بازيابي بيشرفته اطلاعات

فاز دوم پروژه

اعضای گروه:

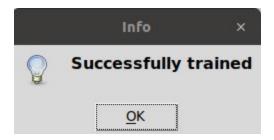
حامد علی محمدزاده - ۹۶۱۰۲۰۲۹ حمیدرضا هدایتی - ۹۶۱۰۹۳۹ آرمین سعادت بروجنی - ۹۶۱۰۵۸۲۹

بخش صفر: معرفي كنسول

در فاز اول برای ایجاد رابط کاربری مناسب و امکان استفاده و تست کردن تمام امکانات سامانه به شکل راحت از کتابخانه tkinter استفاده شد. در این فاز نیز فیچرهای train و test مدلها و همچنین فیلتر برای سرچ به این رابط اضافه شد که در شکل زیر مشاهده می شود.

	MIR Project	_ 0 🔞
Prepare CSV	documents	Prepare XML documents
Enter document desc, Hello! etc.	Enter document title, Intro etc.	Add single document
Enter document ID, 32198 etc.		Delete single document
Enter term, Hello etc.		Show posting-list of a term
Enter term, Hello etc.	Enter document ID, 32198 etc.	Show positions of term in document
Enter bigram terms, ba etc.		Show terms fit in this bigram
Without Compression —	Save index	Load index
Enter your query, Shakespeare etc.	Enter your window size, 5 etc.	No-filter —
Correct my query	LNC-LTC search	Proximity search
Train Models	Test Models	· ·

برای استفاده از امکانات دسته بندی در ابتدا باید دکمه Train Models انتخاب شده و با انتخاب مجموعه داده مورد نظر مدلها train شوند. با اتمام این فرآیند، پنجره زیر ظاهر شده و تمام موفقیت آمیز آموزش را خبر می دهد.



پس از آن برای ارزیابی مدلها می توان دکمه Test Models را زد و با انتخاب مجموعه داده مورد نظر فرایند test انجام شود. پس از پایان رسیدن این فرآیند، نتایج ارزیابی دسته بندها بر اساس معیارهای گفته شده نمایش داده می شود.

توجه شود که فرآیندهای ذکر شده روی هر ۴ مدل classifier اعمال می شود.

دکمه دیگری که اضافه شده در شکل بالا No-filter است که میتوان مقدار آن را به مقادیر Popular و Not-Popular تغییر داد. در هر حالت عملیات سرچ به صورت گفته شده خواهد بود.

- اگر No-filter باشد، عملیات سرچ روی تمام مجموعه دادهها انجام می شود بدون توجه به دسته ی تعیین شده برای آنها. یعنی دقیقا مطابق فاز ۱ است پس به نوعی backward compatible است.
 - اگر گزینه Popular انتخاب شود، عملیات سرچ روی دادههای پربازدید که همان دادههای عضو دسته ۱ هستند صورت می گیرد.
 - اگر گزینه Not-Popular انتخاب شود، عملیات سرچ روی دادههای کمبازدید که همان دادههای عضو دسته ۱- هستند صورت می گیرد.

نکته مهم این است که دادههای فاز ۱ در هنگامی که در سامانه لود می شوند (با انتخاب گزینه Prepare CSV document) دسته بندی می شوند. بنابراین مهم است که قبل از این کار مدل ها train شده باشند.

همچنین نحوه انتخاب مجموعه داده برای Train و Test نیز توسط کاربر از طریق رابط برنامه انتخاب می شود و کاملا داینامیک است. به شکل زیر:

9.	Open	8
<u>D</u> irectory:	/home/rmool/MIR/data/phase2	
test.csv		I»I
File <u>n</u> am	e:	<u>O</u> pen
Files of typ	е:	Cancel

بخش یک: پیادهسازی دستهبندها

پیشپردازش)

هر سطر از جدول یک داکیومنت در نظر گرفته شده است که از description و description است. در گام اول متن هر داکیومنت با استفاده از توابع فاز ۱ و مشابه با همان فرآیند تمیز شد. به عبارتی stop word ها حذف شد، کلمات به حالت ساده خود رفته و سایر مواردی که در مرحله پیش پردازش متنی صورت می گیرد انجام گرفت.

