CRI Pràctica 3

NAÏVE BAYES

Oriol Moreno **1496663**

Jan Moros **1492333**

Adrià Carrasquilla **1492104**

ÍNDEX

INTRODUCCIÓ

En aquesta pràctica se'ns presenta com a problema la creació d'un filtre on valorarem si un tweet té un esperit positiu o negatiu utilitzant l'aprenentatge bayesià **naïve bayes**.

Partim de una base de dades *Sentiment Analysis Dataset* que conté tots els tweets sobre els que entrenarem la xarxa bayesiana i la validació per comprovar els resultats.

Tenim la pràctica dividida en 3 apartats:

- **Apartat C:** Implementar l'algorisme de naïve bayes amb el mètode de validació juntament amb la mètrica per a fer la valoració. Una vegada implentat es farà un anàlisi sobre els resultats obtinguts.
- **Apartat B:** Consisteix en avaluar el model a base fer modificacions en el tamany del diccionari i del conjunt de train, tant alterant el nombre de tweets llegits com el percentatge de train.
- **Apartat A:** Aplicar el **Laplace smoothing** i tornar a avaluar el model de la mateixa manera que en l'apartat anterior.

Sabent això la nostra tasca és plantejar l'estructura de dades per poder aplicar l'algorisme i un cop construït el model, fer diversos anàlisis variant paràmetres, com s'indica als apartats B i A.

SOLUCIÓ PROPOSADA

Estructura de dades

La base de dades que utilitzarem per a l'entrenament del nostre aprenentatge bayesià es basa en una sèrie de 1578628 tweets. De cadascun d'aquests, tenim 4 columnes d'informació: el seu identificador dins de la base de dades, el contingut del tweet, la data en què va ser publicat i, finalment, el label assignat, que pot ser 0, és a dir emoció negativa o 1, emoció positiva. Aquesta base la carregarem en un **dataframe**.

Aquests tweets s'han passat previant per l'algorisme **Lancaster Stemmer**, que en retalla els prefixos i els sufixos, per deixar només l'arrel de la paraula. D'aquesta manera, moltes paraules que tinguin la mateixa arrel (i, per tant, significats relativament similars) es tindran en compte com una sola paraula a l'hora de calcular probabilitats, obtenint així millors resultats. A part d'això, ens trobem que la base de dades que se'ns aporta està ordenada segons el text del contingut del tweet, i això no ens interessa gens a l'hora de separar entre set d'entrenament i de validació. Hem decidit, doncs, fer un **shuffle** a la base de dades, és a dir distribuir les dades aleatòriament(una sola vegada per així poder fer un anàlisi amb més exactitud), abans de tractar-la per evitar aquests problemes.

A partit d'aquí farem ús d'un diccionari per emmagatzemar la taula d'extracció que veurem a continuació.

Apartat C: Algorisme

Per poder tractar la base de dades al nostre algorisme, primer la carreguem amb pandas, i n'obtenim un **dataframe.** A partir d'aquest, creem un parell de diccionaris on tindrem per una banda un diccionari amb les **paraules positives** i per l'altra un diccionari amb les **paraules negatives**. A partir d'aquests dos, farem la **taula d'extracció**, un diccionari on la clau es cada paraula que ha aparegut al conjunt de train, el contingut d'aquest diccionari conté una tupla amb dues probabilitats, la probabilitat de sortir en un tweet positiu i per altra banda la probabilitat de sortir en un tweet negatiu, aquestes probabilitat es calculen de la seguent manera:

$$p(x_i \mid C_{positiu}) = \frac{nElementsPositius}{nParaulesUniques}$$

$$p(x_i \mid C_{Negatius}) = \frac{nElementsNegatius}{nParaulesUniques}$$

On nElementsPositius i nElementsNegatius són el nombre de vegades que apareix la característica x_i al total de mostres positives i negatives, respectivament.

Una vegada ja tenim aquesta taula ja podem fer els càlculs necessaris per a l'algorisme que desenvoluparem a continuació.

Un cop generada la taula d'extracció, amb la probabilitat de que apareixi cada paraula en un tweet si en coneixem l'emoció, podem fer-la servir per predir l'emoció d'un tweet que no ha aparegut al set d'entrenament.

Per fer això, utilitzarem l'algorisme de Naïve Bayes. Aquest es basa en el teorema de Bayes, àmpliament utilitzat a l'estadística:

$$p(A \mid B) = \frac{p(A)\, p(B \mid A)}{p(B)}$$

D'aquesta manera, coneixent la probabilitat condicionada de B donat A, podem conèixer la d'A donat B.

Per a l'algorisme de Naïve Bayes, ens interessa per calcular la probabilitat de que una mostra sigui d'una classe o d'una altra donada la informació de la mostra.
Donada una mostra $X = (x_1, x_2, x_3, \ldots, x_n)$, con cada x_i representa una qualitat d'X (en el nostre cas, cada paraula d'un tweet), volem trobar la probabilitat de cada valor de la classe objectiu donada la mostra:

$p(C_k | x_1, \ldots, x_n)$ per a tots els k possibles valors de la classe.

Si apliquem aquí el teorema de Bayes, tenim que:

$$p(C_k | X) = \frac{p(C_k)\, p(x \mid C_k)}{p(X)}$$

En aquesta fórmula, el denominador no depèn de C, i com que coneixem tots els valors d'x_i podem assumir que és constant. Per tant, diem que només ens interessa el numerador. D'aquesta manera també ens estalviem que falli el programa si el model no ha vist mai una mostra amb exactament la mateixa combinació de característiques, pel que p(X) seria 0.

