UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU EN ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica

Universitat de Barcelona

Framework d'automatització de processos: Classificador de textos fent ús de xarxes neuronals amb la infraestructura de correu electrònic adjacent

---

Autor: Guillem Espín Martí

Director: Dr. Santi Seguí Mesquida

Realitzat a: Departament de Visió per computador i aprenentatge automàtic

Barcelona, 11 de juny de 2023

1 Abstract

Based on the need for process automation for cost and time reduction, this project aims to showcase a demo of a possible marketable product that exploits this opportunity by making use of neural networks and automation of Windows process and email services. This project consists of two main parts: the model and the framework.

The model: The implemented model is a text classification model that makes use of neural networks. It has been trained with the data provided by the Kofax company, with a total of 700 samples from six different categories; invoices, salary notes, etc. The model has an accuracy of around 90%. It has been benchmarked against other popular classification alternatives used today and optimized as much as possible in order to obtain a satisfactory result within the environment in which it will work.

The framework: It consists of a library developed from scratch in order to make use of the model and implement it in a practical, useful and visual way in order to end up with a final product suitable for the market with the title of demo. This framework allows you to deploy an automated email service that works with Outlook. This service consists of receiving, processing, classifying (using the model) and sending data in email format. This library is interchangeable, adaptable and implementable by any user who wants to deploy the service.

2 Resum

Partint de la necessitat d'automatització de processos per la reducció de costos i temps, aquest projecte és una demo d'un possible producte portable a mercat que explota aquesta oportunitat fent ús de xarxes neuronals, automatització de processos de Windows i serveis de correu electrònic. Aquest projecte consta de dues parts principals: el model i el framework.

El model: El model implementat és un model de classificació de textos que fa ús d'una xarxa neuronal. Ha estat entrenat amb les dades facilitades per l'empresa Kofax, amb un total de 700 mostres, constant de sis categories diferents; factures, notes salarials, etc. El model té una precisió d'entorn del 90%. Ha estat comparat contra altres alternatives de classificació popularment utilitzades avui en dia i s'ha optimitzat el màxim possible per tal d'obtenir un resultat satisfactori dins de l'entorn en el qual treballarà.

El framework: Consisteix en una llibreria elaborada des de zero per tal de fer ús del model i implementar-lo d'una manera pràctica, útil i visual per tal d'acabar formant un producte final apte pel mercat amb títol de demo. Aquest framework permet desplegar un servei automatitzat de correu electrònic que funciona amb Outlook. Aquest servei consisteix en la recepció, tractament, classificació (fent ús del model) i enviament de dades en format d'email. Aquesta llibreria és permutable, adaptable i implementable per qualsevol usuari que vulgui desplegar el servei.

3 Agraïments

Primer de tot vull distingir dues parts importants que han format part d'aquest projecte:

Per part de l'empresa vull agrair al meu tutor de pràctiques, en Moez, que sempre s'ha preocupat de mantenir-me i ha escoltat les meves inquietuds a l'estada que m'ha prestat dins de l'empresa, a la vegada, presentant-me al Fermín, el meu tutor del projecte dins de l'empresa, el qual m'ha acompanyat i supervisat el treball que he desenvolupat, accedint a modificacions per tal de suplir els requisits exigits pel meu tutor dins de la universitat. A la vegada m'ha suplit amb totes les mostres i bases de dades necessàries que he utilitzat per poder construir part del meu projecte, concretament m'ha facilitat amb totes les dades que ha pogut per tal que jo aconseguís construir un model de classificació apte. Seguidament i finalment vull donar les gràcies a en Petr, el qual m'ha ajudat moltíssim en el tractament de dades inicial, el qual m'ha servit per poder proccessar les dades dins del meu model i poder construir el model final.

Per part de la universitat vull agrair a en Simone Baloco que a part d'informarme sobre el funcionament, terminis i burocràcia involucrada en la matriculació tant de pràctiques com del treball final de grau, ha anat més enllà i s'ha preocupat pel meu benestar personal a l'hora de cursar les pràctiques així com el TFG.

Finalment, vull agrair immensament a en Dr. Santi Segui, el qual ha sigut una peça principal dins del projecte, a més de prestar-me els seus extensos coneixements sobre la matèria en qüestió, s'ha involucrat des del primer dia, tant amb mi com amb l'empresa per assegurar-nos que tots estem a la mateixa pàgina i perquè el projecte sigui apte i funcional.

Índex

4 Introducció i motivacions

4.1 El projecte

Partint de l'oportunitat de mercat del món de l'automatització de processos, i de la família de productes els quals l'empresa Kofax produeix, i unint la meva curiositat i motivació per la recent cursada optativa que imparteix en Santi Segui d'Aprenentatge automàtic, en Fermin es va oferir per fer una primera proposta de projecte. Aquest consistiria a extreure dades rellevants o importants a partir de documents. Exemple: Extreure noms dels involucrats, les quantitats, el motiu, etc, a partir de documents notarials. Aquesta primera proposta encaixava amb l'assignatura principal en la qual està basada el projecte, ja que involucraria models d'aprenentatge automàtic. Aquest projecte doncs formaria el que seria una demo de producte final per tal d'estudiar la possibilitat de llençar el producte final al mercat.

Després que l'empresa em presentés les dades amb les quals hauria de treballar, ens vam adonar que no tindríem prou recursos per poder assolir l'objectiu final. Teníem una mancança de dades i a la vegada massa varietat entre les mateixes. Teníem molts tipus de documents però pocs documents per cada tipus. L'entorn ideal hauria sigut tenir moltes mostres sobre un mateix tipus de document, de manera que 'estandarditzaríem' la recopilació de dades rellevants.

En tenir aquesta limitació tècnica, i aquesta varietat de documents, vam optar per redirigir l'enfocament inicial del treball i construir un model classificador de text. De manera que partint del text dels documents originals proveïts per l'empresa, el model s'encarregaria de discernir entre tipus de documents. Partint de 6 classes diferents i un total de 700 mostres, el model fent ús de xarxes neuronals, és capaç de discernir entre les 6 classes diferents amb una precisió aproximada del 90%.

Per tal de poder tenir un producte final apte i usable, el model acaba sent una part petita, però no per això menys important de tot el projecte. El projecte sencer doncs consisteix en un framework/app per tal de desplegar un servei de correu electrònic automatitzat de classificació de documents. Aquest framework és adaptable a cada necessitat i projecte, de manera que és implementable per tothon. La peça clau doncs és el model, el qual és permutable depenent de la necessitat.

5 Objectius

Tret de la redirecció inicial de la idea principal del projecte, la idea i objectiu principal sempre han estat el mateix encara que a mesura que ha anat avançat en el projecte, ens hem anat adonant que hi hauria moltes més fites i feina requerida poder acabar desenvolupant la cota inicial. Per tant, els dos objectius principals són:

1. El model:
    - Investigar, implementar i comparar almenys dos models classificadors de textos d'aprenentatge automàtic adients per a la tasca de classificació de textos (Naive bayes i Neural Networks).
    - Desenvolupar el millor model d'aprenentatge automàtic a partir de les dades facilitades per l'empresa per tal d'obtenir un classificador de textos el més precís possible fent servir l'arquitectura més adient.
2. El framework:
    - Desenvolupar una aplicació, infraestructura o llibreria per tal de poder implementar el model de manera pràctica. Obtenir un resultat pràctic fent ús del model desenvolupat per a la classificació de textos a partir de documents adjuntats a correus electrònics i l'enviament pertinent a partir d'aquesta classificació.

- Exemple:

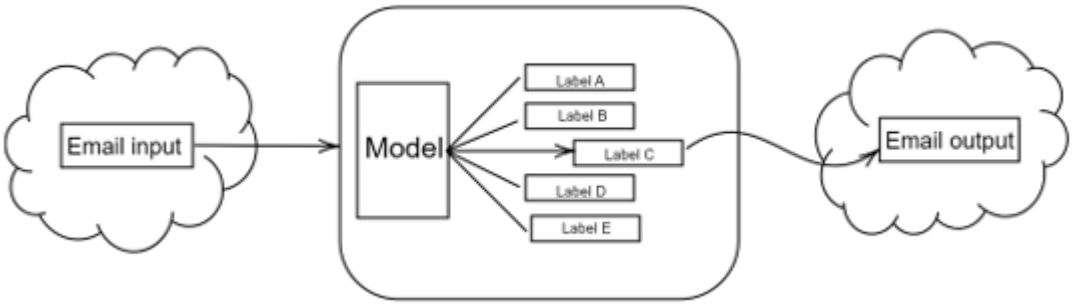

Fig 1.0 - Diagrama exemple de la idea funcional de la implementació.

6 Planificació

1. Recopilació i tractament de dades:
    - Recopilar, tractar i processar la 'raw data' facilitada per l'empresa
    - Neteja de les dades
    - Generar un dataset vàlid per poder començar a fer proves i crear un baseline.
2. El Model:
    - Investigar les diferents opcions disponibles per desenvolupar un model classificador
    - Crear una baseline amb qualsevol model
    - Implementar i comparar les opcions
    - Desenvolupar el model triat el màxim possible dins de les limitacions de l'entorn
    - Testejar i analitzar el rendiment
    - Millorar el model
        - Data augmentation (desenvolupament de dades sintètiques)
        - Estudi i optimització de les capes
    - Exportar el model

3. El Framework:

    - Servei de correu:
        - Estudiar les possibilitats d'automatització de serveis de Windows via Python
        - Estudiar les possibilitats d'automatització de serveis de correu via Python
        - Escollir i desenvolupar una combinació d'opcions
        - Estudiar, desenvolupar i testejar el funcionament general dels serveis involucrats
    - Tractament de fitxers:
        - Estudiar les vies de tractament dels fitxers d'input
        - Escollir i desenvolupar una opció
        - Estudiar i testejar funcionament general de totes les funcionalitats implementades
    - Llibreria — Framework:
        - Elaborar un pla de funcionament o workflow de l'aplicació/llibreria/servei
        - Dissenyar l'aplicació i totes les llibreries involucrades al framework
        - Desenvolupar totes les classes involucrades i funcionalitats
        - Testejar el rendiment de l'aplicació
        - Desenvolupar un exemple d'implementació
        - Comentar la llibreria
    - Memòria
        - Estructuració de l'esquelet general de la memoria
        - Redacció de la memoria
        - Correcció
    - Presentació
        - Desenvolupament
        - Pràctica oral

*Podeu consultar els annexos nº1.1, nº1.2 i nº1.3 per visualitzar un diagrama de Gantt amb les tasques desenvolupades en més detall.

7 Costos

Presentació dels costos teòrics del desenvolupament d'aquest projecte:

Titol	Hores	Import/h	Import total
Obtenció i recopilació de dades (Extern)	10h	25€	250€
Hores treballades (Autor)	774h	20€	14000€
Total invertit	784h	-€	14250€

Fig 2.0 - Detall dels costos associats d'aquest projecte

Tenint en compte que podríem plantejar aquesta demo com a producte final, aquest es comercialitzaria com a producte tancat fent ús de llicències úniques i intransferibles. L'objectiu de la comercialització sent obtenir el màxim benefici net a partir de la inversió inicial. Tot i que també tenim l'opció d'oferir aquest producte com a servei, plantejarem la comercialització del producte amb la venda de les còpies úniques i permanents pels usuaris.

Projecció teòrica del rendiment del producte a partir de les còpies venudes i el preu teòric de venda a particulars (preu/unitat) per aconseguir un benefici respecte a la inversió inicial del 0%, 20%, 50% i 100% (ROI, Return Of Investment).

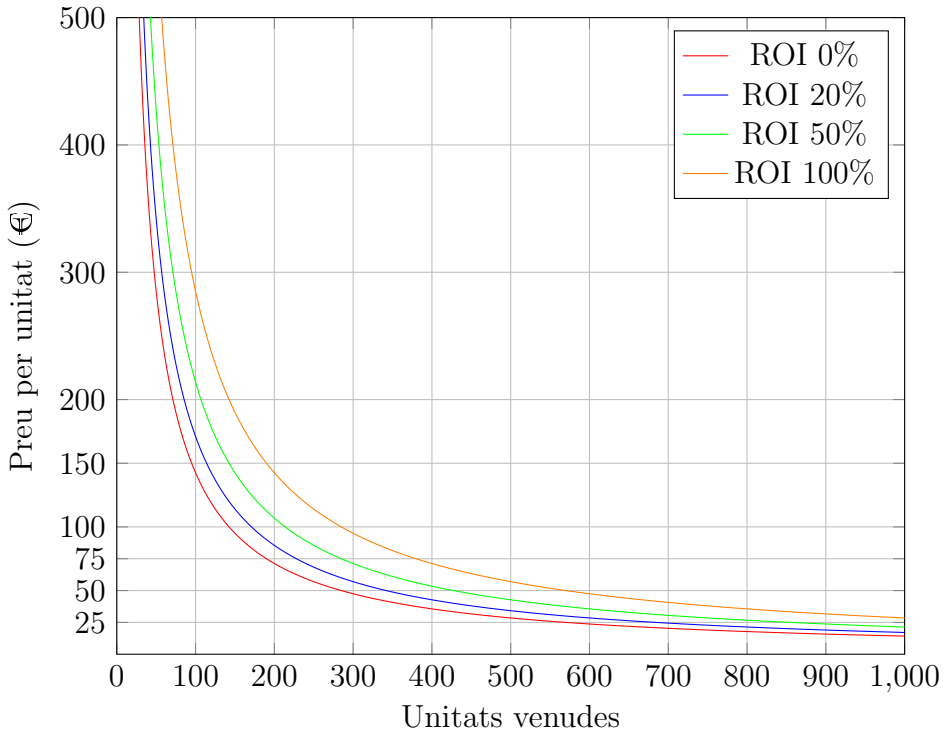

Fig 2.1 - Diagrama representant relació preus i unitats venudes per aconseguir x benefici

8 Classificació de documents

El processament de llenguatge natural és una de les àrees de la intel·ligència artificial que ha experimentat una gran atenció en les últimes dècades i més en els últims anys. Una de les tasques més comunes en aquest camp és la classificació de text, que consisteix a assignar una o diverses etiquetes a un text en funció del seu contingut. En aquest context, dos models de classificació de text àmpliament utilitzats per aquesta tasca, són Naive Bayes i les Xarxes Neuronals. Tots dos models tenen diferents enfocaments i característiques que els fan adequats per a diferents tipus de problemes i conjunts de dades.
En aquest apartat explicaré la idea principal del funcionament d'aquests dos models així com la implementació duta a terme per resoldre el problema de classificació de textos. L'objectiu serà implementar i comparar els dos classificadors i escollir un com a final candidat per tal de millorar-lo el màxim possible i incloure'l com a part de la solució final. L'input doncs d'aquests classificadors seran strings, els quals representaran el contingut de text dels documents.

