NLP per a la resolució de tiquets

Irina Moreno Lahoz

Resum– Aquest projecte s'ha centrat 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. S'ha posat un fort èmfasi en l'establiment d'una estructura de dades adequada i en la seva neteja, així com en l'anàlisi i comparació dels resultats obtinguts. A més, s'ha considerat la implementació d'una solució basada en intel·ligència artificial, un àmbit molt present en l'actualitat. La primera part del projecte consisteix en la preparació de les dades proporcionades per l'empresa SEAT S.A. A continuació, a la segona part, s'ha treballat amb aquestes dades d'entrada utilitzant tècniques d'intel·ligència artificial per processar-les i obtenir una resposta precisa.

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

Abstract– This project has focused on text prediction, on being able to predict a solution to a new problem generated in order to help employees to solve it as simply and quickly as possible. A strong emphasis has been placed on the establishment of an adequate data structure and its cleaning, as well as on the analysis and comparison of the results obtained. Furthermore, the implementation of a solution based on artificial intelligence has been considered, an area that is very present nowadays. The first part of the project consists of the preparation of the data provided by the company SEAT S.A. Then, in the second part, we have worked with this input data using artificial intelligence techniques to process them and obtain an accurate answer.

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

• E-mail de contacte: irina.moreno@autonoma.cat
• Treball tutoritzat per: Oriol Ramos Terrades (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 es tracten són les incidències que tenen els treballadors sobre diferents aplicacions o sistemes que utilitzen. 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.

En aquest projecte es tractarà amb el dataset on es guarda tota la informació sobre els tiquets d'incidències.

Es vol realitzar una ajuda a l'empresa amb la gestió de la solució d'aquests tiquets d'incidències creant un model o un estudi que tingui com a resultat final una solució.

Les motivacions per fer aquest projecte són: el saber tractar amb un gran volum de dades, aprendre sobre NLP i possibles implementacions amb Xarxes Neuronals i poder aportar un mínim d'ajuda a una empresa tan gran i tan reconeguda com SEAT S.A.

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 d'algorismes 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, es vol crear 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 és que en introduir aquestes dades en l'aplicació apareixi 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 dut a terme un primer estudi.

2 PLANIFICACIÓ

Aquest projecte es troba dividit en un seguit de fases.

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

En segon lloc, el primer aspecte a tenir en compte com a començament d'aquest projecte és entendre les dades amb les 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 algorisme 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 per poder ajudar a l'algorisme a trobar millors solucions.

Com a tercer punt, s'estudiaran 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 classificador triat. A partir d'aquests resultats, també es pot saltar a les fases anteriors i fer canvis en la preparació de les dades o buscar algorismes 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* i mètriques 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

3 ESTAT DE L'ART

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. S'ha realitzat una cerca a internet de pàgines webs, llibres i treballs i s'ha trobat eines comunes que seran aplicades en aquest projecte.

S'ha trobat un fòrum a la pàgina web *Kaggle* que tracta sobre una competició de classificació de text [21]. Basantse en els treballs fets, s'utilitzen els algoritmes en aquest projecte els següents algorismes:

3.1 Clustering

El clustering [1] 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. Es vol utilitzar per subdividir les dades, ja que hi ha gran quantitat i així pot sorgir un model més precís.

Un dels algorismes és *K-Means* [2], 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.

3.2 Eines NLP i Extracció de característiques

El projecte es centra en el *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. Les que s'expliquen a continuació són les que més apareixen en els treballs.

Entre les tècniques més populars, i en les que el treball es centrarà, destaca el procés de *Tokenització* [4], que divideix un text en unitats més petites anomenades *tokens*. Els *tokens* poden ser paraules, signes de puntuació o números.

També s'estudiarà tècniques sobre com eliminar les *StopWords* [5], 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 són rellevants i així se centra en les paraules.

A més a més, es treballarà amb procesos com el *Stemming* i la *Lematització* [6]. 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'. *Snowball Stemmer* [17] és l'algorisme de *Stemming* on primerament va ser creat per utilitzar-se en anglès, però s'han creat adaptacions per altres idiomes, com l'espanyol.

