

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería.
Ingeniería de Sistemas y Computación.
Proyecto de Grado.

Predicción del tratamiento para la leishmaniasis cutánea mediante datos génicos e inferencia gramatical

Josue Peña Atencio

Directora: Dra. Gloria Inés Álvarez Vargas
Codirector: Dr. Diego Luis Linares Ospina

26 de Noviembre 2021



Santiago de Cali, 26 de Noviembre 2021.

Señores

Pontificia Universidad Javeriana Cali.

Dr Gerardo Mauricio Sarria Montemiranda

Director Carrera de Ingeniería de Sistemas y Computación.

Cali.

Cordial Saludo.

Por medio de la presente hacemos constatar que hemos revisado el proyecto de grado "**Predicción del tratamiento para la leishmaniasis cutánea mediante datos génicos e inferencia gramatical**" del estudiante Josue Peña Atencio, del cual somos directores, y lo consideramos apto para ser presentado y sometido a consideración del jurado.

Atentamente,



Dra. Gloria Inés Álvarez Vargas
Directora de Trabajo de Grado



Dr. Diego Luis Linares Ospina
Codirector de Trabajo de Grado

Santiago de Cali, 26 de Noviembre 2021.

Señores

Pontificia Universidad Javeriana Cali.

Dr Gerardo Mauricio Sarria Montemiranda

Director Carrera de Ingeniería de Sistemas y Computación.

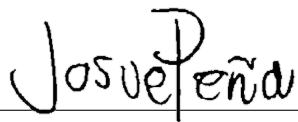
Cali.

Cordial Saludo.

Tengo el placer de presentar ante usted el proyecto de grado titulado como: “**Predicción del tratamiento para la leishmaniasis cutánea mediante datos génicos e inferencia gramatical**”, para ser sometido a consideración del jurado.

Espero que este proyecto reúna los requisitos académicos necesarios para su aprobación.

Atentamente,

A handwritten signature in black ink that reads "Josue Peña". The signature is written in a cursive style with a large, stylized 'P'.

Josue Peña Atencio

Estudiante Ingeniería de Sistemas y Computación

Código: 8935601

Índice general

1. Resumen	1
2. Descripción del Problema	3
2.1. Descripción del Problema	3
2.2. Objetivos	5
2.2.1. Objetivo General	5
2.2.2. Objetivos Específicos	5
3. Marco teórico	7
3.1. Trabajos Relacionados	8
4. Preparación de los Datos	11
4.1. Descripción	11
4.2. Pre-procesamiento	14
4.3. Creación de alfabeto	17
4.4. Codificación de genes	22
4.4.1. Codificación binaria	22
4.4.2. Codificación ternaria	24
4.5. Creación de palabras	25
4.6. Variación de datos	26
5. Algoritmo RPNI	29
5.1. Descripción	29
5.2. Entrenamiento y evaluación	30
5.3. Resultados	34
6. Algoritmo OIL	37
6.1. Descripción	37
6.2. Entrenamiento y evaluación	38
6.3. Resultados	40
7. Algoritmo Integrador	43
7.1. Descripción	43
8. Resultados y Análisis	47
9. Conclusiones	51

Resumen

La leishmaniasis es una enfermedad parasitaria usualmente transmitida por moscas de arena infectadas que suelen vivir en ambientes tropicales. La forma más común de leishmaniasis en Colombia es la leishmaniasis cutánea [1], la cual provoca úlceras en la piel. Para esta forma, el tratamiento actual mediante el medicamento Glucantime tiene un porcentaje de fracaso que varía entre el 19 % y el 81 % [2]. Colombia y otros países afectados tienen poco interés en esta enfermedad que está profundamente relacionada con la pobreza, y no cuentan con el conocimiento médico para garantizar un tratamiento completamente seguro [3].

La dificultad para tratar la enfermedad radica en la compleja interacción entre el parásito y el sistema inmunológico [4], el cual está relacionado con el estado de expresión génica de cada paciente [5]. En el presente trabajo, se hace uso de 7 conjuntos de datos provistos por el CIDEIM de Cali, los cuales recolectan la información de expresión génica de tres tipos de glóbulos blancos provenientes de 14 pacientes de leishmaniasis anónimos antes, durante y después del tratamiento para la enfermedad.

Se utilizan dos algoritmos de inferencia gramatical llamados OIL y RPNI [6] para predecir el posible resultado del tratamiento y así ayudar a prevenir la falla y complicaciones del mismo o para establecer un tratamiento alternativo más adecuado. Estas técnicas han sido aplicadas con éxito en los campos como la biología computacional y el procesamiento del lenguaje natural [7].

Se realizan 54 experimentos para OIL y 54 experimentos para RPNI; en cada uno los experimentos cada algoritmo se entrena y evalúa mediante una validación cruzada 4 iteraciones. Se usan las métricas Accuracy, Precision, Recall y F1-Score para la evaluación. Se llegó a resultados satisfactorios en el trabajo, logrando en múltiples experimentos una tasa muy competitiva del 90 % de Accuracy para RPNI y 68.8 % de Accuracy para OIL.

Palabras Clave: leishmaniasis, enfermedad parasitaria, bioinformática, expresión génica, inteligencia artificial, aprendizaje automático, machine learning, inferencia gramatical, teoría de autómatas

Descripción del Problema

2.1. Descripción del Problema

Por su fuerte aumento de incidencia, la leishmaniasis es considerada una enfermedad grave que ha reaparecido en Colombia y en el mundo. La Organización Mundial de la Salud (OMS) ha anunciado y reiterado que la leishmaniasis es una enfermedad desatendida, es decir, una enfermedad estrechamente relacionada con la pobreza, por lo que hay poca inversión en investigaciones y estrategias para desarrollar tratamientos eficaces [3].

La leishmaniasis es causada por un parásito protozoario del género *Leishmania*, que cuenta con más de 20 especies diferentes en el mundo. La leishmaniasis se transmite por la picadura de flebótomos (moscas de arena) hembra infectados, que tienen que ingerir sangre para producir huevos. La enfermedad existe en tres formas clínicas: leishmaniasis visceral, cutánea y mucocutánea. De estas tres, la leishmaniasis cutánea (enfoque para esta investigación) es la forma más frecuente en Colombia [1]. Ésta causa daños y desfiguraciones en la piel en áreas expuestas del cuerpo humano, especialmente daño ulcerativo, el cual deja cicatrices de por vida y causar discapacidades graves que afectan profundamente las vidas de los pacientes, ya que si bien este tipo de leishmaniasis no es mortal, causa pérdidas económicas, estigma social y posibles complicaciones de salud secundarias en los pacientes. Aproximadamente el 95 % de los casos de leishmaniasis cutánea ocurren en las Américas, la Cuenca del Mediterráneo, Oriente Medio y Asia Central [8]. Se calcula que los casos nuevos cada año oscilan entre 600.000 y 1 millón.

Como principal factor, la pobreza aumenta el riesgo de contraer leishmaniasis. Las viviendas precarias y las deficiencias de saneamiento de los hogares (por ejemplo, la ausencia de sistemas de gestión de residuos o de alcantarillado) pueden promover el desarrollo de los lugares de cría y reposo de los flebótomos y aumentar su acceso a la población humana [8]. Los flebótomos se ven atraídos por el hacinamiento, puesto que constituye una buena fuente de ingesta de sangre. Ciertas pautas de comportamiento humano (por ejemplo, dormir a la intemperie o en el suelo) pueden aumentar el riesgo de contraer la enfermedad.

Debida a la reducida atención que se le presta a la enfermedad en Colombia y en los demás países afectados, no se cuenta con la suficiente experiencia y conocimiento médico para garantizar un tratamiento completamente seguro y exitoso. La medicina de primera línea para la enfermedad, las sales de antimonio o antimoniales pentavalentes (conocida comercialmente como Glucantime), tiene

un porcentaje de falla terapéutica que varía desde el 19 % hasta el 75 % [2], y en hasta el 64 % de los casos, causa efectos negativos o adversos en los pacientes [9]. Los efectos adversos clínicos notificados con mayor frecuencia incluyen fuerte dolor musculoesquelético, trastornos gastrointestinales y dolor de cabeza leve a moderado [9].

El antimonio es un metaloide similar al arsénico y es altamente tóxico, pero es el componente principal de la medicina más efectiva hasta la fecha. El hecho de que el tratamiento con Glucantime falle se debe a factores como su alta toxicidad, al estado inmunológico de los pacientes, o por el abandono del tratamiento por parte de los mismos; ya que estos no siempre llegan a resistir los 20 días en los que se les inyecta el Glucantime diariamente, lo cual los hace experimentar niveles de dolor muscular muy elevados que requieren de supervisión médica.

El fracaso del tratamiento para la leishmaniasis cutánea implica un riesgo y empeoramiento en la calidad de vida de los pacientes, ya que estos no solo tienen que lidiar con cicatrices y discapacidad a causa de la enfermedad que no pudo ser curada, sino también con las severas complicaciones de salud debidas al tratamiento fallido.

Colombia es el segundo país de latinoamérica con más casos de la enfermedad en la actualidad [10]. Es imperativo poder mitigar esta probabilidad previniendo en lo posible el suministro perjudicial del medicamento con base en las condiciones inmunológicas del paciente previas al tratamiento. En su lugar, se emplearían otros tratamientos con medicinas como el pentamidina, anfotericina B o miltefosina, entre otros [11]; las cuales son usadas comúnmente cuando el tratamiento con Glucantime no pudo curar la enfermedad. Si se pudiera saber que el tratamiento con sales de antimonio fallará, —y se emplean tratamientos alternativos desde el principio—, se aumenta las probabilidades de que los pacientes puedan ser curados de una forma más rápida y menos dolorosa, lo cual es primordial para las poblaciones pobres y rurales de zonas tropicales y selváticas del territorio colombiano, las cuales son las principalmente afectadas.

Frente a toda esta problemática, es evidente el peso de la importancia de tomar una decisión bien fundamentada en datos y experiencia médica para realizar el estrictamente necesario, seguro y efectivo suministro de la medicina.

La interrogante que surge entonces es, ¿Cómo se podría apoyar la prevención de tratamientos fallidos mediante Glucantime para la leishmaniasis cutánea en Colombia? Como posible solución se propone emplear técnicas de inferencia gramatical del área de aprendizaje automático que aprovechen los datos genéticos de los pacientes para predecir el posible resultado del tratamiento. Se proponen técnicas de inferencia gramatical debido a su eficacia para trabajar con problemas en los cuales la información tiene cierta estructura, como es el caso con los datos genéticos. Estas técnicas han sido usadas exitosamente en campos como el reconocimiento de patrones, la biología computacional y el procesamiento del lenguaje natural [7].

2.2. Objetivos

2.2.1. Objetivo General

Predecir la efectividad del tratamiento con Glucantime para la leishmaniasis cutánea empleando técnicas de inferencia gramatical con base en los datos génicos de pacientes.

2.2.2. Objetivos Específicos

- Definir una representación gramatical adecuada para los datos génicos.
- Desarrollar las técnicas escogidas.
- Pre-procesar los datos génicos.
- Diseñar experimentos.
- Ejecutar experimentos.
- Evaluar el desempeño de predicción de las técnicas usadas.

Marco teórico

Expresión génica La expresión génica es el proceso mediante el cual las instrucciones que se encuentran codificadas en nuestro ADN se convierten en un producto funcional, como una proteína. Este estricto y regulado proceso permite que las células puedan responder a su entorno. Podría decirse que la expresión génica actúa como un botón de encendido o apagado que dicta cuales proteínas son producidas frente a algún estímulo, y en cuanto volumen [12].

El proceso de expresión génica se divide en dos pasos clave: Transcripción y traducción [12]. La transcripción es cuando el ADN de un gen se copia para producir una transcripción de ARN llamada ARN mensajero (ARNm), teniendo esto lugar en el núcleo de la célula. La traducción se produce después de que el ARN mensajero ha llevado el 'mensaje' transcrito desde el ADN a las fábricas de producción de proteínas en la célula, denominadas ribosomas.

La dificultad para tratar la enfermedad reside en las complejas interacciones que se dan entre el parásito y el sistema inmune [4], estando este a su vez estrechamente relacionado con la expresión génica al ser parte fundamental para la capacidad de un organismo de generar una inmunidad sólida frente a los patógenos [5]. Leishmania es un parásito adaptado a su hospedador que a lo largo de los años ha ido desarrollando numerosas estrategias para lograr evadir los múltiples mecanismos que tiene el sistema inmune para eliminarla [13].

Secuenciación de ARN La secuenciación de ARN (Abreviada comúnmente como 'RNA-Seq' en inglés) es una técnica particular de secuenciación basada en tecnología que utiliza secuenciación de próxima generación (NGS) para revelar la presencia y cantidad de ARN en una muestra biológica en un momento dado [14]. Ésta facilita la capacidad de observar procesos como los cambios en la expresión génica en diferentes grupos o tratamientos [15].

En los datos proporcionados por el CIDEIM, la secuenciación ARN se realiza con muestras biológicas de eosinófilos, monocitos, y neutrófilos; los cuales son tipos de glóbulos blancos en la sangre. Estas son células del sistema inmunológico que participan en la protección del cuerpo contra las enfermedades infecciosas, parásitos e invasores extraños.

Según lo anterior, las mediciones de expresión génica de pacientes con las que se disponen en este trabajo proporcionan una especie de fotografía en el tiempo que permite observar el estado del sistema inmune de los pacientes previo al tratamiento.

Aprendizaje automático El aprendizaje automático o machine learning (ML) se ocupa del aprendizaje automático de las computadoras sin que estas tengan que ser programadas explícitamente. Se centra en realizar predicciones basadas en datos y tiene varias aplicaciones en el campo de la bioinformática [16]. La bioinformática implica el procesamiento de datos biológicos utilizando enfoques basados en la computación y las matemáticas. En los últimos años, el ML se está aplicando en seis subcampos clave de la bioinformática, como microarrays, evolución, biología de sistemas, genómica, minería de textos y proteómica [16].

Inferencia gramatical La inferencia gramatical (IG) es transversal a una serie de campos que incluyen el aprendizaje automático, la teoría de lenguajes formales, el reconocimiento de patrones sintácticos y estructurados, la biología computacional, el reconocimiento de voz, etc [17]. El problema a resolver en la IG es el aprendizaje de la gramática o autómatas que describe un lenguaje a partir de los datos o muestras de observación del mismo.

E. Mark Gold introdujo la identificación del lenguaje en el límite indicando que un algoritmo adecuado eventualmente aprenderá con éxito un lenguaje cuando se proporcione una presentación completa que incluya tanto ejemplos que pertenecen al lenguaje (muestras positivas) como ejemplos que no pertenecen al mismo (muestras negativas) [18]. Existen varios algoritmos para identificar lenguajes regulares basados en el teorema de Gold que utilizan muestras positivas y negativas, en los cuales se encuentran los dos propuestos en este trabajo: RPNI y OIL. Estos algoritmos logran la tarea de aprendizaje del lenguaje a través de autómatas finitos de fusión de estados (o state merging) deterministas y no-deterministas, respectivamente.

