Aprendizaje Evolutivo de Híper Heurísticas de Selección para la Clasificación de Textos

Jonathán de Jesús Estrella Ramírez
Departamento de Ingeniería Electrónica, DICIS
Universidad de Guanajuato
Salamanca, México
jdj.estrellaramirez@ugto.mx

Juan Carlos Gómez Carranza
Departamento de Ingeniería Electrónica, DICIS
Universidad de Guanajuato
Salamanca, México
jc.gomez@ugto.mx

***Abstract*— En este artículo se presenta un modelo evolutivo encargado de aprender híper heurísticas que, para un conjunto de datos de texto que se quiera clasificar en un grupo de categorías determinadas, estas híper heurísticas permitan la selección del mejor modelo de clasificación para ese conjunto de datos en particular. El modelo construye inicialmente estas híper heurísticas utilizando como base una serie de meta características extraídas de un grupo de conjuntos de datos de texto. Las híper heurísticas son después evolucionadas utilizando operadores de cruza y mutación diseñados específicamente para la tarea. Durante el proceso evolutivo, cada híper heurística es evaluada en su desempeño para clasificar cada conjunto de datos de texto utilizando la métrica macro F1. Los resultados muestran que la mejor híper heurística aprendida obtiene un valor para F1 cercano al óptimo general para el grupo de conjuntos datos de texto (.8390).**

Keywords— algoritmos evolutivos, clasificación de textos, híper heurísticas, reglas evolutivas

I. INTRODUCCIÓN

Clasificar textos en una serie de categorías predeterminadas es una tarea muy popular dentro de las áreas de minería de datos y aprendizaje de máquina. Existen muchas aplicaciones para esta tarea, como por ejemplo la detección de sentimientos, el perfilado demográfico de usuarios, la detección de noticias falsas, el filtrado de correos electrónicos, la clasificación jerárquica de documentos, la detección de plagio, entre otros. De igual forma, existe una gran cantidad de modelos de clasificación para resolver los problemas específicos antes mencionados, tales como las máquinas de vectores de soportes, los clasificadores Bayesianos, los clasificadores basados en instancias, los basados en reglas, o toda una familia de modelos construidos con aprendizaje profundo, como las redes neuronales *long short-term memory*, *gated recurrent units* o *bidirectional encoder representations from transformers*.

El desempeño de estos tipos de modelos de clasificación para la solución de un problema específico depende tanto del enfoque que se utiliza para construirlo (probabilístico, basado en instancia, basados en reglas, funciones, ensambles, aprendizaje profundo, etc.) como de la configuración de sus híper parámetros. En los trabajos de la literatura donde se enfocan en la tarea de la clasificación de textos, es muy común encontrar en la sección de experimentación el uso de distintos modelos de clasificación construidos con diferentes valores en los híper parámetros; esto con la finalidad de encontrar el modelo de clasificación que tenga un mejor desempeño en el problema específico de que se trate. Debido a lo anterior, aún si se cuenta con un experto en el área de aprendizaje de máquina, con experiencia en la selección, construcción y ajuste de modelos de clasificación, no se puede asegurar que se tendrá una disminución considerable en el tiempo de búsqueda del mejor modelo de clasificación para el problema específico. Este tipo de procedimientos de selección y ajuste se realizan de forma cotidiana, ya que no existe un modelo de clasificación que sea superior a todos los demás en todas las tareas de clasificación de textos posibles [1].

De forma similar a la tarea de clasificación de textos, existe una amplia variedad de modelos de clasificación en otras tareas de aprendizaje de máquina, como la detección o reconocimiento de personas u objetos en imágenes o videos, o el reconocimiento del habla, ocurriendo el mismo problema para la selección del mejor modelo de clasificación para el problema específico que se esté tratando. Por ello, en años recientes el aprendizaje de máquina automatizado (*Automated Machine Learning* o *AutoML* en inglés) ha surgido como una propuesta para intentar aliviar este problema [2]. Sin embargo, las propuestas derivadas de éste se han enfocado en optimizar el proceso de encontrar la mejor solución para una única tarea para un único conjunto de datos. Adicionalmente, las propuestas de AutoML para la tarea de clasificar textos se han enfocado principalmente en la etapa de preprocesamiento y no han atacado la selección de los modelos de clasificación.

