




# درس سیستم عامل

## مستندات پروژه اول

اعضای گروه :

- محمدحسین عشوری 97149068
- محمدمهدی افخمی 97149008
- نرگس رفیعی زاده 97149048

مشخصات سیستم:

 Memory	8 GB
 Processors	4
 Hard Disk (SCSI)	20 GB

## بیان مسئله:

فایلی متشکل از چندین سند متنی ، مسئله از ما می خواهد که در خروجی شماره ی آن سند متنی را همراه با میانگین TF-IDF خط ، محاسبه و چاپ کنیم و پیاده سازی آن باید توسط multiprocessing انجام شود یعنی استفاده از پردازش های موازی به جای اجرای تک فرایندی.

## راه حل:

باید حل این مسئله به گونه ای پیاده سازی شود که نتیجه نهایی هر فرآیند در فایل خروجی با یکدیگر تداخل نداشته باشد. با استفاده از انحصار متقابل، در زمانی که فرآیند ها در حال نوشتن روی فایل خروجی هستند با اجرا تکنیک های انحصار متقابل، این موضوع حل می شود .  
برای محاسبه TFIDF برای فایل متنی و چاپ آن در فایل count.txt ما اول باید TF و IDF را با متدها، جدا گانه موازی سازی کنیم .  
برای ایجاد فرایند ها می توان از کتابخانه multiprocessing پایتون و کلاس Pool استفاده کرد.  
برای فهمیدن اینکه چند فرایند می توان بر روی سیستم عامل ایجاد کرد میتوان از متد `cpu_count()` در کتابخانه `os` پایتون استفاده کرد .

```
pool = Pool(processes=os.cpu_count())
```

البته بیشتر از این مقدار هم می توان ایجاد کرد که در اینجا مسئله زمان سربار برای Context Switch پیش می آید . وقتی ما بیشتر از حد مجاز فرایند ایجاد میکنیم در حالی که سیستم تا یک تعداد مشخصی توانایی ایجاد فرایند را دارد باعث می شود سیستم عامل، بین فرایند ها Switch کند که این کار با توجه به اطلاعاتی که هر فرایند دارد و این اطلاعات در هر بار context switch باید لود شوند، می تواند زمانبر باشد (اطلاعاتی مانند : متغیر های لوکال و گلوبال و ...)

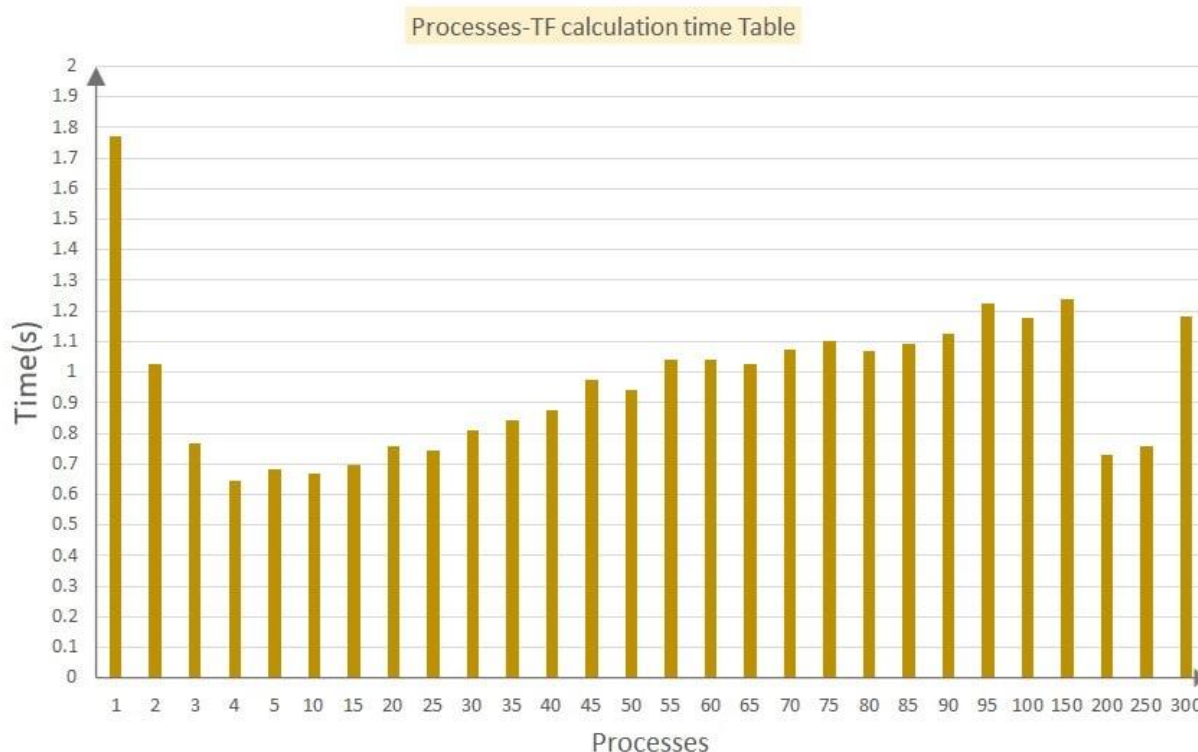
یک pool با تعداد 4 فرایند می سازیم (این فرایند ها با توجه به تعداد هسته های موجود در نظر گرفته شده است)

