**Desarrollos hacia la elaboración de un sistema de transcripción**

**Monografía de grado**

Iván Francisco Montalvo Duque

Julio 2019

Universidad Nacional de Colombia

Departamento de Lingüística

Licenciatura en Español y Filología Clásica

Bogotá D.C.

**Agradecimientos**

Este trabajo no se podría realizar sin el incalculable trabajo de Santiago Buitrago, amigo de toda la vida y colega que dio su tiempo y habilidades para la programación de Perséfone. Espero que este texto le sea de utilidad en algún momento.

Gracias a mi madre, por su inconmensurable compañía y amor, además por colaboración con la interpretación de las tablas. También gracias a Elaine Almquist, por corregir las tablas y brindarme su desinteresado amor. A mi abuela por su cariño y comprensión. A mi padre, por levantarme en momentos oscuros.

Asimismo, agradezco a mis tutores Juan Felipe González Calderón y Flavio Augusto Prieto Ortiz, por sus conocimientos y guía a través de aguas tormentosas.

Gracias a la Universidad Nacional de Colombia, por ser un verdadero espacio de conocimiento.

Por último, agradezco a todos mis amigos por sus consejos y críticas.

**Tabla de Contenidos**

Capítulo 1 Introducción…………………………………………………………………….....................8

Capítulo 2 Marco Teórico……………………………………………………………………..................11

* 1. El “*Physices Tractatus*” y la Colonia......................................................................................12
     1. Contexto Histórico del Manuscrito...................................................................................12
     2. Sobre el Autor del Manuscrito..........................................................................................13
     3. El manuscrito RM-149......................................................................................................13
  2. OCR y Redes Neuronales.......................................................................................................14
     1. El OCR y su historia..........................................................................................................14
     2. Aprendizaje de Máquina y Redes Neuronales: Contexto Histórico y Aplicaciones.........15
  3. Las Nuevas tecnologías y la Filología Clásica.......................................................................16
     1. La Necesidad de las Nuevas Tecnologías en la Filología Clásica.....................................16
     2. Avances en el Contexto Filológico....................................................................................20

Capítulo 3 Metodología y Desarrollo del Trabajo..........................................................................22

* 1. Metodología.............................................................................................................................23
  2. Relatoría del proceso..............................................................................................................24
     1. Tratamiento de Imagen.....................................................................................................24
        1. Búsqueda de Umbralización en formato .JPG...................................................................24
        2. Búsqueda de Umbralización en formato .TIF…………………………………………. 26
        3. Aplicación de perspectiva y selección de renglones……………………………………..26
        4. Conectividad de Astas y Caídos………………………………………………………...28
        5. Primera aproximación a la división de palabras…………………………………………29
        6. Segunda aproximación a la división de palabras………………………………………...29
        7. Tercera aproximación a la división de palabras………………………………………….31
        8. Primera aproximación a la división de carácter………………………………………….32
        9. Segunda aproximación a la división de carácter………………………………………...33
        10. Tercera aproximación a la división de carácter………………………………….34
     2. Creación de GUI…………………………………………………………………………35
        1. Primer intento de creación de GUI………………………………………………………35
        2. Segundo intento de creación de GUI…………………………………………………….36
     3. Implementación de la Inteligencia Artificial…………………………………………….37
        1. Creación del banco de aprendizaje………………………………………………………37
        2. Implementación del “*Scale-Invariant Feature Transform*”(SIFT)……………………...38
        3. Implementación del “*K-means”* (K-medias)…………………………………………….39
        4. Implementación del *Multi-Layer Perceptron* (MLP)……………………………………39
     4. Automatización…………………………………………………………………………..40
        1. Primer proceso de automatización……………………………………………………….40
        2. Segundo proceso de automatización……………………………………………………..41
        3. Tercer proceso de automatización………………………………………………………41
  3. Explicación de los archivos constitutivos del programa……………………………………...42
     1. *GUI.py*……..……………………………………………………………………………..44
     2. *startupFunctions.py*…….………………………………………………………………..45
     3. *thresholdFunctions.py*……………………………………………………………………45
     4. *lineFunctions.py*………………………………………………………………………….46
     5. *wordFunctions.py*……….………………………………………………………………..46
     6. *charFunctions.py*………………………………………………………………………..47
     7. *main.py*……..……………………………………………………………………………48
     8. *SIFTnK.py*………………………………………………………………………………..48
     9. *initializeAI.py*………………………………………………………………………….…49
     10. *dictTranslate.py*……….…………………………………………………………………49

1. Resultados………………………………………………………………………………………50
   1. Generalidades del proceso de aplicación de las pruebas.50
   2. Descripción de las Pruebas Parciales……….51
      1. Prueba Uno. ………………………………………………………………………………54
      2. Prueba Dos. ………………………………………………………………………………57
      3. Prueba Tres. ………………………………………………………………………………60
      4. Prueba Cuatro. …………………………………………………………………………….63
      5. Prueba Cinco. ……………………………………………………………………………...66
      6. Prueba Seis. ……………………………………………………………………………….69
      7. Prueba Siete. ………………………………………………………………………………72
   3. Resultados Finales. …………………………………………………………………………….74
      1. Matriz de confusión prueba cinco. ………………………………………………………….75
      2. Matriz de confusión prueba siete. …………………………………………………………...78
2. Conclusiones. …………………………………………………………………………………82
   1. Conclusiones Teóricas. ……………………………………………………………………...82
   2. Proyecciones del Programa. …………………………………………………………………83
   3. Retos. …………………………………………………………………………………………83
3. Glosario. ………………………………………………………………………………………84
   1. SIFT. ………………………………………………………………………………………...84
   2. K-Means. …………………………………………………………………………………….85
4. Bibliografía. …………………………………………………………………………………..85

Lista de tablas

Tabla 1. Resultados de la prueba Uno. ……………………………………………………………………54

Tabla 2. Sistematización de los resultados de la prueba Uno..…………………………………………….55

Tabla 3. Resultados de la prueba Dos. ……………………………………………………………………57

Tabla 4. Sistematización de los resultados de la prueba Dos………………………………………………58

Tabla 5. Resultados de la prueba Tres………………..……………………………………………………60

Tabla 6. Sistematización de los resultados de la prueba Tres………………………………………………61

Tabla 7. Resultados de la prueba Cuatro……………………………..……………………………………63

Tabla 8. Sistematización de los resultados de la prueba Cuatro………………..…………………………64

Tabla 9. Resultados de la prueba Cinco…………………….……………………………………………..66

Tabla 10. Sistematización de los resultados de la prueba Cinco……………..……………………………67

Tabla 11. Resultados de la prueba Seis………………….…………………………………………………69

Tabla 12. Sistematización de los resultados de la prueba Seis…………….………………………………70

Tabla 13. Resultados de la prueba Siete……………….………………………………………………….72

Tabla 14. Sistematización de los resultados de la prueba Siete…………………………………………….73

Tabla 15. Matriz de confusión de la prueba Cinco…………………………………………………………75

Tabla 16. Matriz de confusión de la prueba Siete………………………………………………………….78

Tabla 17. Total de aciertos de todas las pruebas……………………..…………………………………….81

Tabla 18. Resultados generales de las pruebas…………………………..…………………………………82

Lista de figuras

Fig. 1. Resultado de la umbralización de imagen en formato .JPG………………………………25

Fig. 2. Resultados de la umbralización y proceso de apertura y cierre en formato .TIF………….26

Fig. 3. Aplicación de perspectiva en el primer párrafo folio 21v…………………………………27

Fig. 4. “Línea 3” . ………………………………………………………………………………..28

Fig. 5. Segunda aproximación a división de palabras…………………………………………….31

Fig. 6. Análisis “wide” y análisis “close”...………………………………………………………32

Fig. 7. Resultados de los análisis “wide” y “close”……………………………………………….32

Fig. 8. Resultados de la sumatoria de los análisis “wide” y “close”……………………………..32

Fig. 9. Primera aproximación a división de caracter……………………………………………...33

Fig. 10. GUI final con RM-149 folio 21v cargado. ………………………………………………37

Fig. 11. Flujograma del programa Perséfone…………………………………………………….50

Fig. 12. Imágenes 49 y 94, Prueba 6……………………………………………………………...53

**Capítulo 1**

**Introducción**

“*Science surpasses the old miracles of mythology*”[[1]](#footnote-1) es una cita de Ralph Waldo Emerson, que pareciera cada día hacerse más cierta.

Hace un siglo, los hermanos Wright le ganaron a Ícaro en su afanosa carrera de liberarse de las cadenas que atan al hombre a la tierra y tocar los espacios etéreos exclusivos de los vientos y las aves. Luego, en menos de setenta años, Yuri Gagarin estaba más allá de la bóveda prohibida a los humanos, entre las constelaciones y las estrellas errantes, explorando los aposentos donde el Olimpo debería estar. Con las misteriosas artes de Nikola Tesla, se domó al rayo y al relámpago sin nube o tormenta, a diferencia de como lo hacía Zeus Cronión; dando calor y luz cuál ironía al deseo de Prometeo.

Sin embargo, muchos de los secretos de los dioses aún yacen en los arcanos y crípticos relatos de sus desventuras y proezas. Los humanos, buscando emular a los dioses, han creado autómatas a su imagen y semejanza. Sin embargo, estos sirvientes de acero y silicio no poseen algo enteramente humano: el don de la palabra.

A pesar de que es un regalo innato de los seres humanos, para estos aún es un misterio por comprender y explorar. Desde los cánticos sagrados de los sacerdotes primitivos hasta los videos que iluminan las pantallas de los teléfonos celulares de la multitud ávida por conocer la vida íntima de sus ídolos y líderes, pasando por todos los discursos que enardecen a la multitud y las canciones populares que resuenan en las cantinas de los pueblos, la lengua codifica esos mensajes que intentan animar, informar, conmover y enseñar aquello que nace en el asiento del alma. Sin embargo, todo aquello que es dicho se pierde entre los murmullos del viento e incapaz de mantener su ímpetu frente a los elementos y la distancia. Es ahí, en la necesidad de guardar lo secreto y lo manifiesto, que nace la escritura.

Para los antiguos, este poder era un regalo de los dioses. Así pues, para los egipcios fue Thoth el patrón de la escritura, siendo llamado ese sistema Jeroglífico, del griego *ἱερογλυφικός*: grabado sagrado; y en su lengua:R8S43Z3, *mdw nṯr*, palabras del dios. Sin embargo, este regalo no fue bien visto por el Rey Thamus, quien le dijo a Thoth al terminar su creación:

“*No es, pues, un fármaco de la memoria lo que has hallado, sino un simple recordatorio. Apariencia de sabiduría es lo que proporcionas a tus alumnos, que no es verdad. Porque habiendo oído muchas cosas sin aprenderlas, parecerá que tienen muchos conocimientos, siendo, al contrario, en la mayoría de los casos, totalmente ignorantes, y difíciles, además, de tratar porque han acabado por convertirse en sabios aparentes en lugar de sabios de verdad.*” [[2]](#footnote-2)

Luego de la caída de los dioses antiguos por nuevas creencias, los mitos quedaron reducidos a cuentos, relatos paganos en las márgenes de la historia. Muchos avances se dieron en los estudios de la escritura, materia que es apasionante, más tangencial al asunto entre manos de esta tesis. Por ahora es necesario hablar del consenso al cual la ciencia ha llegado luego de años de aprendizaje.

Debido a su ubicuidad, definir la escritura pareciera innecesario y hasta cierto punto contraproducente. Sin embargo, para John DeFrancis la escritura se puede denominar como “*[u]n sistema de símbolos gráficos que se pueden usar para transmitir cualquier y todo pensamiento*”[[3]](#footnote-3). Esta definición general permite ver cómo ha evolucionado la escritura.

Según los descubrimientos arqueológicos, podemos decir que la escritura fue inventada en Mesopotamia alrededor del 3300 A.C. en tablillas de barro quemadas. Estas tablillas tenían el propósito de mantener la contabilidad de las transacciones comerciales. La escritura era pictográfica, es decir, utilizaban imágenes para representar el objeto: peces, bueyes, palmas, cabezas, etc. El símbolo y el significado eran uno solo. Con el tiempo, los pictogramas dieron lugar a formas de escritura tales como el alfabeto fenicio, el cual dio lugar al alfabeto griego, del cual el etrusco nace, para así originar el abecedario latino. El primer ejemplar atestiguado de este sistema de escritura es la *Fíbula de Presente* que data del siglo Séptimo antes de Cristo.

El latín es innegablemente una de las lenguas más importantes para la historia. Siendo la lengua oficial del Imperio Romano, se expandió por toda la cuenca del Mediterráneo, siguiendo los caminos imperiales pavimentados sobre las conquistas de los generales y el sudor de los esclavos. Luego pasó a ser la lengua del Catolicismo y, por ende, la lengua franca en aspectos intelectuales y comerciales. Debido a eso, el corpus de documentos escritos en esta grandiosa lengua es inconmensurable, siendo la lengua en la que se escribió la magnífica obra de Cicerón “*De re publica*” y los “*Principia Mathematica*” de Newton, obras separadas por alrededor de mil setecientos años. Su impacto en la cultura global es indiscutible. Más adelante, se hablará de la importancia del latín para la sociedad colonial latinoamericana.

Haciendo un salto hacia el tiempo presente, donde los computadores hacen cálculos a velocidades superiores a los de la mente humana y su cercanía a la total autonomía es cada vez más impresionante. Como se dijo anteriormente, una de las características que diferencian a las máquinas de los humanos es el lenguaje. Sin embargo, para programar a una máquina es necesario utilizar el lenguaje. Tal como el Golem de la mitología judía, las máquinas necesitan una entrada de información para poder realizar las tareas que les han sido asignadas. Por esta razón, la filología no puede negarse a tal ayuda. Los avances al respecto se tratarán en el capítulo siguiente, así como la descripción y definición del lenguaje de programación.

La presente tesis pretende resolver la pregunta: ¿cómo optimizar, por medio de OCR y la inteligencia artificial, el proceso de transcripción del manuscrito RM-149? La respuesta a esta pregunta es la creación de un programa para la transcripción del manuscrito RM-149, cuyo autor es Mateo Mimbela, los resultados obtenidos y los retos que trae consigo este avance, además de argumentar sobre la necesidad de incluir la informática dentro del repertorio filológico de herramientas para el ejercicio de la creación y la innovación.

Como herramienta para aquellos lectores no iniciados en el mundo de la programación, se ha creado un glosario con aquellos términos que pueden ser confusos y complejos, de esta manera se busca acercar a aquellos que les pueda interesar esta vía del quehacer filológico sin ser expertos en computación.

**Capítulo 2**

**Marco teórico**

El presente marco teórico expone los tópicos pertinentes al proceso de creación del programa Perséfone. En primer lugar, se presenta el origen del manuscrito RM-149, “*Physices Tractatus*”, así como la vida del autor del texto y el contexto en el que se escribió. En segundo lugar, se exponen los conceptos de reconocimiento óptico de caracter, redes neuronales y aprendizaje de máquina y, por último, la importancia de la utilización de estas tecnologías en el ámbito filológico y los avances que se han realizado en esta materia.

* 1. **El “*Physices Tractatus*” y la Colonia**
     1. **Contexto histórico del manuscrito.**

En el Nuevo Reino de Granada, la transcripción y el oficio del copista seguía vivo a pesar de que en Europa esta forma de creación de documentos había desaparecido, debido a la popularización de la Imprenta[[4]](#footnote-4). Durante esta época, la producción textual manuscrita yacía mayoritariamente en las universidades y demás centros del conocimiento, puesto que la necesidad de los libros para la educación de los estudiantes hacía imperativa la creación de textos por parte de los mismos. La solución a este problema fue la continuidad de las formas escolásticas en las aulas.

Los mecanismos de enseñanza escolástica que atañen a este trabajo son la *Lectio* y la “*Dictatio*”. “*Lectio*” se puede definir como la lectura en voz alta por parte del docente de su libro, mientras que los estudiantes copiaban palabra a palabra el mensaje del catedrático, siendo esta parte la “*Dictatio*”[[5]](#footnote-5). Es necesario aclarar que no solamente la falta de textos de aprendizaje mantenía este estilo de enseñanza vigente. Por un lado, la “dictatio” entrenaba a los estudiantes para ser copistas proficientes, asimismo, estos manuscritos tenían un valor social en un mundo donde el acceso a los textos era bastante limitado, sirviendo inclusive como prueba de la educación recibida.[[6]](#footnote-6) Una característica importante es la nomenclatura de estos textos. Eran llamados mamotretos, cuadernos en blanco encuadernados en pergamino con papel con filigrana. Normalmente, la primera página era finamente decorada, siendo esta una muestra de pericia caligráfica.[[7]](#footnote-7) El “*Physices Tractatus*”, texto sobre el cual se va a trabajar, es uno de los múltiples mamotretos que se realizaron en este ámbito.

* + 1. **Sobre el autor del manuscrito.**

Antes de exponer el manuscrito, es importante conocer su autor. Según Marquínez Argote (2002), Mateo Mimbela nació el 20 de noviembre de 1663 en Fraga, Reino de Aragón (España). Es iniciado como novicio en La Compañía de Jesús el 21 de noviembre de 1677 en Tarragona y empieza allí sus estudios en Humanidades Clásicas. Parte al Nuevo Reino de Granada en misión en 1690, desembarcando el 3 de mayo del mismo año en Cartagena de Indias. Luego se asentó en Santafé de Bogotá, donde es nombrado catedrático en la Universidad Javeriana. Durante estos tres primeros años de cátedra, escribió el “*Physices Tractatus*”, fruto del curso de física.