La fusión de estados en autómatas finitos (Ver Figura 3.1) da como resultado autómatas generalizados que aceptan al menos tantas cadenas como los autómatas no fusionados. Es decir que el lenguaje expresado por los autómatas no fusionados es un subconjunto del lenguaje expresado por los autómatas fusionados. En este sentido, el objetivo es generalizar y expresar el lenguaje determinado mediante un autómata que acepte todas las muestras positivas y ninguna de las negativas. Se dice entonces que los algoritmos escogidos para este estudio, OIL y RPNI, cada uno busca fusionar los estados de un autómata finito inicial para dar como resultado un autómata más general, el cual posibilita realizar predicciones a partir de muestras positivas o negativas que nunca ha observado antes.

3.1. Trabajos Relacionados

DNA Region Grammatical Inference En este artículo se ilustra el uso del algoritmo ALERGIA de inferencia gramatical para generar la definición de un lenguaje para distinguir entre ADN codificante y no codificante [18]. El aprendizaje se realiza mediante muestras positivas e información estadística para inferir lenguajes para codificar el ADN. Poder distinguir entre las dos diferentes secuencias tiene significación analítica y aplicaciones importantes en la biología computacional.

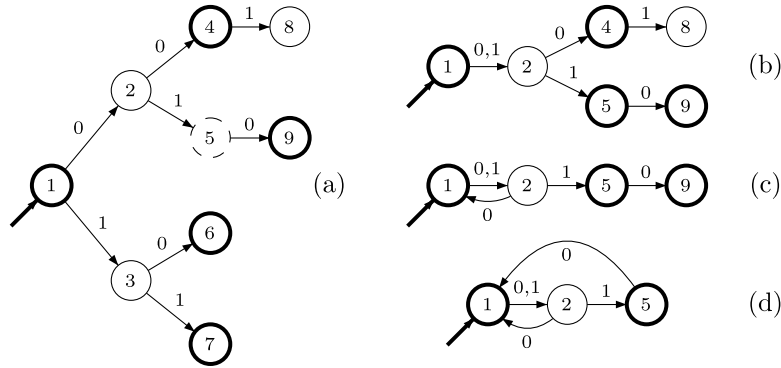


Figura 4.1: Árbol de Prefijos de Moore y Autómatas Intermedios, Producidos al Ejecutar el Algoritmo *RPNI* sobre la Entrada $D_+ = \{\varepsilon, 00, 11, 010, 10\}$ y $D_- = \{0, 1, 001\}$.

Figura 3.1: Fusión de estados producidos por *RPNI* [6]. Estados con borde oscuro son de aceptación, y el resto de rechazo. Nótese como ambos (a) y (d) aceptan las cadenas de D_+ y rechazan las de D_- , siendo (d) más general (puede aceptar o rechazar cadenas/palabras nunca antes observadas).

Variable gene expression and parasite load predict treatment outcome in cutaneous leishmaniasis Según el reporte *Predicting treatment outcome for leishmaniasis* del portal de noticias ScienceDaily [19], investigadores de Brasil han identificado biomarcadores que predicen qué enfermedad de leishmaniasis de los pacientes se resolverá con antimonio y qué pacientes deben recibir una terapia alternativa desde el principio [20]. Usando datos de pacientes con leishmaniasis tratados en Brasil, los investigadores encontraron varios genes cuya expresión se correlacionó con el resultado del tratamiento. Utilizaron una colección de biopsias de piel tomadas antes del tratamiento de 21 personas con leishmaniasis y 7 personas sin la enfermedad que sirvieron como controles. Al reducir los muchos genes que identificaron en este análisis para centrarse en los 250 principales más variables entre los pacientes, encontraron un subconjunto que se correlacionó con el fracaso del tratamiento [19].

Observando un segundo conjunto de datos de diferentes pacientes, pudieron confirmar que ocho de estos genes podían predecir el fracaso del tratamiento en ambos grupos [19]. En ese grupo de 8 genes estaban algunos de los genes que los investigadores ya sabían que eran actores clave en la enfermedad, debido a estudios previos en humanos y ratones realizados en el año 2017.

En la actualidad, este equipo de investigación de Brasil se encuentra trabajando para ver si un método menos invasivo que una biopsia (La cual requiere de una muestra de tejido del paciente) puede proporcionar datos confiables de la enfermedad [19].

Preparación de los Datos

4.1. Descripción

Todos los datos utilizados en este estudio fueron provistos por el CIDEIM de Cali. El CIDEIM es un centro de investigación, desarrollo tecnológico y formación de recurso humano en el campo de la salud. Su propósito es buscar alternativas para disminuir el impacto negativo y los costos de las enfermedades infecciosas [21]. Este centro nacional e internacional es uno de los principales beneficiarios frente a la presente investigación, siendo ellos los que proporcionaron los datos genéticos que son usados en el proyecto, además de la información de trasfondo de la enfermedad y asuntos específicos de los datos brindados; todo bajo el asesoramiento de la Dra. Maria Adelaida Gómez, coordinadora de la Unidad de Biología Molecular y Bioquímica del Programa de investigación de la leishmaniasis del CIDEIM de Cali.

Los datos se encuentran divididos en 7 conjuntos (Ver Cuadro 4.1) que ilustran la información genética de 14 pacientes anónimos durante diferentes citas médicas. Cada paciente está identificado mediante un código único de 4 números y un número para cada uno de los momentos o citas en los que se realizó la medición (1, 2 o 3).

Se cuenta con datos genéticos de 3 tipos diferentes de glóbulos blancos para cada paciente: neutrófilos (neutrophils), eosinófilos (eosinophils) y monocitos (monocytes); medidos a su vez cada uno en 3 momentos diferentes: primera, segunda y tercera cita médica (Ver Cuadro 4.2). Los datos de la primera cita son previos a iniciar el tratamiento, los de la segunda durante, y los de la última luego de terminar el tratamiento. Cada gen presente en estos conjuntos de datos representa una fila y cuenta con un código único que lo identifica. En estos conjuntos de datos genéticos, no hay registros con valores nulos. Para los datos que contienen la etiqueta de salida o resultado de tratamiento (si el paciente curó o no), es posible que sí se tengan valores nulos.

Los conjuntos de datos con nombre `...clinical_table...` (Cuadro 4.3) cuentan con una gran cantidad de columnas (59 cada uno) sobre información conjunta de genes de los 14 pacientes. Entre todos estos atributos, solo nos interesan y se usan `deseq_logfc` y `deseq_adjp` (Cuadro 4.3). El resto de columnas no son relevantes para este enfoque, según se discutió con la Dra. Maria Adelaida Gomez del CIDEIM.

Cuadro 4.1: Resumen de conjuntos de datos utilizados.

Conjunto de datos	Columnas	Filas	Descripción
20210118_EXP_ESPECIAL_TMRC3.csv	57	24	Especifica los datos demográficos de cada paciente, incluyendo resultado del tratamiento (exitoso, fallido o inconcluso)
neutrophil_clinical_table-v202106.xlsx	59	9144	Contiene información de expresión génica de 8791 genes de neutrófilos observados en los pacientes en general
neutrophil_cpm_before_batch-v202106.xlsx	29	19941	Especifica la cantidad de ARN de cada gen de neutrófilos medida durante la primera, segunda y/o tercera cita de cada paciente.
eosinophil_clinical_table-v202106.xlsx	59	10632	Contiene información de expresión génica de 10475 genes de eosinófilos observados en los pacientes en general
eosinophil_cpm_before_batch-v202106.xlsx	21	19941	Especifica la cantidad de ARN de cada gen de eosinófilos medida durante la primera, segunda y/o tercera cita de cada paciente.
monocyte_clinical_table-v202106.xlsx	59	11026	Contiene información de expresión génica de 10653 genes de monocitos observados en los pacientes en general.
monocyte_cpm_before_batch-v202106.xlsx	28	19941	Especifica la cantidad de ARN de cada gen de monocitos medida durante la primera, segunda y/o tercera cita de cada paciente.

Cuadro 4.2: Fragmento de '**neutrophil_cpm_before_batch-v202106.xlsx**'. X1034n1 hace referencia a los datos de un paciente identificado con el código X1034 durante su cita número 1. La fila del gen 'ENSG00000000003' en la columnas X1017n1 reporta cero presencia de ARN mensajero de ese gen en los neutrófilos del paciente X1017 en la cita 1.

row.names	X1017n1	X1034n1	X1034n2	X2052n2	X2052n3
ENSG00000000003	0	0	0	0	0.009
ENSG00000000005	0	0	0	0	0
ENSG000000000419	68.399	36.953	33.276	43.935	40.839
ENSG000000000457	37.484	29.347	27.730	29.372	20.321
ENSG000000000460	11.979	12.480	9.063	8.689	9.372
ENSG000000000938	1221.924	1497.330	1627.961	1149.792	1124.396
ENSG000000000971	0	0.292	0.270	0	0.237
ENSG00000001036	3.091	4.290	4.193	1.162	1.688
ENSG00000001084	3.091	1.657	2.029	4.222	3.190
ENSG00000001167	56.420	47.288	46.397	48.280	47.453

Cuadro 4.3: Fragmento de '**neutrophil_clinical_table-v202106.xlsx**' con los atributos relevantes. **deseq_logfc**: DESeq Log Fold Change, **deseq_adj**: DESeq Adjusted P-value

row.names	deseq_logfc	deseq_adj
ENSG000000000419	-0.1392	0.5728
ENSG000000000457	0.2711	0.2447
ENSG000000000460	0.5703	0.1384
ENSG000000000938	-0.3449	0.1337
ENSG00000001036	-0.1356	0.7356
ENSG00000001084	0.3595	0.3185
ENSG00000001167	0.02867	0.9089
ENSG00000001461	-0.03639	0.9316
ENSG00000001497	-0.726	0.1419
ENSG00000001629	0.1646	0.5349

La columna **deseq_logfc** son datos provistos y recomendados por la experta en el tema, la Dra María Adelaida Gómez. La métrica corresponde a un cálculo realizado sobre los valores de cantidad de ARN para un mismo gen entre todos los pacientes usando DESeq, que es un análisis de expresión diferencial basado en la distribución Gamma-Poisson.

FC significa 'Fold change' y describe cuánto cambia una cantidad o valor entre una medición original y una medición posterior. A los resultados anteriores se les calcula el logaritmo base 10, y se obtiene el resultado final, que es el reportado en la tabla.

En general, **deseq_logfc** indica cuanto disminuyó o aumentó el valor de ARN para un gen entre los pacientes que curan con el tratamiento y los que no curaron. Los genes que presentaron mayor cambio según esta métrica tanto en aumento como en inhibición (valores positivos muy altos y negativos muy bajos) son los que más tarde se utilizarán para la creación de alfabetos en la sección 4.3.

La columna **deseq_adjp** es el valor P ajustado de las mediciones de **deseq_logfc**. Es decir, esta métrica es la probabilidad de que el valor de DESeq que se obtuvo para la medición de un gen haya sido consecuencia del azar. Es mediante este atributo que se realiza uno de los filtros descritos en la sección de Pre-procesamiento de los datos.

Cabe resaltar que no todos los pacientes cuentan con información de los 3 tipos de célula, ni tampoco con las 3 citas. La ausencia o no de una cita en un paciente depende de muchos factores, como que el paciente no haya podido asistir a la cita de control debido a que vive en un lugar alejado al centro médico, o incluso que éste mismo haya decidido desistir con el tratamiento. Por otra parte, la ausencia de los datos de un tipo de célula se debe a problemas en el procesamiento de la muestra. Por ejemplo, en el caso de los eosinófilos, estos solo corresponden al 1 % de de las células sanguíneas, por lo cual es muy difícil aislarlos, y en algunos pacientes esta proporción pueden ser aún más baja. En otros casos, el procesamiento de los datos pudo haber sufrido de degradación, como es el caso con los neutrófilos: estos están llenos de enzimas que degradan el ARN, lo cual hace que las muestras obtenidas no sean aptas para secuenciación.

En secciones subsecuentes veremos como se utilizan todas las citas disponibles de cada célula de cada paciente tanto para el entrenamiento como la prueba de las técnicas. Se podría pensar que solo se debería entrenar y probar los modelos con las primeras citas, sin embargo se estaría desperdiciando la mayoría de los datos que de por si ya son reducidos, sin mencionar que la capacidad de dar un pronóstico es información valiosa.

4.2. Pre-procesamiento

El objetivo del pre-procesamiento consiste en obtener los genes que pueden tener mayor incidencia en el resultado del tratamiento, y para esto se seleccionan los que presentan mayores niveles de inhibición o sobre-expresión génica entre todos los pacientes.

Los archivos a procesar para tal fin son los siguientes:

- neutrophil_clinical_table-v202106.xlsx
- eosinophil_clinical_table-v202106.xlsx
- monocyte_clinical_table-v202106.xlsx

El procesamiento de cada uno de estos conjuntos se ilustra a continuación:

1. Se hace un filtro por la columna **deseq_adjp** de tal forma que solo prevalecen las filas que tienen un valor P ajustado menor o igual a 0.05 (valor significativo estadísticamente hablando).
2. Se organiza cada tabla por la columna **deseq_logfc** de forma descendiente, de tal forma que los genes con mayor valor queden arriba y los de menor valor queden al final de la tabla. Esto permite el fácil acceso a los genes con mayor nivel de expresión génica en inhibición y sobre-expresión.
3. Se eliminan todas las columnas, a excepción de **deseq_logfc**, **deseq_adjp**, y la columna con el código identificador de cada gen.

Una vez procesados los conjuntos de datos de información génica conjunta, se pasa a procesar los conjuntos de datos de citas de pacientes:

- neutrophil_cpm_before_batch-v202106.xlsx
- eosinophil_cpm_before_batch-v202106.xlsx
- monocyte_cpm_before_batch-v202106.xlsx

Para los datos anteriores, se realiza el siguiente procesamiento:

1. Cada paciente tiene un 'outcome' o resultado luego de haber pasado por el tratamiento con Glucantime: 0 (cura), 1 (falla), 2 (inconcluso) o NULL. Se elimina la información de 4 pacientes de todos los conjuntos de datos porque o no aparecen en la tabla de outcomes 20210118_EXP_ESPECIAL_TMRC3.csv o tienen un resultado de tratamiento inconcluso. Esto se realiza por el hecho de que este valor de outcome es la etiqueta de cada dato en las técnicas de aprendizaje supervisado, y es crucial para el entrenamiento y prueba de las mismas.
2. Para el conjunto **monocyte_cpm_before_batch-v202106.xlsx**, se elimina la columna 'X1034m2.', porque está en un formato incorrecto (tiene un punto al final), y porque esta está repetida, al ya existir la columna 'X1034m2'.

