NLP per a la resolució de tiquets

Irina Moreno Lahoz

Resum– Resum del projecte, màxim 10 línies. Aquest projecte s'ha enfocat en la predicció de text, en poder predir una solució a un nou problema generat per poder ajudar als empleats a solucionar-ho de la forma més senzilla i ràpida possible. Abarca molt temes interessants, com la bona estructura i neteja de dades, l'anàlisi i comparació de resultats i una possible implementació lligada a la intel·ligència artificial, un tema molt present en dies d'ara.

Paraules clau– tiquets, NLP, Machine Learning, Inteligència Artificial, solució, text, predicció, model, paràmetres, dades.

Abstract– Versió en anglès del resum. This project has focused on text prediction, on being able to predict a solution to a newly generated problem in order to help employees to solve it as easily and quickly as possible. It covers a lot of interesting topics, such as good data structure and cleaning, analysis and comparison of results and a possible implementation linked to artificial intelligence, a very present topic nowadays.

Keywords– tickets, NLP, Machine Learning, Artificial Intelligence, solution, text, prediction, model, parameters, data.

• E-mail de contacte: irina.moreno@autonoma.cat
• Treball tutoritzat per: Débora Gil Resina (Ciències de la Computació)
• Curs 2022/23

1 INTRODUCCIÓ

AQUEST treball s'ha desenvolupat en el departament d'*IT* de l'empresa SEAT S.A., on es dediquen a la producció i la fabricació d'automòbils. En aquest departament, una de les tasques que tracten són les incidències que tenen els treballadors. Emmagatzemen dades sobre la informació de la incidència en format de tiquets en una base de dades. Aquestes incidències són una part molt important en el rendiment de l'empresa, ja que tenen un gran impacte en la producció i en el treball de l'empresa.

Es vol realitzar una ajuda a l'empresa amb la gestió de la solució d'aquests tiquets d'incidències.

1.1 Objectius

L'objectiu principal del projecte és poder donar una solució o una aproximació a la solució de la incidència que té l'usuari, sent una capa intermèdia entre l'usuari i la solució, i així, poder evitar acudir al servei de suport de tècnics propi de SEAT S.A. o minimitzar aquest accés.

D'una banda, es vol arribar a una solució molt més ràpida perquè l'usuari segueixi amb el seu treball amb total normalitat. I, d'altra banda, es vol minimitzar costos en els contractes de l'ajuda oferta pel servei de suport.

El treball se centrarà en l'estudi de mètodes de Processament del Llenguatge Natural (*NLP*, per les seves sigles en anglès) per trobar bones solucions a les incidències, i després es plantejarà com implementar aquest estudi als programes i processos de l'empresa i que passi a ser una bona ajuda, ja que pot pot tenir impacte económic sobre els contractes de servei i en el temps de desenvolupament i fabricació.

Com a objectiu secundari, que es realitzarà si es té el temps i els recursos necessaris per fer-ho, tenim la creació d'una nova interfície en l'aplicació que l'empresa utilitza en la generació dels nous tiquets. En aquesta aplicació, ara l'usuari introdueix totes les dades sobre la incidència i s'envien al servei de suport tècnic, i, el que es vol aconseguir ara és que en introduir aquestes dades surti una possible solució que l'usuari pot aplicar. Si no és possible aquesta implementació, l'empresa farà servir aquest treball com a inici d'un nou projecte on ja s'ha portat a cap un primer estudi.

1.2 Planificació

Aquest projecte estarà dividit en un seguit de fases.

En primer lloc, es farà un estat de l'art sobre el món de NLP. Es buscaran els mètodes i les eines que existeixen i quines poden ser aplicades per arribar a l'objectiu d'aquest projecte.

En segon lloc, la primera cosa important a tenir en compte com a començament d'aquest projecte és entendre les dades amb els quals es tractarà i saber quina és la millor manera de tractar-les. Hi ha un volum de dades molt elevat. Es realitzaran una sèrie de filtres per eliminar aquelles dades que no són útils pel nostre mètode o aquelles que es troben lluny d'aconseguir el nostre objectiu. També es procedirà a dur a terme subconjunts de dades depenent de la relació que tinguin les solucions i per poder ajudar al mètode a trobar millors solucions.

Com a tercer punt, s'estudiaran tots els models, tècniques i eines que es poden fer servir per arribar a una bona solució. S'analitzaran i es triaran els que puguin donar millors resultats.

Per continuar, s'aplicarà a les dades el mètode triat.

A partir d'aquests resultats, també podem saltar a les fases anteriors i fer canvis en la preparació de les dades o buscar mètodes millors. Són 3 fases que es complementen.

Arribant al final del projecte, s'estudiarà la possibilitat de realitzar la nova interfície de l'aplicació que genera nous tiquets. Es vol parlar amb caps i persones creadores de l'aplicació per a piular codi, crear i connectar tot el necessari.

Com a part final, es crearan les conclusions finals del projecte. Explicacions acompanyades de *dashboards* de l'anàlisis dels resultats.

Es treballarà en un entorn de Jupyter amb el llenguatge de Python.

A continuació, s'introdueix una taula que especifica el temps de dedicació a cada part d'aquest projecte.

Fases	Temps a dedicar
Estat de l'art	2 dies
Gestió de les dades	5 setmanes
Disseny experimental	5 setmanes
Estudi d'integració	1 setmanes
Conclusions	1 setmana

Taula 1: TAULA DE PLANIFICACIÓ

2 METODOLOGIA

En aquesta fase es comentarà com i amb quines eines es realitza qualsevol modificació en la base de dades i s'explicaran les diferents característiques de cada tècnica que s'utilitza per arribar a fer el model NLP.

La següent imatge mostra l'arquitectura que es seguirà al projecte.

