Pràctica 2: Detecció d'opinions

Processament del Llenguatge Humà

Joan Saurina i Ricós, Sergi Tomàs Martínez

14 d'abril de 2023

Índex

1 El problema

En aquesta pràctica s'ha implementat un detector d'opinions positives o negatives utilitzant algorismes d'aprenentatge supervisat disponibles a la llibreria Scikit-learn (sklearn).

S'utilitza el corpus de crítiques de pel·lícules (Movie Reviews Corpus) disponible a NLTK com a conjunt de dades. Es dissenya i s'aplica un protocol de validació per avaluar el rendiment del model, es selecciona el preprocessament de dades més adequat. Després, s'utilitza la tècnica de CountVectorizer per representar la informació.

Finalment, es calcula la accuracy i les matrius de confusió per avaluar l'exercici del model, i es realitza un anàlisis per interpretar els resultats obtinguts.

2 Partició de les dades

En primer lloc, es carrega textos positius i negatius d'un corpus mitjançant el mètode mr.fileids(). A continuació, es converteix els textos en format brut i es divideix en conjunts de train i test mitjançant la funció train_test_split() de la llibreria scikit-learn. Els textos es divideixen en una divisió de train de 80% i test 20%. La llavor aleatòria està configurada per garantir la reproductibilitat.

Després de dividir les dades, es fusiona els conjunts de train positius i negatius tot conservant l'ordre dels textos de cada conjunt. El mateix es fa amb el conjunt de test. El tren i les llistes de proves resultants es poden utilitzar per entrenar i provar un model d'anàlisi de sentiments.

3 Preprocessament

Es defineix una funció anomenada "preprocessador"que pren una frase com a entrada i li aplica diverses tècniques de preprocessament de text. Aquestes tècniques inclouen:

1. Eliminació de caràcters especials com signes de puntuació, números, etc. mitjançant expressions regulars.
2. Eliminació de caràcters no ASCII mitjançant la biblioteca ünicodedata".
3. Conversió d'espais dobles en un únic espai.
4. Conversió de tot el text a lletres estrictament minúscules.
5. Eliminació del caràcter de salt de línea

Després de definir la funció "preprocessador", s'aplica aquesta funció a tot el

corpus de train mitjançant un bucle. La sortida d'aquest bucle és una nova llista anomenada "train_preprocessat", que conté la versió preprocessada de cada frase de la llista "train".

Cal destacar que "train_preprocessat"és un pas comú de preprocessament per a les formes d'identificació de paraules o synsets que es proposen posteriorment.

En procés descrit anteriorment es segueix igual per contruir test_preprocessat.

En cada un dels apartats següents s'especifica el processamaent concret de les dades que es segueix i com es defineixen les dades de train i test en cada cas.

3.1 Lesk synsets

En aquest processament de les dades s'importa el corpus "movie_reviews"de la biblioteca Natural Language Toolkit (nltk) i es baixa els mòduls necessaris. Aleshores, es crea una llista buida llista_syn i es recorre cada review del conjunt d'entrenament.

Per a cada review, s'identifica el text en paraules individuals i troba el synset corresponent (un conjunt de sinònims) mitjançant l'algoritme Lesk del mòdul de desambiguació del sentit de paraules de nltk. Si es troba un synset, s'afegeix a la llista corresponent dins de llista_syn.

El resultat llista_syn és una llista de llistes que contenen synsets per a cada paraula de cada revisió.

3.1.1 Train lesk

Es crea un "vectoritzador"d'objecte CountVectorizer que convertirà les llistes synset en característiques numèriques.

Després, el codi estableix l'analitzador perquè utilitzi els synsets individuals com a característiques i estableix els llindars màxims i mínims de freqüència del document per a les funcions que s'inclouran a la sortida.

A continuació, transforma les llistes llista_syn en una matriu de recomptes de característiques mitjançant el mètode fit_transform de "vectorizerï crea un DataFrame pandas a partir de la matriu escassa resultant X_train_lesk.

Tot seguit, concatena les etiquetes amb la matriu de característiques per crear un nou DataFrame X_train_l, barreja les files aleatòriament i separa les etiquetes de la matriu de característiques per crear un DataFrame X_train_l separat i una sèrie d'etiquetes y_train_l.

3.1.2 Test lesk

Pel que fa al test, es crea una llista buida llista_syn_test i es recorre cada review del conjunt de proves. Per a cada review, identifica el text en paraules individuals i troba el synset corresponent mitjançant l'algorisme de Lesk. Si es troba un synset, s'afegeix a la llista corresponent dins de llista_syn_test.

A continuació, transforma llista_syn_test en una matriu de recomptes de característiques mitjançant el mètode "transform"de "vectoritzador", crea un DataFrame pandas a partir de la matriu escassa resultant X_test_l, concatena les etiquetes amb la matriu de característiques per crear un nou DataFrame X_test_les", barreja les files aleatòriament i separa les etiquetes de la matriu de característiques per crear un DataFrame separat X_test_lesk i una sèrie d'etiquetes y_test_lesk.

3.2 Synsets més freqüents

En aquest processament de les dades s'utilitza la biblioteca Natural Language Toolkit (NLTK) per preprocessar un corpus de text mitjançant l'extracció dels tres synsets (conjunts de sinònims) més freqüents per a cada paraula del corpus. Es comença important el mòdul WordNet des de NLTK, que proporciona accés a una gran base de dades lèxica de paraules angleses i els seus significats.

A continuació, es defineix un objecte CountVectorizer. Després, s'aplica el CountVectorizer a les dades d'entrenament preprocessades i emmagatzema els noms de les característiques resultants (és a dir, les paraules del corpus) en una llista anomenada "names".

A continuació, s'inicialitza un diccionari buit anomenat 'paraules' i itera per cada nom de la llista de 'noms'. Per a cada nom, es recupera els synsets (és a dir, conjunts de sinònims) de WordNet i compta el nombre de vegades que cada lema (és a dir, una forma de paraula específica) apareix al synset. S'emmagatzema els synsets i els seus respectius recomptes de lemes en una llista, que s'afegeix com a valor al diccionari 'paraules' amb el nom com a clau.

Després d'iterar tots els noms i els seus synsets associats, s'ordena els synsets per a cada paraula en 'paraules' pel nombre de recomptes de lemes en ordre descendent. Finalment, es trunca la llista de synsets per a cada paraula en "paraules"només als tres synsets més freqüents.

