Pràctica 1: Modelo de lenguaje para la detección de idioma

Taisiia Prymak

Jiahui Chen

Marzo/2025

Processament del Llenguatge Humà

Intel·ligència Artificial

Universitat Politècnica de Catalunya

Índice

Introducción

El objetivo de este trabajo es diseñar e implementar un detector de idiomas, más en concreto, poder identificar textos en castellano, inglés, italiano, francés, neerlandés y alemán. Dados unos textos de entrenamiento se van a dividir en trigramas para poder asignar probabilidades a los trigramas observados en cada frase del fichero test.

La fórmula del Maximum Likelihood Estimator (MLE) permite calcular la probabilidad de que una cierta secuencia pertenezca a un idioma y se define como:

$$P^T MLE\,(ej\,) \;=\; c^T(ej\,) \,/\, Nt$$

La probabilidad de la secuencia se aproxima, asumiendo la independencia, al producto de la probabilidad de todos los trigramas. A la hora de procesar los trigramas de la secuencia en el fichero test y para calcular la probabilidad de cada uno, se va a dividir el número de veces que aparece el trigrama entre el número total de trigramas del fichero de entrenamiento, después los resultados se multiplican entre sí para obtener la probabilidad.

Pero este enfoque presenta el problema de tener un trigrama en la secuencia de test que nunca ha aparecido en el fichero de entrenamiento lo que al poner el numerador a 0 resulta en una probabilidad nula para toda la secuencia. Por esta razón, se va a aplicar una técnica de smoothing, la Ley de Lidstone (LID), en lugar de asignar probabilidad cero a los trigramas no vistos, el smoothing reserva algo de probabilidad para todos los posibles trigramas en forma de parámetro lambda.

$$P^T LID\,(ej\,) \;=\; (c^T(ej\,) \;+\; \lambda) \,/\, (Nt \;+\; B\lambda)$$

A la hora de agregar probabilidad extra en el numerador, es necesario ajustar el denominador. En la fórmula LID, B es el número de trigramas únicos en el espacio de nuestro modelo y λ es la cantidad de probabilidad "extra" que agregamos a cada uno, como la cantidad total de conteo extra agregada es B·λ este valor se pasará dividiendo.

Preprocesamiento

Para obtener una mayor eficiencia con los modelos es necesario aplicar preprocesamiento sobre los datasets que nos proporcionan como entrada, entre los cuales encontramos el corpus de train y los ejemplos de test. Para cada idioma tenemos un fichero de train y de test, que contiene miles de oraciones o frases numeradas numéricamente.

Train corpus

Los ficheros denominados como *trn.txt* son los ficheros de entrenamiento. Los ficheros de entrenamiento necesitan preprocesamiento para poder obtener trigramas más útiles.
Para todos los idiomas existen carácteres comunes como lo son las comas, los dos puntos, números, símbolos del porcentaje, etc. Se van a eliminar estos símbolos ya que no aportan información útil para poder distinguir los idiomas entre sí. Además, existen otros factores que molestarán al modelo, estos son los múltiples espacios, ya que pueden crear trigramas vacíos o poco informativos.
También, es importante mencionar que los trigramas que aparecen menos de cinco veces van a ser eliminados ya que, a parte de que puede tratarse de errores ortográficos o de palabras atípicas, si hay muchos trigramas con frecuencias muy bajas, el smoothing de la fórmula de Lidstone puede terminar asignando demasiada probabilidad a eventos raros, porque el número de trigramas distintos (B) crece mientras que el numerador queda muy pequeño, empeorando así la calidad del modelo.

Test examples

Los ficheros denominados como *tst.txt* son los ficheros de test que a diferencia de los ficheros de train, que pueden representar cada fichero como una única entrada para entrenar los modelos, hay que separar las distintas oraciones que contiene cada fichero para que los modelos las pueda identificar de manera separada. Para lograr esto sabemos que solo las frases están separadas por dobles espacios, con

esta característica es fácil separar las frases. Después de separar las frases como diferentes elementos de una lista también hay que añadirles dos espacios al inicio y al final de cada oración para acordar con los trigramas que se han construido a partir de las mismas condiciones.

Estructura del código

El programa ejecuta dos grandes pasos principales que son el Entrenamiento del modelo y la Predicción del idioma.

