Informe de la Pràctica 1

Lluc Furriols, Pau Prat

4 de març de 2024

Universitat Politècnica de Catalunya

Grau en Intel·ligència Artificial

Processament del Llenguatge Humà

Resum

Aquest document correspon a l'informe de la primera pràctica de l'assignatura Processament del Llenguatge Humà del Grau en Intel·ligència Artificial de la Universitat Politècnica de Catalunya (UPC).

En aquest informe es detallen els passos que hem seguit per tal d'implementar l'identificador d'idiomes que se'ns proposava en codi Python, així com el posterior anàlisi dels resultats obtinguts.

Índex

1 Implementació

1.1 Preprocessament

En primer lloc, vam crear la funció de preprocessament del text, anomenada *preprocessar_text(text)*. En l'enunciat se'ns demanava que el preprocés havia de realitzar el següent: eliminar els dígits del text, convertir tot el text a minúscula, substituir els espais en blanc continus per un de sol i concatenar totes les frases amb un espai doble al mig.
Per tant, vam utilitzar la llibreria *regular expression* per a fer-ho correctament. A més, vam decidir eliminar també els caràcters especials del text, conservant els signes de puntuació, interrogació i exclamació, ja que vam pensar que podien ser avantatjosos a l'hora d'identificar l'idioma, doncs si ens trobem en el text amb el signe "¿", per exemple, gairebé segur que es tractarà d'un text en castellà.
Per altra banda, per tal de separar les frases amb un espai doble (després d'haver substituït els espais blancs continus per un únic espai en blanc), vam decidir que dividiríem les frases en quan hi hagués un punt seguit d'un espai (`frases = text.split('. ')`). Per tant, la funció textit-preprocessar_text() retorna el text que se li dona com a paràmetre sense digits, sense els caràcters especials mencionats, en minúscula i sense més d'un espai en blanc seguit excepte els espais dobles que diferencies les diferents frases del text.

1.2 Funcions necessàries

Així doncs, un cop creada la funció del preprocessament, vam implementar les següents funcions:

- `get_trigrams(text)`: Genera una llista de trigrames a partir del text. Per a fer-ho, utilitza la funció *ngrams* de la biblioteca *NLTK*. A més, és en aquesta funció que hem usat la tècnica de padding, doncs s'afegeixen dos espais en blanc tant al principi com al final del text. D'aquesta manera, el primer trigrama d'una frase sempre serà `( , ,caràcter)` i l'últim serà `(caràcter, , )`, la qual cosa ajuda a que el programa entengui millor on comencen i acaben les paraules, tant si són del conjunt de train com si són del conjunt de test.

- `remove_infreq_trigrams(trigram_counts, threshold=5)`: Aquesta funció filtra els trigrames infreqüents del diccionari *trigram_counts*, que per cada idioma, conté les freqüències de cada trigram existent. En aquest cas, la funció retorna un nou diccionari que només inclou els trigrames que apareixen com a mínim 5 vegades per cada llenguatge. Tot i així, l'hem implementat per a què es pugui fer servir amb altres llindars, ja que si és el paràmetre *threshold* el que determina a partir de quina freqüència es conserven o no els trigrames.

- `lid(count, total, lambda_, total_add)`: Representa la tècnica de suavitzat de la Llei de Lidstone. Aquesta funció permet assignar una petita probabilitat a trigrames que no han estat observats en el conjunt de dades de train, lo qual millora la capacitat de generalització a dades no vistes anteriorment.

- `extract_language(file, train=True)`: Aquesta funció identifica l'idioma d'un fitxer a partir del seu nom. Com que els diferents arxius tenen com a títol el nom de l'idioma seguit de _trn.txt (train) o _tst.txt (test), la funció utilitza expressions regulars per tal de saber, per cada fitxer, quin és el seu idioma.

1.3 Partició en train, validació i test

Per al projecte hem dividit el text dels fitxers _trn en train/validació amb una relació de 70/30, és a dir, el 70% de les dades s'han utilitzat per a l'entrenament del model i el 30% restant s'han fet servir per a la validació.
El conjunt de dades de train s'ha utilitzat per generar el compte de trigrames, que és una part crucial del nostre model de reconeixement de idiomes. Aquest compte de trigrames l'utilitzem més endevant per predir de quin idioma pertany el trigrama.
D'altra banda, el conjunt de dades de validació l'hem utilitzat per ajustar els hiperparàmetres del nostre model. La partició de les dades en train validació la fa una funció que té el nom de `train_val()`:.

