### **Goal**

1. Implementar un sistema que permita realizar la clasificación de una noticia en 7 diferentes categorías.
2. Realizar un sistema de recomendación según relevancia de cada una de las noticias, permitiendo.

### **Scope**

* Creación modelo clasificador a partir de lexicon.
* Creación modelo clasificador a partir de entrenamiento con noticias pre clasificadas.
* Creación de algoritmo de mención de usuario.
* Creación de algoritmo de mención del sector.
* Creación de modelo de análisis de sentimiento de noticias.
* Algoritmo recomendador de noticias.

# Diseño Técnico

### **Propuesta de diseño Clasificación:**

### **Lexicon Classificator**

En resumen, se construyó lo siguiente:

1. Entrenamiento de modelo para extraer palabras similares a partir de un contexto y una palabra objetivo. (LexiconCategories.train\_model).
2. Comparación de uso de palabras en cada una de las noticias y clasificación de noticias a partir de un Lexicón (bolsa de palabras).

### 

### **Modelo WORLD 2 VEC**

Basados en los antecedentes, se concluye que dicha implementación teórica sería correcta aplicarla, desde y cuando, se tenga un modelo que obtenga una bolsa de palabras (lexicon) a partir de textos que hablen sobre los temas objetivo que obtenga dichas palabras a partir de un contexto (corpus) y de las palabras objetivo. Se propone el uso del modelo Word2 Vector de Logisim [1]

Un modelo World 2 Vec que se basa en reconocer palabras cercanas entre sí a partir de vectores con un contexto que hace que la palabra tenga un sentido principal. En otras palabras, este modelo trata de aprender las palabras de contexto para cada una de las palabras objetivo.

El entrenamiento se transmite, por lo que las “frases” pueden ser iterables, leyendo datos de entrada desde el disco o la red sobre la marcha, sin cargar todo el corpus en la RAM.

### **Preparación de datos**

Limpieza realizada y sugerida para a cada una de las noticias y los títulos de las noticias:

1. Paso de mayúsculas a minúsculas
2. Eliminar acentos y espacios innecesarios entre las palabras.
3. Realizar streaming según parámetros específicos.
4. Eliminación de textos nulos.

Asegurándonos que la noticia no pierda su sentido real y verificando que la noticia aún tenga el sentido y coherencia de la noticia original.

Se recomienda no quitar stopWords para que el contexto no cambie. Ciertas fuentes recomiendan la eliminación de los StopWords pero en nuestros estudios de resultados, se llegó a la conclusión que es mejor el uso de las mismas.

### **Entrenamiento**

Se definió las palabras objetivo las diferentes clasificaciones que se les puede dar a cada noticia (Innovación, Macroeconomía, sostenibilidad, regulaciones, etc) para poder tener un contexto adecuado se tomaron todas las noticias que Bancolombia proveyó para realizar el ejercicio y se entrenó el modelo basado en las frases y contextos encontrados en todas las noticias.

**Nota**: Si se desea obtener el modelo correr el main.py

### **Resultado**

Se seleccionaron 100 palabras similares (Este número se puede sintonizar mejor con mayor número de iteraciones) usando el objeto **wv.most similar** (**Lexicoon Categories.py**) y con ellas cada una de su porcentaje de similitud a la palabra objetivo.