El numerador, per la seva banda, és equivalent al model de la probabilitat conjunta, $p(C_k, x_1, \ldots, x_n)$.
Si apliquem la regla de la cadena a aquest model, podem definir-lo com:

$$p(C_k, x_1, \ldots, x_n) = p(x_1 \mid x_2, \ldots, x_n, C_k)\, p(x_2, \ldots, x_n, C_k) =$$

$$= p(x_1 \mid x_2, \ldots, x_n, C_k)\, p(x_2 \mid x_3, \ldots, x_n, C_k)\, p(x_3, \ldots, x_n, C_k) =$$

$$= p(x_1 \mid x_2, \ldots, x_n, C_k)\, p(x_2 \mid x_3, \ldots, x_n, C_k) \ldots p(x_{n-1} \mid x_n, C_k)\, p(x_n \mid C_k)\, p(C_k)$$

Ara ve la part que caracteritza aquest algorisme, i que li dóna el nom de "Naive" (anglès per ingenu): **assumim que totes les característiques d'A són independents.**

Amb aquesta assumpció, podem dir que $p(x_i \mid x_{i+1}, \ldots, x_n, C_k) = p(x_i \mid C_k)$.

Llavors, el model de probabilitat conjunta es pot expressar com:

$$p(C_k, x_1, \ldots, x_n) = p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

Tornant al que hem demostrat prèviament de que p(X) és una constant, podem dir que la probabilitat de C_k donat X és **proporcional** al model de probabilitat

conjunta. Per tant, la fórmula final de la probabilitat condicionada queda de la següent manera:

$$p(C_k|\, x_1, ...,\, x_n) \propto p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

Si apliquem aquesta fórmula per a tots els k valors possibles de C, no podrem determinar la probabilitat concreta de que X sigui d'aquella classe, però com que el resultat és proporcional a aquesta, sí que podrem determinar quina de les probabilitats serà major. D'aquesta informació, en podem extreure una predicció simplement agafant la classe amb el valor més gran.

Gràcies a tota aquesta simplificació matemàtica prèvia a la construcció de l'algorisme, el funcionament d'aquest és molt directe.

La funció que calcula la taula d'extracció també calcula el nombre total de tweets positius i de negatius de les mostres d'entrenament, i a partir d'ells és directe calcular el **prior**, $p(C_{positiu}) = \frac{n\ positius}{n\ positius + n\ negatius}$.

I pel que fa a la resta de la fórmula, $\prod_{i=1}^{n} p(x_i \mid C_k)$ ho podem calcular directament, perquè el contingut de la taula d'extracció conté $p(x_i \mid C_k)$ per a cada possible valor de x_i, és a dir per a cada paraula del diccionari que hem creat.

Finalment, assigna a cada tweet la seva predicció. Per trobar-la, com hem explicat abans, calcula la probabilitat condicionada per a cada valor de la classe (positiu o negatiu), i el que tingui una probabilitat major, l'assigna com a predicció.

Un cop ho tenim tot llest, ja es pot realitzar l'entrenament i la validació del model.

Per a fer l'avaluació de la qualitat del nostre model hem fet servir principalment l'**accuracy** que consisteix en obtenir la proporció de prediccions correctes dividit entre el nombre total d'elements predits, d'aquesta manera s'obté el percentatge

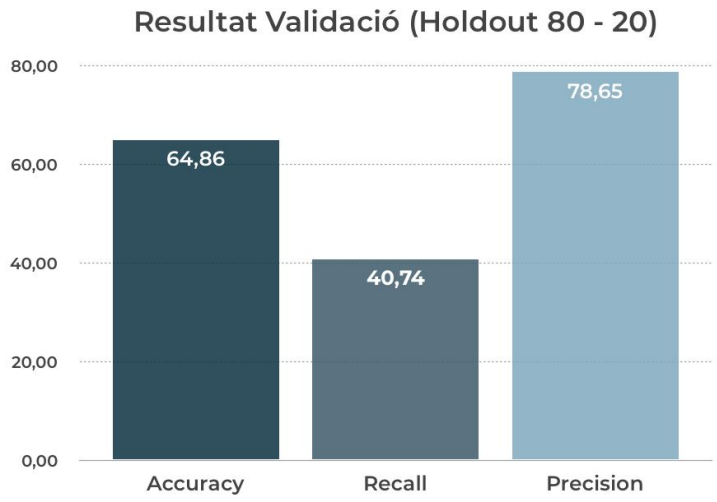
d'encerts. Ens interessa aquesta mètrica ja que el que realment volem és saber el rendiment que tindrà sobre qualsevol tweet que li mostrem.

A més a més per obtenir un millor coneixement sobre la qualitat del model utilitzarem el **recall**, també conegut com a *true positive rate* que comprova si el model classificador ha triat una serie de tweets positius que no hem triat com a negatius alguns tweets que realment eren positius.

Per altra banda també utilitzem com a mètrica la **precisió**, aquesta valora si el classificador té la seguretat de dir que si hem trobat un tweet com a positiu, aquest realment serà positiu.

Donat que hi ha un total de 1,5 M de tweets a la base de dades, es considera que es tenen prou dades com per haver de fer servir tècniques de reaprofitament de informació (cross-validation entre altres) i que usant hold-out és més que suficient per a obtenir bons resultats. S'utilitza un **hold-out** amb un 80% d'entrenament i un 20% de test.

Per tant, en el moment que entrenem el model, aquest rebrà el 80% dels tweets amb la seva respectiva etiqueta per a generar els diccionaris de probabilitats. Un cop es finalitzi aquesta etapa, es predirà per al 20% de tweets restants si és de caire trist o feliç. Al comparar els resultats obtinguts (en funció dels **TP**, **TN**, **FP**, **FN**) s'obtenen diversos valors per a les mètriques. En aquest primer apartat de la pràctica s'obtenen els següents resultats.



