Analitzador de sentiment

“*Si esperes resultats diferents, no facis sempre el mateix*”

- Albert Einstein

Mariano Trebino

Índex de continguts

1 Introducció

Fa ja un parell d'anys vaig treballar per una empresa de publicitat i vaig ser el encarregat de portar un projecte que consistia en analitzar el sentiment de les notícies de premsa, comentaris de Facebook, Twitter, blogs i altres xarxes socials i, per fer-ho, vaig implementar un algorisme en Java.

Ara, en comptes de crear l'algorisme, he fet servir el programa *RapidMiner* amb una extensió específica per al processament de text (*Text Processing*) que facilita molt la feina ja que que inclou moltes eines per *tokenization*, *stemming* i lectura de fitxes de text entre d'altres.

2 Estructura de fitxers

En el .zip lliurat es troba aquest informe i dues carpetes:

- Carpeta **data**. Conté tots els conjunts de dades que s'han utilitzat en aquesta pràctica.
    - La carpeta **pang_lee** conté les dades d'entrenament i testeig.
    - La carpeta **filmaffinity** conté les crítiques obtingudes manualment des de www.filmaffinity.com.
- Carpeta **process**. Aquí és on es troben tots els processos i fitxers que utilitza *RapidMiner* i conté els següents fitxers:
    - Procés **generator**. Procés que a partir del conjunt de dades de la carpeta data entrena, testeja i finalment crea un model.
    - Procés **optimizer**. Procés que aplica el mateix model amb diferents paràmetres per avaluar quina configuració del model és la millor.
    - Procés **scraper**. procés de *scraping* que va a buscar noves dades a Internet, les filtra i crea dades per realitzar prediccions.
    - Procés **predictor_rottentomatoes**. Procés d'aplicació del model a les dades generades amb el procés de *scraping* a www.rottentomatoes.com.
    - Procés **predictor_filmaffinity**. Procés d'aplicació del model a les dades extretes *de* www.filmaffinity.com de manera manual.
- Donat que *RapidMiner* funciona amb els seus repositoris, hi ha alguns fitxers que no els he pogut exportar i són els que es generen amb l'execució dels processos. Aquestes fitxers es crearan dins del repositori en una carpeta anomenada **trebino** un cop s'executin els processos.
    - Fitxer **model**. Model generat amb el procés **generator**.
    - Fitxer **word_vector**. Vector de paraules generat a partir de les dades d'entrenament amb el procés **generator**.
    - Fitxer **data_rottentomatoes**. Dades estretes d'Internet amb el procés **scraper**.

3 Captació de dades

3.1 Entrenament i testeig

Pel que fa a les dades d'entrenament i de testeig jo no he realitzat la captació de dades. He fet servir una versió millorada d'un conjunt de dades molt reconegut i utilitzat en els analitzadors de sentiment que va ser introduït per Pang, Lee i Vaithyanatha al seu llibre *Proceedings of EMNLP.* Aquest conjunt està format per 2000 crítiques de pel·lícules, 1000 positives i 1000 negatives.

3.2 Prediccions

Per tal de fer les prediccions de noves observacions he volgut fer-ho utilitzant dos conjunts de dades molt diferents per poder analitzar el comportament del meu model en diferents condicions.

3.2.1 Film Affinity

Per una banda, he descarregat manualment deu crítiques (cinc positives i cinc negatives) des de *Film Affinity* de les últimes pel·lícules. La particularitat d'aquestes crítiques és que són crítiques formals, extenses i molt detallades.

3.2.2 RottenTomates

Per l'altra banda, he fet un procés de *scraping* amb l'ajuda d'una extensió de *RapidMiner* per descarregar crítiques de pel·lícules de www.rottentomatoes.com (veure Illustration 3.1: Procés de captació de dades de www.rottentomatoes.com). En concret, faig una cerca en tres pàgines de crítiques i recullo 57 noves observacions de la pel·lícula It follows.
Aquestes crítiques pel contrari, són crítiques de dues o tres línies fet que representa un problema per els analitzadors de sentiment com ja comentaré més endavant.