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. Uns exemples són: NLTK, spaCy, Gensim, TextBlob, Stanford CoreNLP i el Natural Language Toolkit de Google, entre altres.

Les tècniques més comuns per extreure característiques del text en NLP amb les que s'ha treballat i finalment es triarà la que més s'ajusta al problema són:

Com a primera opció *CountVectorizer* [7], 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.

Com a segona opció la tècnica de TF-IDF, 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. Com més alt el valor generat, més rellevant és aquell terme. S'utilitza la classe *TfidVectorizer* [8].

3.3 Models de NLP i Xarxes Neuronals

A continuació s'expliquen alguns dels algorismes clàssics més comuns i dels que s'ha trobat més exemples a internet amb bons resultats.

El *MultinomialNB* [9] de *Naive Bayes* és un model probabilístic que es basa en el teorema de Bayes per a la classificació de text.

Pel que fa a *LogisticRegression* de *Linear Model* i *RandomForestClassifier* [10] de *Ensemble*, 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* [11] de *SVM* 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 [12] de *Linear Model* implementa models lineals regularitzats amb aprenentatge de descens de gradient estocàstic, només té en compte un punt aleatori al canviar de pesos.

En relació a les Xarxes Neuronals, es faran proves amb *ANN*, *RNN* i *CNN* [13]. Una xarxa neuronal artificial (ANN) és un grup de múltiples perceptrons/neurones en cada capa i les entrades a les capes tenen una direcció directe, cap endevant, 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 in-

terna 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.

4 METODOLOGIA

La següent imatge, figura 1, mostra l'arquitectura que es seguirà al projecte.

L'arquitectura consta de 4 grans etapes. La primera etapa consisteix en eliminar aquelles dades irrellevants i que no són útils en la nostra base de dades. Seguidament, es processen aquestes dades per acabar d'eliminar parts irrellevants, com signes de puntuació, ja que no transmeten cap tipus d'informació en el text. Com a tercera etapa, es fa l'extracció de característiques, adaptar el text generat fins ara a un format per a què els algoritmes d'aprenentatge automàtic puguin aprendre i generar resultats. A aquesta etapa se li aplica el *clustering* a totes les dades per establir etiquetes de classificació. I, per últim, amb una part de les dades entrenem i escollim el millor classificador que posteriorment serà testejat en un procés de validació.

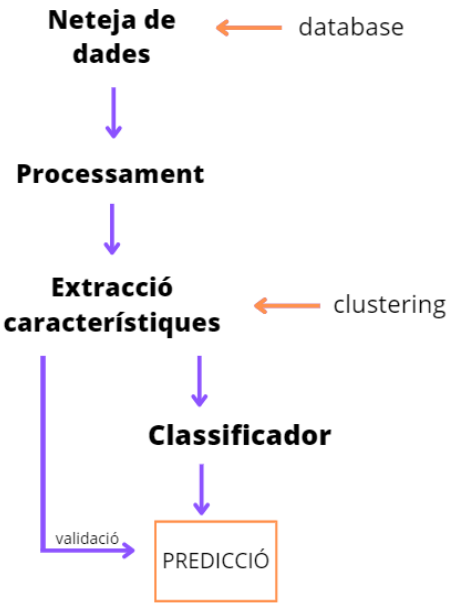

Fig. 1: Arquitectura i organització del treball

A continuació, s'aplica tot el que s'ha explicat anteriorment, afegint i adaptant els algorismes a les dades del projecte amb l'objectiu de crear una bona entrada pel model d'aprenentatge NLP i conseguir bons resultats. S'explicarà pas a pas la feina feta.

4.1 Neteja de dades

El dataset proporcionat 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).

Cada tiquet conté molts camps emmagatzemats, però els següents 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 dóna 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.

S'apliquen una sèrie de filtratges per eliminar dades irrellevants que no serveixin per entrenar el model.