Al concluir su ciclo de enseñanza, vivió tres años en los Llanos Orientales como misionero. Vuelve a Bogotá para presentar el informe sobre los indígenas que vivían en la región, además de otros conocimientos adquiridos en su viaje. A pesar de sus deseos de volver a los Llanos, le ordenan continuar su trabajo académico; durante este periodo de su vida escribe gran parte de su producción intelectual. Muere el 22 de abril de 1736, estando en ejercicio de su tercera designación como rector de la Universidad Javeriana.[[8]](#footnote-8)

* + 1. **El Manuscrito RM-149.**

Luego de esta breve nota biográfica del autor, es importante hablar del texto de su autoría, objeto de este trabajo. El título completo del manuscrito es: “*Physices Tractatus, per R. P. Matthaeum Mimbela Societatis Iesu, Dignissimum philosophia catedrae praeceptorem Huius Xaverianae Accademiae civitates SanctaFidensis. Die vigesima mensis octobris, anno a Nativitate Domini 1693. Iohanne de Herrera, auditore*”. Gracias a este título, se puede saber que el copista que realizó este mamotreto fue Juan de Herrera. El códice posee ciento ochenta y cuatro folios, además de estar bellamente ilustrado a mano; según Marquínez Argote (2002), este texto podría ser el resultado de una transcripción posterior a la clase, debido al esmero de la caligrafía[[9]](#footnote-9). Este texto reside en la Biblioteca Nacional de Colombia bajo la numeración RM-149.

Teniendo presentes los trazos de la historia del autor y parte de las características del manuscrito, el contenido del texto es el siguiente punto por tratar. El “*Physices Tractatus*” es un texto de física, tal como su nombre lo indica; sin embargo, es importante tener en cuenta que la física de la época era una rama de la Filosofía y, por ende, una rama de la Teología. Muchas de sus explicaciones se basan a partir de la Física de Aristóteles, así como su forma de argumentación es netamente silogística. Otra de las fuentes importantes de Mimbela son los trabajos de Santo Tomás de Aquino; buena parte de la argumentación teológica de los fenómenos naturales se basa en la interpretación tomista de Aristóteles. Sin ahondar en las discusiones filosóficas del manuscrito, puesto que son tangenciales en el presente texto, cabe resaltar que sus planteamientos fueron considerados de suma importancia. Las enseñanzas del texto estaban basadas en las enseñanzas de la escuela jesuítica del momento[[10]](#footnote-10), por lo cual eran muy apreciadas en su época.[[11]](#footnote-11)

Resaltando la importancia de este documento en su contexto histórico, es importante ahora hablar de las tecnologías usadas en el programa que transcribirá el manuscrito.

* 1. **OCR y Red Neuronal**
     1. **El OCR y su historia.**

El reconocimiento óptico de caracter, “OCR” por sus siglas en inglés, es uno de los adelantos más importantes en la tecnología del último siglo. Comenzando por la creación de una máquina de lectura para ciegos de Fournier d’Albe[[12]](#footnote-12), la idea de una máquina leyendo un texto ha sido una constante en la psique humana. Durante la década del sesenta, se crea la IBM 1287[[13]](#footnote-13), una máquina que leía caracteres tanto impresos como escritos. Actualmente, Los OCRs existen por doquier, desde el código de barras de cualquier producto, hasta el auto corrector de los celulares. Dicho esto, es importante tener en cuenta qué es un OCR.

Según Line Eikvil “[l]as técnicas de reconocimiento de caracteres asocian una identidad simbólica con la imagen de un caracter”.[[14]](#footnote-14) Esto significa que a la imagen de un caracter se le da un valor determinado. Un ejemplo de ello son los códigos de barra, una imagen que no tiene significado por sí mismo; pero que, cuando la máquina lo lee, identifica el producto porque tiene asociado el producto con esta imagen. Como el ejemplo anterior, para algunos casos de reconocimiento de caracter es necesario tener un *hardware* especializado; otros, por el contrario, pueden ser usados desde un computador no especializado. Un ejemplo de esta situación son los códigos QR, que pueden ser leídos por un teléfono celular que tenga tanto el programa para analizar la imagen como una cámara incluida.

Pasando al contexto de la escritura, uno de los usos del OCR es el reconocimiento de letras. En este ámbito, se puede hablar del reconocimiento especializado en letras impresas, manuscritas y OCR mixtos. Debido al interés y al objeto de la presente tesis, es preciso enfocarse en los OCR destinados a la letra manuscrita. Este tipo de tecnología busca identificar letras y símbolos varios dentro de los textos realizados en letra manuscrita. La forma seleccionada en el presente trabajo fue a partir de entrenar una red neuronal para poder reconocer los patrones comunes entre los caracteres ingresados y así poder dar un resultado. Sobre las redes neuronales artificiales y su forma de aprendizaje se hablará a continuación.

* + 1. **Aprendizaje de máquina y redes neuronales: contexto histórico y aplicaciones.**

Definir las redes neuronales no es tarea sencilla debido a la matemática que hay detrás de este avance tecnológico. Para no agobiar al lector con esta serie de definiciones matemáticas, se puede decir que es una red de neuronas que intenta emular la forma de aprender de un animal. Cada neurona que constituye la red neuronal es un nodo donde se procesa la entrada de datos definidos y arroja un resultado consecuente a este. Este modelo fue planteado por Warren McCulloch y Walter Pitts en su texto “*A logical Calculus of the Ideas Immanent in Nervous Activity*” de 1943. Se han creado diversos modelos de redes neuronales artificiales con el paso del tiempo, el modelo “*Multi-Layer Perceptron*”, se utilizó en el programa Perséfone, eje central de la presente tesis, este modelo es un desarrollo del trabajo de Frank Rosenblatt[[15]](#footnote-15). El “*Multi-Layer Perceptron*” (en adelante MLP) tiene la característica de poseer tres capas de nodos. La primera capa es una capa de input donde se organizan los datos de entrada, para así direccionarlos a los nodos de las capas ocultas correspondientes con la categoría que se considere más apropiada. Luego pasa por una o múltiples capas ocultas, donde se analizan los datos de entrada. Por último, la capa de salida arregla los datos para dar un resultado frente[[16]](#footnote-16).

En cuanto al aprendizaje de máquina, es clave tener en cuenta que hay gran variación dependiendo de qué tipo de red neuronal se esté usando. Como se había comentado anteriormente, la red neuronal que se seleccionó, el MLP, es una red supervisada, es decir, posee un banco de aprendizaje con el cual comparar los resultados obtenidos. El método usado es “*Backpropagation*”, en palabras de Shashi Satyanarayana es “...lograr encontrar los tales pesos, que, para cada vector de entrada en el set de entrenamiento, la red neuronal entrega un vector de salida similar al vector objetivo prescrito.[[17]](#footnote-17)”

Teniendo en cuenta el tipo de tecnología usada en el desarrollo del programa Perséfone, es importante hablar de la necesidad de estos avances en la Filología Clásica.

* 1. **Las Nuevas tecnologías y la Filología Clásica**
     1. **La Necesidad de las Nuevas Tecnologías en la Filología Clásica.**

¿Por qué es necesario que un filólogo tenga dentro de su juego de herramientas las tecnologías anteriormente mencionadas? Puede que para muchos sean un abalorio que entorpezca el quehacer filológico, tal como el rey Thamus le decía a Thoth sobre la escritura en el relato de Platón citado en la introducción. Sin embargo, los ordenadores y las inteligencias artificiales han permeado la vida diaria y negar a una disciplina tan poderoso instrumento es, como mínimo, retrógrado. Según Peter Robinson: “[t]he most immediate value of digital methods is the ability to assign as much as possible of the work—particularly that which is repetitive, exacting, and error-prone—to the computer.”[[18]](#footnote-18)

Ahora bien, es necesario especificar sobre cuál de los trabajos filológicos se estará realizando en este texto, puesto que en el quehacer filológico hay múltiples estadios de trabajo; debido a la naturaleza del proyecto, se enfocará en el proceso de transcripción.

Desde los inicios de la filología, se ha propugnado por mantener la información accesible en términos de lectura y comprensión. Según Reynolds y Wilson en *Scribes and Scholars* (1991), uno de los primeros usos de la transcripción fue para salvaguardar las obras de teatro que se presentaban en Atenas, debido a la necesidad de los actores de tener estos documentos; se especula que estos textos debieron ser transcritos de colecciones privadas.

Estos esfuerzos dieron paso a la creación de colecciones privadas, tales como la de Aristóteles en el Liceo y la Biblioteca de Alejandría. Es importante resaltar que los sabios de la Biblioteca encontraban errores en las diferentes ediciones de los textos que traían y copiaban. Gracias a sus conocimientos de las obras y su juicio personal, componían nuevas versiones sobre los textos. Esto llevó a la creación de ediciones críticas, actividad que inició propiamente la ciencia filológica. Haciendo un salto temporal, vale la pena comentar que el arte de la transcripción se siguió manteniendo viva en la Colonia neogranadina, como se anotó anteriormente.

A pesar del conocimiento de los copistas, ninguna transcripción es perfecta, debido a una serie de errores tipificados por la crítica textual, tales como los errores por adición, omisión, alteración del orden y sustitución.[[19]](#footnote-19) Por este motivo, realizar transcripciones es un proceso complicado. Además de conocer el tipo de fuente usado por el copista, es necesario saber las formas de abreviatura utilizadas en el manuscrito, esto implica tener un conocimiento de la lengua que permita resolver las contracciones y omisiones realizadas por el copista en su afán de mantener el ritmo de escritura. Aunque es necesario aprender estos conocimientos en el proceso de formación profesional, tener que transcribir todos los textos que se vayan a usar en un proceso de crítica textual puede llegar a ser esclavizante. Según Andrews (2012), “Mientras que (el) OCR y otros métodos automatizados puedan acelerar el trabajo significativamente en el futuro, incluso algún día incluso alcance una transcripción automática perfecta… (La) transcripción sigue siendo un ejercicio útil, sí es uno meticuloso.” [[20]](#footnote-20)

Asimismo, hablar de la naturaleza de la transcripción en este caso es importante. Según Robinson y Solopova (1993, 21) “[la] transcripción de una fuente textual primaria no puede ser tomada como una sustitución, sino como una serie de actos de traducción de un sistema semiótico (aquel de la fuente primaria) a otro sistema semiótico (el de la computadora). Como todos los actos de traducción, este debe ser visto como fundamentalmente incompleto y fundamentalmente interpretativo.” [[21]](#footnote-21) Aunque esta definición se puede aplicar a las transcripciones del estilo de Text Encoding Initiative-TEI, donde el texto debe ser cambiado al formato de escritura de este sistema, no es una complicación propia de las transcripciones automáticas. Esto se debe a que en ningún momento se modifica el texto base, solo es alterada la imagen sobre la cual se hace el proceso, por este motivo, es necesario plantear una definición sobre qué es una transcripción automática en el ámbito filológico.

Primero, se puede afirmar que el producto de la transcripción en el caso del programa Perséfone son las imágenes con las más altas probabilidades de reconocimiento, por tanto, el objetivo se enfoca en el proceso de darle un input a la máquina y obtener un output que debe ser curado. Este proceso, que relega al filólogo a un rol de examinador, permite tener una mayor precisión durante el proceso interpretativo si es el caso de una traducción o en las técnicas propias de la crítica textual. Una situación que se ilustra en Robinson (2013), [[22]](#footnote-22) donde un punto por debajo del renglón en una palabra de la Divina Comedia modifica el significado de la palabra; por tanto, dependiendo del tipo de lectura que realice el filólogo, la palabra tendrá un significado diferente. Aunque existen ejemplos como el anterior donde la ambigüedad se puede resolver con relativa simplicidad teniendo en cuenta el contexto de la palabra y la claridad de escritura que posee el testimonio, esta situación no se presenta en la mayoría de los casos. Por esta razón, una herramienta que permita tener mayor exactitud en la lectura del manuscrito es importante para poder realizar cualquier tipo de procedimiento.

Sobre la definición de transcripción automática, se puede indicar que es el mecanismo por el cual el filólogo depura la lectura de un documento por medio de una inteligencia artificial y obtiene un resultado con un alto grado de exactitud que debe ser curado para la realización de cualquier proceso de crítica textual.

Es vital tener en cuenta que esta definición se aplica al contexto de la transcripción automática de manuscritos, ya que otros tipos de transcripciones automáticas no requieren el proceso de lectura interpretativa por parte del usuario. Cabe destacar que con el paso de los años esta definición puede ser rebatida o modificada, en el tiempo actual de elaboración de la presente tesis, la transcripción automática de manuscritos antiguos todavía es un tópico en desarrollo, que conlleva una serie de dificultades y problemáticas propias de una herramienta en construcción. Sin embargo, es importante tratar de definir las nuevas herramientas para así poder darles un uso más acertado; es más, esto puede dar paso a mejores, más exactas y complejas definiciones que permitan afianzar el uso de la tecnología en el día a día del filólogo. A continuación, se darán ejemplos de cómo las nuevas tecnologías ya tienen cabida en los proyectos filológicos.

* + 1. **Avances en el contexto filológico.**

Con base en lo anterior, es importante presentar los esfuerzos realizados sobre la utilización de las nuevas tecnologías en la filología y campos afines.

El primer proyecto por resaltar es TEI, siglas de “Text Encoding Initiative”, mencionado anteriormente; el cual busca desarrollar y mantener un estándar en la representación de los textos en forma digital[[23]](#footnote-23). Utilizando XML para codificar los textos, esta iniciativa que data de 1994 ha sido reconocida por su importancia en la preservación de los textos y su utilización en diversas áreas.[[24]](#footnote-24)

En el marco de la crítica textual, el programa Juxta permite realizar el proceso de *collatio* en múltiples testimonios de un texto que estén en formato XML o TXT. Este programa permite hacer desarrollo personalizado debido a que es *open-source*, lo cual le da la facultad al usuario de usarlo conforme a sus necesidades.[[25]](#footnote-25)

En Colombia, uno de los esfuerzos realizados es la digitalización realizada por la Biblioteca Virtual del Pensamiento Filosófico en Colombia de la Pontificia Universidad Javeriana. El documento digital, realizado por un equipo encabezado por el Dr. Manuel Domínguez Miranda, es una excelente herramienta de trabajo, permite un acceso fácil y directo a los diversos manuscritos, incluyendo al RM-149. Sin embargo, durante el proceso de creación de Perséfone, se utilizaron las digitalizaciones realizadas sin el proceso de edición que hizo el equipo del Dr. Domínguez debido a que la resolución del archivo no permitía los manejos precisos que se realizaron en el programa, las razones de esta decisión serán explicadas posteriormente en el apartado “Búsqueda de la umbralización en formato .JPG.

Con relación a los programas y proyectos enfocados al uso de OCR en las Humanidades Clásicas, el primero a resaltar es tranScriptorium, una iniciativa de múltiples universidades europeas, tales como la Universidad Politécnica de Valencia, la Universidad de Innsbruck, el Centro Nacional para la Investigación Científica de Grecia, entre otras[[26]](#footnote-26). En este proyecto se busca la indexación, búsqueda y transcripción de imágenes de documentos históricos en español, alemán, inglés y holandés[[27]](#footnote-27). Luego de concluido el proyecto tranScriptorium[[28]](#footnote-28) en 2016, se dio paso a READ, el cual posee los mismos intereses, con el aditamento de tener más universidades inscritas al proyecto e incorporar *crowdsourcing*, es decir, la colaboración del público general para resolver problemas. Por otro lado, este proyecto posee su propio programa para realizar transcripciones llamado “*transkribus*”.

El proyecto de transcripción de documentos manuscritos medievales realizado por Markus Wüthrich et al. es un trabajo que vale la pena resaltar, debido a que su utilización de red neuronal, siendo una “*bidirectional long short-term memory*”, les arrojó un 93.32% de reconocimiento usando las imágenes de “*Parzival*”, poema épico escrito en el siglo XIII por Wolfram von Eschenbach.[[29]](#footnote-29)

Un último trabajo por resaltar es el realizado por Markus Diem y Robert Sablatnig, siendo un sistema muy parecido al usado en Perséfone. El objetivo del proyecto de Diem y Sablatnig era la identificación de caracteres del alfabeto glagolítico utilizando SIFT en imágenes no binarizadas. A pesar de ser poco convencional esta aproximación, los resultados que arroja su estudio demuestra que se puede utilizar este sistema en documentos en pobre estado de conservación.[[30]](#footnote-30)

Viendo estos ejemplos exitosos de implementaciones de nuevas tecnologías a los diversos campos de las Humanidades Clásicas, es momento de hablar del programa realizado para cumplir la tarea de transcribir el manuscrito RM-149.

**Capítulo 3 Metodología y Desarrollo del Trabajo**

* 1. **Metodología**

Para abordar este trabajo, se comenzó por plantear una ruta de trabajo, la cual permitiría tener organizado el avance del proyecto. El primer paso de la ruta de trabajo fue dividir en dos partes el proyecto. La primera estaba dedicada al tratamiento de la imagen y el segundo en la elaboración de la red neuronal y el entrenamiento de esta. Durante el proceso, esta división se hizo cada vez más extensa; ya que, al avanzar en el proyecto surgieron diferentes procesos que no estaban relacionados directamente con estos dos temas, pero que eran de vital importancia para su realización. El primer subíndice tratará sobre el proceso de tratamiento de imagen; luego, se hablará sobre el proceso de creación de la interfaz gráfica para el usuario; En tercer lugar, se explicará la composición de la inteligencia artificial y su entrenamiento. Por último, se abordará el proceso de automatización del programa.