ورودی ما لیستی از سند ها هست ما باید متدی را که TF ها را برای هر خط حساب میکند را روی آن لیست نگاشت کنیم. برای این کار ما از متد map استفاده می کنیم .

```
tf_list = pool.map(TF_POOL, indexed_lines)
# tf_list [(index, dic_TF, len(words)) ,
           (index, dic_TF, len(words)) , ....]
```

در واقع ما 4 فرایند ساخته ایم و هر کدام از این فرایند ها این تابع را برداشته اند و هر سری یکی از سند ها را از ورودی بر می دارند و TF هارا حساب میکنند و در آخر همه مقادیر را در متغیر tf\_list ذخیره میکنند(این عملیات ممکن است ترتیبی نباشد).

محاسبه چند فرایندی TF در نمودار پایین آمده است



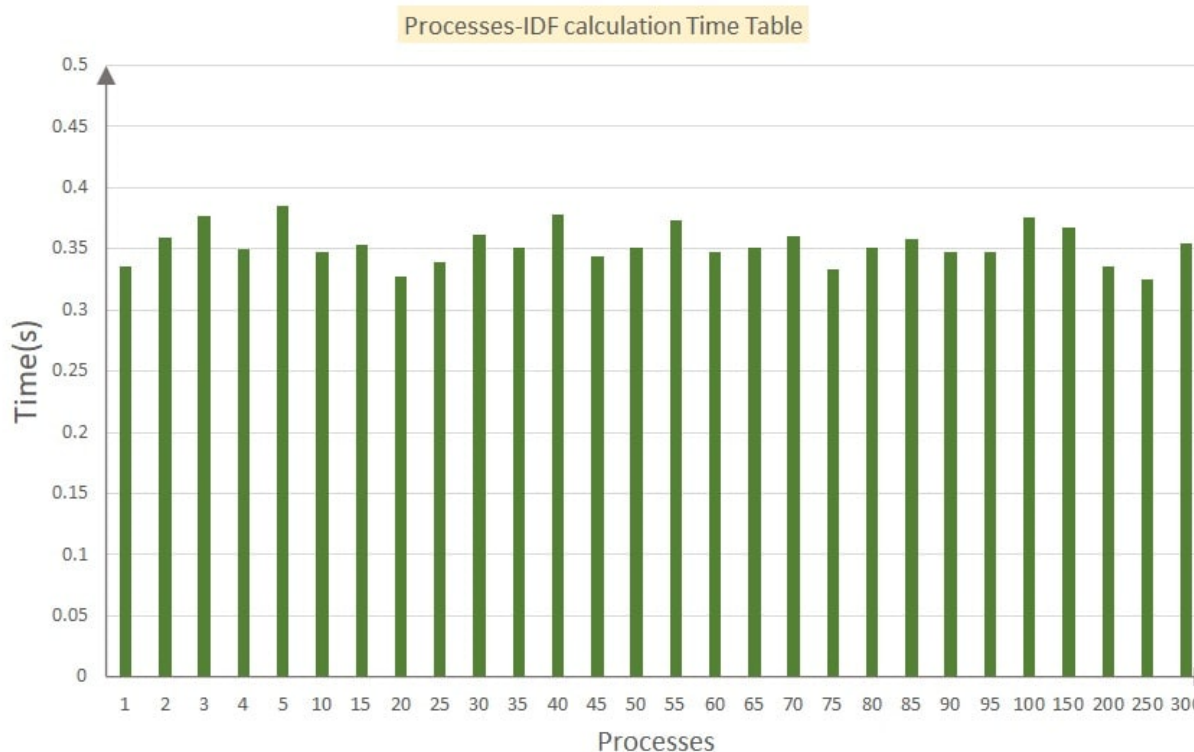
بعد از آنکه در متغیر `tf_list` ما اطاعات TF متعلق به هر خط را بدست می آوریم حال می توانیم IDF , DF را حساب کنیم.