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, s'eliminen aquelles entrades del dataframe que continguin el valor *NaN* en els registres mencionats anteriorment. També s'aplica un filtratge i s'eliminen aquells tiquets que no siguin en espanyol, ja que sinó generaria problemes a l'hora de predir correctament. Això es fa a partir d'un mòdul que detecta l'idioma del text que li arriba com entrada a la seva funció, si aquest idioma és diferent a l'espanyol, s'elimina aquell registre del dataset.

En segon lloc, es treballa amb 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, es procedeix a fer una neteja i s'eliminen les parts generades automàticament i, posteriorment, s'eliminen aquelles solucions on el text es massa curt. S'aplica el filtratge on només se seleccionen les solucions les quals la longitud del text supera els 29 caràcters.

Després d'aplicar aquests filtres, el dataset resultant és de mida 301045, quedant així reduit al 33.75% aproximadament. És un número molt baix, però és perquè no es manté

cap ordre o organització quan es guarden les dades, i sovint es guarden sense tenir en compte com estan estructurades. Com s'ha vist, això té repercussions quan es vol treballar amb aquestes dades més endavant.

Quan ja es té tota la base de dades preparada, es procedirà a realitzar les agrupacions per poder tractar amb les dades que són similars.

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 es té, simplement es considera el nom de cada *CI* en el paràmetre per fer l'agrupació. Es crearà un model diferent per cada *CI*.

Es visualitza per cada *CI* la distribució de tiquets en un histograma mostrat en la figura 2 després de fer el primer pas de la neteja.

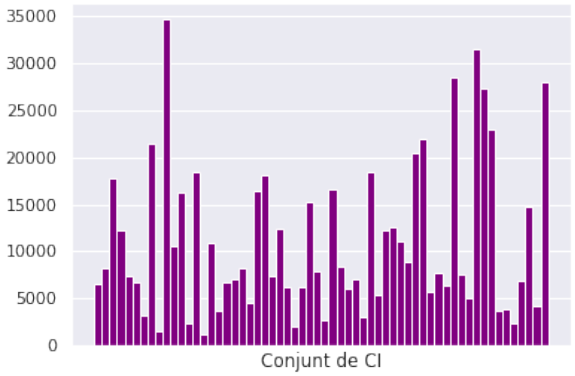

Fig. 2: Histograma CIs

Com es pot 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 18083 registres en total com es pot veure a la figura 3.

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

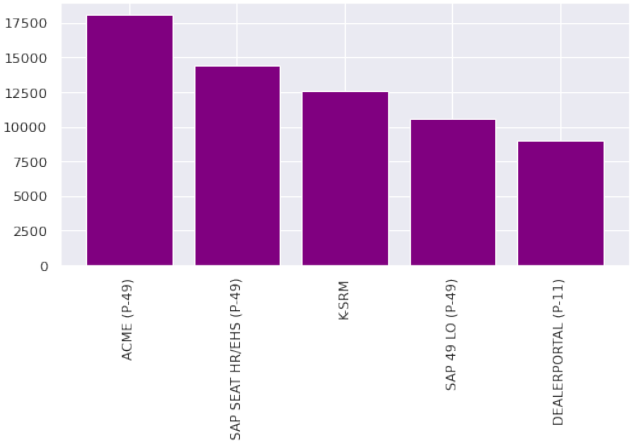

Fig. 3: Recompte aparicions del top 5 CI

Com a següent agrupació, es basarà en la variable anomenada 'Service Tower', categories on es classifiquen les diferents aplicacions que estan relacionades o pertanyen al mateix departament, que es troba en un altre dataset, el qual consta de moltes variables però les que s'utilitzen són '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, s'aplica la funció *merge* [14] 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 es té en compte les files on hi hagi valors coincidents entre datasets) de la columna 'CI Name'.