8.1 Les dades

Les dades facilitades per l'empresa Kofax, consisteixen en una col·lecció de documents de caràcter notarial constant de sis tipus diferents de documents i vàries mostres per cada tipus de document. En total disposo d'un total de 700 mostres repartides de manera no uniforme entre els sis tipus:

- Declaracions de renta
  - Descripció: És un document en el qual cada persona ha d'especificar tots els rendiments que ha obtingut al llarg d'un any fiscal. La finalitat d'elaborar aquest document és actualitzar la nostra situació fiscal i informar d'ella a l'autoritat pública competent del nostre país, com pot ser l'Agència Tributària. Per aquestes característiques podem assumir que el document presentarà molts signes i números.
  - Mida: 3-5 pàgines — 800-20000 paraules
  - Format original del document: PDF
  - Etiqueta: DR
- Factures
  - Descripció: Compte en la qual es detallen les mercaderies comprades o els serveis rebuts, juntament amb la seva quantitat i el seu import, i que es lliura a qui ha de pagar-la. Document que tendeix a tenir poc volum de text.
  - Mida: 1-2 pàgines — 350-5000 paraules
  - Format original del document: PDF
  - Etiqueta: F
- Fotocòpies de DNI
  - Descripció: Document identitari espanyol el qual presenta les dades personals bàsiques de la persona a la qual representa. A més de seguits complexos de simbols i números a la part posterior.
  - Mida: 0.5 pàgines — 200-600 paraules
  - Format original del document: PNG-JPG
  - Etiqueta: FD

- Informes de taxació
  - Descripció: És un escrit realitzat amb metodologia normalitzada per un tècnic especialitzat d'acord amb la finalitat sol·licitada d'obtenir el valor d'un bé com, per exemple: un habitatge, edifici, joia.
  - Mida: +5 pàgines — 2000-60000 paraules
  - Format original del document: PDF
  - Etiqueta: IT
- Notes simples
  - Descripció: És un document públic de caràcter informatiu expedit pels registres, actuals o futurs. També se la coneix com a nota simple registral.
  - Mida: 1-5 pàgines — 500-6000 paraules
  - Format original del document: PDF
  - Etiqueta: NS
- Tiquets de gasto
  - Descripció: Documents demostratius de compres efectuades en comerços de venda al públic, com un per exemple un supermercat o una tenda d'electrodomèstics.
  - Mida: 0.5 pàgines — 200-600 paraules
  - Format original del document: PNG-JPG
  - Etiqueta: TG

Així doncs per mantenir una nomenclatura simple dins del model i del treball, anomenaré les etiquetes per les seves sigles: DR, F, FD, IT, NS i TG.

8.1.1 Recompte i estudi previ

El recompte inicial de les mostres amb les quals vaig comptar per entrenar el model classificador inicialment va ser de 378, distribuïdes de la següent manera:

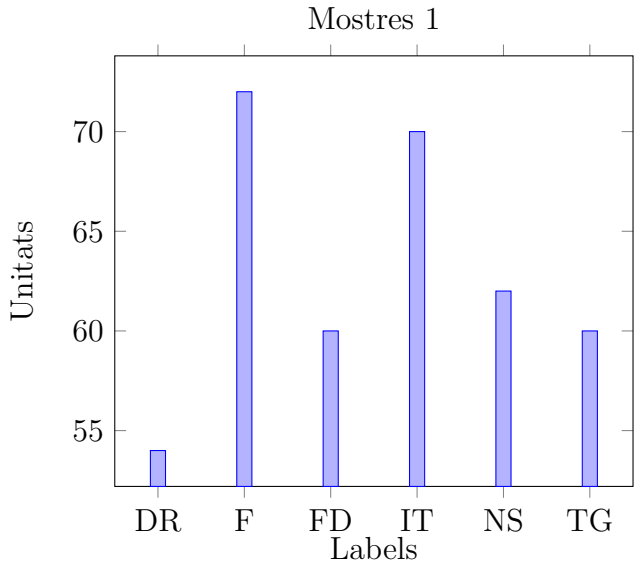

Fig 3.0 - Diagrama representant les mostres incials distribuïdes en els sis tipus diferents (DR,F,FD,IT,NS,TG)

La quantitat de mostres per tipus o etiqueta:
'DR': 54, 'F': 72, 'FD': 60, 'IT': 70, 'NS': 62, 'TG': 60
Com es pot observar, hi ha bastant disparitat entre el nombre de mostres per cada tipus o etiqueta.
Aquestes dades seran utilitzades per formar la baseline i més endavant seran emprades també per comparar els dos models de classificació.

Durant l'evolució del projecte vaig aconseguir una ampliació del dataset, aquestes dades també van estar facilitades per l'empresa, per acabar amb una col·lecció total de 701 mostres:

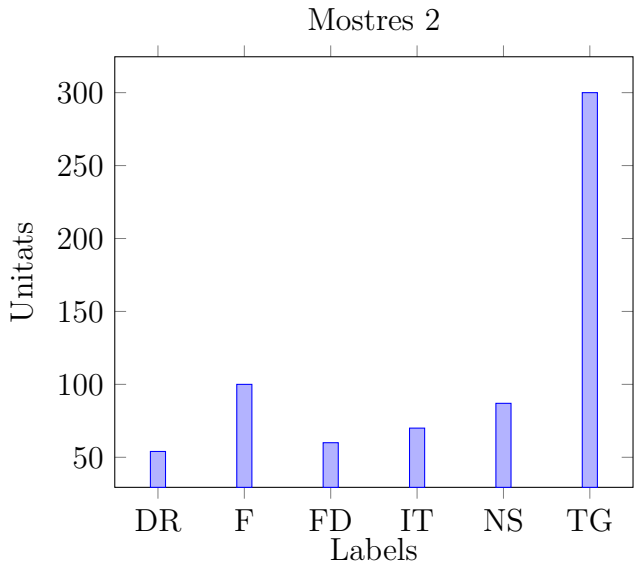

Fig 3.1 - Diagrama representant les mostres finals distribuïdes en els sis tipus diferents (DR,F,FD,IT,NS,TG)

La quantitat de mostres per tipus o etiqueta:
'DR': 54, 'F': 100, 'FD': 60, 'IT': 70, 'NS': 87, 'TG': 330
A simple vista ja podem observar el primer problema. La variància ha augmentat per culpa de les noves mostres, concretament les que tenen l'etiqueta de 'TG'.

Resum i comparació entre datasets:

Dataset	Variància	Desviació estàndard	Mitjana
Petit	46	6.78	63
Gran	11196	105.81	116

Fig 3.2 - Estructura o resum del contingut del fitxer csv

Així doncs, el primer obstacle que m'he trobat ha sigut estudiar la possibilitat de reduir la variància de manera sintètica o estudiar si aquesta alta variància m'afectaria a l'hora de desenvolupar el model. Tot i així, ja que el segon dataset no l'obtindria fins al cap d'un temps, el primer model i baseline es desenvoluparà amb el dataset petit.

8.1.2 Preprocessament

Per tal de poder tractar amb les mostres d'una manera fàcil, còmoda i ràpida, el primer pas era a partir dels fitxers, aconseguir el contingut en text, en altres paraules, en el meu entorn de Python necessitava tractar cada mostra com a una string per poder posteriorment generar un model.

Per tal d'obtenir el text de les 700 mostres, he fet servir el software que distribueix l'empresa Kofax, anomenat Kofax Transformation Modules. Aquest software et permet entre moltes altres coses transformar documents amb format PDF en documents TXT. Tot i que hi ha mostres de les quals disposo que no són PDFs i són imatges en format JPG o PNG, el procés que seguiran és el mateix. El software doncs l'únic que es limita a fer és a convertir tots els documents en un format el qual jo posteriorment podré fer servir per treballar. La metodologia que utilitza el programari per obtenir el text a partir de matrius (imatges) es diu OCR (Optical Character Recognition) el qual també està implementat utilitzant xarxes neuronals.

Un exemple d'execució d'aquest procés previ seria el següent:

1. A partir d'un document exemple...

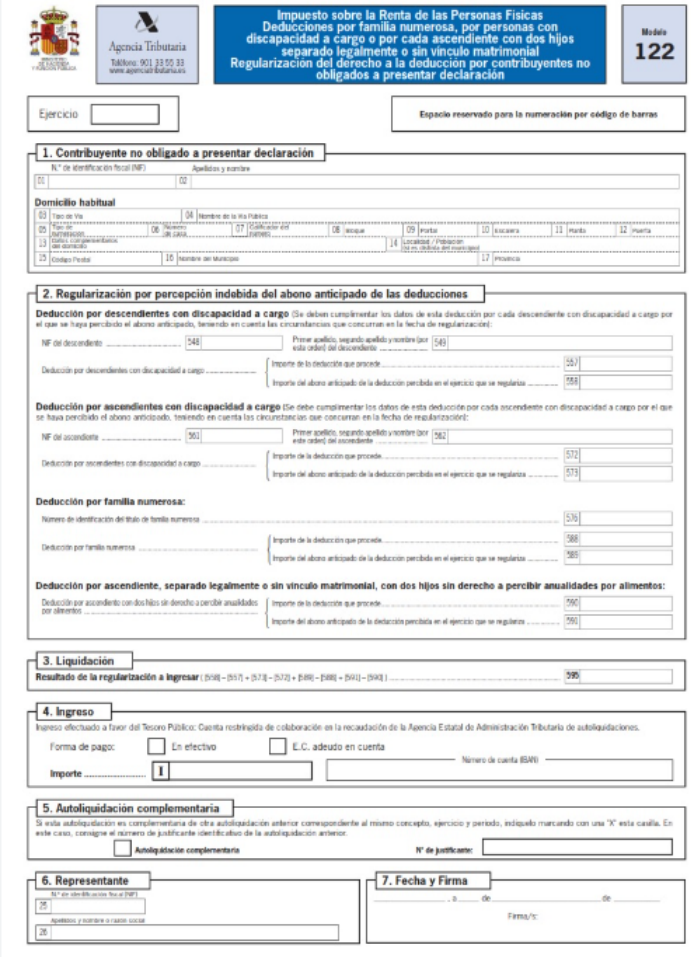

Fig 4.0 - Document exemple (Declaració de renta)

2. Obtenim la string amb tot el contingut del document:

   Impuesto sobre la Renta de las Personas Fisicas Deducciones por familia numerosa. por personas con discapacidad a cargo o p r cada scendierde con dos hijos separado legalment o sin vincuki matrimonial FleplilariZaCID11 del derecho a la doduccon por contribuyentes no obligados a p esentar dedaraización 122 Ensecie repule neme* pene la nornsewlen PM 4(450 ele bino 1 1. Cordr nt? no Mita ? oreeentar lindón a. orcenc"ouon amonInoo Domloalo habitual 1 »liar L?j1 romana. a MON 1¡- smnsin 1 :e in.. s. nen. 11 eteeee 11 2. Roa rvaciin p perwocin'ndobida del abaro andcioado do las doducc' Macean oor descendentes con elscapackled e cene Se ONN NIG NNIN N ?4... N NI4 NON.. My N Deduoclee per aeossanon eso dlecapacIdad? ene enesnwer sms a en sten en cesa wad tonemomer a nao san ess se Momeen, daten pf"1.1-1.0) MON.) b4 ONSINNIN ae caevrie ems *da in denntett NNNIN van' r.ht ore» Iré 'as NNI al N/NI.. netos V /INNIOSINS *Varo: Deducelen pa famas aunara MANN S MIND d le una ININarra Nra. vra. neme ene enea ennuccennensoneetn IV [071 a! sen ens seas ennsmensensespeseennnssus 215 Deducción pee aseendkren seriando legalmenteo sla Minn inatelnwelal. ten des nes al denlo» a perces aelehadáhoei por al-mantos OttInnes sant os kalásna »Cc a wat* hl, INN N y Ny-N sun,. I ..,. Mn NIN 3. Loquedamo RVIaw de inerlerteseltoe ? teireartenhowl -¿fl-mbr-rm-am-rwil 4. Instase nem; dennes s Int Mico Ceses nledéba«latonen es la sennsanec (ata: Kin n*Z.1.1.4Jab Ce 4 4.10)nr. real* pata 111 L- cuelo C.C.adewo encuera lecee ele al» I I I I I 5. Ardoiselacidn complonaflana ? nriwn Onit. INNaJaa N-.. . ? raer cortallo nao r.e.ob. mann mesen cm ars ni. ces tonna si ano es No 4. NNNe M INNINdNet NOnINIII?OPIMIDS RY Inca 6. Reoreseotara 1 la 7. Fecha Y fama

3. Emmagatzem aquest text com a un document de text (.txt) i ho distribuïm en carpetes d'una manera ordenada localment.

4. Repetim de manera automatitzada per a les 700 mostres.

Així doncs l'objectiu i resultat final serà una carpeta pare amb sis carpetes filles cadascuna de les quals emmagatzemarà tots els fitxers en format txt per aquella categoria/label [12]:

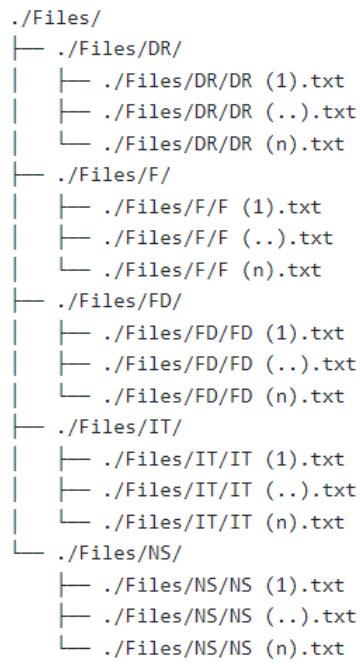

Fig 4.1 - Estructura interna de carpetes i fitxers emprada per elaborar el dataset

8.1.3 Dataset

A partir d'aquí doncs l'objectiu ara seria crear un dataset amb el qual es pugui interactuar, modificar i en definitiva tractar per posteriorment poder elaborar una baseline.

Per elaborar el dataset i poder-lo tractar dins de l'entorn Python, el primer pas ha sigut crear un fitxer el qual contingui totes les 700 mostres en un format que sigui fàcil de treballar i llegir des d'una execució de Python. Aquest fitxer es tracta d'un CSV (Coma separated values), format de fitxer compatible amb molt programari de tractament de dades i amb un ventall d'aplicacions molt ampli.

Aquest fitxer doncs presentarà tres elements per cada fila (mostra):

1. L'etiqueta o Label
2. El nom del fitxer local
3. El contingut en format string d'aquell fitxer

Exemple:

Label	FileName	Text
DR	DR (1).txt	Impuesto sobre la Renta de las Personas Fisicas [...]
...	...	...
DR	DR (n).txt	Impuesto sobre la Renta de las Personas Fisicas [...]
F	F (1).txt	Transportes Genericos SA — 20/12/09 200 eur. [...]
...	...	...

