مصاحبه ی عملی یکتانت

احسان برخوردار

موقعیت شغلی دانشمند داده

تابستان ۹۹

مقدمه

رشد انفجاری مقدار داده های دیجیتالی، تعداد بازدیدکنندگان اینترنتی و نیز گسترش سریع و مداوم وب گستردهٔ جهانی و تجارت الکترونیک، به مشکلی جدی در دنیای وب منجر شده بود.
سیستمهای بازیابی اطلاعات، مانند گوگل تا حدی این مشکل را حل کردند، اما اولویتبندی و شخصی سازی اطلاعات کاربر، رشد محسوسی نکرد تا اینکه، سیستم های توصیه گر برای بهبود و ایجاد پیشرفت در این مشکل پدیدار شدند.
سیستم های توصیه گر، نوعی سیستم پالایش اطلاعاتی هستند که با مشکل اضافه بار اطلاعات سروکار دارند و از طریق فیلترکردن و قطعه قطعه کردن اطلاعات حیاتی، از مقدار زیاد اطلاعات پویای تولیدشده با توجه به علاقه یا رفتار مشاهده شده کاربر در مورد آیتم ها، میکاهند. به بیان دیگر این سیستم ها با توجه به واکنشهای کاربر، میتوانند پیشبینی کنند که گزینهٔ پیشنهادی را می پذیرد یا آن را رد میکند. بر این اساس، این سیستمها با روند رو به افزایشی در وبسایتهای تجاری به کار گرفته شدند.

سیستمهای توصیه کننده هم برای خدمتگزارها و هم برای کاربران مفیدند. آنها در محیط خرید آنلاین، هزینه های معامله و انتخاب گزینه ها را کاهش میدهند. هدف اصلی این سیستم ها فراهم آوردن ابزاری است که به وسیلهٔ آن بتوانند کاربران را در یافتن سریع و مناسب اطلاعات و رفع نیازها یاری کنند.
در حال حاضر بسیاری از سایت های تجارت الکترونیک از مزیتهای این سیستم اطلاعاتی، بهره مند شده اند. برای مثال، آمازون که یکی از بزرگترین سایتهای فروشگاه اینترنتی است و افزون بر ۸۰ میلیون عضو داد، در سال ۲۰۰۶ اعلام کرد که ۳۵ درصد از محصولات خود را به وسیلهٔ سیستم های توصیه گر فروخته است. یا نتفلیکس در همان سال مدعی شد که توانسته ۶۰ درصد از کسب وکار اجاره نامه های خود را با استفاده از سیستمهای توصیه گر معامله کند.
باتوجه به اهمیت این موضوع، تاکنون الگوریتم های فراوانی برای ارتقای سیستم های توصیه گر ارائه شده است. یکی از پرکاربرترین و مهمترین تکنیک هایی که در این زمینه استفاده میشود، پالایش مشارکتی است و همانطور که میدانیم این تکنیک از الگوی رأی کاربران برای توصیه های خود استفاده میکند. در سال ۲۰۰۲ سایت آمازون از مشتریانش دربارهٔ پیشنهادهای نامناسب سیستم توصیه گر خود شکایتهای فراوانی دریافت کرد که بعدها در تحقیقاتی که توسط این سایت انجام شد، نتیجه به فروشندگان بی پروا ختم شد.

پس این ظرفیت وجود دارد که افراد سودجو به این سیستم ها توجه کنند و با ایجاد پروفایلهای جعلی و آرای مغرضانه، درصدد پایین آوردن محصولات رقیبان خود و بالا بردن محصول خود به عنوان محصول برتر سیستم های توصیه گر باشند. به این نوع حمله، حملات شیلینگ میگویند. مهمترین مسئله ای که باعث میشود کاربران به پیشنهاد توصیه گرها توجه داشته باشند و آن را بپذیرند، اعتماد آنها به محصولات و نزدیک بودن توصیه به علایقشان است. از این رو باید به افرادی که با ساختن پروفایل های جعلی و دادن آرای مغرضانه درصدد پایین آوردن محصولات دیگران و بالا بردن محصولات خود هستند، توجه ویژه ای کرد.

تاکنون الگوریتم های مختلفی برای شناسایی پروفایل های مخرب و پیدا کردن آیتم هایی که مورد حمله واقع شده، ارائه شده است که هر یک مشکلاتی دارند. ما در این پروژه به بررسی و پیاده سازی چند روش می پردازیم.