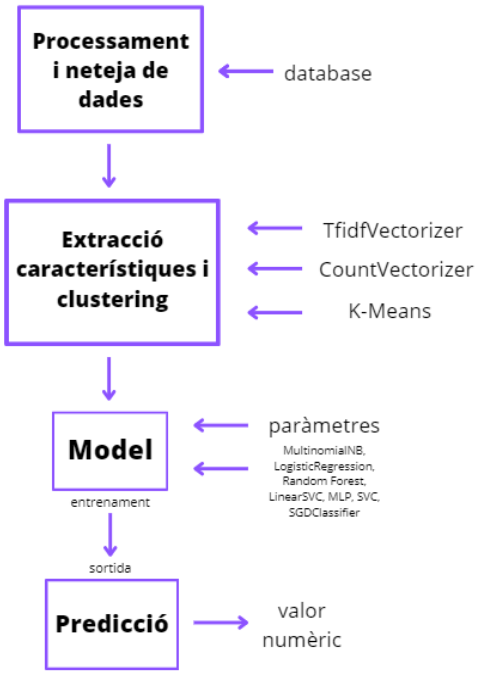

Fig. 1: Arquitectura i organització del treball

Com a primera part tenim la preparació de les dades que provenen de l'empresa SEAT S.A. I, com a segona part, treballem amb la Intel·ligència Artificial, la qual té gran importància en aquest projecte. La *IA* [2] és una branca de la informàtica que busca crear màquines que imitin la intel·ligència humana per a fer tasques i puguin millorar conforme la informació que recopilen. La màquina rep dades, les processa i respon a elles. Per a això, la *IA* empra una àmplia varietat de tècniques: aprenentatge automàtic, el processament del llenguatge natural, la visió artificial i l'aprenentatge profund.

2.1 Neteja i filtatge

L'empresa SEAT S.A. té una gran quantitat de dades emmagatzemades, dades de tota mena. En aquest projecte es tractarà amb el dataset on es guarda tota la informació sobre els tiquets d'incidències. Aquest dataset s'actualitza cada dia i va creixent de mida a mesura que va passant el temps, al dia es poden arribar a generar 800 nous tiquets. El tiquet més antic que hi ha emmagatzemat és de la data 26/08/2015 i el dataset té una mida de 891829 tiquets aproximadament (última setmana de maig).

Aplicarem una sèrie de filtratges per eliminar dades irrellevants que no serveixin per entrenar el model i posteriorment s'aplicarà *clustering* per tractar en conjunt les dades que s'assemblen.

Aquests filtres són, per exemple, l'eliminació de dades buides, de paraules i/o frases repetides, de text massa curt, de text en un altre idioma, entre d'altres.

2.1.1 Clustering

El clustering [6], també conegut com a anàlisi de grups o agrupament, és una tècnica d'aprenentatge automàtic no supervisat que s'utilitza per identificar patrons i estructures ocultes en conjunts de dades. L'objectiu principal és dividir un conjunt de dades en grups o clústers, de manera que els elements dins del mateix clúster siguin similars entre ells i siguin diferents dels elements en altres clústers.

Aquesta tècnica s'utilitzarà per poder tractar la base de dades de manera més específica. En primer terme, es vol utilitzar per subdividir les dades, ja que en tenim moltes i així podem fer un model amb més precissió. En segon terme, s'utilitza per crear les etiquetes de classificació de les solucions. Tractarem amb 3 tipus d'agrupacions.

La primera agrupació, encara que pot ser no considerada agrupació o no, és directament tenir com grup el propi 'CI Name'. Però un aspecte a tenir en compte és que la nostra base de dades consta de un número molt gran de dades, i com la primera idea d'aquest projecte és trobar el millor model, s'ha decidit treballar a partir d'aquesta agrupació. Es vol crear un model diferent per cada *CI* aplicant a tots els mateixos pasos.

Un altre mètode es l'agrupació a partir d'una variable anomenada *Service Tower* que es troba en un altre dataset. Service Tower son unes categories on es classifiquen les diferents aplicacions que estan relacionades o pertanyen al mateix departament per a fer anàlisis en conjunt. Fent aquesta agrupació, es vol aconseguir tenir en compte en l'aprenentatge un conjunt de *CI*, no només un de sol.

El tercer mètode és 'K-means'[5] de la llibreria 'Sklearn', on el seu objectiu principal és agrupar les dades en k grups diferents, basant-se en les seves característiques similars. Comença amb la inicialització de k punts, coneguts com a centroids, que representen els centres dels grups. A continuació, s'assigna cada punt de dades al centroid més proper basant-se en la seva similitud. Aquest procés es repeteix fins que la convergència s'aconsegueix, és a dir, quan els punts ja no es reassignen als centroids. Això permet agrupar paraules o frases en clústers que comparteixen característiques similars. Aquesta agrupació es treballara sobre l'atribut 'Solution' i s'aplicarà un cop aplicat els altres mètodes d'agrupament.

2.2 Aplicacions, Tècniques i Eines de NLP

Ens centrarem en el Processament del Llenguatge Natural (*NLP*)[3], que és una àrea d'estudi centrada en com els ordinadors entenen el llenguatge humà, l'interpreten i processen. El *NLP* és realment la interfície entre la ciència informàtica i la lingüística. Per tant, es basa en la capacitat de la màquina per a interactuar directament amb els humans.

NLP és utilitzat per moltes aplicacions com assistents virtuals (*chatbots*), traducció automàtica de textos, recuperació d'informació, classificació de textos, detecció de sentiments, resum de textos, reconeixement d'entitats, anàlisi de veu, etc.

Entre totes aquestes possibles aplicacions, el treball se centrarà en la classificació de textos i recuperació d'informació, ja que a partir d'un text, haurà de detectar les paraules més rellevants i, finalment, acabarà classificant aquell text en una solució o una altra.

NLP utilitza una sèrie de tècniques que s'utilitzen per a processar, analitzar i comprendre el llenguatge humà mitjançant l'ús d'algorismes i models computacionals.

Entre les tècniques més populars, i en les que el treball es centrarà, ens trobem amb el procés de Tokenització, que divideix un text en unitats més petites anomenades *tokens*. Els *tokens* poden ser paraules, signes de puntuació o números.

També s'estudiaran tècniques com eliminar les *StopWords*, que són paraules que s'eliminen del text per a reduir la quantitat de paraules amb les quals es treballa i centrar-se en les paraules clau o significatives del text, i els signes de puntuació, ja que no son rellevants i així ens centrem en les paraules.

A més a més, es treballarà amb procesos com el *Stemming* i la Lematització. El *Stemming* és el procés de reduir les paraules a la seva forma base, com ara convertir 'corriendo' en 'corre', mentre que la Lematització és bastant similar, però en lloc de reduir les paraules a la seva forma base, les converteix a la seva forma de diccionari, com ara convertir 'corriendo' en 'correr'.

Aquestes tècniques són les que s'utilitzaran en aquest treball, però existeixen moltes més, com per exemple: Anàlisi morfològica, *Tagging part of speech* (PoS), Anàlisi de sentiments, Classificació de text, entre d'altres.