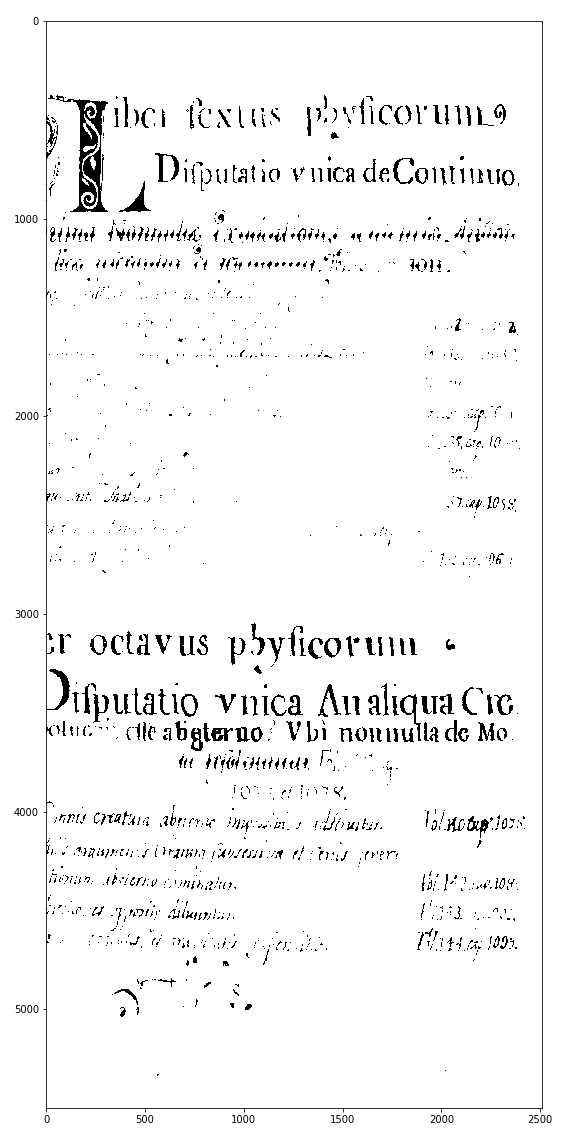
Para la relatoría del trabajo, primero se planteó hacer un diario narrado, el cual tendría consignado todos los avances del proyecto. Este método fue poco efectivo, debido a que muchas sesiones tenían el mismo fin y se tornaba tanto monótona como confusa la lectura del documento. Por esta misma razón, se decidió usar esquemas explicativos del proceso para poder presentar de manera clara y concisa la información. Los esquemas recogen los avances hechos en una subárea, primero comentando el problema que se quería resolver, para luego plantear tanto el proceso realizado como los resultados obtenidos. Dicho esto, la disposición del trabajo está planteada a partir de las temáticas y no del orden temporal del trabajo mismo; ya que en algunas sesiones se adelantaron tareas que no necesariamente estaban concatenadas temáticamente unas con otras.

Cabe aclarar que, por la misma naturaleza del formato, no se harán explicaciones sobre el léxico especializado que se maneja dentro del texto. Teniendo en cuenta que esta monografía tiene como público personas que no están especializadas en tratamiento de imagen o programación, hay un glosario donde los conceptos más ajenos a la disciplina filológica estarán consignados.

* 1. **Relatoría del proceso**
     1. **Tratamiento de Imagen (OCR).**
        1. Búsqueda de Umbralización en formato .JPG

|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Buscar una umbralización que sea apropiada para la lectura del texto por parte de las personas y de fácil identificación por parte de la red neuronal. | Primero, fue hacer una umbralización binaria, pasando del color natural a blanco y negro.  Luego el corte de las márgenes en blanco de la página. Para lograrlo se realizó un histograma que mostraba el primer punto donde había un píxel negro, para así hacer un corte limpio. | La calidad de las imágenes en formato .JPG era muy pobre, haciendo la lectura difícil. Sin embargo, permitió determinar que la umbralización binaria podría ser un buen inicio. Así mismo, se determinó innecesario hacer los cortes de las márgenes.  (Ver fig. 1). |

Fig. 1. Resultado de la umbralización de imagen en formato .JPG.

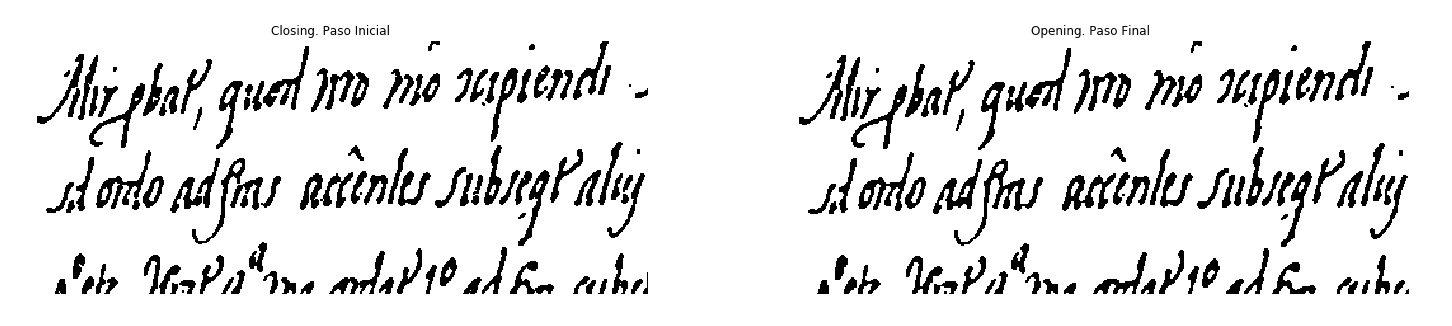


Fuente: elaboración propia.

3.2.1.2 Búsqueda de Umbralización en formato .TIF

|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Buscar umbralización correcta en las nuevas imágenes en formato .TIF. | Durante este proceso se realizaron múltiples tipos de umbralización: binaria, Gaussiana y Otsu. Ninguna de estas umbralizaciones lograba su cometido. Por esto, se sumaron las umbralizaciones Otsu y Gaussiana. Otro proceso realizado fue la apertura y el cerrado del trazo de la imagen. | Uno de los resultados fue la umbralización definitiva para el resto del trabajo junto al proceso de apertura y cerrado para el refinamiento del trazado del texto.  (Ver fig. 2) |

Fig. 2. Resultados de la umbralización y proceso de apertura y cierre en formato .TIF.

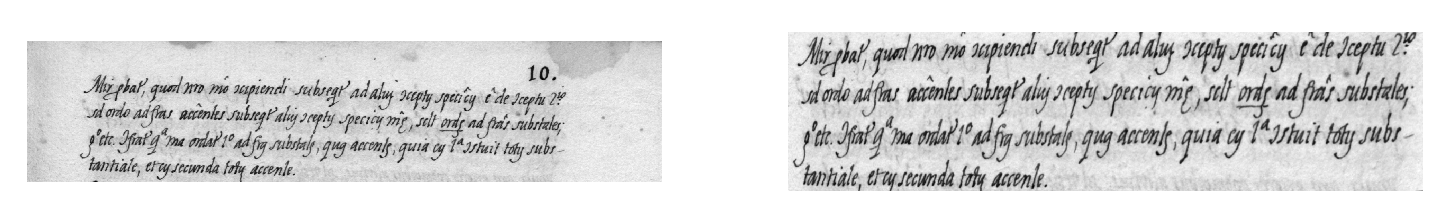


Fuente: elaboración propia.

* + - 1. Aplicación de perspectiva y selección de renglones

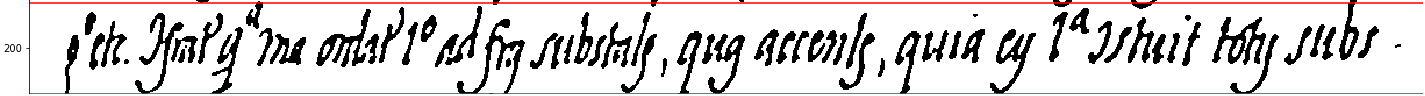
|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Corregir el ángulo de uno del primer párrafo del folio 21v y seleccionar una línea perteneciente a este. | Se emplazaron puntos en posiciones arbitrarias para ver cómo la aplicación de perspectiva modifica la imagen (ver fig.3). Después, la meta era encontrar los puntos exactos, para poder hacer la adquisición de la perspectiva correcta de manera efectiva. La primera parte fue determinar cuál es el primer píxel negro tanto en el eje vertical y en el eje horizontal. Esto se realizó en cada uno de los lados, para así encontrar el rectángulo ideal.  Para seleccionar el renglón, se hicieron proyecciones sobre la distribución de negro en los ejes vertical y horizontal. | Uno de los resultados fue encontrar el punto de perspectiva correcto para el párrafo. Así mismo, se obtuvo la distribución espacial de los renglones, lo cual permitió tener un renglón de referencia para trabajar de ahora en adelante. Este renglón fue denominado como “línea 3”.  (Ver fig. 4). |

Fig. 3. Aplicación de perspectiva en el primer párrafo folio 21v.



Fuente: elaboración propia.

Fig. 4. “Línea 3”.



Fuente: elaboración propia.

* + - 1. Conectividad de Astas y Caídos.

|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Conectar las astas y caídos de las letras a su grafema para así determinar su tamaño y figura. | Se determinó los niveles de negro en el borde inferior de la “línea 3” y el borde superior de la línea siguiente (“línea 4”). De esta manera se obtuvieron una serie de puntos donde coincidían los niveles de negro. Luego, se formuló una matriz vacía, la cual permite asignar valores hacia abajo, en la parte superior de la “línea 4” para conectar los caídos con el cuerpo de la imagen. Aunque en teoría debió funcionar, la matriz no reaccionó frente a los píxeles del trazo.  Este contratiempo llevó a la realización de un proceso de esqueletización del trazo de los grafemas. Sin embargo, no generó resultados ya que la imagen quedaba en negro, impidiendo cualquier tipo de trabajo sobre ella. | A pesar de los resultados negativos de este proceso, genera dos hipótesis sobre el porqué de este resultado. Primero, es importante analizar por qué no se genera conectividad por medio de las matrices vacías. Segundo, investigar sobre el proceso de esqueletización en los textos cursivos y en los archivos de formato .TIF, para así determinar si alguna de estas dos características del trabajo tiene alguna injerencia en los resultados atestiguados.  Este proceso se omitió en el resto del trabajo, debido a las complicaciones que generaba. Por estas razones se deja para próximas investigaciones. |

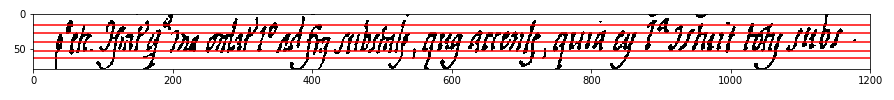
* + - 1. Primera aproximación a la división de palabras.

|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Dividir las palabras de la línea tres de la mejor manera. | El primer paso realizado fue la selección de ciento sesenta y dos puntos blancos dentro del “línea 3” manualmente. Esta selección fue realizada a partir de los resultados de una proyección que arrojaba los valores donde se determinaba la posible división de palabra. | No siempre se separaba de manera correcta las palabras. Por un lado, dividía algunas palabras en dos; por otro lado, generaba agrupaciones de múltiples palabras, dejándolas como un único grupo.  Debido a esto, fue necesario plantear nuevas aproximaciones a este problema |

* + - 1. Segunda aproximación a la división de palabras

|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultados |
| Dividir las palabras de la línea tres de la mejor manera. | Se generaron cinco líneas equidistantes, dividiendo el alto de la línea en seis para obtener porciones isomorfas.  A partir de estas líneas, se determinó la ubicación de píxeles blancos en cada una de estas (ver fig. 5). Cada una de las ubicaciones se puede denominar como un conjunto, los cuales se intersecaron para arrojar un arreglo en donde están alojados los puntos por los cuales se puede trazar una línea recta sin tocar píxeles negros.  Estos resultados daban demasiadas líneas, haciendo el proceso engorroso. Por este motivo, se realizó una función que sumaba las líneas si estas eran de una distancia igual a un píxel.  A pesar del proceso previo, seguía presentando demasiadas líneas, por este motivo se creó un nuevo arreglo en donde se almacenaban los valores repetidos de estas líneas. Se intersecaron con los valores totales y aquellos repetidos se omitieron del resultado. | Esta segunda aproximación permitió determinar el camino para lograr una división más certera de las palabras. Por otro lado, uno de los problemas evidentes de la división es la poca precisión con la cual se divide las palabras. Por esta razón se formuló una nueva aproximación que tomaba las cinco líneas, pero dándole una distancia menor entre ellas. Este punto desarrollará más adelante. |

Fig. 5. Segunda aproximación a división de palabras.

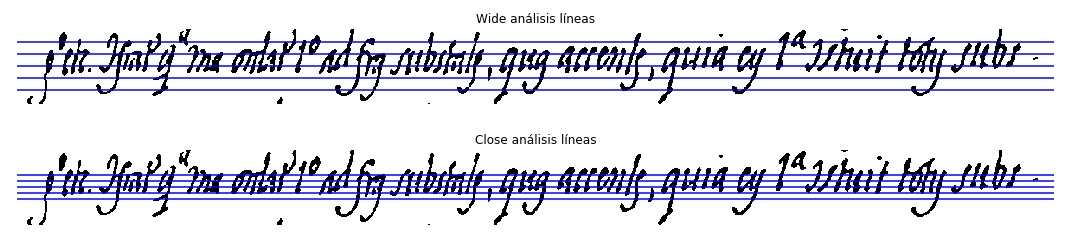


Fuente: elaboración propia.

* + - 1. Tercera aproximación a la división de palabras

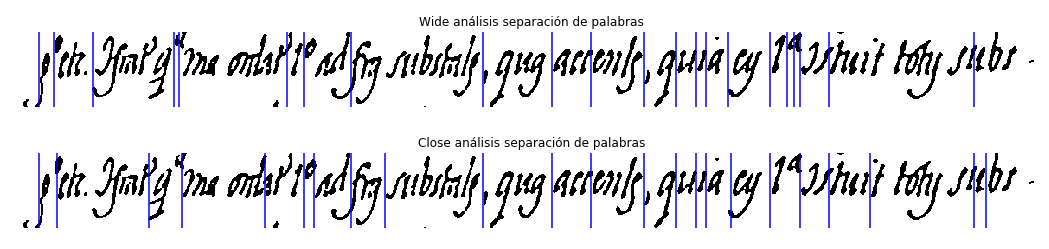
|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Dividir las palabras de la línea tres de la mejor manera. | Tomando lo realizado en el proceso pasado, se posicionaron las líneas sobre el texto. Esto creo el análisis ancho, hecho con las líneas en su posición original y el análisis delgado (ver fig. 6) con este nuevo posicionamiento de las líneas, se realizó un mismo proceso en cuanto a la ubicación de los píxeles blancos.  A partir de la comparación de estos dos conjuntos (ver fig.7), el conjunto de ubicaciones del análisis ancho y delgado. Luego de realizar este proceso, se suman los valores cercanos para dar mayor exactitud (ver fig. 8). | El proceso de separación de palabras, aunque siguen existiendo problemas con palabras que presentan múltiples ligaduras en su escritura, permitió pasar al proceso de separación de carácter. |

Fig. 6. Análisis “wide” y análisis “close”.



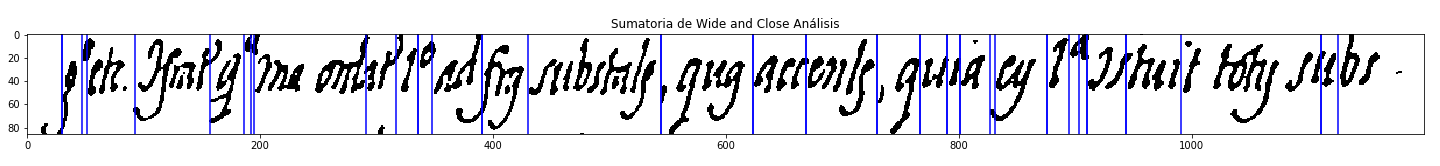
Fuente: elaboración propia.

Fig. 7. Resultados de los análisis “wide” y “close”.



Fuente: elaboración propia.

Fig. 8. Resultados de la sumatoria de los análisis “wide” y “close”.

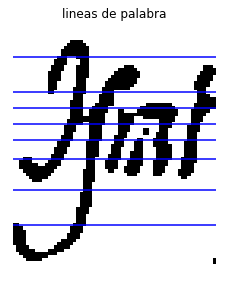


Fuente: elaboración propia.

* + - 1. Primera aproximación a la división de carácter

|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Obtener los caracteres de las palabras presentes en la “línea 3”. | La primera aproximación realizada fue utilizar el método de separación de palabras en la escala de carácter.  A pesar de tener resultados parciales, se dejó de lado aquellas partes de la imagen que contenían las astas y los caídos. Esto se realizó debido a que la cantidad de píxeles blancos es mayor en comparación a los píxeles negros en esa parte de la imagen; estos píxeles blancos generan ruido que dificulta la adquisición de los caracteres. | Aunque el programa seleccionaba algunos caracteres, no era posible dividir partes por las cuales no podía pasar una línea recta. Este problema se solucionó posteriormente implementando una rotación de la palabra a trabajar.  (Ver fig. 9). |

Fig. 9. Primera aproximación a división de carácter.



Fuente: elaboración propia.