Com podem observar la mètrica amb millor resultat és el Precision, seguit del Accuracy i per últim el Recall. Es considera que són resultats acceptables donada la complexitat del problema i la simplicitat del model (estem predint característiques del llenguatge en funció de paraules sense tenir en compte cap dependència entre elles).

Tot i això, és notable el reduït valor del Recall en comparació amb el Precision.

Evidentment, l'accuracy no té un valor suficientment elevat com per considerar el model acurat. Però els resultats obtinguts són notablement millor que si es fes una predicció aleatoria (50-50%).

Per altra banda, la precisió és la mètrica més elevada. Per tant, quan prediu una emoció positiva, podem assegurar amb certa firmesa que realment serà positiu. En canvi el recall és extremadament dolent, pel que estem predint molts tweets positius com a negatius.

Això s'explica si analitzem el funcionament del nostre codi. Com que tenim només dos possibles valors per la classe, trobem el màxim comprovant que un sigui més gran que l'altre. Mirem si la probabilitat de ser positiu és major a la de ser negatiu. Si ho és, assignem a aquell tweet l'etiqueta de positiu, i si no, li assignem la de negatiu.

Això és el desencadenant de la puntuació tan baixa al recall, perquè totes aquelles mostres en què el model calcula que té probabilitat zero de ser tant positiu com negatiu (que ja veurem més endavant que passa sovint), les prediu com a negatives.

Una possible manera d'arreglar-ho seria assignant un valor aleatori a la predicció, però no ho hem arribat a implementar perquè aquest error ja se soluciona aplicant Laplace Smoothing, a l'apartat A.

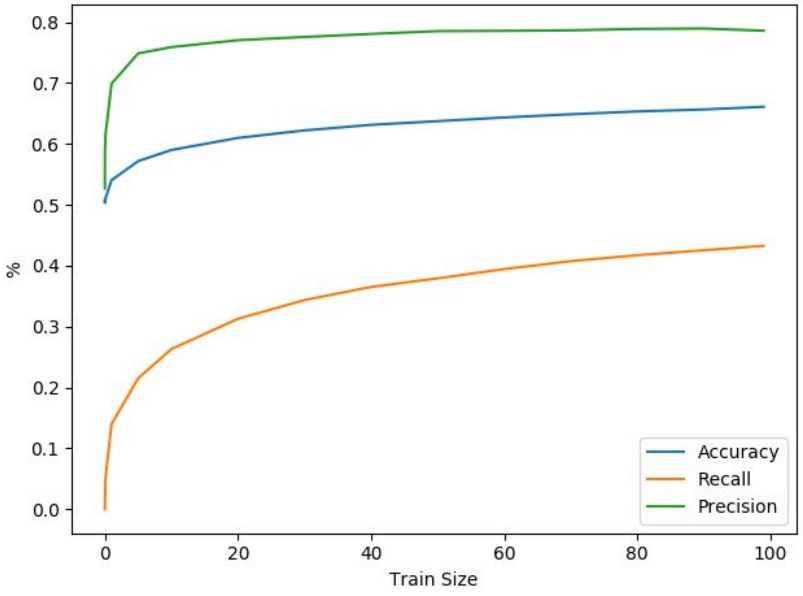
Apartat B: Avaluació

Aquest segon apartat ha consistit en l'anàlisi del comportament del model creat en funció de la variació dels paràmetres que el defineixen. Aquests paràmetres han estat essencialment la proporció del hold-out (mida del train), la mida del diccionari i la mida de les dades llegides a la base de dades. Per a fer cada un dels anàlisis es defineix una funció acord al paràmetre a estudiar.

La primera consisteix en variar la **proporció de dades d'entrenament**. Tenint en compte la gran quantitat de dades s'espera que el resultat serà força similar a tots els casos excepte aquells on s'entreni amb una porció molt reduïda. Les proporcions a provar són les següents (% de dades entrenament respecte el total):

[0.001, 0.01, 0.1, 1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 99]

Aquests són els resultats.



Les nostres expectatives es compleixen. Per a valors molt petits d'entrenament totes les mètriques cauen i a mesura que augmentem les dades, els resultats s'estabilitzen. A més en tot moment les mètriques obtenen valors proporcionals entre elles. Evidentment, en un primer instant els resultats obtinguts són força

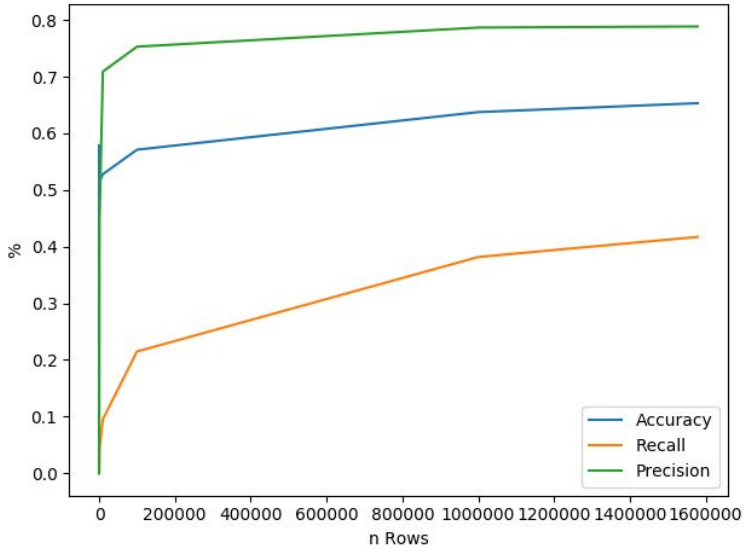
inestables, doncs les dades d'entrenament són tant baixes que li és impossible al model aprendre el suficient com per a realitzar una predicció acceptable.