NOTA: totes les variables relacionades amb synsets més freqüents tenen un 5 i no un 3 perquè originalment s'havia pensat fer els 5 synsets més freqüents.

3.2.1 Train 3 synsets més freqüents

Primer, s'extreu tots els synsets únics del diccionari "paraulesï els emmagatzema en una llista anomenada l·lsynsets".

A continuació, es crea un diccionari buit anomenat 'd1' i l'inicia amb claus iguals als synsets de 'lsysets' i valors iguals a una llista de zeros amb una longitud igual al nombre de files de les dades d'entrenament X.

A continuació, s'itera sobre cada fila i cada paraula de les dades de text preprocessades, i per a cada paraula itera sobre els seus synsets més freqüents. Per a cada synset, s'afegeix el recompte de la paraula corresponent en aquest synset a l'element adequat a "d1".

Després de comptar les ocurrències de paraules per a cada synset de cada fila, es crea un nou DataFrame anomenat "X_train_5_syn"a partir del diccionari "d1", on cada columna correspon a un synset i cada fila correspon a un document de les dades d'entrenament.

A continuació, es concatena el DataFrame "X_train_5_syn"amb les etiquetes de sentiment de cada document per crear les dades d'entrenament finals "X_train_5". Es barreja aleatòriament les files de "X_train_5"mitjançant el mètode "mostraï es restableix l'índex per assegurar-se que les files s'ordenen aleatòriament. Finalment, s'extreu les etiquetes de sentiment i es deixa anar del DataFrame 'X_train_5' per crear la variable objectiu 'y_train_5'.

3.2.2 Test 3 synsets més freqüents

Primer de tot, es transforma les dades de prova preprocessades en una matriu de recomptes de paraules utilitzant el mateix objecte ÇountVectorizer"que abans. La matriu resultant es converteix en un DataFrame 'X' amb files corresponents a documents i columnes corresponents a paraules.

A continuació, es crea un diccionari buit "d1ï l'inicia amb claus iguals als synsets de l·lsynsetsï valors iguals a una llista de zeros amb una longitud igual al nombre de files de les dades de prova X.

Tot seguit, s'itera sobre cada fila i cada paraula de les dades de prova preprocessades i, per a cada paraula, s'itera sobre els seus synsets més freqüents. Per a cada synset, s'afegeix el recompte de la paraula corresponent en aquest synset a l'element adequat a "d1".

Després de comptar les ocurrències de paraules per a cada synset de cada fila, es crea un nou DataFrame anomenat "X_test_5_syn"del diccionari "d1", on cada columna correspon a un synset i cada fila correspon a un document

de les dades de prova.

A continuació, es concatena el DataFrame "X_test_5_syn"amb les etiquetes de sentiment de cada document per crear les dades de prova finals "X_test_5". Es barreja les files de "X_test_5"aleatòriament. Finalment, s'extreu les etiquetes de sentiment i es deixa anar del DataFrame "X_test_5"per crear la variable objectiu "y_test_5".

3.3 Paraules lematitzades

En aquest processament de les dades s'utilitza la biblioteca Natural Language Toolkit (NLTK) per realitzar la lematització de les dades de text d'entrada. La lematització és el procés de reduir les paraules a la seva forma base o diccionari, anomenada lema. Això ajuda a reduir la dimensionalitat de les dades del text agrupant diferents formes de la mateixa paraula.

La biblioteca NLTK es baixa primer per accedir als recursos necessaris per al processament de text, inclosos els corpus "averaged_perceptron_taggerï "wordnet". També es descarrega el corpus 'omw-1.4', que proporciona accés a l'Open Multilingual WordNet, una base de dades lèxica de paraules en diversos idiomes.

La funció de lematització 'lematitzar' pren com a entrada una tupla d'etiquetes (paraula, part de la parla) i retorna el lema de la paraula en funció de la seva part de la parla. Les etiquetes de part de veu s'assignen als codis de part de veu corresponents de WordNet mitjançant un diccionari.

La funció de lematització s'aplica a cada document del corpus d'entrada, després d'haver realitzat primer l'etiquetatge parcial de les paraules del document mitjançant la funció 'pos_tag' de NLTK. Els documents lematitzats resultants s'emmagatzemen en una llista anomenada 'train_lematitzat'.

La classe 'CountVectorizer' de la biblioteca 'sklearn' s'utilitza llavors per convertir la llista de documents lematitzats en un DataFrame pandas anomenat "X_train_word". També s'aplica l'eliminació de stop words i es filtra les paraules que es produeixen amb massa poca o massa freqüència, en funció dels valors especificats de "max_dfi "min_df".

3.3.1 Train paraules lematitzades

Es realitza un preprocessament addicional en un conjunt de dades que ja s'ha preprocessat i convertit en lemes. L'objectiu d'aquest codi és eliminar qualsevol paraula que no sigui útil per a l'anàlisi de sentiments.

En primer lloc, el codi crea dos marcs de dades basats en el valor de la primera

columna del DataFrame original (documents positius i negatius). A continuació, es calcula la suma de cada columna de cada DataFrame. Seguidament, es fa un bucle per cada columna del primer DataFrame i compara la seva suma amb la columna corresponent del segon DataFrame. Si la diferència absoluta entre les sumes és menor o igual a 15, l'índex de columna s'afegeix a la llista de 'result'.

Aleshores, les columnes de la llista de 'result' s'eliminen del DataFrame original. Es crea un nou diccionari de vocabulari sense les paraules els índexs de les quals es troben a la llista de results. El codi crea un nou diccionari que assigna els valors originals als seus nous índexs i després crea un nou diccionari amb els índexs actualitzats. Es crea un nou DataFrame amb les paraules reordenades segons el nou ordre del vocabulari de l'objecte CountVectorizer.

Finalment, el codi concatena les etiquetes de sentiment amb el DataFrame reordenat i barreja les files aleatòriament. Així, s'ha contruït 'y_train_word_lematized' i 'X_train_word_lematized'.

3.3.2 Test paraules lematitzades

En primer lloc, les paraules de cada document del conjunt de proves es lematitzen utilitzant el mateix lematitzador utilitzat per al conjunt d'entrenament. Aleshores, el vocabulari de l'objecte CountVectorizer utilitzat per al conjunt d'entrenament s'actualitza per reflectir el nou vocabulari resultant de l'eliminació de columnes amb poca diferència en documents positius i negatius.

