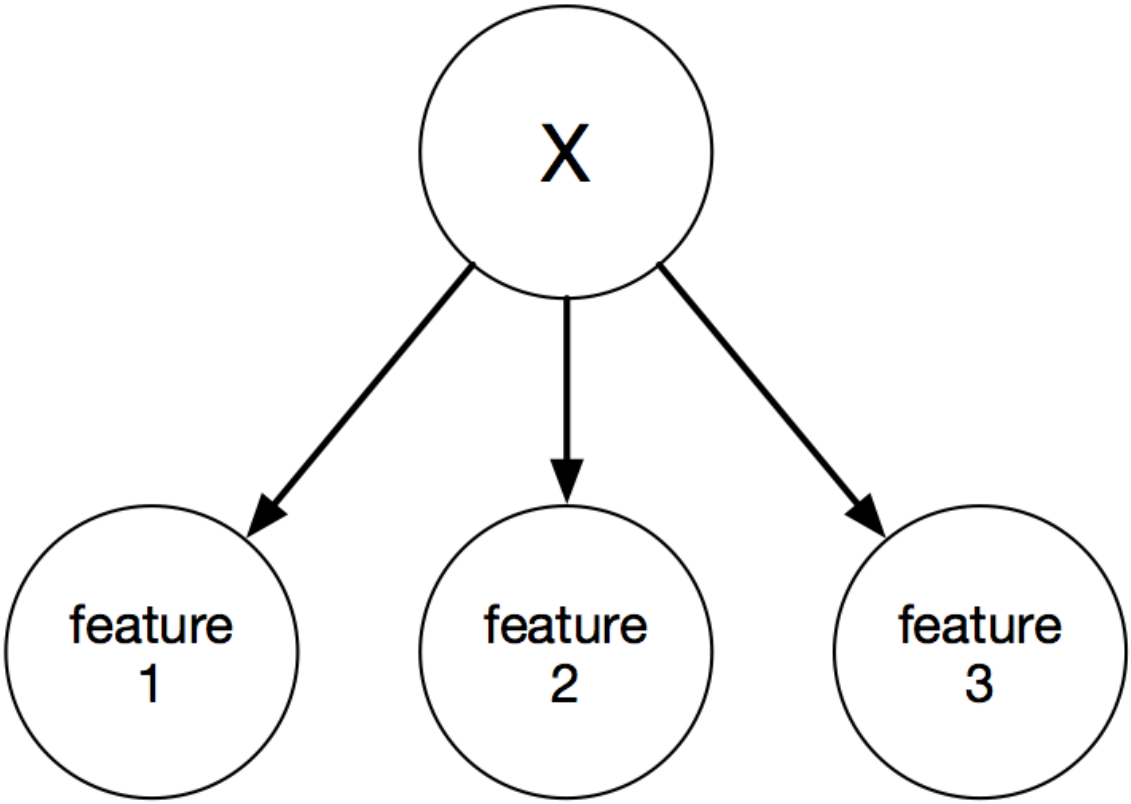
APRENENTATGE BAYESIÁ

Practica 3



Eric Cañas Tarrasón - 1423177
Bernat Martínez Gómez- 1425968
Kevin Martín Fernández - 1423234

Índex

Introducció

El problema de l'aprenentatge bayesià

El problema que se'ns ha plantejat en aquesta pràctica es el següent: Crear des de 0 un software que sigui capaç de llegir una base de dades (en format CSV) y elaborar a partir d'aquestes un model bayesià (Naïve Bayes) que sigui capaç de per a noves entrades no vistes donar una resposta coherent, entenent com a resposta el valor d'una variable objectiu.

De manera més específica, es tracta d'un problema d'aprenentatge supervisat per fer classificació, on es rep una base de dades amb 1.578.627 exemples de tweets reals, on cada exemple conté: un identificador numèric, el text del tweet, la data en que es va escriure, i una etiqueta que expressa si el tweet es negatiu (0) o positiu (1). A partir d'aquestes dades s'ha de ser capaç de predir per a un nou tweet no vist pel classificador, si aquest es positiu o negatiu.

Per a la creació del software que implementes això es donaven un sèrie de restriccions i requeriments: El model elaborat havia de seguir les regles de les xarxes bayesianes, preferiblement una xarxa de tipus Naïve Bayes. A més els resultats havien de poder ser validats utilitzant diferents estratègies de validació (Cross-validation, LeaveOneOut, Kfold...). Y amb diferents mètriques, per d'aquesta manera poder observar el comportament de la classificació de la manera més realista possible.

Els diferents diccionaris que formessin l'estructura interna del classificadors haurien de ser truncables (mitjançant un criteri coherent) per així poder observar com afecta el tamany d'aquests a la classificació del problema.

Els exemples continguts dins del conjunts d'entrenament i validació utilitzats haurien de contenir una distribució de la variable objectiu aproximadament igual a la distribució que aquesta tingues en la base de dades original.

Per últim s'hauria d'aplicar a l'algoritme del classificador un mètode "Laplace Smoothing" per veure com aquest complement afecta a la predicció.

Respecte a la resta del projecte, la presa de decisions era lliure. Es podia utilitzar qualsevol llenguatge de programació, i l'única regla ètica era no utilitzar cap llibreria externa o interna del llenguatge centrada en la resolució d'aquest tipus de problemes.

Decisions de grup: el projecte

Per començar a aixecar aquest projecte s'han hagut de prendre varies decisions. La primera d'elles sobre la organització que s'havia de tindre. Per aquest tema es va prendre la decisió de elaborar un repositori compartit gracies a GitHub que en permetés a tots treballar fàcil i àgilment de manera remota.

La següent i ja si molt important decisió que es pren es: En quin llenguatge desenvolupar el projecte?

En aquest tema s'havien d'avaluar pros i contres de cadascun dels llenguatges que semblaven una bona idea: C++, Java i Python.

C++ va ser descartat molt d'hora ja que tot i ser el que permetia un codi mes òptim, tenia una baixa flexibilitat i era poc àgil. I entre Java i Python, es va decidir finalment utilitzar Python per un motiu: Numpy i Pandas. Aquestes llibreries ens proporcionaven una sèrie d'eines molt útils en el nostre projecte, amb funcions per administrar de manera molt simple la lectura del csv, l'esborrament de columnes, els filtres d'accés (com per exemple x[x=='a']), el càlcul de paràmetres com mitges o sumatoris...

A continuació s'havia de pensar en l'estructura de la xarxa bayesiana, de la que es parlarà de manera més extensa en un apartat posterior

I per últim tenint en compte que es tracta amb una base de dades de 128Mb, establir unes regles bàsiques d'accés i tractament amb certes estructures de dades, ja que un accés de forma errònia podia disparar fàcilment la complexitat de l'algoritme eternitzant els temps d'execució

Distribució de classes

En el projecte en Python anexe a aquest informe, es troba el codi font del software generat. Per una bona comprensió d'aquest codi font cal tenir en compte que es poden distingir les següents classes importants

DataTrimmers: Aquesta classe conté funcions bàsiques de divisió de dataFrames, es tracta d'un contenidor de funcions que permet per exemple donat un conjunt de dades, fer una separació en dades de per a Training o Validació, o per exemple donar els diferents folds de separació en cas de estratègies de tipus K-Fold.

BayesianNetwork: Aquesta classe representa el model, i conté per tant totes les funcions necessàries per construir la seva representació i per donar els resultats de les prediccions. A més d'això conte les diferents funcions encarregades de fer validacions.

Validators: Conte funcions que creen un model o varis models i els avaluen en funció de diferents estratègies de validació.

Error Measuraments: Conté una sèrie de funcions que a partir d'una matriu de confusió extreu el valor de cadascuna de les possibles mètriques d'error.

Main: Classe principal del projecte, que permet fer la construcció dels models i mostra els resultats. Conté funcions que fan d'accés ràpid per representar diferents solucions i resultats

Apartat C: La xarxa Bayesiana, Naïve Bayes.