En este trabajo se propone un modelo evolutivo capaz de generalizar el proceso de selección de modelos para tareas de clasificación de texto. La selección de un modelo es determinada a partir de una híper heurística, la cual es aprendida a través de un proceso de evolutivo con el uso de algoritmos genéticos y operadores específicamente diseñados (selección, cruce, y mutación). Una híper heurística es un grupo de reglas de comparación basadas en meta características. A su vez, las meta características están basadas en estadísticas del contenido textual calculadas para un grupo de conjuntos de datos de texto que son utilizados en diversas tareas de clasificación. Los resultados demuestran que la mejor híper heurística aprendida con el modelo evolutivo propuesto, es capaz de seleccionar un modelo adecuado (con un desempeño óptimo, o cercano a él) para cada conjunto de datos en el grupo de conjunto de datos para diversos problemas de clasificación de texto. El desempeño de una híper heurística es evaluado mediante la métrica de evaluación macro F1.

El resto del artículo está organizado de la siguiente manera. La Sección II presenta una revisión de la literatura y trabajos relacionados. En la Sección III se describe la metodología utilizada para la construcción, aprendizaje, y evaluación del modelo evolutivo. En la Sección IV se presentan la configuración utilizada en el modelo evolutivo, los resultados obtenidos y la discusión de estos. Finalmente, la Sección V está dedicada a las conclusiones del trabajo presentado.

II. Revisión de la literatura

Con el crecimiento masivo de la generación y el almacenamiento de datos en los últimos años, se han desarrollado una gran variedad de métodos que permiten procesarlos y analizarlos de forma efectiva. Una gran parte de estos datos corresponden a información textual generada por la interacción de las personas en diferentes medios, de forma reciente principalmente a través de internet. Debido a esto, la tarea de clasificación de textos ha cobrado una gran relevancia en las últimas décadas en las áreas de aprendizaje de máquina (*Machine Learning* o *ML* en inglés) y procesamiento de lenguaje natural (*Natural Language Processing* o *NLP* en inglés), en donde se ha investigado la creación y aplicación de diversos métodos que analizan y dan solución a problemas dentro de la clasificación de textos. Sin embargo, a pesar de la diversidad de métodos desarrollados [3, 13], la intervención de un humano experto en el área de aprendizaje de máquina aún es necesaria para la selección, construcción y ajuste de modelos.

Así mismo, en años recientes el término AutoML ha sido utilizado en diferentes ámbitos, haciendo alusión a la idea de automatizar los procesos donde ML es aplicado. Esto es, AutoML pretende realizar la mayor automatización posible en diversos problemas de ML, lo que incluye: la selección automatizada del modelo, la búsqueda automatizada de la arquitectura neuronal, la ingeniería automatizada de características [4], entre otros. Algunas de las propuestas más populares y modernas de selección automatizada de modelo, se obtuvieron durante las dos competiciones de ChaLearn AutoML Challenges que ocurrieron entre los años 2015 y 2018 [2]. El enfoque de estas competiciones era en aprendizaje supervisado de ML, resolviendo diversos problemas de clasificación y regresión sin la supervisión humana.

Para el uso de AutoML en la tarea de clasificación de textos, se han desarrollado diferentes propuestas relevantes en los últimos años. Primero, en el trabajo propuesto bajo el nombre de ELMR (*Evolutionary Learning of Meta-Rules* en inglés) en [5], los autores presentan una generalización del proceso de selección de modelos de clasificación por medio de un conjunto de meta reglas definidas por meta características de conjuntos de datos de texto, el conjunto de meta reglas fue aprendido a través de un proceso de aprendizaje evolutivo con uso de algoritmos genéticos. Después, en [6] los autores se enfocaron en la clasificación de conjuntos de datos de texto de acuerdo con la tarea a la que corresponden, esto fue realizado por medio de un conjunto inicial de 72 meta características, incluyendo aquellas utilizadas en [5] y obteniendo otras a partir del contenido textual de un grupo de conjuntos de datos de texto. Más tarde, los autores en [7], con el uso del mismo conjunto de meta características utilizadas en [6], y con representaciones textuales populares y variaciones de éstas, propusieron un método con el cual se buscaba encontrar la representación textual más adecuada para un conjunto de datos de texto. La representación textual es elegida con base en un entrenamiento previo con conjuntos de datos de texto con tareas similares. Como una continuación de los trabajos [6, 7], los autores en [8] presentaron AutoText, un modelo que, a partir de un conjunto de datos de texto dado, éste se encarga de buscar la representación textual más adecuada (aquella que ha funcionado bien con tareas similares), y después, con el uso de AutoSklearn 1.0 [10] realizar el proceso de selección del mejor modelo de clasificación. En [9], se presentó una nueva extensión en la etapa del preprocesamiento de textos para obtener una nueva representación textual a partir de otra realizada previamente, esta nueva representación textual se obtiene a partir de tres pasos. Primero, se calcula un nuevo conjunto de meta características basadas en distancias que definen la nueva representación. Segundo, se aplica un método de dispersión a la nueva representación, eliminando el ruido que pudieran contener las muestras. Por último, se aplica un muestreo selectivo, conservando solo aquellas muestras que no sean conflictivas; permitiendo así que, con esta nueva representación, los métodos de clasificación aprovechen mejor el conjunto de datos en la etapa de entrenamiento.

