uc3m | Universidad Carlos III de Madrid

Análisis Temático de Proyectos de Investigación

Tratamiento de Datos

Eduardo Gómez González
Oier Huici Itarte

27 de diciembre de 2023

Índice general

1. Introducción

En este proyecto se va a resolver una tarea de aprendizaje sobre documentos textuales, en este caso, se trabajará sobre documentos que recogen los resúmenes de los proyectos de investigación financiados por la Comisión Europea dentro del Programa Horizonte 2020 (H2020).

El conjunto de datos seleccionado se proporciona en el fichero 'projects.xlsx' y consta de los 35.378 proyectos dentro del Programa H2020, contando con información relevante para cada uno de los proyectos.

2. Proyecto Básico

2.1. Implemetación de pipeline para preprocesado de textos

Con el fin de obtener una buena representación de cualquier técnica de vectorización de texto, se han realizado con en primer lugar las siguientes técnicas de preprocesado.

Text Wrangling

Para evitar posibles problemas en la vectorización, primero **eliminaremos caracteres simbólicos** que no nos aportan información, como pueden ser: *.,!?@**. También las **URLs** se eliminan antes de la vectorización, ya que no aportan información relevante al contexto general del texto y pueden causar problemas.

Tokenization

Se segmenta el texto en palabras, que pasarán a llamarse **tokens**. Este proceso puede separar por símbolos de puntuación, frases o elementos con significado entre ellos del texto.

Homogeneization

En este paso, se juntan palabras que son semánticamente equivalentes y se representan todas como una única. Antes de realizar este proceso, se ponen en minúscula todas las palabras a modo de paso preliminar.

En este trabajo se ha utilizado la técnica *lemmatization*. Esto se debe a que a la salida se sigue teniendo palabras completas, pudiendo ser más útil que la opción de utilizar *stemming*.

Cleaning

Se eliminan las palabras que son irrelevantes para el significado semántico del texto. Estas son las llamadas *"Stopwords"*, las cuales aparecen a menudo en contextos muy diferentes pero sin dar contexto, como pueden ser artículos o pronombres.

Con esto, se crea un ***pipeline*** que realiza las técnicas de preprocesado descritas como introducción al proceso de vectorización de textos. Esto optimiza el resultado obtenido a posteriori, al eliminar palabras o caracteres sin utilidad alguna que pueden proporcionar errores durante el proceso de vectorización.

En este trabajo, por simplicidad, se ha optado por utilizar una función ***prepare_data***, donde se realizan todas las técnicas de preprocesado, en vez de utilizar la herramienta *pipeline* de Scikit-learn.

2.2. Representación vectorial de los documentos

TF-IDF

TF-IDF ofrece valores más altos a los términos que aparecen frecuenctemente en uno de los documentos, pero no tan frecuentemente en el resto de ellos.

Está basado en el modelo **BoW** (Bag-of-Words), por lo que este modelo de vectorización no tiene en cuenta la posición en la que aparece el término dentro del texto, solamente cuenta el número de veces.

Para la implementación de esta vectorización se han seguido los pasos del notebook $Text\ Vectorization\ I$ del tema de NLP de la asignatura.

Word2Vec

Word2Vec trabaja de forma diferente. En este caso, tras una iteración en la que se tiene en cuenta la posición y el contexto de cada término, se convierte cada término en un vector, de forma que la proximidad entre vectores determinará la similitud entre las palabras.

Para la implementación de esta vectorización se han seguido los pasos del notebook $Text\ Vectorization\ II$ del tema de NLP de la asignatura.

Embeddings calculados a partir de modelos basados en transformers

Estos modelos preentrenados permiten codificar cada término en función del resto de la secuencia, teniendo en cuenta el contexto de cada palabra. Para la tokenización con ellos se ha utilizado la librería *Hugging Face* y se han seguido los pasos desarrollados en el notebook de la asignatura referente a transformers.