* + - 1. Segunda aproximación a la división de carácter

|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Obtener los caracteres de las palabras presentes en la “línea 3” por medio de la rotación de la palabra. | En primer lugar, se encontraron las direcciones principales de la distribución de los puntos de la imagen para encontrar su inclinación. A partir de esto, se giró la imagen siguiendo el ángulo por estos puntos. Sin embargo, este ángulo dado no era de utilidad para rotar la imagen, por este motivo, se experimentó con diferentes proporciones hasta obtener un resultado favorable. Luego de esto, se trazaron líneas por los píxeles blancos para extraer los caracteres. | Se obtuvieron buenos resultados en segmentos cortos, logrando extraer buena parte de los grafemas de una palabra. Sin embargo, en las palabras largas el ángulo de rotación no era suficiente, por ende, no se podía dividir la palabra. |

* + - 1. Tercera aproximación a la división de carácter

|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Obtener una rotación correcta para las palabras largas, además de limpiar la imagen luego del proceso de rotación inverso. | Para solucionar el problema de las palabras con poca rotación, fijaron valores en donde los ángulos inferiores a quince grados fueran iguales a quince y los valores superiores a veinticinco grados fueran iguales a veinticinco. Debido a que durante el proceso de rotación de la imagen se produce ruido, es decir, píxeles negros que quedan mal posicionados, se hizo un proceso de cierre y luego apertura para eliminar estos píxeles después de devolver a las imágenes a su rotación original. | Se obtuvo una división de carácter suficiente para poder trabajar. Sin embargo, debido al tipo de caligrafía, la aproximación a partir de líneas rectas no es el más adecuado para el trazado, puesto que no toma en cuenta el ángulo de trazado ni el espacio que toman las astas y los caídos. Este tipo de implementación es recomendada para futuras investigaciones. |

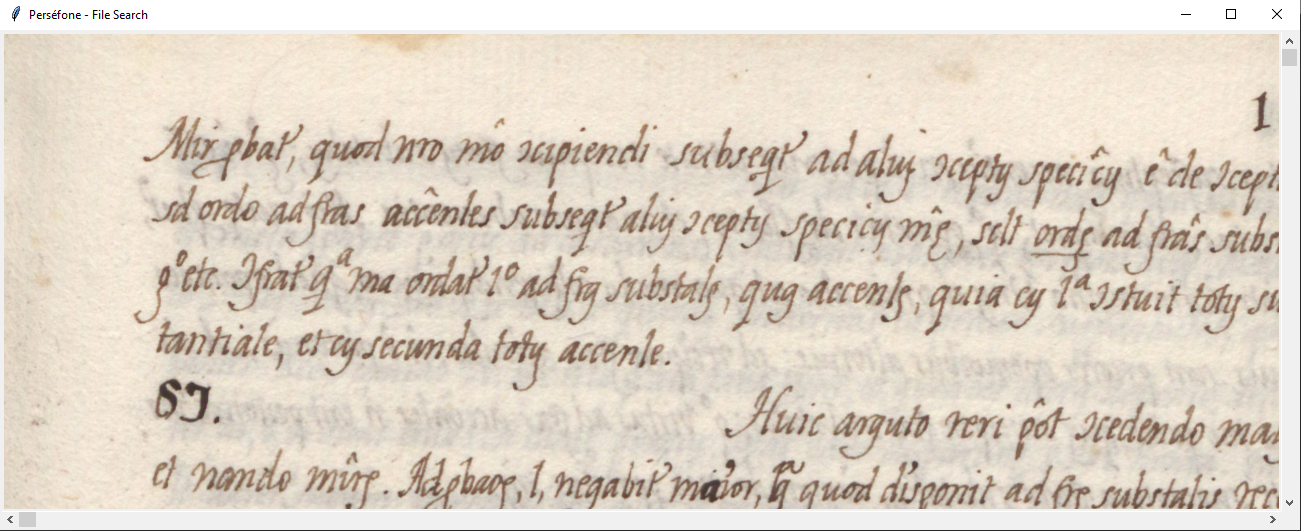
* + 1. **Creación de GUI.**
       1. Primer intento de creación de GUI

|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Crear una GUI (*Interfaz Gráfica para el Usuario*, por sus siglas en inglés) que permitiese mostrar la imagen del folio y seleccionar cuatro puntos para obtener el párrafo a trabajar. | Se buscó una librería de Python que permitiese crear una interfaz gráfica. La librería seleccionada fue “*Tkinter*”. Con esta librería, se planeaba generar un *canvas*, en el cual se pudiese explorar las carpetas contenedoras de archivos e imprima en la ventana emergente la imagen del texto seleccionada. Dentro de la imagen, se seleccionaron cuatro puntos; Estos se seleccionarían para extraer el párrafo a trabajar. | A pesar de múltiples intentos, el posicionamiento del *canvas* nunca coincidía espacialmente con la imagen y la selección de los cuatro puntos no se podía realizar sobre la imagen. Fue necesario replantear el código para el siguiente proceso. |

* + - 1. Segundo intento de creación de GUI

|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Crear una GUI (*Interfaz Gráfica para el Usuario*, por sus siglas en inglés) que permitiese mostrar la imagen del folio y seleccionar cuatro puntos para obtener el párrafo a trabajar. | Teniendo seleccionada la librería con la cual se pensaba trabajar, el primer paso fue revisar el código realizado anteriormente. Luego de corregir el posicionamiento del “*canvas*”, se pudo determinar tanto el tamaño de la ventana emergente en relación con la imagen de prueba (folio 21v). Acto seguido, fue necesario revisar el proceso de división de renglones, puesto que la cantidad de renglones que se había dado como parámetro había sido cuatro, ya que son cuatro los renglones del párrafo de prueba. Se solucionó este problema al darle un comando de “*input*”, en el cual el usuario debe decir cuantas líneas componen el párrafo seleccionado.  Luego de esto, se pasó a función, junto con todo lo que se realizó en el proceso de automatización. | El resultado más importante es tener una GUI funcional, que permite seleccionar el párrafo que se va a transcribir. Por otro lado, en este proceso se determinó la necesidad de realizar una GUI que sea realmente amigable al usuario y que posea instrucciones claras para su uso. Así mismo, se cayó en cuenta de la necesidad de hacer portable el programa y que no dependa de Python para ser usado.  Así mismo, se estableció un orden específico de puntos para el funcionamiento correcto de la obtención de párrafo, siendo el primer punto el de la esquina superior izquierda; el segundo punto se selecciona en la esquina inferior izquierda; el tercer punto en la esquina inferior derecha y el último punto en la esquina superior derecha.  (Ver fig. 10). |

Fig. 10. GUI final con RM-149 folio 21v cargado.



Fuente: elaboración propia.

* + 1. **Implementación de la Inteligencia Artificial.**
       1. Creación del banco de aprendizaje

|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Crear un banco de imágenes como muestra para el aprendizaje de la máquina. | Se seleccionaron los grafemas más comunes dentro del texto: las letras de la “a” a la zeta, los números y las abreviaturas “-*ae*”, “-*am*”, “-*em*”, “*et*”, “-*tur*” y “-*um*”. Luego, de páginas aleatorias dentro del texto se recortaron digitalmente una selección de treinta de los grafemas mejor escritos. El paso siguiente fue umbralizar y binarizar la selección. Esto con el objetivo de dejar el banco de entrenamiento lo más similar al texto tratado. | Se seleccionaron mil doscientos ochenta archivos. Uno de los problemas mayores fue la poca aparición de la letra zeta; por este motivo se tomaron únicamente quince muestras para ese grafema.  Para poder entrenar cualquier tipo de inteligencia artificial, es necesario poseer un banco de aprendizaje, así que este punto es vital para cualquier forma de inteligencia artificial. |

* + - 1. Implementación del “*Scale-Invariant Feature Transform*”(SIFT)

|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Implementar el SIFT en el programa y lograr que los puntos de interés que genera se puedan aplicar a las imágenes a trabajar. | En primer lugar, se instaló la versión de la librería OpenCV necesaria para tener el SIFT. Luego de múltiples intentos infructuosos, se logró instalar efectivamente la librería. Acto seguido, se realizó una prueba con una de las imágenes del banco de aprendizaje. Luego de corroborar que el SIFT funcionaba con la imagen de prueba, se realizó por todas las imágenes de entrenamiento y se intentó comparar una “a” de la imagen del trabajo. | Los resultados de comparar el banco de aprendizaje con un fragmento de la línea 3 fueron negativos, la imagen no estaba siendo reconocida. Por este motivo, se planteó usar un organizador que permitiera distribuir los puntos en un espacio de las n-dimensiones que corresponde al tamaño del descriptor de cada uno de los puntos de interés. |

* + - 1. Implementación del “*K-means”* (K-medias)

|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Implementar el K-medias para organizar los puntos de interés generados por el SIFT. | Se tomó todos los descriptores de todos los puntos de interés de cada uno de los caracteres del banco de entrenamiento.  Se crearon una cantidad de clústeresigual a la multiplicación de la cantidad de grupos, en el presente caso cuarenta y tres, por diez. El programa agrupa los puntos encontrando el valor medio como característica de dicho grupo. Posteriormente, se realizó un histograma de cuatrocientas treinta agrupaciones. | Teniendo estos resultados, el paso a seguir es alimentar a la inteligencia artificial. Esto será discutido a continuación |

* + - 1. Implementación del *Multi-Layer Perceptron* (MLP)

|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Implementar el MLP y que este tome los resultados ordenados por el K-medias en forma de histograma para analizar las imágenes de trabajo. | El MLP toma como entrada el histograma generado por el K-medias, acto seguido, se determina el tamaño de la red del clasificador (MLP), cuya salida es un vector de tamaño cuarenta y tres donde se distribuyen los resultados, enmarcados en una probabilidad.  Para poder generar estos resultados, se creó un diccionario en donde se les asigna un valor alfabético a las diferentes clases que corresponden a cada carácter. | Gracias a los resultados del MLP, se logró obtener un 43% de exactitud en el proceso de transcripción, tomando los cinco resultados más altos de la segunda prueba. Es importante tener en cuenta que la red usada es una red no retroalimentada, lo cual implica un detrimento a su capacidad de aprendizaje. Esto fue resultado de una constricción temporal que se resolverá en futuras iteraciones del programa. Para mayor claridad en cuanto a los resultados y conclusiones, posteriormente se hablará a profundidad de los retos y logros obtenidos. |

* + 1. **Automatización.**
       1. Primer proceso de automatización

|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Automatizar el trabajo realizado en tanto a la umbralización y división de líneas, además de crear las carpetas necesarias para almacenar los archivos resultantes. | Para comenzar este proceso, se realizó una función de división de líneas. Esta función permite tomar las líneas del párrafo de prueba (primer párrafo del folio 21v del manuscrito) y exportar esa selección en archivos por aparte. Para poder alojar estos archivos, se realizó otra función que crea las carpetas contenedoras de los archivos. | Este primer proceso permitió aprender qué dirección se tomaría en tanto a la automatización, siendo la decisión hacer bloques de funciones con un objetivo similar; es decir, todas aquellas funciones relacionadas con la umbralización irían juntas; así con todos los procesos realizados anteriormente. |

* + - 1. Segundo proceso de automatización

|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Convertir todos los procesos realizados en funciones y que corrieran en bloque. | Primero, se creó la función de umbralización, luego se realizó la función de ubicación de renglones. El siguiente bloque fue la separación de palabras.  Cada una de las funciones fueron revisadas y se limpiaron de código basura y pequeños errores. | Gracias a los avances obtenidos, se tomó la decisión de tener cada una de las funciones en un archivo separado. Esto hace que el arranque del programa sea más rápido que si no estuviese dividido de esta manera. |

* + - 1. Tercer proceso de automatización

|  |  |  |
| --- | --- | --- |
| Problema | Proceso | Resultado |
| Automatizar las funciones de la inteligencia artificial y la GUI. Imprimir los resultados finales. | Luego de convertir el código, tanto de la GUI como de la inteligencia artificial, se creó una función que permite imprimir en las imágenes de carácter resultantes los cinco valores más probables del MLP. | Este proceso final permitió tener todas las imágenes con los resultados del MLP. Esto permite tener de una manera organizada el producto, siendo más amigable para el usuario. |

* 1. **Explicación de los archivos constitutivos del programa.**

Conforme a lo anotado anteriormente, en esta sección se explica parte por parte el código que compone el programa, bautizado Perséfone. Se le dio este nombre en honor a la reina del inframundo griego. Según la mitología griega, Perséfone era la hija de Deméter, diosa de la agricultura. Luego del rapto de la joven diosa por parte de Hades, Perséfone se vuelve la reina del inframundo con una condición: volver al mundo de los mortales durante la mitad del año, para así traer de vuelta a la vida el mundo. De manera similar, el programa trae de nuevo a la luz el texto escogido.

Como se observa a continuación, el programa está separado en diferentes archivos, cada subíndice corresponde a un archivo. Se sugiere consultar el glosario debido a que se utilizará lenguaje especializado en programación. Cabe aclarar que las funciones, las variables y el título del archivo están en inglés, esto se determinó con el fin de permitir una mayor universalidad en la lectura del programa.

En primer lugar, todos los archivos importan bibliotecas y módulos. Las bibliotecas son funciones que son instaladas en Python y permiten realizar múltiples tareas; los módulos son funciones nativas de Python. Dentro del programa se importaron las siguientes bibliotecas y módulos. Cabe aclarar que TKinter, aunque sea un paquete, es decir una serie de módulos de Python, se agrupará con los módulos:

* Módulos:
  + glob: este módulo tiene la utilidad de encontrar la dirección de un archivo o carpeta contenedora en un sistema basado en Unix, tal como Windows.[[31]](#footnote-31)
  + os: este módulo permite utilizar las funciones propias del sistema operativo en el ambiente de Python.[[32]](#footnote-32)
  + shutil: este módulo sirve para copiar y pegar archivos en diferentes carpetas, así como moverlos sin necesidad de copiarlos. También puede ser utilizado en la eliminación de archivos de una carpeta.[[33]](#footnote-33)
  + tkinter: paquete que permite crear interfaces gráficas para usuarios (GUI), permitiendo crear ventanas emergentes para diferenciar el programa del ambiente de desarrollo (IDE).[[34]](#footnote-34)
* Bibliotecas:
  + NumPy: Numeric Python, abreviado NumPy, es una biblioteca dedicada a la creación y operación de arreglos, los cuales se pueden entender como vectores o matrices. Esta biblioteca también permite generar números aleatorios cuando es importada su función .random.[[35]](#footnote-35)
  + CV2: OpenCV, abreviación de “Open Source Vision Library”, es una biblioteca especializada en visión de máquina; es decir, problemas relacionados a tratamiento de imagen y video. El algoritmo SIFT es un tipo de función que está dentro de esta biblioteca.[[36]](#footnote-36)
  + Matplotlib: esta biblioteca permite la creación y muestra de gráficos, ya bien sea curvas, histogramas o similares; o bien imágenes importadas, creadas o modificadas por el programa.[[37]](#footnote-37)
  + Sklearn: forma corta de “scikit-learn”, es la biblioteca que permite implementar la inteligencia artificial. En el programa se utiliza “KMeans” y “MLP”.[[38]](#footnote-38)
  + PIL: Abreviatura de “Python Imaging Library” Permite procesar imágenes en el interpretador de Python, además de tener compatibilidad con tkinter.[[39]](#footnote-39)

Antes de continuar con los archivos que componen el programa Perséfone, es necesario explicar las funciones que se importan junto a las bibliotecas y módulos. Como se comentó en el proceso de automatización, el programa utiliza funciones para mejorar su rendimiento. Por esta razón, al principio de la explicación de cada archivo se presentarán, las funciones, las bibliotecas y módulos usados. Asimismo, debido a que cada función hace referencia a su archivo correspondiente, cualquier duda se resuelve dirigiéndose al subíndice correspondiente.

* + 1. *GUI.py*
       1. Bibliotecas y Módulos

tkinter.

CV2.

NumPy.

PIL.

* + - 1. Funciones usadas

*main*.

*startupFunctions*.

* + - 1. Explicación

En este archivo se genera la interfaz gráfica del usuario. En primer lugar, se genera una ventana emergente de los directorios del ordenador. Luego de seleccionar una imagen que cumpla el tipo de archivo requerido, una imagen en formato .TIF, se crea la ventana de la interfaz. Esta ventana está compuesta por un “canvas” en el cual está montada la imagen; este canvas permite seleccionar los puntos de la imagen, con estos puntos se recorta el párrafo a trabajar. Acto seguido, la ventana emergente con la imagen cargada se cierra, para que el programa le pide al usuario que determine el número de renglones presentes en el párrafo. Esto pone en marcha el resto del programa, llamando a la función *main*.

* + 1. *startupFunctions.py*:
       1. Bibliotecas y Módulos:

os.

shutil.

* + - 1. Explicación:

En este archivo se formulan las funciones de generación de las carpetas contenedoras de los archivos, así como la limpieza de estas mismas si ya tenían contenido. La eliminación de los archivos existentes en la carpeta es muy importante, puesto que puede crear contaminación de los resultados al tener muestras de ejecuciones pasadas.

* + 1. *thresholdFunctions.py*:
       1. Bibliotecas y Módulos:

CV2.

NumPy.

* + - 1. Explicación:

En este archivo está la función de umbralización. La umbralización es el proceso en el cual se le asigna un umbral numérico al color de un píxel. Esto debe ser realizado en una imagen en escala de grises. El programa lee las imágenes .TIF, que están en color natural, en esta escala. Se utilizan dos tipos de umbralización: Umbralización Gaussiana y Otsu. Se utilizan para lograr limpiar la imagen del ruido de la imagen original.

Dentro de este mismo archivo se binariza la imagen. Es decir, que a los valores de los píxeles de la imagen se les asignan valores absolutos: para todos los valores menores que el límite establecido en la umbralización se les asigna 0, lo que lo haría igual a todo aquello mayor al umbral se les asigna 1, siendo el valor de 0 el color negro y 1 blanco.[[40]](#footnote-40)

* + 1. *lineFunctions.py*:
       1. Bibliotecas y Módulos:

os.

CV2.

NumPy.

* + - 1. Explicación:

En este archivo se crean las funciones de división de renglones. Dado el número de renglones por parte del usuario, se detectan los puntos de mayor concentración de píxeles blancos en el eje horizontal; esto se puede obtener gracias a los resultados del proceso de umbralización. Teniendo estas coordenadas, se divide el párrafo ingresado por la cantidad de líneas que el usuario introdujo. Luego, se nombran estas líneas resultantes, lo cual se realiza para que sea más fácil realizar los próximos pasos.

* + 1. *wordFunctions.py*:
       1. Bibliotecas y Módulos:

os.

CV2.

NumPy.

* + - 1. Explicación:

En este archivo se formulan las funciones de división de palabra. Primero, sobre los renglones resultante de *lineFunctions*, se realiza el análisis amplio (wide) y el análisis angosto (close). El primero tiene en cuenta todo el alto del renglón, incluyendo astas y caídos, mientras que el segundo toma únicamente el centro del renglón, donde está la mayor concentración de píxeles negros. Estos dos análisis se realizan trazando cinco líneas separadas igualmente, con el objetivo de buscar las intersecciones de píxeles blancos. Después de comparar los resultados, se dividen las secciones que el programa identificó como palabras; acto seguido, se les asigna un nombre a los archivos resultantes.

* + 1. *charFunctions.py*:
       1. Bibliotecas y Módulos:

os.

CV2.

NumPy.

* + - 1. Funciones usadas:

*startupFunctions*.

* + - 1. Explicación:

En este archivo están las funciones de división de caracteres. Primero, se selecciona la imagen y se rota teniendo en cuenta el ángulo de la imagen para poder trazar las líneas de división con mayor precisión. Cabe aclarar que para las palabras muy largas y cortas aplicar este sistema resultó arduo, puesto que los ángulos resultantes eran muy bajos o altos. Por ello, para todos los ángulos inferiores a 15° se fijaron en 15°, de igual forma, todos los ángulos superiores a 25° se tomaron como 25°.