برای حساب کردن IDF به صورت چند فرایندی ما می توانیم از متغیر های گلوبال استفاده کنیم اما از آنجایی که زمان دسترسی به متغیر گلوبال بسیار بالا است ما IDF ها را در تابع MAIN به صورت تک فرایندی و با استفاده از متغیر های لوکال بدست می آوریم و از طرفی چون IDF برای کل برنامه ثابت است پس ما آن را یکبار در تابع MAIN حساب می کنیم و بعد از محاسبه آنها یک کپی از مقادیر IDF ها به هر کدام از فرایندها می دهیم.

در قسمت بدست آوردن IDF در واقع ما به جای استفاده از متغیر گلوبال و به اشتراک گذاری آن بین همه فرایندها (که باعث کند تر شدن زمان دسترسی به متغیر میشود)، یک Queue گلوبال ساختیم و در آن به اندازه تعداد فرایندها موجود متغیر لوکالی که در آن اطلاعات IDF بود را کپی کردیم و هر زمان که هر فرایند برای بار اول اجرا شد از صف یک کپی از IDF ها را بر میدارد و تا پایان برنامه آن کپی را نگه می دارد و در ادامه برای محاسبه TFIDF هر خط از همان متغیر لوکال استفاده می کند این کار باعث افزایش سرعت دسترسی به متغیر می شود.

در واقع به جای استفاده از یک متغیر گلوبال ما از متغیر های لوکال استفاده می کنیم و به هر فرایند می گوییم که یک کپی از این متغیر داشته باشد و تا آخر اجرای فرایند از همان استفاده کند. استفاده از این روش به این

دلیل است که ما تنها عملیات خواندن از متغیر انجام می‌دهیم و گرنه در صورتی که بخواهیم روی متغیر تغییراتی داشته باشیم و بخواهیم این تغییرات همه جا اعمال شود حتما باید از متغیرهای گلوبال استفاده کرد که کلاس Manager که از کتابخانه multiprocessing این امکان را به ما می‌دهد.



#### محاسبه تک فرایندی IDF

همانطور که در نمودار می‌بینید محاسبه IDF به این دلیل که در فرایند اصلی اجرا میشوند سرعت پردازش آن به تعداد فرایند ها وابسته نیست.

بعد از بدست آوردن IDF , TF حال زمان آن است که آنها را ضرب و میانگین را برای هر خط چاپ کنیم. از آنجایی که این عملیات تکراری هست و برای هر فرایند به صورت مستقل می‌تواند اجرا می‌شود میتوان محاسبه آن را مجدداً به فرایند های مختلف داد.

برای این کار دوباره میتوان از همان شیء pool استفاده کرد و متد map را صدا زد.

اینبار ما همان TF هارا دوباره پاس می‌دهیم و با IDF ای که در شروع فرایند ها به آنها پاس داده ایم TF\*IDF را برای هر کلمه محاسبه می‌کنیم و میانگین TFIDF را برای کل خط بدست می‌آوریم و در خروجی چاپ کنیم.

به وسیله این تابع TFIDF را برای هر خط حساب و در خروجی چاپ می کنیم :

```
def TFIDF_POOL(tf_dic):
```