Després d'actualitzar el vocabulari, el conjunt de proves es transforma en una matriu de recomptes de paraules mitjançant l'objecte CountVectorizer actualitzat. Aquesta matriu es converteix en un DataFrame pandas i es concatena amb un DataFrame d'etiquetes, on les primeres 200 entrades corresponen a ressenyes positives i les últimes 200 entrades corresponen a ressenyes negatives.

Finalment, el DataFrame resultant es barreja aleatòriament i es divideix en dos marcs de dades separats: un que conté les etiquetes (y_test_word_lematized) i un que conté les dades del recompte de paraules transformades (X_test_word_lematized). La columna d'etiquetes s'elimina de l'últim DataFrame.

3.4 Paraules originals

L'objectiu és proporcionar una línia de base per a la comparació amb els resultats obtinguts mitjançant les dades dels preprocessaemnts anteriorment exposats. L'ús de les dades d'entrenament preprocessades originals sense cap processament addiciona permet avaluar l'impacte de qualsevol pas addicional de preprocessament en el rendiment de l'algorisme de classificació.

3.4.1 Train paraules originals

Es crea un objecte vectoritzador de recompte amb alguns paràmetres i l'ajusta a les dades d'entrenament preprocessades sense cap preprocessament especial addicional.

A continuació, es transforma les dades d'entrenament preprocessades en una matriu de recomptes de testimonis, que s'emmagatzema en un DataFrame.

Tot seguit, el DataFrame es concatena amb un DataFrame que conté les etiquetes i DataFrame resultant es barreja aleatòriament.

Finalment, les etiquetes s'extreuen en una variable independent i el DataFrame es modifica per eliminar la columna d'etiquetes.

3.4.2 Test paraules originals

Es crea un conjunt test utilitzant l'objecte CountVectorizer original sense cap pas de preprocessament addicional.

A continuació, utilitza l'objecte CountVectorizer countvectorizer_original definit anteriorment per transformar les dades de prova test_preprocessat en una matriu de freqüències de paraules, que després es converteix en un DataFrame 'X_test_world_original'.

Seguidament, aquest DataFrame es concatena amb les etiquetes per crear un DataFrame final 'X_test_world_original' amb la mateixa estructura que el DataFrame d'entrenament.

Finalment, el DataFrame es barreja i les etiquetes i la columna de sentiment es separen en variables separades 'y_test_world_original' i 'X_test_world_original', respectivament.

4 Models d'aprenentatge supervisat

S'entrena tres models d'aprenentatge supervisat: **Decision Tree**, **Random Forest** i **XGBoost**. Per preparar les dades, s'aplica les quatre tècniques de preprocessament exposades anteriorment a l'apartat 3.

S'utilitza CountVectorizer amb dos conjunts de paràmetres: max_df i min_df. El primer conjunt de paràmetres serà 0,9 i 0,1 respectivament, seguit d'un altre conjunt amb 0,85 i 0,05. Es prova cada model amb cada combinació de tècniques i paràmetres de preprocessament per determinar el millor enfocament pel conjunt de dades. Aquest procés permet comparar el rendiment de cada model i escollir-ne el millor.

Per cada model es duur a terme Random Search i Grid Search. En ambdós casos s'inclou 3-fold cross-validation.

Random Search implica seleccionar hiperparàmetres a l'atzar d'una distribució especificada. La distribució pot ser uniforme o no uniforme, i determina la probabilitat de seleccionar cada hiperparàmetre. Se seleccionen aleatòriament un nombre determinat de combinacions d'hiperparàmetres i s'avaluen al conjunt de validació. A continuació, segons el rendiment mostrat, se selecciona un rang d'hiperparàmetres a provar en el Grid Search.

Grid Search, d'altra banda, implica especificar una llista d'hiperparàmetres i els seus respectius valors a provar. Els hiperparàmetres es comencen a provar sistemàticament a través de totes les combinacions possibles. A continuació, es seleccionen els hiperparàmetres amb el millor rendiment com a hiperparàmetres finals per al model.

4.1 Mètriques

A l'hora d'escollir les mètriques que avaluen el rendiment dels diferents models s'ha tingut en compte el fet que Random Search i Grid Search s'utilitzen per ajustar els hiperparàmetres d'un model i tenen com a objectiu maximitzar el rendiment del model en el conjunt de validació (o cross-validation). En aquest cas, la puntuació f1 és una mètrica més adequada per utilitzar durant l'ajustament dels hiperparàmetres perquè té en compte tant la precisió com el recall. Atès que l'objectiu és trobar els hiperparàmetres que donen com a resultat el millor rendiment global del conjunt de validació, la puntuació f1 és una mètrica més completa per utilitzar que la accuracy sola.

Tanmateix, quan es prediu al conjunt de proves, la accuracy és una mètrica més habitual perquè proporciona una indicació clara del percentatge d'instàncies classificades correctament. Tot i que la puntuació f1 segueix sent una mètrica valuosa a tenir en compte, la accuracy és més fàcilment interpretable i propor-

ciona una avaluació més directa del rendiment del model.

En definitiva s'utilitza la puntuació f1 pel Random Search i Grid Search, i la accuracy per l'avaluació final del model en el conjunt de test.

4.2 Decision Tree

4.2.1 Motivació

Els arbres de decisió són un tipus d'algorisme d'aprenentatge automàtic que es pot utilitzar per a tasques de classificació i regressió. Funcionen creant un model de decisions semblant a un arbre basat en determinades característiques o característiques de les dades.

A la part superior de l'arbre hi ha un node arrel que representa tota la població o mostra, i a partir d'aquest node es creen branques per representar diferents decisions o classificacions. A continuació, l'arbre es divideix en subconjunts cada cop més petits en funció dels valors de determinades característiques, fins que el procés resulta en un node fulla (un node terminal) que representa una predicció o decisió.

Hi ha diverses motivacions per utilitzar arbres de decisió, com ara que són fàcils d'entendre i interpretar, fins i tot per a persones amb poc coneixement del problema en qüestió. Això els fa útils per explicar la lògica darrere d'una decisió o predicció als altres, són ràpids de construir i utilitzar, el que els fa adequats per a grans conjunts de dades i aplicacions en temps real, són resistents al soroll i poden gestionar dades que falten o incompletes, cosa que els fa robusts davant la incertesa, es poden utilitzar per a una àmplia gamma de tasques, com ara la classificació, la regressió i la selecció de característiques, i poden gestionar tant dades contínues com categòriques.