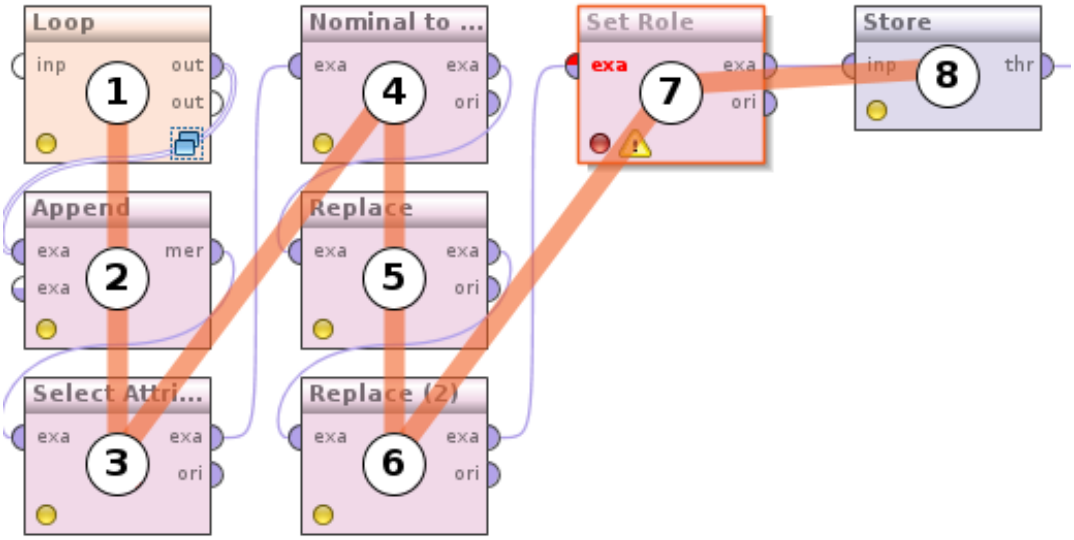

Illustration 3.1: Procés de captació de dades de www.rottentomatoes.com

1. Iterem dues vegades per obtenir les crítiques de les dues pàgines. Dins d'aquest procés, es fa un tall de la pàgina web i es cerca amb expressions regulars i comparació de cadenes de caràcters per trobar el codi HTML on es troben les crítiques de la pel·lícula.
2. Ajuntem totes les crítiques en un únic llistat.
3. Seleccionem només els atributs que ens interessen, en aquest cas, la crítica i la seva valoració (*fresh* o *rotten*).
4. Simplement convertim a text l'atribut que conté la crítica.
5. Convertim el valor de les crítiques '*fresh*' a '*positive*'.
6. Convertim el valor de les crítiques '*rotten'* a '*negative'.*
7. Assignem l'atribut que conté la valoració com a *label*.
8. Guardem les crítiques en el repositori (veure Illustration 3.2: Llistat acotat de crítiques de la pel·lícula It follows extretes de www.rottentomatoes.com) per ser més tard obtingudes i fer prediccions amb un model entrenat.

Row No.	label	text
1	positive	So this is when I am going to break from the critical chorus of praise. It Follows wasn't fla
2	positive	It Follows is bold, provocative, original, artfully made, perfectly acted, and creepy as hell.
3	positive	The next time some old lady starts following you with an emotionless stare on her face, y
4	positive	It Follows has an impressively sustained sense of dread, less explicit gore than measurec
5	positive	Formally and visually it is a work of real craftsmanship, and it displays a conscious unders
6	negative	"Follows" Fails to Follow Through
7	positive	It's a wildly fun conceit ... Mitchell's atmospheric rendering of It Follows gives the film a fa

Illustration 3.2: Llistat acotat de crítiques de la pel·lícula It follows extretes de www.rottentomatoes.com

4 Definició d'objectius

4.1 Objectiu

L'objectiu de tot analitzador de sentiment és determinar si una noticia o si un tros de text és de caire positiu, negatiu o neutre.

En el meu cas, he utilitzat un conjunt de dades per entrenar un model per que sigui capaç de classificar crítiques de les pel·lícules en positives o negatives i, finalment, predir el sentiment de les noves crítiques que es facin.

4.2 Models

Dels diferents models estudiats a classe, els més reconeguts i utilitzats per analitzadors de sentiment són els *SVM (Support Vector Machines)* i les *Naive Bayes*. Gràcies a que amb *RapidMiner* és molt fàcil canviar de model he fet proves amb els dos per avaluar les avantatges i els inconvenients de cadascun d'ells.