Todos estos trabajos anteriores cuentan con el uso de meta características, ya sean basadas en estadísticas, distancias o en el contenido textual. La mayoría de estos se enfocan principalmente en etapas previas al proceso de selección del modelo de clasificación, a excepción de ELMR [5]. Debido a esto, este trabajo se enfoca en desarrollar un modelo evolutivo encargado de aprender una generalización del proceso de selección de modelos para clasificación de texto.

III. Metodología

El modelo inicia con la construcción de una población de un determinado número de híper heurísticas (individuos). Cada híper heurística es un grupo de reglas, y a su vez cada regla contiene un conjunto de condiciones determinadas con base a un conjunto de meta características, un conjunto de operadores de comparación, y un valor elegido al azar delimitado por un valor máximo correspondiente a la meta característica a evaluar. Después, cada híper heurística es evaluada, con el fin de saber su aptitud para la solución del problema. La población se ve involucrada en un proceso de evolución que consta de tres principales operadores evolutivos (selección, cruce, mutación). La manera de determinar qué híper heurísticas serán utilizadas en los operadores evolutivos de cruce y mutación es al azar. Las híper heurísticas generadas por el proceso anterior son evaluadas al igual que la población original. Después, con el uso del operador de selección, la nueva población para la siguiente generación es determinada basándose en su aptitud para la solución del problema. La población nunca cambia de tamaño. Este proceso se repite durante un determinado número de generaciones (criterio de terminación). En la última generación, la híper heurística con la mejor aptitud es presentada como la solución final. El proceso general del modelo es representado en la Fig. 1.

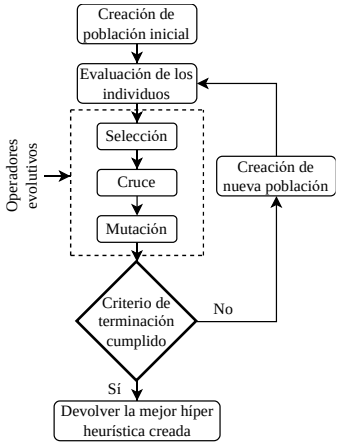

Fig. 1. Proceso general del aprendizaje evolutivo de las híper heurísticas.

A. *Conjuntos de datos*

Para experimentar con el modelo evolutivo propuesto, se utilizó un grupo formado por 10 conjuntos de datos de texto para diferentes tipos de problemas, algunos de ellos muy populares en la literatura. Para cada conjunto se tomaron 50% de los datos de manera estratificada para realizar los experimentos en este artículo. A su vez, cada subconjunto de datos se dividió en 80% para entrenamiento y 20% para prueba.

TABLE I. CONJUNTOS DE DATOS UTILIZADOS PARA LOS EXPERIMENTOS

Conjunto de datos	Tarea	#Categorías	#Docs
20ng	Clasificación de noticias	20	7528
AGNews	Clasificación de noticias	4	127600
AmzSts	Detección de sentimientos	2	1000
classic	Clasificación de temas	4	7097
csdmc	Filtrado de correos electrónicos	2	4327
movies	Detección de sentimientos	2	2000
r52	Clasificación de noticias	51	9099
trec07p	Filtrado de correos electrónicos	2	75419
wipo	Clasificación jerárquica de patentes	114	75249
Yelp	Detección de sentimientos	2	1000

En la Tabla I se muestran los nombres de los conjuntos de datos, el problema en el que se utilizan, el número de categorías y la cantidad de documentos que hay en cada conjunto. Como se puede observar, el grupo de conjuntos de datos cuenta con una variabilidad tanto en el número de documentos, el número de categorías, y en las tareas en las que se utilizan. Esto es adecuado para realizar un mejor análisis del comportamiento del modelo evolutivo en su proceso de generalización.