Per tal de processar, els fitxers d'entrenament i prova, primer hem llegit els fitxers d'entrenament (`text = f.read()`) per tal de dividir-los ja en train i validació, i posteriorment hem llegit els arxius de prova línia a línia, de tal manera que cada línia dels arxius de test es preprocessarà de manera independent i es crearan els trigrames pertinents. D'aquesta manera, després podrem obtenir de manera molt més senzilla l'accuracy del model, així com la matriu de confusió, doncs podrem anar predint cada línia per separat i avaluant si ha estat una predicció encertada o no, obtenint així el nombre total de prediccions correctes i incorrectes per a cada idioma.

Tot seguit, hem creat el diccionari `trigram_counts`, que serveix per a emmagatzemar el nombre de cops que apareix cada trigrama per a cada idioma, és a dir, les claus del diccionari són els 6 idiomes diferents (`'deu', 'eng', 'ita', 'fra', 'nld', 'spa'`), i els valors són diccionaris en què les claus són els diferents trigrames, i els valors són les freqüències dels trigrames en el idioma corresponent.

1.4 Trobar els hiperparàmetres òptims

Per triar els millors hiperparametres hem provat diferents combinacions d'aquests. Hem considerat que era més adient utilitzar els mateixos hiperparàmetres en tots els idiomes ja que així el model de classificació d'idiomes funciona millor, ja que tots els idiomes utilitzen la mateixa escala.
La funció evaluate retorna la accuracy, així doncs, hem provat diferents hiperparàmetres a la funció evaluate i ens hem quedat amb la combinació que ha donat millor accuracy.
Les lambdes que hem provat en el conjunt de validació han estat 0.1, 0.5 i 1. Aquest paràmetre controla el pes que es dóna a la suavització de Lidstone en el càlcul de les probabilitats dels trigrames. Quan lambda és més petit (com 0.1), significa que la suavització té un impacte més gran, ja que està afegint una petita quantitat a tots els comptes dels trigrames, independentment de si s'han vist abans o no. Això pot ser útil per evitar problemes de sobreajustament i millorar la generalització del model. Per contra, quan lambda és més gran (com 1), la suavització té menys impacte, ja que es divideix el nombre de vegades que s'ha vist cada trigrama pel nombre total de trigrames. Això pot ser adequat quan tenim moltes dades i volem confiar més en les freqüències observades dels trigrames.
La tria de les B ha estat de la següent manera: Volíem provar 3 B diferents i avaluar quina funcionava millor.

- `Trigrames observables en el text`: La primera B que hem fet candidata consisteix en calcular el nombre total de trigrames únics en els textos de cada idioma. Hem utilitzat tant els textos d'entrenament com els de prova per a cada idioma per a aquest càlcul. Utilitzant

els dos textos disponibles (train i test) ens assegurem observar el major nombre de trigrames possibles. Finalment hem fet la mitjana de tots els idiomes.

- `Combinacions amb caràcters diferents`: La segona B la hem calculat amb el nombre de caràcters diferents per a cada idioma. Hem establert un criteri que cada caràcter hauria de ser present com a mínim en un 0,1% del total de caràcters del text. A continuació, hem elevat a 3 el nombre de caràcters diferents, ja que això ens permet considerar totes les combinacions possibles de trigrames que poden ser formades pels caràcters. Hem realitzat aquest càlcul per a cada idioma individualment i, finalment, hem calculat-ne la mitjana.
- `Valor entremig les dues B anteriors`: Per últim, la tercera B la hem calculat com la mitjana de les dues B anteriors. Aquest enfocament l'hem fet perquè aquesta B estigui entre els dos extrems representats per les dues B anteriors. Considerem que aquesta nova B hauria de ser més gran que la primera, ja que és possible que hi hagi trigrames que no s'hagin vist en el conjunt de dades, i alhora més petita que la segona, ja que alguns trigrames poden ser molt poc probables d'aparèixer junts, com ara `'vfv'`, `'ppp'`, o `'zxz'`.