A més a més del model he hagut de trobar la configuració (els paràmetres) adient per cada model, ja que això pot fer variar molt la seva eficiència i precisió.

5 Pre-procés

Per tal de pre-processar les dades dels fitxers de text utilitzarem l'operador de *Process Documents from Files*. Aquest operador crea un vector (veure Illustration 5.1: Procés de processament de documents) de paraules i ocurrències a partir dels fitxers de text.

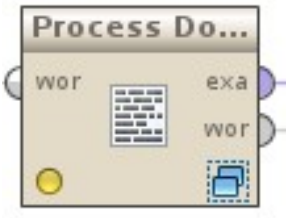

Illustration 5.1: Procés de processament de documents

En el meu cas, per la creació del vector utilitzaré la metodologia *TF-IDF* (*Term Frequency-Inverse Document Frequency*) per tal de donar més pes a les paraules que ocorren menys sovint i evitarem les paraules que sorgeixen molt poques o moltes vegades (veure Illustration 5.2: Vector de paraules creat amb el mètode TF-IDF)

Word	Attribute Name	Total Occurences	Document Occurences	positive	negative
abandon	abandon	91	87	51	40
abil	abil	187	165	92	95
abl	abl	339	290	205	134
abov	abov	170	153	103	67
absolut	absolut	229	199	114	115
absurd	absurd	69	64	23	46

Illustration 5.2: Vector de paraules creat amb el mètode TF-IDF

Per l'altra banda, com a resultat també obtenim un llistat de totes les notícies que conté totes les paraules del vector de paraules com a atributs, amb un pes determinat depenent del nombre de ocurrències en aquell document, en els documents en general i en si ocorre en documents positius o negatius (veure Illustration 5.3: Llistat de crítiques amb el vector de paraules com atribut).

Row No.	label	metadata...	metadata...	metadata...	abandon	abil	abl
1	negative	cv546_127	/home/mar	Feb 16, 20	0	0	0
2	negative	cv622_858	/home/mar	Feb 16, 20	0	0.047	0
3	negative	cv944_150	/home/mar	Feb 16, 20	0.089	0	0.055
4	negative	cv834_231	/home/mar	Feb 16, 20	0	0	0.034

Illustration 5.3: Llistat de crítiques amb el vector de paraules com atribut

Com hem pogut veure a la Illustration 5.1: Procés de processament de documents, aquest operador és aniuat i permet personalitzar la creació del vector de paraules (veure Illustration 5.4: Detall del procés de processament de documents).

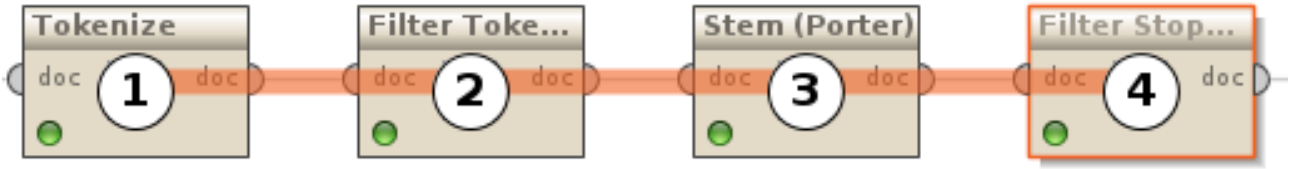

Illustration 5.4: Detall del procés de processament de documents

1. *Tokenize*. Divideix tot el text en diferents *tokens* (parts més petites) d'acord a una política, en el meu cas diferenciem *tokens* quan hi ha caràcters que no són lletres.
2. *Filter Tokens (length)*. De tots els *tokens*, eliminem els que són molt curts o molt llargs (dades atípiques).
3. *Stem* (Porter). Apliquem *Stemming* mitjançant l'algorisme de Porter, el qual extreu de totes les paraules només el *stem*. El *stem* ve a ser l'arrel de la paraula la qual es manté intacta en les diferents variacions i és la part que més significat té d'una paraula (per exemple *game, gamer, games, fish, fishing, fished*).
4. *Filter Stopwords (English)*. Eliminem totes les paraules que no aporten valor des de la perspectiva del sentiment com poden ser els articles o els pronoms i finalment obtenim un vector com el que s'ha vist anteriorment (veure Illustration 5.2: Vector de paraules creat amb el mètode TF-IDF).

