

بخش اول:

اول یک سری تنظیمات را برای عنکبوت ست می‌کنیم:

```
'FEED_FORMAT': 'json',
```

```
'FEED_URI': 'result.json',
```

این تنظیمات باعث می‌شود که آیتم‌هایی که کral می‌کند را در فایل results.json بریزد.

```
'CLOSESPIDER_PAGECOUNT': 2000,
```

این باعث می‌شود بعد کral کردن ۲۰۰۰ صفحه، فرآیند کral متوقف شود

```
'DUPEFILTER_CLASS': 'scrapy.dupefilters.RFPDupeFilter',
```

این باید باعث شود که صفحه‌هایی که قبلا کral کرده را دوباره کral نکند (هرچند بعدا در سرچ بعضا آیتم‌های تکراری مشاهده می‌شود که نهایتا یک آرایه گرفتیم که idها را نگه دارد و قبل از yield کردن چک کند که id تکراری نباشد. اما با این کار هم در سرچ نتایج تکراری مشاهده می‌شد که آن را با استفاده از collapse در کوئری هندل می‌کنیم)

این خط‌ها که در آخر کد هستند، کral را initialize و بعد آن را run می‌کنند:

```
process = CrawlerProcess()
```

```
process.crawl(myspider)
```

```
process.start()
```

اول رفرنسها را جدا می‌کنیم چون می‌خواهیم لینک‌های آنها را دنبال و صفحاتشان را کral کنیم. چون لینک‌های موجود در رفرنسها به این شکل است:

```
paper/The-Lottery-Ticket-Hypothesis%3A-Training-Pruned-Frankle-Carbin/f90720ed12e045ac84beb94c27271d6fb8ad48cf
```

برای همین باید یک "<https://www.semanticscholar.org>" به اولشان بچسبانیم تا لینک کامل شود و بتوانیم آن را برای کral کردن به عنکبوت بدهیم. تابع add\_home\_url همین کار را می‌کند.

همچنین چون در titleها و abstractها بعضا تگ‌های html وجود دارد برای حذف آنها از تابع clean\_html استفاده می‌کنیم.

هم چنین selectorای که برای author پیدا کردیم یک رشته است که نویسنده‌ها را به صورت

```
@article{Frankle2018TheLT,  
  title={The Lottery Ticket Hypothesis: Training Pruned Neural Networks},  
  author={Jonathan Frankle and Michael Carbin},  
  journal={ArXiv},  
  year={2018},  
  volume={abs/1803.03635}
```

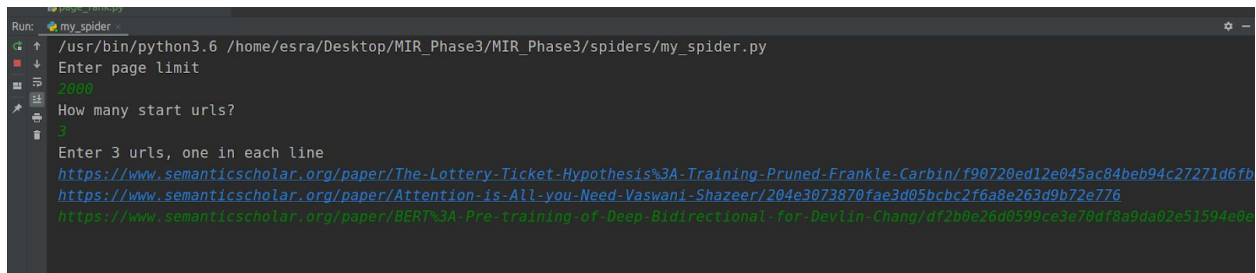
}

دارد، پس باید نویسندگان را جدا کنیم که برای این کار از تابع `find_author` استفاده می‌کنیم. هم چنین سعی می‌کنیم چیزی که به عنوان `year` از همین رشته‌ی بالایی بدست آوردیم را به عدد تبدیل کنیم. اگر یک مقدار عددی نبود عدد ۱۹۴۰ را به عنوان سال چاپ مقاله در نظر می‌گیریم چون مقالاتی که `date` تعریف شده دارند معمولاً بعد از ۱۹۴۰ منتشر شده‌اند.

بعد هم فیلدهای آیتم را در فایل `items.py` تعریف می‌کنیم تا مقادیری که بدست آوردیم در آیتم بریزیم و آنها را `yield` می‌کنیم.

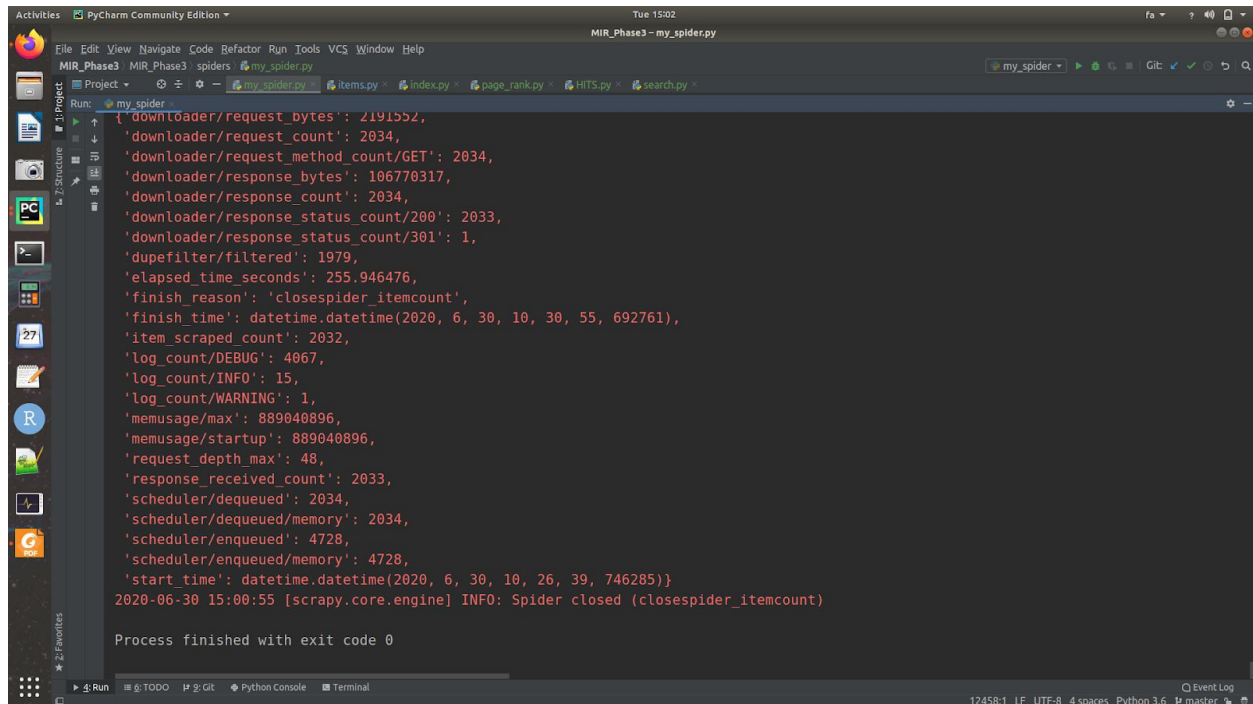
```
class MirPhase3Item(scrapy.Item):
    id = scrapy.Field()
    title = scrapy.Field()
    authors = scrapy.Field()
    date = scrapy.Field()
    abstract = scrapy.Field()
    references = scrapy.Field()
```