Calen eines per poder posar en pràctica totes les tècniques que utilitza NLP per a les diferents aplicacions. Les biblioteques de NLP són conjunts de recursos de programari i dades lingüístiques reconstruïts que permeten als desenvolupadors de programari crear aplicacions de processament automàtic del llenguatge natural.

Les biblioteques de NLP més populars inclouen NLTK, spaCy, Gensim, TextBlob, Stanford CoreNLP i el Natural Language Toolkit de Google, entre altres. Aquestes biblioteques solen ser de codi obert i es poden fer servir de manera gratuïta, encara que pot haver-hi restriccions en relació amb el seu ús comercial.

2.2.1 Extracció de característiques

Hi ha diverses tècniques per extreure característiques del text en NLP [4] i a continuació explico amb les que treballaré i finalment triaré amb la que millors resultars obtingui. Hi ha de més, com per exemple la bossa de paraules o l'anàlisi sintàctica, pero he triat les següents ja que trobo que els seus objectius s'adapten molt millor al que vull aconseguir.

D'una banda trobem *CountVectorizer*, que és una classe de la llibreria d'aprenentatge 'sklearn' que s'utilitza per convertir una col·lecció de documents de text en una representació numèrica, específicament una matriu de comptador de termes. CountVectorizer realitza dos passos principals: tokenització i comptatge de termes.

D'altra banda tenim la tècnica de Tf-idf (Term frequency-inverse document frequency), que té en compte la freqüència d'aparició de cada paraula en un text, però també té en compte la seva freqüència en tots els altres textos dels documents. Això permet destacar les paraules que són importants per a un text específic. Per aquesta tècnica es destaca TfidVectorizer.

2.2.2 Models de NLP i Xarxes Neuronals

La llibreria scikit-learn (sklearn) és una de les més populars per a l'aprenentatge automàtic en Python, i també és àmpliament utilitzada en NLP.

A continuació s'expliquen alguns dels mètodes de la llibreria *Sklearn* [7] que s'utilitzaran per determinar quin és el millor model a aplicar en aquest treball. Cada model s'entrenarà i s'evaluarà. S'ajustaran els paràmetres per tal de millorar-los, però finalment ens quedarem amb el que dona millor solució.

El 'MultinomialNB' de *Naive Bayes* és un model probabilístic que es basa en el teorema de Bayes per a la classificació de text. Aquest model és freqüentment utilitzat per a la classificació de textos en categories específiques.

'LogisticRegression' de *Linear Model* i 'RandomForestClassifier' de *Ensemble* són models de classificació que s'utilitzen per a la classificació de textos i altres tasques de NLP. S'encarreguen de diverses funcions, el primer s'adapta a les dades utilitzant una funció logística per a la predicció de la classe i el segon combina diversos arbres de decisió per a millorar la precisió de la predicció.

'LinearSVC' i 'SVC' de *SVM* són un altre model d'aprenentatge supervisat que es pot utilitzar per a la classificació de textos i altres tasques de NLP. Aquests models s'utilitzen per a la classificació binària i multiclasse i s'adapta a les dades per a la predicció de la classe. 'SVC' s'utilitza per mostres de tamany més reduit que 'LinearSVC'.

'SGDClassifier' de *Linear Model* implementa models lineals regularitzats amb aprenentatge de descens de gradient estocàstic, el gradient de la pèrdua s'estima en cada mostra alhora i el model s'actualitza en el camí amb un programa de força decreixent (taxa d'aprenentatge).

Els models que s'utilitzaran en aquest treball són els anteriors, encara que existeixen molts més com per exemple NearestCentroid, NuSVC, DecisionTreeClassifier, entre d'altres.

En relació a les Xarxes Neuronals, ens proposarem provar 'ANN', 'RNN' i 'CNN' [8]. Una xarxa neuronal artificial (ANN) és un grup de múltiples perceptrons/neurones en cada capa. Es coneix com a xarxa neuronal d'alimentació directa perquè les entrades només es processen en direcció directa, en canvi la xarxa neuronal recurrent (RNN) utilitza retroalimentació de la sortida d'aquella mateixa capa. A més a més, tenen una espècie de memòria interna que permet guardar informació sobre estats anteriors i a la vegada processen noves dades. Per últim, les xarxes neuronals convolucionals (CNN) utilitzen capes convolucionals que apliquen filtres per extreure característiques de l'entrada i es combinen amb capes d'agrupació per reduir la dimensionalitat d'aquestes característiques. Les llibreries utilitzades són 'Sklearn' i 'Keras'.

3 DISSENY EXPERIMENTAL

Ara toca aplicar tot el que s'ha explicat anteriorment, afegint i adaptant els mètodes a les nostres dades amb l'objectiu de crear una bona entrada pel nostre model d'aprenen-

tatge NLP i conseguir bons resultats. S'explicarà pas a pas la feina feta.

3.1 Preparació del conjunt de dades

El conjunt de dades consta d'un número molt elevat de tiquets. Cada tiquet conté molts camps emmagatzemats, però ens quedem amb els següents, ja que són els que tenen més rellevància:

- Ticket Type: si es tracta d'un incident o d'una sol·licitud.
- CI ID: combinació única per a la identificació dels tiquets.
- CI Name: nom de l'aplicació on es troba l'incident.
- Title: títol que l'usuari li dona a l'incident.
- Description: descripció que l'usuari realitza en descriure el seu problema.
- Solution: es descriu la solució al problema de l'usuari.
- Open Time: data i hora la qual el tiquet ha estat creat.
- Resolved Time: data i hora la qual el tiquet ha estat solucionat.
- Current Status: informació sobre el que s'esta fent en aquell moment amb el tiquet.
- Current Assignment: grup al qual ha estat assignat aquest incident.
- Prio: camp numèric que indica la rapidesa en què la incidència s'ha de solucionar.
- Imapct: camp numèric que indica l'impacte de la incidència dins del treball en SEAT.

Les sigles CI (*Configuration Item*) provenen de la llibreria ITIL (*Information Technology Infrastructure Library*), un marc de millors pràctiques per a la gestió de serveis de tecnologia de la informació. Un element CI correspon a qualsevol component o part d'una infraestructura de tecnologia de la informació que ha de ser gestionada i controlada per a garantir la prestació de serveis d'IT d'alta qualitat.