6 Modelització

En quant a la estratègia de validació he utilitzat *X-Validation* amb mostreig estratificat i deu validacions. Considero que *X-Validation* és molt millor que *Split validation* ja que permet fer múltiples validacions amb diferents conjunts d'entrenament i de test (veure Illustration 6.2: Procés de X-Validation).

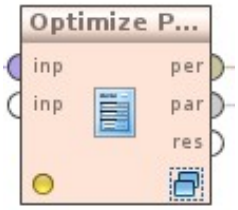

Illustration 6.1: Procés d'optimització de parametres

D'altra banda, per trobar la millor configuració dels diferents models i per tal d'augmentar al màxim la seva qualitat he utilitzat el operador *Optimize parameters* (veure Illustration 6.1: Procés d'optimització de parametres) el qual et permet executar un subprocés múltiples vegades, variant els paràmetres de configuració i avaluar quin d'ells proporciona un millor resultat.

6.1 Support Vector Machine

Els paràmetres més rellevants per les *SVM* són el tipus de funció que utilitza per classificar i el paràmetre de complexitat C. Pel que fa al tipus de funció he utilitzat una funció lineal ja que és molt simple i dona com a resultat un model més genèric que en la majoria de casos és molt bo. D'altra banda el paràmetre ens permet construir un model més rígid amb valors de C grans i un model més flexible amb valors de C més petits. No obstant, s'ha d'anar amb cura ja que valors extrems podrien produir *overfitting* i *underfiting* respectivament. Per aquesta raó, jo he aplicat l'optimització de C en un rang força acotat, entre 0.5 i -0.5.

6.2 Naive bayes

En aquest cas, no hi ha cap paràmetre que es pugui modificar de manera que només hi ha una única configuració possible.

7 Anàlisi de resultats

7.1 Indicadors de qualitat

Per avaluar la qualitat del model he utilitzat el operador *Performance* (veure Illustration 7.1: Procés d'avaluació de performance)

Aquest operador té com a sortida un vector amb diversos paràmetres: *Accuracy*, *Precision*, *Recall*, *AUC*. Jo he prestat especial atenció a la *accuracy* ja que al cap hi a la fi és qui ens diu si ha encertat o no, i a la precisió per veure si quan encerta ho fa amb molta o molt poca confiança.

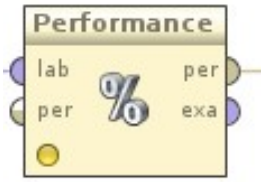

Illustration 7.1: Procés d'avaluació de performance

7.2 Aplicació del model

7.2.1 Naive Bayes

Aquest model (veure Illustration 7.2: Detall del procés de X-Validation amb el model Naive Bayes) s'ha utilitzat molt en anàlisi de sentiment per diverses raons. En primer lloc, és un algorisme simple i que dóna un molt bon resultat i, en segon lloc, per que el seu rendiment és molt bo ja que tant els períodes d'entrenament com els de test són molt curts i el seu consum tant de CPU i com de memòria són molt baixos.