Estructura de la xarxa:

Per resoldre aquest problema, el primer objectiu ha sigut dissenyar una estructura correcta per a la xarxa Bayesiana a utilitzar. Per això era important primer observar les diferents variables de que es disposaven per així observar quines introduïen soroll i quines eren rellevants per a la predicció. A primera vista es observable que la variable "identificador", es completament sorollosa, ja que una variable amb les propietats d'una "Primary Key" no aporta cap tipus

d'informació al problema. Llavors en primera instancia es podia plantejar una xarxa bayesiana completa amb la següent estructura:

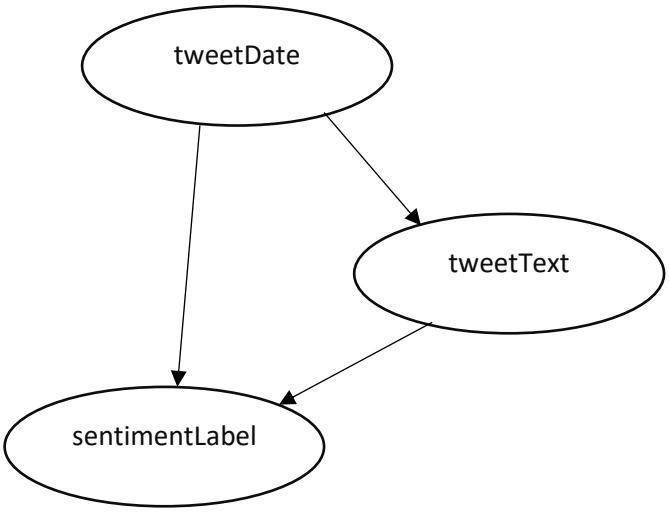


Figura 1: Primer disseny de Xarxa Bayesiana

En un disseny com el de la figura 1 es té en compte que el tweet es veu influenciat per la data (ja que per exemple el text de un tweet pot ser mes propens a contenir paraules de compassió o tristes en dies negres com els dies d'un atemptat, o paraules més festives i alegres en dies de festes nadalenques, per exemple. Mentre que seguint aquest mateix criteri, la classe (positiva o negativa) pot veure's influenciada per la data del tweet i especialment per el text d'aquest. Amb la mateixa analogia anterior, un tweet de tipus "se m'han caigut les llàgrimes" pot tenir una label associada diferent si es escrit el 11 de setembre de 2001 (dia de la caiguda de les torres bessones) o un 17 de juny (dia del pare) on aquestes mateixes paraules poden ser mes propenses a tenir un significat emotiu per les habituals mostres d'afecte entre pares i fills en aquesta data.

Aquest disseny que a priori pot semblar el més fidel a les dades donades, té però varis problemes associats, i es que ens ofereix un efecte de visió dinàmica en un model que es a priori estàtic. Ja que podrem veure a l'analitzar la validació que donaria uns resultats que no podríem prendre com a resultats reals per a la seva futura utilització, ja que en l'aplicació practica i real, el mes habitual serà que ens trobem tweets actuals per als quals no tinguem cap registre associat a la data. I per tant no podríem assegurar l'index d'error obtingut durant la validació on si ens trobàvem amb que el cas més habitual era el de si conèixer aquesta data.

Per aquest motiu el següent pas en l'elaboració de la xarxa es el de considerar aquesta data com a sorollosa per al cas pràctic d'us que tindria aquest model. I per tant no tenir-la en compte. Generant un nou disseny de Xarxa Bayesiana Naïve com el mostrar en la figura 2.

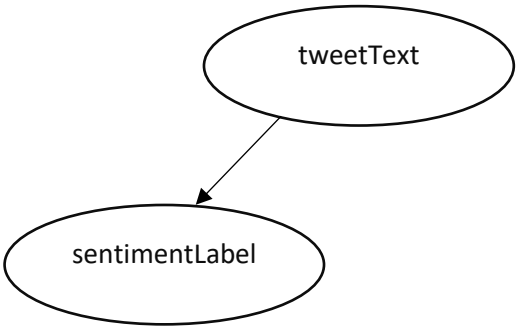


Figura 2: Disseny de Xarxa Bayesiana Naïve utilitzat

El model de la figura 2, ja es tracta d'un model molt més lleuger, amb dos variables. Sentiment label, que serà binaria, i tweetText que no prendrà el valor complet del text del tweet (ja que això introduiria un espai de valors possibles d'aquesta variable on pràcticament hauria una sola repetició per valor en la majoria de casos i la probabilitat de trobar un cas no vist seria molt elevada) si no que farà una divisió en paraules, de manera que la P(tweetText="I have a Mewtwo at level 100"|X) serà mirada realment com la serà mirada realment com (P(tweetText="I"|X) + P(tweetText= "have"|X)...). Això permetrà reduir l'espai de la variable de manera que sigui molt més complicat trobar casos de probabilitat 0 (o èpsilon) a causa de no haver vist la variable durant la creació del model.

Predicció de la xarxa i generació dels diccionaris

Per a realitzar prediccions la xarxa utilitza la següent formula probabilística

$$\begin{aligned}&\boldsymbol{P}(\boldsymbol{sentimentLabel} = \boldsymbol{X_i}|\ \boldsymbol{tweetText} = \text{"}\boldsymbol{My\ favorite\ pokemon\ is\ Magikarp}\text{"})\\ &\quad = \log(\mathrm{P}(\mathrm{word} = \text{"My"}|\mathrm{sentLab} = X_i))\\ &\quad + \log\big(\mathrm{P}(\mathrm{word} = \text{"favorite"}|\mathrm{sentLab} = X_i)\big) + \cdots + \log(P(sentLab = X_i))\end{aligned}$$

On $\boldsymbol{P}(\boldsymbol{word} = \boldsymbol{A}|\boldsymbol{sentLab} = \boldsymbol{X_i}) = \max(\frac{Ocurrences\ Of\ Word\ \mathrm{A}\ in\ words\ observed\ on\ label\ X_i}{Total\ words\ observed\ on\ label\ X_i}, \varepsilon)$

I $\boldsymbol{P}(\boldsymbol{sentLab} = \boldsymbol{X_i}) = \frac{Words\ observed\ of\ label\ X_i}{Total\ words\ observed}$

D'aquesta manera la classificació final donada per la xarxa serà de manera general:

$$\begin{aligned}&\boldsymbol{sentLabPredicted}\\ &= indexOf(\max(\mathrm{P}(\mathrm{sentLab} = X_i|\mathrm{tweetText} = \mathrm{A}), \mathrm{P}(\mathrm{sentLab} = X_{i+1}|\mathrm{tweetText} = \mathrm{A}) \ldots,\\ &\qquad \mathrm{P}(\mathrm{sentLab} = X_n|\mathrm{tweetText} = \mathrm{A})))\end{aligned}$$

On $\boldsymbol{indexOf}\big(\mathbf{P}(\mathbf{sentLab} = \boldsymbol{X_i}|\mathbf{tweetText} = \mathbf{A})\big) = i$

En aquest cas específic, com s'està tractant amb una base de dades on sentimentLabel es un atribut binari (0 o 1), X només podrà prendre aquests 2 valors.

Es molt important notar, que quan es parla de $\mathrm{P}(\mathrm{sentLab} = X_i|\mathrm{tweetText} = \mathrm{A})$, com s'estan aplicant en el seu càlcul logaritmes (per evitar l'underflow d'un multiplicatori de números menors que 1), el resultat no serà estrictament una probabilitat, ja que no mantindrà totes les propietats d'aquesta. Però com només estem buscant el valor de X_i que maximitza la funció, i els logaritmes compleixen que $log_n(X) > log_n(Y)$ ➔ X>Y el fet d'aplicar logaritmes no modificarà la sortida de la funció i soluciona el problema computacional de l'underflow.

Observat tot això llavors està clar que la informació mínima que s'haurà de guardar a memòria seran com a mínim dos diccionaris (o un diccionari de doble clau) (un per cada valor possible de la variable sentimentLabel) on per a cada key (que serà una paraula concreta) s'emmagatzemarà el valor de "Ocurrences Of Word 'Key' observed on Label X_i", es a dir el nombre de vegades que a aparegut aquesta paraula en el conjunt de tweets associats a una sentimentLabel concreta. Aquest diccionaris tal i com el seu significat indica, es generaran de mitjançant el següent algoritme expressat en pseudocodi:

```
GenerarDiccionaris(conjuntDeTraining){
  Per a cada tweet en conjuntDeTraining{
    Per a cada paraula en tweet[text]{
      Si Diccionari[tweet[label]] conte la key paraula{
        Diccionari[tweet[label]][paraula] ++;
      } si no{
        Diccionari[tweet[label]][paraula] = 1;
      }
    }
  }
}
```

Validacions i Resultats obtinguts

Un cop entrenada la xarxa, es necessari observar com de correctament funciona i quin es l'error generat en les prediccions, per poder fer això s'han aplicat tres estratègies diferents de validació que creen diferents conjunts d'entrenament i validació per donar uns resultats d'error amb diferents fiabilitats. Es important que abans d'aplicar qualsevol d'aquests mètodes es realitzi una aleatorització del conjunt inicial de dades, de manera que s'elimini qualsevol ordenació prèvia implícita en el conjunt (en aquest cas estaba aproximadament ordenat per ordre alfabètic

de la variable "tweetText"). Aquesta aleatorització provocarà que per a diferents execucions del procés els resultats d'error siguin lleugerament diferents en els dos primers mètodes de validació (Cross-Validation Basic i K-fold)

Per últim serà important que a l'aplicar aquests mètodes es mantingui la mateixa distribució de la variable objectiu que en el conjunt de dades original. Teòricament això ja serà així al triar n mostres de manera aleatòria, sempre i quan l'algoritme de nombres aleatoris utilitzat sigui equiprobable en tots els seus valors (especialment en conjunts grans com els que s'estan tractant, tot i així s'han aplicat solucions per assegurar-ho amb una fiabilitat completa

Les estratègies de validació utilitzades han sigut doncs les següents:

La primera: **Cross-validation Basic**. En aquest mètode de validació s'ha utilitzat un percentatge fix prèviament extret de la base de dades com a conjunt de validació. Concretament s'ha decidit establir el llindar en 2/3 de les dades per entrenar la xarxa (1.052.418 exemples), i 1/3 d'aquestes per validar-la (526.209 exemples)

La segona: **K-fold**. En aquest mètode de validació, s'ha entrenat no tan sols un model si no K models. De manera que les dades s'han dividit en K seccions diferents, i en cada una de K iteracions, s'han utilitzat una d'aquestes seccions com a Validation i tota la resta com a Training. Finalment l'error real serà la mitja dels errors produïts per cada model. Concretament s'ha decidit establir una K de 100, de mode que s'han generat K parelles de conjunts de Validació amb 15.786 exemples i 1.562.841 exemples de Training.

