FACULTAD DE INFORMÁTICA Y MATEMÁTICA

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

VISUALJCLAL: ENTORNO DE USUARIO PARA EL *FRAMEWORK* JCLAL

Trabajo de diploma para optar por el título de Ingeniero Informático

Autor: Luis Miguel Sánchez Velázquez

Holguín, 2017

UNIVERSIDAD DE HOLGUÍN

FACULTAD DE INFORMÁTICA Y MATEMÁTICA

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

VISUALJCLAL: ENTORNO DE USUARIO PARA EL *FRAMEWORK* JCLAL

Trabajo de diploma para optar por el título de Ingeniero Informático

Autor: Luis Miguel Sánchez Velázquez

Tutores: M. Sc. Eduardo Pérez Perdomo
Ing. Luis David González Orozco

Consultante: Dr. Oscar Reyes Pupo

Holguín, 2017

Dedicatoria

Dedicado a mi madre, eterna luz de mi existir.

Agradecimientos

A mi madre, por cuidarme todos estos años.

A mis hermanos y mi padre, por su apoyo incondicional.

A mi novia, por su comprensión y apoyo.

A mis tutores Eduardo, Oscar y Luis David, por el apoyo y confianza que me han brindado durante el desarrollo de la tesis.

A mis amigos fieles y compañeros de grupo, por todo lo que me han enseñado y ayudado en estos años de estudio.

A todos los profesores que han sido parte de mi formación profesional.

Resumen

El Aprendizaje Automático es una rama de la Inteligencia Artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. A partir de casos pasados se intenta descubrir patrones generalizables para posibles nuevos casos. En este existen varias áreas de estudio, entre ellas se encuentra el Aprendizaje Supervisado, No Supervisado, Semi-Supervisado y Aprendizaje Activo.

En la actualidad existen varias herramientas informáticas del área de Inteligencia Artificial que permiten la aplicación de métodos y algoritmos de Aprendizaje Automático y sus áreas de estudio para la clasificación de conjuntos de datos, solución de problemas e inferencia de conocimientos. El *framework* JCLAL es una herramienta informática de código abierto para investigadores y usuarios que apliquen Aprendizaje Activo.

JCLAL provee los mecanismos necesarios para desarrollar cualquier técnica de Aprendizaje Activo. Aunque JCLAL cuenta con una amplia variedad de elementos positivos como es su portabilidad, elegancia e interoperabilidad, carece de una Interfaz de Usuario al nivel de otros *framework* que les permitiera a los investigadores realizar los experimentos con solo conocer el flujo general del Aprendizaje Activo y que de esta forma más investigadores de la comunidad científica se acerquen al *framework*.

En la presente investigación se propone un entorno de usuario para el *framework* JCLAL. Se analizan los elementos teóricos que sustentan el Aprendizaje Activo y el *framework* JCLAL. Se aplican patrones de diseño y se diseña la estructura del entorno de usuario. Por último, son realizadas pruebas de aceptación para medir la usabilidad de la interfaz propuesta.

Abstract

The Active Learning is a branch of the Artificial Intelligence witch goal is developing techniques that allows to the computers to learn. From past cases we try to discover generalizable patterns for possible new cases. In there are several areas of study, among them is Supervised, Non-Supervised, Semi-Supervised and Active Learning.

At present there are several computer tools in the area of Artificial Intelligence that allow the application of Automatic Learning methods and algorithms and their areas of study for classification of data sets, problem solving and knowledge inference. The JCLAL framework is an open source computing tool for researchers and users who apply Active Learning.

JCLAL provides the interfaces, classes and methods necessary to develop any Active Learning technique. Although JCLAL has a wide variety of positive elements such as portability, elegance and interoperability, it lacks in a User Interface that allows researchers to perform experiments without detailed knowledge on how they are executed, or the internal structure of the framework.

In the present research we propose a user environment for the JCLAL framework. The theoretical elements that support Active Learning and the JCLAL framework are analyzed. Design patterns are applied and the structure of the user environment is designed. Finally, acceptance tests are performed to measure the usability of the proposed interface.

ÍNDICE

Índice de figuras

Índice de tabla

INTRODUCCIÓN

En la actualidad existen varias herramientas informáticas del área de Inteligencia Artificial (IA, por sus siglas en español) que permiten la aplicación de métodos y algoritmos de aprendizaje automático para la clasificación de conjuntos de datos, solución de problemas e inferencia de conocimientos. Estas herramientas informáticas tienen aplicación en varias esferas, por ejemplo, en la industria existen sistemas de análisis del mercado y detección de fraude. Existen otras en el área de la medicina que se utilizan para la predicción de cáncer o enfermedades hereditarias y predicción de ataques al corazón. El procesamiento del lenguaje natural es una de las esferas de más impacto con sistemas de recuperación de información, reconocimiento de voz, minería de datos y extracción de conocimientos.

El Aprendizaje Automático (ML por sus siglas en inglés, *Machine Learning*) es una rama de la IA cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. A partir de casos pasados se intenta descubrir patrones generalizables para posibles nuevos casos [1].En él existen varias áreas de estudio, entre ellas se encuentra el aprendizaje supervisado y no supervisado. En el aprendizaje supervisado la información que se utiliza está previamente clasificada y se conoce la categoría a la que pertenece cada ejemplo. En el aprendizaje no supervisado no existe una clasificación previa de la información, la cual se agrupa automáticamente [1].

Los algoritmos de clasificación tradicionales emplean solamente ejemplos etiquetados en el proceso de entrenamiento. Sin embargo, en la realidad la obtención de ejemplos etiquetados es una tarea costosa y muchas veces requiere del esfuerzo de humanos experimentados. A partir de esta ineficiencia del aprendizaje supervisado surge el aprendizaje semi-supervisado y el Aprendizaje Activo [2].

El Aprendizaje Activo[1] (AL por sus siglas en inglés, *Active Learning*) es un área de estudio la cual tiene como principal hipótesis: si el algoritmo de aprendizaje tiene la oportunidad de elegir los datos desde donde aprende, entonces este tendrá un mejor rendimiento con

[1] Aprendizaje Activo: se le conoce también como aprendizaje por consulta o diseño experimental óptimo en la literatura de estadística.

un menor costo de entrenamiento. El AL trata de superar los obstáculos del etiquetado de datos mediante la consulta en la forma de ejemplos no etiquetados para ser etiquetados por un oráculo (ejemplo: un anotador humano). De esta manera, AL intenta mejorar la precisión de un clasificador usando la menor cantidad posible de ejemplos etiquetados [3].

Un *framework* en el área de las ciencias de la computación se define como un conjunto de técnicas y herramientas que permiten el desarrollo de algún producto, abstrayendo y facilitando el desarrollo de ciertas tareas, aportando soluciones comunes que han sido propuestas con anterioridad, según el dominio para el cual está construido el mismo. Los *frameworks* brindan una estructura para la solución de problemas de un dominio determinado, lo cual permite un ahorro del tiempo en la resolución de un problema [4].

Entre los *framework* que existen en el área de ML uno de los más conocidos es: Entorno para el Análisis del Conocimiento de la Universidad de Waikato (Weka[2], por sus siglas en inglés, *Waikato Environment for Knowledge Analysis*). Weka es el más popular y difundido en la comunidad de investigadores de ML, el cual incluye técnicas clásicas de aprendizaje supervisado y no supervisado. Uno de los factores que contribuye a su popularidad es que cuenta con una Interfaz Gráfica de Usuario (GUI por sus siglas en inglés, *Graphical User Interface*) que le permite a los investigadores experimentados o recién iniciados en el uso del *framework* aplicar las técnicas de ML de forma sencilla y rápida. Sin embargo, este no incluye un tratamiento de las técnicas provenientes del área de AL.

Los investigadores en el área de AL deben realizar experimentos para comparar entre el desempeño de los algoritmos y técnicas antes de su aplicación, lo que suele consumir bastante tiempo. En busca de facilitar el trabajo a los investigadores y por la falta de un *framework* que posea técnicas de esta área de estudio, se crea el *framework* JCLAL, por sus siglas en inglés *Java Class Library for Active Learning* (Biblioteca de Clases Java para el Aprendizaje Activo), para la experimentación y desarrollo de algoritmos de AL [5].

[2] http://www.cs.waikato.ac.nz/ml/weka/

JCLAL es un *software* de código abierto para investigadores y usuarios que apliquen AL. Incluye las estrategias más relevantes de AL que han sido propuestas en la literatura para los tipos de datos *single-label* y *multi-label*. JCLAL cuenta con una amplia variedad de elementos positivos como es su portabilidad, elegancia, interoperabilidad, genérico, paralelo, escalable y distribuido (este último mediante el uso del *framework* Spark[3]). Permite a los desarrolladores adaptar, modificar y ampliar el *framework* de acuerdo a sus necesidades.

La forma de realizar los experimentos en JCLAL es mediante la configuración de un archivo en formato XML donde se incluyen las etiquetas necesarias para la ejecución de los experimentos. Estas etiquetas hacen referencia a los principales elementos del flujo de AL, que son configurables mediante los paquetes de clases de JCLAL. Para ejecutar un experimento se deben usar distintos comandos desde la consola, que permitan al *framework* conocer los parámetros necesarios para desarrollar este proceso como son la dirección donde se encuentra el archivo de configuración, los métodos que se utilizaran en la experimentación o si se desea ejecutar el experimento en paralelo.