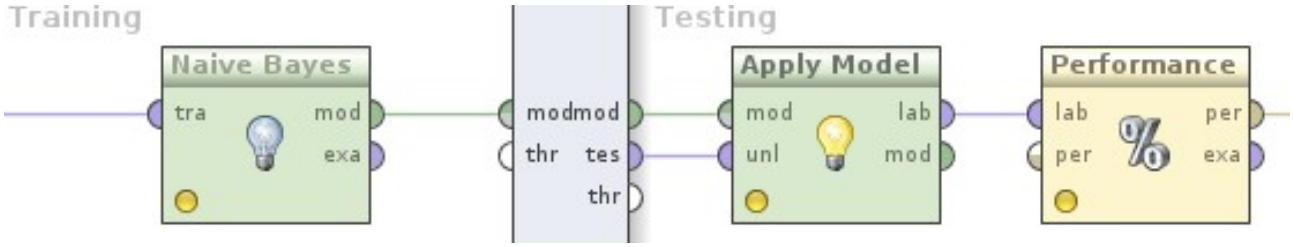

Illustration 7.2: Detall del procés de X-Validation amb el model Naive Bayes

7.2.2 Support Vector Machine

SVM també és un model força utilitzat per anàlisi de sentiment ja que sol donar un molt bon resultat. Té sentit ja que *SVM* construeix una funció, lineal en el meu cas, on és pondera cada atribut (paraula) amb un cert pes. No obstant, donat que és un model que va iterant sobre sí mateix, és molt lent ja que a cada iteració mesura l'error, intenta reduir-lo i torna a iterar.

Per tal de crear el model amb la *SVM* simplement he canviat la caixa del procés (veure Illustration 7.3: Detall del procés X-Validation amb el model SVM(Linear)).

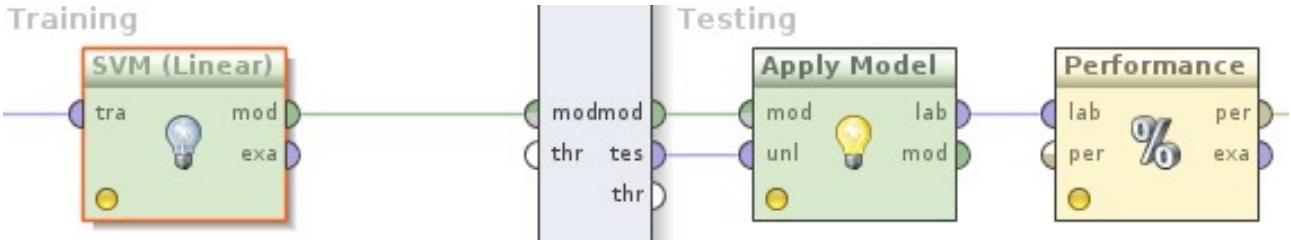

Illustration 7.3: Detall del procés X-Validation amb el model SVM(Linear)

Llavors, he executat el procés amb optimització de paràmetres per avaluar quina configuració donava un millor resultat. Podem veure a la Table 7.1: Comparativa de la performance variant el paràmetre C del model SVM que pràcticament no hi ha diferencia quan la variació d'aquest paràmetre és tan petita. No obstant, variant altres paràmetre de *SVM* o d'altres algorismes, això pot suposar acceptar o rebutjar un model.

Finalment, hem escollit el model amb C = 0.

Log (96 rows, 2 columns)

C	Performance
0	0.855
-0.279	0.850
-0.288	0.850
-0.308	0.850
-0.301	0.850
-0.315	0.850

Table 7.1: Comparativa de la performance variant el paràmetre C del model SVM

7.3 Anàlisi de resultat i qualitat

Per tal de decidir quin model és millor he comparat els resultats del model de *Naive Bayes* i amb el millor model de *SVM* (veure Table 7.2: Comparació de models). Tenint en compte que només estic avaluant l'encert del model, clarament podem veure que *SVM* és molt millor.

Model	Performance	Precisió
Naive Bayes	76.70% +/- 2.01%	77.72% +/- 3.11%
SVM Linear (C=0)	83.75% +/- 2.58%	82.58% +/- 3.71%

Table 7.2: Comparació de models

8 Prediccions

8.1 Film Affinity

Aquest procés és força senzill, simplement aplico el model a les crítiques que vaig descarregar de manera manual per veure què tan encertades són les prediccions que faig.