Primerament, s'eliminen aquells registres rellevants del dataset que estan buits, com per exemple algun tiquet que no tingui solució, *CI*, descripció o títol. Per fer això, eliminem aquelles entrades del dataframe que continguin el valor NaN en els registres mencionats anteriorment. També apliquem un filtratge i eliminem aquells tiquets que no siguin en espanyol, ja que sinó crearía problemes a l'hora de predir correctament, a partir del mòdul *detect* importat de la llibreria *langdetect*. Aquest mòdul detecta l'idioma d'un text que li passa com entrada a la seva funció, si aquest idioma és diferent a l'espanyol, eliminem aquell registre del dataset. Després d'aplica aquests filtres, el dataset resultant és de mida 301732, quedant així reduit un 50% aproximadament.

En segon lloc, ens hem de fixar amb el que realment treballarem, la solució dels tiquets. L'atribut 'Solution' és un camp que pot ser generat directament per una persona i també pot ser que una part es crei automàticament. Per aquesta raó, procedim a fer una neteja i eliminarem les parts generades automàticament i, posteriorment, eliminarem aquelles solucions on el text sigui massa curt. S'aplica el filtratge on només seleccionem les solucions les quals la longitud del text supera els 29 caràcters. Amb aquest filtratge, la base de dades s'ha reduït de mida uns 3000 tiquets, aproximadament.

Quan ja tenim tota la base de dades preparada, es procedirà a realitzar les agrupacions explicades anteriorment. L'agrupació de 'k-means' serà aplicada posteriorment.

Primerament, es mira de separar les dades depenent del seu *CI*, ja que pot ser que existeixin diferents aplicacions que tinguin un mateix problema, però la solució a cada problema sigui totalment diferent. Per això, es mirarà la quantitat de tiquets que es generen a partir de cada *CI*, pel fet que pot ser que no hi hagi suficients incidents d'aquell *CI*. No s'ha de crear, modificar ni eliminar cap camp al dataset que tenim, simplement considerarem el nom de cada *CI* en el paràmetre per fer l'agrupació. Es crearà un model diferent per cada *CI*.

Visualitzem per cada *CI* la distribució de tiquets en un histograma mostrat en la figura 2.

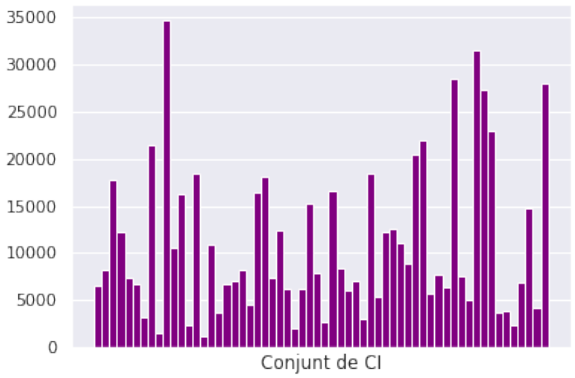

Fig. 2: Histograma CIs

Com podem veure en el gràfic anterior, hi ha una gran diferència en els volums de dades dels diferents *CI*. Estudiant amb més profunditat les dades, s'eliminen els *CI* que contenen menys de 6 registres, ja que poden provocar soroll en el nostre model d'aprenentatge perquè no tenen suficients solucions amb les quals treballar i pot provocar no arribar a solució. S'analitza i es fa un recompte dels 'CI Name' i es veu que el *CI* amb més registres a la base de dades és 'ACME (P-49)', amb 18087 registres en total com es pot veure a la figura 3.

Seleccionem tots aquests registres, creant així l'agrupació explicada de cada 'CI Name' i serà un dataset a part i sobre el qual treballarem.

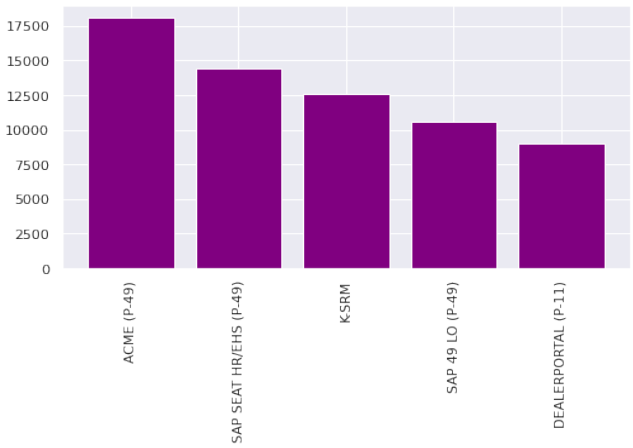

Fig. 3: Recompte aparicions del top 5 CI

Com a següent agrupació, es basarà en la variable anomenada 'Service Tower' que es troba en un altre dataset, el qual consta de moltes variables però les que utilitzarem son 'CI Name' i 'Service Tower', on l'última variable és una cadena de text que representa el grup. Per relacionar la informació d'amdós datasets, apliquem la funció 'merge' de la llibreria 'Pandas', que combina 2 datasets en funció d'una o més columnes que tinguin en comú. En aquest cas, es crea un dataset nou, idèntic al principal, però amb un camp extra anomenat 'Service Tower', generat a partir de l'unió interna (només ens quedem amb les files on hi hagi valors coincidents entre datasets) de la columna 'CI Name'.

En el gràfic de barres de la figura 4 mostrem la quanitat de diferents *CI* que es troben en aquestes categories, en aquest cas hi ha un total d'uns 904 *CI*.

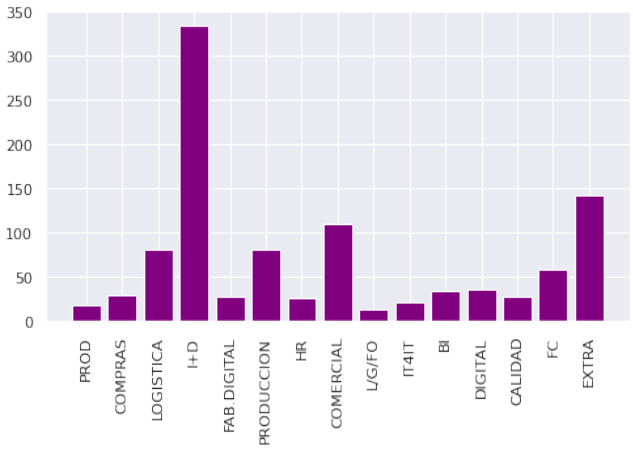

Fig. 4: CI en Service Towers

En el gràfic anterior podem veure que el 'Service Tower' amb més dades és 'I+D', però seleccionarem els registres de 'COMERCIAL' perquè té 110 tipus de *CI* i ja en són suficients per entrenar i fer proves amb el model.