فرمت ورودی:



```
Run: my_spider
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase3/MIR_Phase3/spiders/my_spider.py
Enter page limit
2000
How many start urls?
3
Enter 3 urls, one in each line
https://www.semanticscholar.org/paper/The-Lottery-Ticket-Hypothesis%3A-Training-Pruned-Frankle-Carbin/f90720ed12e045ac84beb94c27271d6fb
https://www.semanticscholar.org/paper/Attention-is-All-you-Need-Vaswani-Shazeer/204e3073870fae3d05bcb2f6a8e263d9b72e776
https://www.semanticscholar.org/paper/BERT%3A-Pre-training-of-Deep-Bidirectional-for-Devlin-Chang/d12b0e26d0599ce3e70df8a9da02e51594e0a
```

خروجی:



```
{ 'downloader/request_bytes': 2191552,
  'downloader/request_count': 2034,
  'downloader/request_method_count/GET': 2034,
  'downloader/response_bytes': 106770317,
  'downloader/response_count': 2034,
  'downloader/response_status_count/200': 2033,
  'downloader/response_status_count/301': 1,
  'dupefilter/filtered': 1979,
  'elapsed_time_seconds': 255.946476,
  'finish_reason': 'closespider_itemcount',
  'finish_time': datetime.datetime(2020, 6, 30, 10, 30, 55, 692761),
  'item_scraped_count': 2032,
  'log_count/DEBUG': 4067,
  'log_count/INFO': 15,
  'log_count/WARNING': 1,
  'memusage/max': 889040896,
  'memusage/startup': 889040896,
  'request_depth_max': 48,
  'response_received_count': 2033,
  'scheduler/dequeued': 2034,
  'scheduler/dequeued/memory': 2034,
  'scheduler/enqueued': 4728,
  'scheduler/enqueued/memory': 4728,
  'start_time': datetime.datetime(2020, 6, 30, 10, 26, 39, 746285)}
2020-06-30 15:00:55 [scrapy.core.engine] INFO: Spider closed (closespider_itemcount)

Process finished with exit code 0
```

فرمت آیتم خروجی:

```
{"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. \n
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 problems
with much smaller networks.", "references":
["34f25a8704614163c4095b3ee2fc969b60de4698",
"1ff9a37d766e3a4f39757f5e1b235a42dacf18ff",
"b0bd441a0cc04cdd0d0e469fe4c5184ee148a97d",
"cc46229a7c47f485e090857cbab6e6bf68c09811",
"642d0f49b7826adcf986616f4af77e736229990f",
"049fd80f52c0b1fa4d532945d95a24734b62bdf3",
"2dfef5635c8c44431ca3576081e6cfe6d65d4862",
"397de65a9a815ec39b3704a79341d687205bc80a",
"c2a1cb1612ba21e067a5c3ba478a8d73b796b77a",
"e8eaf8aedb495b6ae0e174eea11e3cfcdf4a3724"]}]
```

بخش دوم: ایندکس کردن

مسالهای خاصی ندارد. هاست و پورت را می‌گیریم و یک شیء Elasticsearch می‌سازیم. آیتم‌ها را از فایل می‌خوانیم و به صورت bulk ایندکس می‌کنیم. برای حذف ایندکس هم باز یک شیء Elasticsearch می‌سازیم و با آن ایندکس را پاک می‌کنیم. (من فرض کردم قبل از ایندکس کردن مرحله‌ی ۳ را انجام می‌دهیم)

بخش سوم: pagerank

یک old score و یک new score در نظر می‌گیریم. ابتدا new score را برای همه‌ی آیتم‌ها برابر یک مقدار کوچک می‌گیریم و old score را صفر می‌گیریم. بعد در هر مرحله این کار را می‌کنیم:

چک می‌کنیم که جمع اختلافات old score و new score برای هر آیتم، از یک مقداری (که ما ۰.۰۰۱ گذاشتیم) کمتر نشود. اگر کمتر شد همان new score را برمی‌گردانیم و کار تابع تمام می‌شود. اگر نبود old score را new score می‌گذاریم و new score را صفر می‌گذاریم و با این فرمول، new score را برای هر آیتم محاسبه می‌کنیم:

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

i.e. the PageRank value for a page **u** is dependent on the PageRank values for each page **v** contained in the set **B<sub>u</sub>** (the set containing all pages linking to page **u**), divided by the number  $L(v)$  of links from page **v**.