Realment al voltant del 20% s'estabilitza força, tot i això es decideix optar per un hold-out 80-20% per a la resta de tests (el recall té més marge de millora al augmentar la quantitat d'entrenament).

Aleshores s'avança al següent test: variar el nombre de tweets llegits (com a dades totals). Realment l'efecte que hauria de causar aquest canvi es considera idèntic al del anterior anàlisis, doncs si limitem el nombre de dades a llegir, conseqüentment el training set disminuirà en instàncies i el rendiment empitjorarà. Els valors a provar són:

[10, 100, 1000, 10000, 100000, 1000000, 1578628]

I el resultat obtingut:



La hipòtesi es compleix. Per a poques dades els resultats són inestables i a mesura que s'utilitzen més, a part de millorar, els valors de les mètriques s'estabilitzen. De nou es manté una proporcionalitat entre elles.

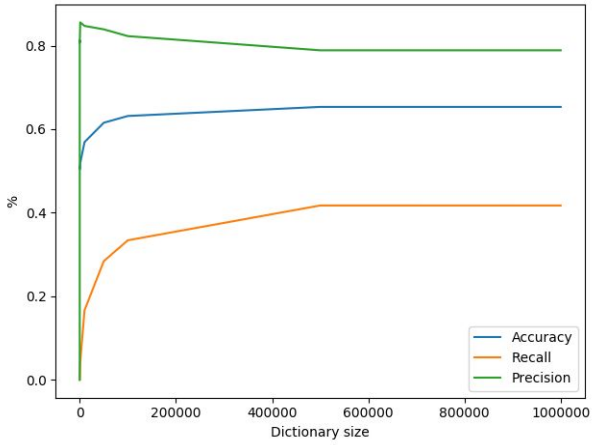
Com a última prova es vol analitzar el rendiment en funció del tamany del diccionari. Tenint en compte que el diccionari es genera en funció de les dades d'entrenament i no pas a partir d'una mida predefinida, per a poder fer aquest test s'haurà d'escurçar aquesta estructura un cop generada.

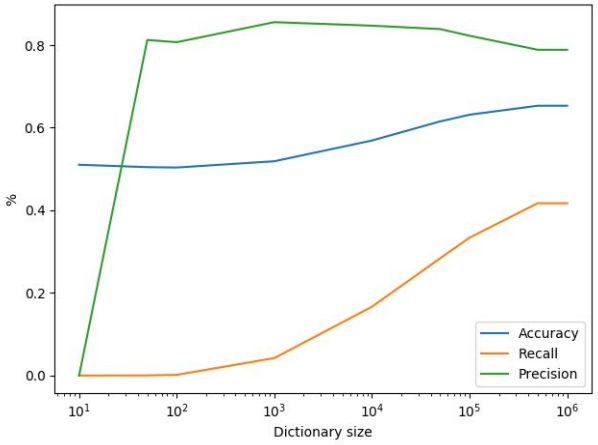
Es considera que per a obtenir millors resultats i amb major estabilitat, per a fitar el diccionari a un nombre determinat de paraules s'ha d'ordenar de major a menor probabilitat. D'aquesta manera el diccionari resultant contindrà les n primeres paraules amb majors probabilitats (que equival a ser les paraules més comunes). Si no s'utilitzés aquest criteri probablement el comportament seria inestable (sobretot a mides petites) doncs potser el formen només paraules amb freqüències molt reduïdes.

Les mides a provar són:

[10, 50, 100, 1000, 10000, 50000, 100000, 500000, 1000000]

De nou, s'espera un resultat molt similar als anteriors casos tot i que el rendiment hauria de millorar abans degut a que tot i ser poques dades (en el diccionari) són més significatives. Això és perquè el training set manté la seva mida inicial utilitzant totes les dades del dataset. Els resultats són els següents:





I per darrer cop, tornem a tenir un resultat similar. En aquest cas hem utilitzat una escala logarítmica per a veure millor la variació en els primers instants. De nou, en els primers instants no és gaire estable donat la poca quantitat de dades que es tenen. A mesura que augmenta la representació, també milloren les mètriques i s'estabilitzen per a valors més i més grans. De fet, la variació a partir d'una mida al voltant dels 5M de paraules és pràcticament nul·la.

A mode resum d'aquest apartat es pot destacar que les proves proposades realment són força equivalents, doncs totes es basen en reduïr o augmentar les dades pel model. En un primer cas es varia el tamany d'entrenament. Evidentment a menors dades, pitjors resultats doncs no tindrà suficients dades per a tenir un mínim criteri. A més a més, si les dades d'entrenament són menys, les de test augmentaran, fet que evidenciarà amb molta més firmesa la invalidesa del model. A mesura que pugui entrenar amb més resultats, el diccionari serà més ampli i podrà ser més precís donades les dades rebudes.

Per altra banda, al modificar el número de tweets llegits té un efecte similar al primer test. Si se'n llegeixen pocs, tot i utilitzar la majoria de les dades per entrenar, aquest training set serà molt petit, fet que equival a entrenar amb una petita porció. Contra més elements de la base de dades es llegeixin, més dades d'entrenament i millors resultats.