Los pasos descritos hasta el momento concluyen la primer parte del pre-procesamiento. La cantidad de pacientes resultante de los que se tiene información es 10, y las citas disponibles por célula de cada uno pueden observarse en la Figura 4.1. La cantidad de muestras por cada tipo de célula se pueden observar en el Cuadro 4.4.

Cuadro 4.4: Cantidad de muestras por tipo de célula después del primer pre-procesamiento.

Célula	Muestras pacientes que curan	Muestras pacientes que no curan	Total
Neutrófilos	9	11	20
Eosinófilos	8	8	16
Monocitos	9	10	19

célula	Neutrófilos			Eosinófilos			Monocitos		
cita	C1	C2	C3	C1	C2	C3	C1	C2	C3
X2052	0	1	1	1	0	0	0	1	1
X2065	1	1	0	0	1	1	1	1	0
X2066	1	1	1	0	1	0	1	1	1
X2068	1	1	1	1	1	1	1	1	1
X2071	1	0	0	0	0	0	1	0	0
X2073	1	1	1	1	1	1	1	1	1
X2162	1	0	0	1	0	0	1	0	0
X2167	0	0	1	0	0	1	1	1	0
X2168	1	1	1	1	1	1	0	1	1
X2172	1	0	0	1	0	0	0	0	0

Figura 4.1: Datos de cita disponibles por cada célula para cada paciente luego de primer pre-procesamiento. 1 indica su existencia y 0 su ausencia. C1: Cita #1, C2: Cita #2, C3: Cita #3.

4.3. Creación de alfabeto

En el contexto de autómatas de estados finitos, un alfabeto es un conjunto finito y no vacío de símbolos (También llamados caracteres). Normalmente, se usa el símbolo Σ para denotar un alfabeto. Una cadena o palabra sobre un alfabeto Σ es una secuencia finita de símbolos extraídos del mismo, y un lenguaje se define como un conjunto finito o infinito de cadenas (palabras) de símbolos que son aceptadas por un autómata particular. Un autómata acepta una palabra si para cada símbolo de entrada de esta, existe una transición siguiente en el autómata, hasta que se alcance un estado final leyendo la palabra completa.

A lo que se quiere llegar con la creación de un alfabeto es poder tener los bloques de construcción para representar el estado de expresión génica de un paciente en una cita particular, de tal forma que esto pueda ser entendido por las técnicas de inferencia gramatical, que hacen uso de palabras o cadenas de símbolos tanto para su entrenamiento (generación de un autómata) como evaluación (evaluación de cadenas sobre el autómata).

Esto se hace transformando cada uno de los genes en los datos del paciente que presentan sobreexpresión o infraexpresión, a símbolos únicos y representativos. En la sección 4.4 se observa cómo se codifica o transforma cada gen a un símbolo.

Se resalta el hecho de que los 3 tipos de células en este estudio son información de expresión génica diferente, y cada una contará con su propio alfabeto y modelos. Es decir que para el tipo de célula neutrófilo se crea un alfabeto el cual es usado para crear palabras a partir de datos de medición de ARN de genes de neutrófilos que producirán modelos de neutrófilos; y lo mismo para las otras células. En el Capítulo 7 (Algoritmo Integrador) se combinan los resultados de los mejores modelos de cada tipo de célula para producir una predicción final sobre el resultado del tratamiento.

Comúnmente se pensaría que para la creación de un alfabeto se necesitarían utilizar caracteres, como 'a', 'b' o 'z'. No obstante, la cantidad de caracteres o símbolos que se necesitan en éste trabajo, en su mayoría de casos, sobrepasa la cantidad de caracteres en el alfabeto latino. No solo esto, sino que la legibilidad de estos símbolos se hace confusa al no poder diferenciar entre genes infra o sobre expresados de una forma simple e inmediata. Es por esto que cada símbolo en la creación de los alfabetos en este estudio está representado por un número entero, como '1', '20' o '60'.

El primer paso para crear el alfabeto de un tipo de célula es realizar el pre-procesamiento descrito en la sección anterior con su tabla (Sección 4.1, Cuadro 4.3), de tal forma que se tienen solo las filas con un valor P ajustado menor a 0.05 y solo 2 columnas: **deseq_adjp** y **deseq_logfc**.

Una vez que se tiene esta tabla inicial, se elige una cantidad de genes a considerar n , y seleccionar las primeras y ultimas $\frac{n}{2}$ filas de esta tabla (Figura 4.2). Se resalta que n es cualquier número entero, y solo debe ser menor o igual a la cantidad de genes disponibles en las tablas.

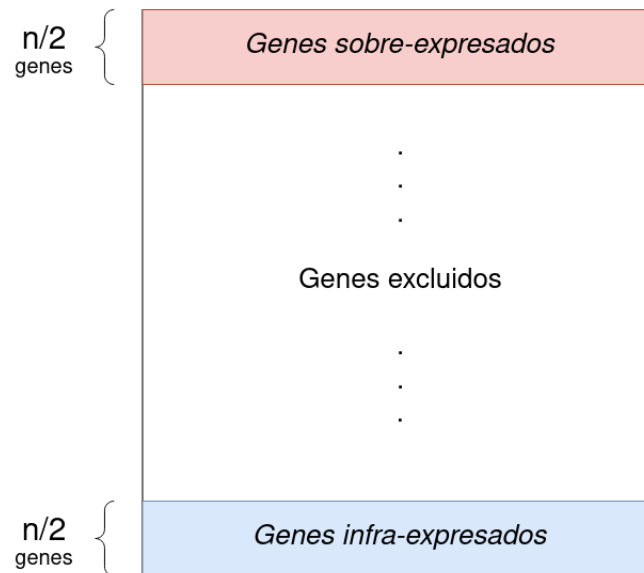


Figura 4.2: Filtración de genes para tablas de datos de células con pre-procesamiento inicial.

Luego, se toman los genes seleccionados y se hace una intersección con los genes del conjunto de datos que tiene la información de citas de pacientes de la célula correspondiente. El resultado de la intersección nos da la tabla (Como la Tabla 4.2) con los valores observados en los pacientes de los genes con mayor inhibición y cambio en general, según la cantidad n definida. Los valores de ARN para cada gen pueden ser muy altos, o muy bajos, o mantenerse en un promedio (con respecto a la media y desviación estándar de los otros pacientes).

En las tablas 4.5, 4.6, 4.7 se recopilan los códigos de los genes seleccionados para cada tipo de célula, para cada valor de n . Notar que conforme se aumenta el tamaño de n , se incluyen todos los genes del valor de n anterior y se le añaden nuevos.

Cuadro 4.5: Valores de n usados en la creación de alfabetos para Neutrófilos y los genes seleccionados para cada uno.

Genes de Alfabetos de Neutrófilos									
$n = 6$	$n = 8$	$n = 10$	$n = 12$	$n = 14$	$n = 16$	$n = 20$	$n = 30$	$n = 40$	
ENSG000000003137	ENSG000000003137	ENSG000000003137	ENSG000000003137	ENSG000000003137	ENSG000000003137	ENSG000000003137	ENSG000000003137	ENSG000000003137	ENSG000000003137
ENSG00000146205	ENSG00000115919	ENSG00000076716	ENSG00000076716	ENSG00000076716	ENSG00000076716	ENSG00000069020	ENSG00000069020	ENSG00000069020	ENSG00000069020
ENSG00000175018	ENSG00000146205	ENSG00000115919	ENSG00000115919	ENSG00000115919	ENSG00000115919	ENSG00000076716	ENSG00000069020	ENSG00000069020	ENSG00000069020
ENSG00000185736	ENSG00000165966	ENSG00000146205	ENSG00000128731	ENSG00000128731	ENSG00000128731	ENSG00000115919	ENSG00000076716	ENSG00000076716	ENSG00000076716
ENSG00000196526	ENSG00000175018	ENSG00000148513	ENSG00000146205	ENSG00000146205	ENSG00000146205	ENSG00000128731	ENSG00000105967	ENSG00000079482	ENSG00000079482
	ENSG00000185736	ENSG00000175018	ENSG00000165966	ENSG00000165966	ENSG00000165966	ENSG00000148513	ENSG00000116157	ENSG00000105967	ENSG00000105967
	ENSG00000196526	ENSG00000196526	ENSG00000185736	ENSG00000185736	ENSG00000185736	ENSG00000162669	ENSG00000120675	ENSG00000115919	ENSG00000115919
	ENSG00000248405	ENSG00000248405	ENSG00000196526	ENSG00000196526	ENSG00000175018	ENSG00000165966	ENSG00000128731	ENSG00000116157	ENSG00000116157
		ENSG00000277150	ENSG00000203812	ENSG00000196526	ENSG00000185736	ENSG00000170214	ENSG00000135116	ENSG00000120675	ENSG00000120675
			ENSG00000248405	ENSG00000198959	ENSG00000186654	ENSG00000170381	ENSG00000141576	ENSG00000122728	ENSG00000122728
				ENSG00000203812	ENSG00000196526	ENSG00000175018	ENSG00000146205	ENSG00000122729	ENSG00000122729
				ENSG00000203812	ENSG00000198959	ENSG00000177294	ENSG00000148513	ENSG00000128731	ENSG00000128731
				ENSG00000248405	ENSG00000203812	ENSG00000185736	ENSG00000153823	ENSG00000134809	ENSG00000134809
					ENSG00000248405	ENSG00000186654	ENSG00000154240	ENSG00000135116	ENSG00000135116
					ENSG00000277150	ENSG00000196526	ENSG00000154447	ENSG00000141576	ENSG00000141576
						ENSG00000198959	ENSG00000162669	ENSG00000146205	ENSG00000146205
						ENSG00000203812	ENSG00000165966	ENSG00000148513	ENSG00000148513
						ENSG00000248405	ENSG00000170214	ENSG00000153823	ENSG00000153823
						ENSG00000277150	ENSG00000170381	ENSG00000154240	ENSG00000154240
							ENSG00000175018	ENSG00000154447	ENSG00000154447
							ENSG00000177294	ENSG00000162669	ENSG00000162669
							ENSG00000179044	ENSG00000163046	ENSG00000163046
							ENSG00000185736	ENSG00000165966	ENSG00000165966
							ENSG00000186654	ENSG00000167680	ENSG00000167680
							ENSG00000196526	ENSG00000170214	ENSG00000170214
							ENSG00000198959	ENSG00000170381	ENSG00000170381
							ENSG00000203812	ENSG00000172594	ENSG00000172594
							ENSG00000248405	ENSG00000175018	ENSG00000175018
							ENSG00000277150	ENSG00000177294	ENSG00000177294
								ENSG00000179044	ENSG00000179044
								ENSG00000185736	ENSG00000185736
								ENSG00000186654	ENSG00000186654
								ENSG00000188672	ENSG00000188672
								ENSG00000196526	ENSG00000196526
								ENSG00000198502	ENSG00000198502
								ENSG00000198959	ENSG00000198959
								ENSG00000203812	ENSG00000203812
								ENSG00000248405	ENSG00000248405
								ENSG00000277150	ENSG00000277150

[illegible]

Cuadro 4.7: Valores de n usados en la creación de alfabetos para Monocitos y los genes seleccionados para cada uno.

Genes de Alfabetos de Monocitos										
$n = 6$	$n = 8$	$n = 10$	$n = 12$	$n = 14$	$n = 16$	$n = 20$	$n = 30$	$n = 40$		
ENSG000000108700	ENSG000000108700	ENSG000000108700	ENSG000000108700	ENSG000000108700	ENSG00000039560	ENSG00000039560	ENSG00000039560	ENSG00000039560	ENSG00000039560	ENSG00000039560
ENSG000000142910	ENSG000000108821	ENSG000000108821	ENSG000000108700	ENSG000000108700	ENSG000000108700	ENSG000000102738	ENSG000000102738	ENSG000000102738	ENSG000000102738	ENSG000000102738
ENSG000000162669	ENSG000000142910	ENSG000000128274	ENSG000000108821	ENSG000000108821	ENSG000000108821	ENSG000000108700	ENSG000000108700	ENSG000000108700	ENSG000000102738	ENSG000000102738
ENSG000000164692	ENSG000000162669	ENSG000000129990	ENSG000000128274	ENSG000000128274	ENSG000000108821	ENSG000000108821	ENSG000000108821	ENSG000000108821	ENSG000000105499	ENSG000000105499
ENSG000000165966	ENSG000000163046	ENSG000000142910	ENSG000000129990	ENSG000000129990	ENSG000000129990	ENSG000000128274	ENSG000000128274	ENSG000000127412	ENSG000000108700	ENSG000000108700
ENSG000000168542	ENSG000000165966	ENSG000000163046	ENSG000000142910	ENSG000000142910	ENSG000000131203	ENSG000000131203	ENSG000000129990	ENSG000000129990	ENSG000000118004	ENSG000000118004
	ENSG000000167768	ENSG000000164692	ENSG000000162669	ENSG000000162669	ENSG000000162669	ENSG000000142910	ENSG000000131203	ENSG000000131203	ENSG000000127412	ENSG000000127412
	ENSG000000168542	ENSG000000165966	ENSG000000163046	ENSG000000163046	ENSG000000163046	ENSG000000162669	ENSG000000142910	ENSG000000142910	ENSG000000128274	ENSG000000128274
		ENSG000000167768	ENSG000000164692	ENSG000000164692	ENSG000000164692	ENSG000000163046	ENSG000000152952	ENSG000000152952	ENSG000000129990	ENSG000000129990
		ENSG000000168542	ENSG000000165966	ENSG000000165966	ENSG000000165966	ENSG000000164692	ENSG000000154240	ENSG000000154240	ENSG000000131203	ENSG000000131203
		ENSG000000203812	ENSG000000167768	ENSG000000167768	ENSG000000167768	ENSG000000165966	ENSG000000162669	ENSG000000162669	ENSG000000137571	ENSG000000137571
			ENSG000000168542	ENSG000000168542	ENSG000000168542	ENSG000000167768	ENSG000000163046	ENSG000000142910	ENSG000000142910	ENSG000000142910
				ENSG000000203812	ENSG000000196526	ENSG000000168542	ENSG000000164692	ENSG000000147138	ENSG000000147138	ENSG000000147138
					ENSG000000197057	ENSG000000170214	ENSG000000165966	ENSG000000148219	ENSG000000148219	ENSG000000148219
					ENSG000000203812	ENSG000000177294	ENSG000000167768	ENSG000000152952	ENSG000000152952	ENSG000000152952
						ENSG000000196526	ENSG000000168542	ENSG000000154240	ENSG000000154240	ENSG000000154240
						ENSG000000197057	ENSG000000170214	ENSG000000157985	ENSG000000157985	ENSG000000157985
						ENSG000000203812	ENSG000000173391	ENSG000000162669	ENSG000000162669	ENSG000000162669
						ENSG000000177294	ENSG00000017294	ENSG000000163046	ENSG000000163046	ENSG000000163046
							ENSG000000177469	ENSG000000164692	ENSG000000164692	ENSG000000164692
							ENSG000000182901	ENSG000000165966	ENSG000000165966	ENSG000000165966
							ENSG000000183117	ENSG000000167768	ENSG000000167768	ENSG000000167768
							ENSG000000188672	ENSG000000168542	ENSG000000168542	ENSG000000168542
							ENSG000000196526	ENSG000000169432	ENSG000000169432	ENSG000000169432
							ENSG000000197057	ENSG000000170214	ENSG000000170214	ENSG000000170214
							ENSG000000203812	ENSG000000173391	ENSG000000173391	ENSG000000173391
							ENSG000000203814	ENSG000000177294	ENSG000000177294	ENSG000000177294
							ENSG000000243566	ENSG000000177469	ENSG000000177469	ENSG000000177469
							ENSG000000277150	ENSG00000017807	ENSG00000017807	ENSG00000017807
								ENSG000000182901	ENSG000000182901	ENSG000000182901
								ENSG000000183117	ENSG000000183117	ENSG000000183117
								ENSG000000188672	ENSG000000188672	ENSG000000188672
								ENSG000000196526	ENSG000000196526	ENSG000000196526
								ENSG000000197057	ENSG000000197057	ENSG000000197057
								ENSG000000203812	ENSG000000203812	ENSG000000203812
								ENSG000000203814	ENSG000000203814	ENSG000000203814
								ENSG000000243566	ENSG000000243566	ENSG000000243566
								ENSG000000249242	ENSG000000249242	ENSG000000249242
								ENSG000000277150	ENSG000000277150	ENSG000000277150