همچنین ستون views به عنوان target value برای هر داکیومنت ذخیره شد.

فضای tf-idf)

پس از آن هر داکیومنت به فضای برداری k برده شد. به این معنا که هر داکیومنت به یک بردار k بعدی تبدیل شد که k تعداد ترمهای موجود در کل مجموعه داده است. که k tf-idf برابر (k tf برابر (k tf-idf برابر (k tf-idf برابر (k tf-idf برابر (k المت.)

البته برای بهینه کد زدن و اشغال کمتر حافظه فقط ترمهایی که در هر داکیومنت وجود داشتند ذخیره شدند. در نهایت هر داکیومنت به شکل یک dict در پایتون ذخیره شد.

ییادهسازی دستهبندها)

هر دستهبند در قالب یک class پایتون نوشته شده است. هر کدام از آنها تعدادی تابع مشابه دارند که مهمترین آنها عبارتند از:

:train(document_set) •

یک مجموعه داده گرفته و با توجه به الگوریتم دستهبند برای یادگیری، پارامترها (و در صورت لزوم hyper-parameter ها) مدل را تعیین می کند.

:predict(document)

یک داکیومنت گرفته و با توجه به مدل دستهبند به دست آمده، مشخص می کند که این داکیومنت به کدام دسته قرار دارد. یعنی تابع مقدار ۱ یا ۱- خروجی می دهد.

:test(document_set) •

یک مجموعه داده به عنوان test set دریافت کرده و برای هر داکیومنت روی این مجموعه، تابع predict را صدا می زند تا دسته هر داکیومنت مشخص شود. سپس دسته پیش بینی شده هر داکیومنت با دسته واقعی آن مقایسه شده و conjugation table داده های تست به دست می آید.

:get_accuracy() ●

با توجه به conjugation table مقدار accuracy را خروجي مي دهد.

:get_precision_c1() •

با توجه به conjugation table مقدار precision را برای دسته پربازدید (که با ۱ مشخص می شود) خروجی می دهد.

:get_precision_c2() ●

با توجه به conjugation table مقدار precision را برای دسته کم بازدید (که با ۱- مشخص می شود) خروجی می دهد.

:get_racall_c1() •

با توجه به conjugation table مقدار recall را برای دسته پربازدید (که با ۱ مشخص می شود) خروجی می دهد.

:get_racall_c2() ●

با توجه به conjugation table مقدار recall را برای دسته کمبازدید (که با ۱- مشخص می شود) خروجی می دهد.

:get_F1_c1() •

با توجه به conjugation table مقدار F1_score را برای دسته پربازدید (که با ۱ مشخص می شود) خروجی می دهد.

:get_F2_c2() ●

با توجه به conjugation table مقدار Fl_score را برای دسته کمبازدید (که با ۱- مشخص می شود) خروجی می دهد. در ادامه به جزئیات ییاده سازی هر دسته بند پرداخته می شود.

:Naive Bayes .1

روش بیز ساده لوحانه یک مدل احتمالاتی است که از قانون بیز استفاده می کند و با به دست آوردن پارامترهای مناسب از روی دادههای یادگیری، احتمال قرارگیری یک داکیومنت در یک دسته خاص را به دست می آورد.

در نهایت به ازای یک داکیومنت هر دسته که احتمال بیشتری داشت، به عنوان دستهی پیشبینی شده انتخاب میشود.