Fig 5.0 - Estructura o resum del contingut del fitxer csv

Tot i que les dues primeres columnes no són rellevants a l'hora de construir i entrenar un model classificador, a l'emmagatzemar totes les mostres de les sis categories diferents en el mateix lloc, s'han inclòs en el fitxer csv per acabar separant els elements X i y del mateix fitxer, X sent el text i y sent l'etiqueta o label. La columna que indica el nom del fitxer és només una manera de comprovar que els fitxers es llegeixen correctament.

Aquest fitxer s'emmagatzemarà localment al mateix directori en el qual executarem el nostre projecte Python per tractar el dataset i posteriorment construir el nostre model classificador.

**Podeu consultar l'annex nº6.1 per veure la implementació que s'ha dut a terme per obtenir el dataset.*

8.2 Models de classificació

Per tal de poder establir un punt de partida bàsic en quant al rendiment del model classificador desenvolupat, el primer pas ha sigut implementar un model classificador Naive Bayes, tot i que el seu ús avui en dia no està gaire estès dins dels models d'aprenentatge automàtic ja que presenta alguns desavantatges, gràcies a la seva simplicitat ens servirà per determinar un punt de partida, a partir d'aquesta 'baseline' tractarem d'anar introduint millores, canvis i altres mètodes per tal d'acabar millorant al màxim el rendiment del nostre model.

8.2.1 Naive Bayes

Els classificadors de Naive Bayes (ingenus) són una col·lecció d'algorismes de classificació basats en el teorema de Bayes. No és un únic algorisme, sinó una família d'algorismes on tots comparteixen un principi comú, és a dir, cada parell de característiques que es classifiquen són independents les unes de les altres.

El conjunt de dades es divideix com en tots els models en dues parts, la matriu de característiques i el vector resposta/objectiu.

La matriu de característiques (X) conté tots els vectors (files) i en conseqüència elements/mostres del conjunt de dades. El vector Resposta/objectiu (y) conté el valor de la variable classe/grup per a cada fila de la matriu de característiques. En el nostre cas l'objecte y serà una de les nostres etiquetes o labels: DR, F, FD, IT, NS o T. Es per això que el dataset el vam estructurar amb les columnes adients per tal de poder extreure fàcilment els dos elements X i y.

Naive Bayes assumeix que cada característica/variable de la mateixa classe fa una contribució independent i igual al resultat, d'aquí el nom de 'Naive' ingenu, es fan suposicions d'independència les quals generalment no són correctes depenent en quines situacions reals.

El teorema de Bayes troba la probabilitat que es produeixi un esdeveniment donada la probabilitat que un altre esdeveniment ja s'hagi produït. El teorema de Bayes es pot resumir en:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- A i B són els esdeveniments.
- $P(A|B)$ és la probabilitat de l'esdeveniment A, donat que l'esdeveniment B és cert (ha passat).
- P(A) és el a priori de A (la probabilitat independent prèvia, és a dir, la probabilitat d'un esdeveniment abans que es vegi l'evidència).
- $P(B|A)$ és la probabilitat de B donat l'esdeveniment A, és a dir, la probabilitat de l'esdeveniment B després de veure l'evidència A.

8.2.1.1 Adaptació per a models classificadors

Així doncs podem adaptar la fórmula original de Naive Bayes a un model classificador constant dels dos elements principals X i y:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

- y és la label o classe.
- X és la matriu de característiques.
- $P(y|X)$ és la probabilitat d'observar la classe y donada la mostra X amb X = (x1,x2,x2, xd).

8.2.1.2 Diferents implementacions

Tot i aquest mateix principi, hi ha diverses implementacions d'aquest mateix algorisme bàsic de Bayes. Aquestes variacions i adaptacions són fruit dels diferents escenaris que ens podem trobar, o en altres paraules, a partir de l'objectiu que vulguem assolir i les dades de les quals disposem, haurem d'escollir un o altre.
Els principals classificadors són:

- Bernoulli
  Basat en el supòsit de presència o absència de paraules en el document és més important que la freqüència. Funciona bé quan el vocabulari és

- Categorical
  Funciona amb probabilitats de característiques, les quals es calculen a partir de la freqüència relativa de cada valor de característiques. Funciona bé quan es tracta de dades categòriques que tenen múltiples valors possibles.

- Complement
  Elaborat per poder treballar amb datasets desequilibrats amb multinomial, corregeix les suposicions de MNB. No és adequat quan l'espai de funcions és complex o el conjunt de dades té un gran nombre de característiques com per exemple tasques de classificació de text on el vocabulari és gran.

- Multinomial
  Funciona construint un model de probabilitat basat en les freqüències de paraules que apareixen en documents de text. Se suposa que les característiques (paraules) són independents entre si, la qual cosa simplifica el càlcul de probabilitats. A partir de les dades de mostra (entrenament) l'algoritme crea un vocabulari de paraules úniques i calcula el recompte de freqüència de cada paraula de cada categoria. Utilitzant aquests recomptes, calcula les probabilitats condicionals de cada paraula de cada categoria. Així doncs a l'hora de classificar, calcularem la probabilitat prèvia d'un document pertanyent a cada categoria, i després calcularem la probabilitat condicional de cada paraula que aparegui en cada categoria i utilitzarem el teorema de Bayes per calcular la probabilitat posterior d'un document pertanyent a cada categoria, i assignarem la categoria o label resultant com la label amb la probabilitat més alta per al document. El classificador multinomial és adequat per a la classificació amb característiques discretes. La distribució multinomial normalment requereix recomptes de característiques en enters. Tanmateix, a la pràctica, els recomptes fraccionaris com ara tf-idf (term frequency times inverse document-frequency) també poden funcionar.

Donades les dades de les quals disposo, tot i que tinc un dataset molt desbalançejat, i l'opció a priori seria implementar un classificador Complement, vaig decidir implementar el model classificador Multinomial, per tres motius:

1. Tenia un gran nombre de característiques per cada mostra i en conseqüència una gran dimensionalitat de vectors de freqüències, ja que disposava de documents molt extensos i molt llargs on la varietat de vocabulari és molt gran, el qual feia que el classificador Complement no fos del tot adequat.

2. El model obtingut a partir del classificador Multinomial serà millorat amb un balanç artificial del dataset (punt 8.2.1.6).

3. Donats tots els factors involucrats en el procés d'obtenció de les dades per part de l'empresa, l'imbalanç de dades o l'obtenció del segon dataset (gran) el vaig patir més endavant durant la implementació d'un tercer model (punt 8.2.2).

8.2.1.3 Multinomial Naive Bayes

Tractament de dades de text

El primer pas doncs serà trobar la millor manera de tractar les nostres mostres X per convertir-les en un input vàlid per un model classificador. Les dades en brut (strings), una seqüència de símbols (és a dir, cadenes) no es poden alimentar directament als models, ja que la majoria d'ells esperen vectors de característiques numèriques amb una mida fixa en lloc dels documents de text en brut amb longitud variable.

Scikit-learn ofereix utilitats per extreure característiques numèriques del contingut del text, com per exemple:

- Tokenizing: Consisteix en el procés de descompondre un text o una frase en unitats individuals, anomenades tokens, que poden ser paraules, frases o símbols, donant un identificador enter per a cada possible token, per exemple, utilitzant espais en blanc i puntuació com a separadors de testimonis.
- Counting: Comptant les ocurrències dels tokens en cada document. De manera que acabarem amb un vector de dimensió 1xn, on n seran el recompte del nombre de característiques o tokens.

Així doncs, el primer pas és realitzar aquest tractament previ de les dades, més concretament de les mostres. L'objectiu sent, obtenir una col·lecció de documents de text en una matriu de recomptes de tokens de manera que cada freqüència de token es tractarà dins del model com una característica. Això es pot aconseguir amb la llibreria de scikit fent us de CountVectorizer. Amb aquest pas obtindríem doncs una matriu de term frequency (TF).

El següent pas seria a partir del TF aconseguir el que se'n diu el TF-IDF (Term Frequency-Inverse Document Frequency). És una mesura estadística que s'utilitza per avaluar la importància d'una paraula per a un document d'un text. La part TF de la fórmula calcula amb quina freqüència apareix un terme en un document, mentre que la part IDF de la fórmula calcula l'únic que és el terme a tot el text.

Així doncs per obtenir el TF-IDF de les nostres mostres, hem de:

1. Calcular la freqüència del terme (TF) per a cada terme del document. Això es fa dividint la freqüència de cada terme pel nombre total de termes del document.
2. Calcular la freqüència inversa del document (IDF) per a cada terme. Això es fa prenent el logaritme de la relació entre el nombre total de documents del text i el nombre de documents que contenen el terme.
3. Calcular la puntuació de TF-IDF per a cada terme del document multiplicant la freqüència del terme per la freqüència inversa del document.

Mitjançant l'ús del TfidfTransformer (també present a la llibreria de Scikit) per transformar una matriu de recomptes de paraules o valors de freqüència de termes en una matriu de valors TF-IDF, podem reduir eficaçment la importància de les paraules que són comunes a molts documents així com articles o preposicions en el nostre cas i augmentar la importància de les paraules que són úniques per a un determinat document o subconjunt de documents com podrien ser els termes: 'DNI', 'Declaración', etc.

8.2.1.4 Exemple

Per poder entendre tots els conceptes i posar-los en pràctica demostraré amb un exemple molt senzill tot el procés que hem de seguir des de que creem el dataset (.csv) fins que obtenim un model classificador fent ús de Multinomial Naive Bayes.

A partir d'un dataset molt limitat, considerem 3 documents d'exemple els quals pertanyen a 2 etiquetes o labels diferents:

- Label 'A':
  - Document 1: La guineu marró salta la gossa blanca.
  - Document 2: La guineu marró és ràpida.
- Label 'B':
  - Document 3: La atmosfera és càlida.

A partir d'aquestes mostres volem classificar un quart document el qual no sabem quina etiqueta o label té: 'La gossa blanca està dormint'. El nostre objectiu doncs serà determinar si aquest document pertany a la Label 'A' o a la Label 'B'.

El primer pas és fer ús del TF i TF-IDF esmentat anteriorment a tots els elements del dataset resultant en la primera matriu TF seria:

	la	guineu	marro	salta	gossa	blanca	es	rapida	atmosfera	calida
Document 1	2	1	1	1	1	1	0	0	0	0
Document 2	1	1	1	0	0	0	1	1	0	0
Document 3	1	0	0	0	0	0	1	0	1	1

Tot i que en aquesta matriu les paraules 'la' i 'es' estan presents, posteriorment s'ignoraran perquè no tendeixen a aportar massa informació al model, així com en angles podrien ser les paraules 'the' o 'to'.

Seguidament, haurem d'obtenir la matriu IDF dels mateixos elements. Per aconseguir això, haurem de primer calcular la freqüència de document per cada paraula:

- df("guineu") = 2
- df("marro") = 2
- df("salta") = 1
- df("gossa") = 1
- df("blanca") = 1
- df("rapida") = 1
- df("atmosfera") = 1
- df(çalida") = 1

Tot seguit podrem calcular els valors IDF per a cada paraula fent ús de la següent fórmula:

$$idf(t) = log(n/df(t))$$

- t és la paraula
- n és el nombre total de documents al dataset
- df(t) és la freqüència de document per paraula

Obtindrem:

(exemple) idf("guineu") = $log(3/2) = 0.4055$

	guineu	marro	salta	gossa	blanca	rapida	atmosfera	calida
IDF	0.405	0.405	1.098	1.098	1.098	1.098	1.098	1.098

Finalment, obtindrem el valor TF-IDF a partir de la fórmula:

$$tf - idf(t, d) = tf(t, d) * log(n/df(t))$$

*on n és el nombre total de documents del conjunt de documents i df(t) és la freqüència del document de t; la freqüència del document és el nombre de documents del conjunt de documents que contenen el terme t.

De manera que necessitem calcular els valors $tf(t, d)$, la matriu resultant és:

*exemple:
$tf("guineu", Document1) = 1/5 = 0.2$
La matriu tf(t,d)seria la següent:

	guineu	marro	salta	gossa	blanca	rapida	atmosfera	calida
Document 1	0.2	0.2	0.2	0.2	0.2	0	0	0
Document 2	0.2	0.2	0	0	0	0.33	0	0
Document 3	0	0	0	0	0	0	0.5	0.5

De manera que la matriu final $tf - idf(t, d)$ ens quedaria així:
*exemple:
$tf - idf("guineu", Document1) = 0.2 * 0.405 = 0.081$

La matriu $tf - idf$ final seria la següent:

	guineu	marro	salta	gossa	blanca	rapida	atmosfera	calida
Document 1	0.081	0.081	0.219	0.219	0.219	0.	0	0
Document 2	0.081	0.081	0	0	0	0.362	0	0
Document 3	0	0	0	0	0	0	0.549	0.549

Finalment, doncs només faltaria processar el nou document 'La gossa blanca està dormint':

El vector $tf(Document4)$:

	guineu	marro	salta	gossa	blanca	rapida	atmosfera	calida
Document 4	0	0	0	0.25	0.25	0	0	0

La matriu $tf - idf$ final per el Document 4 seria la següent:

	guineu	marro	salta	gossa	blanca	rapida	atmosfera	calida
Document 4	0	0	0	0.274	0.274	0.	0	0

Sumem els valors 0.274 + 0.274 i n'obtenim 0.549. Aquesta serà la puntuació aconseguida de la primera label. Si intentéssim fer el mateix amb la segona label obtindrem una puntuació de 0. Per tant, afirmarem que aquest nou document pertany a la Label 'A'.