**Nota:** Si se desea obtener el modelo entrenado y los respectivos lexicons de cada categoria por favor correr main.py

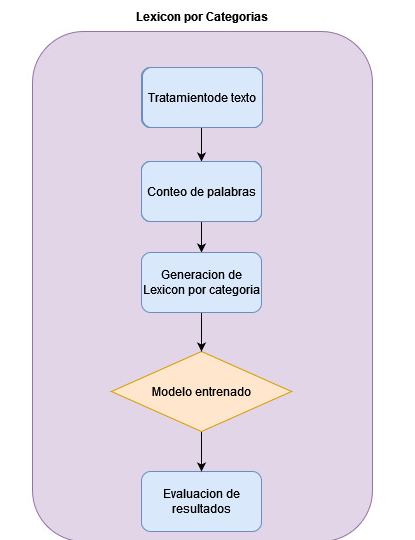
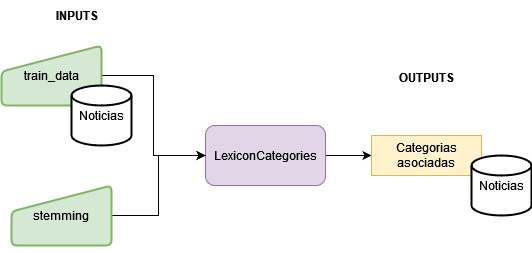
Se procede a realizar un conteo de palabras que aparezcan dentro de cada una de las noticias, basados en cada uno de los lexicones para cada categoría. Realizando este procedimiento también se tiene en cuenta el porcentual de precisión que tiene la palabra para ponderar con la cantidad de palabras y así generar un peso más preciso. 

Figura 1. Descripción clase Lexicon Categories.

**Modelo Spanish RoBERTa**

En resumen se construyó lo siguiente:

1. Generación de un DataSet de noticias con clasificaciones de interés a partir de un API de Google ([NOMBRE DEL MÓDULO])
2. Entrenamiento de un modelo basado en Transformers (ClassificationModel.create\_model).
3. Clasificación de las noticias basados en modelo de clasificación entrenado.

### **Preparación de datos**

Se realizó una investigación para poder conseguir datasets que contengan noticias clasificadas por categoría, y se llegó a la conclusión que requerimos crear un dataset propio con las clasificaciones deseadas y así poder entrenar un modelo que nos permita reconocer una noticia y clasificarla según sea su tema.

Se procede a utilizar el API de Google (SerpApi) [2] y así extraer noticias acerca de cada una de las clasificaciones que se desean tener, **[src/recomendador/dataton\_2022.py]**.

Limpieza realizada y sugerida para a cada una de las noticias y los títulos de las noticias:

1. Paso de mayúsculas a minúsculas
2. Eliminar acentos, signos de puntuación y espacios innecesarios entre las palabras.
3. Eliminación de StopWords.
4. Eliminación de textos nulos.

Se desea tomar el texto sin procesar como entrada y proporciona predicciones del modelo en formato de texto, lo que facilita la realización de inferencias y pruebas en cualquier modelo. Para ello decidimos utilizar “**AutoModelForSequenceClassification**” el cual es un modelo que utiliza transformers (un vector de alta dimensión que representa la comprensión contextual de esa entrada por parte del modelo de Transformador).

**Tokenizadores**

Como cualquier otra red neuronal, los transformadores tampoco pueden procesar el texto de entrada sin procesar directamente, por lo tanto, debe procesarse previamente en una forma que el modelo pueda entender. Este proceso se llama tokenización donde la entrada de texto se convierte en números. Para hacer esto usamos un tokenizador que hace lo siguiente

* Dividir el texto de entrada en palabras, subpalabras o letras individuales que se denominan tokens.
* Mapeo de cada token con un entero único.
* Organizar y agregar las entradas requeridas que son útiles para el modelo.

El proceso de preprocesamiento y tokenización debe realizarse de la misma manera que cuando se entrenó el modelo.

Dado que estamos usando modelos previamente entrenados, necesitamos usar el tokenizador correspondiente para el modelo y esto se puede lograr usando la clase AutoTokenizer.

### **Entrenamiento**

El modelo de transformador (**ClassificationModel.py**) realizado tiene una arquitectura de codificador-decodificador capaz de extraer características útiles e importantes/comprensión contextual del texto de entrada, que nos permite utilizar esa entrada para un modelo clasificador de texto.

La salida del modelo es un vector de alta dimensión y tiene tres dimensiones:

1. **Tamaño del lote**: el número de secuencias procesadas a la vez.
2. **Longitud de secuencia**: la longitud de cada secuencia o representación numérica.
3. **Tamaño oculto:** la dimensión vectorial de la entrada del modelo.