La tercera: **LeaveOneOut**. En aquest últim mètode de validació s'ha utilitzat la mateixa estratègia utilitzada amb k-fold però utilitzant una K igual al nombre de dades que es disposen. De manera que s'han generat n parells on només hi ha un exemple de Validation i n-1 (1.578.626 en el nostre cas) exemples sobre els quals fer training, i un cop més s'ha calculat l'error real com la mitja dels errors dels 1.578.627 models generats. Per poder realitzar aquesta última validació en un temps inferior a 206,983 dies, s'ha hagut de modificar l'estructura habitual de la recomposició de diccionaris de manera que es pogués reconvertir un model en el següent amb una mínima inversió de recursos.

Per últim es important observar que la proporció inicial de exemples de cada classe es de 50,03% (782.652 exemples positius) a 49,97%(781.650 exemples negatius)

A continuació es mostren els resultats obtinguts en cada tipus de validació

Error en Cross-Validation Basic:

Ja que aquest mètode de validació es el mes susceptible a l'atzar, serà aquest el que més ens pot permetre adquirir una idea sobre la estabilitat de la base de dades. Per aquest motiu per efectuar la proba utilitzant Cross-validation Basic amb els paràmetres anteriors, s'ha reproduït la proba en 10 ocasions i els resultats mostrats son una mitja d'aquestes execucions

Basic	Accuracy	Recall class=0	Recall class=1	Precision class=0	Precision class=1
μ	0,74884	0,72316	0,78129	0,80690	0,69071
σ	0,00034	0,00028	0,00044	0,00044	0,00031

Figura 3: Mitges i DesvStd de les mesures d'error més significatives en Cross-Validation

Tal com es pot observar en la figura 3, els resultats de sortida de les 10 execucions de l'algoritme, triant en cadascuna d'elles un 1/3 aleatori i balancejat, de dades per a la validació sobre el

dataFrame total, tenen una desviació estàndard inferior a 10^{-3}, o expressat d'un altre manera, una desviació estàndard en el major dels casos de tan sols el 0,044%. Aquestes desviacions estàndard mostren per tant que gracies a l'elevat numero d'exemples ens trobem davant d'una base de dades bastant estable. Una base de dades així d'estable podria continuar doncs donant bons resultats amb un nombre més reduït d'exemples com es podrà observar en apartats futurs.

Al ser tan estable doncs, es previsible que la resta d'estrategies de validació donin com a sortida valors molt semblants, de manera que deixarem el anàlisis d'aquests resultats, per ser analitzat més endavant amb unes estratègies de validació més fiables que Cross-Validation Basic

Error en K-Fold:

La següent estratègia de validació provada ha sigut K-Fold. Aquesta estratègia pels motius comentats abans es més fiable encara que més lenta de realitzar (7 minuts per K=100) que l'anterior (12 segons per una execució(comptant creació del model)).

Els resultats de les mateixes mesures d'error anteriorment analitzades ara amb una estratègia K-Fold amb K=100 es la següent:

Accuracy	Recall class=0	Recall class=1	Precision class=0	Precision class=1
0.74432	0.71985	0.77501	0.80050	0.68807

Figura 4: Mesures d'error més significatives en 100-Fold

A més per realitzar un anàlisis més visual del resultats de la xarxa pot ser interessant observar la matriu de confusió dels resultats.

	Pred = 0	**Pred = 1**
Class = 0	626.518	243.819
Class = 1	156.133	537.803

Figura 5: Matriu de confusió de la xarxa (100-Fold)

Observant aquests resultats es curiós observar que totes les mesures d'error han donat pitjors resultats que en l'anterior validació, encara que les diferencies son en tots els casos mínimes i mai superiors al 0,7%. Això es curiós per que en aquesta estratègia la xarxa ha sigut entrenada mitjançant diferents conjunts d'entrenament, que en tots els casos contenien un 99% de les dades originals, mentre que en l'anàlisi per Cross-Validation Basic aquesta era entrenada amb tan sols un 66,67% de les dades. Això pot donar intuïcions de que l'augment del volum de dades introduït, ha pogut produir un major soroll en la xarxa Naïve. Un cop més es deixarà el anàlisis explícit dels resultats, per després d'aplicar la següent estratègia de validació ja que aquesta següent donarà uns resultats amb una fiabilitat molt superior als dos anteriors mètodes.

Error en LeaveOneOut:

LeaveOneOut es la mesura de error més fiable, ja que es la de totes tres la que amb un major percentatge de dades treballa. Aquesta estratègia de validació genera N models de manera que

en cadascun d'ells es valida sobre tan sols un exemple. Això en molts casos provoca que tot i ser la mesura més fiable es torni també impracticable ja que el temps de generar els N models es pot eternitzar. Per sort la naturalesa de les xarxes bayesianes Naïve permet poder fer la transformació d'un model a un altre de manera molt ràpida, permetent per tant aplicar una validació de tipus LeaveOneOut en un temps aproximat de 2 minuts.

A continuació es mostren els resultats de les mesures d'error preses utilitzant aquest tipus de validació:

Accuracy	Recall class=0	Recall class=1	Precision class=0	Precision class=1
0.74436	0.71996	0.77495	0.80038	0.68827

Figura 6: Mesures d'error més significatives en LeaveOneOut

A més d'aquestes un cop mes es mostra la matriu de confusió que ara si serà més fiable que l'anteriorment mostrada.

	Pred = 0	**Pred = 1**
Class = 0	626.416	243.657
Class = 1	156.236	537.993

Figura 7: Matriu de confusió de la xarxa LeaveOneOut

En aquestes noves dades preses es pot veure com la matriu de confusió tan sols varia en un màxim de dos centenes en el major dels casos. I les mesures d'error son pràcticament idèntiques a les vistes anteriorment en K-Fold. Això indica doncs que tal com es preveia abans, la base de dades utilitzada conté un nombre enorme de dades, i podria obtenir-ne resultats pràcticament idèntics amb un menor nombre de dades. Tot i així les dades d'entrenament sempre son un incentiu positiu.

Entrant de manera més profunda en les mesures analitzades es poden observar tres mesures diferents que donen diferent informació sobre l'eficàcia de la xarxa:

Accuracy → $\frac{Encerts}{Prediccions\ Totals}$**:** Aquesta mesura dona una visió sistemàtica del percentatge d'encerts de la base de dades (o d'errors observant el complementari (1-accuracy)). Es doncs la mesura que dona una visió més general de com s'està comportant el model. Tal i com es pot veure en aquesta xarxa s'arriba a aconseguir un **74,44%** d'accuracy, un valor que permet donar resultats amb un 25,55% de probabilitat genèrica d'errar. Aquest percentatge mostra que la xarxa es capaç de donar una bona intuïció sobre el sentiment que transmet el tweet, tot i no ser una resposta fiable en àmbits estrictes.

Precision → $\frac{TP}{(TP+FP)}$**:** Aquesta mesura intenta explicar com es de fiable un valor donat tenint en compte la classe a la que pertany. D'una manera més especifica, explica sobre el total de mostres que s'han classificat com de la classe P, quantes realment pertanyien a la classe P. En aquest cas

podem observar doncs que s'aconsegueix una precisió del **80,04%** sobre la classe 0 (tweet negatiu), i un **68,83%** sobre la classe 1 (tweet positiu). Aquesta mesura ens esta mostrant doncs que es un 11,21% més fiable un resultat donat si s'ha predit com a classe 0, que si s'ha predit com a classe 1. Aquesta diferenciació pot ser útil en alguns sistemes on tenir un error de predicció sobre una classe concreta pot tenir unes conseqüències molt pitjors que tenir un error sobre una altre classe. En un cas com aquest on cap de les dues prediccions te conseqüències això es podria o be assumir, o be afegir un petit "threshold" en la classe més perjudicada, per a que en casos de probabilitats molt semblants entre totes dos classes es decidís per l'altre classe. Això podria equiparar la precisió entre totes dues classes. Però es torna una solució innecessària en aquest problema.