8.2.1.5 Baseline

A partir de tots aquests principis, s'elabora un model base per tenir una puntuació en format de precisió global del model. Aquesta baseline ens servirà per introduir millores en el model o comparar amb futurs models. El model elaborat final aconseguit amb el dataset inicial (300 mostres) ha sigut capaç d'obtenir una precisió global d'entorn al 74%. Aquests resultats han estat assolits dividint el dataset sencer entre training i testing en un rati 70/30. Referències: [3], [8] i [16]
Per mostrar més gràficament aquests resultats per categories faré ús de les matrius de confusió:

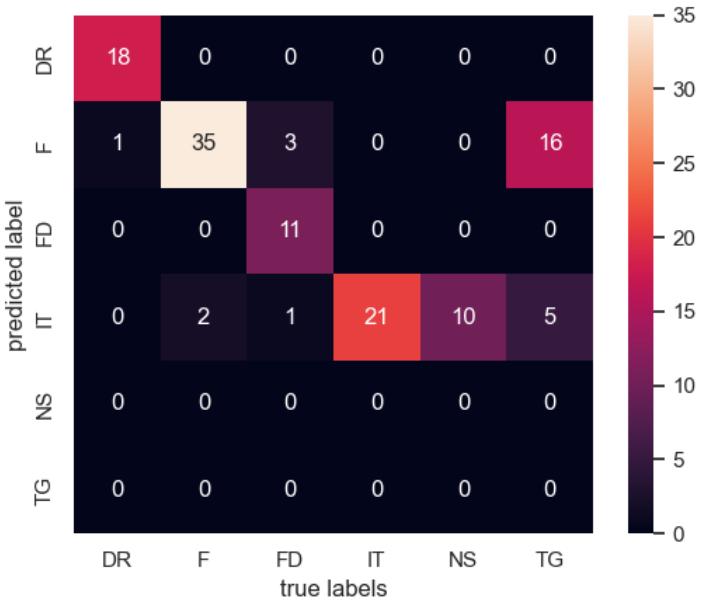

Fig 6.0 - Matriu de confusió [5] resultant del primer model amb el dataset petit

Es pot observar a simple vista on tenim problemes de predicció. A primera instància no hem predit cap document com a 'NS' o com a 'TG', això podria ser parcialment degut a la falta de documents en el primer dataset, tot i així, ho hem fet prou bé en altres etiquetes com per exemple 'DR' la qual presenta una precisió de quasi el 100%. Una altra etiqueta amb bons resultats és 'FD' amb també una precisió d'entorn al 90%.

Els problemes obvis a priori doncs irradien de les etiquetes 'F', 'IT', 'NS' i 'TG'. Aquestes dues, al tractar de documents molt similars seran documents molt difícils de distingir entre ells.

L'objectiu doncs a partir d'aquí és augmentar la precisió global del model, per fer això, ens podem centrar en la precisió parcial per cada etiqueta. Per millorar això, s'introduiran tècniques d'aprenentatge automàtic i de tractament de dades en el mateix model i/o dataset.

Els resultats de precisió per cada etiqueta són els següents:

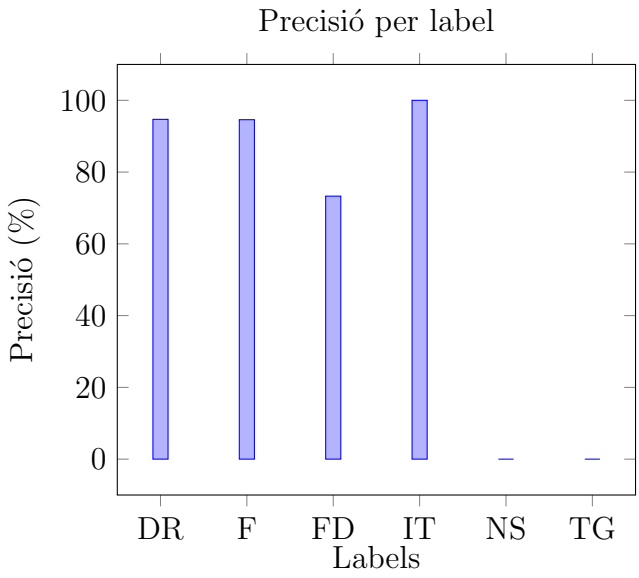

Fig 6.1 - Diagrama representant la precisió per etiqueta del model

Així doncs, podem confirmar que els problemes de precisió ens vénen donats per les etiquetes NS i TG, que es classifiquen erròniament com a les etiquetes IT, i F respectivament. De manera que el model està confonent notes salarials amb informes de taxació i Tickets de gasto com a factures, dos parelles de documents que són molt similars en quant a contingut.

La primera opció que tenim sense tocar el model, és aconseguir més dades. Que ara mateix no tenim de manera que haurem de trobar una manera de 'confeccionar-les'.

8.2.1.6 Dades sintètiques

La primera opció que he trobat per millorar el model sense aconseguir dades reals que augmentin el dataset, és la generació de dades sintètiques.
Les dades sintètiques ofereixen un augment del rendiment de l'aprenentatge automàtic de dues maneres: simplement proporcionant més dades per a la formació del model i utilitzant més mostres sintètiques de classes minoritàries que les disponibles. El rendiment dels models d'aprenentatge automàtic pot augmentar fins a un 15%, depenent del conjunt de dades i del model exactes.

En el meu cas, partint del dataset inicial de 378 mostres, tenia una mancança de mostres de les etiquetes DR, FD, NS i TG, dues de les quals són les que m'han donat problemes fins ara pel que fa a precisió.

A més de resoldre el problema de l'escassetat de dades, la principal aplicació de les dades sintètiques és la protecció de la privadesa de les dades. Les bases de dades personals d'empreses, governs i institucions contenen informació d'identificació personal o altres atributs sensibles com noms complets, números de compte bancaris i documents d'identitat és per això que l'ús de dades sintètiques permet anonimitzar les dades autèntiques sense renunciar a una quantitat de dades suficient per a poder dur a terme una anàlisi, l'elaboració d'un model o la creació d'un programa de gestió de dades.

Després de molta investigació sobre les dades sintètiques, em vaig topar amb dues llibreries, la primera anomenada nlpaug i la segona Huggingface. Nlpgaug és una biblioteca per augmentar el text en experiments d'aprenentatge automàtic. L'objectiu és millorar el rendiment del model generant dades textuals.

Aquestes llibreries com moltes altres, funcionen també amb models d'aprenentatge automàtic, però, ja que no tenia experiència en dades sintètiques, em vaig preocupar d'elaborar una funció que era capaç de combinar de manera mínimament aleatòria diverses funcionalitats de la llibreria. El motiu principal era que no volia alterar amb un patró molt clar les dades. Un exemple seria dividir la mostra en 2 parts i descartar la segona part, generant un text aleatori per complementar la meitat que falta. Aquesta tècnica podria ser versàtil, però no podia fer exactament el mateix per totes les mostres, ja que el que acabaria fent és crear una nova subcategoria de mostres les quals serien igual de similars entre si però no suficientment similars amb les originals.

Així doncs, em vaig limitar a fer un ús combinat d'aquestes tres funcions que ofereix nlpaug depenent del tipus de dades amb les quals estava tractant, referències: [2],[6] i [15]:

- RandomSentAug
  Quan cridem a aquesta funció, li hem de proporcionar una string com a entrada, seleccionarà aleatòriament un nombre determinat de paraules de la frase per substituir-les per sinònims o paraules similars. El nombre de paraules que es substitueixen ve determinat pel paràmetre *augp*, que representa la probabilitat que una paraula sigui substituïda. La funció utilitza diverses tècniques de processament del llenguatge natural per trobar sinònims o paraules similars per substituir les paraules seleccionades a la frase. Aquestes tècniques inclouen incrustacions de paraules, wordnet i altres models de llenguatge.

- ContextualWordEmbsForSentenceAug
  Es pot utilitzar per augmentar una frase cridant al seu mètode d'augment. Aquest mètode pren una frase com a entrada i substitueix algunes de les seves paraules per les seves incrustacions contextualitzades. El nombre de paraules que es substitueixen ve determinat pel paràmetre *augp*, que representa la probabilitat que una paraula sigui substituïda. Admet una varietat de models, com ara BERT, GPT-2, XLNet i RoBERTa, entre d'altres.

- Text-generation pipeline (HuggingFace)
  La pipeline fa servir un model d'idioma entrenat prèviament per Hugging Face, com ara GPT-2 o BERT, per generar text. El pipeline inclou passos de preprocessament i postprocessament de text, així com el pas de modelització de llenguatge. Amb aquesta pipeline, podem generar text cridant al seu mètode de generació i passant un missatge o text d'entrada (strings).

**Totes aquestes funcionalitats han estat implementades al fitxer comprés a l'annex nº6.1.*

Fent ús de totes aquestes tècniques combinades, vaig aconseguir generar un nou dataset. Aquest nou dataset multiplicava per dos les dades que tenia per poder entrenar el model, de manera que ara tenia un total de 733 mostres, la meitat de les quals eren dades sintètiques. Tot i això, vaig voler tornar a generar el dataset i altra vegada entrenar el mateix model utilitzat en el baseline. Els resultats van ser molt satisfactoris (una millora en la precisió global del model de més del 10%)

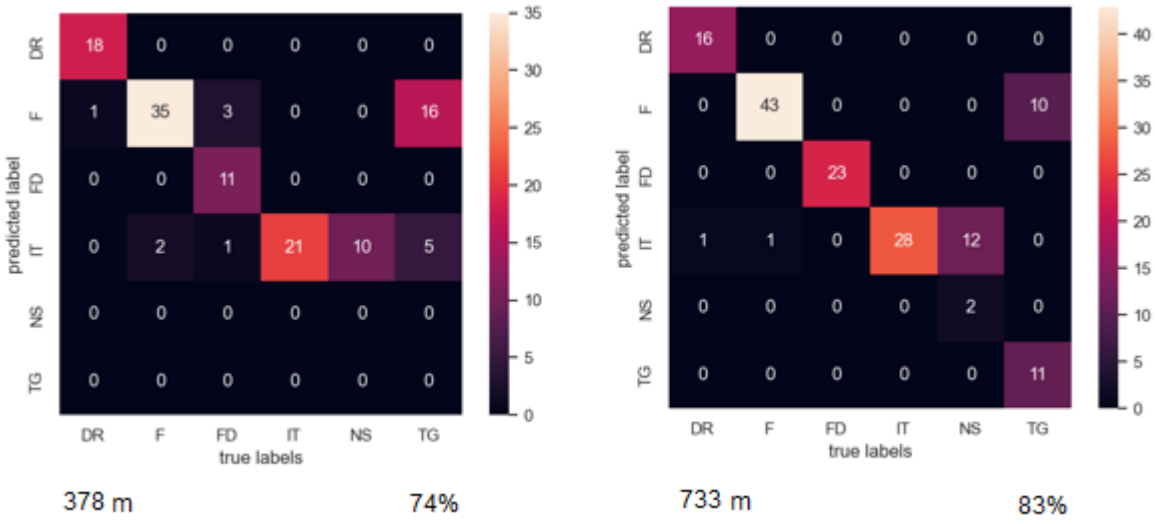

Fig 7.0 - Matriu de confusió resultant del primer model amb el dataset petit (esquerra) i matriu de confusió resultant del primer model amb el dataset gran (dreta).

Només amb aquesta implementació artificial de dades, he obtingut augmentar la precisió del model a quasi el 88%. Aquesta millora, però no és representativa del rendiment que tindria el model al món real, ja que lo ideal seria augmentar el dataset amb dades reals, tot i així, el potencial de millora encara és gran. A més la precisió per categoria també ha augmentat.

Una comparació en la precisió per etiqueta respecte al model anterior:

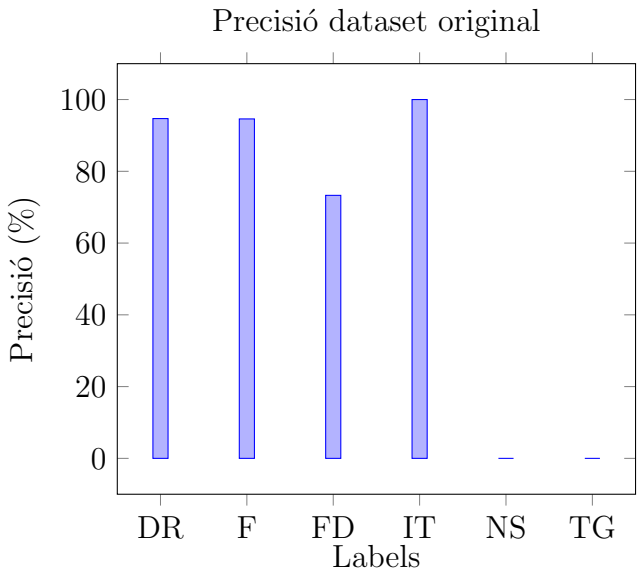

Fig 7.1 - Diagrama representant la precisió per etiqueta del model amb les dades originals

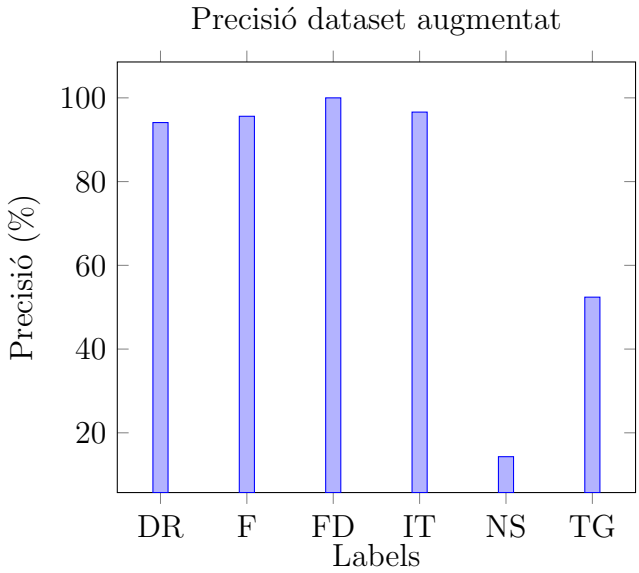

Fig 7.2 - Diagrama representant la precisió per etiqueta del model amb les noves dades sintètiques

Així doncs, sense dependre de dades reals per part de l'empresa i sense alterar el model de baseline ja hem pogut augmentar en un percentatge significatiu la precisió. A més hem pogut passar d'una precisió d'entorn al 70% a un 100% per l'etiqueta FD. La precisió d'una de les etiquetes que ens estava donant problemes prèviament ara ja passa del 50%. Tot i això encara hi ha molt espai per millora, ja que l'etiqueta de NS se'ns resisteix.