Este proceso de experimentación de AL es totalmente funcional. Sin embargo, aspectos mejorables como la facilidad de realización, el tiempo empleado en la configuración, la detección de errores y la usabilidad en general del *framework*, son de necesario acercamiento en función de simplificar la experimentación y aumentar el empleo de JCLAL por parte de la comunidad de investigación científica en el área de AL, y que de esta forma se convierta en referencia para la investigación en esta área de la Inteligencia Artificial.

A partir de la situación problemática anterior se puede identificar el siguiente **problema científico**: ¿Cómo facilitar la realización de experimentos aplicando Aprendizaje Activo en el *framework* JCLAL?

[3] Spark es un framework de código abierto que combina un motor de distribución a través de clúster de máquinas, con un elegante modelo para escribir programas sobre él, lo que hace que la programación distribuida sea verdaderamente accesible a los científicos de datos.

El **objeto de estudio** en el que se encuentra enmarcado el problema es la experimentación aplicando Aprendizaje Activo.

Para darle solución a la deficiencia detectada en el problema científico se propone como **objetivo general**: desarrollar un entorno de usuario para el *framework* JCLAL que permita realizar experimentos usando técnicas de Aprendizaje Activo.

El **campo de acción** delimitado por el objetivo general es: realización de experimentos en el *framework* JCLAL.

Para guiar la investigación se trazaron las siguientes **preguntas científicas**:

- ¿Cuáles son los fundamentos teóricos de las técnicas del Aprendizaje Activo?
- ¿Cuál es el uso actual de las técnicas y métodos de Aprendizaje Activo en el *framework* JCLAL?
- ¿Cómo diseñar un entorno de usuario que facilite la realización de experimentos en el *framework* JCLAL?
- ¿Cómo valorar la facilidad de uso de la Interfaz Gráfica de Usuario propuesta para el *framework* JCLAL?

En función de darle cumplimiento al objetivo de la investigación y de darle respuesta a las anteriores preguntas científicas se trazaron las siguientes **tareas científicas**:

1. Analizar los fundamentos teóricos que rigen las técnicas de Aprendizaje Activo.
2. Analizar el uso actual de las técnicas de Aprendizaje Activo en el *framework* JCLAL.
3. Diseñar un entorno de usuario para el *framework* JCLAL.
4. Implementar un entorno de usuario para el *framework* JCLAL, que facilite la realización de experimentos usando técnicas de Aprendizaje Activo.
5. Valorar la usabilidad de la GUI propuesta para el *framework* JCLAL.

Para dar solución a las tareas planteadas, se utilizó una combinación de métodos de trabajo científico, entre los que destacan los siguientes:

Métodos Teóricos: El método de análisis y síntesis se utilizó para descomponer el problema de investigación mentalmente en sub-problemas y profundizar en su estudio, para luego sintetizarlos en una solución que los integrara. El método de modelación permitió realizar el diseño de la GUI del *framework* JCLAL y de esta manera comprender mejor los elementos que se deben implementar. El método sistémico fue utilizado para crear la estructura de la GUI del *framework*, en el análisis y determinación de las relaciones y dependencias. El histórico-lógico, para el estudio de manera cronológica de investigaciones previas y para usarse como elemento de referencia.

Métodos Empíricos: El método de revisión de la documentación permitió realizar una revisión bibliográfica profunda para plasmar y referenciar el conocimiento adquirido relacionado con el objeto de estudio y la fundamentación de la solución propuesta. El método de observación y el método de entrevista permitieron apreciar y entender la forma actual en que se realizan los experimentos usando el *framework* JCLAL.

El documento está estructurado en introducción, dos capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos. En el Capítulo 1 se exponen las bases teóricas que dan fundamento a la investigación, y a continuación se describen las tecnologías y herramientas que se utilizan para el desarrollo de la GUI propuesta. El Capítulo 2 abarca lo referente a las acciones realizadas en las diferentes etapas que propone la metodología de desarrollo empleada, además se realiza una valoración de la calidad del *software* basado en la usabilidad del mismo mediante encuestas a los usuarios. Por último, se exponen las principales conclusiones a las que se arribaron con el desarrollo de esta investigación.

CAPÍTULO 1: FUNDAMENTOS TEORÍCOS EN EL DESARROLLO DEL ENTORNO DE USUARIO PARA EL FRAMEWORK JCLAL

En este capítulo se plantean los conceptos que enmarcan el objeto de estudio y la solución propuesta, tales como las técnicas y métodos de AL. Además de su uso y aplicación en el *framework* JCLAL, abundando también en lo referente al diseño e implementación de GUI. Además, se describen las tecnologías y herramientas utilizadas en el desarrollo de la solución propuesta y se fundamenta la metodología de desarrollo escogida.

1.1 Aprendizaje Activo

El Aprendizaje Automático es una rama de la IA cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. A partir de casos pasados se intenta descubrir patrones generalizables para posibles nuevos casos. En él existen varias áreas de estudio, entre ellas se encuentra el Aprendizaje Supervisado y No Supervisado. En el Aprendizaje Supervisado la información que se utiliza está previamente clasificada y se conoce la categoría a la que pertenece cada ejemplo. En el Aprendizaje No Supervisado no existe una clasificación previa de la información, la cual se agrupa automáticamente [1].

Una tarea del Aprendizaje Supervisado es la clasificación. Para realizar el proceso de clasificación se alimenta de un conjunto de ejemplos que han sido etiquetados manualmente de antemano y sirven para construir un clasificador, que después será utilizado para detectar a qué clase o categoría pertenecen los ejemplos nuevos.

En la aplicación de ML los métodos tradicionales de Aprendizaje Supervisado y No Supervisado no satisfacen resolver el problema real de contar para la clasificación con un conjunto de entrenamiento que posea un número pequeño de ejemplos etiquetados y un gran número de ejemplos no etiquetados. A partir de esta deficiencia aparece el Aprendizaje Activo como solución a este tipo de problema para tareas de clasificación.

De manera general, el AL trata de obtener un clasificador y un conjunto relativamente pequeño de ejemplos etiquetados para a partir de estos actualizar el clasificador principal. El aprendizaje activo en la tarea de clasificación puede mejorar el clasificador

con menos ejemplos etiquetados de entrenamiento, si se permite elegir activamente los ejemplos de los cuales él aprende [3]. Es decir, el algoritmo de AL funciona tomando como entrada un pequeño conjunto de ejemplos etiquetados y uno más grande de ejemplos no etiquetados de los cuales selecciona de forma activa ejemplos a etiquetar.

La aplicación del aprendizaje activo ha sido enfocada ampliamente en las tareas de:

- Clasificación y filtrado de imágenes, archivos de audio y video [3]
- Clasificación de texto [6] [7] [8].
- Procesamiento del Lenguaje Natural (NLP, por sus siglas en inglés, *Natural Language Processing*), entre las que se encuentran:
    - Filtrado de información adaptable [9]
    - Extracción de información [10]
    - Reconocimiento de entidades [11]
    - Etiquetamiento de partes de la oración [12] [13] [14]
    - Análisis sintáctico [15] [16]
    - Desambiguación del significado de la palabra [17]
    - Comprensión del lenguaje hablado [18]
    - Transliteración automática [19]
    - Segmentación de secuencia [20]

El aprendizaje activo posee diferentes técnicas en el proceso de aprendizaje. De manera general estas técnicas funcionan dividiendo el conjunto de datos en uno de entrenamiento y otro de prueba. Este proceso se repite utilizando el conocimiento recién ganado para escoger cuidadosamente los próximos ejemplos a seleccionar.

1.1.1 Protocolos de aprendizaje activo

Existen varios enfoques y protocolos en la literatura donde se manifiesta la aplicación de estas técnicas, en la Figura 1 se puede apreciar un esbozo general de su funcionamiento:

- *Membership Query Synthesis* (Síntesis de consulta de membresía): se consulta aleatoriamente cada ejemplo no etiquetado en el espacio de entrada, para luego ser etiquetado [3] [22].

- *Stream-based Active Learning* (Aprendizaje Activo basado en flujo): este evalúa un ejemplo a la vez, para luego decidir si el ejemplo consultado es etiquetado o ignorado [3] [23].

- *Pool-based Active Learning* (Aprendizaje Activo basado en conjunto o piscina): dado un conjunto de ejemplos no etiquetados, los ordena de acuerdo a su informatividad y luego consulta la etiqueta para aquellos ejemplos que sean más informativos [7] [3].

El escenario síntesis de consulta de membresía usualmente es eficiente pero el etiquetamiento aleatorio de ejemplos puede ser poco práctico para problemas de dominio real si el oráculo[4] es un anotador humano, precisamente estas limitantes dieron lugar al uso de los otros dos escenarios.

La diferencia principal entre el aprendizaje activo basado en flujo y el basado en conjunto es que el primero realiza la exploración por los ejemplos secuencialmente y toma decisiones de consulta de manera individual, mientras que el otro evalúa y ordena el fondo completo de ejemplos no etiquetados antes de seleccionar las mejores consultas [3].

[4] En aprendizaje activo un oráculo es aquel que se encarga de etiquetar los ejemplos no etiquetados, este puede ser un humano o simulado.