Se han hecho pruebas con dos modelos diferentes, que presentan diferencias en su entrenamiento y arquitectura:

1. **BERT**
2. **RoBERTa**

Finalmente, viendo los distintos resultados para cada uno de ellos, se ha decidido utilizar el modelo *RoBERTa* en este proyecto debido a sus mejores prestaciones.

2.3. Entrenamiento y evaluación de modelos

Se ha optado por probar dos métodos distintos de regresión para evaluar cuál es la mejor opción a la hora de obtener el número de patentes y publicaciones, dependiendo de la representación vectorial escogida.

Red neuronal implementada con PyTorch

En este caso, se ha implementado una red neuronal a partir de la biblioteca PyTorch. Dicha red se ha desarrollado siguiendo una estructura piramidal de capas lineales, introduciento entre ellas capas de activación de tipo *ReLU* y finalizada en una única salida debido a que se desea resolver una tarea de regresión. Como función de pérdidas se ha utilizado el error cuadrático medio (MSE) y el optimizador escogido ha sido de tipo *Adam*.

Además, con el objetivo de reducir el sobreajuste en los datos de entrenamiento, se han utilizado las técnicas de *Dropout* para aleatorizar el uso de neuronas en las diferentes capas y *Early stopping* para detener el entrenamiento con el mejor modelo para el conjunto de validación posible.

Para el entrenamiento de la red, se han seguido los pasos habituales en el uso de esta librería implementados en un bucle cuyo número de repeticiones determina la variable *epochs*:

1. Evaluación del modelo para el conjunto de entrenamiento.
2. Obtención de las pérdidas de la predicción del modelo para el conjunto de entrenamiento.
3. Cálculo de gradientes y actualización de parámetros.
4. Monitorización de las pérdidas para el conjunto de validación.

Técnica implementada de la librería Scikit-learn: Random Forest

Dentro de la variedad de opciones que se tienen dentro de la librería Scikit-learn, se ha optado por abordar la regresión mediante un modelo **Random Forest**. Había más opciones disponibles, como pueden ser K-NN, SVM o XGBoost, pero aunque el coste computacional pueda ser mayor, se ha escogido Random Forest frente al resto de modelos por su robustez, versatilidad y precisión. Como metodología de validación se ha utilizado la función ***cross_val_score*** de Scikit-learn. Esto lo que hace es utilizar cada vez diferentes datos como train/test y el resultado final resulta ser una aproximación entre todos ellos.

Además, también se han utilizado métodos de validación cruzada para optimizar los parámetros de la regresión y obtener el menor error posible, la función ***GridSearchCV*** de Scikit-learn. Para este trabajo, los parámetros obtenidos han sido *n_features* = 120 y *max_features* = 'sqrt'.

2.4. Evaluación de resultados

En primer lugar, cabe destacar que como medida para evaluar el error cometido en la tarea de regresión se ha utilizado el Error Cuadrático Medio (MSE, Mean Squared Error). De cara a la presentación de los resultados, se debe tener en cuenta que la variable objetivo, el número de publicaciones y patentes, se ha normalizado a una media nula y varianza unidad. También se presenta el RMSE (Root Mean Squared Error) debido a su fácil interpretación al encontrarse en las mismas unidades que la variable objetivo.

Además, el conjunto de datos se ha dividivo en una proporción del 80 % para entrenamiento y un 20 % para test.

Con respecto al uso de la red neuronal, cabe destacar en este punto que para cada conjunto de datos se han probado distintas configuraciones de valores de *learning rate*, *weigth decay*, porcentajes de dropout en las diferentes capas e incluso se ha añadido a continuación de la primera capa lineal una capa del tipo $BatchNorm1d$ para reducir el sobreajuste. Finalmente, los valores escogidos han sido:

$$Learning\,rate = 0{,}01 \qquad Weight\,decay = 1 * 10^{-5} \tag{1}$$

En la siguiente figura se presentan los resultados obtenidos para cada tipo de vectorización y ambas técnicas de regresión.