Poc després de desenvolupar aquest últim model, vaig tenir la sort de rebre més mostres reals de l'empresa. Augmentat les mostres de diverses etiquetes. Així doncs, el primer que vaig fer és comparar el rendiment de les dades reals amb les dades augmentades. Tenint en compte que patia d'un desequilibri de dades, tampoc volia cometre l'error d'augmentar de manera artificial només les dades d'una etiqueta concreta, ja que estaria comprometen la integritat de l'etiqueta si aquesta ara estaria formada en un 90% de dades sintètiques de manera que vaig augmentar les dades en factors amb un threshold per marcar la diferència. Els resultats comparatius són bastant interessants. A la pròxima figura represento 4 models diferents:

1. Model amb dataset petit (371 mostres)
2. Model amb dataset petit augmentat (733 mostres)
3. Model amb dataset gran (701 mostres)
4. Model amb dataset gran augmentat (1380 mostres)

Els resultats són els següents:

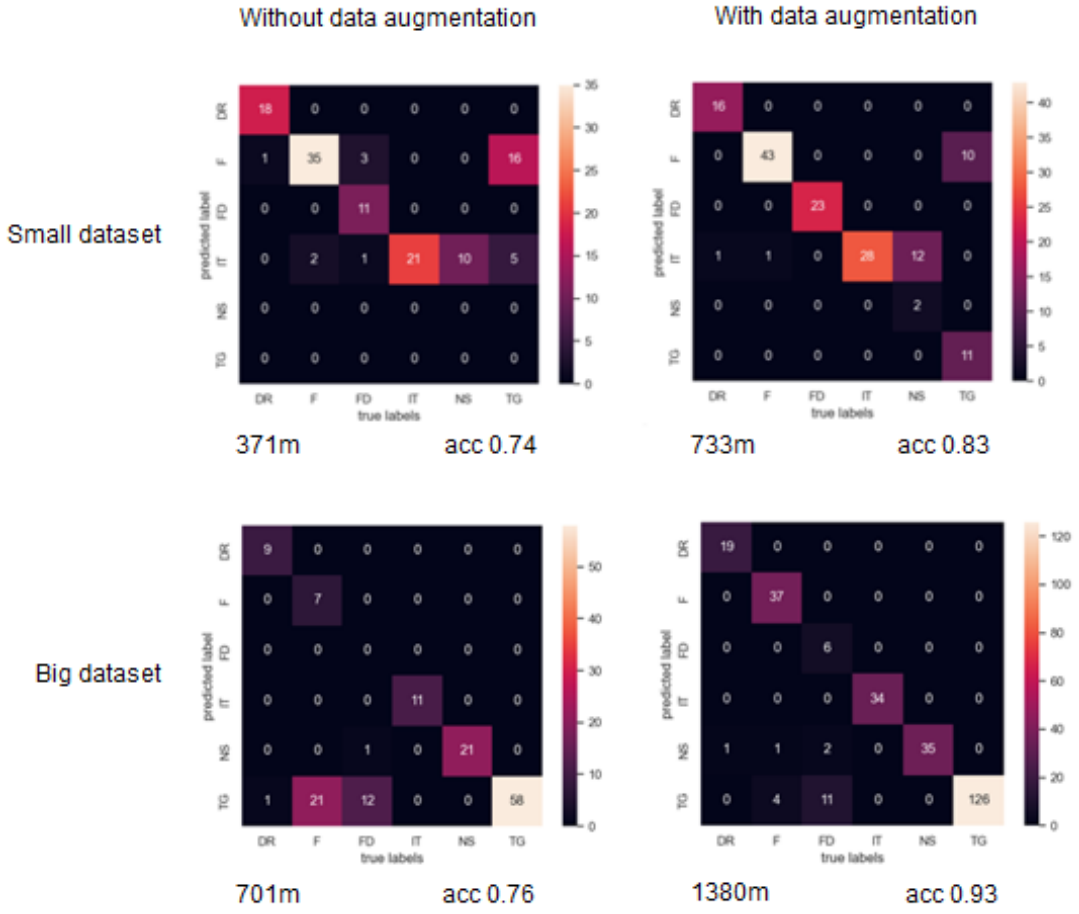

Fig 7.3 - Matriu de confusió resultant dels quatre models

Tot i que hem obtingut uns resultats molt bons, cal destacar que l'aportació de dades reals no va ser suficient per suplir les necessitats del model. Arribant a la conclusió que les dades sintètiques han jugat un millor paper que les dades reals. El motiu principal d'aquest resultat és que les noves dades facilitades per l'empresa principalment pertanyien a dues categories, de manera que l'aportació global al model ha sigut menor. Tot i així, en el model amb el nou dataset de 701 mostres reals ara té una precisió de les dues etiquetes TG i NS de quasi el 100% encara que la precisió global no ha representat una gran millora. Centrant-nos en l'últim model. Tot i que la precisió global sigui molt bona, encara patim per una altra etiqueta, FD. Representació gràfica del problema:

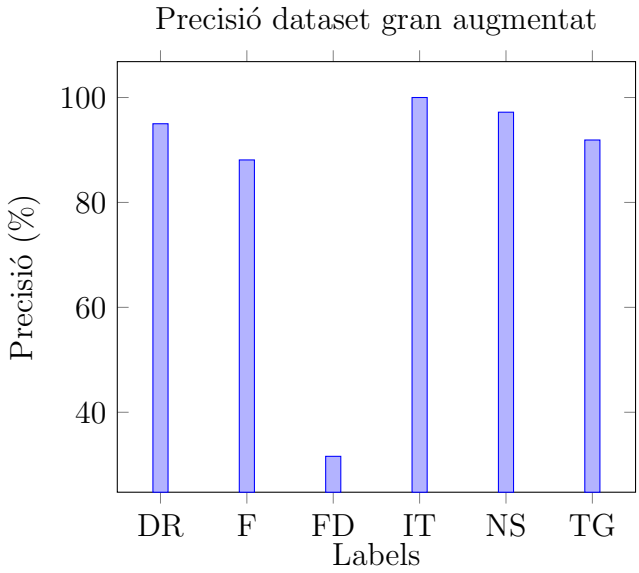

Fig 7.4 - Diagrama representant la precisió per etiqueta del model gran amb dades sintètiques

Com podem observar, encara que el 94% a priori ens sembli un resultat espectacular respecte l'últim 83%, hem de recordar que les gràfiques de precisió per etiqueta continuen sent molt semblants, ara l'etiqueta que pateix més és una altra, però la variància entre precisions d'etiquetes no ha disminuït. El següent pas a partir d'aquí serà confeccionar un nou model, l'objectiu del qual sigui que no estigui tan condicionat pel nombre de dades que rep d'input i que tingui un rang de millora major, com per exemple les xarxes neuronals.

8.2.2 Aprenentatge profund

L'aprenentatge profund és un conjunt de tècniques/algoritmes dins dels models d'aprenentatge automàtic els quals la majoria fan ús de xarxes neuronals, aquest sistema utilitza nodes o neurones interconnectades en una estructura de capa que s'inspira en el cervell humà. Aquestes connexions les quals es determinaran com a pesos, indicaran 'excitació' si són positiu o 'inhibició' si són negatiu, a la vegada aquests pesos s'aniran adaptant a partir dels errors i s'aniran ajustant contínuament a partir d'un dataset d'input. D'aquesta manera, amb les xarxes neuronals podem resoldre problemes complicats, com ara la realització de resums de documents, classificació, reconeixement d'imatges, etc., amb una major precisió.

Aquesta tècnica està molt usada en el món del processament natural del llenguatge (NLP), aquesta és la capacitat de processar el text natural creat pels humans. Les xarxes neuronals ajuden a obtenir informació i significat a partir de dades i documents de text. Els NLPs estan present en moles aplicacions com per exemple:

- Xats i agents virtuals automatitzats
- Organització automàtica i classificació de dades escrites
- Anàlisi de correus electrònics i formularis
- Indexació de frases clau que indiquen el 'sentiment' d'un text, com ara comentaris positius i negatius a les xarxes socials
- Resum de documents i la producció d'articles per a un tema específic

Referències genèriques: [1], [4], [9] i [13]

8.2.2.1 Preprocessament

Similarment, al model anterior de Multinomial Naive Bayes, abans de poder construir un model i poder-lo entrenar necessitem fer un tractament previ de les dades:

- Tokenizer
  Explicat en el punt 8.2.1.3
  El primer pas doncs, és crear un objecte tokenizer, en el nostre cas el factor limitant seran el nombre de paraules a considerar, lo idíl·lic seria un valor el màxim alt possible, però aquest factor anirà en detriment del rendiment i el temps emprat a l'hora d'entrenar el model. El valor utilitat doncs es 20000, fent referència a les 20000 primeres paraules (tokens) més rellevants o comunes que considerarem per cada document.

- Padding
  Consisteix en realitzar l'acció de "padding o pad"en anglès per estandarditzar totes les mostres, aquest procés determinarà la dimensió de l'input d'entrada de les mostres usades en el model. La variable '$max - len$' determinarà la llargada de l'input pel model, en el nostre cas 1000. En cas que una mostra no sigui suficientment gran, omplirà de zeros les variables que falten i en cas que sigui massa gran, aquesta es tallarà. Tenint en compte que un altre cop, lo ideal seria configurar aquest valor una mica més elevat, ens veiem limitats pel temps d'entrenament.

8.2.2.2 Entrenament

Per entrenar una xarxa neuronal de classificació, primer hem de determinar una arquitectura interna de capes, necessitarem doncs una capa d'entrada que rebi les dades de text vectoritzades, una o més capes ocultes que processen les dades i aprenen a reconèixer els patrons i una capa de sortida que produeix les etiquetes previstes, en el nostre cas 6 (DR, F, FD, IT, NS i TG).

La capa de sortida d'una xarxa neuronal de classificació és normalment un conjunt de funcions d'activació softmax, aquesta funció produirà un conjunt de probabilitats per a cada etiqueta, la suma de les probabilitats serà 1 i l'etiqueta amb la probabilitat més alta serà l'output final del model.

Durant l'entrenament, la xarxa neuronal ajusta els seus pesos en funció de l'error entre les etiquetes previstes i les veritables etiquetes obtenies de les dades d'entrenament. Aquest procés, anomenat backpropagation, utilitza descendència de gradient per minimitzar l'error entre les etiquetes previstes i veritables. La xarxa neuronal continuarà ajustant els seus pesos i millorant la seva precisió a mesura que s'exposa a més dades de training.

Un cop formada la xarxa neuronal, es pot usar per predir l'etiqueta de noves mostres. El text d'entrada és tokenizta primer mitjançant la mateixa tècnica que s'utilitza durant l'entrenament, i després s'introdueix al model. La capa de sortida doncs expedirà una de les sis etiquetes presents, aquella tingui el valor més alt.

8.2.2.3 Arquitectura i capes

El model desenvolupat i obtingut ha estat resultat de moltes iteracions, prova i error fins a obtenir un bon resultat en termes de precisió. Aquest model fa ús de capes molt usades dins de models classificadors i l'estructura d'aquestes és bastant simple.
Com s'ha explicat anteriorment, el primer pas per construir el model és tractar les dades per poder usar-les dins del model, ja que parlem d'elements de text (strings) haurem de tractar les dades de la mateixa manera que ho vam fer al model anterior. Un cop hem 'tokenized' les dades, podrem construir el nostre model amb una estructura de capes:

- Capes involucrades
  - Embedding
    Aquesta capa converteix cada paraula del text d'entrada en un vector representatiu. En el nostre cas, el paràmetre $input - dim$ determina la mida del 'diccionari' del model, en el nostre cas està configurat per considerar les 10.000 paraules més comunes. El paràmetre $output - dim$, configura la mida dels vectors a 128 de manera que cada paraula estarà representada en un vector de dimensió 1x128 i $input - lenght$ determina la longitud de les seqüències d'entrada, en el nostre cas 1000, aquest valor vindrà doncs determinat per la mida d'input que hàgim establert en el pas de padding.
  - Bidirectional LSTM
    Consisteix en una capa formada per cèl·lules cadascuna de les quals té tres portes: una porta d'entrada, una porta d'oblit i una porta de sortida que permeten a la cèl·lula processar i controlar selectivament el flux d'informació. A cada pas d'aprenentatge del model, el vector d'entrada i el vector d'estat anterior es passen per un conjunt de funcions d'activació Sigmoid i TANH per calcular els valors d'aquestes portes. A continuació, la cèl·lula LSTM calcula un nou estat de cèl·lula, que es transmet al següent pas d'aquesta manera iterativa, la capa LSTM és capaç de recordar o oblidar selectivament la informació dels passos de temps anteriors, permetent-li processar seqüències llargues de dades de manera més eficaç, utilitzant a la capa LSTM bidireccional significa que aquest procés s'esdevindrà en ambdues direccions, de punta a punta a la vegada [7].
  - Dropout
    Aquesta capa ens permet descartar un percentatge d'input de manera aleatòria, forçant a la xarxa a aprendre característiques més robustes i genèriques, ja que les neurones restants hauran de forçar-se per compensar els valors que falten. El droput s'utilitza en vàries capes per tal d'evitar l'overfiting. El percentatge emprat en el nostre cas és el 20%.

– Dense (relu)
  Una capa dense és una capa completament connectada de manera que totes les neurones estan connectades a totes les neurones de la capa anterior. Cada neurona rep una entrada de totes les neurones de la capa anterior i envia la seva sortida a totes les neurones de la capa següent. El factor manipulable i limitant és el nombre de neurones i el tipus d'activació, en el nostre cas, hem fet ús de dues capes Dense amb activació relu, la primera amb una mida de 128 i la següent amb una mida de 64. L'activació relu ens serveix per determinar si l'input és positiu o negatiu [10].

– Dense (softmax)
  Finalment, l'output del model serà una capa Dense que utilitza activació softmax, aquesta activació ens permet distribuir l'output final a 6 categories diferents. La suma de la probabilitat de totes les categories doncs serà 1. L'output final al classificar doncs serà un conjunt de 6 percentatges representant cada categoria, i escollint doncs el percentatge més alt representant la categoria 'most-likely'.

- Estructura

Per visualitzar models d'aprenentatge automatic, concretament models que utilitzin xarxes neuronals, he utilitzat una eina específica per aquest fi, aquesta s'anomena Netron [11], aquest programari et permet importar models exportats i visualitzar-los amb tot detall. Podeu veure una imatge més detallada a l'annex nº3.1. L'output simplificat del meu model:

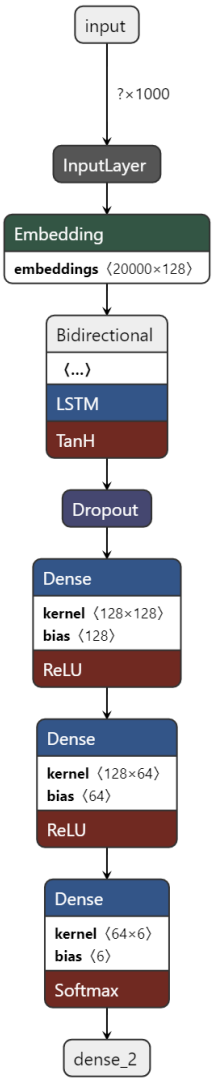

Fig 8.0 - Esquema de l'arquitectura interna del model

8.3 Proves i resultats

Els resultats obtinguts fent ús d'aquest model han estat els esperats, tenint en compte que a més de tenir un model més capaç i adaptable dins d'aquest problema, estic fent ús del dataset gran (700 mostres) juntament amb les llibreries per generar mostres sintètiques extra, acumulant un total de 1380 mostres, les quals es divideixen en un 80% d'entrematant i l'altre 20% de testing. La precisió global final del model és d'entorn al 94% i l'exactitud és aproximadament del 98%, encara que no hem arribat al 100% s'ha de tenir en compte la millor notícia que és que la precisió per cada categoria ha augmentat:

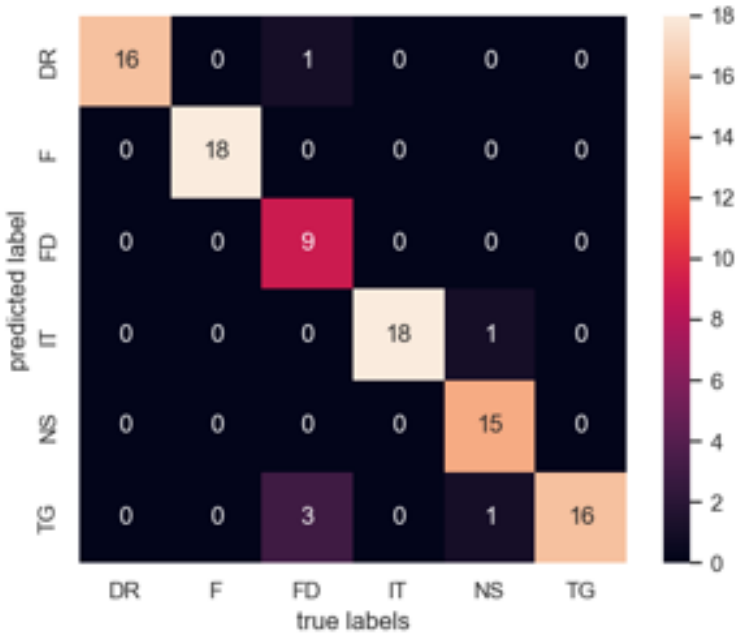

Fig 9.0 - Matriu de confusió del model de xarxes neuronals

La conclusió que podem treure és que tenim un model força precís. L'única label que pateix lleugerament és la de FD, potser quelcom comprensible, ja que al final estem confonent 'fotocòpies de DNI' amb 'Tickets de gasto', dos documents molt similars pel que fa a mida i inclús format de cara al model.