En els arbres de decisió, només es realitza Grid Search per a l'ajustament dels hiperparàmetres. Això es deu al fet que els arbres de decisió són computacionalment barats d'entrenar en comparació amb altres algorismes.

4.2.2 Discussió dels hiperparàmetres disponibles i els valors usats

La classe sklearn.tree.DecisionTreeClassifier de la llibreria scikit-learn té diversos hiperparàmetres que es poden ajustar per controlar el comportament del model d'arbre de decisió. N'hi ha molts, aquí els més importants:

1. **max_depth**: aquest paràmetre especifica la profunditat màxima de l'arbre. El valor predeterminat és None, la qual cosa significa que l'arbre no està limitat en profunditat i pot créixer fins que totes les fulles siguin pures. Establir un valor per a max_depth pot ajudar a evitar el sobreajust limitant la complexitat del model.

2. **min_samples_split**: aquest paràmetre especifica el nombre mínim de mostres necessàries per dividir un node intern. El valor per defecte és 2, el que significa que un node ha de tenir almenys dues mostres per ser considerades per dividir-se.

3. **criterion**: aquest paràmetre especifica la funció utilitzada per avaluar la qualitat d'un split. El valor per defecte és "gini", que mesura la puresa dels nodes en funció de l'índex d'impureses de Gini. Una altra opció és "entropia", que mesura la puresa dels nodes en funció de l'entropia de Shannon.

Es decideix provar combinacions dels hiperaparètres descrits anteriorment entre els rangs i conjunt de valors següents:

max_depth = [5, 8, 10, 12]
min_samples_split = [2, 4, 6]
criterion = ['gini', 'entropy']

En establir el paràmetre **random_state**, es pot assegurar que es seleccionen les mateixes mostres aleatòries cada vegada que s'executa el codi, cosa que pot ser útil per a finalitats de reproductibilitat. La llavor utilitzada per l'experiment és la 23122003.

4.2.3 Decision Tree (max_df 0.90, min_df 0.1)

La combinació d'hiperparàmetres que opté un rendiment major en el Grid Search de cadascuna de les tècniques de preprocessament proposades és la següent:

Grid Search Decision Tree				
Tèc. Prep.	max d.	min s. s.	crit.	F1
Lesk syn.	5	2	gini	0.669
Synsets freq.	5	4	gini	0.654
Paraules lem.	8	6	entropy	0.646
Paraules orig.	5	6	gini	0.658

Taula 1: Grid Search Decision Tree

Finalment, amb aquests hiperparàmetres i en les dades d'entrenament s'aconsegueix l'accuracy següent:

Test	
Tèc. Prep.	Accuracy
Lesk syn.	0.660
Synsets freq.	0.670
Paraules lem.	0.6725
Paraules orig.	0.6375

Taula 2: Test Decision Tree

4.2.4 Decision Tree (max_df 0.85, min_df 0.05)

La combinació d'hiperparàmetres que opté un rendiment major en el Grid Search de cadascuna de les tècniques de preprocessament proposades és la següent:

Grid Search Decision Tree				
Tèc. Prep.	max d.	min s. s.	crit.	F1
Lesk syn.	5	2	gini	0.669
Synsets freq.	5	2	gini	0.657
Paraules lem.	8	2	entropy	0.678
Paraules orig.	5	6	entropy	0.677

Taula 3: Grid Search Decision Tree

Finalment, amb aquests hiperparàmetres i en les dades d'entrenament s'aconsegueix l'accuracy següent:

Test	
Tèc. Prep.	Accuracy
Lesk syn.	0.653
Synsets freq.	0.650
Paraules lem.	0.645
Paraules orig.	0.6375

Taula 4: Test Decision Tree

4.3 Random Forest

4.3.1 Motivació

Un random forest és un mètode d'aprenentatge conjunt per a la classificació i la regressió que utilitza múltiples arbres de decisions per fer prediccions. Cada arbre de decisió del bosc s'entrena en un subconjunt aleatori diferent de les dades d'entrenament, i la predicció final es fa fent la mitjana de les prediccions de tots els arbres del bosc.

Hi ha diverses motivacions per utilitzar arbres de decisió, com ara que el model de bosc aleatori combina les prediccions de molts arbres de decisió, sovint pot fer prediccions més precises que un arbre de decisió únic, que en entrenar cada arbre de decisió en un subconjunt diferent de dades i fent la mitjana de les prediccions, el model de bosc aleatori pot reduir el sobreajust en comparació amb un arbre de decisió únic, que té capacitat de manejar dades d'alta dimensió.

4.3.2 Discussió dels hiperparàmetres disponibles i els valors usats

La classe sklearn.ensemble.RandomForestClassifier de la llibreria scikit-learn té diversos hiperparàmetres que es poden ajustar per controlar el comportament del model de bosc aleatori. N'hi ha molts, aquí els més importants:

1. **n_estimators**: aquest paràmetre especifica el nombre d'arbres de decisió al bosc. El valor predeterminat és 100.
2. **max_features**: aquest paràmetre determina el nombre màxim de funcions que es poden utilitzar en cada arbre de decisió individual del bosc aleatori. Es pot establir en un nombre fix o en una fracció del nombre total de funcions. El valor per defecte és "auto", que estableix max_features=sqrt(n_features). Establir max_features a un valor més baix pot ajudar a prevenir el sobreajust.
3. **max_depth**: aquest paràmetre especifica la profunditat màxima dels arbres del bosc. El valor predeterminat és None, el que significa que els arbres no estan limitats en profunditat i poden créixer fins que totes les fulles siguin pures. Establir un valor per a max_depth pot ajudar a evitar un sobreajust limitant la complexitat dels arbres.
4. **min_samples_split**: aquest paràmetre especifica el nombre mínim de mostres necessàries per dividir un node intern. El valor per defecte és 2, el que significa que un node ha de tenir almenys dues mostres per ser considerades per dividir-se.
5. **min_samples_leaf**: aquest paràmetre especifica el nombre mínim de mostres necessaris per estar en un node full. El valor per defecte és 1, el que significa que un node full ha de tenir almenys una mostra.