Amb la creació d'aquests 2 subdatasets, es comença a posar en pràctica les tècniques explicades anteriorment perquè la màquina processi el llenguatge humà i es faran les prediccions de les solucions a partir de diferents models.

3.2 Preparació del text

Primerament, es comença fent una neteja i una adaptació en els diferents textos que tenim a tractar (Title, Description i Solution), així la màquina el processa i l'entén.

Iniciem aplicant el mètode de tokenitzar important el mòdul 'word_tokenize' de la llibreria 'NLTK'. El que fa aquest mètode exactament és separar el text paraula per paraula i ho emmagatzema en una llista.

Seguidament, eliminem els signes de puntuació de cada element d'aquesta llista. Es mira caràcter a caràcter i, si es troba en una llista anomenada 'punctuation' carregada de la llibreria 'string', vol dir que és un signe de puntuació i s'ha d'eliminar.

Com a següent pas, eliminem les *StopWords*. Importem una llista de paraules sense gran valor, com per exemple 'el','la', 'de', 'en', 'por', 'con', 'para', entre d'altres; del mètode 'corpus' de la llibreria 'NLTK' i busquem i eliminem si alguna d'aquestes paraules es troba en el text de 'Title' i 'Description'. A aquesta llista afegirem manualment una sèrie de paraules que considerem de valor nul i podrien estar presents en el text i no en aquesta llista, com per exemple 'gracias', 'martorell', 'saludos' i 'hola'.

Les anteriors modificacions han estat agrupades en una funció anomenada 'clean_text'.

Afegirem un nou atribut al nostre dataset que serà 'Text', on es trobarà emmagatzemada una llista de paraules que són la unió de l'atribut 'Title' i 'Description'.

A continuació es mostra una comparació de 2 textos. El primer, figura 5, és el text original que conté la base de dades, 'Title' i 'Description' per separat, i el segon, figura 6, és el mateix text, però se li ha aplicat els mètodes anteriors per poder treballar millor amb ell i podem observar que conté una llista de l'unió.

Solicitud apertura de puertos para VPN_TSYS 3/3 Solicitamos añadir VPN_TSYS al perfil de ingenierias para pruebas con aplicación CONNECT. Adjuntamos formulario 3/3

Fig. 5: Text abans de la neteja

['Solicitud', 'apertura', 'puertos', 'VPNTSYS', '33', 'Solicitamos', 'añadir', 'VPNTSYS', 'perfil', 'ingenierias', 'pruebas', 'aplicación', 'CONNECT', 'Adjuntamos', 'formulario', '33']

Fig. 6: Text després de la neteja

Per acabar, queda aplicar la normalització de les paraules per reduir la variabilitat de la llengua, la qual cosa facilita la comparació i l'anàlisi de text. Aquest pas no és aplicat al text 'Solution',

El mètode de lematització descrit anteriorment és útil per a normalitzar les paraules a la seva base canònica o de diccionari. Aplicant aquest mètode a les nostres dades, ens hem adonat que no hi ha gran canvi en la forma de les paraules.

Veiem un exemple a la figura 7.

['solicitud', 'apertura', 'puerto', 'VPNTSYS', '33', 'Solicitamos', 'añadir', 'VPNTSYS', 'perfil', 'ingenierias', 'pruebas', 'aplicación', 'CONNECT', 'Adjuntamos', 'formulario', '33']

Fig. 7: Text després d'aplicar mètode de lematització

Decidim, finalment, treballar amb la llibreria 'nltk', que tracta de normalitzar paraules a la seva forma bàsica o arrel. S'inicialitza carregant el model *SnowballStemmer* per l'idioma espanyol i entren com paràmetres d'entrada les paraules que es troben a la llista de paraules de 'Title' i 'Description'. El resultat acaba sent la mateixa llista que ja teníem, però amb les paraules actualitzades, com podem veure en la figura 8.

['solicitud', 'apertur', 'puert', 'vpntsys', '33', 'solicit', 'añad', 'vpntsys', 'perfil', 'ingenieri', 'prueb', 'aplic', 'connect', 'adjunt', 'formulari', '33']

Fig. 8: Text després d'aplicar mètode *SnowballStemmer*

3.3 Extracció de característiques i Clustering

En aquest apartat, es treballarà en el procés de convertir les dades en un format numèric de forma que el model d'aprenentatge automàtic les pugui entendre i com treballar amb el conjunt de dades.

Aplicarem aquest procés a les variables de la base de dades 'Text', ja que conté la informació per entrenar el model, i a 'Solution', pel fet que com tenim massa tipus de solucions diferents és molt difícil pel model classificar en una de sola, així que realitzem un clustering de les solucions que es troba explicat posteriorment.

Comencem extraient característiques, utilitzat per a extreure informació significativa del text i representar-la en forma de vectors numèrics.

Després de fer vàries probes, TF-IDF-Vectorizer és el mètode que millor s'ajusta al format que volem i el que dona millor solucions. Aquest mètode té una funció anomenada 'fit_transform' que rep com a paràmetre d'entrada una llista de textos, es crea una llista de mida 18087 valors a partir de cada text de cada dada, i retorna una *sparse matrix*, que és una forma d'emmagatzemar matrius on majoritariament els valors són zero, però ho fan ocupant menys memòria utilitzant la representació *Compressed Sparse Row*, que consisteix a emmagatzemar 3 estructures anomenades *array* (llista); una llista pels valors que no són zero, un altre per l'índex de les columnes i l'altre per l'índex de les files. Aquesta estructura té emmagatzemada la representació numèrica de tots els vectors de textos.

Quan tenim tot el text en format numèric, és el moment de la creació de clusters de les solucions. Com és una base de dades on l'atribut 'Solution' no ha estat creat de manera automàtica, sinó que les persones han hagut d'escriure-ho, existeixen moltes combinacions de solucions diferents, la qual cosa fa que sigui molt difícil pel model ja que no podrà classificar. s'ha realitzar una sèrie d'agrupacions a partir del mètode KMeans de la llibreria 'sklearn'. Com es pot observar en la figura 9, s'han creat 10 grups diferents de solucions semblants diferenciats pels colors distribuits per un espai de dimensió reduit, i cada agrupació consta d'un centre, representat amb una creu blava, que representa al promig de cada conjunt de dades, on les dades tendeixen a agruparse. Hi ha grups que es troben molt més centrats i altres que estan més distribuits, hi ha molta varietat de solucions.