Després d'haver provat aquest hiperparàmetres en el conjunt de dades de validació, els hiperparàmetres que han obtingut més accuracy han estat `lamba = 0.1 i B = 19974.833` (la qual correspon a la primera B testada)

1.5 Avaluació del model

Per tal d'avaluar el model, hem creat la funció `evaluate()`, la qual té com a objectiu calcular l'accuracy de les prediccions realitzades pel model i generar una matriu de confusió. Els paràmetres d'entrada de la funció són els següents:

- `X`: Els textos que seran classificats pel model. En el cas de cercar els hiperparàmetres òptims, aquest paràmetre serà `X_val`, mentre que per avaluar el model final serà `X_test`.
- `y`: Una llista amb els idiomes reals de cada text (com abans, `y_val` o `y_test`).
- `lambda_, B`: Els paràmetres de la funció `lid()`, els quals són essencials ja que dins d'aquesta funció es fa una crida a `lid()`, i per tantt és necessari passar-li a la funció `evaluate()` els paràmetres amb els que cridarà a `lid()`.
- `conf_matrix`: Un paràmetre booleà que indica si es vol generar i mostrar la matriu de confusió de les prediccions del model.

La funció, per tant, comença definint un conjunt d'idiomes (`languages`) a partir de la llista `y_train`, que recordem que conté els 6 idiomes diferents. A continuació, inicialitza un diccionari per emmagatzemar els resultats de la matriu de confusió i dos comptadors per a les prediccions totals i correctes.
Per cada text a classificar (`X`) i el seu idioma corresponent (`y`), la funció realitza els següents passos:

1. Calcula els trigrames del text utilitzant la funció `get_trigrams()`.
2. Inicialitza variables per a la probabilitat màxima i l'idioma predit.
3. Per a cada idioma, calcula la probabilitat de que el text pertanyi a aquest idioma basant-se en la freqüència dels seus trigrames, utilitzant els paràmetres `lambda_` i `B` per tal de cridar a la funció `lid()`.

4. Actualitza la matriu de confusió basant-se en l'idioma real i l'idioma predit.
5. Incrementa el total de prediccions i, si l'idioma predit coincideix amb l'idioma real, també incrementa el comptador de prediccions correctes.

Així doncs, finalment la funció retorna l'accuracy del model, que la calcula com la proporció de prediccions correctes entre el total de prediccions.

1.6 Joc de proves

Finalment, hem implementat unes línies de codi que et permeten provar el model de detecció d'idiomes que hem entrenat. Pots introduir una paraula, una frase o un text en qualsevol idioma i el sistema et retornarà una estimació de l'idioma en què es troba el text proporcionat.
Per utilitzar aquesta funcionalitat, només has de substituir la variable `text = ""` amb el text que desitgis provar. Un cop hagis introduït el text, el model intentarà determinar l'idioma amb més probabilitat basant-se en el model de classificació d'idiomes que hem entrenat prèviament.

2 Anàlisi dels Resultats

Un cop executada la comanda `evaluate(X_test, y_test, best_params['lambda_'], best_params['B'], conf_matrix=True)`, hem obtingut una accuracy de **0.99656** i la matriu de confusió de la figura 1. Com es pot observar, els resultats obtinguts són realment satisfactoris, ja que com hem pogut veure l'accuracy és proper a 1, la qual cosa vol dir que la majoria de frases dels arxius de prova han estat predites correctament.