Recall → $\frac{TP}{(TP+FN)}$**:** Aquesta mesura intenta explicar sobre el total d'exemples que pertanyien a una classe, quants d'ells han sigut identificats correctament com a pertanyents a aquesta classe. Aquesta mesura per si sola pot ser enganyosa. Ja que per exemple un model del tipus f(x)=1, tindria un Recall del 100% sobre la classe 1, però això no significaria que es un bon predictor, doncs es patètic. Tot i així calculat sobre totes les classes i en acompanyament d'altres mesures es torna una mesura útil. En aquest cas es pot observar que el model genera te un Recall sobre la classe 0 del **72%**, mentre que sobre la classe 1 te un Recall del **77,5%** la qual cosa vol dir que té una major facilitat per detectar correctament els tweets positius (només un 22,5% d'aquests no son detectats). Aquestes dades tal i com es pot observar son complementaries a les anteriorment analitzades (precision) i expliquen de millor manera la situació que es produeix en el model. La menor precisió sobre la classe 1 (68,83%) que s'havia observat, es deu doncs a que existeix un major nombre d'exemples de la classe 0 que son classificats erròniament (com a classe 1), provocant doncs un descens en la precisió d'aquesta classe.

Quina és la millor estratègia de validació? I la millor mesura d'error?

En aquest cas la resposta a la primera pregunta es clara. De entre les tres estratègies de validació analitzades (Cross-Validation Basic, K-Fold i LeaveOneOut), LeaveOneOut sempre es la que dona uns resultats amb major fiabilitat. Això es degut a que aquesta estratègia permet obtenir l'error entrenant models amb el total de les dades de que es disposa (a excepció de les dades de Test en cas de que s'estiguessin utilitzant aquestes). LeaveOneOut però es habitualment l'estratègia de validació que requereix d'una major quantitat de recursos. La qual cosa es tradueix normalment en un inassolible cost de temps o econòmic. En aquest cas però gràcies a la naturalesa del model ha pogut ser practicable i per tant es pot determinar aquesta com a mesura d'error més fiable. Tot i així, tal com s'havia vist en el cas de la mesura menys fiable, la qual es pot determinar que es Cross-Validation Basic (ja que per defecte genera només únic model), s'ha observat que la desviació estàndard de les diferents execucions no arriba al 0,05%, per tant es pot determinar que gràcies a l'estabilitat de la base de dades utilitzada, qualsevol de les tres estratègies de validació es a efectes pràctics un bona opció.

Respecte a la mesura d'error, es complicat parlar d'una millor o una pitjor mètrica, ja que totes les analitzades presenten informació des de un diferent punt de vista que pot donar informació sobre la base de dades que no es veia en primera instancia. Per simplicitat de presentació la millor mesura possiblement sigui l'Accuracy ja que expressa d'una manera genèrica i que te en compte totes les classes com es comporta de correctament en model generat i per tant com de

fiable pot ser una resposta d'aquest. D'altre banda si es té en compte el donar la informació mes completa i objectiva possible, la millor mètrica a presentar es la "Matriu de confusió" ja que es aquesta l'arrel de qualsevol mètrica i la que permet observar amb exactitud l'èxit o fracàs de la validació sobre cada classe concreta. Per últim com a comprensió més profunda del comportament del model, la Precission i el Recall son dos mètriques força útils també ja que surten de la generalització visual que produeix l'accuracy per mostrar el comportament específic sobre cadascuna de les classes de predicció.

Apartat B: Com afecta el tamany?

El motiu d'aquest anàlisis es observar fins a quin punt es torna important el numero d'exemples utilitzats per elaborar el model, ja que com s'ha vist anteriorment la base de dades semblava força estable i produïa un comportament semblant tant si el model era generat utilitzant un 66,66...% de les dades o un 99,99...% d'aquestes. A més d'analitzar l'afectació que té el tamany del conjunt de dades utilitzat durant la creació del model, un altre terme a analitzar es l'afectació del tamany de diccionari utilitzat. Això voldrà dir el següent: Finalment el model creat, realitza les seves prediccions en base a 2 diccionaris on guarda per a cada paraula un mode d'extreure la probabilitat de que aquesta paraula aparegui en un tweet negatiu o en un tweet positiu. Llavors es pretén analitzar com afecta la quantitat de paraules que conté aquest diccionari a la capacitat de predir del model. Per això es necessari establir un criteri de truncament, el més lògic es doncs intentar guardar truncar paraules en ordre de probabilitat. Eliminant sempre del diccionari las n-m ultimes paraules en ordre descendent de probabilitat (o ocurrències), on n es el tamany del diccionari a truncar i m el nou tamany que se l'intenta donar.

Augment del conjunt d'entrenament amb mida lliure de diccionari

En aquest anàlisis s'ha tractat de veure com afecta en conjunt el creixement de totes dues variables. Al variar la mida del conjunt d'entrenament, la mida del diccionari també creix de manera natural. En la següent gràfica es mostra com varia l'accuracy del model en funció de la mida d'aquest conjunt d'entrenament.

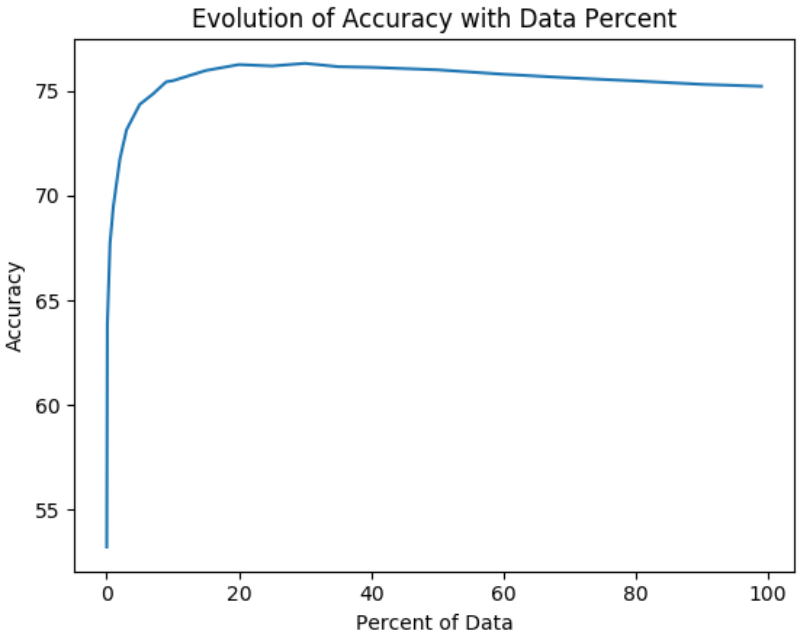


Figura 8: Accuracy generada en funció de la mida del conjunt d'entrenament (Creixement natural del diccionari)

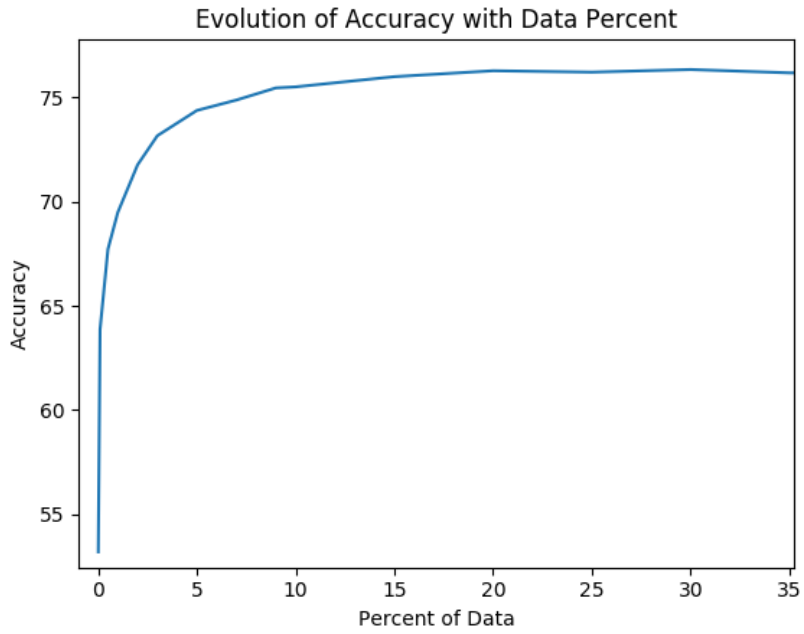


Figura 9: Accuracy generada en funció de la mida del conjunt d'entrenament (detall [0.01%-35%])

En la figura 8 i 9 es mostra aquesta evolució de l'accuracy que es venia intuint. Tal com es pot observar en totes dues figures, des de l'1% de dades que l'accuracy ja es situa per sobre del 71%

