریق رابط کاربری، فایل main را اجرا کرده و از دستور زیر استفاده کنید:

'run HITS %number_of_authors% %elasticsearch_host% %elasticsearch_port%'

```python
def run_hits(number_of_authors, elasticsearch_host, elasticsearch_port):
    es = Elasticsearch([{'host': elasticsearch_host, 'port': elasticsearch_port}])
    total = es.count(index='paper_index', body={'query': {'match_all': {}}})['count']
    responses = es.search(index='paper_index', body={'size': total, 'query': {'match_all': {}}})['hits']['hits']
    authors = {}
    id_authors = {}
    id_references = {}
    id = 0
    for response in responses:
        paper_id = response['_source']['paper']['id']
        for author in response['_source']['paper']['authors']:
            if author not in authors.keys():
                authors[author] = id
                id += 1
        id_authors[paper_id] = response['_source']['paper']['authors']
        id_references[paper_id] = [reference.split('/')[-1] for reference in response['_source']['paper']['references']]
    reference_graph = [[0 for _ in range(len(authors))] for __ in range(len(authors))]
```

شکل 21. استخراج اطلاعات مورد نیاز و ساختن دیکشنری‌ها

```python
reference_graph = [[0 for _ in range(len(authors))] for __ in range(len(authors))]
for id in id_authors.keys():
    for author in id_authors[id]:
        for reference in id_references[id]:
            if reference in id_authors.keys():
                for reference_author in id_authors[reference]:
                    reference_graph[authors[author]][authors[reference_author]] = 1
hub = [1 for _ in range(len(authors))]
authority = [1 for _ in range(len(authors))]
for i in range(5):
    new_hub = [0 for _ in range(len(authors))]
    new_authority = [0 for _ in range(len(authors))]
    for id_author in authors.values():
        for id_reference_author in authors.values():
            new_hub[id_author] += authority[id_reference_author] * reference_graph[id_author][id_reference_author]
            new_authority[id_author] += hub[id_reference_author] * reference_graph[id_reference_author][id_author]
    hub = new_hub
    authority = new_authority
hub = list(numpy.asarray(hub) / numpy.linalg.norm(numpy.asarray(hub)))
authority = list(numpy.asarray(authority) / numpy.linalg.norm(numpy.asarray(authority)))
```

شکل 22. ساخت گراف ارجاع و محاسبه hub و authority

```python
hub = list(numpy.asarray(hub) / numpy.linalg.norm(numpy.asarray(hub)))
authority = list(numpy.asarray(authority) / numpy.linalg.norm(numpy.asarray(authority)))
authority = {i: authority[i] for i in range(len(authority))}
authority = sorted(authority.items(), key=lambda kv: (kv[1], kv[0]), reverse=True)
best_authors = {}
for i in range(number_of_authors):
    index = authority[i][0]
    for author_name in authors.keys():
        if authors[author_name] == index:
            best_authors[author_name] = authority[i][1]
return best_authors
```

شکل 23. جدا کردن نویسندگان برتر

شکل 24. نتیجه اجرای الگوریتم HITS برای یافتن 20 نویسنده برتر

شکل 25. نتیجه اجرای الگوریتم HITS برای یافتن 20 نویسنده برتر

بخش 6

گزارش این بخش در همان فایل نوت‌بوک است.