Per últim, la reducció dels diccionaris també tindran un efecte similar però amb un detall a destacar. L'entrenament utilitzarà el 80% de les dades i el dataset es llegirà per complet. Posteriorment s'ordenaran les paraules per probabilitat (de major a menor) i es mantindran les n més significatives. D'aquesta manera, tot i utilitzar poques paraules, realment són les que més pes tenen. Si més no, el nombre de paraules desconegudes en diccionaris petits serà molt elevada i acabarà tenint un impacte negatiu igualment. El rendiment, però, millora abans que en les altres modificacions.

Apartat A: Laplace Smoothing

L'últim apartat consisteix en l'aplicació d'una tècnica estadística per millorar el rendiment del nostre model.

Aquesta s'anomena Laplace Smoothing, i com el nom indica el seu objectiu és suavitzar dades categòriques.

Per als classificadors de Naïve Bayes, fa que no descartem una mostra sencera només perquè una de les seves característiques sigui desconeguda. Si ens fixem en les fórmules que hem obtingut a l'apartat de l'algorisme ho entendrem millor:

$$p(C_k|\, x_1, ...,\, x_n) \propto p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

$$p(x_i \mid C_k) = \frac{n\ Elements\ k}{n\ paraules\ \acute{u}niques}$$

El problema sorgeix quan intentem predir un nou tweet, per exemple "ferret ar much cool than fish", i resulta que quan hem fet l'entrenament, cap tweet del set de mostres contenia la paraula "ferret". Si seguim la fórmula, veiem que nElementsPositius i nElementsNegatius de "ferret" seran ambdos zero, i per tant la probabilitat condicionada de que el tweet contingui "ferret" i sigui de qualsevol de les classes serà zero també.

Fins aquí tot té sentit, ja que segons les dades que hem vist no hi ha cap tweet existent que contingui la paraula. El problema arriba quan calculem la predicció sobre aquell tweet, perquè al multiplicar totes les probabilitats condicionades, ens donarà que la probabilitat tant de ser positiu com de ser negatiu d'aquest tweet és zero. I realment tenim molta més informació sobre ell, perquè les altres paraules sí que han aparegut anteriorment a les dades.

Per arreglar això, s'afegeix el paràmetre lambda a les fórmules:

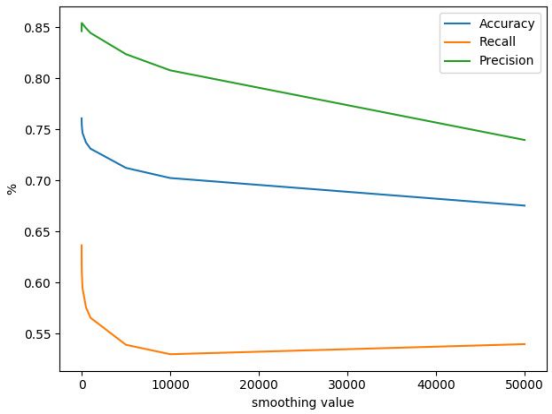
$$p(C_{positiu}) = \frac{n\ positius + \lambda}{n\ positius + n\ negatius + K\lambda}$$

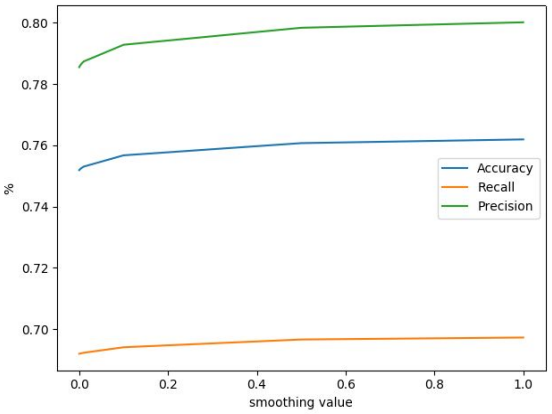
$$p(x_i \mid C_k) = \frac{n\ Elements\ k + \lambda}{n\ paraules\ úniques + \lambda}$$

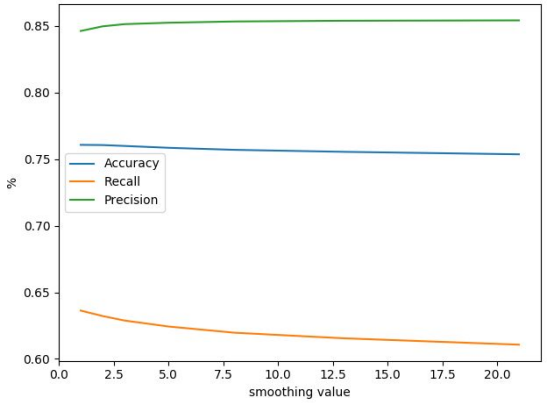
D'aquesta manera, si tornem a l'exemple del "ferret", la seva probabilitat condicionada passa de ser 0 a ser $\frac{\lambda}{n\ paraules\ úniques + \lambda}$. Així seguirà tenint un valor molt inferior a si aparegués realment, però sense causar que la probabilitat condicionada de la que treiem la predicció sigui 0, i ara sí que tenim en compte la resta de paraules del tweet, que ens poden aportar molta més informació.

Aquest canvi soluciona els problemes que teníem a l'apartat C, i ho podem veure molt gràficament perquè pujen totes les mètriques, i la més afectada, el recall, és on més es nota la millora.

Sabem que el valor que s'assigna usualment al paràmetre lambda és 1, però de totes maneres hem provat diversos valors per veure'n la diferència.