(sobre un 75,7% de màxim al seu major pic), a més també es important observar en la figura 8, que a partir d'aproximadament el 33% de dades del total, es troba un punt d'inflexió en la funció a partir del qual comença a decréixer l'accuracy, tal i com ja s'havia pogut observar en apartats anteriors. Aquest efecte pot ser degut al fort creixement d'algunes paraules sorolloses: Aquestes paraules son en molts cassos paraules que només apareixen un cop durant l'entrenament i realment son neutrals (pe. errates d'escriptura, o alguns #hashtags) al tornar-les a trobar durant la validació sumen a la classe a on pertanyien en l'entrenament, i produeixen una major probabilitat d'error. També per exemple provoquen una introducció de soroll algunes mencions (@el_del_quiosc), ja que el fet de que es menciones a aquest senyor en un tweet negatiu no vol dir que el fet de mencionar-ho sigui negatiu. També es important notar que en parlar d'una base de dades d'aquestes dimensions els conjunts de l'1% d'entrenament contenen encara més de 15.000 tweets. Un numero encara més gran traduït a paraules que expressa el motiu per el qual tan sols aquesta part del conjunt total de dades ofereix uns resultats tan propers al millor. Observant la figura 9 es pot observar que el creixement de la funció sembla tenir un punt crític en aproximadament el 10% de dades, i es a partir d'aquí on pràcticament termina el seu creixement, fins arribar al punt d'inflexió abans mencionat del 33%. A l'observar aquestes dades s'ha de tenir en compte que l'accuracy esperable d'un predictor basat en una funció aleatòria entre aquestes dues classes, es del 50%. Per tan queda demostrar que la xarxa generada efectivament es capaç de començar a millorar aquest valor a partir d'un nombre petit de dades.

D'aquest anàlisis es pot extreure que efectivament una porció de les dades de tan sols un 33% de les totals son suficient entrenament per al model com per donar el màxim de les seves capacitats. L'increment a partir d'aquí, no tan sols no aporta cap tipus de benefici si no que arriba a provocar un lleuger perjudici a la predicció. A més d'això podem trobar un punt a partir del 0,5% de dades d'entrenament on a partir del qual la funció de creixement de l'accuracy segons la mida del conjunt d'entrenament (i el diccionari) fa un canvi, indicant que a partir d'aquí cada nova dada comença a afegir cada cop menys informació al model

Afectació de la mida del diccionari amb mida prefixada del conjunt d'entrenament.

En aquest apartat es pretén analitzar l'afectació de la mida del diccionari utilitzat, en els capacitat de predir del model. Per això es prefixarà una mida del conjunt d'entrenament que en aquest cas serà la que s'ha vist anteriorment que era capaç de proporcionar un millors resultats, un 33% de les dades.

Amb aquesta mida fixada s'anirà variant el threshold de truncament entre el mínim (1) i la mida completa del diccionari (agafant el mínim entre la dimensió del diccionari negatiu i el positiu, per tal de que continguin el mateix nombre de paraules del diccionari) i així es podrà veure correctament l'afectació d'aquesta variable a la capacitat del predictor, a continuació es mostra el gràfic de variació generat per aquest anàlisis:

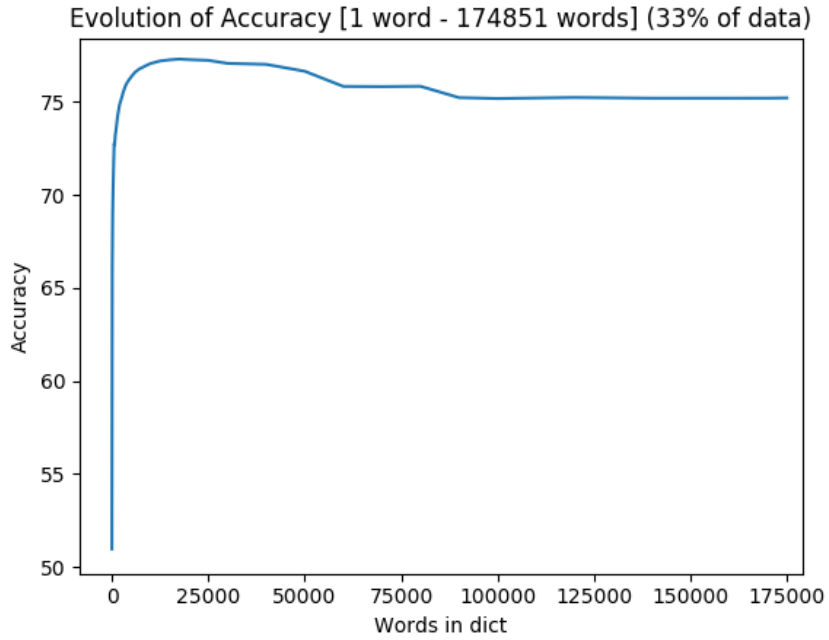


Figura 10: Accuracy generada en funció de la mida dels diccionaris utilitzats (percentatge de data fixat)

Tal com es pot observar en la figura 10 la gràfica de la funció aquesta vegada presenta un creixement i un decreixement molt més abruptes que en l'anterior anàlisis.

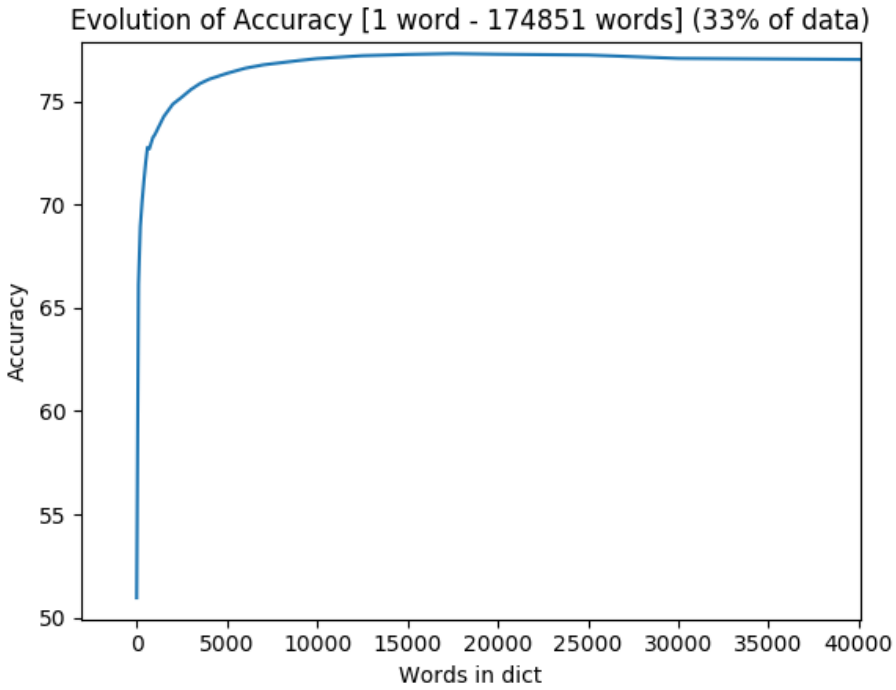


Figura 11: Detall de les 40.000 primeres paraules (sobre la figura 10)

De fet, com es pot observar en el detall de la figura 11, amb les 7.500 paraules més comuns, ja s'arriba pràcticament a un punt de pendent 0 de la funció, i en les 20.000 primeres paraules ja trobem el punt d'inflexió d'aquesta, que en aquest anàlisis dona lloc a un descens més marcat (del 76,4% al 75,2%). A més d'això es pot observar que les primeres 700 paraules son les que realment marquen la major part de l'aprenentatge (des de l'accuracy nul (50%) fins al 72,8%). Tot això ve a indicar que la variable "Nombre de paraules del diccionari" si que te responsabilitat sobre la forma de la funció abans observada, i reafirma la teoria de que el motiu del descens de la funció abans observada, a partir del 33% de les dades (amb creixement natural del diccionari) son les paraules sorolloses introduïdes per un excés de dimensió del diccionari. I truncant aquest es pot arribar a millorar fins a un 0,6% la predicció (amb un model entrenat sobre el 33% de les dades)

Afectació de la mida del conjunt d'entrenament amb mida prefixada del diccionari

Per aquest últim anàlisis s'ha establert la mida del diccionari en el threshold que ha donat millors resultats en l'anterior anàlisis. En aquest cas: 20.000 paraules, punt on s'aconseguia arribar al 76,4% d'Accuracy (sobre un conjunt d'entrenament del 33%). A continuació s'ha anat variant la mida del conjunt d'entrenament, per observar l'afectació d'aquesta variable de manera aïllada. A continuació es mostra una gràfica amb els resultats:

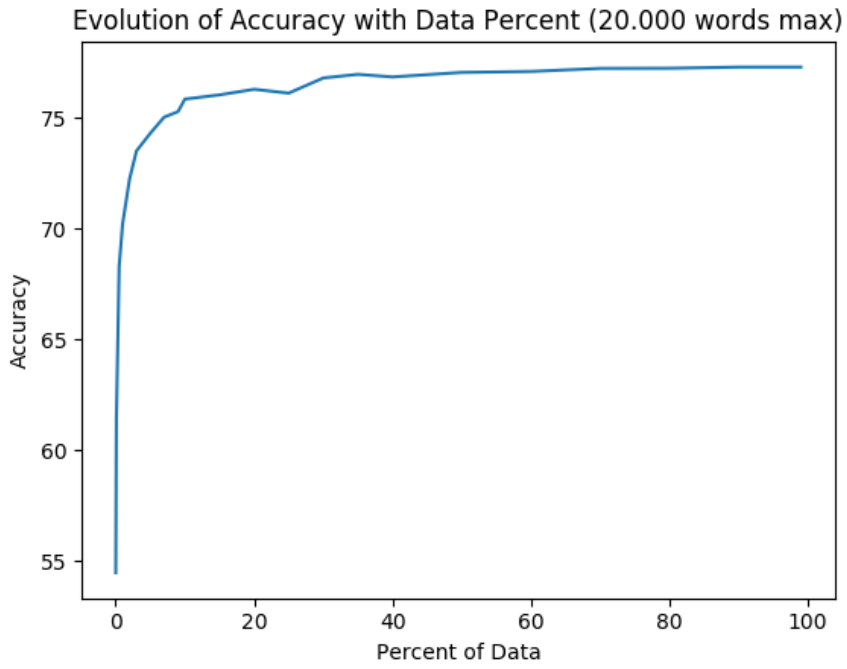


Figura 12: Accuracy generada en funció del percentatge de data utilitzat (mida màxima del diccionari fixada)

Observant la figura 12 es pot arribar a una primera impresio inesperada, la funció s'ha tornat creixent (a excepció de dos petites valls localitzades i aïllades). Aquesta forma creixent de la funció demostra que la variable "data percent" de forma aïllada no provoca una introducció de soroll, donant pes a la teoria de que en la funcio Acc(DataPercent&DictTam) l causa del descens de l'accuracy en augmentar el numero d'exemples observats, era efectivament la mida del diccionari i la quantitat de paraules sorolloses que s'introduien.

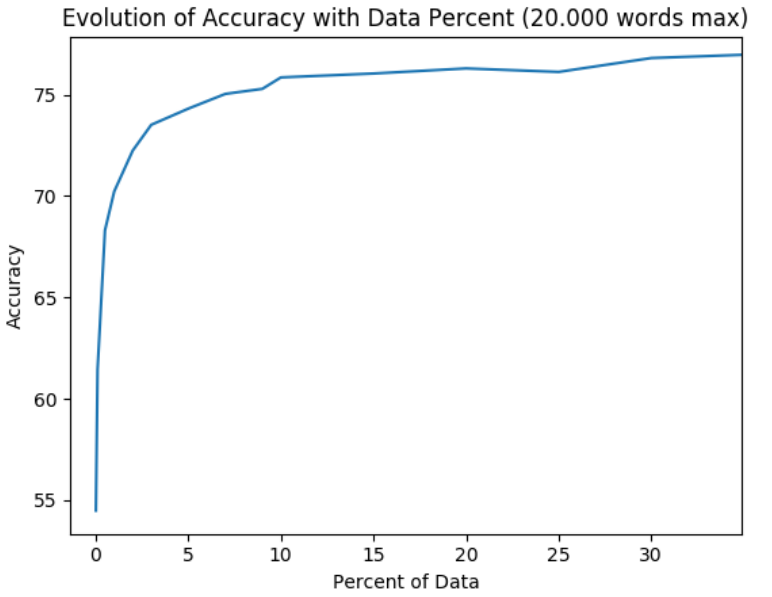


Figura 13: detall [0.01%-35%] sobre la figura 12

Com es pot observar en la figura 13, en arribar al 5% es produeix una reducció abrupta de la millora que proporciona la introducció de nous exemples, fins arribar a aproximadament el 33% (màxim també de la primera funció analitzada) moment on el pendent encara que sempre creixent es torna molt suau.

D'aquests tres anàlisis s'ha pogut extreure la següent conclusió: La Xarxa Bayesiana generada, es sensible a la introducció de paraules sorolloses, i aquest motiu es el que arriba a provocar, que la introducció de nou coneixement (exemples) arribi a generar un descens en l'aprenentatge (capacitat de fer prediccions correctes (Accuracy)). Aquest problema però es resoluble fixant com a constant la variable que provoca la introducció de soroll (nombre de paraules en el diccionari) de manera que es provoca que la Xarxa aprengui només de les paraules que més es repeteixen en cada etiqueta, les quals seran evidentment les que tinguin un significat més vinculat amb el sentiment expressat, que es la variable a predir.

Apartat A: Laplace Smoothing

Una de les possibles opcions per millorar el comportament del model, es l'aplicació en l'algoritme de mètodes de Laplace Smoothing. Laplace Smoothing canvia la formula utilitzada fins al moment:

$$\boldsymbol{P(word = A|sentLab = X_i)} = \max(\frac{Ocurrences\ Of\ Word\ \mathrm{A}\ in\ words\ observed\ on\ label\ X_i}{Total\ words\ observed\ on\ label\ X_i}, \varepsilon)$$

Per la següent:

$$\boldsymbol{P(word = A|sentLab = X_i)} = \frac{Ocurrences\ Of\ Word\ \mathrm{A}\ in\ words\ observed\ on\ label\ X_i + Smooth}{Total\ words\ observed\ on\ label\ X_i + (Count\ of\ diferent\ words\ observed\ on\ Label\ X_i * Smooth)}$$

On **Smooth** = un valor del rang (0,1] (habitualment 1)

Aquest afegit a la funció provoca una sèrie de variacions en el resultat, ja que ara per paraules inexistents, la formula en comptes de traduir-se en:

$$\boldsymbol{P(word = A|sentLab = X_i)} = \varepsilon$$

Es tradueix en:

$$\boldsymbol{P(word = A|sentLab = X_i)} = \frac{Smooth}{Total\ words\ observed\ on\ label\ X_i + (Count\ of\ diferent\ words\ observed\ on\ Label\ X_i * Smooth)}$$

Mentre que en paraules amb un elevat nombre d'ocurrències,
$Ocurrences\ Of\ Word\ \mathrm{A}\ in\ words\ observed\ on\ label\ X_i + Smooth \cong$
$Ocurrences\ Of\ Word\ \mathrm{A}\ in\ words\ observed\ on\ label\ X_i$

Aquest canvi provoca que tinguin una paraula que no apareix al diccionari positiu en comptes de tenir una probabilitat de "ε", tingui associada una probabilitat en funció de la grandària del seu diccionari. La modificació de l'algoritme es basa en la idea de que un succés no vist durant l'entrenament no pot tenir una probabilitat 0 associada (si en tota l'historia de tweeter ningú a escrit la paraula 'kleenex' no vol dir que sigui impossible que la paraula 'kleenex' aparegui en un tweet (de la mateixa manera si en 90.000 tweets s'ha utilitzat la paraula 'cat', comptar la probabilitat amb 90.001 te una afectació imperceptible al resultat de la divisió.

Quina afectació té Laplace Smoothing a la Xarxa Bayesiana construïda?

Per observar la diferencia entre aplicar Laplace Smoothing o no aplicar-ho s'han repetit anteriors procediments d'anàlisis, per d'aquesta manera poder realitzar correctes comparacions. El primer d'ells ha sigut l'estratègia de validació que resultats més fiables havia donat: LeaveOneOut

Accuracy	Recall class=0	Recall class=1	Precision class=0	Precision class=1
0.74436	0.71996	0.77495	0.80038	0.68827

Figura 6: Mesures d'error més significatives en LeaveOneOut sense Laplace Smoothing

Accuracy	Recall class=0	Recall class=1	Precision class=0	Precision class=1
0.77623	0.747631	0.81240	0.83442	0.71797

Figura 14: Mesures d'error més significatives en LeaveOneOut amb Laplace Smoothing

Tal com es pot observar en les dues figures superiors l'aplicació de Laplace Smoothing es capaç d'aconseguir fins a un 3% de millora en el resultat de totes les mètriques d'error preses respecte al model que no utilitzava Laplace Smoothing. Una bona mètrica per realitzar comparacions entre models que no s'havia utilitzat fins ara es l'F1-Score → 2TP/(2TP+FP+FN), una mitja harmònica entre precisió i recall que permet donar una puntuació als resultats d'un model molt útil per a realitzar comparacions:

	Non-Smooth model	Smooth model
F1-Score class = 0	0.75804	0.77335
F1-Score class = 1	0.72905	0.762275

Figura 15: Taula comparativa de F1-Score entre els models sense i amb Laplace

Com es pot veure a la taula superior el model que utilitza Laplace Smoothing aconsegueix estar 0.033 punts per sobre en F1-Score respecte al model que no l'utilitza en la classe 1 i 0.015 punts per sobre en la classe 0. Això permet determinar de manera objectiva que efectivament l'aplicació de Laplace Smoothing en el model comporta una millora efectiva en la predicció respecte a l'anterior model

Per fer la comparació el més detallada possible es mostra en la figura següent una comparació entre la matriu de confusió de tots dos mètodes.