6. **bootstrap**: aquest paràmetre controla si el mostreig d'arrencada s'utilitza o no per crear les mostres individuals utilitzades per a cada arbre del bosc aleatori. Si bootstrap=True, les mostres es dibuixen amb substitució del conjunt de dades original, donant com a resultat mostres diferents per a cada arbre. Si bootstrap=Fals, les mostres es dibuixen sense reemplaçament, donant com a resultat les mateixes mostres per a cada arbre.

Es decideix provar combinacions dels hiperaparètres descrits anteriorment entre els rangs i conjunt de valors següents:

n_estimatos = [5, 10, 20, 50, 100]
max_features = ['log2', 'sqrt']
max_depth = [2, 5, 8, 10]
min_samples_split = [2, 5, 7, 8, 10, 20]
min_samples_leaf = [1, 2, 4, 8, 10]
bootstrap = [True, False]

En establir el paràmetre **random_state**, es pot assegurar que es seleccionen les mateixes mostres aleatòries cada vegada que s'executa el codi, cosa que pot ser útil per a finalitats de reproductibilitat. La llavor utilitzada per l'experiment és la 23122003.

4.3.3 Random Forest (max_df 0.90, min_df 0.1)

La combinació d'hiperparàmetres que opté un rendiment major en el Random Search de cadascuna de les tècniques de preprocessament proposades és la següent:

Random Search Random Forest							
Tèc. Prep.	n. est.	min.s.s.	min.s.l.	max f.	max d.	boot.	F1
Lesk syn.	100	8	10	log2	10	True	0.770
Synsets freq.	100	5	1	log2	8	True	0.737
Paraules lem.	100	2	10	log2	10	False	0.744
Paraules orig.	100	2	10	log2	10	False	0.758

Taula 5: Random Serach Random Forest

Dels resultats anteriors s'obté un nou rang d'hiperparàmetres que es prova en el Grid Search:

Grid Search Random Forest							
Tèc. Prep.	n. est.	min.s.s.	min.s.l.	max f.	max d.	boot.	F1
Lesk syn.	90	7	9	log2	9	True	0.764
Synsets freq.	100	7	7	log2	10	True	0.750
Paraules lem.	90	7	8	log2	9	False	0.763
Paraules orig.	90	7	8	log2	10	False	0.752

Taula 6: Grid Search Random Forest

Finalment, amb aquests hiperparàmetres i en les dades d'entrenament s'aconsegueix l'accuracy següent:

Test	
Tèc. Prep.	Accuracy
Lesk syn.	0.788
Synsets freq.	0.732
Paraules lem.	0.838
Paraules orig.	0.805

Taula 7: Test Random Forest

4.3.4 Random Forest (max_df 0.85, min_df 0.05)

La combinació d'hiperparàmetres que opté un rendiment major en el Random Search de cadascuna de les tècniques de preprocessament proposades és la següent:

Random Search Random Forest							
Tèc. Prep.	n. est.	min.s.s.	min.s.l.	max f.	max d.	boot.	F1
Lesk syn.	100	2	10	log2	10	False	0.801
Synsets freq.	100	10	8	log2	8	Fakse	0.786
Paraules lem.	100	5	10	log2	8	False	0.791
Paraules orig.	100	2	8	sqrt	10	False	0.799

Taula 8: Random Search Random Forest

Dels resultats anteriors s'obté un nou rang d'hiperparàmetres que es prova en el Grid Search:

Grid Search Random Forest							
Tèc. Prep.	n. est.	min.s.s.	min.s.l.	max f.	max d.	boot.	F1
Lesk syn.	100	7	10	log2	8	False	0.791
Synsets freq.	90	7	9	log2	8	False	0.799
Paraules lem.	100	7	9	log2	9	False	0.806
Paraules orig.	90	7	9	log2	10	False	0.789

Taula 9: Grid Search Random Forest

Finalment, amb aquests hiperparàmetres i en les dades d'entrenament s'aconsegueix l'accuracy següent:

Test	
Tèc. Prep.	Accuracy
Lesk syn.	0.813
Synsets freq.	0.782
Paraules lem.	0.828
Paraules orig.	0.805

Taula 10: Test Random Forest

4.4 XG Boost

4.4.1 Motivació

XGBoost és un algorisme d'aprenentatge automàtic que utilitza arbres de decisió per a la regressió i la classificació. Es basa en un gradient boosting framework, que és una tècnica d'aprenentatge per a construir models predictius en que s'utilitzen models dèbils, com ara arbres de decisió, i es van millorant iterativament.

XGBoost és conegut per ser eficient en el temps d'entrenament i per obtenir un bon rendiment en moltes tasques d'aprenentatge automàtic.

4.4.2 Discussió dels hiperparàmetres disponibles i els valors usats

La classe xgboost.XGBClassifier de la llibreria XGBoost té diversos hiperparàmetres que es poden ajustar per controlar el comportament del model. Aquí en destaquem els més importants:

1. **n_estimators**: aquest paràmetre especifica el nombre d'arbres de decisió a utilitzar en el model.
2. **max_depth**: aquest paràmetre especifica la profunditat màxima dels arbres del model.
3. **learning_rate**: aquest paràmetre controla la taxa d'aprenentatge de l'algorisme.
4. **subsample**: aquest paràmetre especifica la fracció de mostres a utilitzar per a entrenar cada arbre del model.
5. **colsample_bytree**: aquest paràmetre especifica la fracció de funcions a utilitzar per a entrenar cada arbre del model.
6. **gamma**: aquest paràmetre controla la quantitat de reducció de la pèrdua necessària per a considerar una divisió.

Es decideix provar combinacions dels hiperparàmetres descrits anteriorment entre els rangs i conjunt de valors següents per al model XGBoost:

- **n_estimators** = [50, 100, 150, 200]
- **max_depth** = [3, 5, 7, 9]
- **learning_rate** = [0.01, 0.1, 0.3]
- **subsample** = [0.5, 0.7, 0.9]
- **colsample_bytree** = [0.5, 0.7, 0.9]
- **gamma** = [0, 1, 5]

En establir el paràmetre **random_state**, es pot assegurar que es seleccionen les mateixes mostres aleatòries cada vegada que s'executa el codi, cosa que pot ser útil per a finalitats de reproductibilitat. La llavor utilitzada per l'experiment és la 23122003.