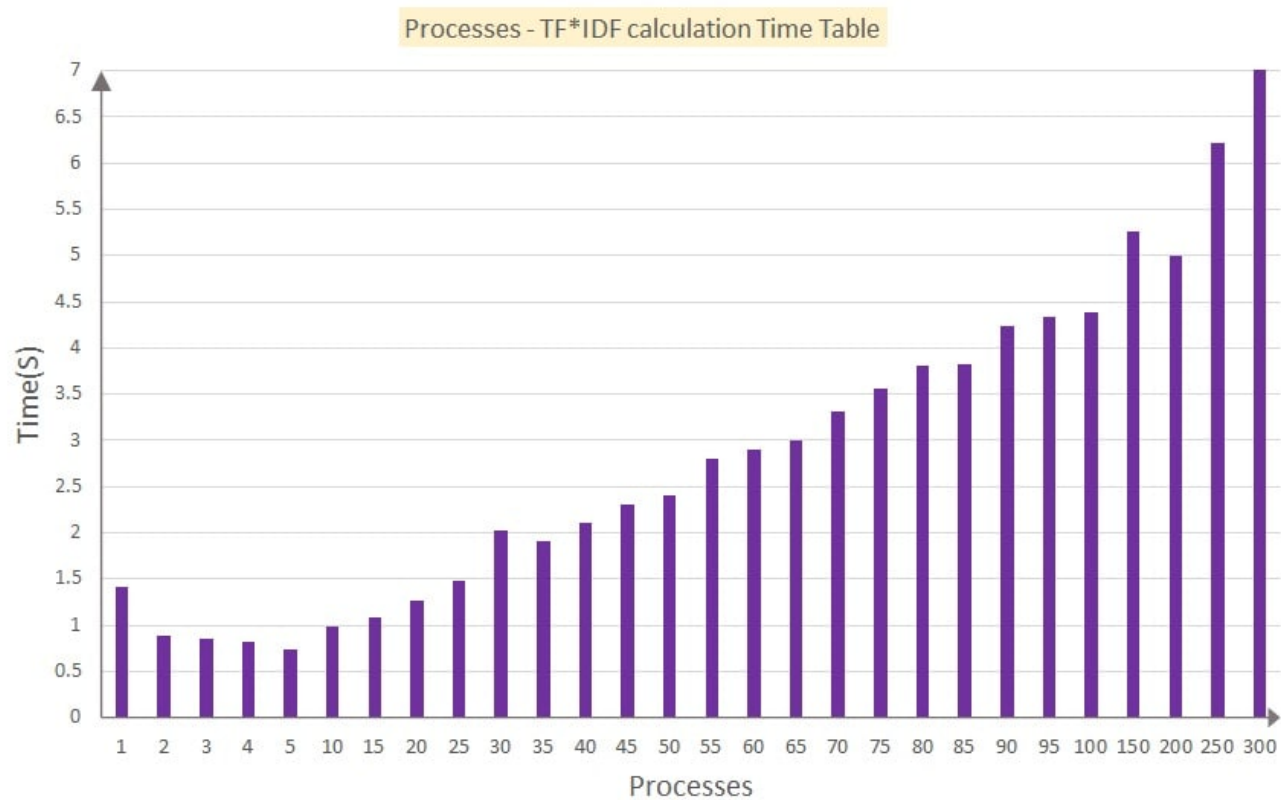
در هنگام چاپ برای ورودی های جدید ممکن است ما به مشکل روی هم نوشتن فرایند برخورد کنیم . در واقع ممکن است چند فرایند به صورت همزمان به نقطه بحرانی (در این پروژه نوشتن روی فایل نقطه بحرانی می باشد) دسترسی داشته باشند که این اتفاق باعث می شود بعضی از خطوطی که در خروجی چاپ میکنیم با یکدیگر تداخل داشته باشند .

برای جلوگیری از این موضوع میتوانیم از ابزار هایی که پایتون و کتابخانه multiprocessing و کلاس Lock استفاده کنیم تا انحصار متقابل را برای فرایند ها ایجاد کنیم .

```
lock = Lock()

with lock:
    with open(sys.path[0] + "//count.txt", "a") as writer:
        writer.write(f"Document {index} has {tf_dic[2]} words.The\naverage of TF - IDF's {sum / float(len(tf_dic[1]))}\n")
```

با این کار بعد محاسبه میانگین  $TF*IDF$  کلمات یک خط میتوان با فراخوانی Lock که به عنوان یک متغیر گلوبال برای تمامی فرایندها است انحصار متقابل را برای فرایندها ایجاد کرد، بدین صورت که هر کدام از فرایندها که زودتر این محاسبات را انجام دادند و به نقطه بحرانی رسیدند Lock برای آنها صدا زده خواهد شد و دیگر هیچ فرایندی نمی تواند وارد شود تا زمانی که آن فرایندی که در نقطه بحرانی است خارج شود.



زمان محاسبه میانگین TFIDF یک خط + نوشتن روی خروجی به ازای تعداد فرایندهای مختلف



زمان اجرای کل به برنامه به ازای تعداد فرایندهای مختلف