علت وجود عبارت سادهلوحانه در نام گذاری آن به دلیل فرضهای ساده کنندهای است که این مدل در نظر گرفت است. به این

صورت که احتمال حضور داکیومنت d در یک دسته را برابر با ضرب احتمال حضور کلمات داکیومنت d در آن دسته قرار داده است. به عبارتی بین حضور کلمات مختلف در یک داکیومنت یا دسته خاص استقلال قائل شده است. رابطه بیز به شکل زیر است:

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)}$$
$$P(c|d) \propto P(c)P(d|c)$$

كه با فرض استقلال نتيجه مي دهد:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

که t_k موجود در داکیومنت t_k است. پس تعداد تکرار هم مهم است.

عبارت بالا به دلیل ضرب تعداد زیادی عدد کوچکتر از ۱ به صفر بسیار نزدیک می شود که در انجام محاسبات با کامپیوتر ممکن است به underflow منجر شود. برای همین لگاریتم عبارت بالا مد نظر قرار می گیرد و در نهایت دسته انتخابی آن دستهای می شود که بیشترین مقدار را به خود اختصاص داده است.

پس با فرض داشتن مقادیر (P(c) و P(c)، داده D اینگونه دستهبندی می شود:

$$c_{\mathsf{map}} = \operatorname*{argmax}_{c \in \mathbb{C}} \ \left[\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k | c) \right]$$

پارامترهای این مدل از روی دادههای یادگیری مطابق با روابط زیر حساب می شوند:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}} \qquad \hat{P}(c) = \frac{N_c}{N}$$

یک مشکل این مدل این است که اگر در داکیومنت d کلمهای آمده باشد که احتمال حضور آن در دسته c صفر تخمین زده شده باشد، احتمال آن دسته برای داشته باشد. برای جلوگیری از این حالت احتمال آن دسته برای داکیومنت d صفر مطلق می شود هرچند که ممکن است کلمات مرتبط بسیاری داشته باشد. برای جلوگیری از این حالت روش Add_one smoothing استفاده می شود که عملا فرض می کند هر term حداقل یک بار در هر دسته ظاهر شده است. در این صورت پارامترها از رابطه زیر به دست می آیند:

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V} (T_{ct'} + 1)} = \frac{T_{ct} + 1}{(\sum_{t' \in V} T_{ct'}) + B}$$

در كد نيز از همين معادله استفاده شد. با صدا زدن تابع train و پاس دادن مجموعه داده يادگيرى، اين پارامترها به دست آمده و پس از آن با train و با استفاده از prediction rule گفته شده در بالا دسته هر داكيومنت تعيين مى شود. در ضمن از تمام داده predict استفاده شد.

:k-NN .3

شرح الگوريتم:

الگوریتم k-NN یک الگوریتم یادگیری supervised است. برای پیدا کردن دسته یک data point این الگورتیم ابتدا اتما نزدیکترین data point به داده مورد نظر را پیدا کرده و طبق اینکه بیشتر به چه دسته ای تعلق دارند داده مورد نظر را به آن دسته اختصاص میدهد. برای محاسبه فاصله بین دو داده که در اینجا بردارهایی n بعدی هستند روشهای مختلفی می توان استفاده کرد، فاصله اقلیدسی، منهتنی و مینکوفسکی، در اینجا ما از روش

منهتنی استفاده کردهایم.

نحوه پيادهسازى

كليت بيادهسازي اين الگوريتم شامل اين ٣ مرحله است:

- 1. محاسبه فاصله یک داده تست با همهی دادههای آموزشی برای پیدا کردن نزدیکترینها
 - 2. مرتبسازی فاصله ها و پیدا کردن kتا از نزدیکترین داده ها
 - 3. پیدا کردن پرتکرارترین لیبل. این لیبل به عنوان دسته ی داده ی تست قرار می گیرد.

توضیح پیاده سازی توابع موجود در class دسته بند (بخش یک)

:train(document_set) •

با توجه به این که k-NN فرایند یادگیری انجام نمی دهد در این تابع صرفا داده های آموزشی به همراه برچسب هرکدام دریافت می شوند و پس از یک پیش پردازش که لیبل را به عنوان درایهی آخر بردارهای داده ی آموزشی قرار می دهد، آن ها را در مدل ذخیره می کند.