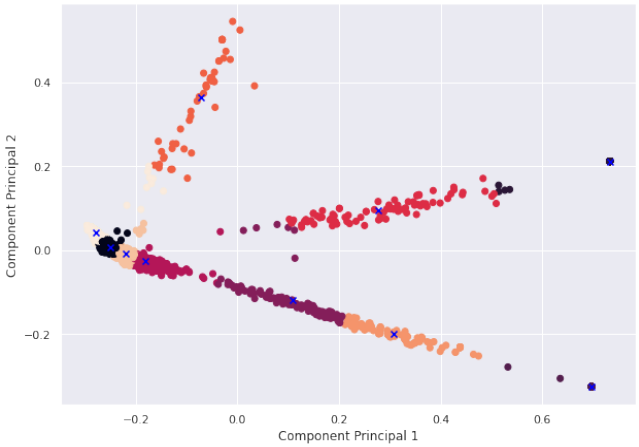

Fig. 9: Clustering amb K-means (PCA)

A partir d'aquestes agrupacions, assignem una etiqueta de 'cluster' a cada registre, la qual es una variable que ens servirà posteriorment com a etiqueta de classificació per entrenar al model.

Cada 'cluster' consta de diferents textos similars i es manté un recompte dels textos agrupats en aquesta agrupació. Pot ser que un mateix text s'hagi trobat 102 vegades en tota la base de dades i altres un 2, però tots 2 textos volen comunicar el mateix i es troben en el mateix 'cluster'. Això ens és útil a l'hora de classificar solucions, ja que el model ens dona com a resultat un valor de 'cluster' i el resultat que donarem és el text que tingui el comptatge més alt.

3.4 Comparació dels Models

Un cop tenim les dades preparades, passem a entrenar amb diferents models per veure quin dona millors resultats. També es farà una comparativa amb ells mateixos però amb diferents paràmetres d'entrada.

L'estructura que es seguirà és la mateixa per tots els models. Comencem amb les dades respresentades amb vectors numèrics i les etiquetes emmagatzemades en la variable 'cluster' del dataset les quals son l'entrada al mètode 'train_test_split' de la llibreria 'sklearn'. El que fa aquest mètode és separar el conjunt de dades que se li passa per entrada en 2 subconjunts, un de 'train' que servirà per entrenar el model i un altre de 'test' per evaluar el rendiment del model amb dades no vistes encara. Amdós subconjunts consten de 2 variables, una per les dades i l'altre per les etiquetes d'aquestes. La divisió es fa de manera aleatòria però es pot especificar la proporció de dades en els 2 subconjunts amb un parametre d'entrada 'test_size' i també es pot especificar un valor en el paràmetre 'random_state' fent

referència a una llavor (*seed*) per generar els números aleatoris, fa que es generi la mateixa subdivisió en les dades en múltiples execucions. Aquest paràmetre és util al comparar resultats en diferents models ja que es treballa sobre les mateixes dades. Aquest paràmetre també s'establirà a la crida dels diferents mètodes, així podem evaluar els diferents valors dels paràmetres sobre el mateix conjunt de dades aleatori.

Seguidament, apliquem el mètode TF-IDF-Vectorizer a les dades dels dos subconjunts ja que el model només treballa amb dades númeriques i fem la crida al model amb els seus respectius paràmetres ajustats per trobar la millor solució. Quan tenim el model ajustat, s'entrenen les dades numèriques amb el mètode 'fit', on el model utilitza algoritmes específics i tècniques d'optimització per ajustar els seus paràmetres interns de manera que minimitzi la diferència entre les prediccions del model i les etiquetes reals en les dades d'entrenament amb l'objectiu de trobar el conjunt òptim de paràmetres que permeti al model realitzar prediccions precises en noves dades no vistes.

I, quan ja tenim les dades entrenades, generem prediccions amb el mètode 'predict' a partir de dades les quals el model encara no ha vist. El model utilitza els paràmetres apresos durant l'entrenament per predir o inferir les etiquetes corresponents a aquestes característiques. Finalment, es calcula les diferents mètriques d'avaluació del mateix model amb diferents funcions.

El resultat és una estimació o predicció del valor objectiu o variable dependent associada a les dades d'entrada, és a dir, un valor de 'cluster' per cada text.

3.4.1 MultinomialNB

El primer model testejat és 'MultinomialNB' ja que te en compte els recomptes fraccionaris de TF-IDF. S'ha establert el paràmetre $alpha = 0.1$, un valor més alt de alpha resulta en una major suavització, la qual cosa redueix la sensibilitat a les característiques poc freqüents en el conjunt d'entrenament, així que li hem donat un valor baix perquè volem que tingui molt en compte les característiques que apareixen poc, no seran gaire útils.

No hem utilitzat cap paràmetre més perquè els predeterminats ja són una bona configuració. Tenim el $fit\_prior = True$ indicant que s'aprenen les probabilitats a priori, aquelles probabilitats inicials o les que s'asumeixen abans de tenir cap informació adicional. També tenim el paràmetre $class\_prior = None$, ja que no tenim cap matriu de probabilitats incial.

3.4.2 RandomForestClassifier

El següent model testejat és 'RandomForestClassifier' que combina múltiples arbres de decisió per a realitzar la classificació. Després de fer diverses proves amb diferents valors de paràmetres, s'ha trobat com a millor opció establir el paràmetre $n_estimators = 40$, que és el número d'arbres de decisió que es tindran. Com més gran sigui aquest valor, més robust i generalitzat sol ser el model, però també augmenta el temps d'entrenament. El valor 40 és l'equilibri entre millor resultat i millor rendiment.

No hem utilitzat cap paràmetre més perquè els predeterminats ja són una bona configuració. Tenim el $min\_samples\_split = 2$ indicant el mínim de mostres requerides per a què un node es divideixi, $criterion =' gini'$ que indica la funció que s'utilitza per evaluar la calidad d'una divisió als arbres, $max\_depth = None$ que és la màxima profunditat de l'arbre i si no hi ha cap, llavors els nodes s'expandeixen fins que totes les fulles siguin pures. Aquests són alguns dels paràmetres predeterminats que ens podem trobar en aquesta funció.

3.4.3 LogisticRegression

'LogisticRegression' s'utilitza per a predir la probabilitat de pertinença a una classe específica. S'ha establert el paràmetre $C = 2$ que és l'invers del paràmetre de regularització (tècnica utilitzada per evitar el sobreajust que agrega penalització a la funció de cost durant l'entrenament), un valor menor de C indica un marge gran i, per tant, vulnerable a errors. S'han fet diverses probes amb el paràmetre 'solver' i s'ha vist que s'arriba al mateix resultat si el valor és 'newton-cg', 'sag', 'saga' i 'lbfgs' (el predeterminat), així que no establim res. Amb els paràmetres de 'solver', l'unic paràmetre vàlid a 'penalty' és 'l2', que ve predeterminat també. Aquest paràmetre determina la tècnica que s'utilitzarà per fer la regularització.