Non-LS	**Pred = 0**	**Pred = 1**
Class = 0	626.416	243.657
Class = 1	156.236	537.993

LS	**Pred = 0**	**Pred = 1**
Class = 0	653.062	220.446
Class = 1	129.590	561.204

Figura 16: Comparació matrius de confusió sense Laplace-Smoothing vs amb Laplace-Smoothing

Com es pot observar en la figura 16, efectivament respecte la utilització de l'algoritme de Laplace Smoothing ha introduït una millora en tots els camps de la matriu de confusió (Increment de TP i TN, i descens de FP i FN).

A continuació es mostren les gràfiques de variació de l'accuracy en funció de la mida del conjunt d'entrenament (deixant créixer de manera natural la mida del diccionari). En blau la gràfica sense aplicar Laplace Smoothing (idèntica a l'anterior) i en taronja la gràfica resultat de generar una xarxa que valida amb Laplace Smoothing sobre el mateix conjunt de dades.

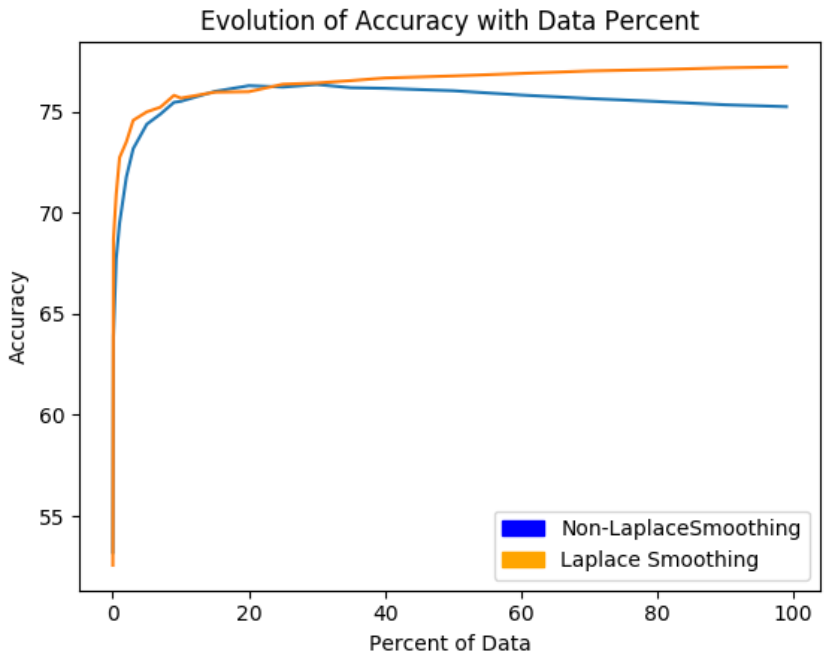


Figura 16: Accuracy generada en funció de la mida del conjunt d'entrenament amb creixement natural del diccionari (comparació)

Com es pot observar entre les dos funcions, l'aplicació de Laplace Smoothing en la predicció, provoca un substancial canvi en aquesta, arribant a tornar la funció creixent. A més com ja s'havia observat en les figures 14 i 15, la introducció de l'algoritme de Laplace arriba a produir una millora en l'Accuracy fins a 3 punts superior. Observant el primer tram de la funció [0%-5%], es possible observar que la predicció de Laplace Smoothing aconsegueix arribar als mateixos resultats d'Accuracy que funció que no ho utilitza, amb un conjunt de dades d'entrament menor, proporcionant per tant un creixement mes ràpid. En conclusió sembla que a efectes pràctics produeix uns millors resultats en la creació d'una xarxa amb creixement lliure.

Per poder analitzar com afecten a aquest algoritme cadascun dels paràmetres de forma aïllada s'estableix un cop mes la mateixa mida anterior sobre el conjunt d'entrenament (33%), es mostra en la gràfica següent com afecta la variació del threshold del diccionari a l'accuracy dels resultats del model (en blau la validació sense Laplace Smoothing i en taronja la validació amb Laplace Smoothing)

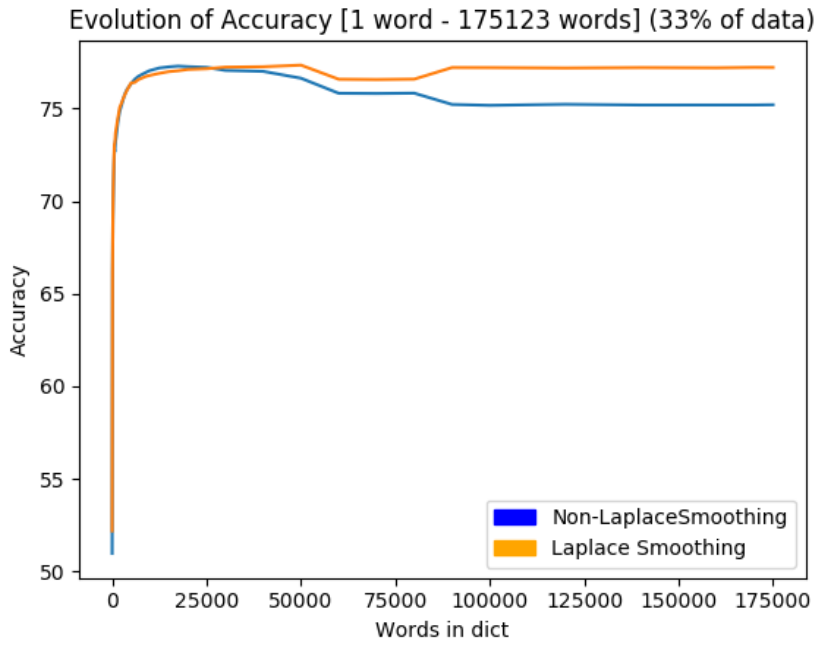


Figura 17: Accuracy generada en funció de la mida dels diccionaris utilitzats (percentatge de data fixat)

En la figura 17, es pot observar ara el motiu de millora de l'algoritme de Laplace Smoothing. Aïllant la variable "Data percent" es pot observar que un augment en el nombre de paraules del diccionari (i per tant un creixement de les paraules sorolloses) no provoquen en aquest cas un descens en l'accuracy sinó que mantenen la funció plana (A excepció duna petita vall localitzada). Aquest fet es capaç d'explicar que la millora que aporta l'algoritme de Laplace Smoothing resideix principalment en una major protecció davant del soroll provocat per paraules amb un nombre mínim (habitualment 1) de repeticions.

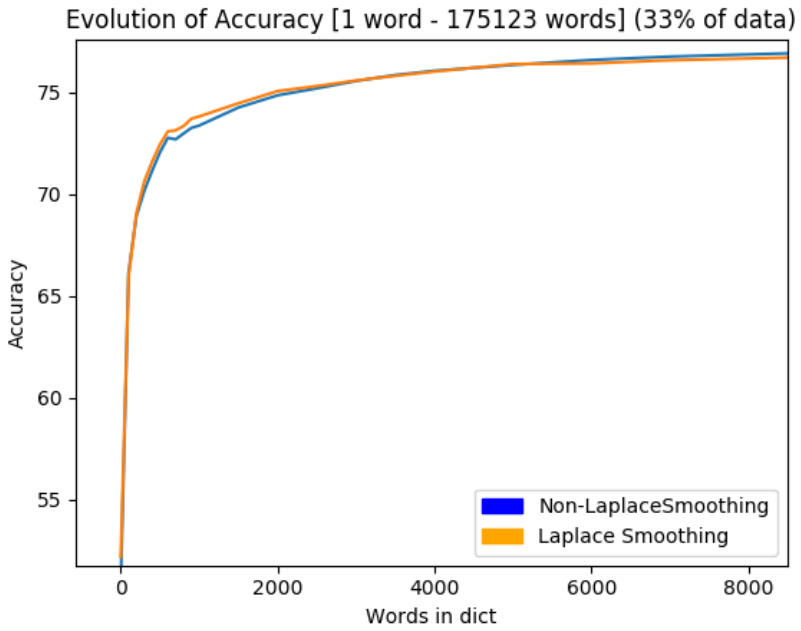


Figura 18: Detall de les 8.000 primeres paraules (figura 17)

En el detall de la figura 18, es pot observar que en els primers trams de la funció tant la que no utilitza Laplace Smoothing com la que si ho utilitza, son pràcticament idèntiques (tot i tornant presentant la segona un creixement lleugerament més ràpid) això es esperable de manera lògica ja que com s'havia observat anteriorment, amb el 33% de dades d'entrenament prefixat, l'afectació per paraules sorolloses no comença a afectar fins a mida = 20.000. Per tant no es fins aquest punt que ambdues funcions es desmarquen de manera notable.