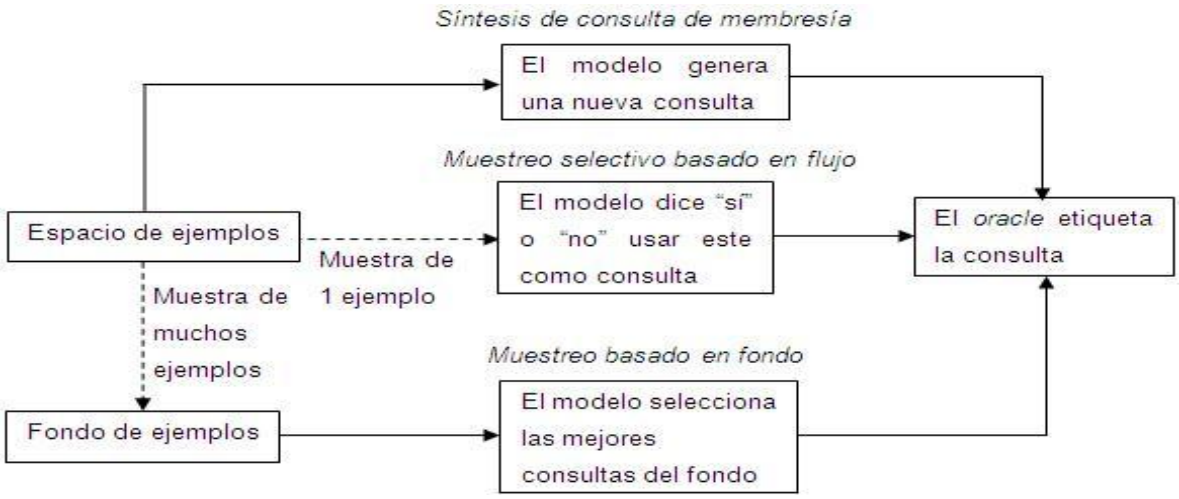

Figura 1: Protocolos de aprendizaje activo. Fuente: Adaptada de (SETTLES, 2010).

El protocolo basado en fondo ha sido mucho más aplicado en tareas reales como en clasificación y recuperación de imágenes [21] [24] , clasificación y recuperación de videos [25], clasificación de texto [25] [26] [24], extracción de información [27] [28], reconocimiento del habla [29] [18] y en el diagnóstico de cáncer [30].

El protocolo basado en flujo es más apropiado cuando la memoria o el poder de procesamiento pueden ser limitados, como con los dispositivos móviles e incrustados. Es aplicado en tareas como el reconocimiento de palabras [13], programación de sensores [3] y en el área de recuperación de la información [3].

1.1.2 Estrategias de consulta

Como se explicó en el epígrafe anterior una de las partes importantes dentro de las técnicas de aprendizaje activo es el método de consulta de la etiqueta de los ejemplos a etiquetar. De ahí, que en todos los protocolos anteriormente explicados se emplee el término consulta como proceso de evaluación de información de los ejemplos no etiquetados.

En la bibliografía existen diferentes criterios para formular las estrategias de consulta, tal es el caso de:

- Muestra de incertidumbre [3] [7]

- Consulta por comité [3] [31] [32]
- Reducción esperada del error [3] [26]
- Reducción de la varianza [3] [25] [33]
- Densidad y diversidad [3] [27]

1.2 *Framework* JCLAL

JCLAL es un *software* de código abierto para investigadores y usuarios que apliquen técnicas y métodos de AL. Está inspirado en la arquitectura del *framework* JCLEC (por sus siglas en inglés, *Java Class Library for Evolutionary Computating*) para computación evolutiva creado por el grupo de investigación KDIS (por sus siglas en inglés, *Knowledge Discovery and Intelligent Systems*) de la Universidad de Córdova, España. Posee una arquitectura que sigue los principios de la programación orientada a objetos, donde es común y fácil reutilizar código. Las principales características de JCLAL son:

- Genérico: Permite incluir nuevos métodos de AL a través de una estructura de clases flexible, así como contar con la posibilidad de adaptar, modificar o extender el *framework* en función de las necesidades del desarrollador.
- Portable: el *framework* fue creado con código en el lenguaje de programación Java, asegurando su portabilidad a través de todas las plataformas que implementen un Máquina Virtual de Java.
- Elegante: el *framework* posee una estructura de clases coherente que sigue los buenos principios de la programación genérica y orientada a objetos.
- Interoperable: el uso de XML brinda una base común para las herramientas de desarrollo y para vincularlo con otros sistemas mediante el uso de *wrappers* que lo vuelven a la vez flexible pues permite la integración con cualquier otro *framework*.
- Código abierto: el código es libre y lanzado bajo la licencia *GNU/GPL* (*General Public License*), por tanto, puede ser distribuido y modificado sin ningún cargo. Se

puede encontrar en *SourceForge*[5], *GitHub*[6], OSSRH repositorio proporcionado por *Sonatype*, y *Maven Central Repository*.

- Distribuido: mediante el uso del *framework Spark* permite ejecutar los experimentos de JCLAL en clústeres distribuidos en la red incrementando la velocidad de experimentación.

JCLAL ofrece varias estrategias de AL de última generación para los paradigmas de aprendizaje *single-label* y *multi-label*. JCLAL trata de brindar todos los beneficios del *software* de código abierto de ML a los investigadores en el área de AL.

1.2.1 Configuración y ejecución de experimentos en el *framework* JCLAL

Los experimentos de AL en el *framework* JCLAL se pueden ejecutar mediante XML o directamente desde el código Java del *framework*, siendo la primera forma más aconsejable de usar para la experimentación y la segunda para la aplicación del *framework* en empresas para darle solución a problemas más específicos e integrarse a otros sistemas. En un archivo escrito en XML y editado en cualquier gestor de texto se deben configurar una serie de parámetros necesarios para ejecutar el experimento, a continuación, se muestran y explican algunas de las etiquetas más importantes de esta configuración (en el Anexo 1 se puede encontrar una configuración completa de un experimento):

- Método de evaluación: este establece el método de evaluación usado en el proceso de aprendizaje (ver código 1.1), los valores a elegir pueden ser: *FivePerTwoCrossValidation, HoldOut, kFoldsCrossValidation y LeaveOneOutCrossValidation.*

```
<process evaluation-method-type="net.sf.jclal.evaluation.method.HoldOut">
```

(1.1)

[5] https://sourceforge.net/projects/jclal/files/

[6] https://github.com/ogreyesp/JCLAL

- Conjunto de datos: el conjunto de datos (a partir de aquí *dataset*) para ser utilizado en la realización de un experimento puede ser declarado de varias maneras. Los usuarios pueden elegir extraer los conjuntos *labelled* y *un-labelled* del conjunto de entrenamiento seleccionado. Por otra parte, estos conjuntos pueden ser seleccionados directamente en el XML como parámetros. También se puede establecer seleccionando los conjuntos de entrenamiento y de prueba mediante una etiqueta o simplemente elegir el tipo de *dataset* y la ubicación (ver código 1.2).

```
<file-dataset type="net.sf.jclal.dataset.WekaDataset">
       <path>dataset/iris.arff</path>
</file-dataset>
```

(1.2)

- Algoritmo de AL: este parámetro establece el protocolo de AL usado en el proceso aprendizaje. Actualmente el algoritmo clásico de AL es soportado (ver código 1.3).

```
<algorithm type="net.sf.jclal.activelearning.algorithm.ClassicalALAlgorithm">
```

(1.3)

- Escenario de AL: establece el escenario usado en el proceso de AL (ver código 1.4). En el momento, cuenta con los escenarios *Pool-based sampling* (muestreo basado en piscina) y *Stream-based sampling* (muestreo basado en flujo).

```
<scenario type="...jclal.activelearning.scenario.PoolBasedSamplingScenario">
```

(1.4)

- Criterio de parada: este parámetro especifica cuando debe detenerse el experimento. El *framework* JCLAL provee diferentes maneras de establecer el criterio de parada. Los usuarios pueden definir nuevos criterios de parada de acuerdo a sus necesidades. El número máximo de iteraciones debe ser declarado para garantizar que el proceso de AL termine (ver código 1.5).

```
<stop-criterion type="net.sf.jclal.activelearning.stopcriteria.MaxIteration">
       <max-iteration>2</max-iteration>
</stop-criterion>
```

(1.5)

- Estrategia de consulta: establece la estrategia de consulta usada en el proceso de AL (ver código 1.6). Hasta el momento, las estrategias de consulta más significativas que han aparecido para los criterios de *multi-label* y *single-label* son suministrados.

```
<query-strategy type="….querystrategy.EntropySamplingQueryStrategy">
```

(1.6)

- El oráculo: este parámetro denota el oráculo usado en el proceso de AL. Se puede elegir entre dos tipos de oráculos. Un Oráculo Simulado (ver código 1.7) que revela la clase oculta en un conjunto no etiquetado seleccionado y un Oráculo Humano-Consola que iterativamente le pregunta al usuario la clase de un conjunto no etiquetado seleccionado. Los usuarios pueden definir nuevos tipos de oráculo de acuerdo a sus necesidades.

```
<oracle type="net.sf.jclal.activelearning.oracle.SimulatedOracle"/>
```

(1.7)

- *Listener*: se puede definir un *listener* (reporte) que permite mostrar los resultados del proceso de AL (ver código 1.8). El reporte es almacenado en un archivo y también es mostrado por consola. Una ventana donde el usuario puede visualizar y seleccionar la medida de evaluación a mostrar del proceso de AL.

```
<listener type="net.sf.jclal.listener.ClassicalReporterListener">
     <report-title>Default test</report-title>
     <report-frequency>1</report-frequency>
      <report-directory>reports/default test</report-directory>
      <report-on-file>true</report-on-file>
</listener>
```

(1.8)

Una vez configurado el experimento por el usuario en XML con todos los parámetros necesarios, existen varias formas de ejecutar el experimento. Los experimentos se pueden ejecutar mediante: el uso de archivos JAR y los IDEs NetBeans y Eclipse.