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

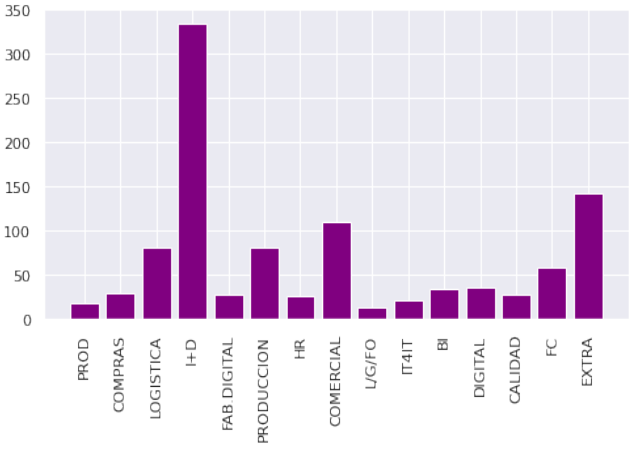

Fig. 4: CI en Service Towers

En el gràfic anterior es veu que el 'Service Tower' amb més dades és 'I+D', però es seleccionen 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. Es treballarà sobre el dataset de 'CI_Name' 'ACME (P-49)'.

4.2 Processament del text

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

S'inicia aplicant l'algorisme de tokenitzar important el mòdul *Word Tokenize* [15]. El que fa aquest algorisme exactament és separar el text paraula per paraula i ho emmagatzema en una llista.

Seguidament, s'eliminen 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* [16], es troba en el cas de ser un signe de puntuació i s'elimina el caràcter.

Com a següent pas, s'eliminen les *StopWords*. S'importa una llista de paraules sense gran valor, com per exemple 'el','la', 'de', 'en', 'por', 'con', 'para', entre d'altres; de l'algorisme *Corpus* [5] i es busca i s'elimina si alguna d'aquestes paraules es troba en el text de 'Title' i 'Description'. A aquesta llista s'afegeix manualment una sèrie de paraules que s'han considerat 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*.

S'afegeix 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 algorismes anteriors per poder treballar millor amb ell i s'observa 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.

L'algorisme de *Lematització* descrit anteriorment és útil per a normalitzar les paraules a la seva base canònica o de diccionari. Aplicant aquest algorisme a les dades, es considera que no hi ha gran canvi en la forma de les paraules. Es veu 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 l'algorisme de Lematització

Finalment, es decideix treballar amb l'algorisme *Snowball Stemmer*. El resultat acaba sent la mateixa llista que ja es tenia, però amb les paraules actualitzades, com es veu 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 l'algorisme *Snowball Stemmer*

4.3 Extracció de característiques

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 en el procés de crear agrupacions per la classificació.

Es comença extraient característiques, utilitzat per a extreure informació significativa del text i representar-la en forma de vectors numèrics. S'aplica aquest procés a les variables 'Text' i 'Solution', ja que contenen la informació per entrenar el model.

S'utilitza *Count Vectorizer* i *Tf-idf Vectorizer*. Els 2 generen com a resultat una matriu de valors numèrics representant les dades. Per saber quin dels 2 dóna millor resultat, es comprova i es testejen en la crida dels diferents classificadors.

4.4 Clustering

Quan tot el text es troba 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 realitzat una sèrie d'agrupacions a partir de l'algorisme K-Means.

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. Aquesta figura representa les agrupacions sobre l'algorisme *Tf-idf Vectorizer*. Aquesta gràfica s'ha realitzat amb PCA [18], un mètode que serveix per trobar variabilitat en les dades i poder-les representar reduint la seva dimensió.

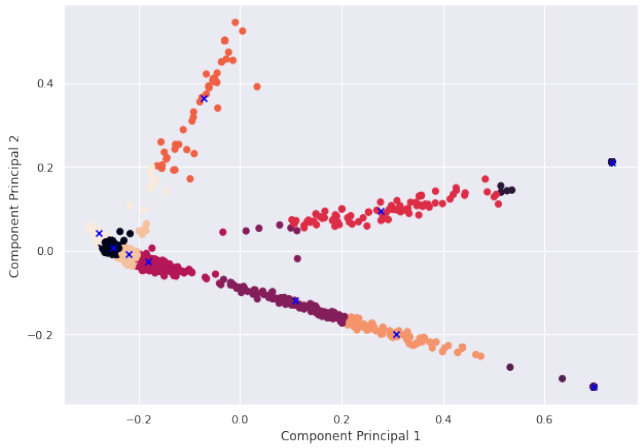

Fig. 9: Clustering amb K-means (PCA)

A partir d'aquestes agrupacions, s'assigna una etiqueta de 'cluster' a cada registre, una variable que servirà posteriorment com a etiqueta de classificació per entrenar al model. Aquesta etiqueta s'estableix a totes les dades del dataset.