Para cada parte de entrenamiento de cada conjunto de datos se extrajeron 16 meta características que representan la distribución de los datos dentro de cada conjunto, las meta características correspondientes al análisis de componentes principales (*Principal Component Analysis* o *PCA* en inglés) se calculan a partir de una representación TF-IDF. Las características extraídas se muestran en la Tabla II:

TABLE II. CONJUNTO DE META CARACTERÍSTICAS UTILIZADAS

Abreviatura	Descripción
nDocs	Número de documentos
nTops	Número de categorías
dptAvg	Media de documentos por categoría
dptMed	Mediana de documentos por categoría
dptStd	Desviación estándar de documentos por categoría
dptStdAvg	Razón entre dptStd y dptAvg
dptEnt	Entropía de documentos por categoría
wpdAvg	Media de palabras por documento
wpdMed	Mediana de palabras por documento
wpdStd	Desviación estándar de palabras por documento
wpdStdAvg	Razón entre wpdStd y wpdAvg
wpdEnt	Entropía de palabras por documento
pca10	Porcentaje de varianza explicada por los primeros 10 componentes de PCA
pca20	Porcentaje de varianza explicada por los primeros 20 componentes de PCA
pca30	Porcentaje de varianza explicada por los primeros 30 componentes de PCA
cmxWds	Porcentaje de palabras complejas en todo el conjunto de datos

En la Tabla III se muestran los valores de las meta características para cada parte de entrenamiento de cada conjunto de datos.

TABLE III. VALORES DE LAS META CARACTERÍSTICAS PARA LOS CONJUNTOS DE DATOS

	20ng	AGNews	AmzSts	classic	csdmc	movies	r52	trec07p	wipo	yelp
nDocs	3011	51040	400	2838	1730	800	3639	30167	30099	400
nTops	20	4	2	4	2	2	51	2	114	2
dptAvg	150.55	12760	200	709.5	865	400	71.3529	15083.5	264.0263	200
dptMed	152.5	12748	200	561.5	865	400	10	15083.5	123.5	200
dptStd	13.223	57.2756	2	338.562	311	1	247.8879	4959.5	404.4089	3
dptStdAvg	0.0878	0.0045	0.01	0.4772	0.3595	0.0025	3.4741	0.3288	1.5317	0.015
dptEnt	2.7878	1.3863	0.6931	1.3863	0.6931	0.6931	3.1907	0.6931	4.5735	0.6931
wpdAvg	143.7901	24.3722	5.2025	60.0123	171.3954	351.9825	63.0753	399.5376	63.9485	5.4325
wpdMed	86	24	4	56	88	323.5	43	121	61	5
wpdStd	315.2008	6.7934	3.172	47.6815	403.1663	157.6205	65.2423	919.4115	29.7861	3.1702
wpdStdAvg	2.1921	0.2787	0.6097	0.7945	2.3523	0.4478	1.0344	2.3012	0.4658	0.5836
wpdEnt	5.5708	3.2124	2.3834	4.7886	5.5998	5.87	4.9618	6.2478	4.7579	2.423
pca10	0.0296	0.023	0.1242	0.0508	0.1026	0.0354	0.1467	0.1315	0.0337	0.1169

	20ng	**AGNews**	**AmzSts**	**classic**	**csdmc**	**movies**	**r52**	**trec07p**	**wipo**	**yelp**
pca20	0.0489	0.0377	0.2055	0.0776	0.1496	0.065	0.192	0.1894	0.0531	0.1841
pca30	0.0646	0.0494	0.2704	0.0995	0.1774	0.0909	0.2221	0.2284	0.0687	0.2388
cmxWds	0.1814	0.1898	0.1499	0.3813	0.2005	0.1895	0.0833	0.2071	0.3203	0.1266

B. Representación de las híper heurísticas

Dentro del modelo evolutivo, se representa a una híper heurística como un grupo de reglas, donde cada regla a su vez está formada por un grupo de condiciones. Cada condición evalúa a alguna de las meta características de los conjuntos de datos mencionadas previamente. La condición se expresa como una 3-tupla de la forma (característica, operador, valor de referencia). El grupo de reglas de una híper heurística puede ser representado como se muestra en la Figura 2.

IF $C_{1,1}$ AND $C_{1,2}$ AND … AND $C_{1,n1}$ THEN **ACCIÓN**$_1$

IF $C_{2,1}$ AND $C_{2,2}$ AND … AND $C_{2,n2}$ THEN **ACCIÓN**$_2$

…

IF $C_{m,1}$ AND $C_{m,2}$ AND … AND $C_{m,nm}$ THEN **ACCIÓN**$_m$

ELSE **ACCIÓN**$_{m+1}$

Fig. 2. Representación de las reglas que componen a una híper heurística.