Luego se determinan los límites superiores e inferiores de la imagen, así como los límites horizontales de la misma para poder volver a estas medidas cuando la imagen sea recortada. Acto seguido, se realizan los análisis amplio y angosto, de la misma manera que se hizo para la separación de palabras en *wordFunctions*.

Después de retornar la imagen a su tamaño original, ya que en el proceso de rotación se alarga la imagen para tener mejores resultados, se generan secciones resultantes de la división de palabras; en estas secciones se guardan los caracteres procedentes de la operación. Por último, los caracteres se guardan en una carpeta aparte y se les asigna un nombre.

* + 1. *main.py*.
       1. Bibliotecas y módulos usados:

os.

CV2.

NumPy.

* + - 1. Funciones usadas:

charFunctions.

lineFunctions.

startupFunctions.

thresholdFunctions

wordFunctions.

* + - 1. Explicación general

Este archivo llama las otras funciones anteriormente descritas. Primero, llama las funciones de creación y limpieza de carpetas de *startupFunctions*, de este modo, se tiene donde guardar los archivos resultantes de las operaciones, si es el caso de creación; si es el caso de limpieza, se realiza para no se generar contaminación de los resultados. Después, llama las funciones creadas en *thresholdFunctions*, para umbralizar y binarizar el párrafo seleccionado en la función *GUI*. Enseguida, se exportan las líneas resultantes del proceso de la función *lineFunctions*, para luego iniciar las funciones de división de palabras por medio de la función *wordFunctions*. Por último, inicia las funciones de *charFunctions*, para obtener la división de caracteres.

* + 1. *SIFTnK.py*
       1. Bibliotecas y Módulos

glob.

os.

CV2.

NumPy.

Matplotlib.

* + - 1. Funciones usadas

*initializeAI*.

*startupFunctions*.

* + - 1. Explicación

En este archivo se realiza la aplicación del SIFT. Primero, limpia la carpeta de resultados, para no tener contaminación con resultados previos. Luego, toma las imágenes creadas por las funciones anteriormente mencionadas y, por medio del SIFT son descritas cada una de las imágenes. Después, por medio del K-Means son agrupados los descriptores de los puntos de interés. Acto seguido, la inteligencia artificial compara los descriptores de los puntos de interés de las imágenes de prueba con los generados por el material de entrenamiento. A cada caracter encontrado le asigna las cinco probabilidades más altas, para luego darle nombre y guardarlas en formato .png.

* + 1. *initializeAI.py*
       1. Bibliotecas y Módulos

glob.

os.

CV2.

sklearn.

* + - 1. Funciones usadas

*dictTranslate*.

* + - 1. Explicación:

En este archivo se inicia la inteligencia artificial. Primero se utiliza SIFT para que pueda detectar los puntos de interés del material de entrenamiento. Luego, agrupa los descriptores de estos puntos de interés por medio del K-Means. Para que sea comprensible por el Multi-Layer Perceptron, se transforman los valores de cada uno de los descriptores en histogramas. Luego de iterar, la inteligencia arroja las predicciones que son traducidas de las clases a los valores de los grafemas.

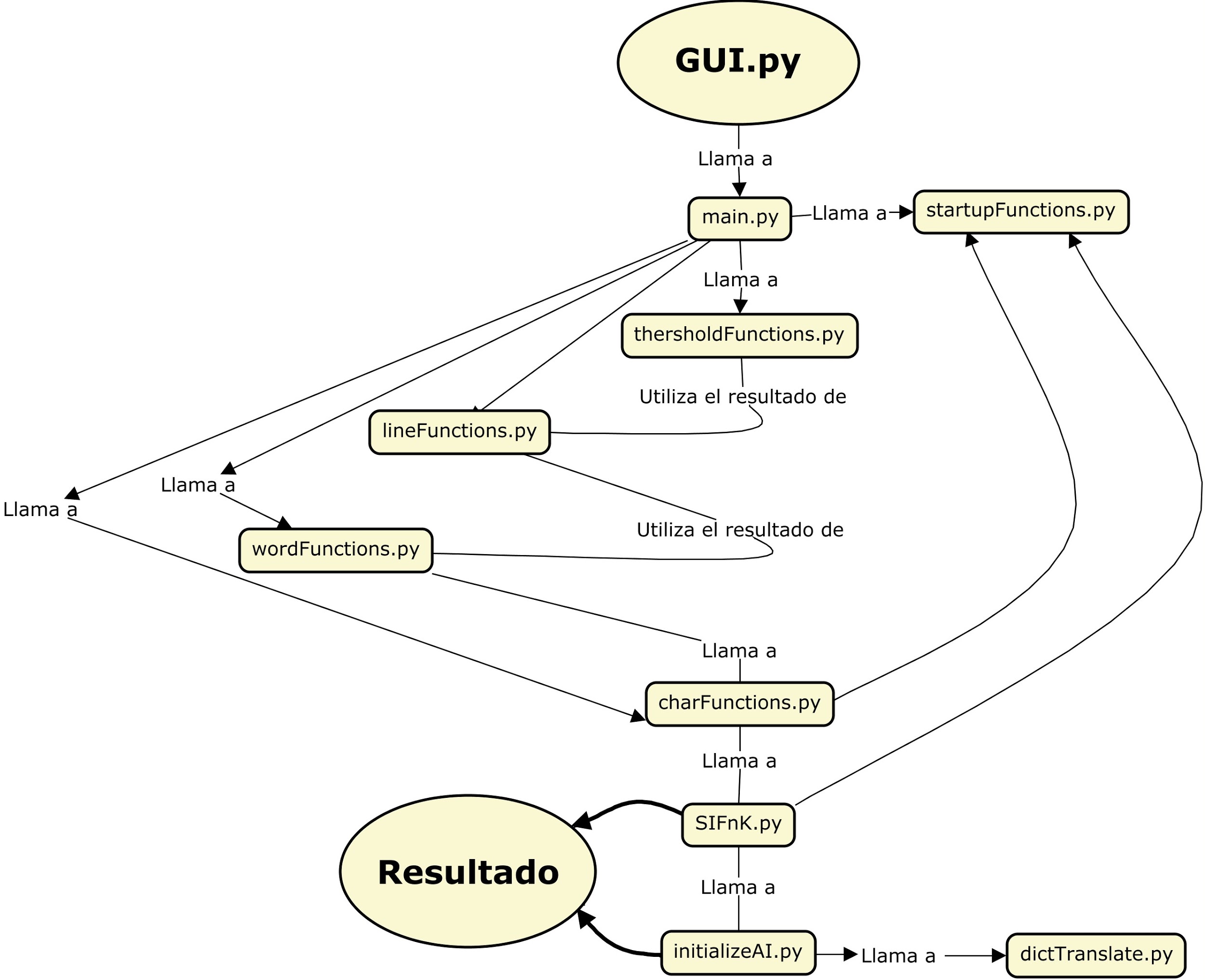
* + 1. *dictTranslate.py*

Este archivo es especial, puesto que es una sola función la que traduce las clases originadas por los histogramas generados a los valores de los grafemas alimentados.

Viendo como fue realizado el programa y los componentes que lo constituyen, es momento de revisar el desempeño de Perséfone.

El siguiente flujograma muestra cómo los archivos interactúan entre ellos.

Fig. 11. Flujograma del programa Perséfone.



Fuente: elaboración propia.

1. **Resultados**.
   1. **Generalidades del proceso de aplicación de las pruebas.**

En este apartado se consignarán los resultados de cada una de las pruebas, siete en total. Las pruebas se realizaron con el primer párrafo del folio 21v del documento, puesto que, como se ha mencionado anteriormente, ha sido el objeto de experimentación durante todo el proceso de creación del programa. Asimismo, se dará un resultado final, el cual está compuesto por el promedio y la desviación estándar, producto del proceso de análisis de las pruebas parciales.

Otro punto importante para explicar es la cantidad de valores dados como resultado de las pruebas parciales. En la primera prueba solamente se tomaron los tres valores más altos que el programa daba como posible significado del grafema. Aunque parecía un número óptimo al principio, debido al bajo resultado de esta prueba, se tomaron los cinco primeros resultados, con el objetivo de ampliar las posibilidades de acierto.

Cabe aclarar que la operación anteriormente mencionada no se efectuó para subir artificialmente la exactitud de los resultados; se realizó porque en varias ocasiones el resultado correcto se encontraba entre la cuarta y quinta opción. Debido al funcionamiento del SIFT, muchos grafemas son tomados como otro, un ejemplo de ello es el grafema “q” que sí se rota especularmente es similar al grafema “p”; sí este es rotado en el eje horizontal queda como “d” y si este es rotado especularmente es parecido a “b”.

Por otro lado, es importante resaltar el problema de la selección de palabras por parte del programa. Debido a la forma en que está realizada la división de palabra y de caracter, como fue explicado anteriormente en la sección epónima, en múltiples ocasiones el programa se salta ciertas palabras; este salto es arbitrario y afecta tanto la numeración como el orden mismo de los caracteres en la carpeta; esta situación hace complicada la determinación de los resultados. Por este motivo, uno de los hallazgos es la necesidad de tener el párrafo procesado a la mano para poder dilucidar cuál de los resultados es el indicado para cada uno de los caracteres, esto se explicará a fondo en la sección de conclusiones.

* 1. **Descripción de las pruebas parciales.**

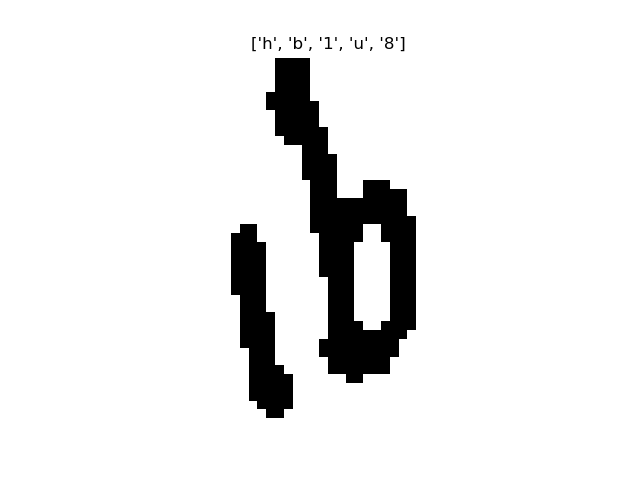
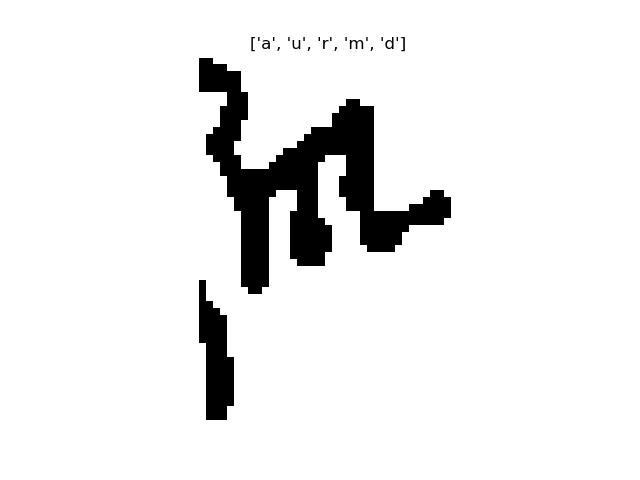
Es clave tener en cuenta el diseño de las tablas de los resultados de las pruebas parciales. La primera columna muestra el número del caracter resultante; la segunda columna contiene los cinco resultados que el programa dio a cada grafema; la tercera columna ilustra el grafema correcto entre los cinco posibles. En la columna de los resultados, se utilizó tres colores para distinguir si el resultado fue correcto, parcial o errado. El color verde corresponde a un resultado correcto, el amarillo a un resultado parcial y el rojo a un error.

Asimismo, una cuestión relevante por clarificar es el tópico de un resultado parcial, ya que sucede con frecuencia en todas las pruebas. Debido a la separación de caracteres, en diversos momentos se puede ver más de un grafema, ya bien sean completos o incompletos, en la imagen resultante. Cuando el programa logra identificar algo dentro de la imagen, normalmente es una sola letra. Solo se han atestiguado dos situaciones en la cual el programa logra identificar ambos caracteres. Los casos concretos son los números 49 y 94 de la sexta prueba (ver fig. 12). La imagen 49 es un fragmento de “fra”, que significa “*forma*”; el programa solamente toma de la imagen las letras “r” y “a”. Asimismo, en la imagen 94 se pueden ver una parte del grafema de la “u”, seguida por la “b”. Sin embargo, debido a que en cada caso se puede tomar cada imagen como una entidad única, se ha decidido tomar este resultado como un único punto dentro del sistema de clasificación, siendo ambos resultados medios puntos.

El sistema de clasificación se basa en asignar un punto por cada acierto que el programa logra; medio punto (0,5) por cada acierto de una imagen de doble caracter y no se asigna punto al fallar. Conforme a lo anterior, se darán los resultados de cada prueba teniendo en cuenta la cantidad de caracteres identificados. De igual forma, en las tablas se presentarán los valores de los puntos totales, los puntos parciales y el porcentaje de precisión de los resultados.

Por otro lado, para sistematizar los resultados de cada tabla se realizaron una serie de tablas en las cuales están consignados la cantidad aciertos de cada grafema en las pruebas. Esta cantidad de aciertos es la sumatoria de aciertos “correctos” y los puntos medios, que en la tabla están nominalizados como aciertos “parciales”. Es importante tener en cuenta que en estas tablas solamente están consignados aquellos grafemas que aparecen en el texto original.

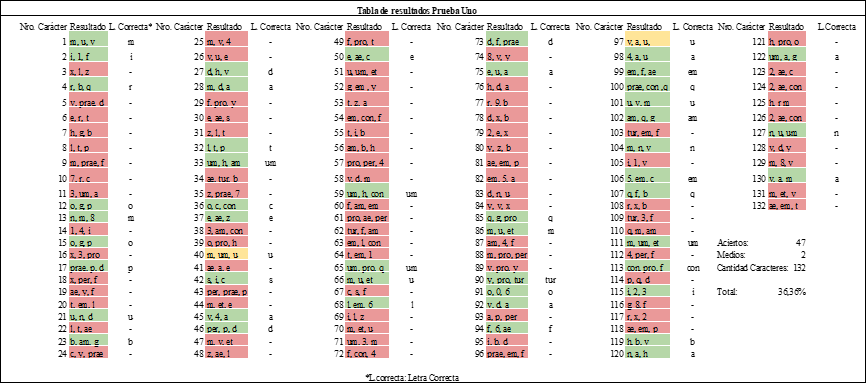
Fig. 12. Imágenes 49 y 94, Prueba 6.



Fuente: Elaboración Propia.

* + 1. **Prueba Uno.**

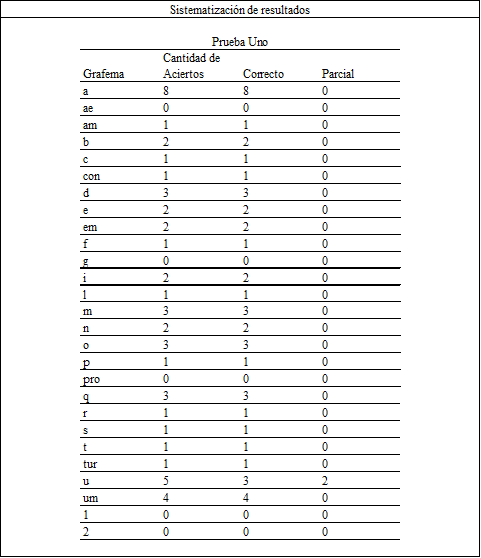
Tabla 1. Resultados de la prueba Uno.



Fuente: elaboración propia.

Se observa en la Prueba Uno que el programa *Perséfone* arrojó ciento treinta y dos caracteres, de los cuales se tuvo como aciertos cuarenta y siete grafemas debido al proceso de división de palabras y caracteres. Acertó en la transcripción de cincuenta y dos caracteres, además de dos resultados parciales. A su vez, se obtuvieron ochenta y tres resultados erróneos conforme a los problemas mencionados anteriormente. Tomando en cuenta la sumatoria de los aciertos y los puntos medios y dividido por la cantidad de caracteres presentes en la prueba, el promedio total de aciertos es del 36, 36% de precisión de transcripción de los caracteres del párrafo seleccionado.

Tabla 2. Sistematización de los resultados de la prueba Uno.

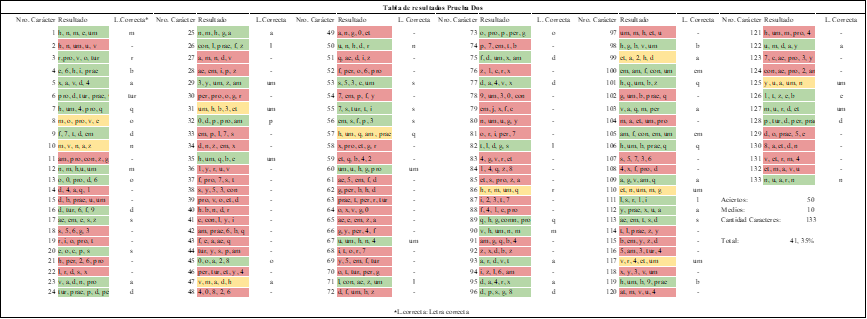


Fuente: elaboración propia.

Se observa que el programa Perséfone detectó y transcribió en la primera prueba cuarenta y nueve grafemas, cuarenta y siete de manera correcta y dos parcial. En esta prueba la letra “a” fue la más transcrita, con ocho veces, seguido de las letras “u”, “d”, “m”, “q” con rangos entre 3 y 5 veces de transcripción. Las letras “c”, “e”, “f”, “i”, “l”, “r, “s”, “t” fueron transcritas una y dos veces. La abreviatura um apareció 4 veces correctamente, las abreviaturas “con-”, “-em”, “-tur” fueron transcritas una vez. El programa no transcribió las abreviaturas “-ae” “pro-, las letras “g” y los números 1 y 2.