4.4. Codificación de genes

Una vez que se han realizado la serie de filtros descritos en la sección 4.3, se ha creado el alfabeto, y se tiene la lista de códigos de genes que se van a considerar, se pasa a la siguiente fase del pre-procesamiento. Esta consiste en codificar la cantidad de ARN mensajero que presenta cada paciente en sus diferentes citas. El primer paso para realizar tal codificación es tomar, por cada gen seleccionado, la media y desviación estándar de la cantidad de ARN de ese gen en cada cita de cada paciente. Estas dos métricas se utilizan para clasificar de cada gen como infra-expresado, sobre-expresado, o estable; como se verá en las siguientes secciones.

Para ilustrar el funcionamiento de las codificaciones, se muestra un ejemplo de una palabra creada en cada una, con base en los datos de la Tabla 4.8. En esta, podemos observar datos de expresión génica de un paciente al igual que la media y desviación estándar de cada gen. En la codificación de genes, se tienen dos opciones: realizar una codificación binaria (2 tipos de símbolos) o una ternaria (3 tipos de símbolos). Ambos tipos de codificación son explicados a detalle en las secciones 4.4.1 y 4.4.2, respectivamente. En este ejemplo, si se usa la codificación binaria, la palabra resultante que describe al paciente Z sería (13, 4, 15, 18, 20) (Tabla 4.9). Alternativamente, si se usa la codificación ternaria con esta misma información, la palabra resultante sería (21, 13, 4, 15, 27, 18, 20) (Tabla 4.10).

Cuadro 4.8: Ejemplo datos de cantidad de ARN de 10 genes en la primer cita de un paciente Z con su código, numero de fila, media y desviación estándar. El número de fila corresponde a la posición de arriba hacia abajo en la que se encuentra el gen en la tabla de información génica conjunta correspondiente.

Fila (i)	Código de Gen	Mean (\bar{x})	Std	Paciente Z Cita 1
1	ENSG00000006432	0.603	0.492	0.773
2	ENSG00000006451	8.794	3.014	12.753
3	ENSG00000006453	4.223	1.691	6.569
4	ENSG00000006459	216.116	61.661	124.205
5	ENSG00000006468	0.077	0.083	0.386
6	ENSG00000006530	0.510	0.283	0.773
7	ENSG00000006534	26.419	8.325	25.505
8	ENSG00000006555	2.525	1.609	5.024
9	ENSG00000006576	17.569	6.980	22.413
10	ENSG00000006606	0.054	0.102	0.386

4.4.1. Codificación binaria

La Figura 4.3 ilustra los puntos de corte establecidos para clasificar el valor de un gen en alguna de las 2 categorías existentes en la codificación binaria: infra expresado o sobre expresado. La Tabla 4.9 muestra un ejemplo de una palabra creada con esta codificación.

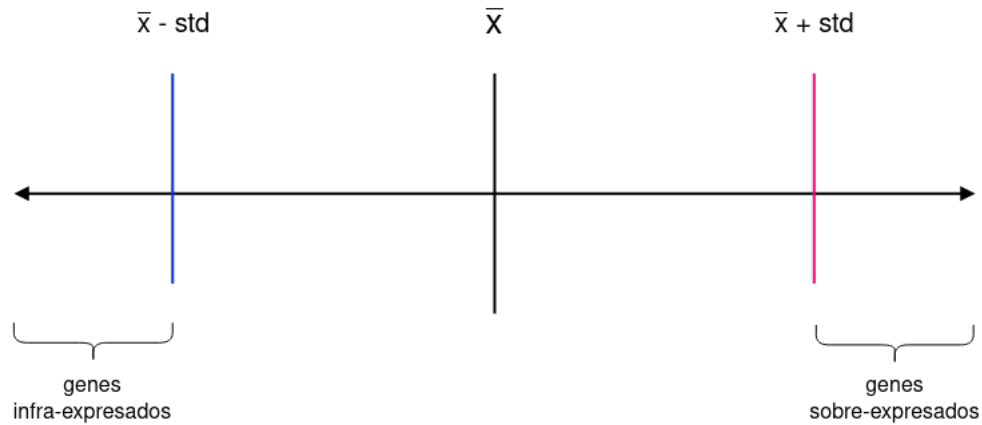


Figura 4.3: Puntos de corte para clasificación de genes según media y desviación estándar en codificación binaria.

Cuadro 4.9: Creación de palabra para paciente Z en codificación binaria según datos del Cuadro 4.8. $n = 10$, tamaño de alfabeto = 20. Palabra resultante = (13, 4, 15, 18, 20)

Fila (i)	$\bar{x}-std$	$\bar{x}+std$	Paciente Z	Clasificación	Símbolo
1	0.111	1.095	0.702	No clasifica	No tiene
2	5.780	11.808	10.753	No clasifica	No tiene
3	2.532	5.914	6.569	Sobreexpresado	$i + n = 13$
4	154.445	277.777	124.205	Infraexpresado	$i = 4$
5	-0.006	0.160	0.386	Sobreexpresado	$i + n = 15$
6	0.227	0.793	0.773	No clasifica	No tiene
7	18.094	34.744	25.505	No clasifica	No tiene
8	0.916	4.134	5.024	Sobreexpresado	$i + n = 18$
9	10.589	24.549	22.413	No clasifica	No tiene
10	-0.048	0.156	0.386	Sobreexpresado	$i + n = 20$

Cuando el valor de ARN registrado en un paciente es inferior a la media menos una desviación estándar, se considera que este gen está inhibido o infra-expresado. A este gen se le asigna un número i entre 1 y n , siendo n la cantidad de genes en la tabla de datos de pacientes determinada. Este i es el número de la fila en la que se encontraba el gen.

Lo anterior quiere decir que los genes que tienen un símbolo entre $[1, n]$ son infra-expresados, mientras que los que tengan un símbolo entre $[n + 1, 2n]$ son sobre-expresados. Es decir que cuando se utiliza una cantidad m de genes para crear un alfabeto con codificación binaria, el tamaño de este alfabeto será $2m$.

Los genes sobre-expresados siempre serán fáciles de reconocer y diferentes a los infra-expresados. Esto ayuda en la legibilidad de las palabras, y también le ayuda a las técnicas usadas. Esto sucede porque a cada gen, dependiendo de su estado de expresión, se le asigna un símbolo único. Lo anterior implica que un mismo gen, en 2 pacientes diferentes, puede tener asociado 2 símbolos diferentes: uno cuando está infra-expresado, y otro cuando está sobre-expresado. Esta distinción de cuando un gen está sobre o infra expresado le da más información a las técnicas para reconocer el lenguaje objetivo, y representa de forma más precisa el estado de expresión génica del paciente.

4.4.2. Codificación ternaria

En la Figura 4.4 podemos observar los puntos de corte establecidos para clasificar el valor de ARN de un gen en alguna de las 3 categorías: infra expresado, sobre expresado, o estable. Se utiliza $\frac{1}{4}$ de desviación estándar porque generó los mejores resultados de desempeño para las técnicas. En la Tabla 4.10 se ve un ejemplo de una palabra creada con esta codificación.

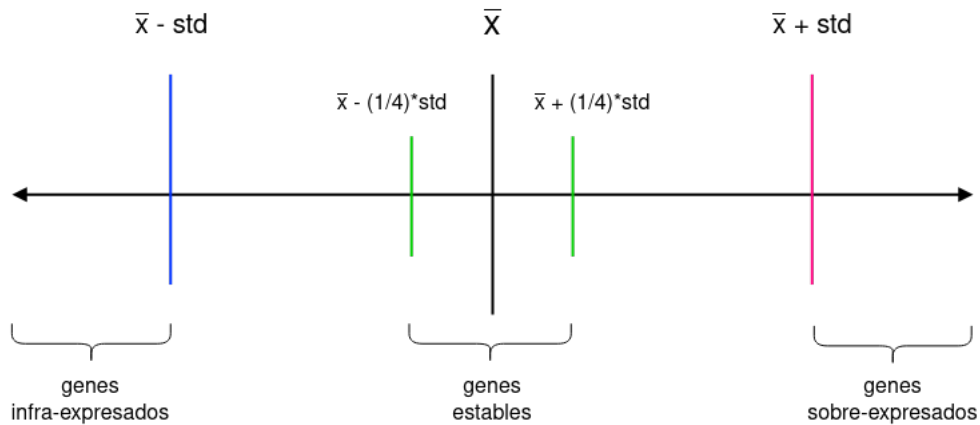


Figura 4.4: Puntos de corte para clasificación de genes según media y desviación estándar en codificación ternaria.

Se incluye la noción de genes 'estables' con el fin de obtener un poco más de información del perfil de expresión génica de cada paciente en la cita correspondiente. Se usa $\frac{1}{4}$ de la desviación estándar para clasificar un gen como estable: si la cantidad de ARN del gen se encuentra entre $\frac{1}{4}$ de la desviación estándar sobre o bajo la media, se dice que el gen se ha mantenido estable, y se le asigna un símbolo entre $[2n + 1, 3n]$; Es decir tomando la fila en la que se encontraba, i , y sumando $2n$.

En esta codificación, de la misma forma que con la codificación binaria, los rangos $[1, n]$ y $[n+1, 2n]$ corresponden a genes infra y sobre expresados, respectivamente. Es decir que si se usan m genes para crear un alfabeto en codificación ternaria, el tamaño de este alfabeto será $3m$.

Se realizaron experimentos con $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{8}$ y $\frac{1}{16}$ como fracción para la desviación estandar. Los experimentos consistieron en generar los alfabetos ternarios con los tamaños establecidos, entrenar las técnicas, evaluar y obtener métricas de desempeño. Cuando se usaron estas fracciones, el desempeño fue inferior a cuando se usó $\frac{1}{4}$. Se captaban demasiados o muy pocos genes estables. $\frac{1}{4}$ captura una cantidad razonable de genes estables y fue el que mejor resultados presentó, razón por la cual se seleccionó.

Cuadro 4.10: Creación de palabra para paciente Z en codificación ternaria según datos del Cuadro 4.8. $n = 10$, tamaño de alfabeto = 30. Palabra resultante = (21, 13, 4, 15, 27, 18, 20)

Fila (i)	\bar{x} -std	$\bar{x}-\frac{1}{4}\text{std}$	$\bar{x}+\frac{1}{4}\text{std}$	\bar{x} +std	Paciente Z	Clasificación	Símbolo
1	0.111	0.480	0.726	1.095	0.702	Estable	$i + 2n = 21$
2	5.780	8.041	9.548	11.808	10.753	No clasifica	No tiene
3	2.532	3.800	4.646	5.914	6.569	Sobreexpresado	$i + n = 13$
4	154.445	200.701	231.531	277.777	124.205	Infraexpresado	$i = 4$
5	-0.006	0.056	0.098	0.160	0.386	Sobreexpresado	$i + n = 15$
6	0.227	0.439	0.581	0.793	0.773	No clasifica	No tiene
7	18.094	24.338	28.500	34.744	25.505	Estable	$i + 2n = 27$
8	0.916	2.123	2.927	4.134	5.024	Sobreexpresado	$i + n = 18$
9	10.589	15.824	19.314	24.549	22.413	No clasifica	No tiene
10	-0.048	0.0285	0.0795	0.156	0.386	Sobreexpresado	$i + n = 20$

4.5. Creación de palabras

Como se ha mencionado antes, las palabras se van a utilizar para representar el estado de expresión génica de un paciente en una cita determinada. También, se definió qué era un lenguaje en este contexto.

En nuestro caso particular, lo que nos interesa es reconocer en qué instancias el tratamiento de un paciente va a fallar, lo cual es una ocurrencia minoritaria entre todos los casos del mundo real de la enfermedad. Se dice entonces que nos interesa generar autómatas que acepten las palabras de pacientes que no van a curar, es decir, que acepten el lenguaje de las palabras de pacientes cuyo tratamiento va a fallar en el futuro. En la inferencia gramatical, a las palabras que pertenecen al lenguaje objetivo también se les llama palabras positivas o muestras positivas. De ahora en adelante en el documento se utilizarán estos términos.

Por otra parte, las palabras que no nos interesan reconocer, o que no hacen parte del lenguaje, son las de pacientes cuyo tratamiento va a ser exitoso, que son la mayoría de casos en esta enfermedad. Se dice entonces que los autómatas que nos interesa generar deben rechazar estas palabras que no pertenecen al lenguaje objetivo. Las palabras que no pertenecen al lenguaje también se les denomina palabras o muestras negativas. Al igual que con las palabras o muestras positivas, se utilizarán estos términos de ahora en adelante en el documento.

Que una palabra sea positiva o negativa es como una etiqueta que identifica a cada una, y es un dato fundamental que es requerido por realizar el entrenamiento y evaluación en ambas técnicas. Estas se crean iterando de forma secuencial sobre los genes presentes de cada paciente, y se aplica la codificación de cada gen como se mostró en las secciones anteriores. El resultado es una tupla de símbolos que representan la palabra de expresión génica de un paciente en una cita particular, y son estas las que se alimentan directamente a las técnicas para su entrenamiento o para recibir una predicción de resultado de tratamiento. Cabe aclarar que, como se describió en la sección 4.1, se cuenta con la información del desenlace del tratamiento para todos los pacientes usados en este estudio. Es decir que cada palabra cuenta con su etiqueta respectiva: el número 1 si es positiva, o el número 0 si es negativa.