Inicialmente, se crea una población de estas híper heurísticas de manera aleatoria. Para formar una híper heurística primero se determina al azar cuántas reglas debe tener, con un mínimo de 2 reglas y un máximo de 7. Adicional a las reglas, se agrega al final de la híper heurística una acción final, correspondiente a un 'else', que se ejecutará en caso de que ninguna regla se cumpla. Posteriormente, para cada regla se determina al azar cuántas condiciones debe tener, con un mínimo de 1 condición y un máximo de 4. Finalmente, para cada condición se determina aleatoriamente la 3-tupla que la compone, seleccionando la meta característica a evaluar, el operador a aplicar y el valor de referencia. Se establecieron valores máximos para los valores de referencia para cada meta característica de acuerdo con los valores encontrados para las meta características en las partes de entrenamiento de los conjuntos de datos. Los operadores de comparación son '>' o '<'. Se decidió no incluir el operador '=', ya que sería un criterio muy estricto que rara vez puede cumplirse. Una misma meta característica puede estar siendo evaluada dentro de una misma regla, lo que permitiría establecer rangos de comparación (p. ej. dptAvg > 70 AND dptAvg < 11618). Para un mejor entendimiento al aplicar los operadores evolutivos, las híper heurísticas pueden ser visualizadas por representaciones en bloques [11], como se observa en la Fig. 3.

Cada una de las reglas, incluyendo la regla 'else' (regla *m* + 1 de la Fig. 3), cuentan con una acción, la acción corresponde a un modelo de clasificación. Para el trabajo presentado en este artículo, si se cumplen todas las condiciones de una regla de una híper heurística al ser comparadas con las meta características de un conjunto de datos, el modelo de clasificación correspondiente a la acción es el elegido para ser entrenado con tal conjunto de datos, y evaluado con el respectivo conjunto de datos de prueba.

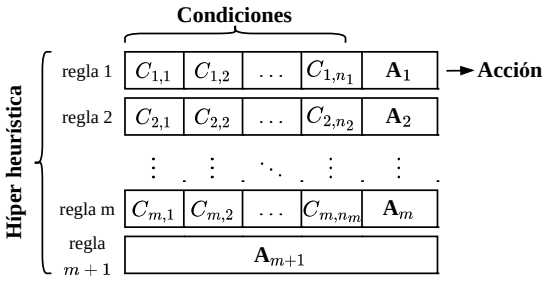

Fig. 3. Representación en bloques de una híper heurística, n_1, n_2, …, n_m son valores al azar entre 2 y 7.

El modelo evolutivo utiliza los siguientes modelos de clasificación: Multinomial (MNB), Complemento (CNB), y Bernoulli (BNB) de Naïve Bayes; K-Vecinos cercanos (KNN) con k = 1, 5, 10 o 20 y similitud de coseno; Árboles de decisión (DT) con criterio = 'entropy' o 'gini'; Regresión logística (LR) con C = 0.1, 1 o 10 y el solucionador 'lbfgs'; Máquinas de vectores de soporte (SVM) con kernel lineal, C = 0.1, 1 o 10 en su forma dual. En total, se tiene un grupo de 15 modelos de clasificación con diferentes enfoques e híper parámetros.

C. Operadores evolutivos

1) Operador de cruce: En su forma más sencilla, cuando los individuos son representados mediante cadenas binarias, dos individuos son elegidos y cada uno es dividido en una posición escogida al azar, las partes de los individuos son intercambiadas para producir dos nuevos individuos. Para aplicarlo sobre las híper heurísticas generadas *en el modelo evolutivo propuesto*, se utiliza la técnica de cruce de un único punto "single point crossover" [12] a nivel de bloques. Dos híper heurísticas son elegidas al azar, depués un punto de división es elegido al azar con base en el valor de longitud de la híper heurística con el menor número de reglas, las híper heurísticas son divididas en este punto, obteniendo dos subgrupos de reglas de cada híper heurística. *Una primer* nueva híper heurística es creada a partir del primer subgrupo de reglas de la híper heurística 1, y el segundo subgrupo de la híper heurística 2. *Una segunda* nueva híper heurística se crea con el primer subgrupo de la híper heurística 2, y con el segundo subgrupo de la híper heurística 1. Este proceso puede ser visualizado en la Fig. *4*.

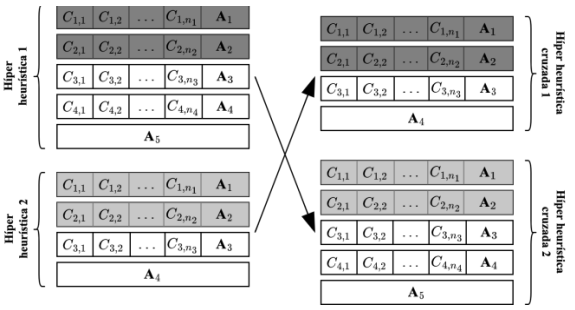

Fig. 4. Operador de cruce a nivel de bloques.