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 dues vegades, però tots 2 textos volen comunicar el mateix i es troben en el mateix 'cluster'. Aquesta informació resulta útil per a la classificació de solucions, ja que el model proporciona com a resultat un valor numèric de 'cluster', i el resultat final que serà proporcionat és el text amb el recompte més elevat.

5 Experiments

Un cop ja es tenen les dades preparades, la següent fase és la d'entrenar els models explicats anteriorment a l'estat de l'art per veure quin dóna millors resultats. També es farà una comparativa amb ells mateixos amb diferents paràmetres d'entrada.

L'estructura que es seguirà és la mateixa per tots els models. Es comença amb les dades respresentades amb vectors numèrics i les seves respetives etiquetes. Aquestes dades es divideixen de manera aleatòria en 2 subconjunts, un d'entrenament i un per evaluar el rendiment del model, i seguidament s'apliquen per separat els 2 algorismes per extreure característiques per tal de decidir quin és millor. Tot seguit, es defineixen els diferents classificadors i els seus meta paràmetres. s'ha utilitzat l'algorisme *Grid Search CV* [19] i les proves manuals, prova a mà de diferents valors i executar cada cop, per escollir els millors paràmetres.

Per últim, s'entrenen les dades i es generen prediccions. 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. S'evalua el model amb diferents mètriques, com per exemple la precissió o el rendiment, comparant-les amb dades que el model encara no ha vist.

Després de diverses proves, l'extractor de característiques que dóna millor resultat és *Tf-idf Vectorizer*, ja que s'obtenia un valor de precissió del 68.96% mentre que amb *Count Vectorizer* s'obtenia un 49.74%. S'ha testejat sobre l'algorisme *Logistic Regression*, un model que té un rendiment bó per poder fer diferents proves en un temps curt.

5.1 MultinomialNB

El primer model testejat és *Multinomial NB* ja que té en compte els recomptes fraccionaris de TF-IDF. S'ha establert el paràmetre alpha = 0.1, un valor més alt d'alpha resulta en una major suavització, redueix la sensibilitat a les característiques poc freqüents en el conjunt d'entrenament, així que s'ha assignat un valor baix perquè es vol tenir en compte les característiques que apareixen poc, no són considerades gaire útils.

No s'ha utilitzat cap paràmetre adicional perquè els predeterminats ja són una bona configuració. S'inclou 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 es troba el paràmetre class_prior = None, ja que no existeix cap matriu de probabilitats incial.

5.2 RandomForestClassifier

El següent model testejat és *Random Forest* 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 s'ha utilitzat cap paràmetre més perquè els predeterminats ja són una bona configuració. Es disposa del paràmetre 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 es troben en aquesta funció.

5.3 LogisticRegression

Logistic Regression 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 s'estableix 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 s'ha utilitzat cap paràmetre més perquè els predeterminats ja són una bona configuració. Es disposa de 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 es vol ajustar el model per casos binaris o de multiclasse. El paràmetre solver = 'lbfgs' estableix model de multiclasse, class_weight = None indicant que no hi ha cap matriu de probabilitats inicial i si s'estableix a 'balanced' dóna resultats molt dolents, ja que no s'ha de balançar res perquè existeixen el mateix volum de classes aproximadament.

5.4 LinearSVM

El següent model testejat és *Linear-SVM*, l'algorisme de classificació lineal. S'han descobert 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. S'estableix 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 es troba 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 classificació. Aquests són alguns dels paràmetres predeterminats més comuns que es poden trobar en aquesta funció.