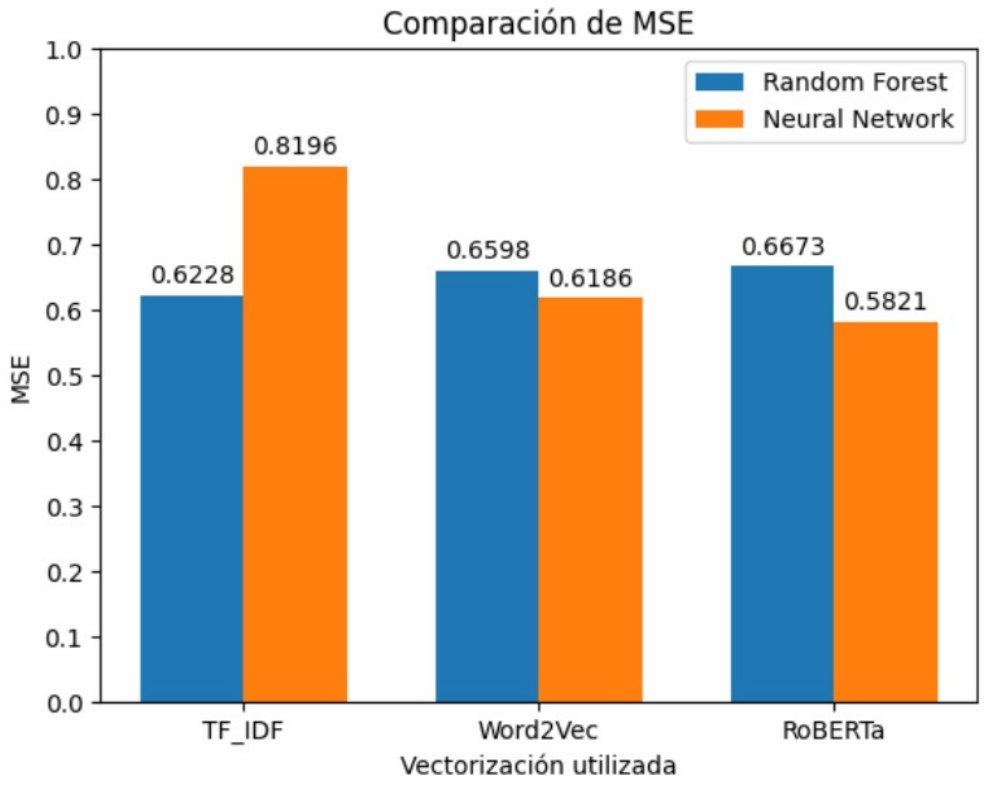

Figura 1: Comparación del MSE para el proyecto básico

Del gráfico anterior se pueden obtener varias conclusiones:

- Cierta mejora en la resolución del problema por parte de la red neuronal al utilizar las vectorizaciones Word2Vec o con el modelo RoBERTa.
- Error notablemente mayor en el caso de la utilización de TF-IDF y la red neuronal. Esto se puede deber al hecho de haber utilizado matrices dispersas para la representación de los textos y a una no correcta adaptación de estas al modelo implementado en *PyTorch*.

Al tomar la raiz cuadrada del MSE, **RMSE** (Root Mean Square Error), se vuelve a la escala original de la variable dependiente, lo cual facilita la interpretación. Por la tanto, el gráfico 1 puede derivar en el siguiente:

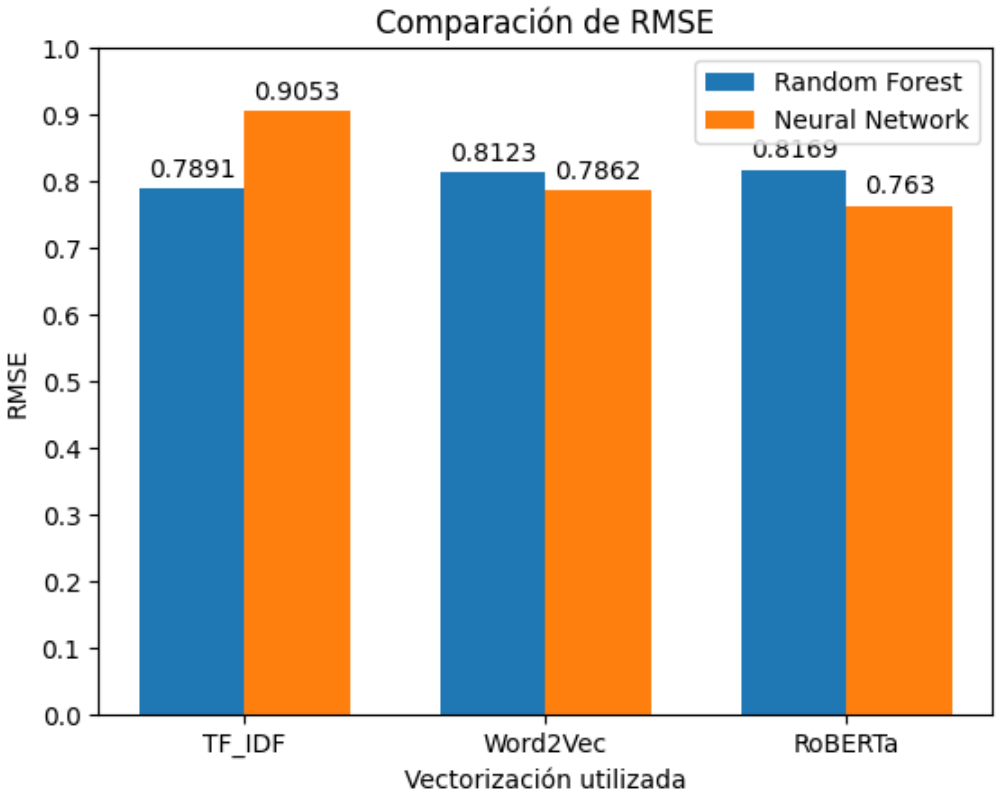

Figura 2: Comparación del RMSE para el proyecto básico

Al utilizar la raíz cuadrada, atenuamos el impacto de los valores extremos de la regresión. Comúnmente, en problemas de regresión se utiliza RMSE en vez de MSE debido a su interpretabilidad y a que se puede hacer una comparación más directa al estar en la misma escala.

3. Extensión

3.1. Elección de la técnica de extensión

En este caso, se han decidido explorar el uso de tesauros en el preprocesado de los textos así como la utilización de características adicionales del dataset para mejorar las predicciones en la tarea de regresión.

El uso de tesauros consiste en añadir palabras semánticamente relacionadas a las presentes en los textos con el objetivo de aportar mayor variabilidad léxica y enriquecer la vectorización de los documentos al disponer de un contexto más amplio. En nuestro caso se ha escogido el tesauro $WordNet$ al ser uno de los más conocidos y utilizados y se han añadido sinónimos de cada palabra. El uso de esta herramienta no se ha evaluado junto a la utilización de $transformers$ ya que estos tienen su propio proceso de tokenización y vectorización.

En cuanto a las características adicionales añadidas, de todas las columnas que aparecen en *projects.xlsx* se han escogido las siguientes:

- totalCost: Coste total del proyecto.
- ecMaxContribution: Financiación de la UE.
- rcn: Número de control del registro.
- coordinatorCountry: País que coordina el proyecto.

Se podrían haber escogido otras columnas distintas pero, como se va a ver en el posterior apartado, simplemente utilzando estas marcadas aquí ya se mejora considerablemente la salida.

3.2. Evaluación de resultados

En primer lugar, se presentan los resultados de MSE obtenidos al añadir al conjunto de datos las características adicionales mencionadas.