1.2.2 *Frameworks* de desarrollo de aprendizaje automático con GUI

El *framework* JCLAL está basado en la arquitectura de otros *framework* de ML, por tanto, el diseño de su GUI será similar en cuanto a su diseño visual al de estos otros *frameworks.* Entre los *framework* que existen en el área del ML con GUI se encuentran: Weka y Extracción del Conocimiento basado en Aprendizaje Evolutivo (*Keel*[7], por sus siglas en inglés, *Knowledge Extraction based on Evolutionary Learning*). *Weka* es el más popular y difundido en la comunidad de investigadores de ML, el cual incluye técnicas clásicas de aprendizaje supervisado y no supervisado. Por otra parte, *Keel* abarca las áreas de aprendizaje supervisado y no supervisado desde un punto de vista de la computación evolutiva. Los *framework* descritos anteriormente no incluyen ninguna técnica proveniente del área del AL [2].

1.2.2.1 *Framework* Weka

Weka es un software para aprendizaje automático y minería de datos desarrollado en la Universidad de Waikato, Nueva Zelanda. Es libre, de código abierto, desarrollado en Java y distribuido bajo los términos de la licencia *GNU* (*General Public License*). Es compatible con la mayoría de las plataformas y ha sido probado con los sistemas operativos *Linux*, *Windows* y *Macintosh*. Está orientado al desarrollo de experimentos utilizando aprendizaje supervisado y no supervisado. Su interfaz inicial es mostrada en la Figura 2. Posee implementados métodos de procesamiento y algoritmos de aprendizaje que se pueden aplicar de manera sencilla a los conjuntos de datos, así como herramientas para evaluar el resultado del aprendizaje. Incluye métodos para los problemas estándar de minería de datos como son: regresión, clasificación, agrupamiento y selección de atributos [34]. Contiene una colección de herramientas de

[7] http://www.keel.es

visualización y algoritmos para análisis de datos y modelado predictivo, unidos a una interfaz gráfica de usuario para acceder fácilmente a sus funcionalidades.

Figura 2: GUI del *framework* Weka. Fuente: adaptada de (BOUCKAERT et al., 2011).

1.2.2.2 *Framework* Keel

Desarrollado en la Universidad de Granada, España. Es un *software* libre, de código abierto, implementado con el lenguaje de programación Java. Soporta la mayoría de las técnicas de pre procesamiento de datos, además de que incorpora algoritmos de aprendizaje supervisado y no supervisado. A diferencia de *Weka*, *Keel* incorpora varios algoritmos de aprendizaje automático que se basan en el uso de técnicas de computación evolutiva y *softcomputing* [35]. Una parte de su interfaz es mostrada en la Figura 3.

Figura 3: GUI del *framework* Keel. Fuente: Adaptada de (ALCALÁ-FDEZ et al., 2010).

1.3 Diseño de Interfaces Gráficas de Usuario

En la actualidad nos encontramos con interfaces gráficas de usuario no solo en las computadoras sino en objetos de uso diario como celulares, cajeros automáticos, entre muchas cosas más.

La interfaz tiene un papel fundamental para que el producto sea o no competitivo. El producto será exitoso si el usuario consigue concretar una acción de manera rápida o entiende la secuencia de pasos a seguir. También sí encuentra con facilidad cómo concretar la acción que necesita o cuando considera atractivo el diseño de la aplicación que está utilizando. El diseño de la GUI suele considerarse como tarea secundaria, sin embargo al analizar por qué algunas aplicaciones fracasan, se evidencia que el diseño de la GUI tiene una gran influencia en dicho fracaso [36].

1.3.1 Diseño de GUI en Java

El diseño de GUI en Java comenzó por AWT (*Abstract Window Toolkit*) la primera Interfaz de Programación de Aplicaciones (API, del inglés *Application Programming Interface*) orientada a esta programación. Luego de la introducción de AWT para Java, Sun Microsystems crea JFC (*Java Foundation Classes*) como un súper conjunto de AWT con muchas nuevas características. Los principales API de JFC son *Swing* (para la creación de GUI más sofisticadas), *Java 2D* (para la creación de gráficos de alta calidad) y *Drag-and-Drop* (para arrastrar y soltar componentes de AWT o *Swing*). En el epígrafe 1.4.3.1 se amplía un poco acerca de las características de *Swing*.

1.4 Tecnologías y herramientas de desarrollo

A continuación, son descritas las herramientas y tecnologías que se analizaron y seleccionaron para desarrollar la solución propuesta, de acuerdo con las características que debe poseer la GUI del *framework*.

1.4.1 Metodologías de desarrollo de *software*

La metodología de desarrollo de *software* es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Existen metodologías que permiten desarrollar software de excelente calidad debido a varios

aspectos como: el control que las mismas proporcionan; la posibilidad de crear las bases para el desarrollo del *software* y su éxito en el tiempo y costo fijados. Cada una posee características que se relacionan en mayor medida con las necesidades de una organización y las características específicas de cada proyecto a desarrollar.

Entre las metodologías más extendidas y usadas se encuentran: Proceso Unificado Racional (RUP, por sus siglas en inglés, *Rational Unified Process*), Programación extrema (XP, por sus siglas en inglés, eX*treme Programming*) y ICONIX.

RUP constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos [37]. La metodología RUP es más apropiada para proyectos grandes, dado que requiere un equipo de trabajo capaz de administrar un proceso complejo en varias etapas. En cada ciclo de iteración, se hace exigente el uso de artefactos (un documento, un modelo, o un elemento de modelo), siendo por este motivo una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software [37].

Por otra parte, ICONIX hereda la participación y el compromiso de los usuarios finales como metodología ágil para verificar la completitud y el cumplimiento de los requisitos. No proporciona una rigurosa y extensa documentación como RUP y no puede superar la agilidad de XP. ICONIX emplea el modelo de desarrollo de software basado en el enfoque o paradigma de desarrollo de software iterativo e incremental. Utiliza el método orientado a objeto con el enfoque unificado basado en el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modeling Language*) [38].

A continuación, se tratará con mayor profundidad la metodología XP, la cual fue seleccionada para la modelación de la GUI propuesta para el *framework* JCLAL.

1.4.1.1 Metodología de desarrollo de software XP

La Programación Extrema (a partir de ahora XP) forma parte de un grupo de metodologías ligera (o ágil) junto a SCRUM, DSDM y Crystal (ágil) de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado.

La metodología XP se basa en cuatro variables:

- Costo: se refiere a las máquinas, especialistas y oficinas que se necesiten en la creación de un proyecto.
- Tiempo: esta variable se refiere al tiempo de entrega del proyecto, así como al tiempo total invertido en el desarrollo de este.
- Calidad: la calidad que posea el proyecto realizado interna y externamente.
- Alcance: el alcance se refiere a la intervención necesaria del cliente durante el desarrollo del *software*.

La metodología XP se basa en una serie de principios básicos agrupados en cuatro categorías.

- Retroalimentación a escala fina: sus principios enuncian la importancia de la planificación, las pruebas, la interacción fuerte con el cliente y la programación en equipo simultáneamente.
- Proceso continuo en lugar de por lotes: sus principios se centran en la integración continua de las versiones del código, el análisis del diseño continuo y la entrega en períodos de tiempo corto (como máximo 3 semanas).
- Entendimiento compartido: esta categoría engloba los principios referentes al diseño simple del *software*, la propiedad colectiva del código y el uso de un estándar de codificación que permita la legibilidad del código.
- Bienestar del programador: su único principio defiende la semana de 40 horas debido a que un programador cansado produce código de poca calidad.

1.4.2 Entorno de Desarrollo Integrado

Para el desarrollo de la GUI se utilizó un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés, *Integrated Development Environment*). Un IDE es un programa informático compuesto por un conjunto de herramientas de programación, orientado a

uno o varios lenguajes de programación para facilitar el desarrollo de cualquier aplicación a los programadores.

1.4.2.1 IntellijIDEA

IntellijIDEA es un IDE liberado por JetBrains en enero de 2001, posee dos versiones *Ultimate Edition* (privativa) y *Community Edition* (*software* libre). Es un IDE multiplataforma e "inteligente" que permite la codificación en los lenguajes Java, XML y Groovy. Incluye soporte para varios *framework* entre ellos *Swing* de Java. Posee una navegación, inspección de código y búsqueda muy rápida. Además, posee integración con los Sistemas de Control de Versiones más populares como: Git, Subversion y CVS.