4.4.3 XG Boost (max_df 0.90, min_df 0.1)

La combinació d'hiperparàmetres que opté un rendiment major en el Random Search de cadascuna de les tècniques de preprocessament proposades és la següent:

Random Search XG Booost							
Tèc. Prep.	subs.	n. est.	max. d.	l. rate	gamma	cool b.	F1
Lesk syn.	0.5	200	7	0.1	1	0.7	0.786
Synsets freq.	0.7	100	3	0.1	1	0.5	0.758
Paraules lem.	0.5	200	7	0.1	1	0.7	0.778
Paraules orig.	0.9	200	5	0.1	1	0.7	0.751

Taula 11: Random Search XG Boost

Dels resultats anteriors s'obté un nou rang d'hiperparàmetres que es prova en el Grid Search:

Grid Search XG Booost							
Tèc. Prep.	subs.	n. est.	max. d.	l. rate	gamma	cool b.	F1
Lesk syn.	0.7	150	5	0.1	1	0.5	0.786
Synsets freq.	0.7	150	5	0.1	1	0.8	0.757
Paraules lem.	0.9	200	5	0.1	1	0.5	0.781
Paraules orig.	0.7	200	3	0.1	1	0.5	0.761

Taula 12: Grid Search XG Boost

Finalment, amb aquests hiperparàmetres i en les dades d'entrenament s'aconsegueix l'accuracy següent:

Test	
Tècnica de Prep.	Accuracy
Lesk syn.	0.773
Synsets freq.	0.803
Paraules lem.	0.805
Paraules orig.	0.750

Taula 13: Test XG Boost

4.4.4 XG Boost (max_df 0.85, min_df 0.05)

La combinació d'hiperparàmetres que opté un rendiment major en el Random Search de cadascuna de les tècniques de preprocessament proposades és la següent:

Random Search XG Booost							
Tèc. Prep.	subs.	n. est.	max. d.	l. rate	gamma	cool b.	F1
Lesk syn.	0.5	150	9	0.1	0	0.9	0.807
Synsets freq.	0.9	200	9	0.1	1	0.7	0.810
Paraules lem.	0.7	200	3	0.1	0	0.5	0.812
Paraules orig.	0.9	200	5	0.1	1	0.7	0.816

Taula 14: Random Search XG Boost

Dels resultats anteriors s'obté un nou rang d'hiperparàmetres que es prova en el Grid Search:

Grid Search XG Booost							
Tèc. Prep.	subs.	n. est.	max. d.	l. rate	gamma	cool b.	F1
Lesk syn.	0.7	150	3	0.1	0.1	0.5	0.818
Synsets freq.	0.7	150	5	0.1	1	0.5	0.818
Paraules lem.	0.9	200	9	0.1	0.1	0.5	0.815
Paraules orig.	0.9	200	9	0.1	1	0.8	0.816

Taula 15: Grid Search XG Boost

Finalment, amb aquests hiperparàmetres i en les dades d'entrenament s'aconsegueix l'accuracy següent:

Test	
Tècnica de Prep.	Accuracy
Lesk syn.	0.838
Synsets freq.	0.789
Paraules lem.	0.842
Paraules orig.	0.798

Taula 16: Test XG Boost

4.5 Conclusió

En aquest projecte, s'implementa un model d'anàlisi de sentiments mitjançant algorismes d'aprenentatge supervisat de la biblioteca Scikit-learn.

Després de provar diferents tècniques de preprocessament i de freqüència de paraules, es troba que la lematització i els paramètres max df i min df establerts a 0.85i 0.05 respectivament, són les millors tècniques juntament amb el model XGBoost, ja que aconsegueix una alta accuracy (0.842).

En general, aquest projecte demostra l'eficàcia de combinar biblioteques NLTK i Scikit-learn per construir un model d'anàlisi de sentiments amb alta precisió.

5 Models d'aprenentatge no supervisat

5.1 Model bàsic

S'ha dissenyat un model, que sense necessitat d'entrenament relaitza la mateixa funció de detecció binària d'opinions. El model utilitza el tokenitzador per paraules de nltk. Aquestes paraules son evaluades per el paquet SentiWordNet, el qual proporciona un *rating* positiu, un de negatiu i un de neutre. La suma dels valors positiu i negatiu de cada paraula son sumades respectivament i el model prediu el sentiment que al llarg de del text ha obtingut un rating total més alt.