:predict(document)

در این تابع مراحل یک تا سه توضیح داده در بالا پیاده سازی شدهاند و در پایان برای هر داده ی ورودی یک دسته اختصاص می یابد. این تابع ابتدا یک پیش پردازش روی داده ی تست ورودی انحام می دهد. به این شکل که بردارها را طوری بازسازی می کند که شامل همهی ترمهای دیکشنری باشد. در صورتی که یک ترم وجود داشته باشد مقدار آن صف قرار داده می شود و اگر یک ترم در

دادهی تست باشد که در دیکشنری وجود نداشته باشد حذف می شود.

:test(document_set) •

مانند توضیحات داده شده در بخش یک.

پیدا کردن k بهینه

این بخش در تابع train پیاده سازی شده است. ابتدا ۳۰ درصد از دادههای آموزشی به طور رندوم انتخاب شده و ۱۰ درصد از آنها

به عنوان داده validation به طور رندم جدا می شد. پس از ذخیره داده آموزشی در مدل، به ازای مقادیر مختلف k = 1, 5, 9 مدل

تست می شود و آن مقدار k که منجر به بیشترین accuracy شود به عنوان k بهینه انتخاب می شود.

به دلیل اردر زمانی k-NN که به خاطر محاسبه فاصله هر داده تست از مجموعه آموزشی رخ می دهد، از ۳۰ درصد از دادههای

آموزشی استفاده کردیم و از ۱۰ درصد آن به عنوان داده validation استفاده کردیم. انتخاب دادهها به طور رندوم انجام می شود.

نتيجه:

K بهینه: ۵

accuracy: ..or

Accuracy for 1-NN: 0.5175438596491229

Accuracy for 5-NN: 0.5350877192982456

Accuracy for 9-NN: 0.4649122807017544

Selected K for K-NN is 5 with accuracy of 0.5350877192982456

:SVM .4

هدف از این الگوریتم ایجاد یک مرز بین مجموعه داده هاست. این الگوریتم یک روش supervise است. نحوه کار آن به این شکل است که تعدادی داده برچسب زده شده به آن داده می شود و در نهایت یک hyperplane در فضای برداری مسئله ساخته می شود که مرز بین دو دسته ی مختلف از دادگان است. ابر صفحه ای بهتر است که margin به نسب بزرگتری از داده های هر دسته داشته باشد.

برای پیاده سازی این روش از دسته بند SVC موجود در پکیج sklearn.svm استفاده کرده ایم. برای train کردن ابتدا با استفاده از تابع fit_transform داده های آموزشی ورودی را تبدیل به وکتورهایی می کنیم که بتوان برای فیت کردن مدل از آن استفاده کرد. زیرا ممکن است وکتوها ترم هایی داشته باشند که در دیکشنری موجود نباشد و برعکس. سپس مدل را train میکنیم.

پیدا کردن C بهینه

این بخش در تابع train پیاده سازی شده است. از ۱۰ درصد از دادههای آموزشی به عنوان داده validation استفاده شده است. پس از fit کردن مدل با ۹۰ درصد از دادههای آموزشی، به ازای مقادیر مختلف C = 0.5, 1, 1.5, 2 مدل تست می شود و آن مقدار C که منجر به بیشترین accuracy شود به عنوان C بهینه انتخاب می شود.

نتيجه:

C بهینه: ۲

accuracy: .. 5 V

Accuracy for SVM with C = 0.5 is 0.6069868995633187

Accuracy for SVM with C = 1 is 0.6637554585152838

Accuracy for SVM with C = 1.5 is 0.6681222707423581

Accuracy for SVM with C = 2 is 0.6724890829694323

Selected C for SVM is 2 with accuracy of 0.6724890829694323

:Random Forest .5

این الگوریتم به این صورت کار می کند که تعدادی decision tree روی مجموعه داده های یادگیری ساخته و جواب نهایی را میانگین جواب این درختها در نظر می گیرد.