آخر حلقه هم مقدار new score برای هر صفحه را به جمع مقادیر new score آیتم‌ها تقسیم می‌کنیم تا نرمالایز شود و بتوان آن را همگرا کرد.

```
for k, v in new_score_list.items():  
    v /= total
```

بعد هم new score هر آیتم را به عنوان یک فیلد page\_rank به آن آیتم در لیست آیتم‌ها اضافه می‌کنیم.

```
for d in dict_list:  
    d.update({"page_rank": scores[d["id"]]}))
```

بعد این لیست جدید را در فایل ranked\_results.json می‌ریزیم

#### بخش ۴: سرچ

در این بخش نکته غیر بدیهی اصلی نحوه امتیازدهی (function score) در فرمت بدنه‌ی کوئری است. Function score از دو بخش تشکیل شده: query و script score. Query، امتیاز داک را نسبت به پرسمان حساب می‌کند. چون کلمات پرسمان باید هم در قسمت title و هم در قسمت abstract جستجو شوند از نوع تابع multi match استفاده می‌کنیم و از نوع cross section. برای اینکه وزن‌های title و abstract با هم فرق کند، وزن آنها را در قسمت fields در multi match وارد می‌کنیم.

Script score: نمره‌ی جدید بودن سند و رنک بالا بودن سند را اینجا محاسبه می‌کنیم. زبان اسکریپت mvel است که خیلی شبیه java است. برای داک‌هایی که از سال خواسته شده در query جدیدترند، یک وزن قرار دادم که اگر داک از آن سال جدیدتر باشد، نمره‌اش در آن وزن ضرب می‌شود. طبیعتاً جمع هم می‌توان زد و ضرب کردن وزن date را در نمره‌دهی بیشتر می‌کند تا ضرب. راه دیگر هم این بود که از decay استفاده کنیم اما decay به صورت پله‌ای به داک‌ها بر اساس تاریخشان امتیاز می‌داد و بنابراین اینکه پرسمان تاریخ چه باشد خیلی اهمیتی پیدا نمی‌کرد.

برای تاثیر دادن page\_rank هم یک page\_rank\_coefficient تعریف می‌کنیم که اگر قرار بود page\_rank تاثیر داده شود برابر یک است و اگر قرار نبود تاثیر داده شود 0 می‌شود. این وزن ضرب در لگاریتم page\_rank در امتیاز داک ضرب می‌شود. (البته قانونی وجود دارد که امتیازها نباید منفی شوند و page\_rank‌ها چون نرمال شده‌اند و کوچکتر از ۱ اند لگاریتمشان منفی می‌شود که برای همین این لگاریتم را با ۱۰ جمع می‌کنیم.)

نکته‌ی آخر هم collapse است که جواب کوئری را طوری فیلتر می‌کند که از فیلدی که به آن داده‌ایم از هر مقدار آن فیلد تنها یک آیتم در جواب سرچ وارد شود. فیلد id را به آن پاس دادیم تا از آنهایی که آیدی یکسان دارند یکی را بیاورد و duplicate‌ها را حذف کند.

```
ch = {
  "query": {
    "function_score": {
      "query": {
        "multi_match": {
          "query": query_string,
          "type": "most_fields",
          "fields": ["abstract^" + str(abstract_weight), "title^" + str(title_weight)]
        }
      },
      "script_score": {
        "script": {
          "params": {
            "date_weight": date_weight,
            "date_query": date_query,
            "pg_rank": page_rank_coefficient
          },
          "source": " _score * (params.pg_rank > 0 ? Math.log(params.pg_rank * (doc['page_rank'].value + 1) * ((doc['date'].value >= params.date_query) && params.date_weight > 1 ? params.date_weight : 1)) : 1 )"
        }
      }
    }
  },
  "collapse": {
    "field": "id.keyword"
  }
}
```

قالب ورودی:

```
run: search
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase3/MIR_Phase3/search.py
Enter query string:
gene expression profiling
Enter the year after which the retrieved docs should have been published:
2000
Found 219 results. Top 93 results are returned.
8255d1c48c8d100985f7959c49989af9389db51f Knowledge-based analysis of microarray gene expression 2000 page rank: 0.00016420361247947453 authors: ['M.P.S. Br
72d7c465ef199a9670b3da7a318b0227f5cc3229 Visual Referring Expression Recognition: What Do Systems Actually Learn? 2018 page rank: 0.00016420361247947453 auth
7385d651417afa19f9e8b75b071e8144b04621a8 Gene expression and localization of opioid peptides in immune cells of inflamed tissue: Functional role in antinocicept
d623fbed40fb161b1cfa391f8ed27520ed3e226a An Integral Expression for the Log Likelihood Ratio of Two Gaussian Processes 1966 page rank: 9.852216748768472e-05 a
d6fc5ad5499644b41b80d58f4ce50c85f0d6175d Calcitonin gene-related peptide, substance P and nitric oxide are involved in cutaneous inflammation following ultravio
f1dc71148be5a73b35726dea4e6b8fd235185c78 Cloning and expression of an isoform of the rat mu opioid receptor (rMOR1B) which differs in agonist induced desensitiz
8faad7901db9a73cacaf92ecddebace87d95f92 Kernel k-means: spectral clustering and normalized cuts 2004 page rank: 0.00016420361247947453 authors: ['Inderjit
b35e340a4356eb8953b04c17375fffb631da68916 Learning with Compositional Semantics as Structural Inference for Subsentential Sentiment Analysis 2008 page rank:
df5056049a1ee83f7b55d012dfe39cf113d05d17 Reducibility among combinatorial problems" in complexity of computer computations 1972 page rank: 8.21422704123542e-6
b4b8964e55d1b1b5df38ee2df7f6a490f3fb05d8 The analgesic effect of oral delta-9-tetrahydrocannabinol (THC), morphine, and a THC-morphine combination in healthy su
9930563ac8c8c21ac30d1e534c03c41002d87c399 Considerations on models of movement detection 2004 page rank: 0.00016420361247947453 authors: ['Tomaso Poggio', '
08513358dbf09b8d4ae83741c575756a5b944cd0 USFD2: Annotating Temporal Expressions and TLINKS for TempEval-2 2010 page rank: 0.0004926108374384236 authors: ['Le
USFD2 identifies and anchors temporal expressions, and also attempts two of the four temporal relation assignment tasks. A rule-based system picks out and anchor
ae3e2451491f7d6ealeec587fd8c811b4200c07 Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks 2015 page rank: 0.00012297097884899
c42a74f01ec26f1f8e5346e197e1d5af65f7e85 mu Opioid receptor knockout in mice: effects on ligand-induced analgesia and morphine lethality. 1998 page rank: 0.0
398440707f5b0d57c02710199a7fc2d28e934150 Pseudo-Word for Phrase-Based Machine Translation 2010 page rank: 7.037297677691766e-05 authors: ['Xiangyu Duan', 'Mi
72b393f791ecf492938cc22144f46d9ecacbb0a IBAL: A Probabilistic Rational Programming Language 2001 page rank: 0.0002463054187192118 authors: ['Avi Pfeffer']
8829e3873846c6bbad5ac11e64f9d2c1b24299 Deep Sequential Neural Network 2014 page rank: 9.857072449482504e-05 authors: ['Ludovic Denoyer', 'Patrick Gallin
d84b57362e2010f6f65357267df7e0157af30684 Distant supervision for relation extraction without labeled data 2009 page rank: 0.0007378258730939498 authors: ['Mi
89d5b41b7fb0a122f811be270e6d5f72fc59d680 PANDA: Pose Aligned Networks for Deep Attribute Modeling 2014 page rank: 0.00012297097884899163 authors: ['Ning Zha
a509fdc4690ee441221f3623d546613d7b9a9876 From Regular Expressions to Deterministic Automata 1986 page rank: 0.0002464268112370626 authors: ['G\\\'e]rard Be
```