5.5 SGDClassifier

En *SGD-Classifier* s'identifica com a primer paràmetre loss = squared_hinge fent referència a la funció de pèrdua del problema i penalitza els errors més grans, 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 existeixen 'penalty', 'max_iter', 'tol', 'n_iter_no_change' i 'epsilon'. Tots han estat explicats en models anteriors.

5.6 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 es disposa de suficient espai a memòria per l'execució d'aquestes i es necessita molt de temps, més de dues hores per prova aproximadament. *ANN* és una Xarxa Neuronal bastant ràpida i útil per aquest projecte, les altres 2 Xarxes Neuronals seran aplicades al seguir l'estudi i el projecte sobre aquest treball.

5.7 Predicció de la solució

Un cop ajustats tots els models amb els seus paràmetres corresponents, es passa a evaluar quin és el millor model. Per fer-ho, s'utilitzen unes funcions d'un paquet anomenat *Metrics* [20].

L'evaluació es focalitza 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; *precision* que és la relació entre les prediccions correctes i el nombre total de prediccions correctes previstes; 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	Precision	Temps(s)
RandomForest	69.46%	65.18%	11.72
SVC	68.96%	63.836%	78.78
LogisticRegression	68.76%	64.05%	4.97
LinearSVC	68.53%	63.38%	11.81
SGDClassifier	67.62%	62.46%	79.35
MLPClassifier	66.57%	61.77%	43.48
MultinomialNB	64.11%	56.91%	0.03

Taula 2: TAULA DE RESULTATS

Com s'observa, no s'obtenen valors de percentatge de resultats molt alts, però el focus es troba en l'anàlisis d'aquests. Tots tenen resultats semblants però si fixant-se en el temps d'execució d'aquests, s'identifica una gran diferència. Així que el millor model serà el que dóna bon resultat i té bon rendiment. En aquest cas es detecta *Random Forest* i *Linear-SVC* al capdavant d'aquest rànking.

Es fa l'anàlisis de les *Confusion Matrix* d'aquests 2 models i s'oberva que s'obtenen resultats similars, però l'anàlisi es fixa en *Random Forest* ja que s'oberven resultats superiors, figura 10. La diagonal d'aquesta matriu és on es reflexa la predicció del model i no està gaire definida. La classe on predominen 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.

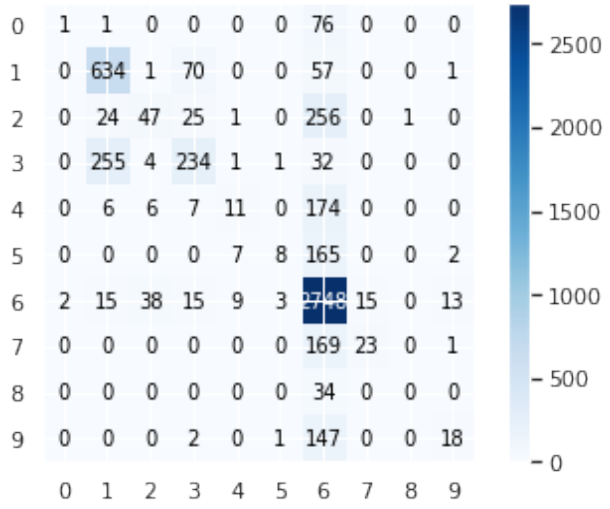

Fig. 10: *Confusion Matrix*

Quan ja s'ha triat el model que dóna millors resultats, en aquest cas *Random Forest*, s'accedeix al valor de les predicccions que ha generat i la solució amb més recompte d'aquell 'cluster' serà la solució que se li assigna a aquell nou problema/incidència.

6 INTEGRACIÓ

Com a objectiu secundari, explicat a la introducció d'aquest treball, 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.

Aquest estudi s'ha centrat en aplicar tècniques NLP a les dades i en aplicar-ho a models clàssics de classificació. S'ha 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 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.