* + 1. **Prueba Dos.**

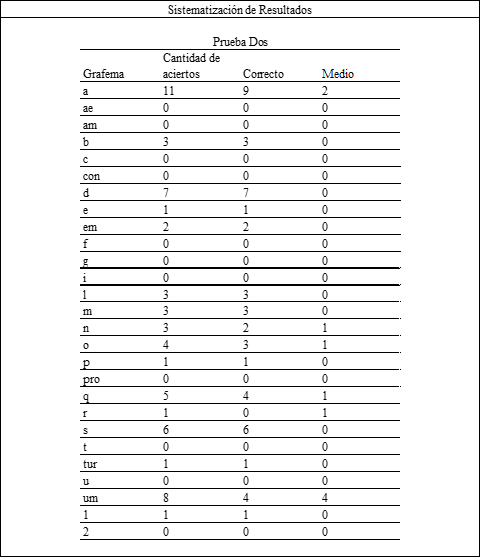
Tabla 3. Resultados de la prueba Dos.



Fuente: elaboración propia.

El programa *Perséfone* en la Prueba Dos del total de ciento treinta y tres caracteres detectados, tuvo cincuenta y dos aciertos. Se observó la detección de diez puntos medios y setenta y un errores en el proceso de transcripción. Como promedio final de la prueba se obtuvo un 42, 85% de exactitud en la transcripción.

Tabla 4. Sistematización de los resultados de la prueba Dos.

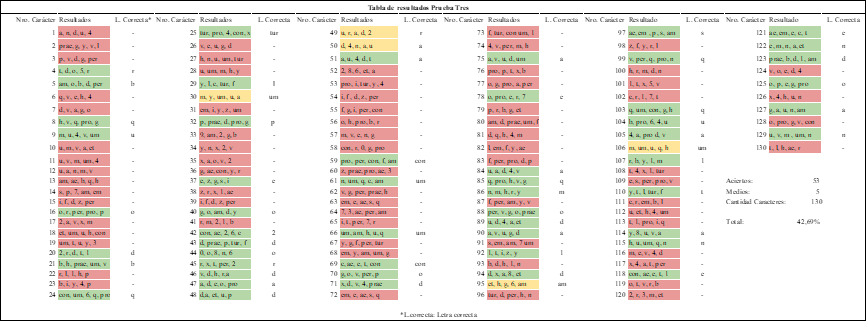


Fuente: elaboración propia.

En la prueba dos, el programa Perséfone arrojó once transcripciones de la letra “a”, nueve correctas y dos parciales, de la letra “d” siete detecciones y transcripción de manera exacta, seguido de la letra “s” con seis transcripciones correctas. Las letras “b”, “l”, “m”, “n”, “o”, “q” están en los rangos de transcripción de cinco a dos veces. La abreviatura con mayor detección fue “-um” con ocho veces, cuatro parciales y cuatro correctas, las abreviaturas “-tur” y “em-” se transcribieron una y dos veces respectivamente. Sólo se transcribieron una vez las letras “e”, “p”, “r” y el número 1. No se realizó detección y transcripción de las abreviaturas y letras “-ae”, “-am”, “c”, “con-”, “f”, “g”, “i”, “pro-”, “t” y el número 2.

* + 1. **Prueba Tres.**

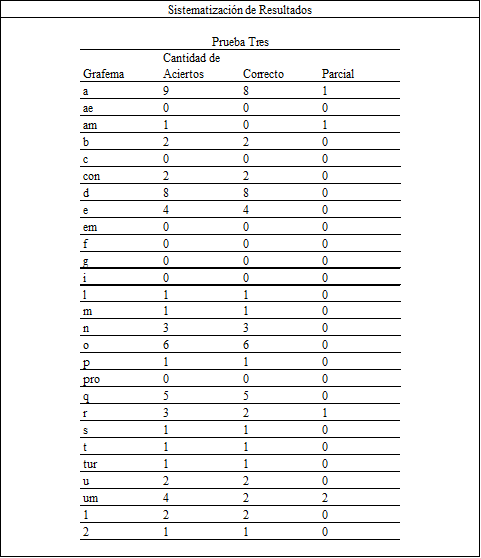
Tabla 5. Resultados de la prueba Tres.



Fuente: elaboración propia.

El programa *Perséfone* en la Prueba Tres realizó el proceso de transcripción de ciento treinta caracteres, acertó cincuenta y tres grafemas. Los puntos medios con un total de cinco, las fallas en la transcripción fueron setenta y dos. El resultado final de la Prueba Tres fue del 42, 69% de exactitud en la transcripción.

Tabla 6. Sistematización de los resultados de la prueba Tres.

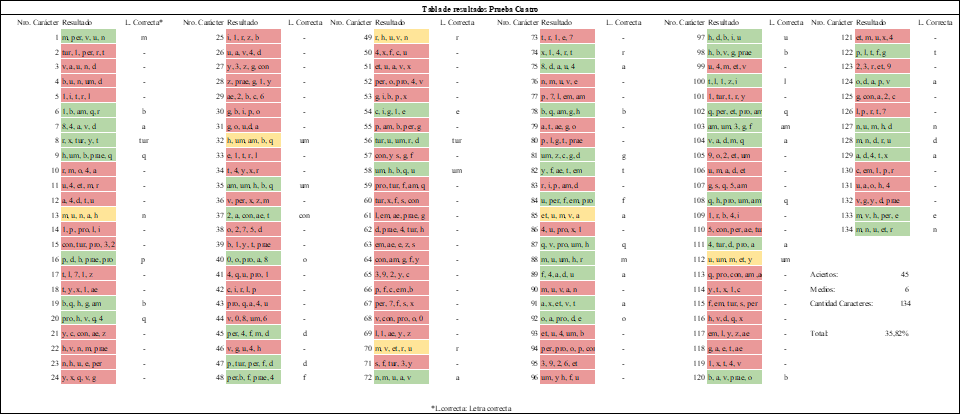


Fuente: elaboración propia.

El programa *Perséfone* del total de cincuenta y ocho transcripciones correctas y parciales, las letras “a” y “d” con un mayor número de transcripciones de nueve y ocho veces respectivamente. La abreviatura “-um” obtuvo cuatro transcripciones correctas. Con rangos entre dos y seis veces de transcripción se encontraron las letras y abreviaturas “b”, “con-”, “e”, “n”, “o”, “q”, “r”, “u” y el número 2. El programa sólo transcribió una vez la abreviaturas y letras “-am”, “l”, “m”, “p”, “s”, “t”, “-tur” y 2.

* + 1. **Prueba Cuatro.**

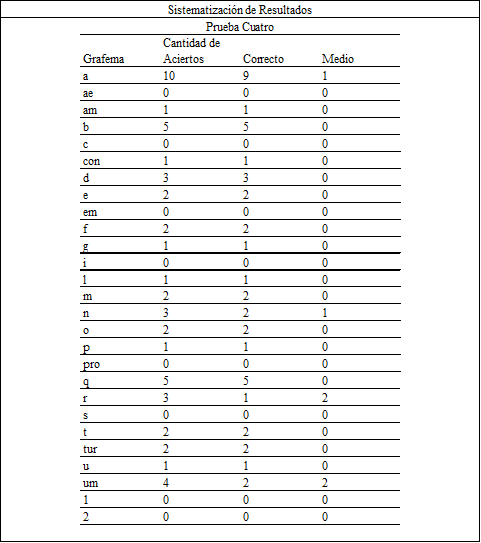
Tabla 7. Resultados de la prueba Cuatro.



Fuente: elaboración propia.

En la Prueba Cuatro, el programa *Perséfone* logró la detección y transcripción de ciento treinta y cuatro caracteres, acertados fueron cuarenta y cinco y los puntos medios fueron 6. Las fallas sumaron setenta y tres caracteres. El promedio ponderado de precisión en la transcripción en la Prueba Cuatro fue de 35, 82%.

Tabla 8. Sistematización de los resultados de la prueba Cuatro.

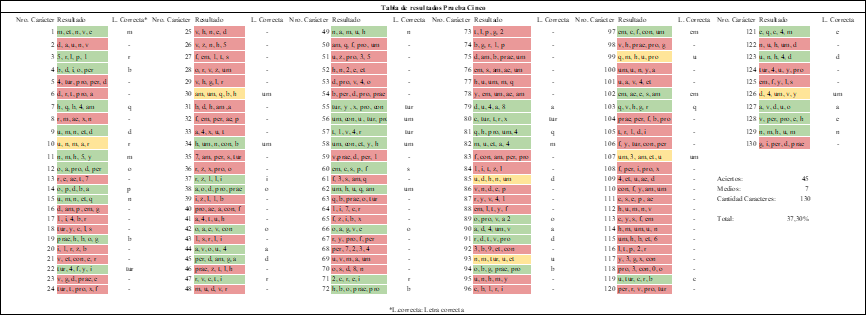


Fuente: elaboración propia.

En el proceso de sistematización de la Prueba Cuatro del Programa *Perséfone* se presentaron con mayor número de detecciones y transcripciones las letras “a” con diez veces y las letras “b” y “q” con cinco veces cada una de manera correcta. El número de veces de transcripción en esta prueba fue bajo con rangos entre cuatro y dos de las letras “d”, “e”, “f”, “m”, “n”, “o”, “r”, “t” y las abreviaturas “-tur” y “-um”. Sólo una vez se logró la detección y transcripción de “-am”, “con-”, “g”, “l”, “p”, “u”. No se obtuvo transcripción de “-ae”, “c”, “i”, “pro”, “s”, los números 1 y 2.

* + 1. **Prueba Cinco.**

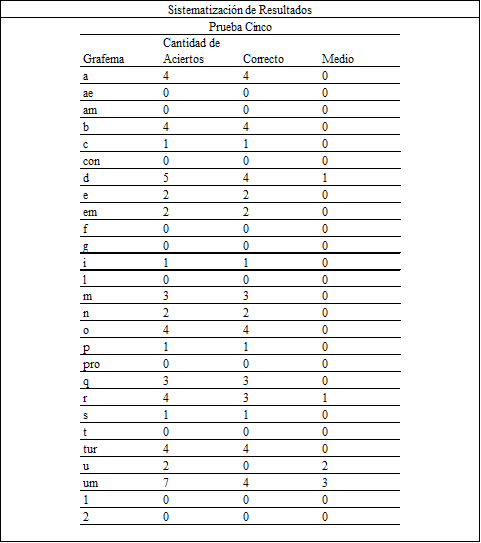
Tabla 9. Resultados de la prueba Cinco.



Fuente: elaboración propia.

Al aplicar la Prueba Cinco en el programa *Perséfone*, se observó un nivel bajo de aciertos con 45 caracteres transcritos de un total de ciento treinta caracteres detectados. Los puntos medios fueron siete y sesenta y ocho errores presentes en el resultado. El promedio de acierto en la transcripción fue de 37, 30%.

Tabla 10. Sistematización de los resultados de la prueba Cinco.

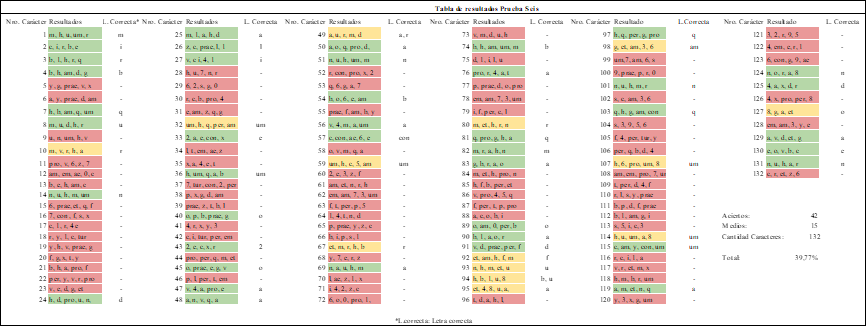


Fuente: elaboración propia.

Se observa en la sistematización de las detecciones y transcripciones de caracteres en la prueba cinco del programa *Perséfone* que se obtuvo un nivel parecido a la prueba cuatro con cuarenta y cinco aciertos en la transcripción correspondientes en un rango de siete veces la abreviatura “-um” y de cinco veces la letra “d”. Los demás caracteres están en el rango entre dos y cuatro veces de transcripción como las letras “a”, “b”, “e”, “m”, “n”, “o”, “q”, “r”, “u”, de igual forma las abreviaturas “-em” y “-tur”. Con cero transcripciones se observan “-ae”, “-am”, “con-”, “f”, “g”, “l”, “pro-”, “t”, los números 1 y 2.

* + 1. **Prueba Seis.**

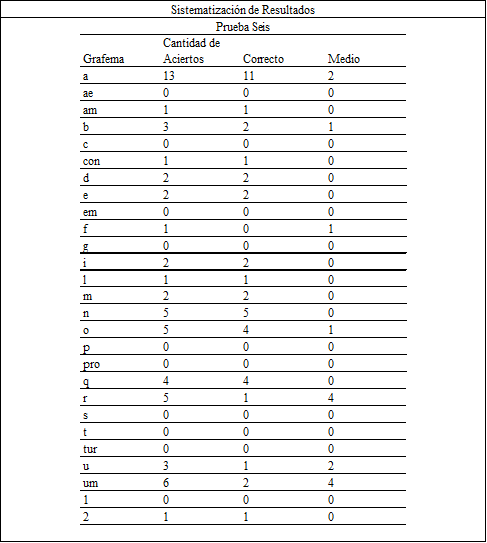
Tabla 11. Resultados de la prueba Seis.



Fuente: elaboración propia.

El programa *Perséfone* en la Prueba Seis arrojó una cantidad de ciento treinta y dos caracteres detectados y transcritos. Transcribió de manera acertada cuarenta y dos caracteres, de manera parcial quince caracteres y en forma incorrecta setenta y cinco caracteres. El promedio acertado de detección y transcripción del programa fue de 39, 77%.

Tabla 12. Sistematización de los resultados de la prueba



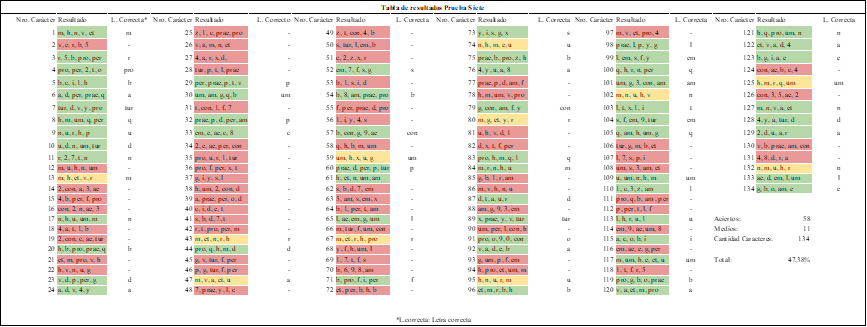
Seis. Fuente: elaboración propia.

Conforme a la sistematización de los resultados de la prueba seis del programa *Perséfone*

de las cincuenta y siete transcripciones correctas y parciales, se logró la transcripción de trece veces de la letra “a” y seis veces de la abreviatura “-um”. Entre los rangos de transcripción de cinco y dos veces se encontraron sólo letras correspondientes a “b”, “d”, “e”, “i”, “m”, “q”, “r” y “u”. En esta prueba sólo se transcribió una vez “-am”, “con-”, “f”, “l” y el número 2. No se obtuvo transcripción de las abreviaturas y letras “-ae”, “-em”, “p”, “pro-”, “s”, “t”, “-tur” y del número 1.

* + 1. **Prueba Siete.**

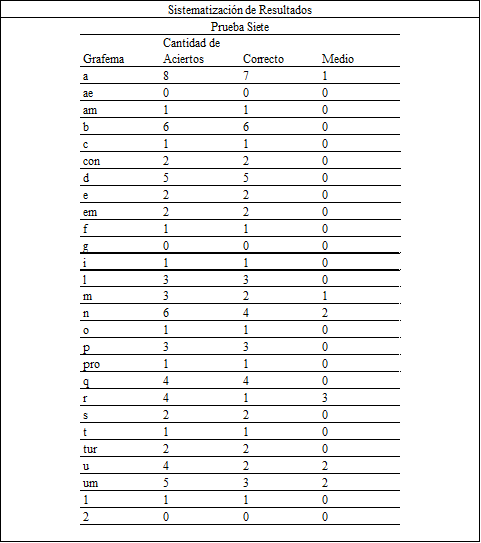
Tabla 13. Resultados de la prueba Siete.



Fuente: elaboración propia.

El programa *Perséfone* en la prueba Siete se logró la cantidad total de ciento treinta y cuatro caracteres, cincuenta y ocho que corresponde las detecciones y transcripciones acertadas, once transcritas parcialmente y arrojó sesenta y cinco errores en general. El promedio de exactitud de aciertos en la detección y transcripción de *Perséfone* en esta prueba fue de 47, 38%.

Tabla 14. Sistematización de los resultados de la prueba Siete.



Fuente: elaboración propia.

Los resultados de la sistematización del total de sesenta y nueve aciertos y medios en la detección y transcripción de caracteres por parte del programa *Perséfone* en la prueba Siete, se evidenciaron con mayor número de veces la letra “a” con ocho, las letras “b” y “n” con seis. Con rangos entre cinco y dos veces de detección y transcripción se encontraron las letras y abreviaturas “con-”, “d”, “e”, “l”, “m”, “p”, “q”, “r”, “u” y “-um”. Con una sola vez de transcripción se presentaron las abreviaturas “-am”, “pro-”, el número 1 y las letras “c”, “f”, “i”, “o” y “t”. Con resultado en cero en esta prueba se observó la abreviatura “-ae”, el número 2 y la letra “g”.

* 1. **Resultados Finales.**

A continuación, se presentarán los resultados finales del programa Perséfone conforme a la aplicación de las siete pruebas para la transcripción de letra manuscrita a digital del párrafo seleccionado del texto RM 149 folio 21v.