L'únic punt negatiu a destacar és l'etiqueta de FD, encara que la variància entre les precisions per categoria ha disminuït concretament ara mateix tenim una variància entre precisió per categoria de 0.60, la variància entre recall per categoria és de 1.56. Tot i això podem considerar que hem pogut desenvolupar un model robust a partir de les dades limitades i desequilibrades de les quals hem partit i que hem pogut desenvolupar un model molt superior respecte la baseline amb la que vam començar.

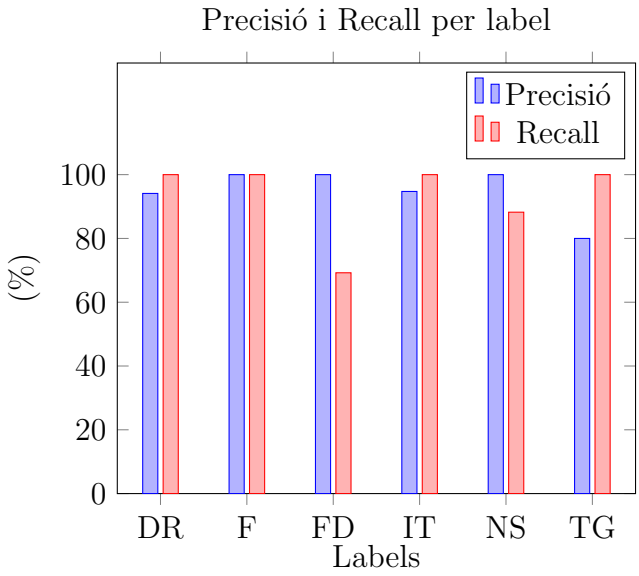

Fig 9.1 - Diagrama representant la precisió i recall per etiqueta del model de xarxes neuronals amb el dataset gran i augmentat.

8.3.0.1 Comparació amb NB

Farem doncs una recapitulació de tots els models usats així com el rendiment resultant entre ells amb tots els diferents datasets.

L'objectiu sent veure d'una manera gràfica la millora dels model de manera progressiva. Nomenclatura dels models:

- MNB S: Multinomial Naive Bayes amb Small Dataset
- MNB B: Multinomial Naive Bayes amb Big Dataset
- MNB SS: Multinomial Naive Bayes amb Small Dataset amb Synthetic data
- MNB BS: Multinomial Naive Bayes amb Big Dataset amb Synthetic data
- NN BS: Neural Networws amb Big Dataset amb Synthetic data

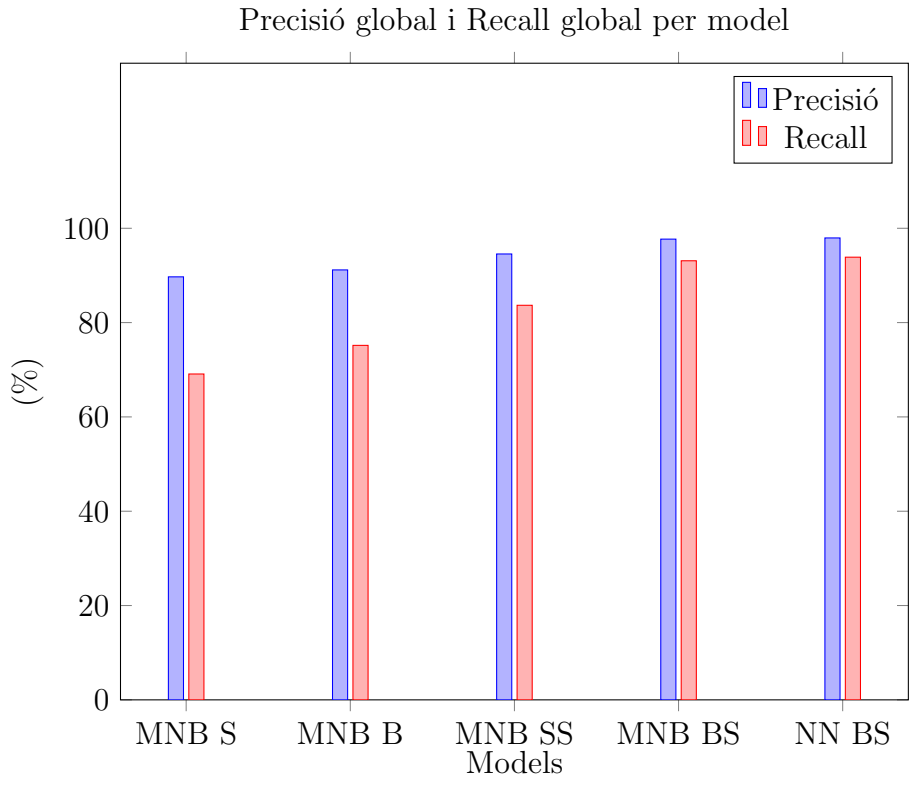

Fig 10.0 - Diagrama representant la precisió i recall per cada model

És bastant aparent la millora substancial partint de la baseline, encara que no ho puguem veure, també hi ha una millora del model respecte als models MNB BS i NN BS, aquesta es pot apreciar més si representem la variància de la precisió i recall per cada label de cada model:

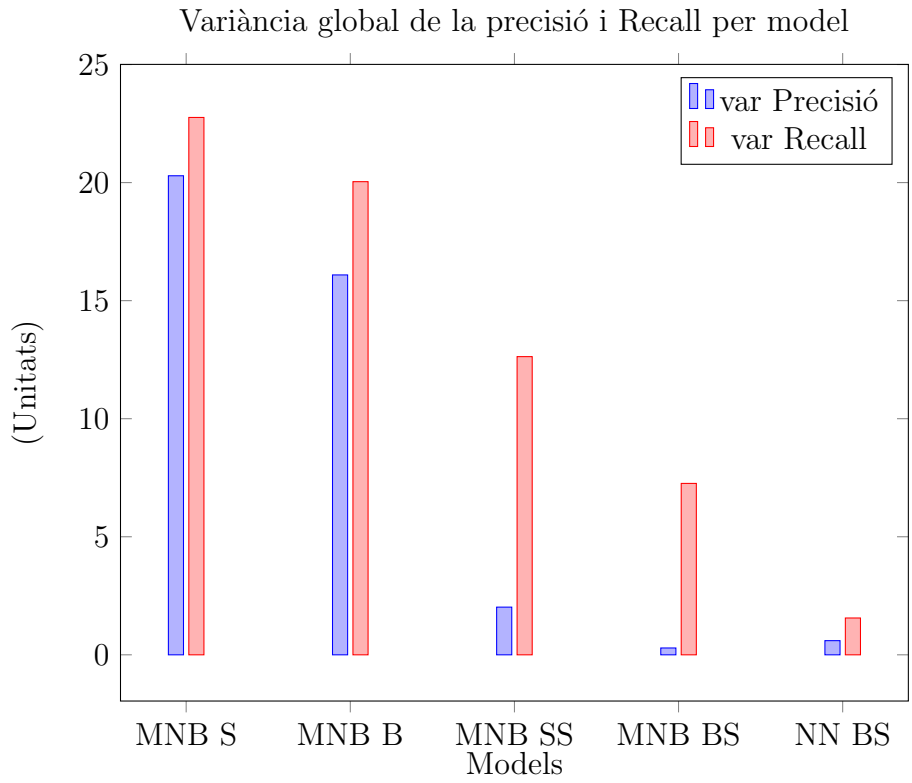

Fig 10.1 - Diagrama representant la variància de la precisió i recall per cada model

Amb la variància d'aquestes mètriques per categoria podem concloure doncs que la millora ha sigut pràcticament lineal. Així doncs aquest model serà l'escollit i utilitzat dins del framework per poder classificar els documents que rebem per correu d'entrada. El següent pas serà poder usar aquest model d'una manera portable dins de qualsevol entorn.

**Podeu consultar totes les matrius de confusió de tots els models així com càlculs addicionals en l'annex nº4.1.*

8.4 Portabilitat

Tenint el model final elaborat, l'objectiu seria poder exportar aquest model d'una manera simple i poder importar aquest model dins d'un altre entorn, poder executar el model i obtenir els mateixos resultats que hem obtingut en l'entorn inicial. La motivació darrere aquest objectiu és per així acabar tractant el model com a un objecte o fitxer el qual podem executar segons ho necessitem dins de la nostra aplicació.

8.4.1 Exportació i importació del model

Per aconseguir això hem de recórrer a l'ús del mètode que de Python natiu que s'utilitza per serialitzar i desar un objecte Python en un fitxer així com recuperar les dades serialitzades i reconstruir el mateix objecte. Tot i que el format de fitxer amb el qual emmagatzema'm els fitxers i els llegirem de disc és indiferent, ja que el fitxer en si no serà llegible, farem servir una extensió de fitxer "pkl"per poder identificar models serialitzats a un sistema de fitxers entre altres objectes. Els dos mètodes emprats per aconseguir això seran:

- dump: Serialitza l'objecte Python en qüestió i l'emmagatzema en un fitxer local.
- load: A partir del fitxer local serialitzat, genera l'objecte Python original.

Ja que les nostres dades d'entrada dins del codi acabaran sent strings, necessitarem tractar també aquestes dades d'input exactament de la mateixa manera que hem tractat les dades usades per entrenar el model, és per això que juntament amb el model, també exportarem el nostre vectorizer, per poder tractar les dades abans de passar-les pel model i així classificar el nostre input.

El procés resultant són dos fitxers locals emmagatzemats en disc, el model ocupant aproximadament 30Mb i el vectorizer ocupant aproximadament 6Mb. El següent pas doncs serà llegir aquests fitxers i recuperar el model i el vectorizer dins de l'entorn Python per poder processar els documents que rebem d'input en el nostre framework.

**Tot el desenvolupament de tots els models presents en el treball així com el tractament de dades i l'exportació del model es troba a l'annex nº6.1.*

9 El framework

La idea del framework és poder gaudir d'una interfície modular ja sigui de codi o gràfica per tal que l'usuari final pugui posar en pràctica un model que disposi, en el nostre cas aquest model serà l'últim desenvolupat. Així doncs l'usuari després de desplegar el servei, podrà rebre correus electrònics a la direcció de correu del servidor i aquests seran processats segons el criteri i la solució que vulgui implementar l'usuari. L'objectiu sempre sent el de classificació gràcies al model classificador que sempre se situarà en el centre del framework i l'output cap a altres direccións. En tots els casos, necessitaríem doncs un compte de correu electrònic 'servidor' per gestionar els processos, aquest per exemple podria ser 'info@empresa.com' aquesta direcció doncs gestionaria tot l'input per part de tercers i l'ouptut cap a les direccions designades.

Un exemple d'implementació o use case seria un separador de correus spam i no spam dins d'una empresa. La interfície permetria a l'usuari desplegar el servei amb un model classificador de Spam/No spam i podria reenviar per exemple els correus marcats amb una label de "No spam"als destinataris originals i els correus marcats amb la label de "spam"esborrar-los.

Un altre use case, seria la possibilitat d'automatitzar tot el procés de classificació intern d'una empresa, assumint per exemple que tenim un secretari que s'encarrega de fer arribar els correus electrònics als departaments on toquen, i així distribuir correctament el flux de correus interns que té l'empresa, el framework es podria desplegar i desenvolupar aquesta tasca interna de classificació de contingut de correus, l'objectiu sent que els correus electrònics arribin als departaments corresponents ex: Vendes, Comptabilitat, Recursos Humans, etc, a partir del contingut d'aquests. Aquesta solució és la que implementaré com a exemple de funcionament fent ús del model desenvolupat anteriorment.

9.1 Funcionament

Aquest framework desplegarà un servei de correu electrònic el qual estarà escoltant a l'input de la bústia de la direcció que haguem utilitzat com a servidor i processarà tots els missatges que entrin. Aquests missatges es filtraran per diversos factors limitants els quals podrà escollir l'usuari, com per exemple una llista blanca d'adreces de correu electrònic per processar.

Un cop processat el missatge, aquest es segmentarà en diversos objectes: la direcció de correu d'origen, l'assumpte del correu electrònic, el cos, i els arxius adjunts.

Un cop disposem dels arxius adjunts, processarem cada fitxer ja sigui una imatge, un document de text o un pdf per extreure el text d'aquests documents. Això es farà de manera seqüencial de manera que si tenim dos fitxers adjunts el contingut del segon fitxer s'ajuntarà amb el contingut del primer fitxer. L'objectiu sent obtenir una sola string.

A l'obtenir la string representant dels fitxers adjunts, aquesta la processarem fent us del model classificador que hàgim escollit, en el nostre cas el model desenvolupat al punt anterior.

Una vegada aconseguit l'output del model en format string representant les labels que nosaltres hàgim configurat, recopilarem una altra vegada tots els fragments que componien el correu electrònic inicial, inclosos els fitxers adjunts i l'enviarem a una direcció de correu electrònic configurada, o al mateix remitent indicant l'output del model classificador a l'assumpte del nou correu.