4.6. Variación de datos

Con respecto a parámetros a parte del tipo de codificación o tamaño del alfabeto, el orden en el que aparecen los símbolos en una palabra no es importante. Como se discutió con la Dra Maria Adelaida Gomez, el orden de aparición de los genes tiene poca relevancia en términos biológicos. Esto se corroboró cuando se hicieron experimentos observando diferentes ordenes de palabra: estas variaciones no producían ninguna mejora tangible al desempeño, por lo cual este parámetro fue descartado.

Cuadro 4.11: Lista de valores para las variables para realizar los experimentos.

Variable	Posibles valores
Cantidad de genes (n)	6,8,10,12,14,16,20,30,40
Codificación	<i>Binaria, Ternaria</i>

Con lo anterior, los parámetros a variar en el entrenamiento y evaluación de las técnicas se aprecian en el Cuadro 4.11. Es decir que se usaran todas las combinaciones posibles de estas 3 variables para probar las dos técnicas de RPNI y OIL.

Los valores de n que son mostrados fueron seleccionados porque representan de forma adecuada la variedad de resultados posibles. Otros valores mayores o intermedios a los mostrados no brindan resultados novedosos o muy diferentes. Se hicieron experimentos con cantidades de genes en el rango $[1, 100]$, tomando el alfabeto resultante, entrenando las técnicas, y observando los resultados de desempeño. Los valores de n en ese rango no incluidos en el estudio mostraron desempeños similares a las 9 seleccionados.

Por otra parte, se debieron eliminar todas las palabras inconsistentes en el procesamiento de los datos (es decir, palabras iguales en las cuales una pertenece a un paciente que curó y la otra a uno que no curó), reduciendo la cantidad de datos disponibles para entrenar y evaluar las técnicas. En total (es decir, tomando en cuenta todas las variaciones de tamaños de alfabeto, tipo de codificación y tipo de célula), se crearon 990 palabras. 98 de estas eran inconsistentes, y fueron eliminadas. La cantidad de palabras resultante fue 892. En el Cuadro 4.12 se observan las cantidades de símbolos de los 54 alfabetos generados en total para todos los experimentos. Los genes de cada tipo de célula usados para la creación de cada alfabeto binario y ternario son diferentes, como se observó en los Cuadros 4.5, 4.6 y 4.7.

Cuadro 4.12: Cantidad de símbolos para los 9 alfabetos con codificación binaria y los 9 alfabetos con codificación ternaria generados para cada tipo de célula.

Neutrófilos		Eosinófilos		Monocitos	
Alfabetos Binarios	Alfabetos Ternarios	Alfabetos Binarios	Alfabetos Ternarios	Alfabetos Binarios	Alfabetos Ternarios
Cantidad de Símbolos	Cantidad de Símbolos	Cantidad de Símbolos	Cantidad de Símbolos	Cantidad de Símbolos	Cantidad de Símbolos
12	18	12	18	12	18
16	24	16	24	16	24
20	30	20	30	20	30
24	36	24	36	24	36
28	42	28	42	28	42
32	48	32	48	32	48
40	60	40	60	40	60
60	90	60	90	60	90
80	120	80	120	80	120

Estas palabras inconsistentes surgen porque los genes presentes en los conjuntos de datos provistos parecen no ser representativos para el estado de expresión génica de todos los pacientes. Esto quiere decir que, dados los genes en los conjuntos de datos, la cantidad de ARN de cada gen entre citas de múltiples pacientes termina siendo muy similar, causando que cada paciente no esté bien caracterizado o diferenciado con respecto a otros pacientes, lo cual dificulta el reconocimiento de patrones de los algoritmos usados. Algunos pacientes quedan representados con una palabra vacía tanto si curan como si no lo hacen, y en algunos otros casos dos pacientes con resultados de tratamiento diferentes terminan con exactamente la misma palabra. Como trabajo futuro se deja

el elaborar conjuntos de datos de genes más representativos entre los pacientes, de tal forma que la existencia de palabras inconsistentes sea imposible.

Algoritmo RPNI

5.1. Descripción

Algoritmo 19 $RPNI(D_+, D_-)$

```

1:  $M := APM(D_+, D_-)$ 
2:  $lista := \{u_0, u_1, \dots, u_r\}$  // estados de M en orden lexicográfico,  $u_0 = \lambda$  //
3:  $lista' := \{u_1, \dots, u_r\}$ 
4:  $q := u_1$ 
5: while  $lista' \neq \emptyset$  do
6:   for  $p \in lista$  and  $p \ll q$  (en orden lexicográfico) do
7:     if  $mezcladet(M, p, q) \neq M$  then
8:        $M := mezcladet(M, p, q)$ 
9:     exit for
10:   end if
11: end for
12:  $lista :=$  Eliminar de  $lista$  los estados que no están en M
13:  $lista' :=$  Eliminar de  $lista'$  los estados que no están en M
14:  $q := first(lista')$ 
15: end while
16: Return  $M$ 

```

Figura 5.1: Algoritmo RPNI [6]

El algoritmo de RPNI (Regular Positive Negative Inference) consiste en generar un autómata de prefijos inicial (llamado 'árbol de prefijos de Moore' (APM)) a partir de las palabras positivas (D_+) y negativas (D_-); y realizar la mezcla (Figura 5.3) de la mayor cantidad de estados posibles hasta llegar al autómata mínimo que generalice al lenguaje que se está tratando de identificar.

Todo esto garantizando que los valores de salida de cada estado sean consistentes (que un estado no sea de aceptación y rechazo al mismo tiempo); y que no quede ningún tipo de no-determinismo (múltiples transiciones con un mismo símbolo) (Figura 5.2)

Un árbol de prefijos de Moore, como su nombre lo indica, es un árbol de prefijos construido a partir de las palabras positivas y negativas de entrenamiento. Cada estado representa un prefijo posible de las palabras de entrada; y al traversar el árbol, al llegar a ciertos estados, se habrá recorrido el

Algoritmo 13 *mezcladet*(M, p, q)

Require: $M = (Q, \Sigma, \{0, 1, ?\}^{\delta}, \Phi, q_0)$
Require: $p, q \in Q$, $p \ll q$ en orden lexicográfico

```

1:  $M' := M$ 
2:  $lista := \{(p, q)\}$ 
3: while  $lista \neq \emptyset$  do
4:    $(r, s) := first(lista)$ 
5:    $M_1 := mezcla(M', r, s)$ 
6:   if  $M_1 = M'$  then
7:     Return  $M$ 
8:   else
9:      $M' := M_1$ 
10:    for  $a \in \Sigma$  do
11:      if  $\delta(p, a)$  y  $\delta(q, a)$  están definidos then
12:         $lista := append(lista, (\delta(p, a), \delta(q, a)))$ 
13:      end if
14:    end for
15:  end if
16: end while
17: Return  $M'$ 

```

Figura 5.2: Función de mezcla determinista de estados [6]

camino de símbolos correspondiente a una palabra de entrenamiento. Si esta palabra es positiva, decimos que el estado es de aceptación. De lo contrario si la palabra es negativa, el estado será de rechazo.

Es posible que luego de que se genera el Árbol de prefijos de Moore (que es un autómata), cuando se hacen todos los recorridos posibles en el árbol, esto de como resultado algunas palabras que no pertenecen ni al conjunto de palabras positivas ni negativas. Cuando un camino en el árbol da como resultado a una de estas palabras, el estado final de ese camino se le denomina estado "indefinido". Durante la ejecución del algoritmo, si estos estados se mezclan con otro que sea de aceptación o de rechazo, el estado resultante será de aceptación o rechazo. Cuando el algoritmo ha terminado y ha producido el autómata final, es posible que en este queden algunos estados indefinidos. En ese caso, estos pasan a ser de rechazo.

5.2. Entrenamiento y evaluación

El entrenamiento consiste en tomar los datos pre-procesados en forma de conjuntos o muestras de palabras y escribirlos en un archivo de texto junto con su etiqueta (es decir si la palabra es positiva o negativa) como se observa en la Figura 5.4. Cuando se crea este archivo se le pasa su ruta al algoritmo de la técnica, al igual que la ruta del archivo en el que el algoritmo escribirá el resultado. Este resultado es el autómata final luego de realizar todas las mezclas intermedias (Ver

Algoritmo 12 *mezcla*(M, p, q)

Require: $M = (Q, \Sigma, \{0, 1, ?\}^{\delta}, \Phi, q_0)$ **Require:** $p, q \in Q$, $p \ll q$ en orden lexicográfico1: **if** $\Phi(p) \neq ? \wedge \Phi(q) \neq ? \wedge \Phi(p) \neq \Phi(q)$ **then**2: Return M 3: **end if**4: **if** $\Phi(p) = ?$ **then**5: $\Phi(p) := \Phi(q)$ 6: **end if**7: **for** $s \in \Sigma$ **do**8: **for** $r : \delta(r, s) = q$ **do**9: $\delta := \delta \cup \{\delta(r, s) = p\}$ 10: $\delta := \delta \setminus \{\delta(r, s) = q\}$ 11: **end for**12: **end for**13: **for** $s \in \Sigma$ **do**14: **for** $x : \delta(q, s) = x$ **do**15: $\delta := \delta \cup \{\delta(p, s) = x\}$ 16: $\delta := \delta \setminus \{\delta(q, s) = x\}$ 17: **end for**18: **end for**19: $Q := Q \setminus \{q\}$ 20: Return M

Figura 5.3: Función de mezcla de estados [6]

Figura 5.5).

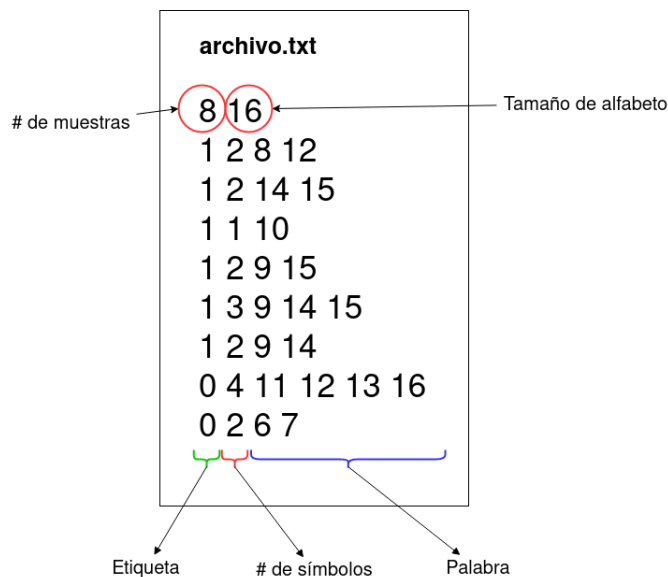


Figura 5.4: Formato de archivo de conjunto de muestras de entrenamiento. Producido con alfabeto de tamaño 16 y en codificación binaria con datos de Eosinófilos. Etiqueta '1': Paciente asociado a la palabra no curó (palabra positiva), '0': paciente sí curó (palabra negativa).

Cuadro 5.1: Figura 5.4 en formato de tabla.

Etiqueta	Palabra
1	(8, 12)
1	(14, 15)
1	(10)
1	(9, 15)
1	(9, 14, 15)
1	(9, 14)
0	(11, 12, 13, 16)
0	(6, 7)

Para hacer evaluaciones, se utilizan los autómatas producidos en la fase de entrenamiento y se le alimentan palabras nuevas (es decir palabras que la técnica nunca ha visto antes), y se observa si este las acepta o las rechaza. Estas palabras nuevas cuentan con la etiqueta del resultado real del tratamiento (falló o curó), y en la evaluación se comparan estas etiquetas con las producidas por el autómata, el cual no tiene acceso a las etiquetas de resultado real.

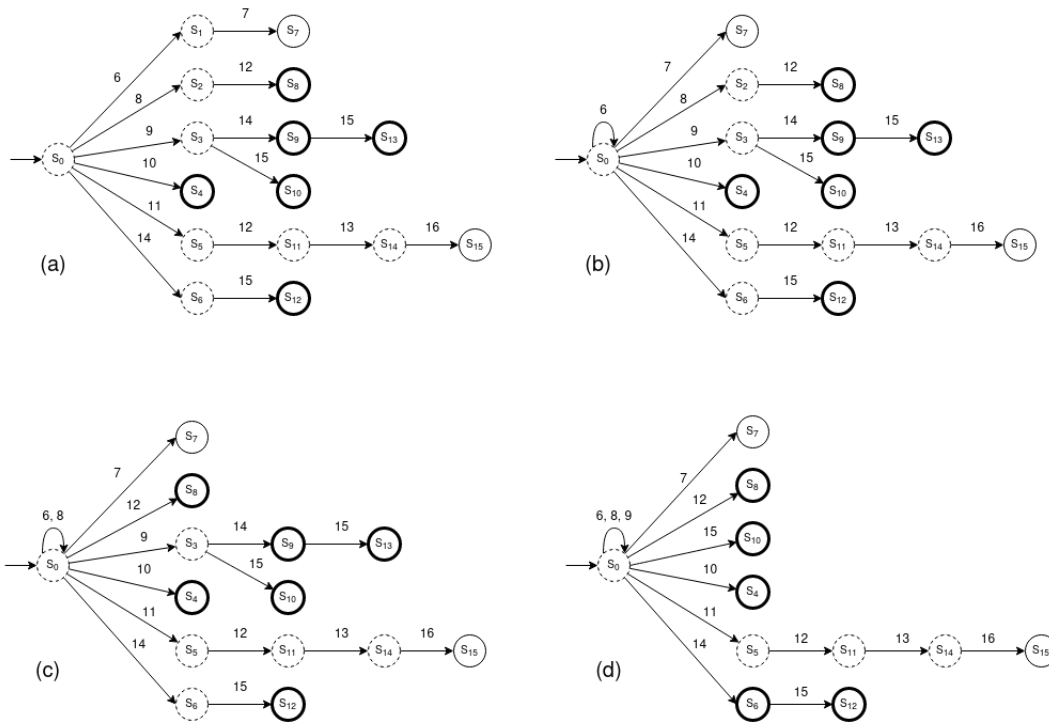


Figura 5.5: Ejemplo de proceso de mezcla de algoritmo RPN según el conjunto de muestras del Cuadro 5.1. (a): Autómata inicial. (b): Se mezcla el estado S_1 con S_0 , como ahora son el mismo estado, S_0 tiene una transición a sí mismo con '6', que era hacia S_1 originalmente. (c): Se mezcla estado S_2 con S_0 . (d): Se mezcla S_3 con S_0 lo cual crea no-determinismo (2 transiciones con el símbolo '14'), se elimina el no-determinismo. Línea gruesa: estado de aceptación, línea punteada: estado indefinido, línea delgada: estado de rechazo.