2) Operador de mutación: Este operador permite explorar diferentes áreas del espacio de búsqueda. Cuando los individuos son representados mediante cadenas binarias, al aplicar el operador de mutación existe una probabilidad (con valores menores al 20%, según la literatura [12]) para que se realice un cambio en los valores de la cadena. Con respecto al modelo evolutivo presentado, cuando se cumple la probabilidad de que el operador de mutación sea aplicado a una híper heurística, hay dos opciones, las cuales son eliminar una regla o crear y agregar una nueva a la híper heurística seleccionada. Para poder agregar una regla se tienen que cumplir una de dos condiciones, (1) que el número de reglas de la híper heurística sea menor que el máximo de reglas permitidas más la regla 'else', o (2) que el número de reglas sea el mínimo de reglas más la regla 'else'. En el caso de que no se cumplan ninguna de las dos condiciones anteriores, se eliminará una regla elegida al azar, se omite a la regla 'else' en el conjunto de reglas posibles a eliminar o agregar. Ambos estados son ejemplificados en la Fig. 5.

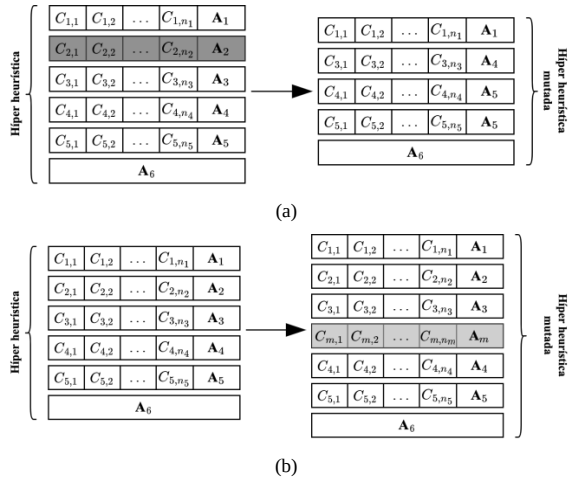

Fig. 5. Operador de mutación a nivel de bloques, (a) Eliminar una regla y (b) Crear y agregar una nueva regla.

D. Evaluación y selección

Dada una población, cada híper heurística dentro de ella se evalúa con la métrica macro F1. Con base en esta métrica se determina su posición entre toda la población, un mayor valor es una mayor probabilidad de ser seleccionada para la siguiente generación.

Para los experimentos desarrollados en este artículo, un método para la aceleración de la etapa de aprendizaje evolutivo del modelo fue llevado a cabo en una etapa previa, en la cual se entrenó cada uno de los 15 modelos de clasificación propuestos con cada uno de los conjuntos de datos del grupo de entrenamiento, y cada modelo de clasificación fue evaluado con el respectivo conjunto de datos del grupo de prueba. Para entrenar y probar los modelos, cada conjunto de datos fue transformado utilizando el método TF-IDF para obtener una representación matricial. De esta forma, se tiene una aceleración para la obtención del desempeño que tendrá la acción de una regla de una híper heurística para todo el grupo de conjuntos de datos, evitando el entrenamiento y la evaluación repetitiva de acciones (modelos de clasificación).

Con base a lo anterior, cada híper heurística de una población se evalúa con todo el grupo de conjuntos de datos, de la siguiente manera:

1. Obtener las meta características del conjunto de datos.

2. Evaluar cada una de las reglas de la híper heurística. En este paso cada regla evalúa las condiciones, y si se cumplen todas las condiciones contenidas en ella se aplica la acción (modelo de clasificación) correspondiente a esa regla. Posteriormente, se obtiene el valor del desempeño obtenido para la métrica macro F1. En caso de que las meta características del conjunto de datos no cumplan con ninguna regla, se aplica la acción asociada con la regla 'else'. Finalmente, se repite el proceso desde el paso 1 para el siguiente conjunto de datos.

3. Una vez hayan sido aplicadas las acciones para todo el grupo de conjuntos de datos, se suman los valores macro F1 obtenidos en la evaluación de la híper heurística y se promedian. El valor obtenido es el desempeño general de la híper heurística para el grupo de conjuntos de datos

Esta secuencia de tres pasos es aplicada a cada una de las híper heurísticas de la población, obteniendo una lista con los desempeños de cada una de ellas.

En la generación inicial se evalúan directamente las híper heurísticas de la población creadas al azar. A esta población inicial, llamada población padre, se le aplica el operador de cruce para generar una población hija con el mismo tamaño de la población padre. Posteriormente, el operador de mutación es aplicado con una probabilidad de 20% sobre la población hija. La población hija obtenida después de aplicar los operadores es evaluada con el mismo proceso descrito con anterioridad.

Finalmente, se mezclan amblas poblaciones, padre e hija, en una población total, se ordenan las híper heurísticas de acuerdo con su desempeño, y se seleccionan las mejores para ser los padres de la siguiente generación. El tamaño de la población padre siempre es el mismo a través de las generaciones.