Para seleccionar el IDE se tuvo en cuenta primeramente que fuera de código abierto y multiplataforma. El IDE IntellijIDEA *Community Edition* fue el escogido porque se encuentra en constante mejora y actualización, es uno de los más utilizados por la comunidad de desarrolladores y satisface las necesidades de implementación de la GUI del framework.

1.4.3 Lenguaje de programación Java

Java es un lenguaje de programación orientado a objetos, desarrollado a principios de los años 90 por SunMicrosystems. Entre sus características se encuentran: es multiplataforma, orientado a objetos, robusto, multiproceso y distribuido. Entre los factores que influyeron en la elección de este lenguaje para desarrollar la GUI propuesta se encuentran:

- La GUI requiere compatibilidad con el *framework* JCLAL, el cual fue desarrollado en Java.
- Posee un amplio respaldo de la comunidad de código abierto (la documentación y código fuente disponible hacen a este lenguaje mucho más accesible para los programadores).
- Experiencia del desarrollador con el uso del presente lenguaje de programación.

1.4.3.1 *Framework* Java Swing

Swing es un *framework* MVC (Modelo Vista Controlador) de la JFC derivada de AWT, utilizado como biblioteca de clases para la creación de GUI que viene incluida en el JDK (siglas en inglés de *Java Development Kit*). Este cuenta con muchas características que lo vuelven diferente de su predecesor a pesar de usar algunas de sus funciones, entre algunas de estas mejoras se encuentran las que siguen:

- El *look-and-feel* de la GUI se puede definir o crear por el desarrollador o simplemente se puede utilizar el específico de la plataforma donde se ejecute la aplicación.
- Los componentes de *Swing* que no son contenedores y alguno de los contenedores son completamente manejados por Java de manera que pueden tener cualquier característica que sea necesaria.

Esta cuenta con las siguientes ventajas para su elección:

- Utiliza código Java que le permite vincularse fácilmente con JCLAL.
- Es multiplataforma dado a su desarrollo en Java.
- Swing cuenta con más componentes y estos a su vez con más características que AWT.

Conclusiones parciales

En este capítulo se explicó el proceso de AL, analizando los elementos fundamentales que lo integran, entre los que se encuentran las estrategias de consultas y los escenarios. Se definieron los conceptos principales vinculados al problema y al objeto de estudio.

Mediante la revisión bibliográfica se pudo analizar el *framework* JCLAL y como se realizan los experimentos de AL en este hasta el momento. Además, se pudo apreciar la necesidad de implementar una GUI para facilitar la realización de los experimentos en JCLAL.

Fueron seleccionadas las tecnologías para desarrollar la solución propuesta, las cuales son herramientas de software libre. Se utiliza el lenguaje de programación Java con la biblioteca gráfica Swing para el desarrollo de la GUI propuesta. Por otra parte, se adoptó a XP como metodología de desarrollo de software por las características que presenta, y para implementar la solución propuesta se eligió al IDE IntellijIDEA.

CÁPITULO 2: DISEÑO E IMPLEMENTACIÓN DEL ENTORNO DE USUARIO PROPUESTO

En este capítulo se describen las etapas de planificación, diseño, desarrollo y pruebas del entorno de usuario propuesto para el *framework* JCLAL usando la metodología ágil de desarrollo XP. Se incluye en este capítulo la descripción detallada de los Requerimientos Funcionales (RF) y los Requerimientos No Funcionales (RNF) que debe cumplir la solución propuesta.

2.1 Exploración

En el proceso de desarrollo de la aplicación es un paso de vital importancia la captura de requerimientos funcionales y no funcionales, que representan las acciones que el sistema debe realizar para satisfacer las necesidades para las cuales se creó. Es por ello que su redacción es fundamental como base para la realización de futuras pruebas.

2.1.1 Requerimientos funcionales

RF1: Configurar un nuevo experimento de JCLAL.

- RF1.1: Configurar el método de evaluación.
- RF1.2: Configurar el método de muestreo.
- RF1.3: Configurar los datos del *dataset*.
- RF1.4: Configurar el algoritmo.
- RF1.5: Configurar la estrategia de consulta.
- RF1.6: Configurar los *listeners*.

RF2. Guardar la configuración de un experimento de JCLAL.

- RF2.1: Seleccionar la ubicación para guardar la configuración.

RF3. Cargar una configuración de un experimento de JCLAL.

RF4. Crear una configuración dinámica de los experimentos.

RF5. Ejecutar un experimento usando JCLAL.

RF5.1: Ejecutar un experimento en paralelo usando JCLAL.

RF5.2: Ejecutar un experimento de forma distribuida usando JCLAL.

RF6. Visualizar los resultados de un experimento.

RF6.1: Visualizar el tiempo de ejecución de los distintos experimentos.

RF6.2: Visualizar de forma gráfica o en texto el resultado de los experimentos.

2.1.2 Requerimientos no funcionales

- Apariencia o Interfaz de Usuario

RNF1: El diseño debe ser sencillo.

RNF2: Usar colores, íconos y componentes agradables para el usuario.

RNF3: Usar interfaces que se adapten a la resolución de pantalla del usuario.

- Usabilidad

RNF4: El entorno de usuario debe permitir realizar los experimentos de forma sencilla y rápida.

RNF5: El entorno de usuario debe incluir atajos de teclado para facilitar su uso por parte de los usuarios.

- Ayuda y Documentación

RNF6: Crear un manual de usuario del sistema en un formato estandarizado.

- Portabilidad

RNF7: El entorno debe poder ejecutarse en las plataformas Windows, Linux y Mac OS.

2.1.3 Personas relacionadas con el entorno de usuario

Las personas relacionadas con el entorno de usuario son aquellas que obtienen algún beneficio del uso de la misma. Este entorno de usuario está orientado a investigadores en el área de AL que deseen realizar experimentos sobre conjuntos de datos utilizando el *framework* JCLAL.

En el caso del entorno de usuario propuesto los investigadores que deseen utilizar JCLAL para la experimentación deberán realizar la configuración de los experimentos, asumiendo el rol de usuario.

2.1.4 Historias de Usuario

Tabla 1: Historia de usuario No. 1 Crear prototipo no funcional del entorno de usuario

Historia de Usuario	
No.:1	Nombre: Crear prototipo del entorno de usuario.
Usuarios: Todos	
Prioridad del negocio: Alta	Nivel de complejidad: Bajo
Estimación: 2 semana	Iteración asignada: 1
Descripción: Este prototipo debe servir como base para implementar el entorno de usuario.	
Información adicional (Observaciones): Este prototipo tendrá todos los componentes y clases necesarias para poder trabajar pero sin ninguna funcionalidad.	

Tabla 2: Historia de usuario No. 2 Configurar un nuevo experimento de JCLAL

Historia de Usuario

No.:2	Nombre: Configurar un nuevo experimento de JCLAL.
Usuarios: Todos	
Prioridad del negocio: Alta	Nivel de complejidad: Alto
Estimación: 3 semanas	Iteración asignada: 1
Descripción: El Investigador podrá configurar los distintos parámetros del experimento en el entorno de usuario.	
Información adicional (Observaciones): Configurar el experimento incluye configurar los datos del método de evaluación, el método de muestreo, el *dataset*, el algoritmo, la estrategia de consulta y los *listeners*.	

Tabla 3: Historia de usuario No. 3 Guardar la configuración de un experimento de JCLAL.

Historia de Usuario	
No.:3	Nombre: Guardar la configuración de un experimento de JCLAL.
Usuarios: Todos	
Prioridad del negocio: Alta	Nivel de complejidad: Alto
Estimación: 2 semana	Iteración asignada: 2
Descripción: El Investigador debe poder guardar los datos de un experimento que configure en el entorno de usuario.	
Información adicional (Observaciones): Guardar la configuración incluye poder seleccionar la ubicación donde hacerlo, además de hacerlo en formato de XML con extensión *cfg*.	

Tabla 4: Historia de usuario No. 4 Cargar la configuración de un experimento de JCLAL.

Historia de Usuario	
No.:4	Nombre: Cargar la configuración de un experimento de JCLAL.
Usuarios: Todos	
Prioridad del negocio: Alta	Nivel de complejidad: Alto
Estimación: 2 semana	Iteración asignada: 2
Descripción: El Investigador debe poder cargar los datos de un experimento que configure en el entorno de usuario.	
Información adicional (Observaciones): Cargar la configuración de un experimento creado en el entorno o previamente creado por el Investigador que este en formato XML con extensión *cfg*.	

Tabla 5: Historia de usuario No. 5 Crear una configuración dinámica de experimentos.

Historia de Usuario	
No.:5	Nombre: Crear una configuración dinámica de los experimentos.
Usuarios: Todos	
Prioridad del negocio: Alta	Nivel de complejidad: Alto
Estimación: 1 semana	Iteración asignada: 3
Descripción: El Investigador debe tener la opción de seleccionar una configuración base y repetir esta configuración variando algunos de los parámetros.	

Información adicional (Observaciones): Los parámetros del experimento que se pueden variar son: el *dataset,* los clasificadores, las estrategias de consulta y los escenarios.

Tabla 6: Historia de usuario No. 6 Ejecutar un experimento usando JCLAL.