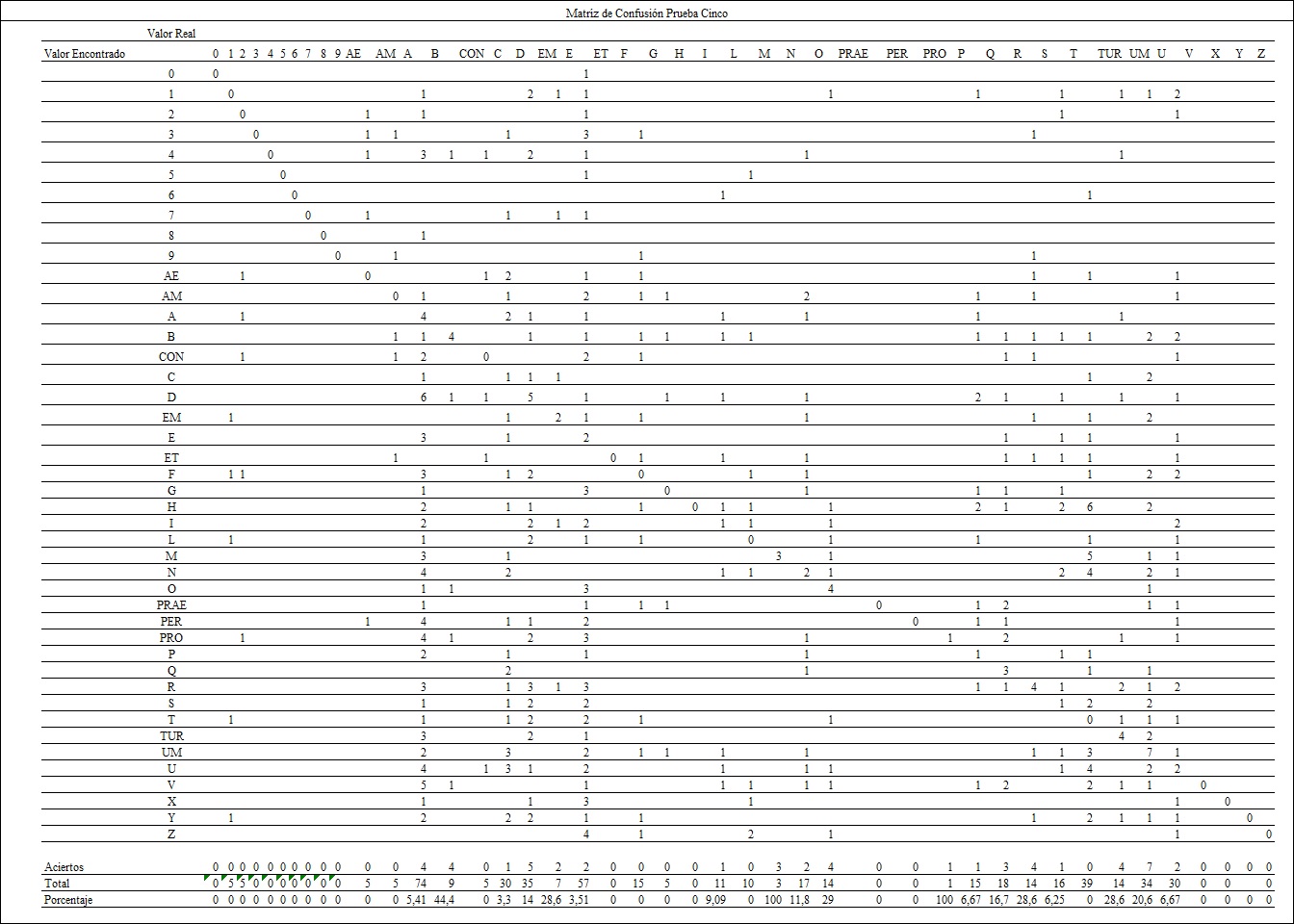
En la primera sección se presentan dos matrices de confusión[[41]](#footnote-41) con el fin de analizar la comparación entre la información real del texto y las predicciones realizadas por el programa Perséfone. Estas predicciones corresponden a los resultados establecidos en el proceso de sistematización del número de caracteres y de aciertos totales y parciales.

La segunda sección presenta los resultados de aciertos por grafema y posteriormente los resultados totales del proceso de transcripción del programa Perséfone.

Conforme a los valores resultantes del proceso del programa se organizaron unas matrices de confusión con los valores reales del texto y los valores encontrados en las pruebas cinco y siete. Es importante anotar que no todos los valores encontrados corresponden a los valores reales, por este motivo se tienen como ejemplo los valores de los números del “0” y del “3”, “4”, “5”, “6”, “7”, “8” y “9”, así como “v”, “x”, “y” y “z” no tienen representación en los valores reales pero sí se presentan en los valores encontrados por el proceso de clasificación del programa Perséfone. Cabe aclarar que una matriz de confusión busca comparar los resultados dados por la clasificación de la inteligencia artificial con los valores reales del texto. Esto se realiza con el objetivo de visualizar el desempeño del algoritmo de selección. Se escogieron estas pruebas puesto que representan el porcentaje más alto obtenido (la prueba siete) y un porcentaje cercano al promedio (prueba cinco), de esta manera se puede ver con mayor claridad cómo está el estado del programa y las posibilidades de mejora de este mismo.

* + 1. **Matriz de Confusión Prueba Cinco.**

Tabla 15. Matriz de confusión de la prueba Cinco.



Fuente: elaboración propia.

Se observa en la Matriz de Confusión de la Prueba Cinco conforme al algoritmo elaborado en el programa Perséfone para la detección y transcripción del texto, la distribución y comparación de los valores reales y valores encontrados, en la línea vertical se ubican los valores reales de los caracteres del texto seleccionado y en la línea horizontal los valores encontrados.

A manera de ejemplo la letra “a” de acuerdo con la comparación del valor real del texto “a” y del valor encontrado “a” resultante del proceso de Perséfone, tuvo 4 aciertos en total, lo que corresponde a un 4, 4% de exactitud del programa.

En comparación del valor real de la letra “a” con otros valores encontrados diferentes se observan los números “1”, “2” y “8” con una sola vez y el número “4” con tres veces. Las abreviaturas “-am” y “prae-” con una sola vez, “con-” y “-um”, dos veces, “-tur” con tres veces, “per-” y “pro-” cuatro veces.

Ahora bien, en comparación del valor real de la letra “a” con valores diferentes encontrados correspondientes a otras letras se encontraron, por ejemplo, “b”, “c”, “g”, “l”, “o, “s”, “t”, “x” con una sola vez de aparición. Como valor encontrado diferente al valor real “a”, también aparecen las letras “d” con seis veces, “v” con cinco veces, “n”, “u” con cuatro veces, “e”, “f”, “m”, “r” con tres veces, “h”, “i”, “p”, “y” con dos veces respectivamente. No hubo ninguna correspondencia en la comparación de valores en los números “0”, “5”, “6”, “7”, “9”; en las abreviaturas “-ae”, “-em”, “et” y en las letras “q” y “z”.

En consecuencia, se tienen como resultados finales de la comparación entre valores reales y valores encontrados de las letras ya sean de manera correcta o diferente como total de “a” setenta y cuatro veces, como aciertos cuatro para obtener un porcentaje de detección y transcripción correcta del 5,4%. En la letra “b” fueron en total nueve, aciertos cuatro para un porcentaje del 44%. En la letra “c” se obtuvo un acierto, un total de treinta detecciones de otros valores encontrados diferentes y como resultado de exactitud del 3,3%.

En la letra “d” se obtuvieron cinco aciertos, un total de valores encontrados diferentes de treinta y cinco, con un porcentaje del 14% de certeza en el resultado de detección y transcripción.

La letra “e” tuvo como total de detecciones cincuenta y siete, aciertos dos y un porcentaje de 3,51% de transcripción correcta. La letra “i” con un porcentaje de exactitud de detección y transcripción de 9,09% resultante de once detecciones y un acierto. La letra “m” obtuvo un porcentaje exacto de detección y transcripción del 100% porque tuvo tres detecciones y tres aciertos respectivamente. La letra “n” del total de diecisiete, tuvo dos aciertos para un 11,8% de exactitud. La letra “o” con un porcentaje del 29% de transcripción correcta, resultante del total de catorce, se acertaron cuatro veces. A su vez, “p” obtuvo un acierto de quince valores encontrados para un porcentaje de 6,67%. La letra q con un porcentaje de 16,7% del resultado de tres aciertos sobre dieciocho en total. La letra “r” obtuvo 28,6% del total de catorce valores encontrados sobre cuatro aciertos en la transcripción.

La letra “s” del total de dieciséis sobre un acierto obtuvo 6,25% de exactitud. La letra “u” con un porcentaje del 6,67% del resultado total de treinta sobre dos aciertos.

La abreviatura “-em” obtuvo un porcentaje de exactitud de 28,6% del total de siete valores encontrados sobre dos aciertos. La abreviatura “pro-” con un acierto sobre un valor encontrado para un 100% de exactitud. La abreviatura “-tur” obtuvo un porcentaje de 28,6% del total de catorce detecciones sobre cuatro aciertos. La abreviatura “-um” del total de treinta y cuatro sobre siete aciertos obtuvo un 20,6% de transcripción correcta.

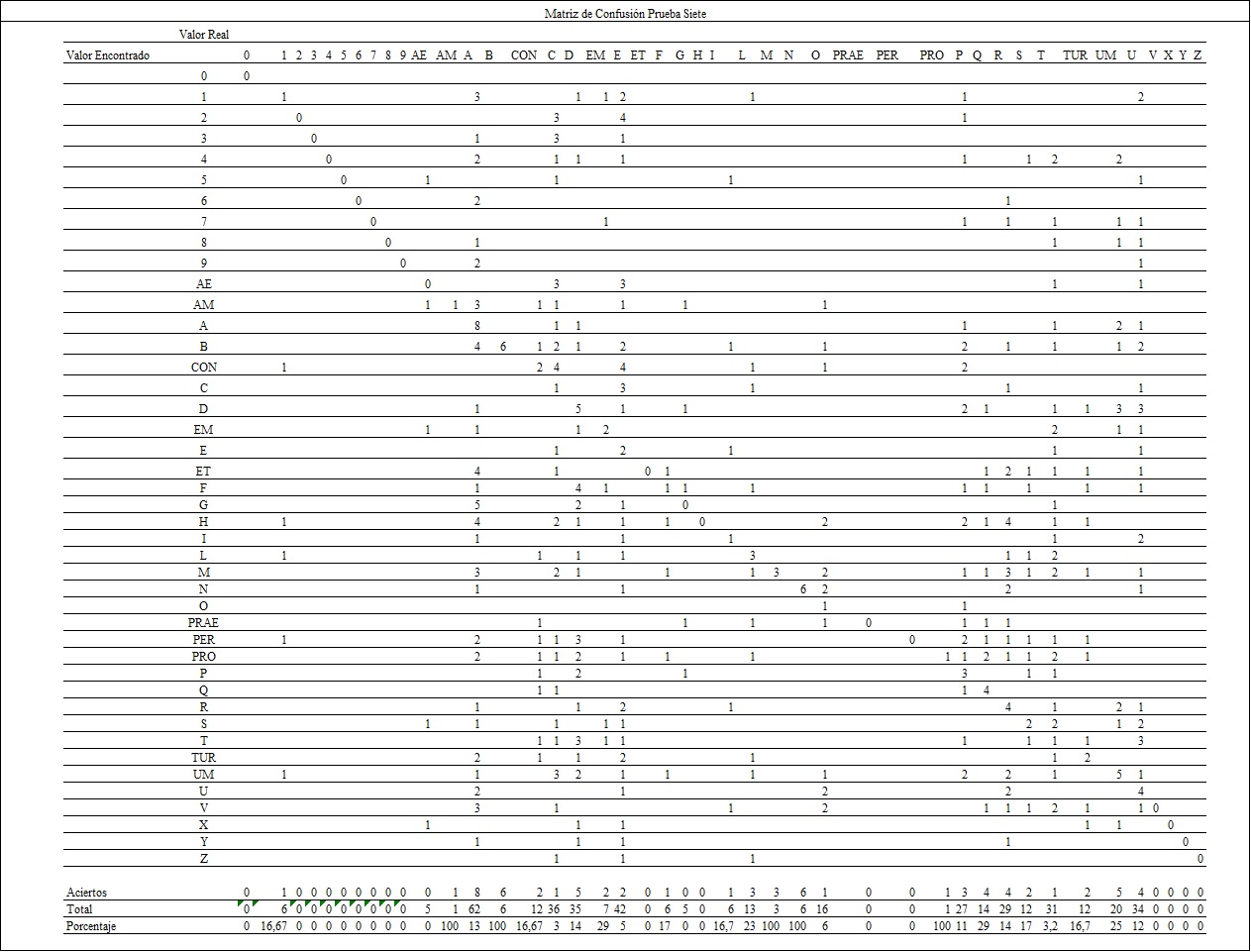
No se lograron valores encontrados acertados y/o diferentes de los números “0”, “3”, “4”, “5”, “6”, “7”, “8” y “9”. Los números “1” y “2” tuvieron cinco valores encontrados diferentes al real correspondientes a los números. Por ejemplo, el número “1” tuvo valores encontrados como “-em”, “f”, “l”, “t” “y”.

De igual forma, las abreviaturas “-ae”, “-am”, “con-” obtuvieron cinco veces en valores encontrados diferentes, “et”, “prae-”, “per-” no obtuvieron ningún resultado en el programa.

Asimismo, en las siguientes letras se observaron valores encontrados diferentes al real: “f" con quince por ejemplo en los números “3” y “9”, “g” con cinco por ejemplo en la letra “a” y en varias abreviaturas, la letra“l” con diez por ejemplo en el número “5”, “t” con treinta y nueve por ejemplo en el número siete, en las abreviaturas “-ae”,“-em”, “-um”, “et”; y las letras “h”, “v”, “x”, “y”, “z” con cero veces, es decir, sin resultado de transcripción y detección.

* + 1. **Matriz de Confusión Prueba Siete.**

Tabla.16. Matriz de confusión de la prueba Siete.



Fuente: elaboración propia.

Conforme a los resultados del algoritmo descrito en la Matriz de Confusión de la Prueba Siete se encontraron los siguientes hallazgos en los números:

No se encontraron aciertos ni valores diferentes en los números “0”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9” por tanto el resultado es cero en la transcripción y detección. El único número con valores encontrados fue el “1” con un acierto y cinco con valores diferentes como “con-”, “h”, “l”, “per-”, “-um”.

Los resultados de las abreviaturas “-ae” arrojó cinco valores diferentes al real, es decir, no tuvo aciertos, “-am” al igual que “pro-” lograron un acierto en la transcripción del valor real para un porcentaje del 100%. La abreviatura “-em” de un total de siete valores, arrojó dos aciertos para un porcentaje de transcripción de 29%, seguido de “-um” con un total de veinte valores encontrados, dos aciertos o valores reales y quince valores diferentes para un porcentaje del 25% y “-tur” con un porcentaje de transcripción de 16,7% resultante del total de diecisiete valores encontrados y dos aciertos.

Las abreviaturas “et”, “prae-” y “per-” arrojaron resultados de cero tanto en los valores encontrados como en los aciertos.

Se observan los resultados de detección y transcripción de *Perséfone* de las letras y los porcentajes conforme a la aplicación de la matriz de confusión: el caracter “a” con un mayor número de valores encontrados de sesenta y dos valores en total, obtuvo ocho aciertos para un 13% de transcripción. La letra “b” arrojó seis valores reales y encontrados con un porcentaje del 100%, la letra “c” arrojó treinta y seis en total de valores encontrados y un solo acierto para un 2,8%.

La letra “d” del total de treinta y cinco valores encontrados, cinco aciertos para un 14% de transcripción. El caracter “e” del total de cuarenta y dos valores encontrados, arrojó dos aciertos con un 5%. La letra “f” con un porcentaje del 16,6% resultante de un acierto sobre seis en total de valores encontrados. En el grafema “i” se observó un acierto del total de seis valores encontrados y para un porcentaje de 16,7% de transcripción. Sobre la letra “l” se logró un porcentaje del 23% proveniente del total de trece valores encontrados y tres aciertos.

Los caracteres “m” y “n” obtuvieron 100% en los resultados de detección y transcripción conforme a la correspondencia del número de valores encontrados y aciertos de tres y seis respectivamente.

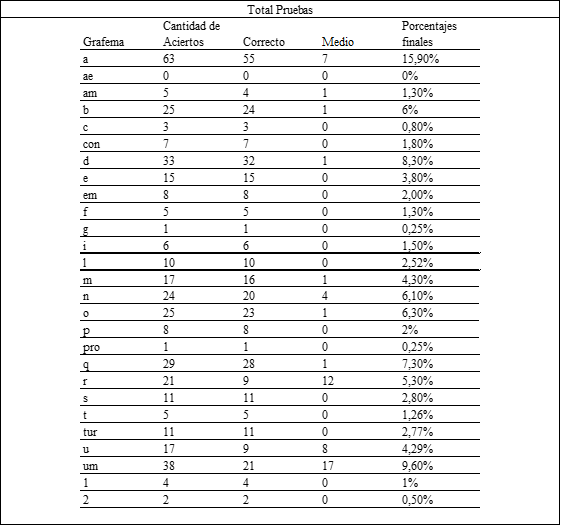
La letra “o” con un porcentaje de 6% como consecuencia de obtener dieciséis valores encontrados y un acierto. La letra “p” con tres aciertos de veintisiete valores encontrados y un 11% de transcripción correcta. El grafema “q” con un 29% de transcripción correcta resultante del total de catorce valores encontrados y cuatro aciertos. El caracter “r” obtuvo 14% de cuatro aciertos sobre veintinueve valores encontrados. La letra “s” con un resultado del 17% de transcripción correcta conforme a dos aciertos y un total de doce valores encontrados. El grafema “t” del total de treinta y un valores encontrados, arrojó un acierto que corresponde a 3,2%.

En el caracter “u” se logró 12% de transcripción correcta como resultado de treinta y cuatro valores encontrados y cuatro aciertos.