5.1.1 Resultat

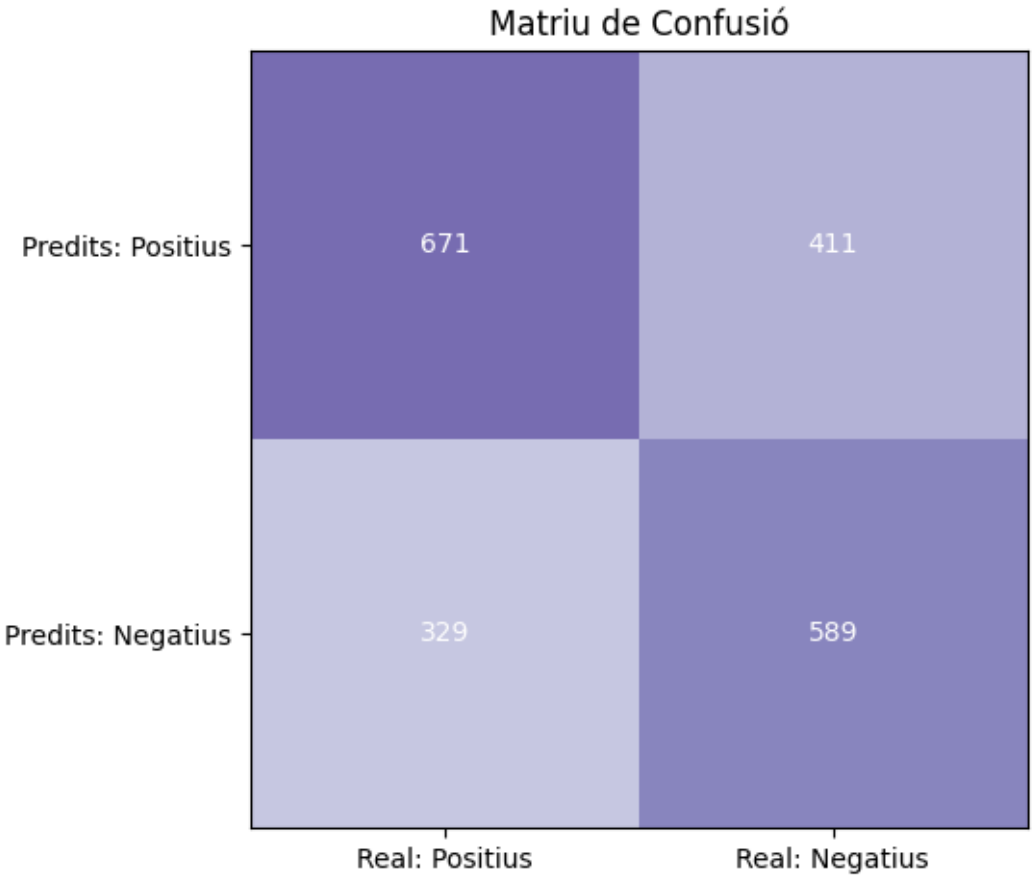

Figura 1: Matriu de confusió del baseline no supervisat.

Els resultats del model son clarament inferiors als del model supervisat. Sembla ser més senzilla per al model l'identificació de textos de sentiment positiu, obtenint un $accuracy = 0.63$ i un $recall = 0.671$.

5.2 Model Ponderat

Per tal de balancejar aquesta situació, s'ha decidit fer ús de pesos per a potenciar l'impacte de les connotacions negatives per a cada paraula. S'executa el mateix model però afegint un pes al *rating* negatiu total al final, de manera que la predicció final ve donada per $max(pos\_rating, weight * neg\_rating)$. Aquest *weight* serà testejat amb valors entre 1 i 1.1 entre intèrvals de 0.005.

5.2.1 Resultat

Test					
Weight	TP	TN	Weight	TP	TN
1	671	589	1.05	596	674
1.005	669	591	1.055	589	686
1.01	661	602	1.06	582	694
1.015	653	616	1.065	569	700
1.02	647	626	1.07	564	707
1.025	638	635	1.075	557	714
1.03	629	644	1.08	555	719
1.035	619	651	1.085	545	726
1.04	618	660	1.09	538	732
1.045	609	671	1.095	532	739
			1.1	525	744

Taula 17: Resultats del model ponderat amb adjectius

Tot i haver esbiaixat el model molt, el millor *accuracy* que s'ha aconseguit ha sigut del 64%, amb $weight = 1.045$ que amb prou forces millora la original. La corva ROC es presenta pràcticament lineal i no hi ha grans canvis en el rendiment del model.

5.3 Model amb adjectius

Com a tècnica per a intentar substraure la informació més rellevant dels textos s'ha proposat la extracció dels adjectius del text per a que sigui avaluat únicament amb aquests. Aquesta es realitza a través de la crida a la funció *adjectius(texto)* L'objectiu d'això és intentar reduir el soroll que pot portar a situacions ambigües.

A més a més, s'ha realitzat el mateix procés de la variable, weight.

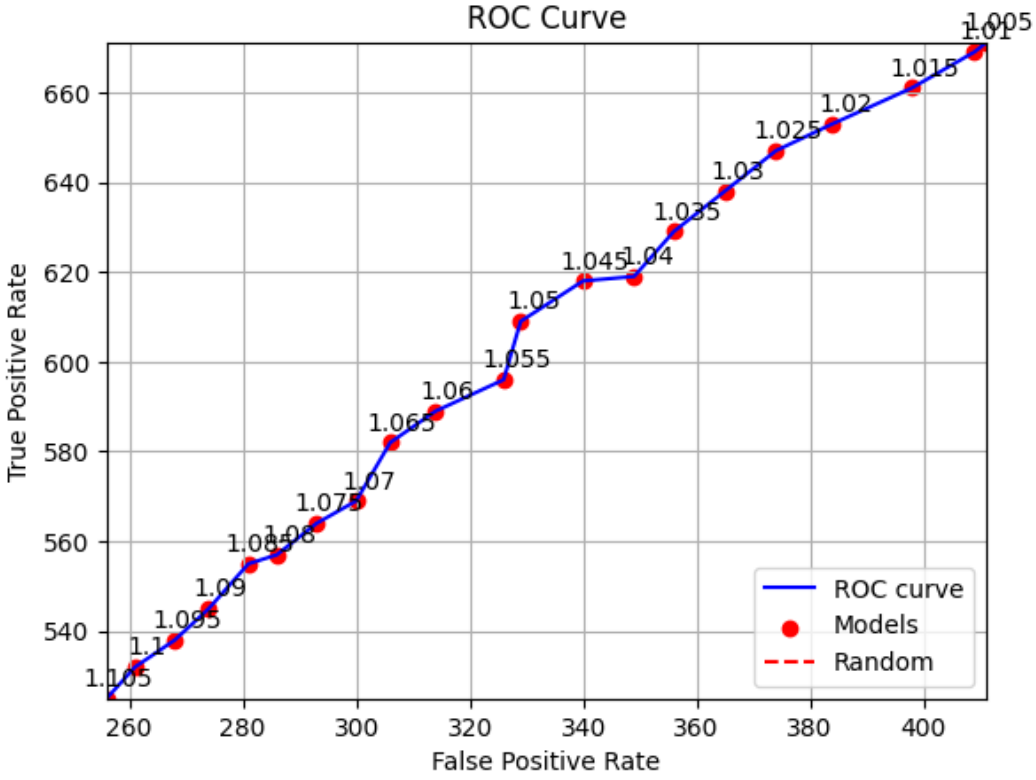

Figura 2: Corva ROC del model ponderat

5.3.1 Resultat

Test					
Weight	TP	TN	Weight	TP	TN
1	667	636	1.05	628	671
1.005	667	636	1.055	627	675
1.01	666	637	1.06	620	680
1.015	662	639	1.065	611	685
1.02	659	644	1.07	607	689
1.025	652	646	1.075	599	692
1.03	645	652	1.08	592	695
1.035	641	655	1.085	587	697
1.04	635	658	1.09	583	700
1.045	632	665	1.095	575	707
			1.1	569	712

Taula 18: Resultats del model ponderat amb adjectius

El resultat amb variacions del valor *weight* no és especialment diferencial però sí que, per norma general, es presenta lleugerament superior aquest model