Historia de Usuario	
No.:6	Nombre: Ejecutar un experimento usando JCLAL.
Usuarios: Todos	
Prioridad del negocio: Alta	Nivel de complejidad: Alto
Estimación: 2 semanas	Iteración asignada: 3
Descripción: El Investigador podrá ejecutar un experimento de AL usando el *framework* JCLAL.	
Información adicional (Observaciones): Ejecutar un experimento incluye poder hacerlo en Paralelo y de forma Distribuida usando el *framework* Spark.	

Tabla 7: Historia de usuario No. 7 Visualizar los resultados de un experimento.

Historia de Usuario	
No.:7	Nombre: Visualizar los resultados de un experimento.
Usuarios: Todos	
Prioridad del negocio: Alta	Nivel de complejidad: Alto
Estimación: 3 semanas	Iteración asignada: 4

Descripción: El Investigador podrá visualizar los resultados de un experimento que ejecute en el entorno de usuario.
Información adicional (Observaciones): Visualizar el experimento incluye poder ver el tiempo de ejecución, datos del experimento y una gráfica de los parámetros de este.

Cabe destacar que en la estimación del tiempo de duración de cada historia se utilizó como unidad la semana laborable que consiste en 5 días, cada uno con 8 horas laborables. Todo esto se realiza con la práctica de la metodología utilizada que afirma que se debe trabajar como máximo 40 horas a la semana.

2.2 Planificación

2.2.1 Plan de iteraciones

Para la selección del trabajo de cada iteración se tuvo en cuenta que este no tuviera más de 4 semanas de desarrollo, siguiendo las prácticas de la metodología seleccionada. En la tabla 7 se muestra la distribución de las Historias de Usuario por cada iteración.

Tabla 8: Distribución de las historias de usuario por iteración

Distribución de las historias de usuario por iteración		
Iteraciones	Orden de las historias de usuario a implementar	Tiempo
1	Crear prototipo del entorno de usuario. Configurar un nuevo experimento de JCLAL.	5 semanas
2	Guardar la configuración de un experimento de JCLAL. Cargar la configuración de un experimento de JCLAL.	4 semanas
3	Crear una configuración dinámica de los experimentos. Ejecutar un experimento usando JCLAL.	3 semanas

4	Visualizar los resultados de un experimento.	3 semanas

2.2.2 Plan de entregas

El plan de entregas es el compromiso final del equipo de trabajo con los clientes. Es una cuestión de vital importancia para el negocio entre ambas partes, ya que la entrega tardía o temprana de la solución, repercute notablemente en la economía y moral de todos los involucrados.

La estimación es uno de los temas complicados dentro del proceso de desarrollo de un proyecto utilizando la metodología XP y es por ello que resulta de vital importancia tener bien claros los requerimientos del cliente, el estilo de trabajo del equipo de desarrollo y el tiempo con que dispone el cliente para tener en sus manos la solución.

Tabla 9: Cronograma de entregas

Plan de entregas			
1ra Iteración	2da Iteración	3ra Iteración	4ta Iteración
13 de febrero de 2017	14 de marzo de 2017	5 de abril de 2017	27 de abril de 2017

2.3 Arquitectura del sistema

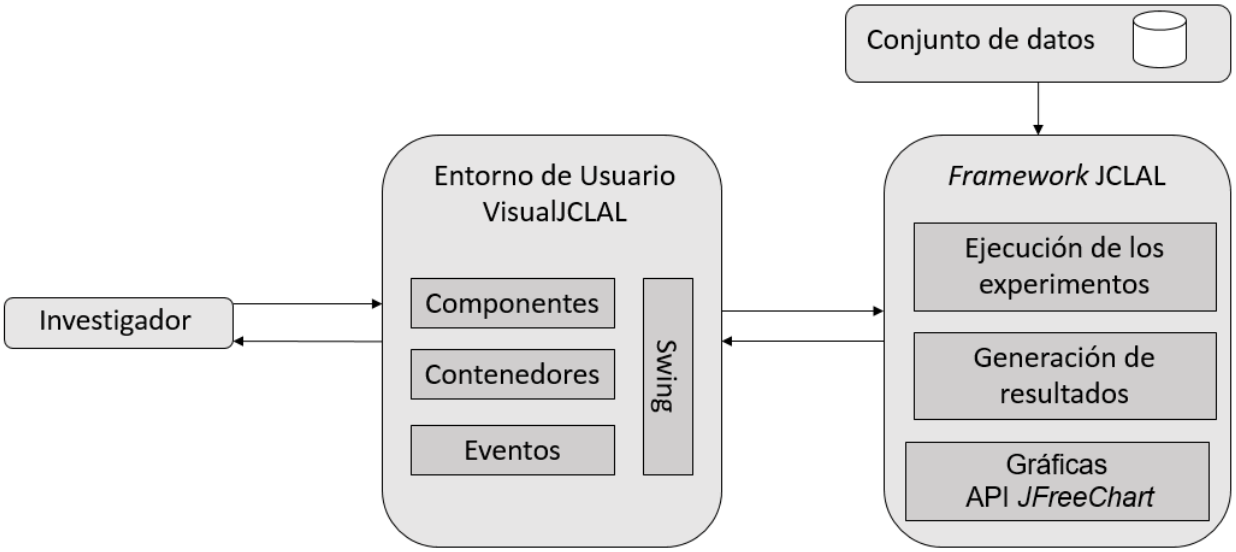

Figura 4: Arquitectura del entorno de usuario

2.4 Implementación

La metodología propone comenzar la implementación partiendo de una arquitectura lo más flexible posible para evitar grandes cambios en las próximas iteraciones y en los cambios que generalmente el cliente propone. Ya que la solución que se propone posee un gran componente técnico, es necesario desde un inicio tener bien definida la arquitectura.

Trazada la misma, se procede a la primera iteración.

2.4.1 Iteración 1

El principal objetivo de esta iteración es crear una aplicación con la estructura necesaria para implementar el estilo arquitectónico seleccionado, además de implementar la configuración de los experimentos en el entorno de usuario. Para ello se trazaron las siguientes tareas:

1. Tarea 1: Crear prototipo no funcional del entorno de usuario.
2. Tarea 2: Crear interfaz de usuario para configurar el método de evaluación e insertar los posibles parámetros a elegir por el Investigador.

3. Tarea 3: Crear interfaz de usuario para configurar el método de muestreo e insertar los posibles parámetros a elegir por el Investigador.
4. Tarea 4: Crear interfaz de usuario para configurar el *dataset* e insertar los posibles parámetros a elegir por el Investigador.
5. Tarea 5: Crear interfaz de usuario para configurar el algoritmo e insertar los posibles parámetros a elegir por el Investigador.
6. Tarea 6: Crear interfaz de usuario para configurar la estrategia de consulta e insertar los posibles parámetros a elegir por el Investigador.
7. Tarea 7: Crear interfaz de usuario para configurar los *listeners* e insertar los posibles parámetros a elegir por el Investigador.

Tabla 10: Tarea 1. Crear prototipo no funcional del entorno de usuario

Tarea	
No.:1	No. de la HU: 1
Nombre de la tarea: Crear prototipo no funcional del entorno de usuario.	
Tipo de tarea: Desarrollo	Estimación: 3 días
Fecha de inicio:12 de enero de 2017	Fecha de fin: 15 de enero de 2017
Programador responsable: Luis Miguel Sánchez Velázquez	
Descripción: Crear un prototipo no funcional del entorno de usuario para luego trabajar sobre el mismo. Crear las interfaces *GUIMain* y *MainExperimenterFrame* para la configuración, pero sin ninguna funcionalidad.	

Tabla 11: Tarea 2. Crear interfaz de usuario para configurar el método de evaluación e insertar los posibles parámetros a elegir por el Investigador.

Tarea	
No.:2	No. de la HU: 2
Nombre de la tarea: Crear interfaz de usuario para configurar el método de evaluación e insertar los posibles parámetros a elegir por el Investigador.	
Tipo de tarea: Desarrollo	Estimación: 5 días
Fecha de inicio:15 de enero de 2017	Fecha de fin: 20 de enero de 2017
Programador responsable: Luis Miguel Sánchez Velázquez	
Descripción: Crear interfaz de usuario *EvaluationMethodPanel* que permita configurar los datos del método de evaluación. Además insertar a partir de ficheros (con extensión *props*) algunos de los posibles parámetros que puede elegir el Investigador.	

Tabla 12: Tarea 3. Crear interfaz de usuario para configurar el método de muestreo e insertar los posibles parámetros a elegir por el Investigador.

Tarea	
No.:3	No. de la HU: 2
Nombre de la tarea: Crear interfaz de usuario para configurar el método de muestreo e insertar los posibles parámetros a elegir por el Investigador.	
Tipo de tarea: Desarrollo	Estimación: 7 días
Fecha de inicio: 20 de enero de 2017	Fecha de fin: 27 de enero de 2017
Programador responsable: Luis Miguel Sánchez Velázquez	

Descripción: Crear interfaz de usuario *ProcessPanel* que permita configurar los datos del método de muestreo. Además insertar a partir de ficheros (con extensión *props*) algunos de los posibles parámetros que puede elegir el Investigador.

Tabla 13: Tarea 4. Crear interfaz de usuario para configurar el *dataset* e insertar los posibles parámetros a elegir por el Investigador.