Cuando el modelo evolutivo se encuentra en la última generación y ha realizado las evaluaciones de las híper heurísticas, la híper heurística con el mejor desempeño es seleccionada como el resultado final de la etapa del aprendizaje evolutivo. Siendo ésta la que, en principio, permite una generalización del proceso de la selección de modelos de clasificación.

Los diferentes componentes del modelo evolutivo fueron implementados en Python, utilizando los módulos scikit-learn, numpy, spacy, nltk, random y math.

IV. RESULTADOS

A continuación, se presentan los resultados obtenidos del modelo evolutivo. Los resultados corresponden a una ejecución del modelo, con los siguientes parámetros generales:

- Tamaño de la población: 50.
- Número de generaciones: 100.
- Máximo número de reglas: 7.
- Mínimo número de reglas: 2.
- Máximo número de condiciones: 4.
- Mínimo número de condiciones: 1.
- Probabilidad de mutación: 20%.

En la Tabla IV se muestran los modelos de clasificación óptimos, en conjunto con el desempeño obtenido con la métrica de evaluación macro F1, para cada conjunto de datos utilizado en los experimentos. El promedio general de los desempeños de los modelos óptimos da 0.8390, lo que indicaría el óptimo a ser alcanzado. En esta tabla se observa que los modelos de clasificación basados en SVM son los que predominan, con valores diferentes en su híper parámetro C.

TABLE IV. MODELOS DE CLASIFICACIÓN ÓPTIMOS PARA LOS CONJUNTOS DE DATOS DE PRUEBA

Conjunto de datos	Modelo de clasificación	Macro F1
20ng	LR C=10	.8794
AGNews	SVM C=0.1	.9077
AmzSts	SVM C=1	.8697
classic	SVM C=1	.9735
csdmc	SVM C=10	.9711
movies	LR C=10	.8350
r52	SVM C=10	.6980
trec07p	SVM C=1	.9961
wipo	SVM C=1	.4420
yelp	CNB	.8174
Promedio		.8390

En la Figura 6 se observa el comportamiento de las híper heurísticas durante el proceso evolutivo de las 100 generaciones. Se muestran el desempeño de la mejor híper heurística, la peor híper heurística y el promedio de toda la población para cada generación. En esta figura es posible observar cómo el desempeño de las híper heurísticas va mejorando con cada generación y se tiene una convergencia del desempeño de las híper heurísticas de toda la población alrededor de la generación 30. Es decir, a partir de esa generación, la mejor y peor híper heurística tienen comportamientos similares.

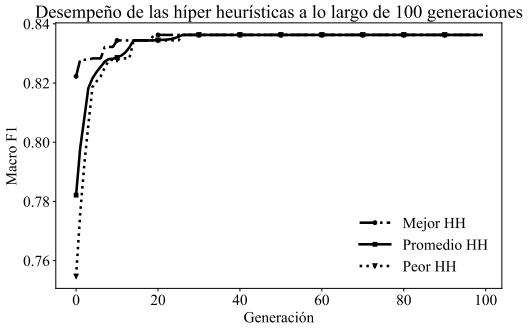

Fig. 6. Desempeño de la población de híper heurísticas a lo largo de 100 generaciones.

Este experimento dio como resultado la híper heurística mostrada en la Figura 7, la cual alcanza un desempeño promedio sobre todo el grupo de conjuntos de datos de prueba de .8362, el cual es un desempeño muy cercano al desempeño óptimo de .8390, mostrado en la Tabla IV.

Regla 1: IF wpdMed>265.4069 AND cmxWds>0.0269 AND wpdAvg>328.8453 THEN MNB

Regla 2: IF dptStdAvg>2.0275 AND cmxWds<0.9997 AND dptStdAvg>2.6203 AND wpdEnt>0.1083 AND dptStd<3072.6780 AND dptEnt>2.3230 THEN SVM C=10

Regla 3: IF dptEnt<1.9896 AND cmxWds<0.1326 AND pca20<0.6360 AND dptStd<3960.6352 AND wpdMed<139.8962 AND wpdEnt<4.2479 THEN CNB

Regla 4: ELSE SVM C=1

Fig. 7. Mejor híper heurística obtenida después de 100 generaciones.