حالا اینجا date weight را ۴ گرفته بودیم که مقدار خوبی هم نیست (مقدار خوب برای date weight به طور شهودی باید چیزی بین ۱.۱ و ۱.۲ باشد (طبعاً به ترجیحات کاربران بستگی دارد



ولی چون دارد در score ضرب می شود اصلا منطقی نیست که اگر مقاله ای جدیدتر باشد score اش ۴ برابر شود.) می خواهیم ببینیم اگر date weight را صفر می گرفتیم چه فرقی می کرد (که در کوئری بین ۱ و date weight ماکس گرفته ایم و گزینه در این حالت همه صفر می شدند)

```
Run: search
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase3/MIR_Phase3/search.py
Enter query string:
gene expression profiling
Enter the year after which the retrieved docs should have been published:
2000
Found 219 results. Top 93 results are returned.
8255d1c48cd100985f7959c49989af9389db51f Knowledge-based analysis of microarray gene expression 2000 page rank: 0.0001642036124794
7385d651417afa19f9e8b75b071e8144b04621a8 Gene expression and localization of opioid peptides in immune cells of inflamed tissue: Functi
72d7c465ef199a9670b3da7a318b0227f5cc3229 Visual Referring Expression Recognition: What Do Systems Actually Learn? 2018 page rank: 0.
d623fbed40fb161b1cfa391f8ed27520ed3e226a An Integral Expression for the Log Likelihood Ratio of Two Gaussian Processes 1966 page rank
d6fc5ad5499644b41b80d58f4ce50c85f0d6175d Calcitonin gene-related peptide, substance P and nitric oxide are involved in cutaneous inflam
fldc71148be5a73b35726dea4e6b0fd235185c78 Cloning and expression of an isoform of the rat mu opioid receptor (rMOR1B) which differs in a
df5056049alee83f7b55d012dfe39cf113d05d17 Reducibility among combinatorial problems" in complexity of computer computations 1972 page
8faad7901db9a73cacaf92ecdcdbaece87d95f92 Kernel k-means: spectral clustering and normalized cuts 2004 page rank: 0.0001642036124794
c42a74f01ec26f1f8e5346e197e1d55af65f7e85 mu Opioid receptor knockout in mice: effects on ligand-induced analgesia and morphine lethality
a509fdc4690ee1441221f3623d546613d7ba9876 From Regular Expressions to Deterministic Automata 1986 page rank: 0.0002462468112370626

Derivatives of regular expressions correspond to state transitions in finite automata. When a finite automaton makes a transition under

Marking of regular expressions yields an expression with distinct input symbols. Following McNaughton and Yamada (1960), we attach subs
eae275046b909dec7a062a35862376c750e60463 The awk programming language 1988 page rank: 0.0002463054187192118 authors: ['Alfred V. Aho
b35e340a4356be8953b04c17375ff6b31da68916 Learning with Compositional Semantics as Structural Inference for Subsentential Sentiment Anal
75ffaa9fd92e72c10c046f2cab543d46d3b25a35 FORMALIZATION OF PROPERTIES OF PROGRAMS 1968 page rank: 5.473453749315818e-05 authors: ['Z
0d509454804e4ec625f70904e27cb93ce9dcbb2 The list set generator: a construct for evaluating set expressions 1970 page rank: 4.9261
1cec3715c4e869dfbc718d361f5ffcd09396fd9 Lower bounds on the size of Boolean formulas: Preliminary Report 1975 page rank: 8.21018062
9ab533eaf9d9aef09d7dba2747fa786b1003d3e0 Retention of heroin and morphine-6 $\beta$ -glucuronide analgesia in a new line of mice lacking exon 1
9ab533eaf9d9aef09d7dba2747fa786b1003d3e0 The analgesic effect of oral delta-9-tetrahydrocannabinol (THC), morphine, and a THC-morphine
9ab533eaf9d9aef09d7dba2747fa786b1003d3e0
```

مشاهده می کنیم به ازای همان کوئری مقالات قدیمی تری در نتایج اول ظاهر می شوند.