7 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. S'ha partit desde 0 d'una base de dades amb una cantitat elevada de dades, i a partir d'aquí s'ha estudiat i s'han près les decissions necessàries per treballar amb aquesta base de dades.

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. Amb el pas del temps s'ha vist que també pot ser aplicat la generació de text automàtic per la creació de la solució, però no es dispossava del temps ni recursos necessaris, així que s'estudiarà i s'aplicarà com a continuació del projecte.

En resum, aquest treball ha tractat de classificar text de manera automàtica. S'ha estudiat les dades amb detall i s'ha netejat el text de forma minuciosa per garantir que els resultats siguin exactes i fiables en la classificació. Amb aquesta aproximació rigorosa i sistemàtica, s'ha 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 aquesta enorme oportunitat de formar part dels nous projectes, agraïments especials al tutor de pràctiques i Treball de Fi de Grau de l'empresa, Miguel Montano, que ha ajudat en tot el possible per explicar detall a detall i ajudar en tot el necessàri, i per últim agraïments al tutor de Treball de Fi de Grau de la univeristat, Oriol Ramos, per tot el seguiment realitzat i les recomanacions i l'ajuda per poder millorar al màxim aquest treball.

REFERÈNCIES

[1] 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)

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

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

[4] https://nlpcloud.com/es/nlp-tokenization-api.html Web que parla sobre la Tokenització. (Data darrer accés: 12/05)

[5] Chetna Khanna, Text pre-processing: Stop words removal using different libraries. (Data darrer accés: 14/05)

[6] Saumyab271, Stemming vs Lemmatization in NLP: Must-Know Differences. (Data darrer accés: 14/05)

[7] https://towardsdatascience.com/basics-of-countvectorizer-e26677900f9c Web que tracta sobre CountVectorizer. (Data darrer accés: 10/05)

[8] Unnikrishnan C S, How sklearn's Tfidfvectorizer Calculates tf-idf Values. Article que parla sobre Tfidfvectorizer. (Data darrer accés: 11/05)

[9] Victor Roman, Algoritmos Naive Bayes: Fundamentos e Implementación. Article que dóna informació de MultinomialNB. (Data darrer accés: 11/05)

[10] Asel Mendis, Random Forest vs Logistic Regression in Python. Article que compara informació de Logistic Regression i Random Forest. (Data darrer accés: 06/06)

[11] https://stackoverflow.com/questions/27912872/what-is-the-difference-between-svc-and-svm-in-scikit-learn Web que tracta sobre LinearSVC i SVC. (Data darrer accés: 06/06)

[12] Vinay Patlolla, How to make SGD Classifier perform as well as Logistic Regression using parfit. Article que parla sobre SGD Classifier.(Data darrer accés: 12/06)

[13] Aravindpai Pai. CNN vs. RNN vs. ANN – Analyzing 3 Types of Neural Networks in Deep Learning. (Data darrer accés: 11/06)

[14] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.merge.html Documentació sobre Merge. (Data darrer accés: 12/06)

[15] https://www.nltk.org/api/nltk.tokenize.word_tokenize.html Documentació sobre Word tokenize. (Data darrer accés: 12/06)

[16] https://docs.python.org/es/3/library/string.html Documentació sobre els mètodes de la llibreria String. (Data darrer accés: 12/06)

[17] Snowball: A language for stemming algorithms, Documentació sobre Snowball Stemmer. (Data darrer accés: 14/06)

[18] https://statologos.com/analisis-de-componentes-principales-2/?utm_content=cmp-true Web que explica el l'anàlisis de PCA. (Data darrer accés: 15/06)

[19] https://panjeh.medium.com/scikit-learn-hyperparameter-optimization-for-mlpclassifier-4d670413042b Web que explica i mostra un exemple per trobar els millors meta paràmetres de cada model. (Data darrer accés: 22/06)

[20] https://scikit-learn.org/stable/modules/model_evaluation.html Web que proporciona totes les mètriques per avaluar models. (Data darrer accés: 23/06)

[21] https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/overview Web que conté un conjunt de treballs sobre NLP. (Data darrer accés: 10/06)