بخش اصلی کد این بخش توسط کتابخانه sklearn پیاده سازی شد. به این صورت که تابع predict کلاس RandomForest از یک sklearn که sklearn در اختیار گذاشته است استفاده می کند و خروجی ۱ یا ۱- می دهد و مابقی کارها مشابه سایر الگوریتم ها است.

این classifier در هنگام یادگیری (در تابع train) روی دادهها fit می شود. که برای اینکار می توان یک عمق به آن به عنوان ورودی داده که عملا عمق می فرد که عملا عمق می فرد و حتی ممکن است به overfitting منجر شود. در ضمن با افزایش عمق، زمان یادگیری نیز افزایش می یابد. با بررسی چندین حالت و دیدن نتایج، مقدار ۳۰ برای عمق انتخاب شد.

برای آموزش این دسته بند از تمام مجموعه داده یادگیری استفاده شد.

بخش دو: بهبود سیستم بازیابی اطلاعات فاز اول پروژه

در این بخش ما تلاش کردیم که کدهایی که برای فاز یک زده بودیم را با دادههای Train شده برچسب بزنیم، در این راستا نخست فرض می کنیم که کاربر در رابط کاربری با کلیک کردن بر روی Train Models و انتخاب داده Train نخست مدل را Train می کند. بعد از آن با اضافه کردن دادههایی که میخواهیم روی آنها Search را انجام دهیم، دادهها با استفاده از SVM Classifier برچسبگذاری می شوند و این برچسبها در دیکشنری Label ذخیره می شوند:

Prepare CSV documents					Prepare XML documents		
Enter document desc, Hello! et	c. E	Enter document title, Intro etc.			Add single document		
Enter document ID, 32198 etc.			Open		Dala	e single document	
Enter term, Hello etc.	Director	<u>Directory</u> : /home/hamidreza/MIR			_	osting-list of a term	
Enter term, Hello etc.	.git	cache	index	he	tignore lper.py ain.py	ons of term in document	
Enter bigram terms, ba etc.	class com	classifiers preprocess compressor search		README.md report_phase1			
Without Compression	data		iii venv	□ re	quirements.t	Load index	
Enter your query, Shakespeare	ter your query, Shakespeare ϵ Files of type:				Open Cancel	No-filter -	
Correct my query		L	NC-LTC search		Р	roximity search	
Train Models			Test Models				

بعد از انتخاب فایل ted_talks.csv، داده ها به وسیله Trainی که انجام شده برچسبگذاری می شوند.

در گذشته می توانستیم با سرچ کردن یک عبارت، داکیومنتهای مرتبط به آن عبارت را پیدا کنیم، هم اکنون گزینه جدیدی به فیلد سرچ کردن اضافه شده که در تصویر هم مشاهده می کنید (که گزینه پیشفرض آن، No-filter است و به Not Popular هم می تواند تبدیل شود)

برچسبگذاری با استفاده از دسته بند SVM انجام شدهاست زیرا SMV هم accuracy بهتری به ارمغان می آورد و هم فرآیند Testing سریع تری دارد که باعث می شود برچسبگذاری سریع تر انجام شود.

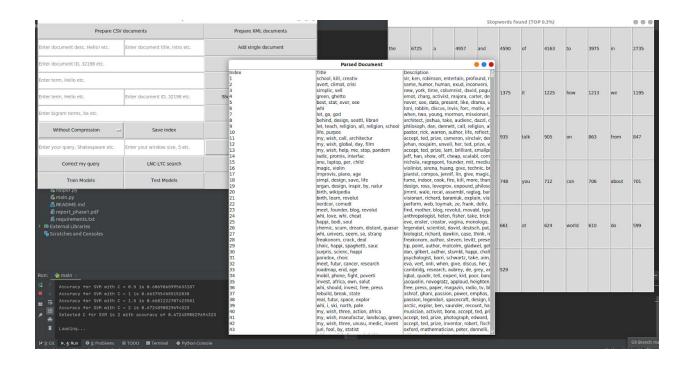
پس از برچسبگذاری دادهها، موقع ذخیرهسازی و بارگذاری، اعمال روی دیکشنری label هم اعمال

می شود در نتیجه می توانیم هر بار مجبور به Train کردن مدل نشویم.