حالا تاثیر وزن title را بررسی می کنیم. date weight را ۱.۱ می گذاریم که اثر بقیه را از بین نبرد،  
 title weight = 1 است و abstract weight = 2

```
Run: search
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase3/MIR_Phase3/search.py
Enter query string:
Turing Machine
Enter the year after which the retrieved docs should have been published:
2010
Found 1511 results. Top 100 results are returned.
ef2aa11a9e5dac4577a90b65978f894fa3f4b193 A Universal Turing Machine with Two Internal States 1956 page rank: 0.0002463054187192118
e77e131297b22e49ad034bf8ab0abd44ec14d4f0 On the Inference of Turing Machines from Sample Computations 1972 page rank: 0.000164284540
e5e44b3ee7a44b56212b1f19a8df042e4e1617cd Computational Complexity of One-Tape Turing Machine Computations 1968 page rank: 0.00030103
cf91dd4d5358df1c6ba873360a94a469c6ed6ba5c Two tapes versus one for off-line Turing machines 1993 page rank: 5.473453749315818e-05 auth
c3823aacea60bc1f2cabb9283144690a3d015db5 Neural Turing Machines 2014 page rank: 0.00016420361247947453 authors: ['Alex Graves', 'G
ff433838bb2b178e1d6b1f045b0215385e1757f5 The Inversion of Functions Defined by Turing Machines 1956 page rank: 0.0005421389847215378
c70cd87b7b4c0b68c81cb71999eba87c18d80df8 Two tapes are better than one for off-line Turing machines 1987 page rank: 6.157635467980
03b57770387aee919b1f5c89333ee65a426f32e2 Speed-Up of Turing Machines with One Work Tape and a Two-Way Input Tape 1987 page rank: 7.
39d5978ca6236fe4b9b76cd424c20ebb00afbc3b A Study of Logic and Programming via Turing Machines Universal Turing Machine 2011 page rank
9f2e0a85f123d699e667a958fd14895b99ce36a8 An Optimal Lower Bound for Turing Machines with One Work Tape and a Two- way Input Tape 19
4b5a9490e85b90925a28079e541029bdc3561bf2 On Relating Time and Space to Size and Depth 1977 page rank: 9.852216748768472e-05 authors:
bb461389958dfdcfa9272ab4bde57924e58c9af5 Tape-Reversal Bounded Turing Machine Computations 1968 page rank: 5.473453749315818e-05 auth
75caeb5274630bd52cbcd8f549237c30d108e2ff Quantum complexity theory 1993 page rank: 0.00016420361247947453 authors: ['Ethan Bernstein'
We also consider the precision to which the transition amplitudes of a quantum Turing Machine need to be specified. We prove that O(log
We give the first evidence indicating that quantum Turing Machines are more powerful than classical probabilistic Turing Machines. We s
We also give evidence suggesting that quantum Turing Machines cannot efficiently solve all of NP. Specifically, we prove that relative
3da7e3419d879fad9b02e993758a40e4d205d0e8 The Chemical Bases of Morphogenesis (Reprinted in AM Turing 2011 page rank: 0.000287356321
6407680b5b51bccc50e62eaf94ab036a4d7eaa9 One-Tape, Off-Line Turing Machine Computations 1965 page rank: 0.0007252326217843457 auth
410ea9b1ec562f05d52b75cef1fc9aa231bd52e Tape Reversal Complexity Hierarchies 1968 page rank: 0.0002463054187192118 authors: ['Patri
2842d53d0f2389618f8ad1c47d2ffef2c344d22f Computational complexity of probabilistic Turing machines 1974 page rank: 0.0005424583626554
42ccba1ecff310c1cd022d63c169bae2491326 Matching Upper and Lower Bounds for Simulation of Several Tapes on One Multidimensional Tape
8a91d094f89b1d2e318b494ad0198ab072f53d1e On Formalisms for Turing Machines 1965 page rank: 5.473453749315818e-05 authors: ['Patrick C
```

که مثلا Quantum Complexity Theory و On Relating Time and Space to Size and Depth عنوانشان Turing Machine ندارند.

حالا اگر title weight را ۱۰۰ بگذاریم:

```
Project search.py
Run: search.py
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase3/MIR_Phase3/search.py
Enter query string:
Turing Machine
Enter the year after which the retrieved docs should have been published:
2010
Found 1511 results. Top 100 results are returned.
7cd4f36092bbd6872303282479754911160646a4 Collected Works of Alan Turing Morphogenesis Universal Turing Machine 2011 page rank: 0.0007
89585cf6859fbaba66806176d19c53fb1395090e Mathematical theory of ENIGMA machine Universal Turing Machine 2011 page rank: 0.00079501
39d5978ca6236fe4b9b76cd424c20ebb00afbc3b A Study of Logic and Programming via Turing Machines Universal Turing Machine 2011 page rank
bb461389958dfdcfa9272ab4bde57924e58c9af5 Tape-Reversal Bounded Turing Machine Computations 1968 page rank: 5.473453749315818e-05 auth
52f2febfb8568dbb995c67e83e8299680f2cecb79 Can a Machine Think ? The World of Mathematics Universal Turing Machine 2011 page rank: 0.
6407680b5b551bccf50e62eaf94ab036a4d7eaa9 One-Tape, Off-Line Turing Machine Computations 1965 page rank: 0.0007252326217843457 auth
58298f9c4e64e29ac00d70d34c1909fclab899d3 On Computable Numbers. . . Proc Universal Turing Machine 1940 page rank: 0.0012397372742200
8dea163a55333ac0b28963fd50efc509bfa789a Computer machinery and intelligence Universal Turing Machine 1940 page rank: 0.001098600359
7b7bb5c2b9ca333681418bc3f9a99c9bd2a60f07 Turing machine recognizers for general rewriting systems 1964 page rank: 7.037297677691766e
de83e862e346bclcc80b22671a9611b48f300aa4 Intelligent machinery ( Written in 1947 . ) Universal Turing Machine 2011 page rank: 0.0014
f9aa451310e865d812514573e665c607a555c8fa Intelligent Machinery : A Heretical View ' Universal Turing Machine 2011 page rank: 0.0008
c47f9d2da6da7a44035fa3c3f13447ac924c450c Handwritten essay : Nature of Spirit Universal Turing Machine 2011 page rank: 0.000477558839
98576f4b2df33503c19b82c11f28fdcc2a4c41dc On the Gaussian error function Universal Turing Machine 2011 page rank: 0.0004761904761904
07dd65bd4eb09525e75a7208494bb4108a58a406 The Automatic Computing Engine : Papers by Alan Turing and Michael Woodger Universal Turing Ma
ef2aa11a9e5d4c4577a90b65978f894fa3f4b193 A Universal Turing Machine with Two Internal States 1956 page rank: 0.0002463054187192118
e5e44b3ee7a44b56212b1f19a8df042e4e1617cd Computational Complexity of One-Tape Turing Machine Computations 1968 page rank: 0.00030103
e127369c02476cfa65184dff4ca02a66e419f263 Systems of logic defined by ordinals Universal Turing Machine 2011 page rank: 0.001046798029
6e7247eee00b3879ab9011e2069b9a1b9d38ab12 Alan Turing-father of Modern Computer Science Father of Modern Computer Science Universal Turi
6d34169cc65cf76311b57add12b4eece5ca530f47 Lecture to the London mathematical society Universal Turing Machine 2011 page rank: 0.0012
7a6100e739be584e95bdcf02a6e6eb581462da14 The chemical theory of 185 . morphogenesis Universal Turing Machine 2011 page rank: 0.0009
b948b8efa46927fd1f7c2d34ff4e659620ddc5f5 Collected Works : Mathematical Logic Amsterdam etc Universal Turing Machine 2011 page rank
d8bf54ba7ddab2e1f6bc9b90daecb274eedb730 Rounding-o errors in matrix processes Universal Turing Machine 2011 page rank: 0.00037219
114:1 LF UTF-8 4 spaces Python 3.6 P master
```