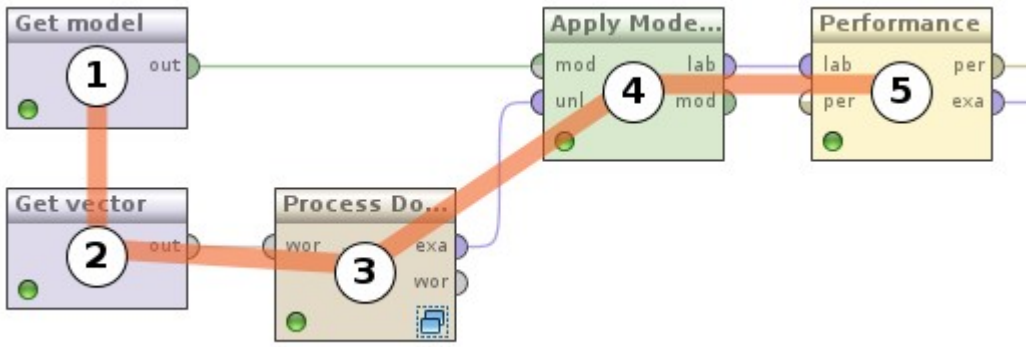

Illustration 8.1: Procés predictor per noves observacions de FilmAffinity

1. Llegim del repositori el model *SVM* generat anteriorment.
2. Llegim del repositori el vector de *tokens* generat durant la creació del model.
3. Processem les noves observacions amb el vector de *tokens* obtingut anteriorment.
4. Apliquem el model de la *SVM* a les crítiques.
5. Avaluem la seva qualitat de les prediccions (veure Illustration 8.2: Llistat de les prediccions de Film Affinity fetes per la SVM).

Row No.	label	prediction...	confid...	confidenc...
2	negative	negative	0.949	0.051
4	negative	negative	0.762	0.238
1	negative	negative	0.544	0.456
5	negative	negative	0.524	0.476
3	negative	negative	0.519	0.481
7	positive	negative	0.516	0.484
6	positive	positive	0.379	0.621
9	positive	positive	0.181	0.819
8	positive	positive	0.121	0.879
10	positive	positive	0.112	0.888

Illustration 8.2: Llistat de les prediccions de Film Affinity fetes per la SVM

Com podem veure, encertem 9/10, i la gran majoria d'elles amb una confiança força elevada (precisió al voltant del 90%). De fet, la única critica que fallem, ho fem amb una

confiança molt petita. Tot i que no és comú que surti un valor més alt que amb les dades d'entrenament i de testeig, la impressió que hem de tenir, és que és un predictor bo i molt fiable.

8.2 Rotten Tomatoes

Per tal d'avaluar el comportament del model amb observacions molt més curtes que no les anteriors he utilitzat el procés de *scraping* web descrit al principi del informe (veure Illustration 3.1: Procés de captació de dades de www.rottentomatoes.com). Llavors, el procés d'aplicació del model queda molt més simple (veure Illustration 8.3: Procés predictor per noves observacions de Rotten Tomatoes).