El entrenamiento y la justificación de hiperparametros elegidos

Durante el entrenamiento, la función *train()*, preprocesamos los textos de entrenamiento y definimos los hiperparametros *N, B* i $c(ej)$, ya mencionados, para cada idioma. Para ello, necesitamos dividir el texto en trigramas usando la librería *nltk*, cuyo resultado ya nos proporciona un diccionario donde la clave es el trigrama y el valor es las veces que aparece este trigrama en el texto, así obtenemos el hipermarámetro $c(ej)$ directamente. *N* es el número de trigramas en total con lo cual, sumamos todos los $c(ej)$ obtenidos anteriormente. El parámetro *B*, por definición, son todos los trigramas posibles de un idioma, que serían 256^3, pero incluye muchas combinaciones no coherentes con el idioma. Además, por su magnitud, predominará sobre el factor *counter.* Por esta y muchas razones más, es imposible obtener una *B* ideal. Dado que también estamos limitando la *N* a trigramas con frecuencia mayor o igual que 5, estimaremos la *B* como trigramas posibles diferentes en el corpus de entrenamiento con una frecuencia mayor o igual que 5, en nuestro caso, sería la longitud del diccionario donde almacena todos los trigramas posibles.
Por último, hay que destacar que definimos el valor de λ recomendado, es decir que tiene como valor 0.5.

La predicción

Durante la predicción se itera por cada secuencia de cada fichero test. Para cada idioma se calcula la probabilidad de pertenecer a este de cada secuencia aplicando la fórmula de *LID* conocidos los parámetros *N, B* y el diccionario de ocurrencias de cada trigrama obtenidos en la fase anterior. Se guardan todas las probabilidades en un diccionario para después sacar el máximo y el idioma que corresponde a esa probabilidad es el resultado final de la detección para este ejemplo.

Evaluación de los modelos

Para evaluar los modelos, se han utilizado diversas métricas, destacando principalmente el *accuracy* y la matriz de confusión. El accuracy es una métrica que mide la proporción de predicciones correctas realizadas por el modelo respecto al total de casos. Nuestro modelo ha logrado un valor de accuracy muy elevado, lo que indica un alto nivel de aciertos en sus predicciones. Además, otras métricas complementarias, como la precisión, el recall y el F1-score, respaldan estos resultados, demostrando que el modelo no solo acierta en términos generales, sino que también maneja adecuadamente tanto los falsos positivos como los falsos negativos.

Accuracy	0.9563
Precision	0.9648
Recall	0.9562
F1	0.9576

Observamos en la matriz de confusión que el modelo clasifica correctamente la mayoría de los ejemplos. Aunque la magnitud de los errores es pequeña en comparación con la cantidad de aciertos, el modelo falla en algunas ocasiones al predecir el neerlandés. En particular, se observa que confunde otros idiomas con el neerlandés, especialmente el inglés.

Esto puede ser indicio de que son dos lenguas que se parecen bastante o, al menos, el corpus de test tienen alguna semejanza. De momento no tenemos más medidas o evidencias que afirman que estas dos lenguas son parecidas, se podría hacer un análisis más profundo de la relación entre estos dos idiomas, pero estamos limitados por los corpus de entrenamiento.

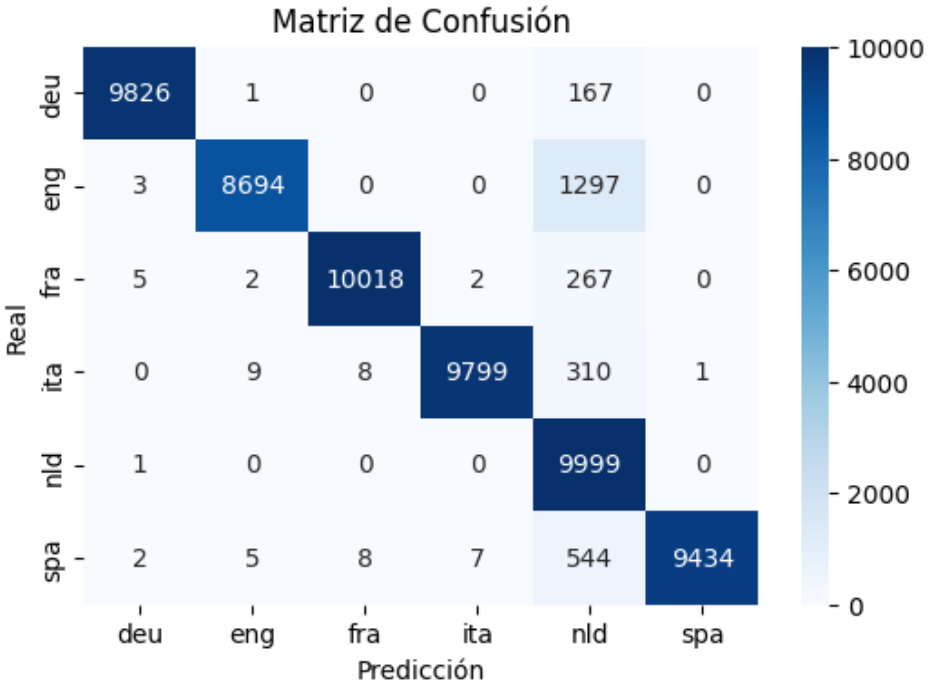

Figura 1: Matriz de confusión del modelo

Análisis de los errores

Para analizar, ¿Por qué algunos ejemplos han fallado con el método de la versemblanza? Se han agrupado las oraciones que no ha acertado el modelo en un diccionario, donde las claves son los idiomas originales y el valor las frases de ese idioma mal classificadas.