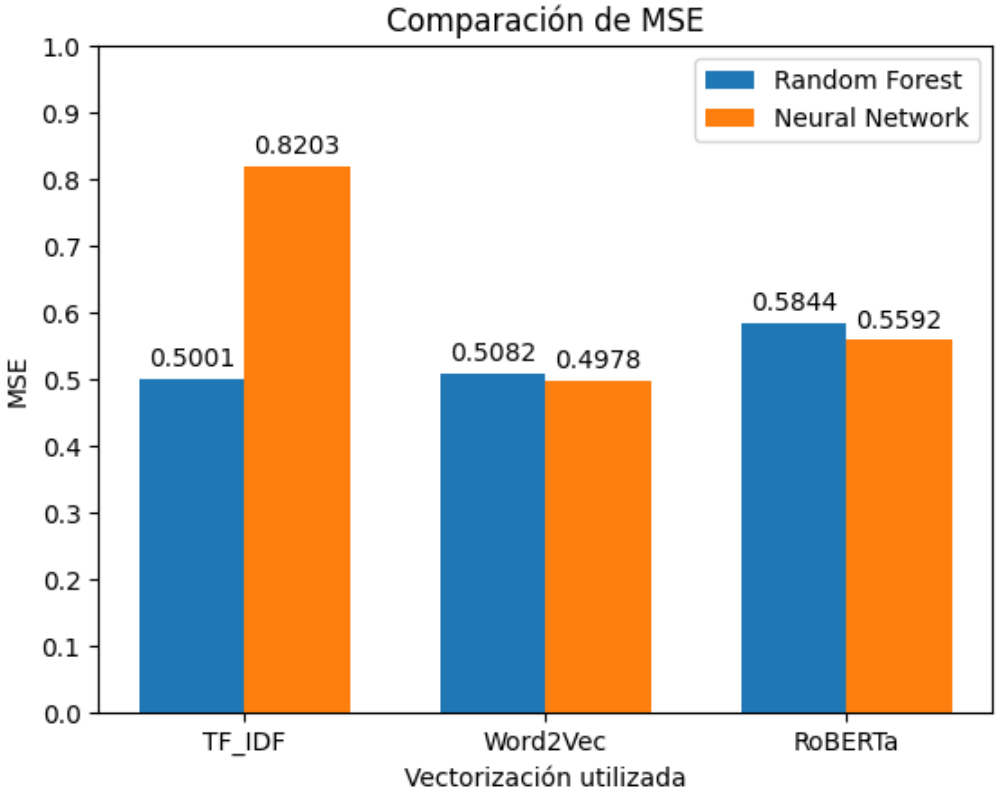

Figura 3: Comparación del MSE tras añadir características adicionales

Como podemos ver en la gráfica 3, el introducir más columnas en la regresión mejora la precisión del modelo. Esto es algo lógico, ya que al introdcir información relevante, el modelo es capaz de predecir con mayor exactitud.

Aún se podrían introducir más columnas de las disponibles en *projects.xlsx*, como por ejemplo *euroSciVocCode*, y ser capaces de mejorar la precisión del modelo.

Con respecto a la utilización de *tesauros*, se han evaluado los resultados utilizando la red neuronal. Sin embargo, no se ha apreciado una mejora considerable con su utilización, lo que se puede deber o bien a una mala implementación de esta técnica en el código o bien a que para este caso específico su uso no es de gran beneficio. En concreto, los valores de MSE obtenidos son:

- TF-IDF: 0.7643
- Word2Vec: 0.7007

4. Reconocimiento de autorías

En primer lugar, cabe mencionar la utilización de los notebooks de la asignatura referidos al tema de NLP como base para la realización del preprocesado de textos y la obtención de las representaciones *TF-IDF*, *Word2Vec* y con el uso de *transformers*.

Por otra parte, para la implementación de la red neuronal se han utilizado fragmentos de código del notebook de la asignatura referente a dicho tema, así como de los siguientes tutoriales:

- Implementing Regression Problem using Torch
- Building a Regression Model in PyTorch