Nota: Si se desea entrenar el modelo de clasificación por favor correr **main.py**

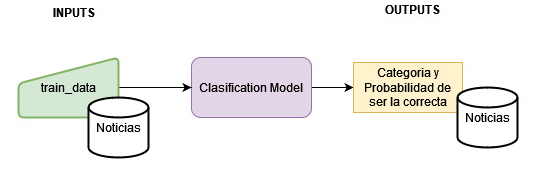
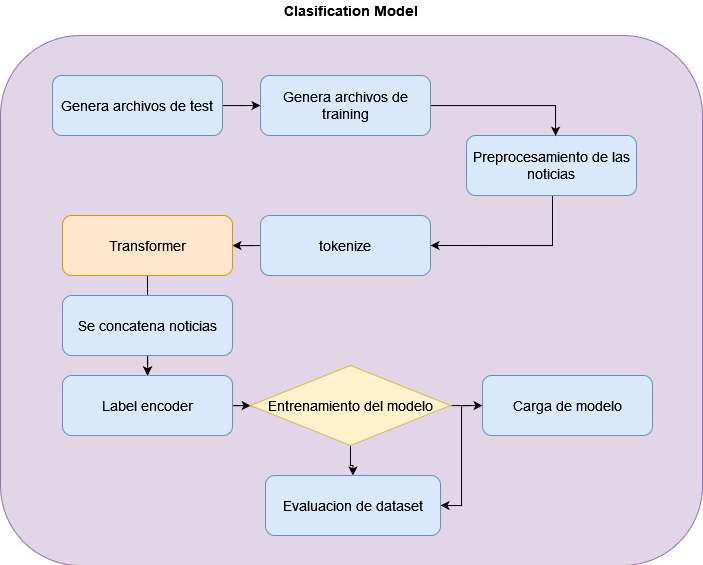


Figura 2. Descripción de pasos realizados dentro de la clase ClassificationModel

**Recomendación**

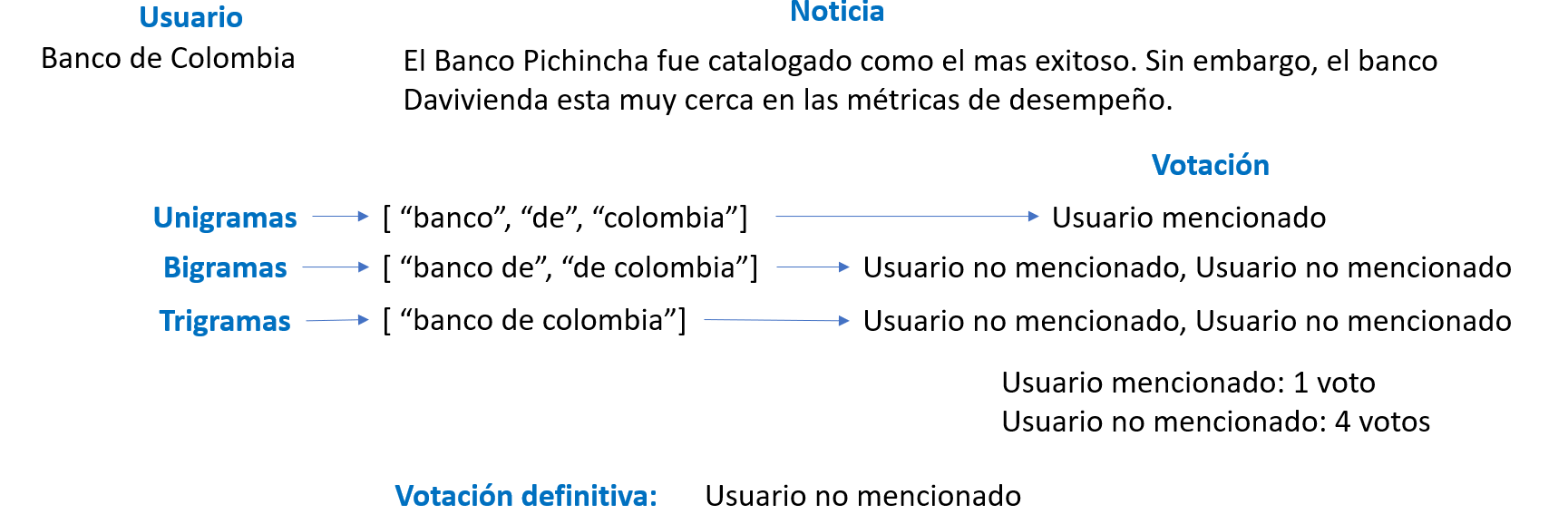
Para realizar la recomendación de la noticia nos basamos en 4 modelos:

1. Identificación de la mención del usuario dentro de la noticia y cantidad de veces que se hace.
2. Identificación de la mención del sector dentro de la noticia
3. Analizar el sentimiento de la noticia

**Mención del usuario.**

**N - gramas**

Los N-gramas son secuencias continuas de palabras o tokens en un documento. En términos técnicos, se pueden definir como las secuencias vecinas de elementos en un documento. El procedimiento implementado para saber si un usuario es mencionado dentro de una noticia fue extraer los diferentes n-gramas (unigramas, bigramas, trigramas, hasta la longitud del nombre) y buscar estos elementos dentro de la noticia. Para poder comparar los correspondientes n-gramas dentro de la noticia, se realizó un preprocesamiento en donde se convirtió todo el texto a minúscula y se eliminaron todos los caracteres que no son alfanuméricos. Una vez realizado el preprocesamiento, se analizó si cada n-grama estaba presente dentro del texto de la noticia, y se utilizó un sistema de votación para decidir si el usuario es mencionado dentro de la noticia. Para la votación, se asignó un voto para el unigrama y 2 votos para los n-gramas mayores.

Figura 3. Explicación del sistema de votación implementado en n-gramas para extraer si un usuario es mencionado en una noticia.

**Mención del Sector**

### **Preparación de datos**

Limpieza realizada y sugerida para a cada una de las noticias y los títulos de las noticias:

1. Paso de mayúsculas a minúsculas
2. Eliminar acentos, signos de puntuación y espacios innecesarios entre las palabras.
3. Eliminación de StopWords.
4. Eliminación de textos nulos.

### **Entrenamiento**

Se utilizó un **SentenceTransformers** que es un marco de Python para incrustaciones de oraciones se usa este marco para calcular incrustaciones de oraciones/texto en español.

Se tomaron las incrustaciones de cada uno de los sectores que pueden los clientes y se comparan con las noticias utilizando similitud de coseno para encontrar los sectores con un significado similar dentro de la noticia.