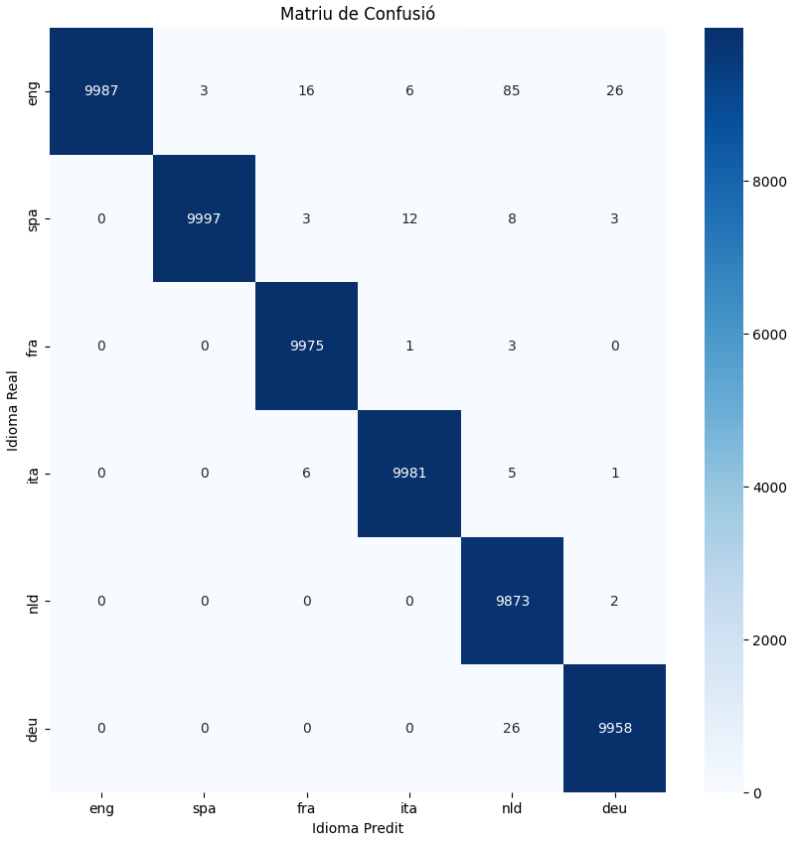

Figura 1: Matriu de Confusió

En quant a la matriu de confusió, aquesta mostra el nombre de vegades que els idiomes han estat correctament i incorrectament predits pel model per cada una de les frases dels arxius de test. Els valors diagonals de la matriu, per tant, representen les prediccions correctes de cada idioma, mentre que els valors fora de la diagonal indiquen els diversos errors de predicció (falsos positius i falsos negatius).
Analitzant les dades de la matriu, podem veure que la majoria de les prediccions incorrectes s'agrupen en uns pocs idiomes, per lo que podem deduir que el model pot estar confonent idiomes amb característiques lingüístiques similars. Per exemple, observem que l'anglès (eng) ha estat confós amb el neerlandès (nld) en 85 ocasions i amb l'alemany (deu) en 26 ocasions.
Tot i això, el nombre d'errors és realment baix en comparació amb el nombre de prediccions cor-

rectes, com ho demostra l'elevat valor de l'accuracy. Això indica que el model té una capacitat bastant significativa per a distingir entre els diferents idiomes del conjunt de dades.

Per tal d'analitzar més en profunditat aquests resultats, hem decidit calcular també per a cada idioma, l'accuracy, precisió, recall i f1-score, i hem obtingut la següent informació:

Idioma	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Anglès (eng)	99.77	100.00	98.66	99.32
Castellà (spa)	99.95	99.97	99.74	99.86
Francès (fra)	99.95	99.75	99.96	99.85
Italià (ita)	99.95	99.81	99.88	99.84
Neerlandès (nld)	99.78	98.73	99.98	99.35
Alemany (deu)	99.90	99.68	99.74	99.71

Taula 1: Mètriques obtingudes per a cada idioma

Un cop obtinguts els resultats anteriors, recordem què vol dir cada mètrica:

- Accuracy: proporció de prediccions correctes (tant positives com negatives) respecte al total de prediccions.
- Precisió: proporció de veritables positius respecte al total de prediccions positives (la suma de veritables positius i falsos positius).
- Recall: proporció de veritables positius respecte al total d'exemples reals positius en el conjunt de dades (la suma de veritables positius i falsos negatius).
- F1-Score: mitjana harmònica de la precision i el recall.

Per tant, podem treure certa informació interessant, com ara que quan el sistema prediu que un text està en anglès, sempre és correcte, sense falsos positius, ja que l'anglés té una precisó del 100%. També podem veure que els idiomes amb un percentatge més alt d'accuracy són el castellà, francès i italià (amb un 99.95%), i que l'idioma amb un major Recall és el neerlandès, és a dir, té molts pocs falsos positius (en concret, únicament 2). En canvi, és l'idioma amb menys precisió (amb diferència), per lo que és el que té una major proporció de falsos positius, en què el sistema classifica erròniament textos d'altres idiomes com si pertanyessin a aquest idioma en particular.

Tot i això, en general tots els idiomes han donat uns resultats realment positius en totes les mètriques anterior, ja que en tots els casos -excepte la precisió del neerlandès- els valors obtinguts han estat superiors al 99%, per lo que podem concloure que té un molt bon rendiment en la tasca d'identificar idiomes, per la qual cosa estem realment satisfets amb el treball fet durant aquesta primera pràctica.