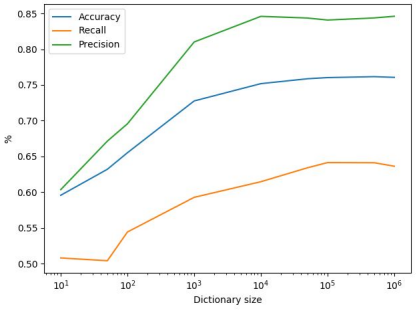


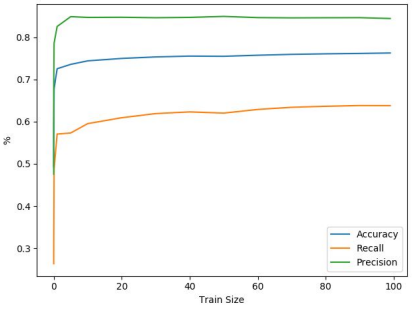


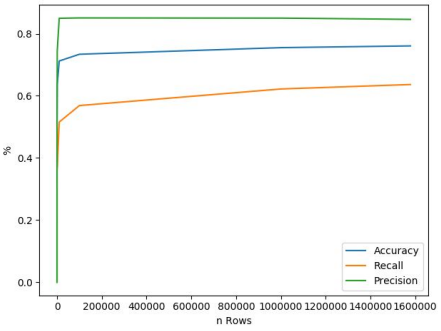


Efectivament, podem veure com totes les proves ens afirmen que el millor valor per a lambda és 1.
Per valors superiors a 1, atribuïm l'empitjorament a que s'atribueix massa importància als valors que el model no ha vist a l'entrenament, i això s'allunya de la realitat de les dades.
Per valors inferiors a 1, hi ha menys diferència, però a mesura que ens apropem al zero també empitjora perquè ens apropem cada cop més a no aplicar l'Smoothing, disminuint el pes de tota la resta de paraules dels tweets.

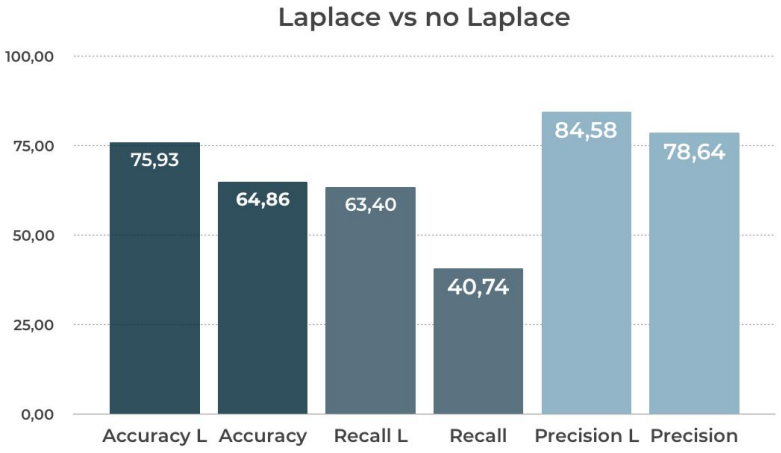
Per finalitzar aquest apartat ens disposem a realitzar els mateixos tests que en l'anterior apartat per comprovar les millores.







Si contrastem aquests resultats amb els del anterior apartat, realment segueixen la mateixa forma. Tot i això ara s'arriben a valors majors per a cada mètrica. Anem a comparar els valors obtinguts en l'execució principal (amb tot el dataset i amb un hold-out 80-20) per al model amb i sense laplace smoothing.

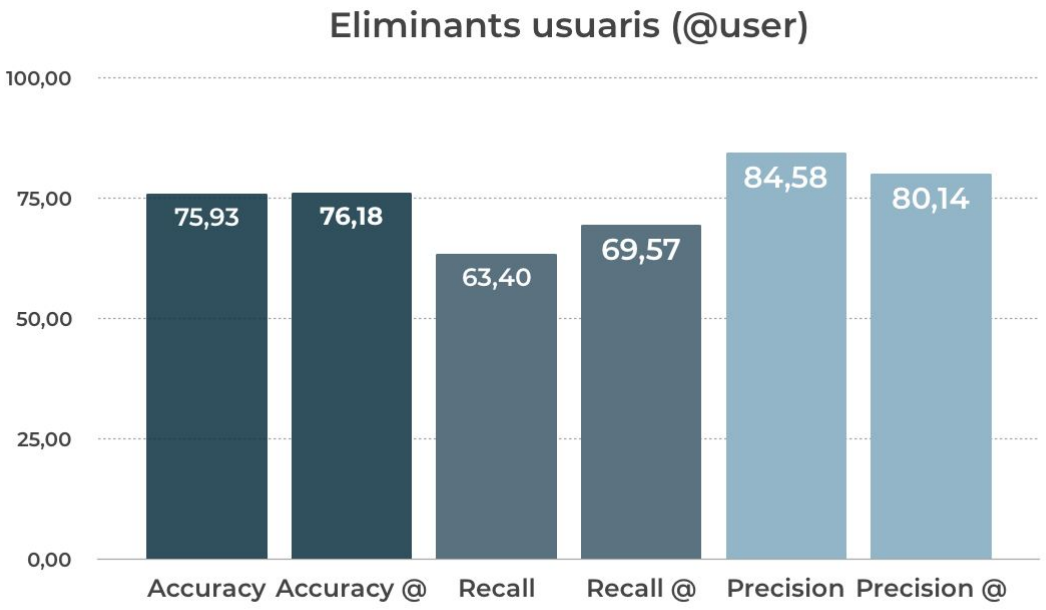


Com es pot observar s'ha obtingut un 11% de millora en l'accuracy, un 23% en el Recall i un 6% en la Precision. Per tant s'ha obtingut una millora molt significativa en totes les mètriques, fet que reafirma la gran utilitat d'aquest procediment.