Tarea	
No.:4	No. de la HU: 2
Nombre de la tarea: Crear interfaz de usuario para configurar el *dataset* e insertar los posibles parámetros a elegir por el Investigador.	
Tipo de tarea: Desarrollo	Estimación: 7 días
Fecha de inicio: 27 de enero de 2017	Fecha de fin: 2 de febrero de 217
Programador responsable: Luis Miguel Sánchez Velázquez	
Descripción: Crear interfaz de usuario *DatasetPanel* que permita configurar los datos del *dataset*. Además insertar a partir de ficheros (con extensión *props*) algunos de los posibles parámetros que puede elegir el Investigador.	

Tabla 14: Tarea 5. Crear interfaz de usuario para configurar el algoritmo e insertar los posibles parámetros a elegir por el Investigador.

Tarea	
No.:5	No. de la HU: 2
Nombre de la tarea: Crear interfaz de usuario para configurar el algoritmo e insertar los posibles parámetros a elegir por el Investigador.	

Tipo de tarea: Desarrollo	Estimación: 4 días
Fecha de inicio: 2 de febrero de 2017	Fecha de fin: 6 de febrero de 2017
Programador responsable: Luis Miguel Sánchez Velázquez	
Descripción: Crear interfaz de usuario *AlgorithmPanel* que permita configurar los datos del algoritmo. Además insertar a partir de ficheros (con extensión *props*) algunos de los posibles parámetros que puede elegir el Investigador.	

Tabla 15: Tarea 6. Crear interfaz de usuario para configurar la estrategia de consulta e insertar los posibles parámetros a elegir por el Investigador.

Tarea	
No.:6	No. de la HU: 2
Nombre de la tarea: Crear interfaz de usuario para configurar la estrategia de consulta e insertar los posibles parámetros a elegir por el Investigador.	
Tipo de tarea: Desarrollo	Estimación: 3 días
Fecha de inicio: 6 de febrero de 2017	Fecha de fin: 9 de febrero de 2017
Programador responsable: Luis Miguel Sánchez Velázquez	
Descripción: Crear interfaz de usuario *QueryStrategyPanel* que permita configurar los datos de la estrategia de consulta. Además insertar a partir de ficheros (con extensión *props*) algunos de los posibles parámetros que puede elegir el Investigador.	

Tabla 16: Tarea 7. Crear interfaz de usuario para configurar los *listeners* e insertar los posibles parámetros a elegir por el Investigador.

Tarea

No.:7	No. de la HU: 2
Nombre de la tarea: Crear interfaz de usuario para configurar los *listeners* e insertar los posibles parámetros a elegir por el Investigador.	
Tipo de tarea: Desarrollo	Estimación: 4 días
Fecha de inicio: 9 de febrero de 2017	Fecha de fin: 13 de febrero de 2017
Programador responsable: Luis Miguel Sánchez Velázquez	
Descripción: Crear interfaz de usuario *ListenersPanel* que permita configurar los datos de los *listeners*. Además insertar a partir de ficheros (con extensión *props*) algunos de los posibles parámetros que puede elegir el Investigador.	

2.4.2 Iteración 2

Esta iteración tiene como objetivo lograr que el entorno de usuario permita guardar y cargar los datos de un experimento configurado dentro o fuera del entorno. Para ello se trazaron las siguientes tareas:

1. Tarea 8: Seleccionar ubicación para guardar la configuración del experimento de JCLAL.
2. Tarea 9: Cargar la configuración de un experimento de JCLAL.

Tabla 17: Tarea 8. Seleccionar ubicación para guardar la configuración del experimento de JCLAL.

Tarea	
No.:8	No. de la HU: 3
Nombre de la tarea: Seleccionar ubicación para guardar la configuración del experimento de JCLAL.	
Tipo de tarea: Desarrollo	Estimación: 14 días

Fecha de inicio: 13 de febrero de 2017	Fecha de fin: 27 de febrero de 2017
Programador responsable: Luis Miguel Sánchez Velázquez	
Descripción: Crear la opción de salvar mediante una barra de menú y un botón “salvar” para guardar la configuración actual de un experimento. Además, debe permitir seleccionar la ubicación donde almacenar el fichero.	

Tabla 18: Tarea 9. Cargar la configuración de un experimento de JCLAL.

Tarea	
No.:9	No. de la HU: 4
Nombre de la tarea: Cargar la configuración de un experimento de JCLAL.	
Tipo de tarea: Desarrollo	Estimación: 14 días
Fecha de inicio: 27 de febrero de 2017	Fecha de fin: 14 de marzo de 2017
Programador responsable: Luis Miguel Sánchez Velázquez	
Descripción: Crear la opción de cargar una configuración mediante una opción en una barra de menú y un botón “abrir” para establecer los parámetros en la interfaz que ya estén definidos en el fichero. Estos ficheros solo deben cumplir con la forma de configuración de JCLAL, no importa si han sido creados en el entorno de usuario o de forma manual por el Investigador.	

2.4.3 Iteración 3

Esta iteración tiene como objetivo lograr que el entorno de usuario permita ejecutar un experimento de AL. Para ello se trazaron las siguientes tareas:

1. Tarea 10: Crear una configuración dinámica de los experimentos.
2. Tarea 11: Ejecutar un experimento.

3. Tarea 12: Ejecutar un experimento en paralelo.

4. Tarea 13: Ejecutar un experimento de forma distribuida.

Tabla 19: Tarea 10: Crear una configuración dinámica de los experimentos.

Tarea	
No.:10	No. de la HU: 5
Nombre de la tarea: Crear una configuración dinámica de los experimentos..	
Tipo de tarea: Desarrollo	Estimación: 7 días
Fecha de inicio: 14 de marzo de 2017	Fecha de fin: 21 de marzo de 2017
Programador responsable: Luis Miguel Sánchez Velázquez	
Descripción: Crear una interfaz que le permita al Investigador seleccionar los elementos de AL que desea combinar para crear las configuraciones de experimentos.	

Tabla 20: Tarea 11. Ejecutar un experimento.

Tarea	
No.:11	No. de la HU: 6
Nombre de la tarea: Ejecutar un experimento.	
Tipo de tarea: Desarrollo	Estimación: 3 días
Fecha de inicio: 21 de marzo de 2017	Fecha de fin: 24 de marzo de 2017
Programador responsable: Luis Miguel Sánchez Velázquez	
Descripción: Crear una función *runExperiment()* que permita al Investigador ejecutar un experimento de forma normal usando JCLAL.	

Tabla 21: Tarea 12. Ejecutar un experimento en paralelo.

Tarea	
No.:12	No. de la HU: 6
Nombre de la tarea: Ejecutar un experimento en paralelo.	
Tipo de tarea: Desarrollo	Estimación: 4 días
Fecha de inicio: 24 de marzo de 2017	Fecha de fin: 28 de marzo de 2017
Programador responsable: Luis Miguel Sánchez Velázquez	
Descripción: Crear una opción dentro de la interfaz que permita decidir si se desea utilizar la facilidad que brinda JCLAL de ejecutar los experimentos en paralelo y permitir su configuración.	

Tabla 22: Tarea 13. Ejecutar un experimento de forma distribuida.

Tarea	
No.:13	No. de la HU: 6
Nombre de la tarea: Ejecutar un experimento de forma distribuida.	
Tipo de tarea: Desarrollo	Estimación: 7 días
Fecha de inicio: 28 de marzo de 2017	Fecha de fin: 5 de abril de 2017
Programador responsable: Luis Miguel Sánchez Velázquez	
Descripción: Crear una interfaz que permita configurar los parámetros necesarios para ser usados por el *framework* Spark que permite mediante JCLAL distribuir el experimento.	

2.4.4 Iteración 4

Esta iteración tiene como objetivo permitirle al Investigador tener un seguimiento de la ejecución del experimento, así como poder revisar los resultados obtenidos por el mismo. Para ello se trazaron las siguientes tareas:

1. Tarea 14: Visualizar el tiempo de ejecución de los distintos experimentos.
2. Tarea 15: Visualizar de forma gráfica o en texto el resultado de los experimentos.

Tabla 23: Tarea 14. Visualizar el tiempo de ejecución de los distintos experimentos.

Tarea	
No.:14	No. de la HU: 7
Nombre de la tarea: Visualizar el tiempo de ejecución de los distintos experimentos.	
Tipo de tarea: Desarrollo	Estimación: 14 días
Fecha de inicio: 5 de abril de 2017	Fecha de fin: 19 de abril de 2017
Programador responsable: Luis Miguel Sánchez Velázquez	
Descripción: Crear una interfaz que muestre el tiempo de ejecución así como posibles errores de ejecución del experimento.	

Tabla 24: Tarea 15. Visualizar de forma gráfica o en texto el resultado de los experimentos.

Tarea	
No.:15	No. de la HU: 7
Nombre de la tarea: Visualizar de forma gráfica o en texto el resultado de los experimentos.	
Tipo de tarea: Desarrollo	Estimación: 7 días

Fecha de inicio: 19 de abril de 2017	Fecha de fin: 27 de abril de 2017
Programador responsable: Luis Miguel Sánchez Velázquez	
Descripción: Crear una interfaz que muestre de forma gráfica o en texto el resultado de los experimentos. Para ello se utilizará el API JFreeChart para la creación de los gráficos y una interfaz de usuario para mostrar el texto.	