1. Train Models

2. Prepare Documents (برچسب گذاری اینجا انجام می شود)

Prepare CSV documents				Prepare XML documents		
Enter document desc, Hello! et	c.		Add single document			
Enter document ID, 32198 etc.		Open		Dol	e single document	
Enter term, Hello etc.	Directo	ory: /home/hamidreza/MIR/d	osting-list of a term			
Enter term, Hello etc.		ase2 rsian.xml I_talks.csv			ons of term in document	
Enter bigram terms, ba etc.		TOTAL			rms fit in this bigram	
Without Compression	4			b	Load index	
Enter your query, Shakespeare	E	name: ted_talks.csv	No-filter			
Correct my query		LNC-LTC search		ı	Proximity search	
Train Models		Test Models				



3. Phrase Searching

بدون فيلتر:

Prepare CS\	/ documents	Prepare XML documents	Document ID: 70 with similarity 1.2241087376277724 Document ID: 1968 with similarity 1.1671418864967293
Enter document desc, Hello! etc.	Enter document title, Intro etc.	Add single document	Document ID: 1333 with similarity 1.0345607078564654 Document ID: 1995 with similarity 0.7477742912126837 Document ID: 468 with similarity 0.7172399036241228
Enter document ID, 32198 etc.		Delete single document	Document ID: 719 with similarity 0.6441004475803275 Document ID: 1172 with similarity 0.6237040332516852 Document ID: 494 with similarity 0.5895675690539892
Enter term, Hello etc.		Show posting-list of a term	Document ID: 889 with similarity 0.5817583822879818 Document ID: 208 with similarity 0.5790946568853144 Document ID: 2441 with similarity 0.5790946568853144
Enter term, Hello etc.	Enter document ID, 32198 etc.	Show positions of term in document	Document ID: 1096 with similarity 0.5693612803648336 Document ID: 1989 with similarity 0.5670279310529712 Document ID: 1136 with similarity 0.5601027563374329
Enter bigram terms, ba etc.		Show terms fit in this bigram	Document ID: 2505 with similarity 0.5533625992784504 Document ID: 1052 with similarity 0.5428646805928399
Without Compression —	Save index	Load index	Document ID: 437 with similarity 0.5348217962517015 Document ID: 490 with similarity 0.5271261173939978 Document ID: 584 with similarity 0.5197533610430709
cognit neuroscientist	Enter your window size, 5 etc.	No-filter	Document ID: 1382 with similarity 0.5176233142886607
Correct my query	LNC-LTC search	Proximity search	
Train Models	Test Models		

فيلتر Popular:

Prepare CSV	documents	Prepare XML documents	Document ID: 70 with similarity 1.2241087376277724 Document ID: 1968 with similarity 1.1671418864967293
Enter document desc, Hello! etc.	Enter document title, Intro etc.	Add single document	Document ID: 1333 with similarity 1.0345607078564654 Document ID: 1995 with similarity 0.7477742912126837 Document ID: 468 with similarity 0.7172399036241228
Enter document ID, 32198 etc.		Delete single document	Document ID: 889 with similarity 0.5817583822879818 Document ID: 208 with similarity 0.5790946568853144 Document ID: 2441 with similarity 0.5790946568853144
Enter term, Hello etc.	Enter term, Hello etc.		Document ID: 1096 with similarity 0.5693612803648336 Document ID: 1989 with similarity 0.5670279310529712
Enter term, Hello etc.	inter term, Hello etc. Enter document ID, 32198 etc.		Document ID: 1136 with similarity 0.5601027563374329 Document ID: 2505 with similarity 0.5533625992784504 Document ID: 1052 with similarity 0.5428646805928399
Enter bigram terms, ba etc.		Show terms fit in this bigram	Document ID: 437 with similarity 0.5348217962517015 Document ID: 490 with similarity 0.5271261173939978 Document ID: 584 with similarity 0.5197533610430709
Without Compression —	Save index	Load index	Document ID: 1382 with similarity 0.5176233142886607 Document ID: 2062 with similarity 0.5126815572242197 Document ID: 1235 with similarity 0.5058907731953838
cognit neuroscientist	Enter your window size, 5 etc.	Popular	Document ID: 1967 with similarity 0.48703102589005964
Correct my query	LNC-LTC search	Proximity search	
Train Models	Test Models		