Possibles millores

En aquest apartat hem plantejat una serie de possibles millores sobre el tractament de les paraules amb l'objectiu de millorar el rendiment del model.
Hem decidit provar les següents idees:

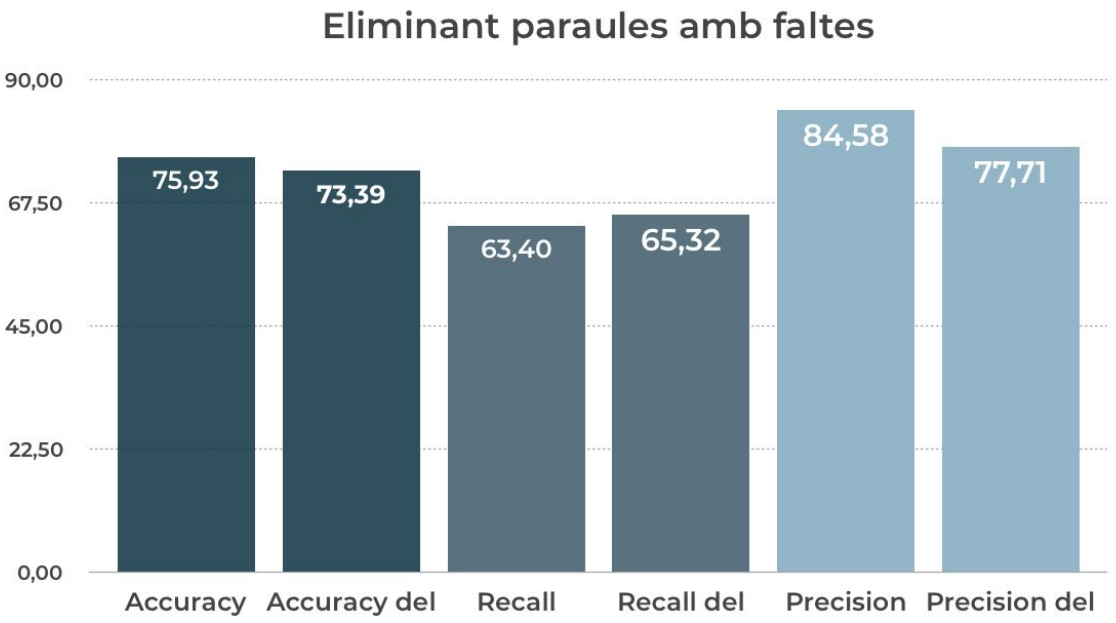
- Eliminar els noms d'usuari dins dels tweets. Es pensa que el nom d'usuari no té perquè tenir influència en el sentiment d'un tweet.



Com podem veure estavem en el contrari, es pensa que això es pot deure al fet que hi ha usuaris els quals poden compartir més pensament d'un tipus que el d'un altre, per tant el nom d'usuari ens pot ajudar a la predicció.

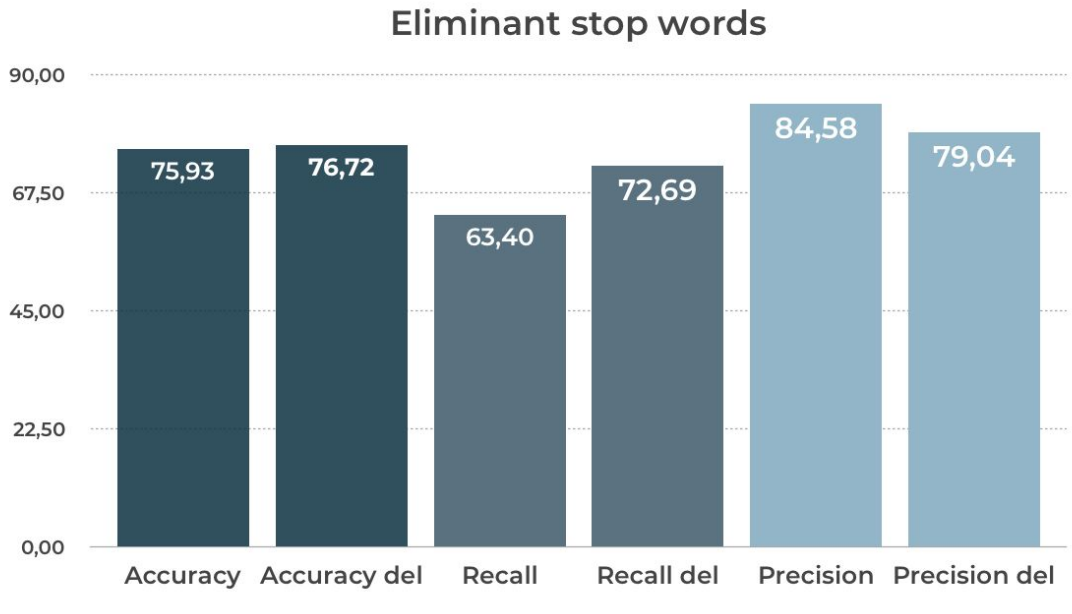
- Corregir aquelles paraules amb faltes d'ortografia. Es planteja que corregir les paraules pot donar millors resultats. Ens donem conta però que els algorismes per corregir faltes d'ortografia són massa costosos i per tant

requereixen de molt temps (>5 segons per 1000 paraules) per tant abandonem aquesta idea i provem de omitir les paraules amb errors ortogràfics.