9.1.1 Exemple

A continuació exposo la solució implementada, que servirà com a exemple d'aplicació de tot el framework poder entendre el funcionament intern en més detall:

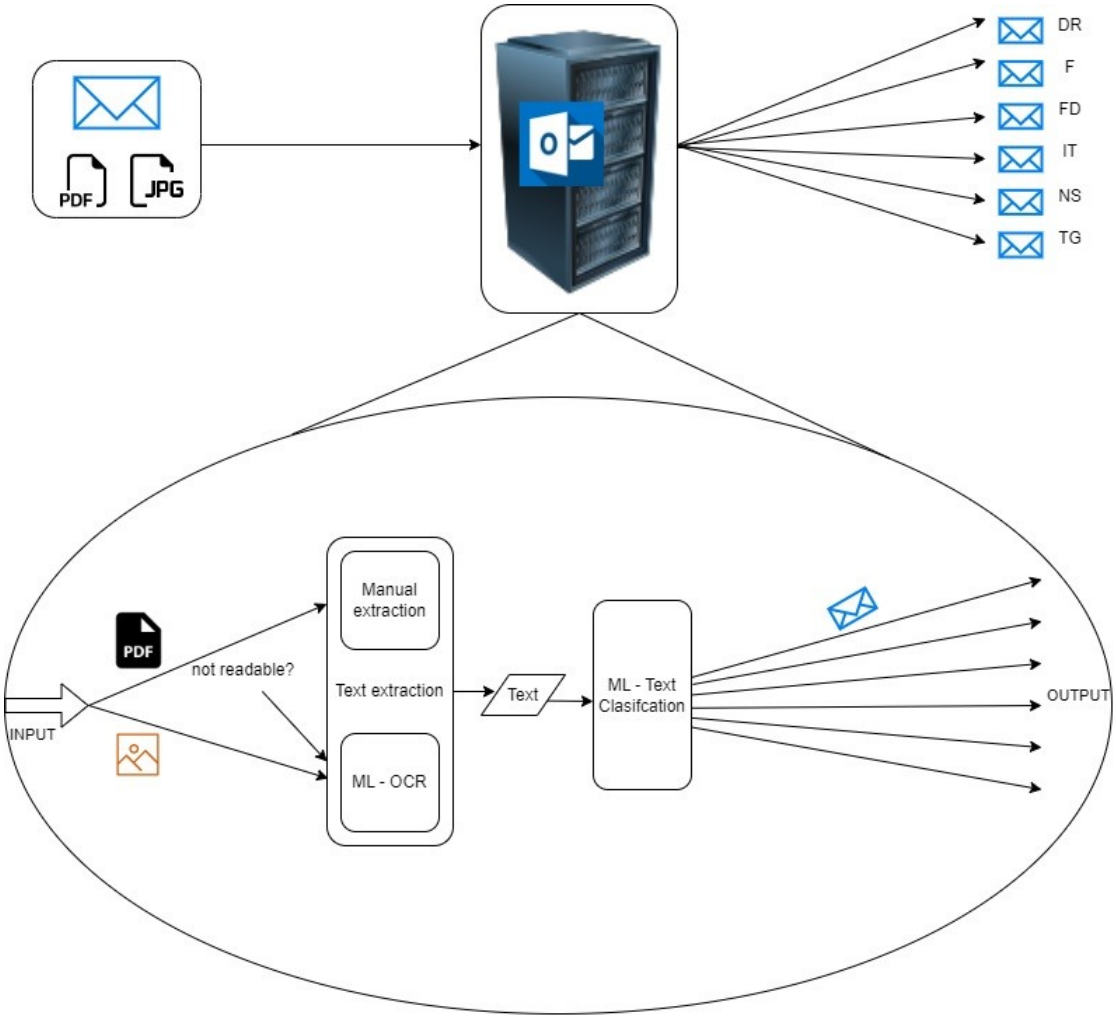

Fig 11.0 - Esquema del funcionament i disseny del framework

Aixi doncs el funcionament del framework una vegada desplegat serà el següent:

1. Comencem amb un correu electrònic d'input arribant al servidor, aquest podrà contenir un o més arxius en formats: .pdf, .txt, .jpg o .png.

2. Els arxius adjunts al correu se segmenten en totes les parts que es componen: destinatari, emissor, assumpte, cos i arxius adjunts.

3. Descarreguem els arxius i els tractem per obtenir el text que contenen, en cas que estiguem tractant amb pdf no interactius o imatges, utilitzarem un model OCR ja disponible, en cas contrari, llegim el text.

4. A partir del text conjunt aconseguit pels fitxers, fem una crida al model per classificar la nova mostra i obtenim el resultat de la classificació (DR,F,FD,IT,NS o TG).

5. L'output serà doncs l'enviament del correu electrònic inicial al destinatari pertinent a partir de la classificació anterior i/o l'enviament del correu electrònic inicial a l'emissor inicial indicant l'output del classificador a l'assumpte del correu.

9.2 Disseny i classes

Seguiré un model simple d'implementació basant-me en principis d'escalabilitat, adaptació i modularitat.

De cara a l'usuari, tindrem una sola classe anomenada AutoMailClassifier, l'usuari haurà de crear un objecte d'aquesta classe i fer ús de totes les funcions per tal de desenvolupar la seva solució. La classe AutoMailClassifier a la vegada internament fa crida a altres classes en forma de cascada depenent de la necessitat. Cada classe representarà un fitxer Python i el conjunt sencer de fitxers Python conformen la solució final del producte. A més a més hi ha un exemple d'implementació per mostrar totes les funcionalitats del producte.

Fitxer	Descripció
Implementation	Exemple d'implementació: Crea un objecte d'AutoMailClassifier i demostra totes les funcions del producte.
AutoMailClassifier	Classe principal de la solució (llibreria): Controla tot el servei, gestiona i utiliza el model escollit, implementa un objecte de la classe inferior.
EmailService	Classe de gestió del servei de correu: Recepció i enviament de correus, implementa un objecte de la classe inferior.
TextConverter	Classe de conversió a text: Converteix qualsevol input a text, implementa un objecte de la classe inferior.
ImageProcessor	Classe de OCR: Implementa una solució externa per obtenir un text a partir d'una imatge.

Fig 12.0 - Taula representant els diferents fitxers i classes involucrades en la solució final del framework

9.2.1 AutoMailClassifier

Aquesta classe com hem dit és la solució final de tota l'implementació d'aquest projecte. Aquesta classe doncs és la que seria facilitada com a documentació un usuari que vulgui implementar la seva solució en el seu entorn, serà l'usuari doncs que s'encarregui d'entendre el funcionament de la classe per poder desplegar el servei.

Aquesta classe s'encarrega de gestionar tots els serveis involucrats en tot el framework, des del model classificador fins al servei de correu electrònic. En aquest apartat doncs explicaré totes les funcionalitats i de manera indirecta el funcionament intern del framework.

9.2.1.1 Variables

- input-mail: String per determinar la direcció de correu electrònic del servidor, aquesta serà doncs el correu que realitzarà la recepció, classificació i enviament de l'output cap a altres correus.
- white-list: Llista de strings, serveix per determinar "a qui escoltem", per motius de seguretat, només processarem doncs els correus electrònics entrants a la bústia si estan en aquesta llista blanca de direccions de correus acceptats.
- output-mail-list: Llista de strings, determinem a qui volem enviar l'output del nostre processament.
- categories: Nombre de categories les quals volem processar, això determina també la classificació del nostre model, quantes categories volem classificar.
- mail-categorization: Boolean per determinar si aquesta característica està activada o desactivada, aquesta voldrà dir si el nombre de categories és el mateix que la llargada de la llista de correus d'output, de manera que farem servir un correu per categoria. Si aquesta característica està desactivada, l'output es realitzarà a l'unic correu de la llista d'output indicant el resultat de la classificació.
- labels: Llista de strings les quals determinen el títol que li volem donar a les nostres labels, ja que l'output del nostre classificador serà un nombre enter representant cada categoria.
- email-service: Objecte de la classe EmailService necessària per gestionar tot el servei de correu, aquest objecte farà servir algunes de les variables introduïdes en aquesta classe.
- classifier-file: String indicant el path físic al nostre fitxer del model classificador.
- tokenizer-file: String indicant el path físic al nostre fitxer del tokenizer, necessitat per poder fer servir el model classificador.
- debug: Boolean indicant si l'opció de debug està activada o no, aquesta servirà per debugejar el servei sencer en temps real fent us de fitxers locals de text.
- debug-path: String indicant la direcció on volem emmagatzemar els fitxers de debug.
- model: Objecte de tipus model classificador que serà usat dins l'entorn per classificar noves mostres.
- toc-filename: String modificable per determinar el path relatiu del tokenizer.
- limit-days: Int per indicar el rang temporal retrospectiu per limitar el funcionament del servei a partir de les dates d'entrada dels correus electrònics.

9.2.1.2 Mètodes

- validate-emails: Fent ús d'expressions regex, aquest mètode comprova que totes les adreces de correu introduïdes siguin vàlides gràcies al package 're'.
- set-classifier: A partir de la variable del classificador i del tokenizer, carrega el model i el tokenizer passat per paràmetre i l'assigna a les variables corresponents.
- reset-classifier: Reverteix el model classificador al que tenim per defecte (el desenvolupat en els punts anteriors).
- set-processing-limit-days: A partir d'un int, determinem la variable de limit-days determinant un límit màxim de trenta dies.
- enable-dissable-mail-categorization: Activem o desactivem mail-categorization.
- enable-disable-log-debugging: Activem o desactivem el sistema de debug a través de fitxers.
- reset-debug-files: Eliminem el directori i el contingut dels fitxers de log.
- classify: Fent ús del model carregat, classifiquem una mostra passada per paràmetre en format string i retornem el resultat de la classificació. Això és possible gràcies a tot el conjunt de paquets de 'keras'.
- try-service: Executem de manera única el servei per testejar el funcionament d'aquest. L'execució es tancarà un cop s'hagin processat tots els correus.
- start-service: Executem de manera indefinida el servei.

9.2.2 EmailService

Aquesta classe és implementada a través d'un objecte per la classe d'AutoMail-Classifier. L'objecte s'encarrega d'executar els mètodes pertinents per tal de poder rebre, processar i enviar correus electrònic de manera automatitzada. Aquesta classe basa el seu servei en implementacions de la llibreria de 'os' la qual a la vegada es basa en automatitzar processos de Windows de manera que l'entorn on es vulgui executar el servei serà important que sigui compatible amb la solució. Aquesta classe implementa un objecte de la classe TextConverter.

** Em limitaré a descriure les variables i mètodes no inclosos en la classe anterior, ja que moltes d'aquestes variables són comunes en totes les classes.*

9.2.2.1 Variables

- input-dir: String per determinar el path on s'emmagatzemaran els fitxers obtinguts a partir del processament dels correus electrònics d'input.
- MESSAGE-TO-SEND: Diccionari distribuït en 4 elements, el primer element determina la adreça de correu a la qual es vol enviar el correu, el segon element determina l'assumpte del correu electrònic, el tercer element és el cos del correu i l'últim element és una llista dels fitxers adjunts.
- converter: Objecte de la classe TextConverter utilitzat per convertir l'input necessari en text.

9.2.2.2 Mètodes

- get-messages: Gràcies a les llibreries d'automatització de processos de Windows per Python 'win32con' i 'os' aconseguim el client de correu Outlook instal·lat localment a la màquina, l'obrim i obtenim tota la inbox de correu per processar els missatges.
- process: Aquest mètode s'encarrega de processar un missatge a partir d'una carpeta local, cada carpeta representa un element de la inbox d'entrada, separat en diferents parts, l'output d'aquest mètode és el text present en els fitxers adjuntats en el correu.
- reset-message-to-send: Amb aquest mètode, revertim el missatge a enviar cada cop que fem l'enviament d'un missatge.
- send-message: Similar al mètode de get messages, un per un enviem els correus electrònics que tinguem pendents per enviar.

9.2.3 TextConverter

Aquesta classe s'encarrega de permetre a la classe anterior convertir qualsevol fitxer d'input a una string com a output. Aquest procés es fa a través de la creació d'un objecte d'aquesta classe i la invocació de diferents mètodes disponibles. A més aquesta classe fa ús de paquets com per exemple 'pdf2image' i 'pdfplumber' per convertir PDFs interactius a text i implementa un objecte de la classe ImageProcessor.

** Em limitaré a descriure les variables i mètodes no inclosos en la classe anterior, ja que moltes d'aquestes variables són comunes en totes les classes.*

9.2.3.1 Variables

- image-processor: Determinem el path físic dins del nostre disc on s'emmagatzema el nostre processador d'imatges d'OCR, en el nostre cas 'Tesseract-OCR' el qual és un paquet gratuit per processar imatges i obtenir text en entorns Python.

9.2.3.2 Mètodes

- set-install-folder: Determinem el valor del path de la variable image-processor. En cas que el path que nosaltres haguem utilitzat no sigui el correcte o vulguem utilitzar un altre model d'OCR.
- file-to-text: Mètode 'mare' per aconseguir el text a partir d'un fitxer, l'input sent el path del fitxer i l'ouput una string. Cridarem a diferents mètodes en funció del tipus de fitxer que estiguem tractant a partir de la seva extensió.
- pdf-to-image: Fa ús dels paquets mencionats anteriorment per tal de poder convertir un pdf no interactiu en una imatge.
- pdf-to-text: Mètode que fa ús del mètode extract-text() per tal de poder extreure text a partir d'un fitxer pdf interactiu.
- image-to-text: A partir del objecte creat d'ImageProcessor, processarem una imatge a partir del seu filepath i aconseguirem el text a partir del model OCR que haguem utilitzat.
- text-to-text: A partir d'un fitxer de text (.txt) aconseguirem la string que el representa.

9.2.4 ImageProcessor

Aquesta classe forma la última peça del trencaclosques per tal de fer el preprocessament dels fitxers que rebem com a input adjuntats als correus electrònics. Aquest component únicament s'encarrega de transformar imatges a text fent ús de models ja disponibles a la llibreria de 'pytesseract', tot i així aquest component segueix sent una classe a part i un objecte de la classe TextConverter per mantenir la modularitat del framework i oferint flexibilitat a l'hora d'implementar una solució pròpia.[14]

9.2.4.1 Variables

- installation-path: Aquesta variable determina el path fisic dins del nostre disc de la instal·lació del motor d'OCR que volem utilitzar per transformar imatges a text. En cas de voler utilitzar un motor propi o alternatiu ho podrem fer.

9.2.4.2 Mètodes

- process: Fent ús de la llibreria mencionada anteriorment, a partir del path físic d'una imatge processarem aquesta amb el motor d'OCR esmentat per tal d'obtenir el text suposadament present en la imatge retornant com a output una string.

9.3 Exemple d'implementació

Per tal de demostrar la funcionalitat tant del framework com del model, el fitxer implementation.py demostra un exemple d'implementació, en el meu cas l'esmentat fent ús del model de xarxes neuronals. Aquest exemple demostra totes les variables necessàries a més d'opcionals per tal de poder posar en marxa el servei.