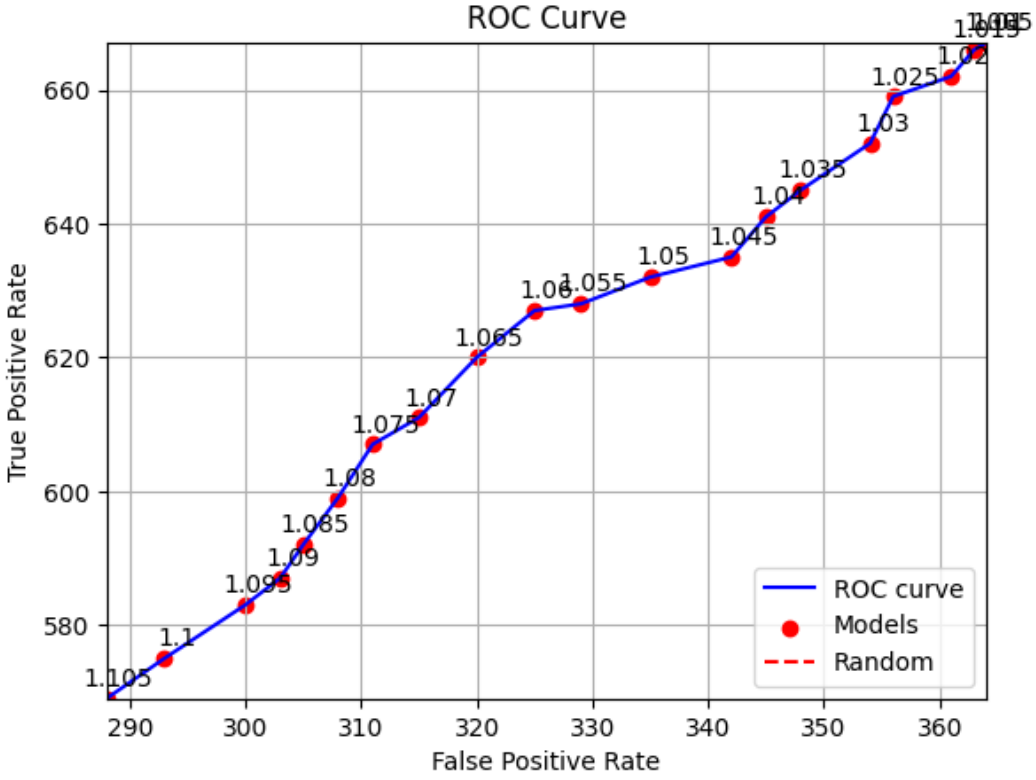

Figura 3: Corva ROC del model amb adjectius ponderat

al anterior. El millor model s'ha trobat amb *weight* = 1, amb un 0.6515 d'accuracy.

5.4 Model no supervisat final i conclusions

Finalment, doncs, ens quedem amb el model tractat sobre els adejctius i sense ponderar, aquest és el seu rendiment:

Accuracy:0.6515, F1 Score:0.667

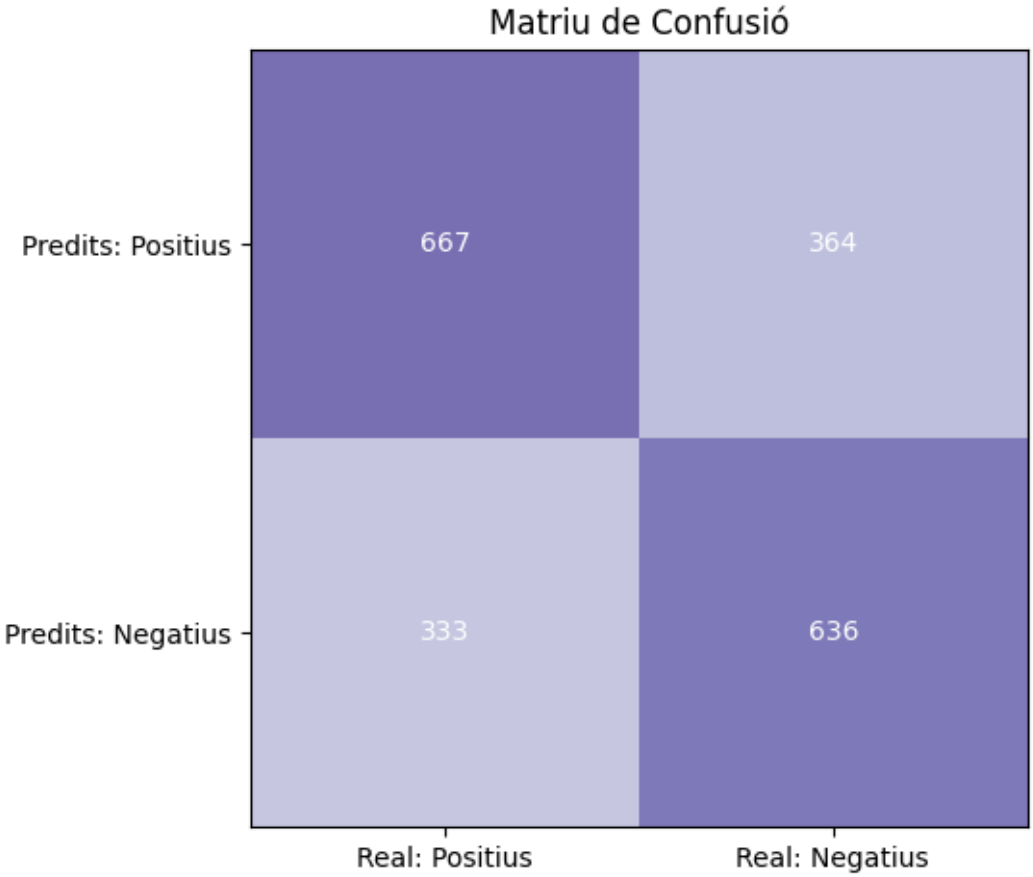

Figura 4: Matriu de confusió del model final no supervisat

És possible que amb algún mètode d'extracció i filtrat d'informació més avançat es pugui millorar el rendiment del model però, fins on hem pogut veure, els models supervisats es mostren clarament superiors als no supervisats. Amb aquests ultims, però, t'estalvies tot el temps i còmput d'entrenament i, a més a més, el procés d'experimentació que hem duut a terme és considerablement menys complex i més ràpid.