می بینیم که تقریباً همه‌ی عناوین اول در خود Turing Machine را دارند.  
بخش ۵: HITS

ابتدا دیکشنری صفحات را می‌سازیم تا صفحات را بتوانیم از روی id نشان پیدا کنیم. بعد یک دیکشنری می‌سازیم که کلیدهای آن نویسندگان هستند و مقادیر آن لیست نویسندگانی هستند که یک مقاله‌ی نویسنده‌ای که کلید است به مقاله‌ی آن‌ها ارجاع دارد (تکراری هم اضافه می‌کنیم). با iterate کردن روی لیست آیتم‌ها و پیدا کردن ارجاع‌هایشان، این دیکشنری را پر می‌کنیم.

برای نویسندگان صفحاتی که آیتم آنها وجود دارد هم سه تا دیکشنری می‌سازیم: hlast یا hub score قدیمی، h یا hub score جدید، a یا auth score.

مشابه pagerank در هر مرحله چک می‌کنیم که جمع اختلافات hlast و h برای همه‌ی نویسندگان موجود، از یک آستانه‌ای که تعیین کرده‌ایم بیشتر نشود. اگر بیشتر شد، دیکشنری a را به عنوان auth score های نهایی برمی‌گردانیم. اگر بیشتر نشد، ابتدا h را در hlast می‌ریزیم و سپس hlast را صفر می‌کنیم. بعد a را به این فرمول از روی h مرحله‌ی قبل آپدیت می‌کنیم:

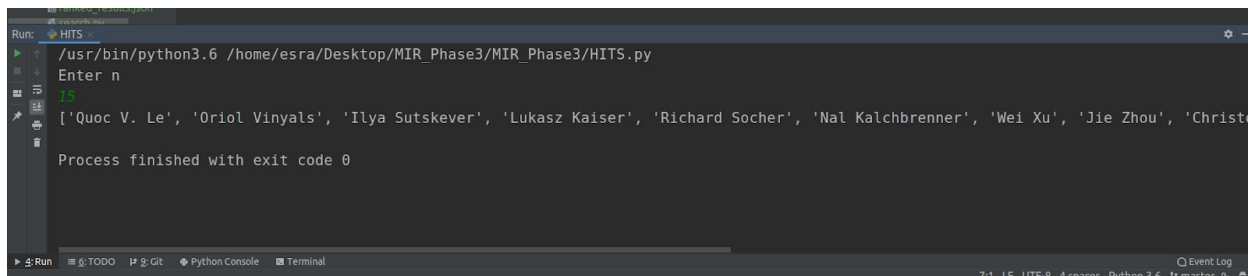
$$Authority(V_i) = \sum_{V_j \in In(V_i)} e_{ji} \cdot Hub(V_j) \quad (1)$$



بعد هم h را با این فرمول از روی a جدید آپدیت می‌کنیم.

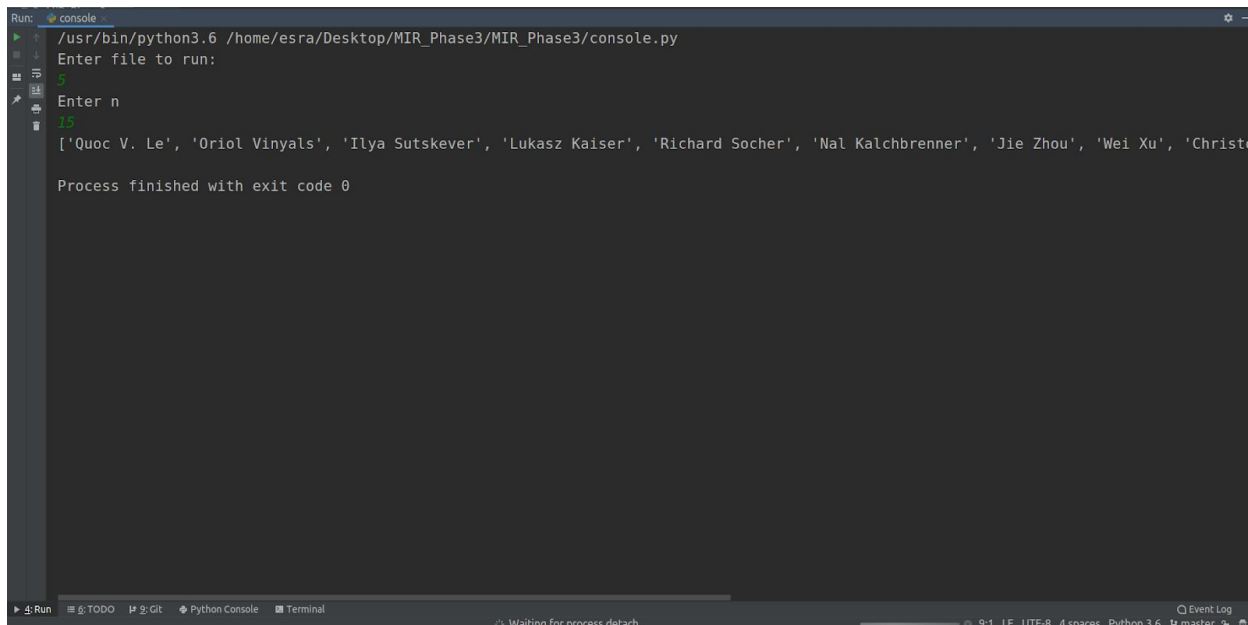
$$Hub(V_i) = \sum_{V_j \in Out(V_i)} e_{ij} \cdot Authority(V_j) \quad (2)$$