No hem utilitzat cap paràmetre més perquè els predeterminats ja són una bona configuració. Tenim el $max\_iter = 100$ indicant el nombre màxim d'iteracions permeses per a la convergència. S'han fet proves amb diferents valors i s'ha vist que la seva variabilitat no afecta gaire en el resultat. $multi_class =' auto'$ que es pot indicar si volem ajustar el model per casos binaris o de multiclasse. Com tenim el paràmetre $solver =' lbfgs'$ estableix model de multiclasse. També tenim el paràmetre $class\_weight = None$, ja que no tenim cap matriu de probabilitats inicial i si establim 'balanced' dona resultats molt dolents, ja que no s'ha de balançar res perquè tenim el mateix volum de classes aproximadament.

3.4.4 LinearSVM

El següent model testejat és 'LinearSVM', el mètode de classificació lineal. Trobem alguns paràmetres semblants als altres models com per exemple 'class_weight', 'max_iter' i 'C', però en aquest últim el valor que millor funciona és $C = 0.5$ el qual és un valor òptim. Podem establir el parametre $tol = [0.8 - 1.6]$ que indica el nivell de tolerància per a determinar quan es considera que l'algorisme ha convergit i ha trobat una solució òptima. També tenim el paràmetre $multi\_class =' crammer_singer'$ on s'especifica l'enfocament molt més directe que 'ovr' utilitzat per a manejar problemes de classificació multiclasse. Per últim el paràmetre $loss =' hinge'$ controla la funció de pérdua del problema, busca maximitzar el marge entre les classes.

El paràmetre 'dual', per tractar amb formació dual o primal depenent del número d'atributs i mostres, i el paràmetre 'fit_intercept', que indica si s'ha d'ajustar o no el terme d'intercepció en el model, no afecten en la nostra classificació. Aquests són alguns dels paràmetres predeterminats que ens podem trobar en aquesta funció, hi ha més però he considerat que aquests són els mes comuns.

3.4.5 SGDClassifier

En 'SGDClassifier' tenim com a primer paràmetre $loss = squared\_hinge$ el qual és la funció de pèrdua del problema ....., la constant $alpha = 0.001$ que multiplica el terme de regularitzaió, $learning\_rate =' adaptive'$ per indicar que la tasa (velocitat) a la que un model aprèn s'ha d'adaptanr/regulant en funció de la pèrdua d'entrenament, $shuffle = False$ per no barrejar les mostres a cada iteració,

Com a paràmetres predefinits tenim 'penalty', 'max_iter', 'tol', 'n_iter_no_change' i 'epsilon'. Tots han estat explicats en models anteriors.

3.4.6 ANN, RNN i CNN

S'ha testejat la xarxa neuronal 'ANN' multicapa 'MLPClassifier' amb una sèrie de paràmetres predeterminats com $max\_fun = 15000$, $max\_iter = 200$, $n\_iter\_no\_change = 10$, $momentum = 0.9$, i $batch\_size =' auto$, entre altres. Els valors que poden prendre aquests paràmetres no afecten en la solució del millor resultat trobat.

La millor combinació trobada per aquest model és amb el paràmetre $activation =' tanh'$ indicant que s'utilitza la funció tangent hiperbòlica a la funció d'activació, $solver =' sgd'$ indicant que s'utilitza el descens del gradient (per minimitzar funció de costos), $alpha = 0.001$, $learning\_rate =' adaptive' o' constant'$, $learning\_rate\_init = 0.1$ per definir la tasa d'aprenentatge anteriorment nombrada, $shuffle = False$ per no barrejar les mostres a cada iteració i $tol = 1.2$ o més gran que indica el nivell de tolerància per a determinar quan es considera que l'algorisme ha convergit i ha trobat una solució òptima.

S'han fet proves amb 'RNN' i 'CNN', però no disposo de suficient espai a memòria per l'execució d'aquestes i es necessitava molt de temps, més de 2 hores per prova aproximadament. 'ANN' és una XN bastant ràpida i útil per aquest projecte, les altres 2 XN seran aplicades al seguir l'estudi i el projecte sobre aquest treball.

3.5 Predicció de la solució

Un cop ajustats tots els models amb els seus paràmetres corresponents, passem a evaluar quin és el millor model. Per fer-ho, utilizem unes funcions pròpies de la llibreria 'sklearn', un paquet anomenat 'metrics' [9].

Ens centrarem en: l'*accuracy*, amb quina freqüència és correcte al classificador; *confusion_matrix*, taula que permet visualitzar el rendiment del model ja que ensenya la quantitat de mostres classificades correctament i incorrectament per a cada classe; *f1-score* que combina la precissió (relació entre les prediccions correctes i el nombre total de prediccions correctes previstes) i el recall (relació entre les prediccions positives correctes i el nombre total de prediccions positives) en una sola mètrica per avaluar el rendiment del model; i el temps en que es tarda a fer la classificació.

En la següent taula es mostra els valors dels resultats dels models a comparar.

Model	Accuracy	F1-Score	Temps(s)
RandomForest	69.46%	62.81%	11.72
SVC	68.96%	63.26%	78.78
LogisticRegression	68.76%	62.28%	4.97
LinearSVC	68.53%	60.82%	11.81
SGDClassifier	67.62%	58.92%	79.35
MLPClassifier	66.57%	62.77%	43.48
MultinomialNB	64.11%	54.11%	0.03

Taula 2: TAULA DE RESULTATS

Com podem observar, no obtenim valors de percentatge de resultats molt alts, però ens centrem en l'anàlisi d'aquests. Tots tenen resultats semblants però si ens fixem en el temps d'execució d'aquests ens trobem amb una gran diferència. Així que el millor model serà el que dona bon resultat i té bon rendiment. En aquest cas tindriem 'RandomForest' i 'LinearSVC' al capdavant d'aquest rànking.