Per últim i de la mateixa manera que abans, es mostra l'evolució de la gràfica d'Accuracy en funció de la mida del conjunt d'entrenament amb una mida de diccionari prefixada en 20.000 paraules (mida que millors resultats donava en la funció que no aplica Laplace). Per veure l'afectació d'aquesta variable de manera aïllada en cadascuna de les dues xarxes.

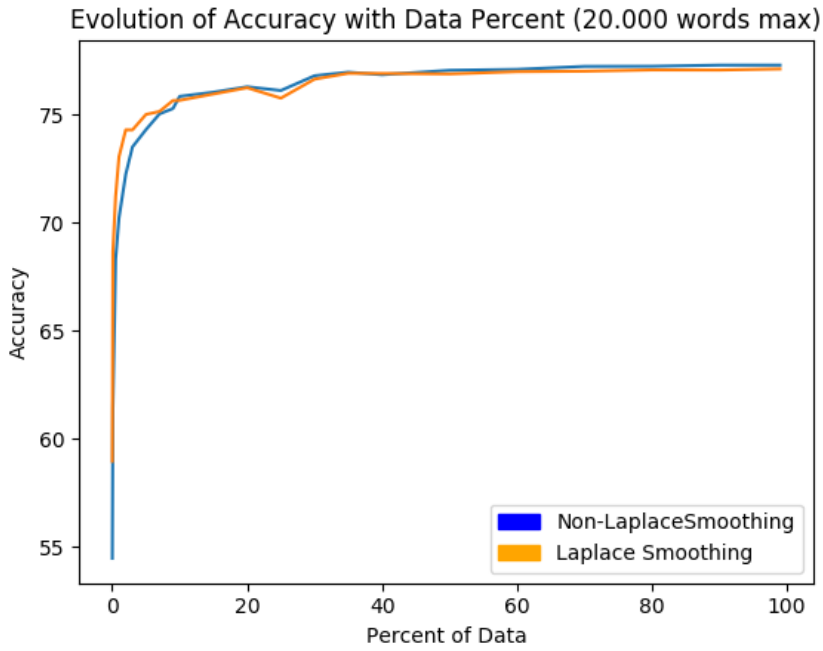


Figura 19: Accuracy generada en funció del percentatge de data utilitzat (mida màxima del diccionari fixada)

Tal com es pot observar, en les condicions de truncament de diccionari optimes, totes dues xarxes presenten un comportament practicament identic (inclos molt lleugerament superior) l'algoritme de predicció que no utilitza Laplace Smoothing

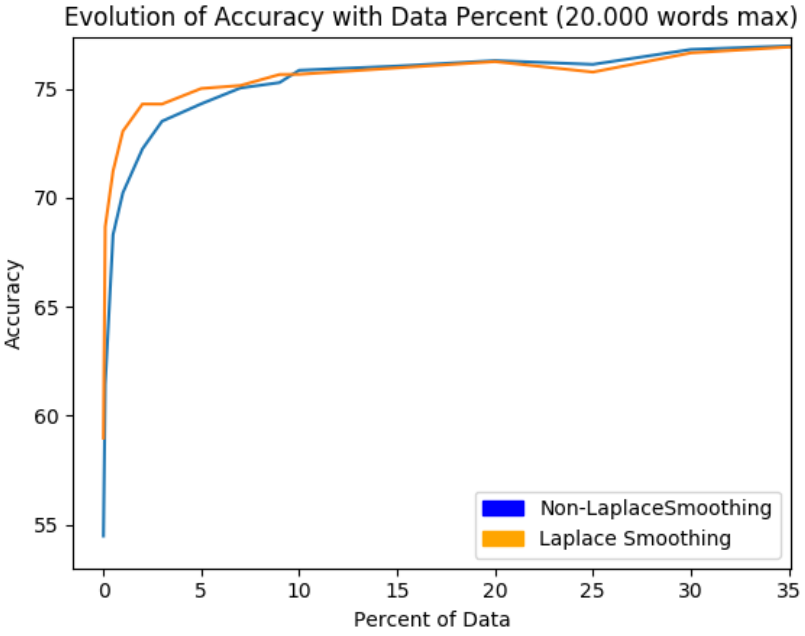


Figura 20: Detall [0.01%-35%] (figura 19)

Tot i així es pot observar que com ocorre en el detall de la figura 20 que en la primera secció de la funció l'algoritme de Laplace Smoothing, continua tenint un creixement molt més ràpid, degut a que en aquest punt la mida del diccionari encara no arriba al threshold marcat i per tant l'afluència de paraules sorolloses (per a les quals Laplace Smoothing ofereix una protecció) es major.

De tot el anàlisis fet es pot extreure com a conclusió que l'algoritme de Laplace Smoothing es capaç d'oferir uns millors resultats de manera general degut a la protecció que ofereix en contra de paraules sorolloses. La qual cosa provoca que aquestes no perjudiquin de manera notable el seu aprenentatge com si ocorria en l'algoritme de predicció anteriorment analitzat. Encara així existeixen altres eines de protecció en contra d'aquestes paraules sorolloses, com es per exemple el truncament del diccionari (mitjançant un criteri de ordre descendent). L'aplicació d'aquest truncament permet doncs obtenir resultats pràcticament idèntics amb tots dos algoritmes, tornant ambdues funcions creixents amb respecte al nombre d'exemples analitzats.

Problemes sorgits durant la pràctica:

Durant la elaboració de la practica, han anat sorgint una sèrie de problemes que s'han anat podent resoldre fins arribar al software final, a continuació s'exposen els mes rellevants:

Problemes amb la lectura de la base de dades: A l'hora de llegir la base de dades, vam tenir unes hores de mals de caps, perquè no coneixíem la utilització de la llibreria "Pandas" i ens vam haver d'informar a fons per poder exprimir el màxim de possibilitats que ens oferia amb la instrucció de lectura de csv. Per exemple a l'inici vam poder detectar que per defecte estàvem perdent la informació dels hashtags ja que aquests eren detectats com a missing vàlues, i vam haver de trobar com eliminar aquests filtres ja que vam determinar que els hashtags eren realment paraules que donaven una informació força rellevant

Altres problemes amb Pandas: Durant l'elaboració del software van sortir moltes pèrdues indesitgades d'eficiència a causa d'una utilització incorrecta de la llibreria "Pandas". Aquestes per sort van poder ser detectades a temps i resoltes abans de ocasionar grans pèrdues de temps de treball.

Conclusions:

Durant l'elaboració d'aquesta pràctica s'han pogut assentar correctament els coneixements adquirits a teoria sobre xarxes bayesianes i Naïve Bayes, i s'ha pogut veure quin es el seu comportament sobre bases de dades reals. A més d'això s'han pogut veure els efectes reals de les diferents estratègies de validació com poden ser "Cross-Validation", "K-Fold" o "LeaveOneOut" s'ha comprovat com afecta cada una i quina es millor en una ocasió o altre. (Per exemple LeaveOneOut que es una estratègia que dona una estimació d'error molt propera a la real, però es torna en alguns casos concrets una estratègia inaplicable pel temps que trigaria en executar-se)

A més s'ha pogut comprovar realment la utilitat de les diferents mètriques d'error per a poder observar el comportament d'una solució des de diferents punts de vista, ja que mentre que

accuracy per exemple dona una visió molt simple i genèrica per donar una idea general del funcionament. Altres mètriques com la precisió o el recall que poden ser mesurades sobre una classe concreta donen una visió més detallada de com s'està comportant la predicció sobre classes determinades, increïblement útils per a problemes on els errors en cada classe tenen una penalització diferent. A més d'això mètriques com l'F1-Score, poden ser molt útils per a realitzar comparacions entre models.

També s'ha pogut observar de manera pràctica com afecten a un model les variables sorolloses i com es que alguns mètodes com per exemple "Laplace Smoothing" poden ser útils per obtenir una major protecció davant dels inconvenients que aquestes provoquen. Gràcies a això s'ha pogut determinar que en problemes que busquen fer un anàlisis de significança en frases cal tenir molt de cura en quines son aquelles paraules de la frase analitzada que realment estan tenint un pes sobre el significat complet, per evitar així caure en el soroll de paraules irrellevants en allò que busquem.

Com a competències transversals, s'ha pogut millorar la capacitat de treball en grup, el coneixement sobre diferents llibreries del llenguatge Python, incloses Numpy i Pandas, i altres eines de treball compartit.