9.3.1 Requisits

Per tal de poder implementar la solució d'exemple o qualsevol altra involucrant el framework, és necessari tenir certs requisits per tal que tot el servei pugui funcionar.

Item	Descripció
Sistema operatiu	Windows 10 (tot i que no s'ha testejat el funcionament en altres sistemes operatius, és probable el funcionament en versions posteriors).
Outlook	És necessari tenir instal·lat l'Outlook inclòs en el lot de Microsoft office 16, aquesta versió és l'única suportada actualment, ja que les posteriors (365) no són automatitzables. A més és necessari tenir la sessió iniciada dins de l'equip.
Tesseract OCR	Aquest és el motor d'OCR usat en la solució final, tot i que és possible fer ús d'altres motors, el recomanat i testat és el Tesseract OCR - Windows 64 5.3.0.2022 [14].
Packages	Es necessari tenir instal·lat i actualitzats tots els requisits i paquets de tots els fitxers Python distribuïts en la solució.
Altres	Es necessari una connexió a internet estable i permisos d'escriptura i lectura a la carpeta on es vol implementar la solució. L'usuari que executi la solució ha de ser administrador del sistema. Un espai lliure mínim en el disc on s'executi de 10GB.

9.3.2 Consideracions

Per la naturalesa del funcionament del framework, l'aplicació emmagatzema els fitxers adjunts obtinguts dels correus electrònics que es troben a la bústia d'entrada del servidor, és per això que el sistema només emmagatzemarà aquells fitxers els quals provinguin dels correus electrònics enviats per les direccions compreses a la 'white-list'. Tot i això, sempre cap la possibilitat de rebre un fitxer infectat amb un virus el qual acabem descarregant dins del nostre sistema. Cal prendre les precaucions pertinents.

L'aplicació doncs requereix un espai d'emmagatzemament dins del disc dur on s'estigui executant l'aplicació, és recomanable determinar una quota de disc o un esborrat automatitzat d'aquests fitxers de manera que s'esborrin un cop hagin passat els dies que tenim configurats d'obtenció d'input dels correus, de manera que els fitxers no ocuparan espai de manera il·limitada sinó que seran descarregats, tractats i eliminats per tal de mantenir un límit a l'espai que necessiten aquests processos.

EXEMPCIÓ DE RESPONSABILITAT: No em faig responsable dels mals o perjudicis causats pel mal ús d'aquest projecte.

9.3.3 Passos

Per tal d'implementar la solució final, s'ha de crear un entorn Python amb els fitxers compartits i seguir l'exemple d'implementació present al fitxer 'implementation.py':

1. En una variable determinar l'adreça de correu electrònic del servidor, aquesta ha de ser la que tenim associada a l'aplicació del equip d'Outlook.
2. En una variable crear una llista blanca d'adreces de correu. Aquesta llista serà la que considerarem a l'hora de processar els correus electrònics d'entrada.
3. En una variable crear la llista d'adreces de correu electrònic d'output. Aquesta llista representaran les direccions que voldrem utilitzar en funció de l'output del nostre classificador. En l'exemple només hem afegit una sola direcció.
4. En una variable determinarem el nombre de categories que classificarem.
5. En una variable determinarem les labels per les quals voldrem classificar en format de diccionari, on la clau és el valor que obtindrem de la classificació del model i el valor per cada clau és el títol que voldrem utilitzar com a output cap a l'usuari.
6. En dues variables determinarem i localitzarem tant el model classificador com el tokenizer o vectorizer que s'ha utilitzat per tractar les dades d'entrenament, aquest ha de ser el path relatiu.
7. Crearem un objecte de tipus AutoMailClassifier amb nom 'classifier' passant per paràmetre les cinc primeres variables creades anteriorment per ordre (input-mail, white-list, output-mail-list, categories, labels).
8. (Opcional) A través de l'objecte creat al pas anterior, podem activar debugg logging passant per paràmetre 'true' al mètode enable-disable-debug-logging.
9. (Opcional) A través del mateix objecte resetejar els fitxers de debug amb el mètode reset-debug-files.
10. (Opcional) En cas de no necessitar la funcionalitat de mail-categorization (Aquesta és la característica que permet enviar els correus electrònics a diferents destinataris a partir de l'otuput del model classificador) cridem al mètode enable-disable-mail-categorization passant per paràmetre 'False'.
11. (Opcional) Podem resetejar els fitxers del classificador fent ús de reset-classifier.
12. (Opcional) Determinem el nombre de dies per tractar els inputs de la safata d'entrada del correu.

13. Podem ara fent ús del mètode try-service, comprovar que tots els paràmetres estan correctament configurats i que no tenim cap problema. Aquesta línia doncs efectuarà tot el procés del servei un sol cop, tot i així processarà tots els inputs de la safata d'entrada que compleixi les característiques les quals hem configurat prèviament.

14. Si el pas anterior ha funcionat correctament, ja podem obrir de manera indefinida el servei fent ús del mètode start-service, aquest és un bucle infinit ininterromput. En cas que vulguem parar el servei, cancel·lem l'execució del programa.

9.4 Proves i resultats

Per tal d'assegurar el correcte funcionament de tot el framework, es van fer diferents proves de rendiment i correcte funcionament amb cada component per separat. Un cop em vaig assegurar que cada component feia el que tocava, es van desenvolupar les classes esmentades anteriorment i es van casar els components per tal d'assegurar un correcte funcionament. La implementació dels fitxers de debug va ser una necessitat de desenvolupament, ja que era molt complicat debugejar tots els fitxers quan quelcom fallava a l'execució; tot i això, l'he deixada com una eina extra de cara a l'usuari final.

Les proves de la solució final (parlem del framework i el model classificador), s'han elaborat al meu ordinador personal, fent servir un compte de correu electrònic creat només per aquest fi.

S'han fet diverses proves per testejar el funcionament del model, principalment des de la meva compta personal de la UB, enviant correus amb tota mena de fitxers al compte del servidor.
Els resultats són els esperats, tot i això s'ha de tenir en compte que juga un gran paper i a la vegada té una afectació al rendiment de la solució tot el tema de l'extracció de text a partir d'imatges. És d'esperar doncs que si tractem de fer una fotografia amb el mòbil d'un document, i aquesta ens ha sortit borrosa o desenfocada, en enviar aquest element, és possible no obtenir la resposta esperada.

Tenint en compte aquestes limitacions, el framework compleix les expectatives que s'havien plantejat inicialment inclús superant-les gràcies a la modularitat de la implementació i flexibilitat d'aquesta, permitent al 'end user' acabar plantejant una solució excel·lent pel seu problema.

Per mostrar d'una manera més dinàmica tot aquest funcionament plantejat, he gravat el comportament del framework en el meu entorn mostrant un exemple pràctic, aquest simplement mostra com un usuari procediria a l'execució del servei per classificar un parell d'imatges adjuntades al correu electrònic, rebent com a output final la classe de fitxers que ha enviat. Si us plau, consulteu l'annex nº5.1.

Per visualitzar el desenvolupament en brut dels components individuals implementats al framework final podeu consultar els annexos nº6.2 i 6.3.

10 Conclusions i treball futur

A partir de tota la feina feta i comparant amb els objectius principals, podem concloure que s'han complert les expectatives, almenys en termes de desenvolupament, s'hauria de mirar ara si el producte és apte per presentar al mercat i si aquest triomfarà.

S'hauria de plantejar també la feina requerida que necessitaríem aplicar a aquest projecte per poder presentar diferents implementacions del mateix producte o modificacions necessàries o opcionals per poder garantir la utilitat del producte. Possibles millores:

- Estandarditzar el model OCR: Fer servir el mateix model OCR per obtenir les dades inicials per desenvolupar el model i el que s'utilitza en el processament d'una imatge nova. Això ens garantiria que les dades extretes d'un mateix document són les mateixes per ambdós models, augmentant la precisió de cara a l'usuari final a l'hora de classificar nous documents.

- Preprocessament: Elaborar mètodes de preprocessament per a noves mostres. Augmentar la qualitat de les mostres que l'usuari final vulgui classificar amb el model, realitzant un tractament previ abans de la classificació (orientació, neteja de soroll, escala de grisos, augment del contrast, etc.).

- Experimentar amb una interfície gràfica per tal de donar l'oportunitat a qualsevol persona que no estigui familiaritzada amb el desenvolupament de codi, de desplegar el servei d'una manera fàcil amb una GUI.

Una part molt important dels coneixements usats en el desenvolupament d'aquest projecte, han estat adquirits en les assignatures d'Aprenentatge automàtic, Programació, Disseny de software i Ciberseguretat entre altres. Tot i això, ja que aquest projecte és un conjunt de moltes disciplines diferents, he hagut d'afrontar molts problemes que no havia experimentat fins ara, el qual ha requerit una extensa recerca a diferents recursos d'internet així com tutories i consultes a experts en el tema per tal de trobar un camí i poder acabar desenvolupant el producte que s'havia plantejat en un inici. El desenvolupament d'aquest projecte m'ha servit per poder aplicar moltes de les tecnologies, disciplines i funcionalitats que he anat aprenent durant tota la carrera, és possible que hagi sigut el projecte més enriquidor i divertit que he participat mai, és per això que espero poder continuar desenvolupant aquest projecte implementant les millores esmentades.

11 Bibliografia web

Referències

[1] Alexander Amini, Massachusetts Institute of Technology (MIT): Introduction to Deep Learning — Series Chapters 1-4, https://www.youtube.com/watch?v=QDX-1M5Nj7s&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&ab_channel=AlexanderAmini, 2023.

[2] DataRobot: Introduction to Dataset Augmentation and Expansion, https://www.datarobot.com/blog/introduction-to-dataset-augmentation-and-expansion/, 2018.

[3] Grant Sanderson; 3Blue1Brown: Bayes theorem, the geometry of changing beliefs, https://www.youtube.com/watch?v=HZGCoVF3YvM&ab_channel=3Blue1Brown, 2020.

[4] Grant Sanderson; 3Blue1Brown: But what is a neural network? — Series Chapter 1-4, Deep learning, https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&ab_channel=3Blue1Brown, 2018.

[5] Hugo Ferreira: Confusion matrix and other metrics in machine learning, https://medium.com/hugo-ferreiras-blog/confusion-matrix-and-other-metrics-in-machine-learning-894688cb1c0a, 2018.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding , https://huggingface.co/docs/transformers/main/model_doc/bert#:~:text=BERT-,Overview,Kenton%20Lee%20and%20Kristina%20Toutanova., 2018.

[7] Josh Starmer; StatQuest with Josh Starmer: Long Short-Term Memory (LSTM), Clearly Explained, https://www.youtube.com/watch?v=YCzL96nL7j0&t=1s&ab_channel=StatQuestwithJoshStarmer, 2021.

[8] Josh Starmer; StatQuest with Josh Starmer: Naive Bayes, Clearly Explained, https://www.youtube.com/watch?v=O2L2Uv9pdDA&t=138s&ab_channel=StatQuestwithJoshStarmer, 2021.

[9] Josh Starmer; StatQuest with Josh Starmer: Recurrent Neural Networks (RNNs), Clearly Explained, https://www.youtube.com/watch?v=AsNTP8Kwu80&t=3s&ab_channel=StatQuestwithJoshStarmer, 2022.

[10] Josh Starmer; StatQuest with Josh Starmer: Neural Networks Pt. 3: ReLU In Action,
https://www.youtube.com/watch?v=68BZ5f7P94E&ab_channel=StatQuestwithJoshStarmer, 2021.

[11] Lutz Roeder: Netron 5.0 — Viewer for neural network, deep learning and machine learning models
https://github.com/lutzroeder/netron, 2023.

[12] Nathan Friend: tree.nathanfriend.io — Tree-like utility for generating ASCII folder structure diagrams
https://tree.nathanfriend.io/, 2020.

[13] Nikolai Janakiev: Practical Text Classification With Python and Keras ,
https://realpython.com/python-keras-text-classification/#author, 2020.

[14] Ray Smith, Hewlett-Packard; Tesseract OCR: Open source OCR engine - Apache License 2.0 (link here) ,
https://tesseract-ocr.github.io/tessdoc/, 2023.

[15] Sakares Saengkaew, Binoy Dalal, Emrecan Çelik: Data augmentation NLP library
https://nlpaug.readthedocs.io/en/latest/augmenter/augmenter.html, 2019.

[16] Scikit-learn team, (Link here): Supervised learning library
https://scikit-learn.org/stable/supervised_learning.html, 2020.

12 Annexos

1.1 Diagrama de Gantt, Timeline del projecte, Format .pdf:
https://drive.google.com/file/d/1FVxj8mN95vIwlbSuc2x6B-9lWNXjJ8HQ/view?usp=share_link

1.2 Diagrama de Gantt, Timeline del projecte, Format .gantt:
https://drive.google.com/file/d/1D2Dmin0Z3sQJQabM1EBtx6q7HZFWfoBQ/view?usp=share_link

1.3 Diagrama de Gantt, Timeline del projecte, Format .xlsx:
https://docs.google.com/spreadsheets/d/11dzkCgZqpzaYVfxNfL0n154qbXDCLA_e/edit?usp=share_link&ouid=103562109287860881656&rtpof=true&sd=true

2.1 Esquema del funcionament del framework, Format .jpg:
https://drive.google.com/file/d/1pVMCp2DJXS6he_xq6FiD4UrzYRjOgKLW/view?usp=share_link

3.1 Esquema del l'arquitectura interna del model classificador de xarxes neuronals, Format .png:
https://drive.google.com/file/d/1-whi5UKZxE6cS3uyqVw2RB6_1BLti55O/view?usp=share_link

4.1 Matrius de confusió de tots els models i càlculs adjacents, Format .xlsx:
https://docs.google.com/spreadsheets/d/11CC2wHOmYh47SpTxKq3_1yxeVlQz144J/edit?usp=share_link&ouid=103562109287860881656&rtpof=true&sd=true

5.1 Demostració del servei amb un exemple pràctic (Vídeo), Format .mp4:
https://drive.google.com/file/d/1cRImAz2HU-UwABaiQDGUiNDO6gCjUX2Q/view?usp=share_link

6.1 Desenvolupament de tots els models classificadors implementats al treball així com el tractament de dades i exportació del model, Format .py:
https://drive.google.com/file/d/1aurvDqin5d_uw_pXnwsBQHBmbkrKc--r/view?usp=share_link

6.2 Proves en brut per al desenvolupament del servei de correu electrònic, Format .py:
https://drive.google.com/file/d/1mbif5kzzPfe4JIVchR3Je-KEl78XK5mV/view?usp=share_link

6.3 Proves en brut per al desenvolupament del servei de processament d'imatges, Format .py:
https://drive.google.com/file/d/1T8YX6GKeafa0z9Hp7gq5noU83jBKvafz/view?usp=share_link