Si ens fixem en les *ConfusionMatrix* d'aquests 2 models, observem que obtenim resultats similars, però ara ens fixarem en *RandomForest* ja que és algo millor, figura 10. La diagonal d'aquesta matriu és on es reflexa la predicció del model i no està gaire definida. Trobem que la classe on predomina les bones prediccions és la 6, havent predit correctament 2748 mostres, sent aquesta també la classe on la resta de classes es confonen i també prediuen com 6. La classe 1 i 3 van seguides de la 6 en millor classe predita. 8

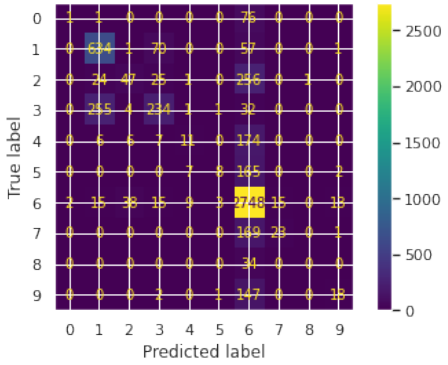

Fig. 10: *Confusion Matrix*

Quan ja hem triat el model que creiem que dona millors resultats, en aquest cas *RandomForest*, accedim al valor de les prediccions que ha generat i la solució amb més recompte d'aquell 'cluster' serà la solució que li assignem a aquell nou problema/incidència.

4 INTEGRACIÓ

Com a objectiu secundari es va establir la integració en una nova aplicació anomenada App Service Desk SEAT. És una aplicació creada pel departament on l'usuari emplena una sèrie de camps i crea un tiquet d'incidències. En aquest moment són enviats al servei de suport per a la seva solució i és guardat a la base de dades.

Aquest treball ha servit de base per entendre tot el procés que s'ha de dur a terme. Per poder acabar predint text (solucions) s'ha de tenir una bona base de dades neta, és a dir,

amb tots els processos d'eliminació de dades innecessàries per poder treballar correctament.

El meu estudi s'ha centrat en aplicar tècniques NLP a les dades i en aplicar-ho a models clàssics de classificació. He treballat també amb Xarxes Neuronals, però al tenir tantes dades i una màquina poc adecuada per l'entrenament d'aquestes, el rendiment no ha sigut el més adient. El següent pas és seguir treballant en aquest projecte, juntament amb més companys de l'empresa, i crear aquesta nova implementació que pugui generar nou text.

Unes idees d'implementacions per dur a terme són: abans de donar-li una possible solució, es faran una sèrie de preguntes (arbre de preguntes) a l'usuari que ajudin el model a augmentar l'èxit (percentatge de probabilitat final) de trobar la solució, detallar un llistat d'accions per a ajudar al servei de suport a acostar-se i arribar abans a la solució, i per últim crear el script que en executar-se solucioni el problema de l'usuari.

5 Conclusions

En aquest treball, s'ha abordat el desafiament de la classificació de text, amb l'objectiu de poder trobar una bona solució a les incidències. Al llarg d'aquesta recerca, s'han dut a terme diverses etapes crítiques, incloent-hi la comprensió exhaustiva de les dades i la implementació d'una rigorosa neteja de text. Aquestes tasques inicials han requerit un temps considerable, però han estat fonamentals per a garantir la qualitat i confiabilitat dels resultats obtinguts.

En primer lloc, s'ha realitzat una anàlisi detallada de les dades disponibles i ha permès definir estratègies adequades per a la neteja i preprocessament de les dades, amb la finalitat d'eliminar soroll, normalitzar el text i assegurar una representació homogènia.

L'etapa de neteja de dades ha implicat l'eliminació de paraules buides, signes de puntuació i caràcters especials. A més, s'han aplicat tècniques de stemming per a reduir les paraules a la seva forma base, així com l'eliminació de paraules irrellevants o poc informatives. Aquest procés ha requerit una iteració contínua per assegurar que les dades es trobin en un estat adequat per al modelatge.

Una vegada completada la neteja de les dades, s'ha procedit a la selecció i entrenament de models de classificació. S'han utilitzat algorismes d'aprenentatge automàtic i Xarxes Neuronals on s'han ajustat els paràmetres per a obtenir els millors resultats possibles. S'ha avaluat el rendiment dels models i s'han obtingut resultats prometedors que demostren la capacitat dels models per a classificar correctament els textos en les categories corresponents.

Es seguirà treballant en aquest tema per que pugui ser aplicat a diferents implementacions que SEAT S.A. té presents i a punt de començar.

En resum, aquest treball ha tractat de classificar text de manera automàtica. Hem estudiat les dades amb detall i hem netejat el text de forma minuciosa per garantir que els resultats siguin exactes i fiables en la classificació. Amb aquesta aproximació rigorosa i sistemàtica, hem aconseguit entendre bé les dades i obtenir resultats de qualitat en la classificació de text.

Agraïments

Agraïments a l'empresa SEAT S.A. per presentar-me aquesta enorme oportunitat de formar part dels nous projectes, agraïments especials al meu tutor de pràctiques i TFG desde l'empresa, Miguel Montano, que m'ha ajudat en tot el possible per tirar això endevant explicant-me detall a detall i ajudant-me en tot el necessàri, i per últim agraïments al meu tutor de TFG, Oriol Ramos, per tot el seguiment realitzat i les recomanacions i l'ajuda per poder millorar al màxim aquest treball.

Referències

[1] http://en.wikibooks.org/wiki/LaTeX Web que conté manuals per la creació d'un document en LaTeX. (Data darrer accés: 16/06)

[2] Iñigo Esteban. Aplicaciones de la inteligencia artificial en los negocios: ¿el mejor aliado de los emprendedores? (Data darrer accés: 27/04)

[3] Na8, Procesamiento del Lenguaje Natural. (NLP) (Data darrer accés: 12/05)

[4] Extracción de características e incrustaciones en el procesamiento del lenguaje natural, publicat com part del 'Blogatón de ciencia de datos'. (Data darrer accés: 10/05)

[5] AGRAWAL, Ayush; GUPTA, Utsav. Extraction based approach for text summarization using k-means clustering. International Journal of Scientific and Research Publications, 2014. (Data darrer accés: 30/05)

[6] Clustering: qué es y cuál es su uso en Big Data, article publicat a la web 'La universidad de internet'. (Data darrer accés: 1/06)

[7] https://scikit-learn.org/stable/ Web que proporciona l'explicació de com aplicar els diferents mètodes i funcions aplicats. (Data darrer accés: 12/06)

[8] Aravindpai Pai. CNN vs. RNN vs. ANN – Analyzing 3 Types of Neural Networks in Deep Learning. (Data darrer accés: 15/06)

[9] https://scikit-learn.org/stable/modules/model_evaluation.html Web que proporciona totes les mètriques per avaluar models. (Data darrer accés: 12/06)