بعد هم همه‌ی مقادیر a و h را به مجموع مقادیرشان تقسیم می‌کنیم تا نرمالایز شود و بتواند همگرا شود. بعد هم که شرط همگرایی برقرار شد n کلید اول دیکشنری a را که بزرگترین مقادیر را دارند به عنوان نویسندگان برتر برمی‌گردانیم.



```
Run: HITS
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase3/MIR_Phase3/HITS.py
Enter n
15
['Quoc V. Le', 'Oriol Vinyals', 'Ilya Sutskever', 'Lukasz Kaiser', 'Richard Socher', 'Nal Kalchbrenner', 'Wei Xu', 'Jie Zhou', 'Christo
Process finished with exit code 0
```

یک فایل console.py هست که با ران کردن آن، ابتدا شماره بخشی که کد آن باید ران شود را می‌گیرد، بعد کد آن بخش را ران می‌کند. نمونه ورودی (مثلا به ازای ۵، کد بخش ۵ یعنی HITS را ران می‌کند):



```
Run: console
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase3/MIR_Phase3/console.py
Enter file to run:
5
Enter n
15
['Quoc V. Le', 'Oriol Vinyals', 'Ilya Sutskever', 'Lukasz Kaiser', 'Richard Socher', 'Nal Kalchbrenner', 'Jie Zhou', 'Wei Xu', 'Christo
Process finished with exit code 0
```

به طور مشابه برای سرچ:

```
Run: console
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase3/MIR_Phase3/console.py
Enter file to run:
4
Enter query string:
temperature control
Enter the year after which the retrieved docs should have been published:
2000
Found 1700 results. Top 100 results are returned.
0bd78f04a3a1b212560f59dba77fb3f170211b93 Temperature Discrimination in the Skin 1964 page rank: 4.9188391539596654e-05 authors: ['
14f993a2899cd3f36d3358d5f56b5a800c914c30 Temperature-dependence of resistance at an electrotonic synapse. 1969 page rank: 8.21018062
9f1e1028371cfcc72fc2b784e3bbf2ea4a8c7e8e The effect of temperature on the melanophores of fishes 1940 page rank: 0.0002463054187192
38851fd21fdb97bc94fea460fd9f196ef74987f3 Cord cells responding to touch, damage, and temperature of skin. 1960 page rank: 4.91883915
968cf694928893db751405d1e30a85722c951232 Multiple Temperature-Sensitive Spots Innervated by Single Nerve Fibers 1967 page rank: 4.
3b768e90b4e81a44bf13a510603a428f5fb1762a Adaptive control systems 1959 page rank: 0.0004926108374384236 authors: ['E. Mishkin', 'Lud
8a205615577a32de2af924f01fa1bc6148ecb403 Pattern recognizing control systems 1964 page rank: 9.852216748768472e-05 authors: ['Berna
237f33308e8e9dc794e56307649155e6aa7a5882 Control structures for programming languages 1970 page rank: 5.473453749315818e-05 authors:
1c9354082fbbfe728cf4ba17662e01541dd208d7 Abstraction in Control Learning 1992 page rank: 0.0002463054187192118 authors: ['Richard Y
dbe387b8bfbcb112bda3bd1a561d595fda73cd0ba On Fuzzy Mapping and Control 1972 page rank: 6.157635467980295e-05 authors: ['Sheldon S. L.
4716f0d81346698ae6aeccc5a7afd05f6f86d46f The excitatory and recovery processes in the nerve fibre as modified by temperature changes
1. An exponentially rising current excites an isolated single nerve fibre when, and only when, it rises above and crosses the rheobase
2.
2. A brief subthreshold shock produces an excitatory state which first rises and then falls after the end of the shock. The time course
3.
3. At every temperature, the earliest return of excitability occurs immediately after the end of the spike. The process of recovery is
e4930af7e0ad2ebf41730dc892204894a08f3d9 TEMPERATURE COEFFICIENT OF THE ACTION OF  $\beta$ -RAYS UPON THE EGG OF NEREIS 1940 page rank: 9.
de230cfbbafbf0d0837e1dd629a79b14e78675 On the Control of Control: The Role of Dopamine in Regulating Prefrontal Function and Working
```

بخش ۶: Ranking\_svm

تابع `read_data` داده‌ها را از فایل می‌خواند و در آرایه‌ای می‌ریزد که در آن هر عنصر، یک خط از فایل است. پارامتر `z` در آن تعداد `fold` هایی است که می‌خواند. این کار را برای `train`, `vali`, `test` انجام می‌دهیم.