Se puede observar que los modelos de clasificación utilizados en las reglas corresponden a modelos Naïve Bayes (MNB y CNB) y SVM; mientras que los modelos de clasificación de KNN, DT, LR, no son considerados en la híper heurística, lo que indica que su desempeño es dominado por los clasificadores mencionados. También cabe señalar que, de acuerdo con los resultados de la Tabla IV, se aspiraba a encontrar reglas con acciones correspondientes a modelos SVM, que son los dominantes, lo cual se consigue en la híper heurística final. Además, se puede visualizar que la meta característica que indica el porcentaje de palabras complejas en un documento (cmxWds) ha resultado ser relevante para cada una de las reglas que componen la híper heurística. También en la Tabla V se muestran el uso de cada una de las reglas de la mejor híper heurística obtenida al clasificar el grupo de conjuntos de datos de prueba, con un dominio del clasificador SVM, como era esperado.

TABLE V. FRECUENCIA DEL USO DE REGLAS DE LA MEJOR HÍPER HEURÍSTICA

Regla	Frecuencia
Regla 1	1
Regla 2	1
Regla 3	1
Regla 4	7

V. Conclusiones

En este trabajo se ha presentado un modelo evolutivo que permite generalizar el proceso de selección de modelos para problemas de clasificación de texto. Por medio de un proceso de aprendizaje evolutivo el modelo ha logrado aprender una híper heurística, la cual permite la selección de los modelos de clasificación más aptos para la clasificación de conjuntos de datos de texto, El desempeño logrado con la mejor híper heurística obtenida al final del proceso de aprendizaje evolutivo ha sido muy cercano al valor óptimo. En la mejor híper heurística encontrada, se encontró que los modelos basados en Naïve Bayes y en SVM presentan un dominio sobre el resto de los modelos. Lo anterior corresponde en buena medida con los modelos óptimos para cada conjunto de datos. De igual forma, se observó que la meta característica que indica el porcentaje de palabras complejas en un documento (cmxWds) es relevante dentro de las reglas que conforman la mejor híper heurística.

Como trabajo futuro se puede establecer el uso de otros modelos de clasificación dentro del grupo de posibles acciones, tales como aquellos basados en aprendizaje profundos. De igual forma, se pueden agregar otros operadores de cruza o mutación que afecten no solo a las reglas si no a las condiciones dentro de ellas. Adicionalmente, se puede explorar el uso de algoritmos evolutivos multi objetivo en donde no solo sea relevante el desempeño en la clasificación, si no el tiempo de entrenamiento y clasificación, lo cual puede ser relevante cuando se trabaje con modelos de clasificación complejos como aquellos basados en aprendizaje profundo.

References

[1] Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. IEEE transactions on evolutionary computation, 1(1), 67-82.

[2] Guyon, I., Sun-Hosoya, L., Boullé, M., Escalante, H. J., Escalera, S., Liu, Z., ... & Viegas, E. (2019). Analysis of the automl challenge series. Automated Machine Learning, 177.

[3] Khurana, D., Koli, A., Khatter, K., & Singh, S. (2022). Natural language processing: State of the art, current trends, and challenges. Multimedia Tools and Applications, 1-32.

[4] Yao, Q., Wang, M., Chen, Y., Dai, W., Li, Y. F., Tu, W. W., ... & Yu, Y. (2018). Taking human out of learning applications: A survey on automated machine learning. arXiv preprint arXiv:1810.13306.R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[5] Gomez, J. C., Hoskens, S., & Moens, M. F. (2017, July). Evolutionary learning of meta-rules for text classification. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (pp. 131-132).

[6] Madrid, J. G., & Escalante, H. J. (2019, October). Meta-learning of text classification tasks. In Iberoamerican Congress on Pattern Recognition (pp. 107-119). Springer, Cham.

[7] Madrid, J. G., Escalante, H. J., & Morales, E. (2019, September). Meta-learning of textual representations. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 57-67). Springer, Cham.

[8] Madrid, J. (2019). Autotext: AutoML for text classification.

[9] Cunha, W., Canuto, S., Viegas, F., Salles, T., Gomes, C., Mangaravite, V., ... & Rocha, L. (2020). Extended pre-processing pipeline for text classification: On the role of meta- feature representations, sparsification, and selective sampling. Information Processing & Management, 57(4), 102263.

[10] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. Advances in neural information processing systems, 28.

[11] Gomez, J. C., & Terashima-Marín, H. (2018). Evolutionary hyper-heuristics for tackling bi-objective 2d bin packing problems. Genetic Programming and Evolvable Machines, 19(1), 151-181.

[12] Affenzeller, M., Wagner, S., Winkler, S., & Beham, A. (2009). Genetic algorithms and genetic programming: modern concepts and practical applications. Chapman and Hall/CRC.

[13] Gasparetto, A., Marcuzzo, M., Zangari, A., & Albarelli, A. (2022). A Survey on Text Classification Algorithms: From Text to Predictions. Information, 13(2), 83.