اگر دقت کنیم می بینیم که docuement id 70 به عنوان یک نتیجه popular نشان داده شده که در داده ها نزدیک به ۲ میلیون view دارد که نشان می دهد برچسب گذاری به صورت منطقی ای انجام داده شده است.

فيلتر Not-Popular:

Prepare CSV documents		Prepare XML documents	Document ID: 719 with similarity 0.6441004475803275 Document ID: 1172 with similarity 0.6237040332516852
Enter document desc, Hello! etc.	Enter document title, Intro etc.	Add single document	Document ID: 494 with similarity 0.5895675690539892 Document ID: 1981 with similarity 0.4708910662476737 Document ID: 1213 with similarity 0.44143064563483086
Enter document ID, 32198 etc.		Delete single document	Document ID: 1700 with similarity 0.41973290567504723
Enter term, Hello etc.		Show posting-list of a term	
Enter term, Hello etc.	Enter document ID, 32198 etc.	Show positions of term in document	
Enter bigram terms, ba etc.	inter bigram terms, ba etc.		
Without Compression —	Save index	Load index	
cognit neuroscientist	Enter your window size, 5 etc.	Not-Popular —	
Correct my query	LNC-LTC search	Proximity search	
Train Models	Test Models		

با تست كردن اين موارد با Proximity search هم به نتايج مشابه خواهيم رسيد.

بخش سوم: ارزیابی نهایی

شرح روشهای ارزیابی)

با اجرای تابع test در هر مدل دستهبند، conjugation table برای دستههای ۱ و ۱- محاسبه و ذخیره می شود. Conjugation table به شکل زیر است:

	Class 1 Predicted	Class 2 Predicted	
Class 1 Actual	TP	FN	
Class 2 Actual	FP	TN	

دسته ۱ همان دسته یربازدید است که با c1 نشان داده می شود.

دسته ۱- همان دسته کمبازدید است که با c2 نشان داده می شود.

با توجه به جدول بالا نحوه محاسبه هر یک از معیارهای ارزیابی توضیح داده می شود.

- Accuracy = $(TP + TN) \div (TP + FP + TN + FN)$
- Precision_c1 = $(TP) \div (TP + FP)$
- Precision_c2 = $(TN) \div (FN + TN)$
- Recall_c1 = $(TP) \div (TP + FN)$
- Recall_c2 = $(TN) \div (TN + FP)$

رابطه F1 هم به شكل زير است:

$$F_1 = \frac{1}{\frac{1}{2}\frac{1}{P} + \frac{1}{2}\frac{1}{R}} = \frac{2PR}{P + R}$$

که R همان Recall و P همان Precision است.

برای به دست آوردن F1 هر دسته از R و P همان دسته استفاده می شود.

گزارش نتایج)

نتایج به دست آمده از تست کردن هر ۴ دسته بند روی داده های آموزش و آزمون به تعبیر هر ۴ روش ارزیابی در شکل زیر آمده است که خروجی رابط کاربری است.

سمت راست مربوط به دادههای آزمون و سمت چپ مربوط به دادههای آموزش است.