Pasando a la evaluación sistemática de la cual se presentan los resultados en la siguiente sección, se resalta que para poder utilizar todos los datos disponibles, y para evaluar de una forma más precisa la efectividad de los modelos producidos, se utiliza una **validación cruzada de 4 bloques** para cada tipo de célula (neutrófilos, eosinófilos, y monocitos).

Primeramente se utilizan los datos individuales de cada tipo de célula para construir alfabetos en codificación binaria y ternaria en diferentes tamaños, al igual que para pre-procesar los datos de pacientes o convertirlos en secuencias de símbolos o palabras. En el entrenamiento del algoritmo se parte de una semilla para generar las particiones aleatoriamente de los 4 bloques para utilizar como conjunto de evaluación. En cada iteración de la validación, el algoritmo RPNI utiliza las palabras de entrenamiento escritas en un archivo de texto para generar un autómata determinista, el cual luego se le alimentan las palabras de prueba para obtener el resultado de tratamiento de cada una. Se calcula la matriz de confusión y a partir de esta las métricas de desempeño Accuracy, Precision, Recall y F1-Score, y se guardan los resultados. Al final de la cuarta iteración, se tienen 4 valores para cada métrica. A estos se les calcula la media y desviación estándar y se compilan en una tabla como se ve en la sección de Resultados.

5.3. Resultados

Tamaño: cantidad de símbolos en el alfabeto generado.

Cuadro 5.2: Métricas de Desempeño RPNI - Neutrófilos - Codificación Binaria

Tamaño	Accuracy		Precision		Recall		F1-Score	
	mean	std	mean	std	mean	std	mean	std
12	0.854	0.172	0.875	0.250	0.917	0.167	0.867	0.163
16	0.875	0.144	0.854	0.172	1.000	0.000	0.914	0.102
20	0.750	0.300	0.708	0.344	0.833	0.333	0.725	0.320
24	0.750	0.300	0.708	0.344	0.833	0.333	0.725	0.320
28	0.650	0.252	0.708	0.344	0.708	0.344	0.642	0.263
32	0.650	0.252	0.708	0.344	0.708	0.344	0.642	0.263
40	0.650	0.252	0.708	0.344	0.708	0.344	0.642	0.263
60	0.550	0.100	0.583	0.500	0.458	0.417	0.417	0.289
80	0.600	0.283	0.500	0.577	0.250	0.319	0.325	0.395

Cuadro 5.3: Métricas de Desempeño RPNI - Neutrófilos - Codificación Ternaria

Tamaño	Accuracy		Precision		Recall		F1-Score	
	mean	std	mean	std	mean	std	mean	std
18	0.850	0.100	0.875	0.250	0.833	0.192	0.817	0.137
24	0.800	0.400	0.800	0.400	1.000	0.000	0.833	0.333
30	0.700	0.258	0.658	0.299	0.833	0.333	0.697	0.292
36	0.900	0.200	1.000	0.000	0.833	0.333	0.875	0.250
42	0.750	0.191	0.750	0.500	0.458	0.417	0.542	0.417
48	0.700	0.115	0.750	0.500	0.375	0.285	0.492	0.350
60	0.650	0.191	0.500	0.577	0.292	0.344	0.367	0.427
90	0.600	0.163	0.500	0.577	0.208	0.250	0.292	0.344
120	0.550	0.191	0.500	0.577	0.146	0.172	0.225	0.263

Cuadro 5.4: Métricas de Desempeño RPNI - Eosinófilos - Codificación Binaria

Tamaño	Accuracy		Precision		Recall		F1-Score	
	mean	std	mean	std	mean	std	mean	std
12	0.833	0.333	0.750	0.500	0.583	0.500	0.625	0.479
16	0.917	0.167	0.750	0.500	0.667	0.471	0.700	0.476
20	0.917	0.167	0.750	0.500	0.667	0.471	0.700	0.476
24	0.583	0.319	0.583	0.500	0.500	0.430	0.450	0.332
28	0.812	0.375	0.500	0.577	0.500	0.577	0.500	0.577
32	0.812	0.375	0.500	0.577	0.500	0.577	0.500	0.577
40	0.729	0.356	0.375	0.479	0.500	0.577	0.417	0.500
60	0.667	0.312	0.250	0.500	0.167	0.333	0.200	0.400
80	0.562	0.125	0.333	0.471	0.375	0.479	0.292	0.344

Cuadro 5.5: Métricas de Desempeño RPNI - Eosinófilos - Codificación Ternaria

Tamaño	Accuracy		Precision		Recall		F1-Score	
	mean	std	mean	std	mean	std	mean	std
18	0.688	0.473	0.500	0.577	0.417	0.500	0.450	0.526
24	0.750	0.500	0.500	0.577	0.500	0.577	0.500	0.577
30	0.750	0.500	0.500	0.577	0.500	0.577	0.500	0.577
36	0.688	0.473	0.438	0.515	0.500	0.577	0.464	0.539
42	0.667	0.312	0.250	0.500	0.167	0.333	0.200	0.400
48	0.667	0.312	0.250	0.500	0.167	0.333	0.200	0.400
60	0.562	0.375	0.250	0.500	0.125	0.250	0.167	0.333
90	0.500	0.354	0.250	0.500	0.125	0.250	0.167	0.333
120	0.438	0.375	0.250	0.500	0.125	0.250	0.167	0.333

Cuadro 5.6: Métricas de Desempeño RPNI - Monocitos - Codificación Binaria

Tamaño	Accuracy		Precision		Recall		F1-Score	
	mean	std	mean	std	mean	std	mean	std
12	0.854	0.172	0.750	0.500	0.667	0.471	0.700	0.476
16	1.000	0.000	0.750	0.500	0.750	0.500	0.750	0.500
20	0.917	0.167	0.750	0.500	0.750	0.500	0.750	0.500
24	0.917	0.167	0.625	0.479	0.750	0.500	0.667	0.471
28	0.938	0.125	0.875	0.250	1.000	0.000	0.917	0.167
32	0.938	0.125	1.000	0.000	0.917	0.167	0.950	0.100
40	0.938	0.125	1.000	0.000	0.917	0.167	0.950	0.100
60	0.700	0.200	0.833	0.333	0.708	0.344	0.667	0.236
80	0.688	0.253	0.667	0.471	0.625	0.479	0.617	0.433

Cuadro 5.7: Métricas de Desempeño RPNI - Monocitos - Codificación Ternaria

Tamaño	Accuracy		Precision		Recall		F1-Score	
	mean	std	mean	std	mean	std	mean	std
18	0.875	0.144	0.750	0.500	0.667	0.471	0.700	0.476
24	0.875	0.144	0.750	0.500	0.667	0.471	0.700	0.476
30	0.825	0.119	1.000	0.000	0.667	0.236	0.783	0.158
36	0.775	0.206	0.500	0.577	0.417	0.500	0.450	0.526
42	0.725	0.222	0.500	0.577	0.417	0.500	0.450	0.526
48	0.838	0.111	0.875	0.250	0.812	0.239	0.798	0.162
60	0.775	0.206	0.625	0.479	0.688	0.473	0.631	0.442
90	0.575	0.287	0.500	0.577	0.312	0.473	0.350	0.473
120	0.525	0.189	0.250	0.500	0.062	0.125	0.100	0.200

Algoritmo OIL

6.1. Descripción

Algoritmo 17 $OIL((D_+^{(1)}, D_-^{(1)}), \dots, (D_+^{(n)}, D_-^{(n)}))$

Require: $(D_+^{(1)}, D_-^{(1)}), \dots, (D_+^{(n)}, D_-^{(n)})$ son bloques no vacíos

- 1: $M = AM(D_+^{(1)}) // M = (Q, \Sigma, \{0, 1, ?\}, \delta, \Phi, I)$
- 2: $D_- := D_-^{(1)}$
- 3: $E :=$ Una enumeración de los estados de $AM(D_+^{(1)})$ usando los enteros entre 1 y N_1 , donde N_1 es el número de estados de $AM(D_+^{(1)})$
- 4: $M := mezclaNfas(E, M, D_-)$
- 5: $i := 2$
- 6: **while** $i \leq n$ **do**
- 7: $D_- := D_- \cup D_-^{(i)}$
- 8: $M' := AM(D_+^{(i)}) // M' = (Q', \Sigma, \{0, 1, ?\}, \delta', \Phi', I')$
- 9: $N_i := |Q'|$
- 10: **if** $\neg consistente(M, D_-^{(i)})$ or $\neg consistente(M, D_+^{(i)})$ **then**
- 11: **if** $\neg consistente(M, D_-^{(i)})$ **then**
- 12: $i := 1$
- 13: $M := AM(D_+^{(1)})$
- 14: $E :=$ Una enumeración de los estados de $AM(D_+^{(1)})$ usando los enteros entre 1 y N_1 , donde N_1 es el número de estados de $AM(D_+^{(1)})$
- 15: **else**
- 16: $N_i := |Q'|$
- 17: Eliminar de M' las muestras consistentes en M
- 18: $M := M \cup M' // M = (Q \cup Q', \Sigma, \{0, 1, ?\}, \delta \cup \delta', \Phi \cup \Phi', I \cup I')$
- 19: Extender E asignando identificador aleatorio a los estados de Q' con valores enteros entre $(\sum_{j=1}^{i-1} N_j) + 1$ y $\sum_{j=1}^i N_j$
- 20: **end if**
- 21: $M := mezclaNfas(E, M, D_-)$
- 22: **end if**
- 23: $i := i + 1$
- 24: **end while**
- 25: Return M

Figura 6.1: Algoritmo OIL [6]

Algoritmo 18 *mezclaNfas*(E, D_+, D_-)

Require: E es una enumeración de Σ^*

Require: D_+ y D_- son una muestra característica

- 1: $M = AM(D_+) // M = (Q, \Sigma, \{0, 1, ?\}\delta, \Phi, I)$
- 2: $lista := \{x : x \in Q\} //$ Ordenanda ascendentemente de acuerdo a la enumeración
- 3: **for** $j \in lista$ **do**
- 4: $mezclado := falso$
- 5: **for** $i \in lista$ and $i \ll j$ and $\neg mezclado$ **do**
- 6: $M' := mezcla(M, i, j) // M = (Q', \Sigma, \{0, 1, ?\}\delta', \Phi', I')$
- 7: **if** $M' \neq M$ and $j \in I$ **then**
- 8: $I' := I' \cup \{i\}$
- 9: **end if**
- 10: **if** *consistente*(M', D_-) **then**
- 11: $M := M'$
- 12: $lista := lista \setminus \{x : x \in lista \text{ and } x \notin Q\}$
- 13: $mezclado = verdadero$
- 14: **end if**
- 15: **end for**
- 16: **end for**
- 17: Return M

Figura 6.2: Función *mezclaNfas* [6]

Este algoritmo [6] fue implementado en el lenguaje de programación C++. Se implementó un wrapper en Python para utilizarlo desde Jupyter en los experimentos.

A diferencia de RPNI, este algoritmo utiliza autómatas no-deterministas en lugar de deterministas. En vez de tener dos conjuntos de palabras positivas y negativas, OIL hace uso de particiones de los conjuntos de palabras para realizar la mezcla de estados. Cuando se intenta realizar una mezcla y esta causa inconsistencias (mezclar un estado de aceptación con uno de rechazo), el algoritmo genera un autómata extra con un estado de inicio adicional desconectado del resto del autómata general, aprovechando la cualidad de no-determinismo y acomodando el patrón de observación que quiere ser mezclado (Figura 6.3).

6.2. Entrenamiento y evaluación

Para el entrenamiento de este algoritmo, al igual que con el RPNI, consiste en tomar los conjuntos de palabras ya procesados y escribirlos en un archivo de texto. El nombre de este archivo se ingresa como un parámetro para el algoritmo RPNI, al igual que el nombre del archivo en el que se desea guardar el resultado final, después de que se han realizado todas las mezclas por el algoritmo.

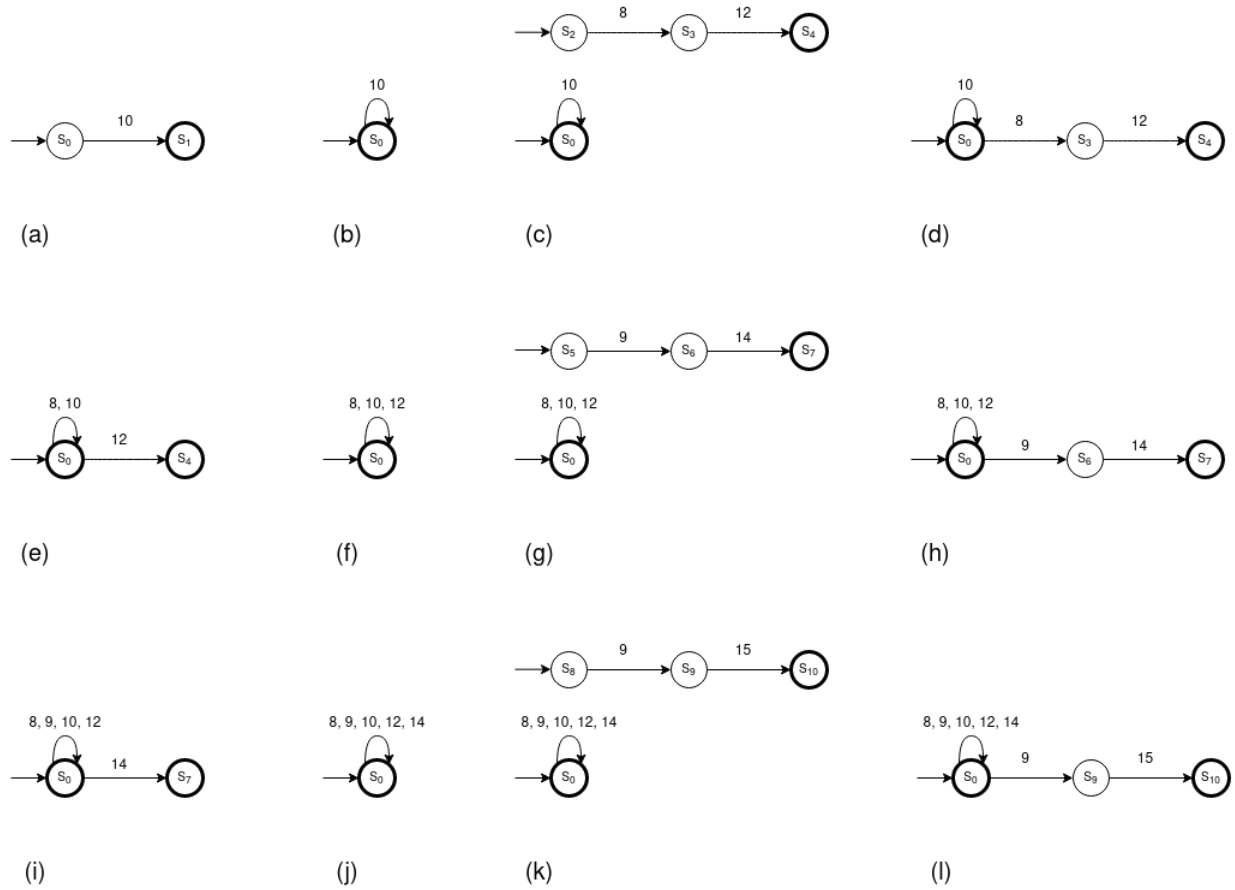


Figura 6.3: Ejemplo de autómatas intermedios en el funcionamiento del algoritmo OIL según el conjunto de muestras del Cuadro 5.1. (a): Autómata inicial. Se adiciona la muestra (10). (b): Se mezcla el estado S_1 con S_0 . (c): Se adiciona la muestra (8, 12). (d): Se mezcla el estado S_2 con S_0 . (e): Se mezcla el estado S_3 con S_0 . (f): Se mezcla el estado S_4 con S_0 . (g): Se adiciona la muestra (9, 14). (h): Se mezcla el estado S_5 con S_0 . (i): Se mezcla el estado S_6 con S_0 . (j): Se mezcla el estado S_6 con S_0 . (k): Se adiciona la muestra (9, 15). (l): Se mezcla el estado S_8 con S_0 .