A manera de ejemplo, el caracter “g” arrojó cinco valores encontrados diferentes al real como “-am”, “d”, “f”, “prae-”, “p”.

Sobre los grafemas “h”, “v”, “x”, “y”, “z” el programa no realizó el proceso de detección y transcripción por tanto sus resultados se observan en cero.

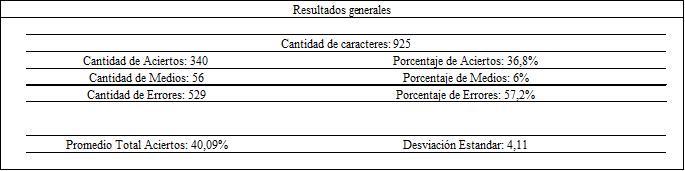
Tabla.17. Total de aciertos de todas las pruebas.



Fuente: elaboración propia.

Se observa en la presente tabla, los resultados totales y porcentajes de los aciertos y puntos medios en el proceso de detección y transcripción del programa *Perséfone.* Los caracteres con mayores porcentajes de transcripción son “a” con 15,90%, “-um” con 9,60%, “d” igual a 8,30%, “q” con 7,30%, “o” correspondiente a 6,30% y “n” representa el 6,10%.

Tabla 18. Resultados generales de las pruebas.



Fuente: elaboración propia.

La aplicación del programa *Perséfone* conforme a la sumatoria de las siete pruebas generó un resultado de novecientos veinticinco caracteres. En el proceso de detección y transcripción correcta un total de trescientos cuarenta aciertos, que corresponde a un 36,8%; los puntos medios o parciales: cincuenta y seis, equivalente a un 6%. La cantidad de errores fueron quinientos veintinueve, es decir, 57,2%.

El promedio total de aciertos es del 40,9% y la desviación estándar es 4,11.

**Capítulo 5**

**Conclusiones**

A continuación, se presentan los resultados, dudas y retos en el proceso de la realización del programa.

**5.1. Conclusiones Teóricas**

Una de las primeras dudas que surgen al abordar el tema de la transcripción automática de manuscritos antiguos es el significado de este proceso. Aunque en el marco teórico se hizo una aproximación en la definición, es clave poder realizar un estudio profundo de esta categoría. En esta tesis se analizó la transcripción como un fin en sí mismo y sus resultados como una forma de evaluar los resultados del programa. Sin embargo, es necesario plantear una definición cuando se considere y se desarrolle el programa como una herramienta dentro de los procesos y no como un fin.

Otra duda como reflexión teórica es el rol de la transcripción automática en la crítica textual. Por motivos de caracter conceptual, no se abordaron los temas de la crítica textual, ya que no eran de cierta manera atinentes al proceso de realización de este programa, empero, es una duda que se convoca a ser explorada. Una de las formas de explorar esta cuestión es la visión de la transcripción automática como fin en sí mismo; así, la pregunta abre la posibilidad para la generación de procesos de creación de herramientas orientadas a la crítica textual dentro de alguna de las ramas de la transcripción automática.

* 1. **Proyecciones del programa**

En tanto a las proyecciones del programa viene a la mente el proceso de optimización de Perséfone. En primer lugar, es necesario replantear la forma de adquisición de palabras y letras. Aunque es funcional, es posible mejorar los resultados con otras aproximaciones. Una posibilidad es la aplicación de conectividad para las astas y caídos, permitiendo una mejor definición de los caracteres. Por otro lado, viendo la aproximación del proyecto realizado por Markus Diem y Robert Sablatnig[[42]](#footnote-42) se podría experimentar con un proceso identificación sin utilizar binarización. Una de las necesidades del programa es la creación de una GUI que sea amable al usuario, asimismo que no dependa de Python, es decir, que quede como archivo ejecutable (.EXE).

Por otro lado, la continuidad de este trabajo es necesario para seguir desarrollando esta serie de tecnologías en el país; de esta manera, se evita caer en el atraso tecnológico y procedimental. Al momento de redactar estas líneas, como resultado y aporte del proceso de la tesis, se está gestionando un semillero de investigación con los docentes de Filología Clásica e Ingeniería Mecatrónica con el fin de continuar con la propuesta del programa *Perséfone* y otros proyectos de interés de los estudiantes.

* 1. **Retos**

Uno de los retos evidentes es la multidisciplinariedad. Para un filólogo sin entrenamiento en lenguaje de programación, adentrarse en este mundo es complicado, debido a la cantidad de matemática que requiere comprender los conceptos que son usados. Para solucionar esta falencia, se puede tomar cursos enfocados a las Humanidades Digitales; sin embargo, este campo no ha sido ampliamente explorado en el Departamento de Lingüística, con contadas excepciones, dirigidas a lingüistas.

Otro reto que vale la pena mencionar es el tamaño del equipo de trabajo. Este programa fue realizado por dos personas en un periodo de tres meses. Teniendo un equipo más grande y mayor tiempo, se puede llegar a un grado mayor de exactitud y avance.

Para terminar, es importante resaltar la necesidad de tener una aproximación holística al conocimiento, una aproximación con la cual la Ingeniería y la Filología puedan cantar aunadas, tal como Urania y Calíope recitaban en el Parnaso. Sin esta mentalidad, no sería posible tener este trabajo; una mentalidad que permite hacer realidad los milagros de la mitología.

**Capítulo 6**

**Glosario**

En este glosario se tratarán dos términos que no han sido explicados en el transcurso del presente documento. Para poder comprender el funcionamiento del programa, es necesario entender estos dos términos. A continuación, se dará una definición general, sin embargo, para una comprensión profunda de los temas, se recomienda revisar las referencias, dónde están explicados los conceptos con la matemática que conllevan.

**6.1. SIFT**

Scale-Invariant Feature Transform, o SIFT, es un detector-descriptor desarrollado por D. Lowe en 2004. Un detector-descriptor es un tipo de algoritmo que detecta puntos de interés sobre una imagen y los describe a partir de vectores. Ya teniendo identificados los puntos de interés, el algoritmo puede comparar una segunda imagen y encontrar puntos de interés similares o idénticos dentro de la segunda imagen. Una de las características más importantes del SIFT es que funciona correctamente a pesar de la rotación de la imagen, es decir, puede encontrar las similitudes entre imágenes con diferentes ángulos de rotación; asimismo, este algoritmo puede identificar imágenes que estén parcialmente ocluidas, haciéndolo una herramienta ideal para el trabajo del programa Perséfone.[[43]](#footnote-43)

**6.2. K-Means**

KMeans es un algoritmo de agrupación, es decir, un algoritmo que reúne en conjuntos elementos que comparten similitudes. Para lograr estos resultados, el algoritmo utiliza unos centroides sobre los cuales segmenta los datos ingresados. Este tipo de algoritmo es da buenos resultados en largos bancos de datos, por este motivo es utilizado en el programa Perséfone, puesto que la cantidad de descriptores resultantes del SIFT es muy alta.[[44]](#footnote-44)

**Capítulo 7**

**Bibliografía**

Mimbela, Mateo S.J. (2002) *Physices tractatus,* [Biblioteca Virtual del Pensamiento Filosófico en Colombia, Colección Fuentes Filosóficas, Serie I: Periodo Colonial 1620-1820, Vol.1: obras filosóficas del periodo colonial], Bogotá: Pontificia Universidad Javeriana, Instituto Pensar.

Emerson, Ralph Waldo (2010). The complete works. Vol VIII. Letters and Social Aims.

Platón. Fedón.Editorial Gredos (1988). Madrid, España. pp 403-404.

DeFrancis, John (1989). Visible Speech: The Diverse Oneness of Writing System*.*

Marquínez Argote, Germán (2002). Mateo Mimbela (1663-1736), el Maestro Aragonés que Enseñó Filosofía y Teología en el Nuevo Reino de Granada. Recuperado de: https://www.uco.es/ucopress/ojs/index.php/refime/article/view/9352/8849

Python Software Foundation: shutil — High level file operations. URL https://docs.python.org/3/library/shutil.html#module-shutil

Python Software Foundation: os — Miscellaneous operating system interfaces. URL https://docs.python.org/3/library/os.html

Python Software Foundation: glob — Unix style pathname pattern expansion. URL https://docs.python.org/2/library/glob.html

Python Software Foundation: tkinter — Python interface to Tcl/Tk. URL https://docs.python.org/3/library/tkinter.html#module-tkinter

Oliphant, Travis E.: Guide to NumPy. URL https://docs.scipy.org/doc/\_static/numpybook.pdf

OpenCV: Introduction. URL https://docs.opencv.org/4.1.0/d1/dfb/intro.html

The Matplotlib development team: History. URL https://matplotlib.org/users/history.html

Lundh, Fredrik: Python Imaging Library. URL http://effbot.org/imagingbook/overview.htm

Scikit-learn developers: scikit-learn user guide. URL https://scikit-learn.org/stable/\_downloads/scikit-learn-docs.pdf

Fischer, A., Wuthrich, M., Liwicki, M., Frinken, V., Bunke, H., Viehhauser, G., & Stolz, M. (2009). *Automatic Transcription of Handwritten Medieval Documents. 2009 15th International Conference on Virtual Systems and Multimedia.* doi:10.1109/vsmm.2009.26

Diem, M., & Sablatnig, R. (2010). *Recognizing characters of ancient manuscripts. Computer Vision and Image Analysis of Art.* doi:10.1117/12.843532

McCulloch, W. S., & Pitts, W. (1943). *A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, 5(4), 115–133.* doi:10.1007/bf02478259

Eikvil, Line: OCR — Optical Character Recognition. URL https://pdfs.semanticscholar.org/9484/96f9d73cab9c7b4fd5c3b656d1e5b1dc50d3.pdf

Nielsen, Michael: Neural Networks and Deep Learning*.* URL http://neuralnetworksanddeeplearning.com/index.html

IBM: IBM 1287 Optical Reader. URL: https://archive.org/details/TNM\_IBM\_1287\_optical\_reader\_hand-printed\_numeric\_\_20170911\_0168

Hamilton, Howard: Confusion Matrix. URL http://www2.cs.uregina.ca/~dbd/cs831/notes/confusion\_matrix/confusion\_matrix.html

Sathyanarayana, Shashi. (2014). A Gentle Introduction to Backpropagation. Numeric Insight, Inc Whitepaper.

TranScriptorium: Objectives. URL http://transcriptorium.eu/pagina/objectives/

TranScriptorium: Consortium. URL http://transcriptorium.eu/consortium/

TranScriptorium: tranScriptorium ends. READ starts. URL http://transcriptorium.eu/transcriptorium-ends-read-starts/

READ Project: About. URL https://read.transkribus.eu/about/

READ Project: Services. URL https://read.transkribus.eu/services/

OpenCV: Image Thresholding. URL https://opencv-python-tutroals.readthedocs.io/en/latest/py\_tutorials/py\_feature2d/py\_sift\_intro/py\_sift\_intro.html

Robinson, P. (2012). Towards a Theory of Digital Editions. Variants 10.

Andrews, T. (2013). The Third Way: Philology and Critical Edition in the Digital Age. Variants.

Text Encoding Initiative: TEI: Text Encoding Initiative URL https://tei-c.org/

Text Encoding Initiative: TEI: History URL https://tei-c.org/about/history/

NINE: About. URL http://www.juxtasoftware.org/about/

Scikit-learn developers: 2.3. Clustering. URL https://scikit-learn.org/stable/modules/clustering.html#k-means

Mordvintsev, Alexander. K, Abid: Introduction to SIFT (Scale-Invariant Feature Transform). URL https://opencv-python-tutroals.readthedocs.io/en/latest/py\_tutorials/py\_feature2d/py\_sift\_intro/py\_sift\_intro.html

Robinson and Solopova, E. 1993. Guidelines for transcription of the manuscripts of the Wife of Bath’s Prologue. In Norman F. Blake and Peter M. W. Robinson (eds.), The Canterbury Tales Project: Occasional Papers. Vol. 1. London: Office for Humanities Communication.

Reynolds, L.D. Wilson, N. D (1991). Scribes and Scholars.

Rosenblatt, Frank (1961). Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Recuperado de: https://safari.ethz.ch/digitaltechnik/spring2018/lib/exe/fetch.php?media=neurodynamics1962rosenblatt.pdf

Haykin, Simon (1999). Neural Networks: A Comprehensive Foundation.

Hauger, Scott J. (1995) Reading Machines for the Blind: A Study of Federally Supported Technology Development and Innovation.

Rivas Sacconi, José Manuel (1993). El Latín En Colombia: Bosquejo Histórico del Humanismo Colombiano.

Silva, Renán (2004). Saber, Cultura y Sociedad en el Nuevo Reino de Granada Siglos XVII y XVIII.

1. Tomado de *The complete works. Vol VIII. Letters and Social Aims.* Emerson, Ralph Waldo (2010). Trad: “La ciencia sobrepasa los viejos milagros de la mitología”. [↑](#footnote-ref-1)
2. Tomado de *Fedón*, Platón. Editorial Gredos (1988) pp 403-404. [↑](#footnote-ref-2)
3. Tomado de *Visible Speech: The Diverse Oneness of Writing System. DeFrancis, John* (1989) [↑](#footnote-ref-3)
4. *Saber, cultura y sociedad en el Nuevo Reino de Granada, siglos XVII y XVIII.* Silva, Renán. (2004). pp. 72-76 [↑](#footnote-ref-4)
5. Ibidem. [↑](#footnote-ref-5)
6. Ibidem. [↑](#footnote-ref-6)
7. *El Latín en Colombia*. Rivas Sacconi, José Manuel. (1993) p. 93. [↑](#footnote-ref-7)
8. *Mateo Mimbela (1663-1736), El Maestro Aragonés que Enseñó Filosofía y Teología en el Nuevo Reino de Granada*; Marquínez Argote, Germán (2002) pp. 297-300. [↑](#footnote-ref-8)
9. Ibidem. p. 300. [↑](#footnote-ref-9)
10. Ibidem. p. 301. [↑](#footnote-ref-10)
11. *El Latín en Colombia*. Rivas Sacconi, José Manuel. (1993) p. 100. [↑](#footnote-ref-11)
12. *Reading Machines for the Blind, A Study or Federally Supported Technology Development and Innovation*. Hauger, J. Scott. (1995) p. 11. [↑](#footnote-ref-12)
13. *IBM 1287 Optical Reader*. IBM. (1966). [↑](#footnote-ref-13)
14. *OCR- Optical Character Recognition*. Eikvil, Line. (1993). p. 33. [↑](#footnote-ref-14)
15. *Neural Networks and Deep Learning*. Nielsen, Michael A. (2015). cap. 1. [↑](#footnote-ref-15)
16. *Neural Networks, A Comprehensive Foundation*. Haykin Simon. (1990). cap. 4. [↑](#footnote-ref-16)
17. *A Gentle Introduction to Backpropagation*. Sathyanarayana, Sashi (2014). p.12 [↑](#footnote-ref-17)
18. Tomado de: *Literary and Linguistic Computing*. Robinson, Peter (1989). Trad: “El valor más inmediato de los métodos digitales es la habilidad de asignar tanto del trabajo -particularmente aquel que es repetitivo, demandante y propenso a los errores- al computador.” [↑](#footnote-ref-18)
19. *Manual de Crítica Textual*. Blecua, Alberto (1983). pp. 19-20. [↑](#footnote-ref-19)
20. The third way: philology and critical edition in the digital age. Andrews, Tara L. (2012). p.3. [↑](#footnote-ref-20)
21. *Guidelines for transcription of the manuscripts of the Wife of Bath’s*. Robinson, Peter. Solopova, Elizabeth (1993). p. 21. [↑](#footnote-ref-21)
22. Towards a Theory of Digital Editions. Robinson, Peter. (2013). p. 115. [↑](#footnote-ref-22)
23. *TEI: Text Encoding Initiative*. TEI. [↑](#footnote-ref-23)
24. *TEI: History*. TEI. [↑](#footnote-ref-24)
25. *About*. NINES. [↑](#footnote-ref-25)
26. *Consortium*. tranScriptorium. (2019). [↑](#footnote-ref-26)
27. *Objectives*. tranScriptorium. (2019). [↑](#footnote-ref-27)
28. *TranScriptorium ends. READ starts*. tranScriptorium. (2016). [↑](#footnote-ref-28)
29. *Automatic Transcription of Handwritten Medieval Documents*. Wüthrich, Markus, et al. (2009). [↑](#footnote-ref-29)
30. *Recognizing Characters of Ancient Manuscripts*. Diem, Markus. Sablatnig, Robert. (2010). [↑](#footnote-ref-30)
31. *glob* **—** *Unix style pathname pattern expansion*. Python Software Foundation. (2019). [↑](#footnote-ref-31)
32. *os — Miscellaneous operating system interfaces*. Python Software Foundation. (2019). [↑](#footnote-ref-32)
33. *Shutil —**High-level file operations.* Python Software Foundation. (2019). [↑](#footnote-ref-33)
34. *tkinter — Python interface to Tcl/Tk*. Python Software Foundation. (2019) [↑](#footnote-ref-34)
35. *Guide to NumPy*. Oliphant, Travis E. (2006). [↑](#footnote-ref-35)
36. *Introduction*. OpenCV. (2019). [↑](#footnote-ref-36)
37. *History*. The Matplotlib development team. (2019). [↑](#footnote-ref-37)
38. *scikit-learn user guide, release 0.21.2*. scikit-learn developers (2019). (Véase en el Glosario). [↑](#footnote-ref-38)
39. *Python Imaging Library Overview*. Lundh, Fredrik. (2005). [↑](#footnote-ref-39)
40. *Image Thresholding*. OpenCV. (2018). [↑](#footnote-ref-40)
41. *Confusion Matrix*. Hamilton, Howard (2018). [↑](#footnote-ref-41)
42. *Recognizing Characters of Ancient Manuscripts*. Diem, Markus. Sablatnig, Robert. (2010). [↑](#footnote-ref-42)
43. Introduction to SIFT (Scale-Invariant Feature Transform). Mordvintsev, Alexander. K, Abid. (2013) [↑](#footnote-ref-43)
44. *Clustering*. scikit-learn developers. (2007-2019). [↑](#footnote-ref-44)