دیتاست

دیتاست ما داده های مووی لنز است:
در پوشهresources که همراه با این سند ضمیمه شده است، یک مجموعه داده از ۳۸۸۳ فیلم، ۶۰۴۰ کاربر، و ۹۹۹۲۳۲ رکورد از نمره دهی کاربران به فیلم ها (از ۱ تا ۵) و همچنین زمان مربوط به لحظه ی نمره دهی (به ثانیه) آورده شده است.

تعاریف

در تصویر زیر **انواع حمله** را مشاهده می کنید:

Filler Items	Target
random ratings	r_{max}

random attack

Targeted Segment	Filler Items	Target
r_{max}	r_{min}	r_{max}

segmented attack

Filler Items	Target
item average ratings	r_{max}

average attack

Frequently Rated	Filler Items	Target
r_{max}	*random ratings*	r_{max}

bandwagon attack

برای در امان ماندن سیستم های پیشنهاد دهنده از حملات شیلینگ چند روش کلی وجود دارد. روش اول طراحی یک سیستم پیشنهاد دهنده قوی برای جلوگیری از اثرات شیلینگ روی سیستم است. روش دوم تشخیص کاربرهای مخرب است. در این روش سیستم با یک سری فرضیات سعی دارد کاربران مشکوک را از سیستم پیشنهاد دهنده دور نگه دارد.
کاری که ما در این پروژه قصد انجام آن را داریم پیاده سازی الگوریتمی است که بتوان با آن حملات شیلینگ را روی فیلم ها تشخیص داد.
البته اینکه این الگوریتم به صورت real time و دارای یک api برای برگرداندن اکانت های مخرب باشد هم یکی از چالش های مسئله است.

پیاده سازی

در ابتدا ما روش مطرح شده در مقاله (چنگ و چاکرابارتی ۲۰۰۶) را بررسی و پیاده سازی کردیم.
در این مقاله دو مقیاس برای تشخیص یک حمله شیلینگ مطرح شده.
اولی sample entropy است که میزان پراکندگی یا غلظت توزیع را نشان می دهد و sample average که درجه محبوبیت یک آیتم را بر اساس میانگین رای های داده شده نشان می دهد.
فرمول هرکدام از آنها در زیر آمده است (M(X) همان میانگین نمونه و H(X) همان آنتروپی نمونه است)

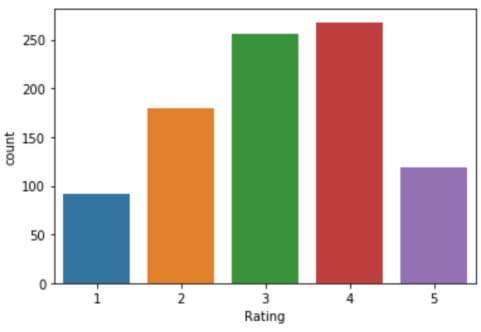
$$M(X) = \frac{\sum_{i=1}^{r_{max}} n_i * i}{S} \qquad H(X) = -\sum_{i=1}^{r_{max}} (\frac{n_i}{S}) \log_2(\frac{n_i}{S}),$$

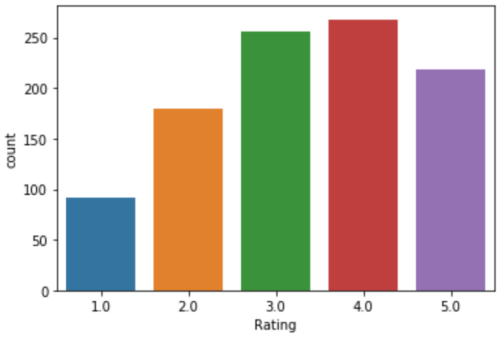
به صورت کلی مقاله سعی در توضیح روشی دارد که با بررسی سری زمانی رای های داده شده به یک آیتم و میزان پراکندگی و میانگین این سری برخی بازه های مشکوک را شناسایی کرده و حملات شیلینگ را تشخیص دهد.
بعد از بارگزاری داده ها توسط کتابخانه pandas یک فیلم که تعداد رای بالایی هم دارد را به عنوان آیتمی که به آن حمله شده در نظر میگیریم. داده های زیر مربوط به این فیلم است:

	UserID	MovieID	Rating	Timestamp
1646	15	70	2	978212287
2519	20	70	4	978143369
2767	22	70	4	978136888
3061	23	70	4	978460739
7026	52	70	4	977947827
...	...	...	...	...
988213	5972	70	3	963615119
993821	6007	70	3	956791980
994257	6010	70	4	956860159
996390	6023	70	2	956751792
996701	6025	70	2	956730684

914 rows × 4 columns

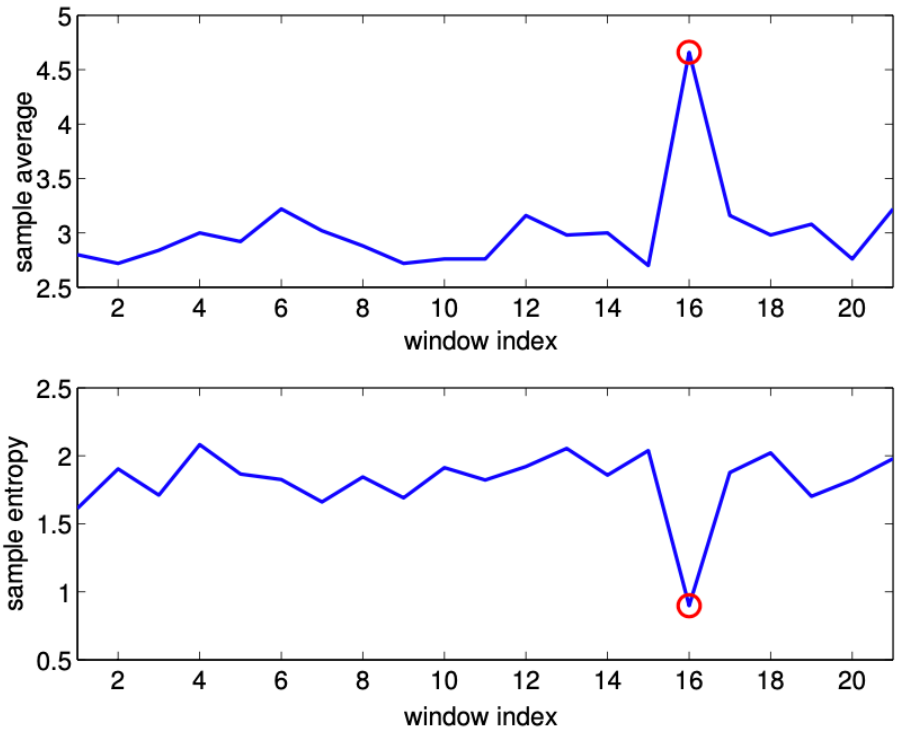
برای تست الگوریتم نیاز داریم یک حمله را شبیه سازی کنیم. برای اینکار ما از حمله random attack استفاده میکنیم. اندازه حمله را ۱۰۰ در نظر گرفتیم.
نمودار نوع و تعداد رای ها قبل و بعد حمله به شکل زیر است:

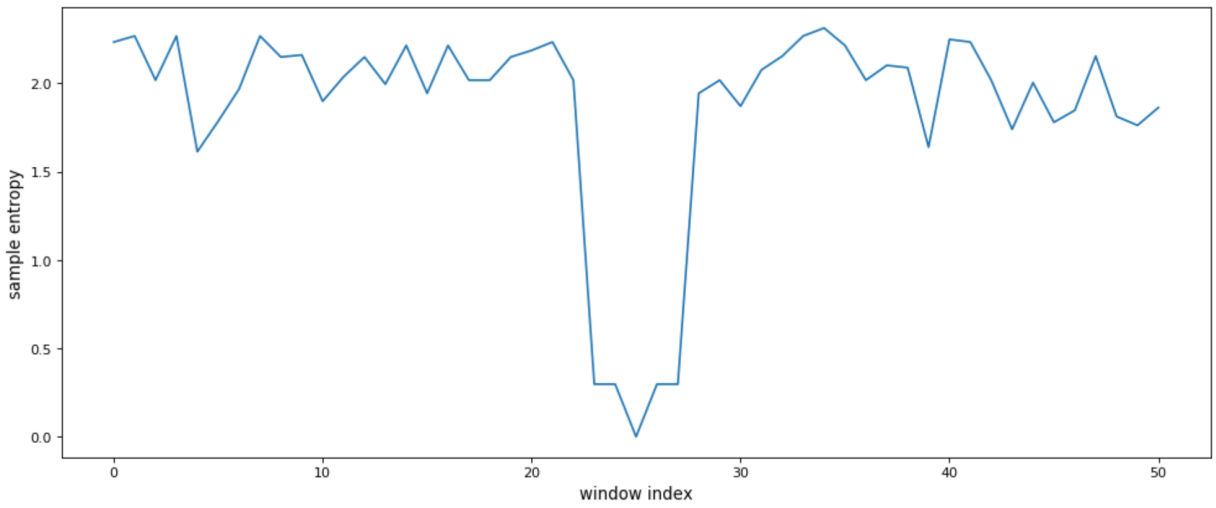




تغییر در تعداد رای ۵ به صورت ملموسی قابل مشاهده است.

بعد از آن شروع به پیاده سازی تابع محاسبه sample entropy و sample average میکنیم.
نحوه کار این توابع بدین شکل است که یک سری زمانی از رای های داده شده دریافت میکنند. بعد از تکه تکه کردن آنها به بازه های کوچکتر(پنجره ها) برای هر بازه از فرمول های ذکر شده برای محاسبه مقدار نهایی استفاده میکنند.
بعد از پیاده سازی این توابع ما نمودار سری زمانی نهایی را میکشیم:





به وضوح مشاهده می شود که قسمت خم شده نمودار به سمت پایین همان زمان حمله شیلینگ است.
دلیل آن هم این است که در زمان حمله به دلیل تراکم رای های مهاجمان درجه آنتروپی کاهش می یابد.

در مقاله توضیح میدهد که توزیع رای به یک فیلم تقریبا توزیع نرمال (گوسیین) در حالت عادی است و اگر z-score سری بدست آمده از مرحله قبل را محاسبه کنیم متوجه می شویم در پنجره هایی که مقدارz-score کاهشی است همان بازه حمله است.

PROPOSITION 1. *If ratings to an item are i.i.d. with mean* μ *and variance* σ^2,

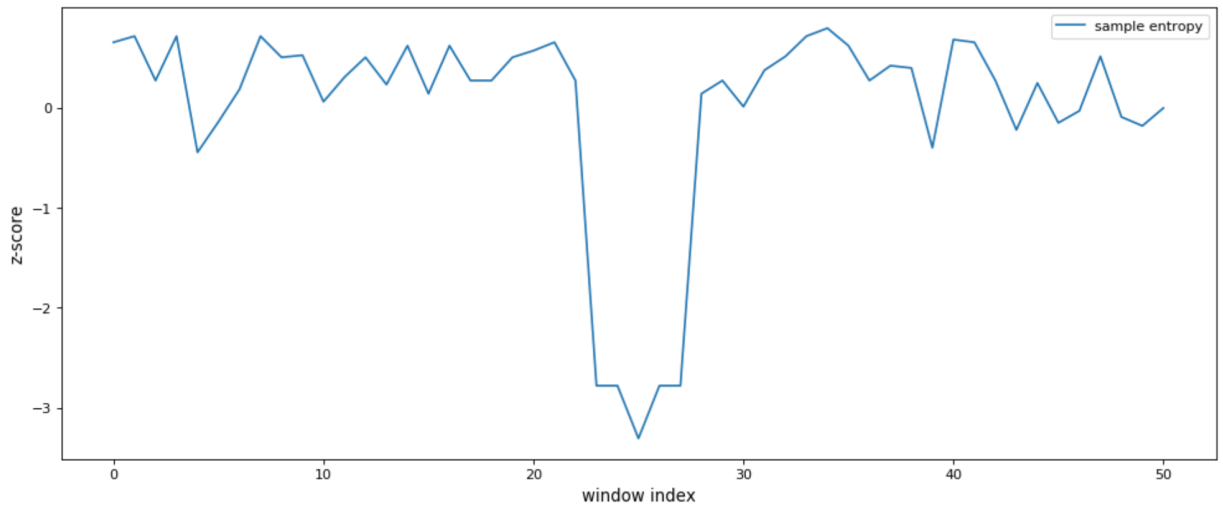
$$\frac{M(w_j) - \mu}{\sigma/\sqrt{k}} \to N(0,1).$$

In other words, the sample average for the item can be approximated using a normal distribution with mean μ *and standard deviation* $\sigma/\sqrt{k}$.

The above proposition follows from the Central Limit Theorem [15].