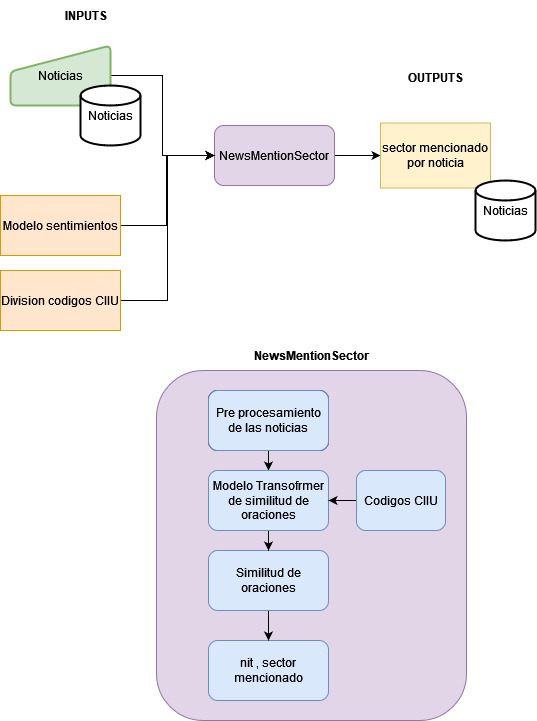
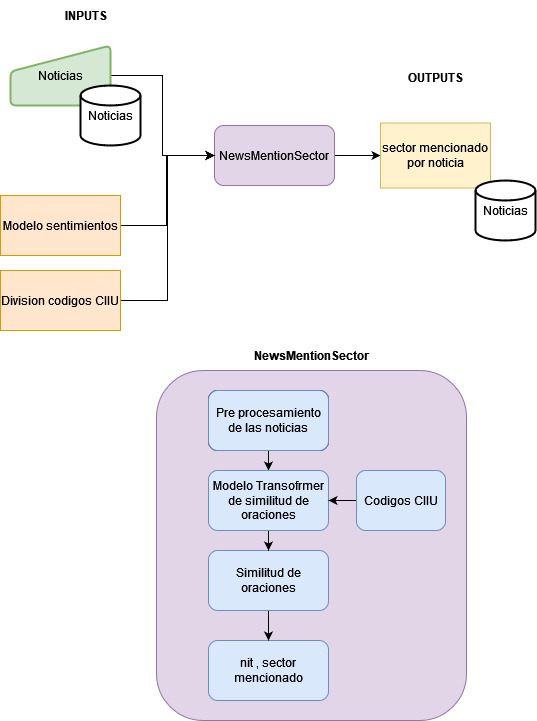
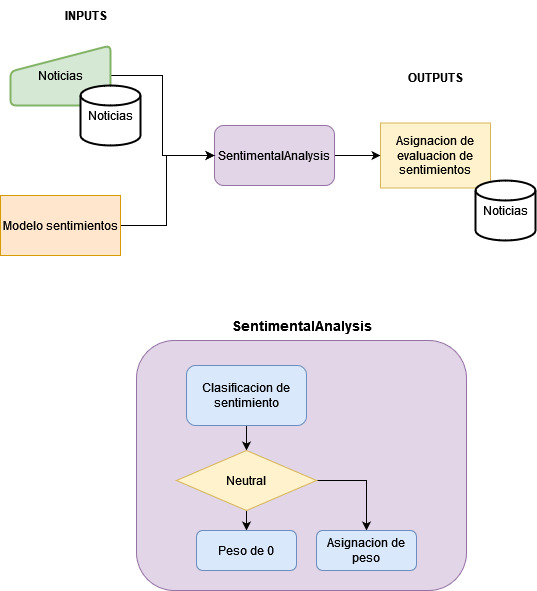
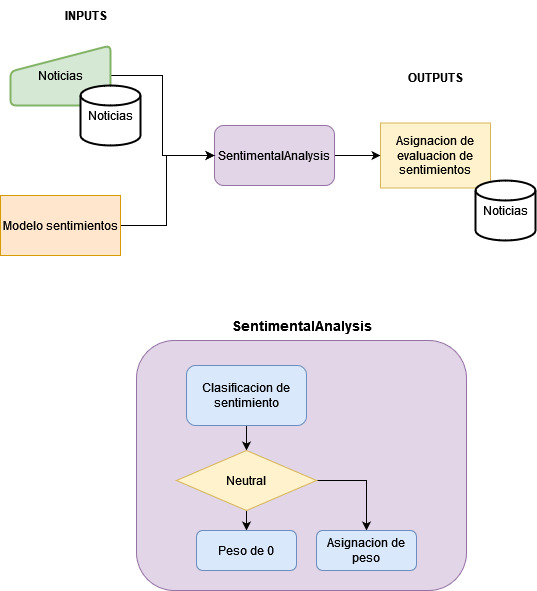


Figura 4. Descripción de la clase NewsMentionSector 

**Análisis de sentimiento**

Se realizó un proceso de análisis de sentimientos con el cual se evaluó las noticias y los respectivos títulos. Este proceso se logró mediante modelos pre entrenados para tal fin. Dicho modelo se descargó de la plataforma “Hugging Face”. El cual tiene su validez mediante tecnologías avanzadas de inteligencia artificial, que realizan el procesamiento de texto para identificar la información subjetiva y definir en la noticia , si esta corresponde a una noticia positiva o negativa.