Evaluation - 🗆 🛚

NaiveBayes: Accuracy: 0.9895 F1_C1: 0.9896 Precision_C1: 0.9854 Recall_C1: 0.9851 F1_C2: 0.9894 Precision_C2: 0.9938 Recall_C2: 0.9851

KNN:

Accuracy: 0.6532 F1_C1: 0.593 Precision_C1: 0.7232 Recall_C1: 0.8054 F1_C2: 0.6978 Precision_C2: 0.6155 Recall_C2: 0.8054

SVM:

Accuracy: 0.9673 F1_C1: 0.9676 Precision_C1: 0.9639 Recall_C1: 0.9632 F1_C2: 0.967 Precision_C2: 0.9708 Recall_C2: 0.9632

RandomForest: Accuracy: 0.9656 F1_C1: 0.9647 Precision_C1: 0.9963 Recall_C1: 0.9965 F1_C2: 0.9664 Precision_C2: 0.9381 Recall_C2: 0.9965

Evaluation -

NaiveBayes: Accuracy: 0.6196 F1_C1 : 0.6255 Precision_C1 : 0.587 Recall_C1 : 0.5746 F1_C2 : 0.6135

Precision_C2 : 0.6581 Recall_C2 : 0.5746

KNN:

Accuracy: 0.5373 F1_C1: 0.4434 Precision_C1: 0.5165 Recall_C1: 0.6716 F1_C2: 0.604 Precision_C2: 0.5488 Recall_C2: 0.6716

SVM:

Accuracy: 0.6824 F1_C1: 0.6368 Precision_C1: 0.6961 Recall_C1: 0.7687 F1_C2: 0.7178 Precision_C2: 0.6732 Recall_C2: 0.7687

RandomForest: Accuracy: 0.6392 F1_C1: 0.6378 Precision_C1: 0.609 Recall_C1: 0.6119 F1_C2: 0.6406 Precision_C2: 0.6721 Recall_C2: 0.6119

اطلاعات بالا در جدول زير قابل مشاهده است.

جدول زیر مربوط به دادههای آموزش است:

	Accuracy	F_c1	P_c1	R_c1	F_c2	P_c2	R_c2
Naive Bayes	0.9895	0.9896	0.9854	0.9851	0.9894	0.9938	0.9851
5-NN	0.6532	0.593	0.7232	0.8054	0.6978	0.6155	0.8054
SVM(C=1.5)	0.9673	0.9676	0.9639	0.9632	0.967	0.9708	0.9632
RandomForest	0.9656	0.9647	0.9963	0.9965	0.9664	0.9381	0.9965

جدول زیر مربوط به دادههای آزمون است:

	Accuracy	F_c1	P_c1	R_c1	F_c2	P_c2	R_c2
Naive Bayes	0.6169	0.6255	0.587	0.5746	0.6135	0.6581	0.5746
5-NN	0.5373	0.4434	0.5165	0.6716	0.604	0.5488	0.6716
SVM(C=1.5)	0.6824	0.6368	0.6961	0.7687	0.7178	0.6732	0.7687
RandomForest	0.6392	0.6378	0.609	0.6119	0.6406	0.6721	0.6119

همانطور که ملاحظه می شود به طور کلی عملکرد دسته بندها روی مجموعه آموزش با اختلاف بهتر است که منطقی هم هست چرا که روی همان مجموعه هم fit شده اند.

اما ملاحظه می شود که در روش k-NN عملکرد روی مجموعه داده آموزش نیز خیلی بالا نیست.

دلیل این امر این است که برای آموزش k-NN از حدود ۳۰ درصد دادههای آموزش استفاده شد اما نتیجه بالا ارزیابی k-NN روی تمام مجموعه

داده را نشان میدهد. یعنی k-NN هم روی تمام داده آموزش تست شد در صورتی که فقط روی ۳۰ درصد آن train شد. در نهایت ملاحظه می شود که accuracy دسته بند SVM روی داده های آزمون از همه بیشتر است. بنابراین از این دسته بند برای دسته بندی داده های فاز ۱ استفاده شد.

پایان!