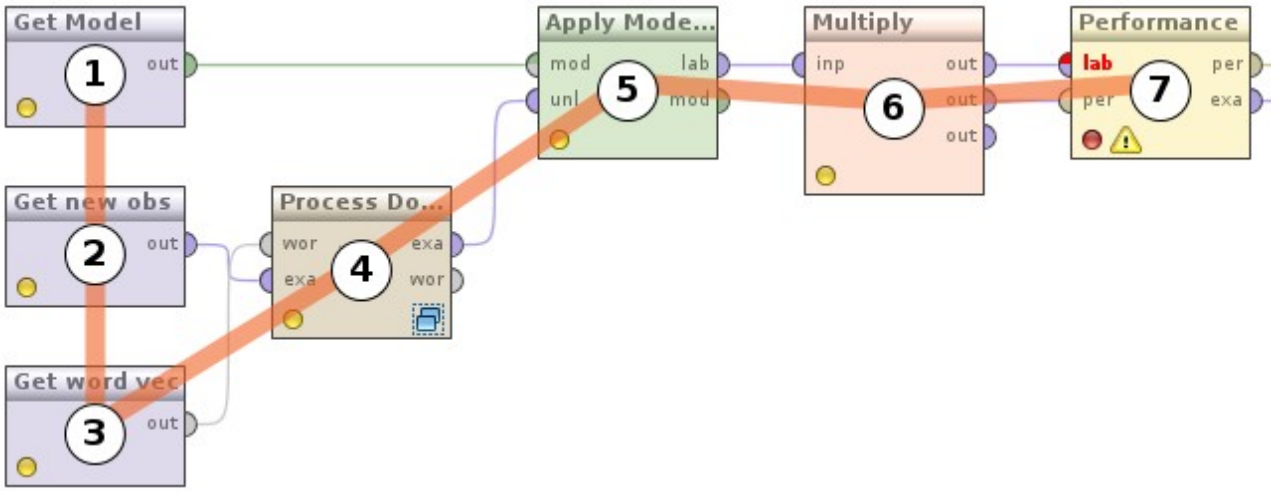

Illustration 8.3: Procés predictor per noves observacions de Rotten Tomatoes

1. Llegim del repositori el model *SVM* generat anteriorment.
2. Llegim del repositori les noves crítiques de pel·lícules generades amb el procés de *scrapping*.
3. Llegim del repositori el vector de *tokens* generat durant la creació del model.
4. Processem les noves crítiques amb el vector de *tokens* obtingut anteriorment.
5. Apliquem el model de la *SVM* a les crítiques.
6. Enviem una sortida cap als resultats, per veure com són les prediccions (veure Illustration 8.4: Llistat acotat de les prediccions de Rotten Tomatoes fetes per la SVM) i una altra cap a *Performance*.
7. Avaluem la seva qualitat de les prediccions.

Row No.	label	prediction(l...	confidenc...	confidenc...	abandon	abil	abl	abov	absolut	absurd
1	positive	positive	0.194	0.806	0	0	0	0	0	0
2	positive	positive	0.286	0.714	0	0	0	0	0	0
3	positive	negative	0.521	0.479	0	0	0	0	0	0
4	positive	positive	0.347	0.653	0	0	0	0	0	0
5	positive	positive	0.243	0.757	0	0	0	0	0	0
6	negative	positive	0.496	0.504	0	0	0	0	0	0
7	positive	positive	0.293	0.707	0	0	0	0	0	0
8	positive	positive	0.219	0.781	0	0	0	0	0	0

Illustration 8.4: Llistat acotat de les prediccions de Rotten Tomatoes fetes per la SVM

Per últim, com a resultat obtenim que classifica correctament al voltant del 70% de les crítiques, un valor molt més baix (entre un 15% i 20% més baix) que amb les crítiques de *FilmAffinity* o amb les dades d'entrenament i testeig. A més a més, la precisió també és incomparable, aquí en la majoria de cassos no té prou evidències per classificar una observació amb prou confiança, contràriament amb l'anàlisi anterior.

Tot això és donat a que aquestes crítiques són molt més curtes, i donat que amb el pre-procés ja eliminem algunes paraules (paraules molt curtes, molt llargues, *Stop words*, etc) encara es fan curtes i, per tant, més difícils d'analitzar, ja que una simple paraula pot canviar el resultat global de la predicció. La millor manera d'evitar-lo és tenir un vector de paraules molt gran per tal de aconseguir el màxim de *hits* possible a l'hora d'analitzar les noticies. Això es tradueix, en tenir un conjunt de dades d'entrenament i testeig molt més gran i per tant incrementar la complexitat del model i els temps computacionals.

9 Conclusions

Abans que res m'agradaria destacar la facilitat que suposa utilitzar un programa especialitzat, en aquest cas *RapidMiner* per realitzar anàlisis i prediccions. La dificultat i el temps invertit a fer-ho en Java o *RapidMiner* no es poden comparar. Mentre que per fer aquell algoritme en Java vaig tardar gairebé un mes per la implementació i un altre per trobar la configuració òptima de paràmetres de la fòrmula, amb *RapidMiner* ho he fet en un parell de setmanes.

No obstant, també vull destacar, que la implementació, la idea és força similar. D'una manera semblant a la que ho fa *RapidMiner*, el meu vector de paraules era una base de dades de milers de paraules amb pesos positius i negatius i el meu algoritme simplement buscava aquestes paraules al text i aplicava una funció.

Pel que fa als models de classificació que he fet servir, m'agradaria comentar-los breument. Primer de tot vaig provar el model de *Naive Bayes* el qual tenia un encert al voltant del 75%. Jo personalment, considero que 75% és una bona xifra i a més a més, aquest model destaca per la rapidesa del entrenament, trigava no més de 10 segons en entrenar-se i testejar-se amb *X-Validation*. No obstant, vaig rebutjar aquest model en front de la *SVM*, el qual té un encert quasi de 10 punts més. Tot i aixo, aquest model triga moltíssim en entrenar-se i testejar-se (gairebé 3 minuts), ja que és un model iteratiu i que avalua l'error que comet, el corregeix i torna a calcular. Llavors, la meva conclusió pel que fa als models és que cada aplicació, cada empresa té unes necessitats concretes, a vegades la certesa del model no ho és tot, ja que pot ser no ets pots permetre que el model trigui dies en entrenar-se o en realitzar prediccions. Hem d'avaluar les avantatges i inconvenients de cada model i veure quin s'adapta millor a les nostres necessitats. En el meu cas, al ser un treball pràctic, he escollit la *SVM*, però, si aquest projecte fos per una empresa, probablement hagués escollit les *Naive Bayes* ja que en conjunt, tenint en compte temps i encerts, considero que és millor.