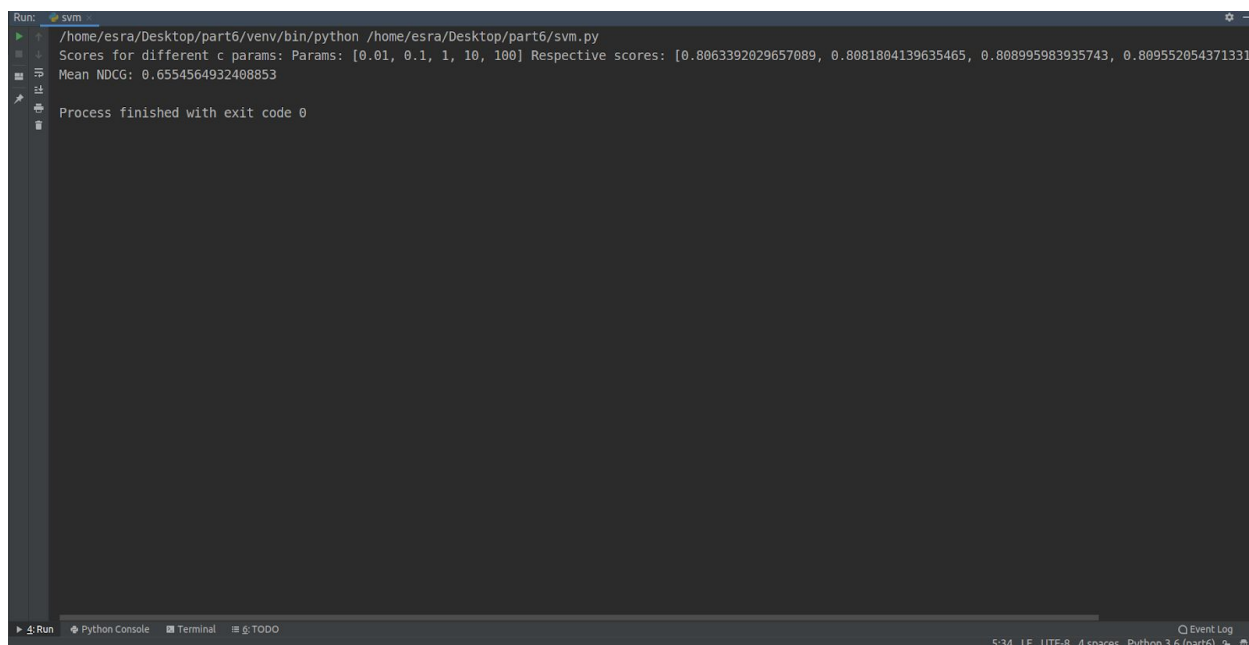
تابع `structure_data` داده‌ها را به صورت دو دیکشنری از دیکشنری‌ها در می‌آورد. به این صورت که دیکشنری اول بردارهای ویژگی‌ها را در خود دارد و دیکشنری دوم امتیازها (`relevance` ها) یعنی مثلاً `[x[query_id]][doc_id]` یک بردار ۴۶ تایی است که ویژگی‌های داک `doc_id` نسبت به کوئری `query_id` است.

تابع `pairwise_transition` این بردارها و برچسب‌ها (`relevance` ها) را می‌گیرد و برای هر کوئری تفاضل هر دو داده (هم تفاضل بردارها و هم تفاضل برچسب‌ها) را حساب می‌کند و در دو دیکشنری جدید می‌ریزد و حساب می‌کند. این کار را برای داده‌های `train` و `Validation` انجام می‌دهیم.

روی داده‌های `train` حاصل از `pairwise_transition` مدل‌مان را `fit` می‌کنیم و سپس `score` مدل در پیش بینی داده‌ی `validation` حاصل از `pairwise_transition` را محاسبه می‌کنیم. این کار را به ازای 5 مقدار `c` مختلف (۰.۰۱، ۰.۱، ۱، ۱۰، ۱۰۰) انجام می‌دهیم و سرانجام `c` را انتخاب می‌کنیم که بیشترین `score` را می‌دهد. این `c` به ازای هر `fold` ۵ با هم ۱ شد. بنابراین از `svm` با `c = 1` برای پیش‌بینی داده‌های تست استفاده می‌کنیم.

وقتی svm را با  $c = 1$  روی داده‌ی train، فیت کردیم، بردار وزنهای آن را استخراج می‌کنیم (w). بعد به ازای هر کوئری‌ای که در داده‌ی تست داریم این کار را انجام می‌دهیم:

به ازای هر داک موجود در کوئری، بردار ویژگیهای آن را با بردار وزنهای svm ضرب داخلی می‌کنیم و این مقدار ضرب داخلی را به عنوان score پیش بینی شده‌ی آن داک به ازای آن کوئری در آرایه‌ای می‌ریزیم. Score واقعی را هم که همان relevance است در آرایه‌ای می‌ریزیم. سپس روی این دو آرایه تابع ndcg را صدا می‌زنیم. چون ممکن است داک‌های هر کوئری از ۵ کمتر باشد، برای پارامتر k در تابع ndcg بین ۵ و طول آرایه‌های score، مینیمم می‌گیریم. یک نکته‌ای که در نظر می‌گیریم این است که کوئری‌هایی که relevance همه‌ی داک‌هایشان یکسان است مهم نیست رنکینگ پیش‌بینی‌شده‌شان چه باشد پس آنها را در محاسبه‌ی NDCG در نظر نمی‌گیریم. نهایتاً آیدی هر کوئری را با مقدار ndcg نظیرش در یک فایل ذخیره می‌کنیم. نمونه‌ی خروجی ndcg برای ۱ fold تا ۵:



```
Run: svm
/home/esra/Desktop/part6/venv/bin/python /home/esra/Desktop/part6/svm.py
Scores for different c params: Params: [0.01, 0.1, 1, 10, 100] Respective scores: [0.8063392029657089, 0.8081804139635465, 0.808995983935743, 0.8095520543713315, 0.8095520543713315]
Mean NDCG: 0.6554564932408853
Process finished with exit code 0
```

Ndcg به تفکیک کوئری:

```
svm.py full_ndcg.txt
184 "11386": 0.6191463544777536,
185 "11387": 0.6164336326286644,
186 "11419": 0.2870204396558807,
187 "11446": 1.0,
188 "11457": 0.6934264036172708,
189 "11488": 1.0,
190 "11494": 0.20210734650054754,
191 "11495": 0.9558295932317544,
192 "11503": 0.6992148198508501,
193 "11511": 0.57064171895532,
194 "11531": 1.0,
195 "11553": 0.5703281064914795,
196 "11565": 0.6578242262397571,
197 "11577": 0.57064171895532,
198 "11581": 0.8274043288765652,
199 "11596": 0.4202371738099799,
200 "11624": 0.787702056960637,
201 "11639": 1.0,
202 "11673": 0.19595127901151166,
203 "11725": 0.8223779611386883,
204 "11739": 0.6309297535714573,
205 "11743": 0.38685280723454163,
206 "11759": 0.7543701539552119,
207 "11777": 0.8597186998521971,
208 "11828": 0.8772153153380493,
209 "11843": 0.0,
210 "11889": 0.6338006170256999,
211 "11893": 0.5745154706993483,
```

🔍 Event Log 5:34 LF UTF-8 4 spaces Python 3.6 (part6)