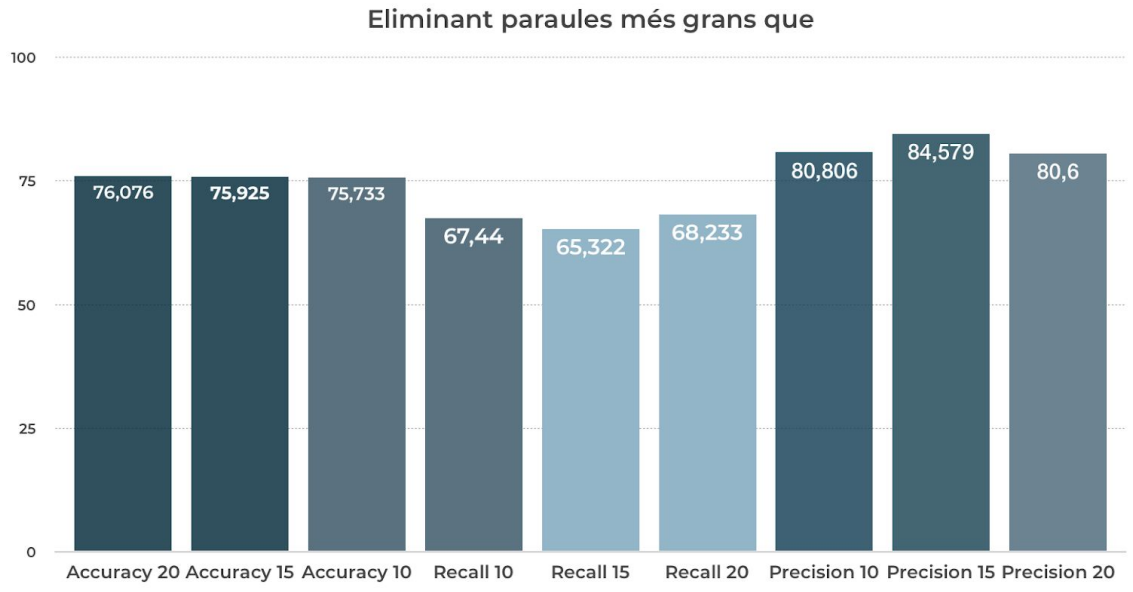
Observem però que això disminueix el rendiment general del model, potser al eliminar paraules mal escrites perdem informació útil per a les prediccions.

- Una altre possible millora pot ser la eliminació de les anomenades *stop words* que son aquelles paraules que no aporten informació semàntica al contingut de les frases de manera que al eliminar-les ens podria donar millors resultats.



Confirmem el que havíem mencionat anteriorment, provoca una lleugera millora en el rendiment.

- Eliminar paraules amb una longitud determinada, hem pensat que eliminar aquestes paraules pot portar a una millora, veiem els resultats:



No es confirma el que havíem predit ja que no millora respecte els resultats anteriors, els empitjora lleugerament això es pot deure a varies causes, una possible hipòtesi podria ser que les paraules molt llargues poden representar elements de spam el que potser classificariem aquestes paraules com a negatives.

Com a conclusió d'aquest apartat veiem que l'únic canvi que ens ha resultat en millora ha sigut el **eliminar els stop words.**

PROBLEMES

Pel que fa a aquesta pràctica ens hem topat amb menys problemes que en les anteriors el desenvolupament d'aquesta ha estat més fluït, tot i així hem tingut alguns petits problemes que ja han estat solucionats:

- A l'hora de predir si el tweet pertany a una classe o una altre no teniem en compte la probabilitat total de pertànyer a la classe positiva o a la classe negativa (probabilitat a priori), el que ens causa una disminució del rendiment del model.

- El Laplace smoothing en primer lloc només sumava el valor de smoothing en les paraules no existents en la taula d'extracció, però no ho fèiem per totes les paraules del diccionari que ja teniem, el que volia dir que una paraula amb 0 aparicions en la fase d'entrenament tenia el mateix pes que una paraula amb una aparició.

CONCLUSIONS I TREBALLS FUTURS

Abans de fer les conclusions sobre els resultats de la pràctica, farem menció al fet de que en aquesta pràctica hem millorat molt la dinàmica de grup i hem sapigut dimensionar millor la càrrega de treball, a més a més ens hem vist més preparats degut a la experiencia obtinguda en les pràctiques anteriors, ja que abans de fer aquestes pràctiques mai començàvem desde 0, sinó que sempre partíem d'un esquelet.

Ara sí comencem amb les conclusions dels resultats, com hem vist anteriorment l'ús de naïve bayes no és el més adequat per aquest tipus de problema, però la simplicitat d'ús i implementació lleugera fa que pugui valer la pena en algunes circumstàncies on potser no ens interesa obtenir tanta fiabilitat. Els resultats obtinguts no són els millors en cas de que es demani un model d'alt rendiment, diguem en un mercat competitiu, però donada la complexitat que comporta fer un anàlisi del llengüatge humà es consideren uns resultats òptims.

Per altre banda es destaca la natura del dataset: és molt important tractar-lo per '*estandaritzar*' les dades. Tenint en compte que són comentaris d'internet, el vocabulari és molt complex i és molt comú fer moltes variacions de la mateixa paraula. En el cas dels diccionaris convencionals, el nombre de paraules diferents és molt elevat, però a internet aquest valor creix moltíssim ja que s'accepten faltes d'ortografia i modificacions úniques a criteri de l'usuari. Això implica la necessitat d'analitzar moltíssimes més dades o buscar la manera d'unificar variacions per significat.

És per això que en aquest problema el Laplace smoothing té un paper força important per a augmentar el rendiment del model creat, doncs és l'encarregat de tractar aquells casos no vistos en l'entrenament.