Al igual que en el algoritmo RPNI, se hace uso de una validación cruzada de 4 bloques para realizar el entrenamiento y evaluación de la técnica.

6.3. Resultados

Tamaño: cantidad de símbolos en el alfabeto generado.

Cuadro 6.1: Métricas de Desempeño OIL - Neutrófilos - Codificación Binaria

Tamaño	Accuracy		Precision		Recall		F1-Score	
	mean	std	mean	std	mean	std	mean	std
12	0.542	0.220	0.250	0.500	0.083	0.167	0.125	0.25
16	0.412	0.118	0.000	0.000	0.000	0.000	0.000	0.000
20	0.500	0.200	0.250	0.500	0.062	0.125	0.100	0.200
24	0.500	0.200	0.250	0.500	0.062	0.125	0.100	0.200
28	0.450	0.252	0.000	0.000	0.000	0.000	0.000	0.000
32	0.450	0.252	0.000	0.000	0.000	0.000	0.000	0.000
40	0.450	0.252	0.000	0.000	0.000	0.000	0.000	0.000
60	0.450	0.252	0.000	0.000	0.000	0.000	0.000	0.000
80	0.450	0.252	0.000	0.000	0.000	0.000	0.000	0.000

Cuadro 6.2: Métricas de Desempeño OIL - Neutrófilos - Codificación Ternaria

Tamaño	Accuracy		Precision		Recall		F1-Score	
	mean	std	mean	std	mean	std	mean	std
18	0.600	0.283	0.500	0.577	0.375	0.479	0.417	0.500
24	0.550	0.191	0.500	0.577	0.146	0.172	0.225	0.263
30	0.650	0.252	0.750	0.500	0.458	0.417	0.542	0.417
36	0.550	0.300	0.500	0.577	0.312	0.473	0.350	0.473
42	0.450	0.252	0.000	0.000	0.000	0.000	0.000	0.000
48	0.450	0.252	0.000	0.000	0.000	0.000	0.000	0.000
60	0.450	0.252	0.000	0.000	0.000	0.000	0.000	0.000
90	0.450	0.252	0.000	0.000	0.000	0.000	0.000	0.000
120	0.450	0.252	0.000	0.000	0.000	0.000	0.000	0.000

Cuadro 6.3: Métricas de Desempeño OIL - Eosinófilos - Codificación Binaria

Tamaño	Accuracy		Precision		Recall		F1-Score	
	mean	std	mean	std	mean	std	mean	std
12	0.458	0.417	0.000	0.000	0.000	0.000	0.000	0.000
16	0.583	0.319	0.500	0.577	0.167	0.192	0.250	0.289
20	0.417	0.500	0.000	0.000	0.000	0.000	0.000	0.000
24	0.417	0.500	0.000	0.000	0.000	0.000	0.000	0.000
28	0.542	0.363	0.000	0.000	0.000	0.000	0.000	0.000
32	0.542	0.363	0.000	0.000	0.000	0.000	0.000	0.000
40	0.542	0.363	0.000	0.000	0.000	0.000	0.000	0.000
60	0.542	0.363	0.000	0.000	0.000	0.000	0.000	0.000
80	0.500	0.354	0.000	0.000	0.000	0.000	0.000	0.000

Cuadro 6.4: Métricas de Desempeño OIL - Eosinófilos - Codificación Ternaria

Tamaño	Accuracy		Precision		Recall		F1-Score	
	mean	std	mean	std	mean	std	mean	std
18	0.479	0.443	0.000	0.000	0.000	0.000	0.000	0.000
24	0.479	0.443	0.000	0.000	0.000	0.000	0.000	0.000
30	0.479	0.443	0.000	0.000	0.000	0.000	0.000	0.000
36	0.542	0.417	0.250	0.500	0.083	0.167	0.125	0.250
42	0.542	0.363	0.000	0.000	0.000	0.000	0.000	0.000
48	0.542	0.363	0.000	0.000	0.000	0.000	0.000	0.000
60	0.500	0.354	0.000	0.000	0.000	0.000	0.000	0.000
90	0.500	0.354	0.000	0.000	0.000	0.000	0.000	0.000
120	0.500	0.354	0.000	0.000	0.000	0.000	0.000	0.000

Cuadro 6.5: Métricas de Desempeño OIL - Monocitos - Codificación Binaria

Tamaño	Accuracy		Precision		Recall		F1-Score	
	mean	std	mean	std	mean	std	mean	std
12	0.667	0.312	0.500	0.577	0.417	0.500	0.450	0.526
16	0.500	0.430	0.000	0.000	0.000	0.000	0.000	0.000
20	0.500	0.430	0.000	0.000	0.000	0.000	0.000	0.000
24	0.479	0.443	0.000	0.000	0.000	0.000	0.000	0.000
28	0.479	0.267	0.000	0.000	0.000	0.000	0.000	0.000
32	0.479	0.267	0.000	0.000	0.000	0.000	0.000	0.000
40	0.479	0.267	0.000	0.000	0.000	0.000	0.000	0.000
60	0.475	0.250	0.000	0.000	0.000	0.000	0.000	0.000
80	0.413	0.272	0.000	0.000	0.000	0.000	0.000	0.000

Cuadro 6.6: Métricas de Desempeño OIL - Monocitos - Codificación Ternaria

Tamaño	Accuracy		Precision		Recall		F1-Score	
	mean	std	mean	std	mean	std	mean	std
18	0.688	0.315	0.500	0.577	0.417	0.500	0.450	0.526
24	0.688	0.315	0.500	0.577	0.417	0.500	0.450	0.526
30	0.588	0.236	0.500	0.577	0.250	0.289	0.333	0.385
36	0.675	0.150	0.500	0.577	0.292	0.344	0.367	0.427
42	0.575	0.171	0.500	0.577	0.146	0.172	0.225	0.263
48	0.475	0.250	0.000	0.000	0.000	0.000	0.000	0.000
60	0.475	0.250	0.000	0.000	0.000	0.000	0.000	0.000
90	0.475	0.250	0.000	0.000	0.000	0.000	0.000	0.000
120	0.475	0.250	0.000	0.000	0.000	0.000	0.000	0.000

Algoritmo Integrador

7.1. Descripción

El propósito de este algoritmo es combinar el poder de predicción de los mejores modelos producidos por los datos de cada tipo célula (Cuadro 7.1). Es decir que para el diagnostico o pronostico de cada paciente se utilizarían (en el mejor caso) 3 modelos. Uno generado a partir de eosinófilos, otro de neutrófilos y otro de monocitos, cada uno usando la mejor técnica entre OIL y RPNI, la mejor codificación, y el mejor tamaño de alfabeto. La utilización de cada uno de estos modelos depende de la disponibilidad del dato de célula correspondiente. Idealmente, este es el algoritmo final que se utilizaría en un ambiente de producción para realizar predicciones de nuevos pacientes.

El funcionamiento general del algoritmo se ve ilustrado en la Figura 7.1. Cuando se obtienen los datos de la primera cita de un nuevo paciente de algún tipo de célula (neutrófilos, eosinófilos o monocitos), estos son pre-procesados y transformados cada uno en una palabra. Luego cada palabra es alimentada al modelo correspondiente: si la palabra proviene de neutrófilos, se usa con el modelo ya entrenado de los neutrófilos, y así sucesivamente con las otras células. Por cada palabra, los modelos retornan su predicción a forma de voto con respecto al resultado de tratamiento del paciente: '0' si el paciente cura, y '1' si el paciente no cura. El voto con mayor popularidad es el ganador, y si llegase a existir un empate (solo se usan 2 modelos que votan de forma contraria), se encoje la predicción que provenga del modelo con mejor F1-score, la cual es una métrica que se obtiene de las fase de entrenamiento de los algoritmos expuesta anteriormente.

Se usan los conjuntos de datos completos de células para entrenar a cada modelo, a excepción de los datos de 2 pacientes que son usados para realizar una demostración corta del funcionamiento del algoritmo integrador. Se usan datos completos porque que se corroboró el rendimiento de cada modelo con la validación cruzada usando datos parciales.

Cuadro 7.1: Mejor modelo para cada tipo de célula.

Célula	Algoritmo	Codificación	Tamaño Alfabeto	Accuracy	Precision	Recall	F1-Score
Neutrófilos	RPNI	Binaria	16	0.875	0.854	1.000	0.914
Eosinófilos	RPNI	Binaria	16	0.917	0.750	0.667	0.700
Monocitos	RPNI	Binaria	32	0.938	1.000	0.917	0.950

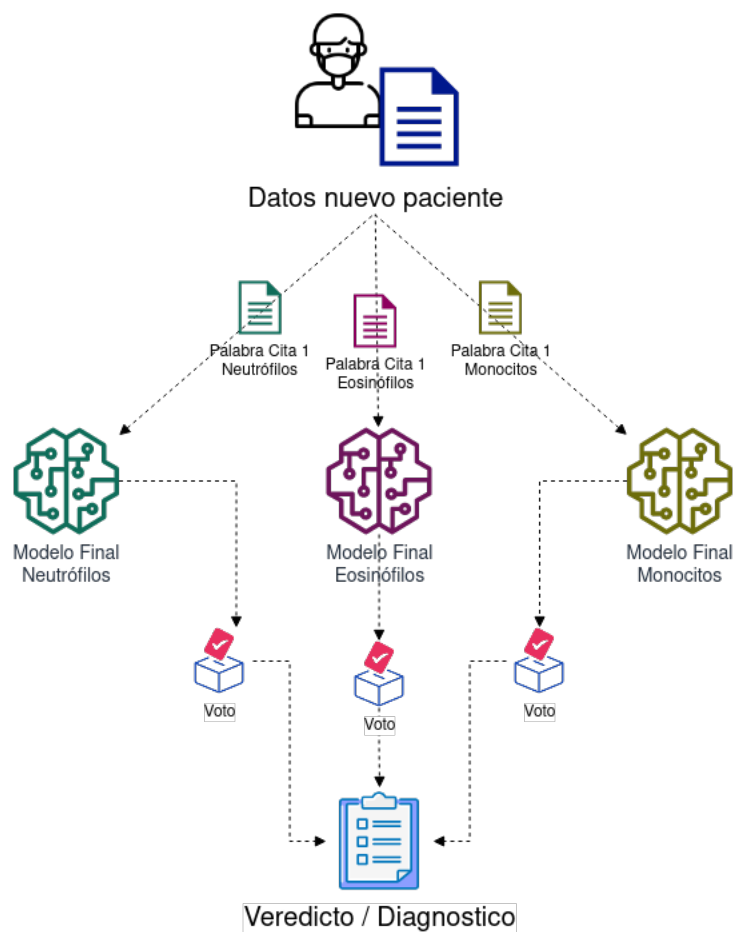


Figura 7.1: Diagrama general sobre funcionamiento del algoritmo integrador

Cuadro 7.2: Prueba de predicción algoritmo integrador con paciente X2052

Paciente = X2052		Veredicto = 1
Célula	Palabra	Votación modelo
Neutrófilos	(11)	1
Eosinófilos	No tiene dato de célula	No participa en votos
Monocitos	(19, 26, 28, 29)	1

Cuadro 7.3: Prueba de predicción algoritmo integrador con paciente X2071

Paciente = X2071		Veredicto = 0
Célula	Palabra	Votación modelo
Neutrófilos	(15)	0
Eosinófilos	No tiene dato de célula	No participa en votos
Monocitos	(30)	0

Desafortunadamente, como se usaron todos los datos para el entrenamiento de este algoritmo, no se cuentan con más datos para probar el desempeño del mismo; por lo cual no se pueden proveer resultados de desempeño, ni tampoco se puede saber si presenta una mejora en el desempeño con respecto a los tres modelos por separado. No obstante, si se puede demostrar a forma de prueba corta cómo funciona éste, utilizando todos los datos de entrenamiento disponibles pero excluyendo los datos de 2 pacientes (X2052 que falla, y X2071 que cura) con los cuales se evidencia el funcionamiento. Estas pruebas se exponen en los Cuadros 7.2 y 7.3, en donde se observa como por cada palabra disponible de cada paciente, el modelo correspondiente retorna una predicción, y a partir de estos votos se llega a un veredicto final sobre el desenlace del tratamiento para ese paciente.

Resultados y Análisis

En las tablas de resultados de RPNI de neutrófilos 5.2 y 5.3 se puede observar cómo la métrica de F1-score va disminuyendo conforme se aumenta el tamaño del alfabeto. Lo mismo sucede en los resultados de OIL 6.2 (neutrófilos) y 6.6 (monocitos). La disminución del desempeño con mayores tamaños de alfabeto parece estar presente en la mayoría de resultados. No obstante hay casos en los que este comportamiento es contrario, como en la tabla de monocitos 5.6, en donde el F1-score aumenta conforme aumenta el tamaño del alfabeto, aunque este luego empieza a disminuir de nuevo cuando sobrepasa los 40 símbolos.

Cuando se utilizan más de 40 genes para crear un alfabeto ya sea binario o ternario, se terminan creando palabras muy largas, mientras que el número de muestras se mantiene igual. Es como un tipo de maldición de la dimensionalidad: se aumenta el campo de atributos (cantidad de genes posibles), y para tener datos que muestran ejemplos de todas las variaciones posibles se necesita una cantidad gigantesca de datos.