La clasificación de estas noticias por medio del análisis de sentimiento permite asignar una capa extra de información y clasificación de las noticias. Dando mayor validez o relevancia a las noticias que contengan un sentimiento más marcado sobre otra más neutral. Esto se logró asignando pesos a las noticias según el nivel que retornara el modelo; permitiendo descartar las noticias neutrales.

Figura 5. Descripción de la clase SentimentalAnalysis.

**Recomendador Final**

Para asignar el ranking a las noticias se utilizó la importancia descrita en las especificaciones entregadas por el challenge. Del mismo modo, se utilizó el valor del análisis de sentimientos, en donde se partió del hecho de que noticias donde tengan un sentimiento positivo o negativo son más importantes que noticias en donde se tenga un sentimiento neutral. El análisis de sentimientos retorna una probabilidad y una etiqueta, ya sea positivo, negativo o neutral. Se utilizó la probabilidad como segundo ítem de ordenamiento de las noticias.

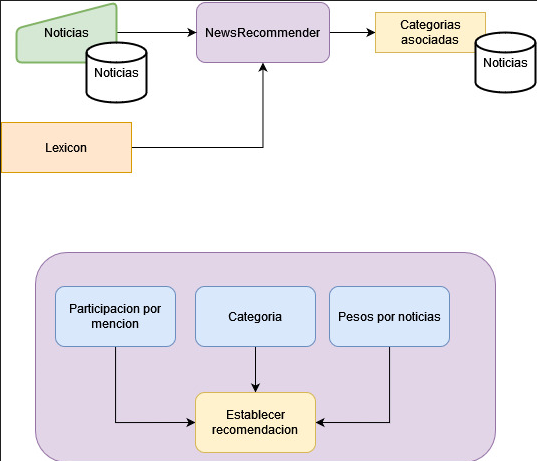


Figura 6. Descripción de la clase NewsRecomender

## **Conclusión**

1. Hemos visto que podemos descargar los modelos y tokenizadores previamente entrenados y usarlos directamente en nuestro propio conjunto de datos. Por lo tanto, podemos concluir que estos modelos pre entrenados se pueden usar para crear sistemas NLP de última generación y, al ajustarlos en nuestros propios conjuntos de datos, podemos obtener resultados sorprendentes al obtener puntajes más altos en la métrica requerida.
2. Se realizó un Lexicón para poder conocer si se realizaba la mención del sector dentro de la noticia pero no es exitoso utilizar un lexicón cuando el target es más de una palabra, por ende no se recomienda el uso de bolsa de palabras cuando el target contiene más de una palabra.
3. Se implementó un modelo no supervisado para clasificar las noticias automáticamente, pero no se tuvo un resultado lógico por lo que se cree que fue la representación de las noticias como vectores, no permitió una clusterización en clusters apropiada para las noticias.

## **¿Qué se hubiese hecho si se tuviera más tiempo?**

1. Realizar un scraping de más noticias para que el modelo de Word2vect sea más preciso.
2. Implementar diferentes técnicas de scraping para el uso de fuentes más extensas y menos limitadas para la captura de las noticias.
3. Definir algunas de las funciones como servicios en servidores externos para optimización de tiempos.
4. Definir la totalidad del proyecto como parte de un servicio que soporte la concurrencia necesaria para su uso continuo.
5. Realizar un Dashboard para la entrega de los resultados de manera visual y entendible para áreas comerciales.

## **Bibliografía**

(V, radimrehurek, 2022)

(V, s.f.)

https://towardsdatascience.com/attaining-attention-in-deep-learning-a712f93bdb1e