PROPOSITION 2. *If ratings to an item are i.i.d. from a distribution* P,

$$\frac{H(w_j) - H}{\sqrt{Var(-\log_2 p(x))}/\sqrt{k}} \to N(0,1),$$



مقدار بهینه برای این پنجره ها بر محاسبات مقاله طبق فرمول زیر است.

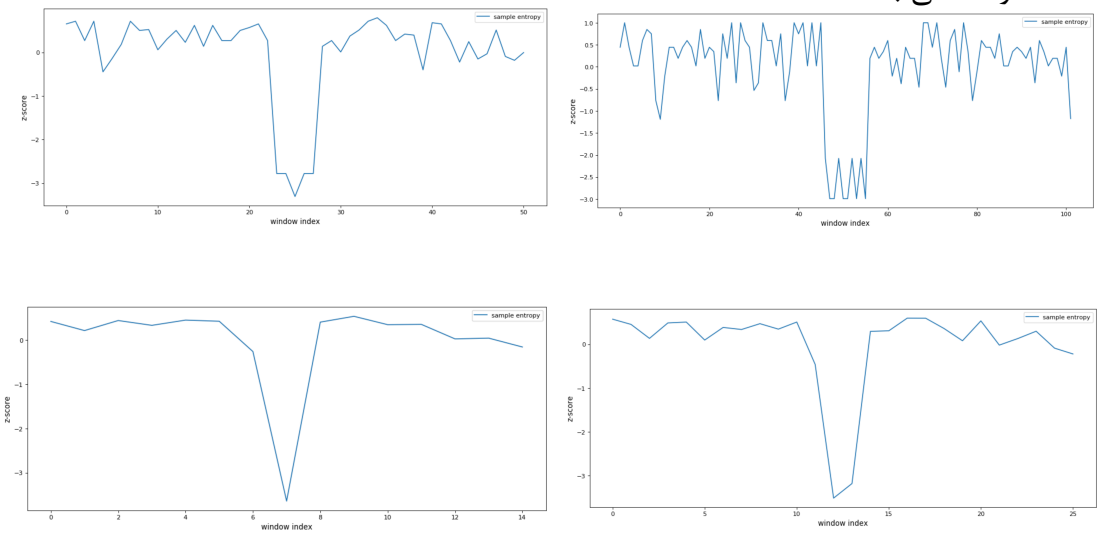
در این فرمول n اندازه حمله است:

$$k = \frac{2+\sqrt{7}}{6}n.$$

در این مرحله نکته مهم این است که طول این پنجره ها برچه مبنایی محاسبه می شوند؟
مقاله در جواب روشی را شرح داده که بدین صورت عمل میکند:

۱. ابتدا یک مقدار به عنوان پیش فرض(مثلا ما ۲۰ در نظر گرفتیم) برای اندازه پنجره درنظر میگیرم.
۲. بعد تعداد نقطه هایی که از حد معمول کمتر یا بیشتر هستند رو حساب میکنیم و با ضرب اونها در اندازه پنجره تعداد حدودی اندازه حمله رو بدست میاریم.
۳. با استفاده از فرمول مرحله قبل اندازه پنجره جدید رو حساب میکنیم.
۴. انقدر این کار رو ادامه میدیم که به جای چند نقطه مشکوک با مقدار خیلی کم در نمودار به یک نقطه برسیم.

نمونه الگوریتم بالا را ما برای خود پیاده سازی کردیم و نمودار های زیر هرکدام به ترتیب پنجره با اندازه ۱۰، ۲۰، ۴۰ و ۷۰ می باشند.



اکنون ما یک الگوریتم داریم که با استفاده از ان میتوانیم بازه های حمله را تشخیص دهیم. برای اینکار ابتدا یک فیلم که مشکوک به حمله است در نظر میگیرم بعد سری z-score ان بر حسب آنتروپی نمونه بدست میاوریم.
بعد از آن با استفاده از الگوریتم قبلی اندازه بهینه برای پنجره را بدست می آوریم و دوباره مقادیر z-score sample entropy رو حساب میکنیم.
اگر از یک threshold خاص کمتر بود (مثلا ما ۲ درنظر گرفتیم) یعنی آن نقطه حمله است.
در تمام آزمایش های فوق ، با مشاهده سری زمانی فیلم های مورد نظر، حملات شناسایی می شوند.