Los rangos de tamaños de alfabeto (cantidad de símbolos) probados fueron [12, 80], para codificación binaria y [18, 120], para la codificación ternaria. Se tomó cada tamaño, se entrenaron y evaluaron las técnicas y se observaron los resultados de desempeño. Los 16 tamaños de alfabetos presentados en este documento (9 para codificación binaria y 9 para codificación ternaria, para cada tipo de célula) son los más representativos en cuanto a la variedad de métricas de desempeño. Los no incluidos mostraron desempeños muy similares a los seleccionados. Cabe mencionar que a partir de aproximadamente 60 o 70 símbolos, los resultados de desempeño producidos por los alfabetos empiezan a empeorar y ser similares a cantidades anteriores. Es decir más allá de estos tamaños, no se alcanza una mejoría substancial en el desempeño de los algoritmos.

Por otra parte, en cuanto a los resultados según el tipo de codificación se puede observar que la que presenta los mejores resultados en el algoritmo RPNI de forma dominante es la binaria. En las tablas de monocitos 5.6 y 5.7 de RPNI se observa como la codificación ternaria es igual de buena que la binaria en cuanto a F1-score cuando los tamaño de los alfabetos son pequeños, pero cuando estos sobrepasan los 25 símbolos aproximadamente, la binaria empieza a tener mejores métricas de desempeño. No obstante, para el algoritmo OIL, el tipo de codificación que dio mejores resultados en cuanto a F1-score promedio fue la ternaria: en la tabla de métricas para neutrófilos en codificación binaria 6.1 se observa como solamente se obtuvo un F1-score de 0.125 con un tamaño de alfabeto de 12 genes, mientras que en la tabla de codificación ternaria 6.2 se obtienen multiples F1-scores por

encima de 0.125. En las tablas 6.5 y 6.6 de los resultados de OIL se presenta un comportamiento similar.

Como OIL es un algoritmo más complejo que RPNI, y la cantidad de datos con la que estamos trabajando es pequeña, es normal obtener resultados poco precisos. En casi todos los casos, el autómata resultante producido por el algoritmo no tiene la suficiente información o variabilidad de datos para generalizar el lenguaje. A consecuencia de esto, se termina rechazando a la mayoría o todas las palabras de prueba sean positivas o negativas. Si se tuvieran conjuntos de datos bastante más grandes, se podría apreciar el poder de predicción de esta técnica. En algunas instancias cuando se tienen tamaños de alfabetos grandes, como en la tabla 6.6, se observa que las métricas de Recall y Precision son 0, mientras que Accuracy es mayor que 0. Esto sucede debido a que de los pocos aciertos que tuvo el algoritmo, todos estos fueron de muestras de la clase negativa, es decir muestras de pacientes cuyo tratamiento sí fue exitoso.

Al observar los resultados de tanto de OIL como RPNI, se observa que de los 3 tipos de células usados, el que tuvo menor desempeño en general fue eosinófilos. Esto era de esperarse, ya que como se observó en la cantidad de muestras disponibles para realizar los experimentos (Tabla 4.4), las muestras minoritarias eran las de eosinófilos: solo se tienen 16, mientras que los otros dos tipos de células se tienen 20 y 19 muestras. De haber tenido una cantidad similar, los resultados para eosinófilos en ambas técnicas pudieron haber sido mejores

Con base en los resultados que tuvieron mayor desempeño entre las diferentes técnicas y parámetros, se pudieron obtener la lista de genes que parecen estar correlacionados con una buena predicción de tratamiento, para cada uno de los tipos de células. En la Tabla 8.1 se recopilan estos genes por tipo de célula, cuyos modelos hacen uso de **RPNI** y codificación **binaria** con tamaños de alfabetos entre 16 y 32 (Tabla 7.1). Con base en estos tamaños de alfabeto, y haciendo uso de las listas de genes usados para la creación de todos los alfabetos usados en todos los experimentos (Tablas 4.5, 4.6 y 4.7), se obtienen los genes mostrados en la Tabla 8.1: La columna de Neutrófilos son los genes de neutrófilos seleccionados cuando $n = 8$ (Lo cual resulta en un alfabeto de tamaño 16), la columna de Eosinófilos son los genes seleccionados cuando $n = 8$ (alfabeto de tamaño 16), y la columna de Monocitos son los genes seleccionados cuando $n = 16$ (alfabeto de tamaño 32).

Cuadro 8.1: Genes de cada tipo de célula que produjeron los mejores resultados de desempeño para la predicción de la falla del tratamiento de la Leishmaniasis Cutánea.

Neutrófilos	Eosinófilos	Monocitos
ENSG00000003137	ENSG00000115155	ENSG00000039560
ENSG00000115919	ENSG00000136235	ENSG00000108700
ENSG00000146205	ENSG00000137571	ENSG00000108821
ENSG00000165966	ENSG00000162669	ENSG00000128274
ENSG00000175018	ENSG00000163046	ENSG00000129990
ENSG00000185736	ENSG00000187569	ENSG00000131203
ENSG00000196526	ENSG00000198178	ENSG00000142910
ENSG00000248405	ENSG00000277150	ENSG00000162669
		ENSG00000163046
		ENSG00000164692
		ENSG00000165966
		ENSG00000167768
		ENSG00000168542
		ENSG00000196526
		ENSG00000197057
		ENSG00000203812

Conclusiones

Se logró predecir la efectividad del tratamiento de la leishmaniasis cutánea haciendo uso de técnicas de inferencia gramatical y datos génicos de pacientes, el cual es el objetivo general del proyecto. A su vez, se diseñó la representación gramatical de los datos génicos que fueron pre-procesados, y se diseñaron y ejecutaron diversos experimentos para entrenar y evaluar el desempeño de las 2 técnicas implementadas y usadas; cumpliendo de esta forma los objetivos específicos de la investigación.

RPNI demuestra mejores resultados para la predicción de la falla de tratamiento que el algoritmo OIL, logrando una tasa muy competitiva de más del 90 % de accuracy en múltiples experimentos. Luego, a pesar de que la cantidad de datos disponibles fue muy reducida, los resultados obtenidos con ambos algoritmos son muy prometedores. Si se pudiese acceder a más muestras de pacientes se podría llegar a unos resultados más confiables y representativos, tanto para cada modelo por separado como para el algoritmo integrador. Aunque este último no pudo ser evaluado en profundidad, es claro que muestra una manera de sacarle partido a los mejores modelos previamente entrenados.

Para trabajos futuros, se necesita hacer un análisis más minucioso con respecto a las decisiones iniciales como en el filtrado de datos: aspectos como el valor P ajustado a considerar como significativo se pudo realizar con más detalle, ya que se podría estar incluyendo más genes de los que son adecuados luego de realizar el pre-procesamiento, y esto podría impactar los resultados de los algoritmos.

En cuanto al impacto frente a las investigaciones sobre la enfermedad, es muy valioso proveer los genes particulares de cada tipo de célula que parecen estar correlacionados con buenos desempeños de predicción de tratamiento, los cuales quedaron documentados en la Tabla 8.1.

Finalmente, se concluye que la inferencia gramatical es una técnica que debe ser tenida en cuenta en este tipo de problemas. En este proyecto la técnica demostró que se le puede sacar provecho a la estructura de los datos que se tenga para trabajar, llegando a resultados de desempeño respetables, incluso en condiciones en las que los conjuntos de datos son tan pequeños.

Bibliografía

- [1] M. Salamanca, B. Londoño, L. Urquijo, J. López, V. Álvarez y G. Rey, “Guía para la Atención clínica Integral del Paciente con Leishmaniasis,” Organización Panamericana de la Salud, 2010. dirección: <https://www.minsalud.gov.co/Documents/Salud%20P%C3%BAblica/01a%20invernal/Clinica%20Leishmaniasis.pdf>.
- [2] D. A. Vargas, M. D. Prieto, A. J. Martínez-Valencia, A. Cossio, K. E. V. Burgess, R. J. Burchmore y M. A. Gómez, “Pharmacometabolomics of Meglumine Antimoniate in Patients With Cutaneous Leishmaniasis,” *Frontiers in Pharmacology*, vol. 10, jun. de 2019, ISSN: 1663-9812. DOI: [10.3389/fphar.2019.00657](https://doi.org/10.3389/fphar.2019.00657). dirección: <https://www.frontiersin.org/article/10.3389/fphar.2019.00657/full>.
- [3] G. S. Nancy. “Leishmaniasis.” Accedido: 2020-10-04, dirección: <http://www.cideim.org.co/cideim/en/our-research/researchareas/leishmaniasis.html>.
- [4] S. Duran, C. mendez, L. Mejia, C. Ovalle, C. Arenas, Y. Camargo, L. Rubiano y A. Cossio, “Lineamientos para la Atención Clínica Integral de Leishmaniasis en Colombia,” Ministerio de Salud y Protección Social, abr. de 2018. dirección: <https://www.minsalud.gov.co/sites/rid/Lists/BibliotecaDigital/RIDE/VS/PP/PAI/Lineamientos-leishmaniasis.pdf>.
- [5] Y. G. Chen, A. T. Satpathy y H. Y. Chang, “Gene regulation in the immune system by long noncoding RNAs,” *Nature Immunology*, vol. 18, n.º 9, págs. 962-972, sep. de 2017, ISSN: 1529-2916. DOI: [10.1038/ni.3771](https://doi.org/10.1038/ni.3771). dirección: <https://doi.org/10.1038/ni.3771>.
- [6] G. I. Á. Vargas, “Estudio de la mezcla de estados determinista y no determinista en el diseño de algoritmos para inferencia gramatical de lenguajes regulares,” Tesis doct., Universitat Politècnica de València, dic. de 2007. DOI: [10.4995/Thesis/10251/1957](https://doi.org/10.4995/Thesis/10251/1957). dirección: <https://riunet.upv.es/handle/10251/1957>.
- [7] A. Stevenson y J. R. Cordy, “A survey of grammatical inference in software engineering,” *Science of Computer Programming*, vol. 96, págs. 444-459, 2014, Selected Papers from the Fifth International Conference on Software Language Engineering (SLE 2012), ISSN: 0167-6423. DOI: <https://doi.org/10.1016/j.scico.2014.05.008>. dirección: <http://www.sciencedirect.com/science/article/pii/S0167642314002469>.
- [8] OMS. (mar. de 2020). “Leishmaniasis.” Accedido: 2020-10-04, dirección: <https://www.who.int/es/news-room/fact-sheets/detail/leishmaniasis>.
- [9] L. F. Oliveira, A. O. Schubach, M. M. Martins, S. L. Passos, R. V. Oliveira, M. C. Marzochi y C. A. Andrade, “Systematic review of the adverse effects of cutaneous leishmaniasis treatment in the New World,” *Acta Tropica*, vol. 118, n.º 2, págs. 87-96, 2011, ISSN: 0001-706X. DOI: <https://doi.org/10.1016/j.actatropica.2011.02.007>. dirección: <http://www.sciencedirect.com/science/article/pii/S0001706X11000325>.

- [10] D. Cabrera. (oct. de 2019). “Leishmaniasis, con más de 12.000 casos en Colombia | RCN Radio.” Accedido: 2020-10-26, dirección: <https://www.rcnradio.com/salud/los-municipios-de-colombia-con-mas-riesgo-de-leishmaniasis>.
- [11] E. Torres-Guerrero y R. Arenas, “Leishmaniasis. Alternativas terapéuticas actuales,” *Dermatol Rev Mex*, vol. 62, págs. 400-409, 5 2018. dirección: <https://www.medigraphic.com/pdfs/derrevmex/rmd-2018/rmd185e.pdf>.
- [12] (Ene. de 2015). “What is gene expression? | Facts | yourgenome.org.” Accedido: 2020-11-01, dirección: <https://www.yourgenome.org/facts/what-is-gene-expression>.
- [13] G.-T. Cabetas, “Inmunoparasitología de Leishmania,” Facultad de Farmacia Universidad Complutense de Madrid, 2017, Accedido: 2020-10-30. dirección: <http://147.96.70.122/Web/TFG/TFG/Memoria/MACARENA%20GARCIA-TREVIJANO%20CABETAS.pdf>.
- [14] Y. Chu y D. R. Corey, “RNA Sequencing: Platform Selection, Experimental Design, and Data Interpretation,” *Nucleic Acid Therapeutics*, vol. 22, 4 ago. de 2012, ISSN: 2159-3337. DOI: [10.1089/nat.2012.0367](https://doi.org/10.1089/nat.2012.0367).
- [15] C. A. Maher, C. Kumar-Sinha, X. Cao, S. Kalyana-Sundaram, B. Han, X. Jing, L. Sam, T. Barrette, N. Palanisamy y A. M. Chinnaiyan, “Transcriptome sequencing to detect gene fusions in cancer,” *Nature*, vol. 458, 7234 mar. de 2009, ISSN: 0028-0836. DOI: [10.1038/nature07638](https://doi.org/10.1038/nature07638).
- [16] K. A. Shastry y H. A. Sanjay, “Machine Learning for Bioinformatics,” en *Statistical Modeling and Machine Learning Principles for Bioinformatics Techniques, Tools, and Applications*, K. G. Srinivasa, G. M. Siddesh y S. R. Manisekhar, eds. Singapore: Springer Singapore, 2020, págs. 25-39. DOI: [10.1007/978-981-15-2445-5_3](https://doi.org/10.1007/978-981-15-2445-5_3). dirección: https://doi.org/10.1007/978-981-15-2445-5_3.
- [17] R. Aguilar, L. Alonso, V. López y M. N. Moreno, “Incremental discovery of sequential patterns for grammatical inference,” 2005, págs. 59-67. dirección: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84877599894&partnerID=40&md5=4f2840983be1dbd081c619fe3fd259ae>.
- [18] C. Cook, “DNA region grammatical inference,” 2017, págs. 1577-1583. DOI: [10.1109/BIBM.2016.7822755](https://doi.org/10.1109/BIBM.2016.7822755). dirección: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85013297590&doi=10.1109%5C%2fBIBM.2016.7822755&partnerID=40&md5=f2f1309f20800b097e1f9c1da46f076a>.
- [19] (Nov. de 2019). “Predicting treatment outcome for leishmaniasis – ScienceDaily.” Accedido: 2020-11-01, dirección: <https://www.sciencedaily.com/releases/2019/11/191120141843.htm>.
- [20] C. F. Amorim, F. O. Novais, B. T. Nguyen, A. M. Misic, L. P. Carvalho, E. M. Carvalho, D. P. Beiting y P. Scott, “Variable gene expression and parasite load predict treatment outcome in cutaneous leishmaniasis,” *Science Translational Medicine*, vol. 11, n.º 519, 2019, ISSN: 1946-6234. DOI: [10.1126/scitranslmed.aax4204](https://doi.org/10.1126/scitranslmed.aax4204). eprint: <https://stm.sciencemag.org/content/11/519/eaax4204.full.pdf>. dirección: <https://stm.sciencemag.org/content/11/519/eaax4204>.

- [21] “Centro Internacional de Entrenamiento e Investigaciones Médicas (CIDEIM).” Accedido: 2020-10-26, dirección: <http://www.cideim.org.co/cideim/es.html>.