D'altra banda, m'agradaria destacar algunes característiques especials de les dades que he analitzat i que també em vaig trobar quan ho vaig fer fa un parell d'anys i que són problemes molt comuns quan es tracta d'analitzar text.
Primer de tot, he fet prediccions amb dos tipus molt diferents de texts en quant a longitud perquè quan vaig treballar en això fa uns anys, l'anàlisi de comentaris de *Facebook*, *Twitter* i *Blogs*, era entre un 10% i 20% menys encertat que el d'altres textos i volia

analitzar aquest cas.

En el meu cas, he vist un comportament similar. He pogut veure que, amb texts llargs el comportament de l'algorisme és molt més fiable i estable, ja que si es troben paraules atípiques aquestes seran poques i no seran capaces d'afectar el resultat global de l'algorisme donat que hi ha moltes paraules. No obstant, com he comentat, analitzant crítiques molt curtes, de dues o tres línies, una classificació dolenta d'una única paraula pot provocar que la observació es classifiqui globalment de forma incorrecta i fa que l'algorisme sigui molt poc fiable i poc precís. Per aquesta raó, hi ha una diferència de encerts molt gran, un 90% contra un 70% i una diferència de precisió elevadíssima.

L'anàlisi d'aquests textos curts és el gran repte dels analitzadors de sentiment. Ja que molt sovint, a més de ser curts solen contenir faltes d'ortografia, abreviacions o fan servir un llenguatge més informal amb paraules 'urbanes' i de altres llengües. No obstant, avui dia és aquí, en les xarxes socials, on hi ha més activitat i on podem trobar més informació (*Facebook* va comprar *WhatsApp* per 19.000 milions per alguna raó) i per això crec que no passarà gaire temps fins que surtin a la llum nous algoritmes i programes que permetin solucionar els problemes que he esmentat anteriorment per aconseguir un encert més alt.

Un altre problema, encara que no tant important ja que la majoria de vegades és fàcilment solucionable és l'idioma. Depenent de l'idioma hem de cercar o ignorar unes paraules o unes altres. La solució és utilitzar algorismes que mitjançant la freqüència de les lletres o la combinació que es fa d'elles, aconsegueixen determinar l'idioma. Tot i això, quan els textos que s'analitzen són curts aquests algoritmes fallen ja que no tenen prou dades per fer prediccions encertades, un altre motiu pel qual aquests textos són difícils de classificar.

D'altra banda, donat que està de moda fer servir emoticones cada cop més, els analitzadors de sentiment comencen a incloure, com si fossin paraules, cares i símbols que expressen felicitat, ràbia i altres emocions per fer més exactes els seus models.

L'únic problema que considero que mai es podrà solucionar, ja que és un tret de l'ésser humà, és el sentit del humor al escriure, la ironia i el sarcasme. A vegades, ni nosaltres som capaços de detectar-lo quan ho llegim, com podem pretendre que ho faci una maquina?

10 Bibliografia

Bo Pang and Lillian Lee, Movie review Data. USA, Computer Science departament of Cornell Univerisity. Disponible en www.cs.cornell.edu/people/pabo/movie-review-data

Movie Soulmates, Film Affinity. Disponible en www.filmaffinity.com

Aylien, Text analysis blog. Disponible en blog.aylien.com

Technology Blog, Structuring unstructred data. Disponible en www.corequant.com

Flixter Inc, Rotten Tomatoes: Movies | Tv shows | Movie trailers | Reviews. Disponible en www.rottentomatoes.com

Dataprix translation | Software and Information Technologies, Example 22, Optimization of Parameters. Disponible en www.dataprix.net