نکته: در هرفراخوانی api برای اضافه کردن یک رای جدید، الگوریتم ما یکبار عدم وجود حمله را در فیلم مورد رای چک کرده و در صورت وجود حمله کاربرانی که مشکوک هستند را به عنوان کاربر مخرب به دیتابیس اضافه میکند.

رویکرد ما در تشخیص حمله مبتنی بر این فرض است که مدت زمان یک رویداد حمله نسبتاً کوتاه است، بنابراین توزیع امتیاز برخی از موارد در آن مدت تغییر می کند. ما استدلال می کنیم که این یک جمع بندی معقول به دلیل دو نکته زیر است. اولاً ، مهاجمان شیلینگ (به عنوان مثال ، تولیدکنندگان اقلام) معمولاً امیدوارند که رتبه بندی حملات آنها در اسرع وقت قابل اجرا باشد ، زیرا الگوهای عادی رتبه بندی کاربران نیز ممکن است در طول حملات تغییر کند. دوم ، اگر یک رویداد حمله نتواند تغییرات قابل توجهی را در توزیع رتبه بندی موارد هدف در یک دوره زمانی ایجاد کند ، در نتیجه اثر این رویداد حمله به طور کلی ناچیز خواهد بود.

ما در این پروژه یک روش دیگر را هم امتحان کردیم که در مقاله ساکشی-گوپا-۲۰۱۸ آمده است. در این روش ابتدا باید feature extraction انجام داد و بعضی از مقادیر را برای هر کاربر بدست آورد و به وسیله آن مقادیر و برچسب های زده شده برای کاربر های مهاجم یک classifier ساخت.
این classifier می تواند KNN، Random Forest، SVM و... باشد. بعد از مرحله میتوان از این classifier برای تشخیص حملات استفاده کرد.
نحوه کار این الگوریتم به شکل زیر است:

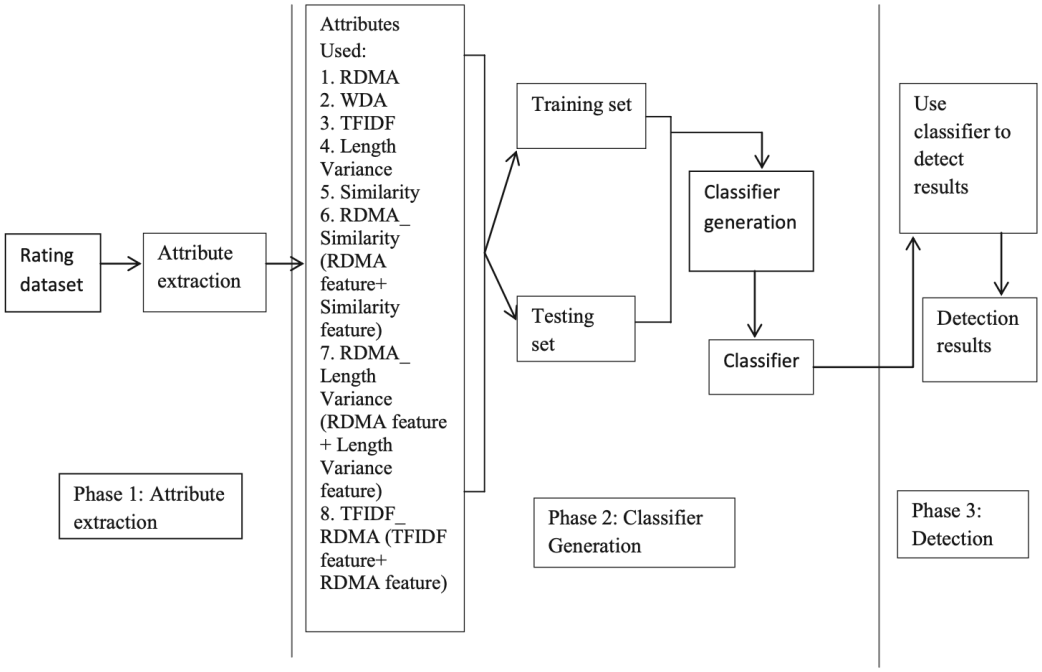


Figure 1. Framework of the proposed approach.

ما از دو شاخص RDMA و DegSim برای ساخت classifier خود بهره بردیم.