Se podría inspeccionar la longitud de los ejemplos fallidos y averiguar si la longitud influye en la calidad de la classificación. La siguiente gráfica resume las longitudes de los ejemplos detectados erróneamente, desde frases de una única longitud a 1300, pero observamos que hay un *gap* (entre 600 y 1200) en la distribución donde no hay frases mal clasificadas. Dado la variedad de los ejemplos de entrada, es poco probable que no existan frases de esa longitud, por tanto, podemos concluir

que las frases de longitud media se clasifican mejor comparado con frases más cortas o largas.

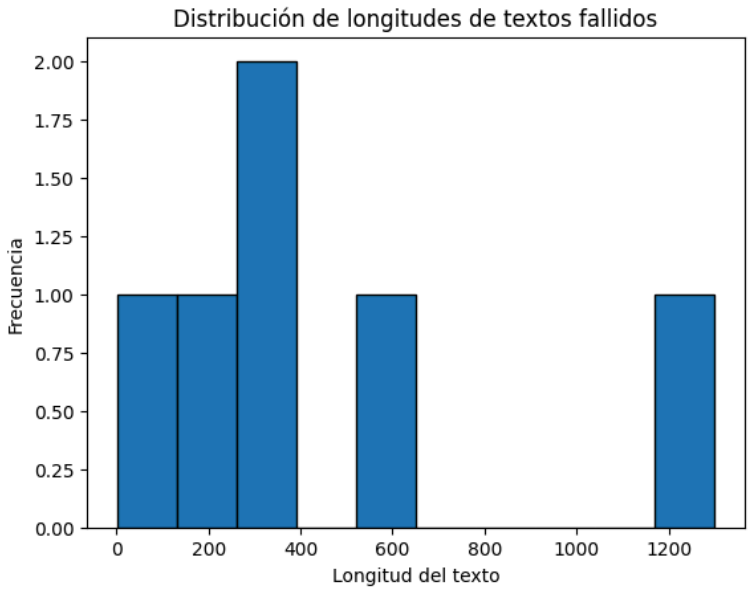

Figura 2: Histograma de las longitudes de los textos fallados

Otro enfoque posible sería analizar qué idiomas han sido más confundidos. En el siguiente *barplot* destaca el inglés como la lengua más confundida con otros idiomas. Podría ser razonable dado que el inglés no tiene estructuras sintácticas o morfológicas complejas y por tanto puede repetir el mismo patrón en otros idiomas.

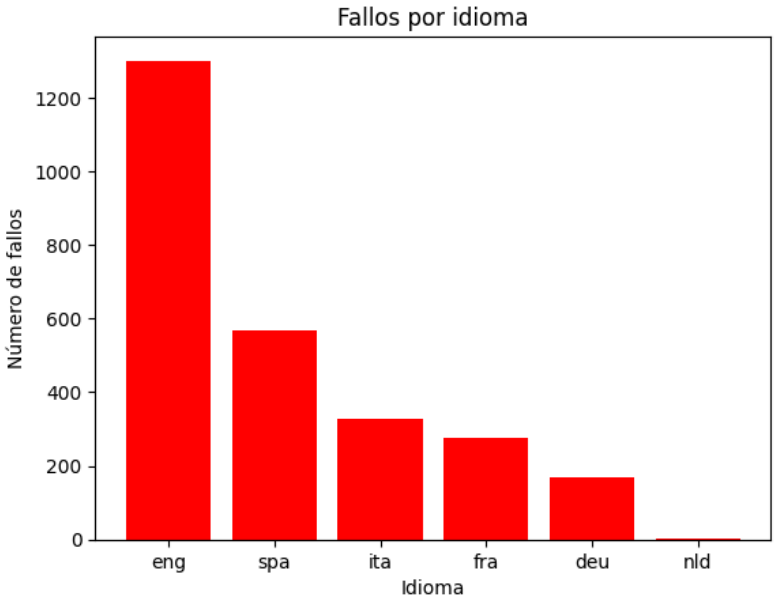

Figura 3: Barplot de ejemplos fallados por idioma