The Degree of Similarity with Top Neigh- bours (DegSim)

Rating Deviation from Mean Agreement (RDMA)

فرمول های این دوشاخص به شرح زیر است:

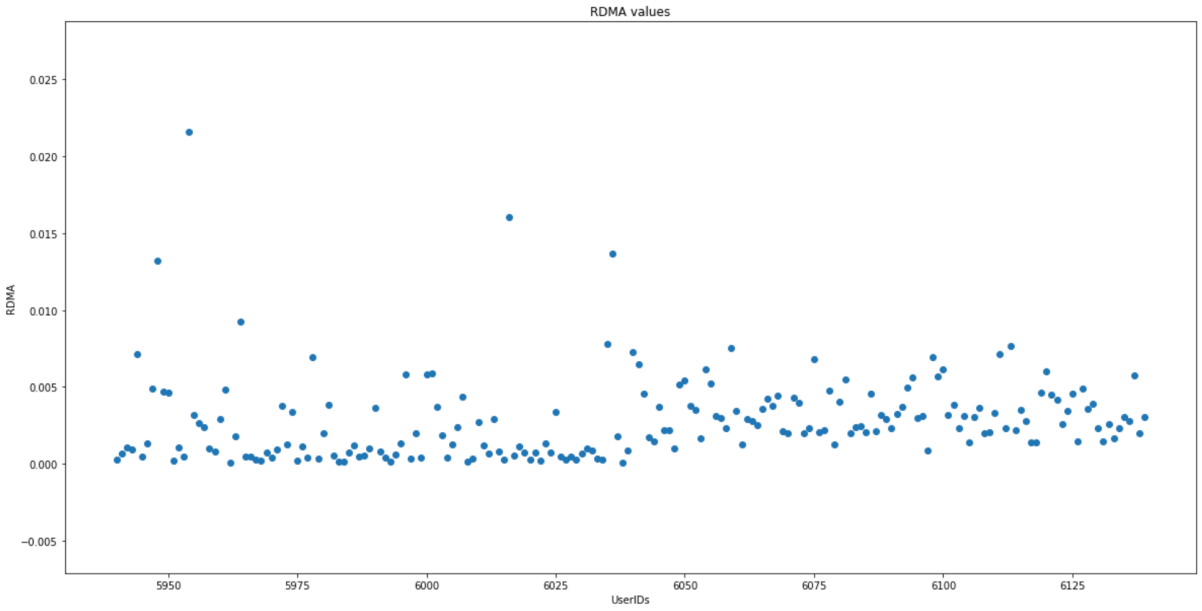
$$\mathrm{DegSim} = \sum_{i=1}^{x} Z_{i,j} \qquad \mathrm{RDMA}_x = \frac{\sum_{x=0}^{T_u} \frac{\left|r_{x,i} - \overline{r}_i\right|}{R_{x,i}}}{N_x}$$

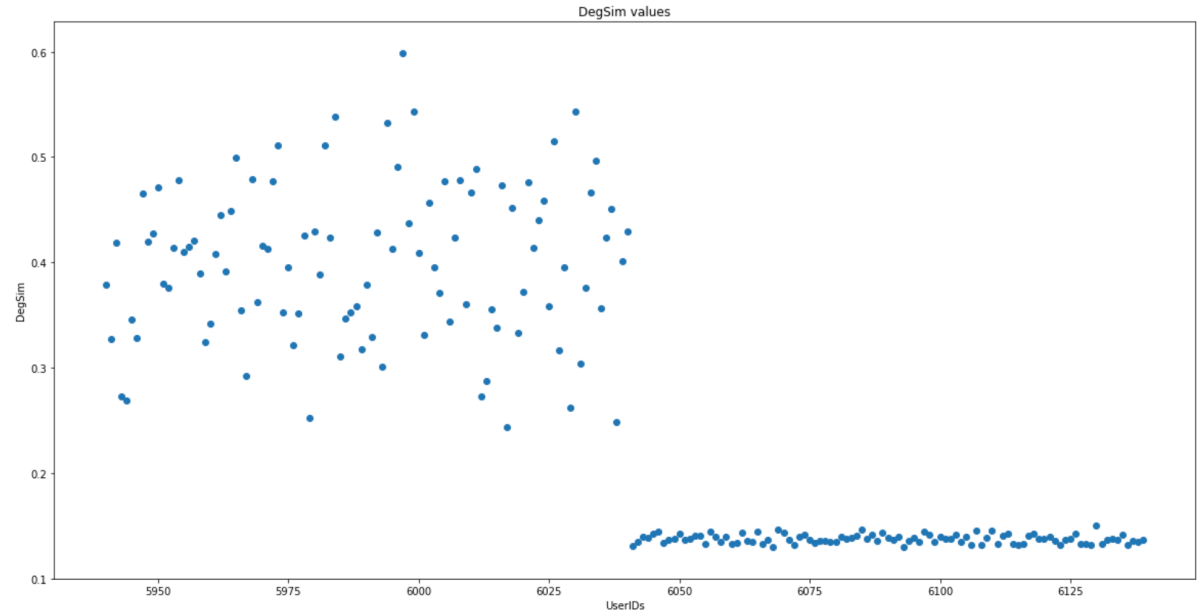
برای شبیه سازی حمله ما از همان تابع random attack در مرحله قبل بهره می بریم.
برای بدست آوردن مقادیر DegSim ما نیاز داریم تا ماتریس آیتم-رای را رسم کنیم و به جای خانه های خالی آن (قسمت هایی که کاربر به یک آیتم رای نداده) مقدار صفر بگذاریم.

بعد از محاسبه ماتریس برای هر کاربر ۲۰ کاربر نزدیک به آن را با توجه به فاصله پیرسون بدست می آوریم.

UserID	top1	top2	top3	top4	top5	top6	top7	top8	top9	top10	top11	top12	top13	top14	top15	top16	top17	top18	top19	tc
1.0	1.0	5343.0	5190.0	1481.0	1283.0	5705.0	5762.0	1359.0	1476.0	541.0	1858.0	681.0	1850.0	2177.0	4718.0	634.0	96.0	5887.0	2566.0	27
2.0	2.0	3108.0	95.0	4601.0	2814.0	3995.0	3361.0	2303.0	300.0	810.0	2500.0	5169.0	558.0	2792.0	4786.0	4388.0	2683.0	204.0	3553.0	35
3.0	3.0	3000.0	479.0	5691.0	3500.0	311.0	4320.0	2262.0	2435.0	1904.0	5237.0	2923.0	4889.0	803.0	5264.0	685.0	2787.0	4060.0	3281.0	8
4.0	4.0	4143.0	1575.0	5876.0	562.0	87.0	5842.0	3666.0	5628.0	5237.0	2347.0	3461.0	1349.0	1578.0	4002.0	5994.0	756.0	526.0	3631.0	35
5.0	5.0	1484.0	5452.0	3538.0	281.0	1407.0	5749.0	5826.0	5718.0	3240.0	5496.0	1636.0	2918.0	1255.0	225.0	4607.0	944.0	1104.0	2870.0	50

بعد از بدست آوردن مقادیر RDMA و DegSim نمودار آن را رسم میکنیم:





به وضوح می توان دریافت که نقطه های آبی پایین نمودار DegSim همان نقاط حمله هستند.

و در انتها با تقسیم داده ها به تست و آموزش آن ها را به classifier داده و نتیجه دقت را چاپ میکنیم.

```
# Random Forest

random_forest = RandomForestClassifier(n_estimators=10)
random_forest.fit(train, train_labels.ravel())
Y_pred = random_forest.predict(test)
random_forest.score(train, train_labels)
acc_random_forest = round(random_forest.score(train, train_labels) * 100, 2)
acc_random_forest
```

```
97.01
```

```
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(train, train_labels.ravel())
Y_pred = neigh.predict(test)
neigh.score(train, train_labels)
acc_neigh = round(neigh.score(train, train_labels) * 100, 2)
acc_neigh
```

```
86.57
```

مقالات

Zhang, Sheng, Amit Chakrabarti, James Ford, and Fillia Makedon. "Attack detection in time series for recommender systems." In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 809-814. 2006.

Gunes, Ihsan, Cihan Kaleli, Alper Bilge, and Huseyin Polat. "Shilling attacks against recommender systems: a comprehensive survey." *Artificial Intelligence Review* 42, no. 4 (2014): 767-799.

Nehriri, Javad, and Sasan Hosseinali Zadeh. "A new mechanism to improve the detection rate of shilling attacks in the recommender systems." *Journal of Information Technology Management* 9, no. 4 (2017): 871-892.

Kapoor, Saakshi, Vishal Gupta, and Rohit Kumar. "An Obfuscated Attack Detection Approach for Collaborative Recommender Systems." *Journal of computing and information technology* 26, no. 1 (2018): 45-56.