2.5 Prueba

El entorno de usuario fue sometido a varias pruebas en las que se observó un correcto funcionamiento y cumplimiento de los requisitos funcionales definidos con anterioridad. Los tiempos de respuestas de las diferentes funcionalidades del sistema se mantuvieron dentro de un rango razonable y la gestión de los datos se comportó de manera estable y efectiva. La interfaz gráfica tuvo una buena aceptación debido a su sencillez y facilidad de uso, lo cual propicio, de forma general, que se obtuviera una buena Experiencia de Usuario.

2.5.1 Pruebas de Aceptación

Las pruebas de aceptación también son conocidas como pruebas de caja negra, ya que es el propio cliente quien la realiza en compañía de uno de los representantes del equipo de desarrollo y se orientan a las funcionalidades del sistema. Su objetivo es comprobar, desde la perspectiva del usuario final, el cumplimiento de las especificaciones realizadas en las historias de usuario.

A continuación, aparecen las pruebas de aceptación realizadas a la solución propuesta:

Tabla 25: Caso de Prueba de Aceptación HU2-P1

Caso de Prueba de Aceptación	
Código: HU2-P1	Historia de usuario: 2
Nombre: Configurar un nuevo experimento.	

Descripción: Prueba la funcionalidad de crear una nueva configuración para un experimento de JCLAL.
Condiciones de ejecución: Para poder crear un nuevo experimento los datos de experimentación deben estar vacíos.
Entrada/Pasos de ejecución: Acceder a la interfaz de configuración mediante la interfaz principal haciendo clic en el botón *New Experiment*, en la opción de la barra de menú de igual nombre o pulsando el atajo del teclado “Ctrl + N”. En la interfaz de configuración que aparece llenar o seleccionar los datos necesarios.
Resultado esperado: Poder llenar los campos sin errores y seleccionar aquellos que sean correctos en función de la configuración.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 26: Caso de Prueba de Aceptación HU3-P2

Caso de Prueba de Aceptación	
Código: HU3-P2	Historia de usuario: 3
Nombre: Guardar la configuración de un experimento.	
Descripción: Prueba la funcionalidad de guardar una configuración de un experimento de JCLAL.	
Condiciones de ejecución: Para poder guardar un experimento los datos del experimento que se deseen ser salvados deben estar llenos en la interfaz de crear configuración.	
Entrada/Pasos de ejecución: Una vez en la interfaz de configuración hacer clic en el botón de la barra de menú *“Save experiment”* o pulsando el atajo del teclado “Ctrl + S”.	

En la interfaz de salvar que aparece seleccionar la ubicación para almacenar el archivo de configuración.
Resultado esperado: Se debe crear en la dirección seleccionada un archivo con extensión *cfg* y el nombre indicado por el usuario con aquellos datos que hayan sido seleccionados para salvar.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 27: Caso de Prueba de Aceptación HU4-P3

Caso de Prueba de Aceptación	
Código: HU4-P3	Historia de usuario: 4
Nombre: Cargar la configuración de un experimento.	
Descripción: Prueba la funcionalidad de cargar una configuración de un experimento de JCLAL.	
Condiciones de ejecución: Para poder cargar un experimento los datos deben estar almacenados en un archivo de extensión *cfg*.	
Entrada/Pasos de ejecución: Una vez en la interfaz de configuración o desde la interfaz inicial hacer clic en el botón *"Make from XML…",* pulsando el atajo del teclado "Ctrl + O" o haciendo clic en la opción de "*Open XML configuration*" de la barra de menú. En la interfaz de elección seleccionar el archivo a ser cargado por el programa.	
Resultado esperado: La interfaz de configuración debe aparecer o ser modificada en caso de ser visible con los valores de los parámetros que se encuentran en el archivo de configuración seleccionado.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 28:Caso de Prueba de Aceptación HU5-P4

Caso de Prueba de Aceptación	
Código: HU5-P4	Historia de usuario: 5
Nombre: Crear una configuración dinámica de experimentos.	
Descripción: Prueba la funcionalidad de crear una configuración dinámica de los experimentos usando VisualJCLAL.	
Condiciones de ejecución: Todos los parámetros de configuración deben estar correctamente completados por el usuario.	
Entrada/Pasos de ejecución: Una vez en la interfaz de configuración hacer clic en la opción de la barra de menú "*Create multiple experiments configuration*" o pulsando la combinación de teclas "Ctrl. + D".	
Resultado esperado: Debe mostrarse en caso de que todos los parámetros estén correctamente completados una interfaz que permita elegir el directorio donde almacenar las configuraciones de los experimentos y luego de elegir, debe aparecer otra que muestre los parámetros que se pueden combinar. Una vez elegidos debe almacenar en la ubicación elegida tantas configuraciones como el Investigador haya elegido.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 29: Caso de Prueba de Aceptación HU6-P5

Caso de Prueba de Aceptación	
Código: HU6-P5	Historia de usuario: 6
Nombre: Ejecutar un experimento usando JCLAL.	
Descripción: Prueba la funcionalidad de ejecutar un experimento usando JCLAL.	

Condiciones de ejecución: Todos los parámetros de configuración deben estar correctamente completados por el usuario.
Entrada/Pasos de ejecución: Una vez en la interfaz de configuración hacer clic en el botón *"Run experiment"*.
Resultado esperado: Debe mostrarse en caso de que todos los parámetros estén correctamente completados una interfaz que muestre el tiempo de ejecución del experimento así como al terminar o durante la ejecución un texto con la medida de los distintos parámetros.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 30: Caso de Prueba de Aceptación HU6-P6

Caso de Prueba de Aceptación	
Código: HU6-P6	Historia de usuario: 6
Nombre: Ejecutar un experimento en forma distribuida usando JCLAL.	
Descripción: Prueba la funcionalidad de ejecutar un experimento de forma distribuida usando JCLAL y el *framework* Spark.	
Condiciones de ejecución: Todos los parámetros de configuración deben estar correctamente completados por el usuario, en especial los de la interfaz de configuración de Spark.	
Entrada/Pasos de ejecución: Una vez en la interfaz de configuración hacer clic en el botón *"Run experiment"*.	
Resultado esperado: Debe mostrarse en caso de que todos los parámetros estén correctamente completados una interfaz que muestre el tiempo de ejecución del experimento, así como al terminar o durante la ejecución un texto con la medida de los	

distintos parámetros. Por supuesto efectuando el experimento de forma distribuida utilizando los clúster de Spark.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 31: Caso de Prueba de Aceptación HU7-P7

Caso de Prueba de Aceptación	
Código: HU7-P7	Historia de usuario: 7
Nombre: Visualizar los resultados de un experimento.	
Descripción: Prueba la funcionalidad de visualizar los resultados de un experimento ejecutado en JCLAL.	
Condiciones de ejecución: Contar con uno o varios de los resultados de un experimento guardado para poder visualizar los resultados.	
Entrada/Pasos de ejecución: Una vez en la interfaz de inicial hacer clic en el botón *"View experiment",* y en la interfaz que aparece seleccionar en el menú la opción de añadir un nuevo reporte.	
Resultado esperado: Debe mostrarse, en caso de que el usuario seleccione un reporte de un experimento de JCLAL, una gráfica donde se visualice el desempeño de los parámetros resultantes del experimento.	
Evaluación de la prueba: Prueba satisfactoria.	

2.6 Conclusiones Parciales

En el presente capítulo se abordaron todos los temas referentes a la metodología de desarrollo de software escogida en la construcción de la solución. Además, se presentó

el diseño de la arquitectura del módulo y las tareas que se llevaron a cabo para construirla.

La aplicación de la metodología en la implementación de la solución propuesta permitió el diseño y posterior desarrollo de la aplicación de forma rápida y organizada. Las fechas de entrega según lo planificado por el cronograma estuvieron acordes a la complejidad de desarrollo de las distintas tareas asignadas a cada iteración. El desglose de la implementación por iteraciones resulto de gran utilidad para simplificar el trabajo y organizar el período de pruebas.

La efectividad del desarrollo dirigido por pruebas y la aplicación de las pruebas de aceptación demuestran ser muy altas en el proceso de desarrollo de software. Ambas juegan un papel fundamental en el proceso de construcción del entorno de usuario con una metodología ágil, aun cuando el equipo de desarrollo es de un solo integrante.

ANEXO 1

CONCLUSIONES

El análisis de los fundamentos teóricos que rigen el Aprendizaje Activo y cómo se utilizan en el *framework* JCLAL, permitieron sentar las bases de conocimientos necesarios para que con la presente investigación se cumpliera el objetivo trazado.

La arquitectura y el grupo de tecnologías utilizadas favorecieron el análisis, diseño e implementación del entorno de usuario. Con el uso de la metodología ágil XP para guiar el proceso y de esta forma lograr construir un producto que se caracteriza por su fácil uso y extensión.

Las características y ventajas que ofrece el entorno de usuario *VisualJCLAL* logran un ahorro del tiempo y una interacción menos asistida en la experimentación usando el *framework* JCLAL, lo cual posibilita la aplicación y uso de JCLAL por más investigadores.

ANEXO 1

RECOMENDACIONES

Por los resultados obtenidos en esta investigación y para continuar el desarrollo de este trabajo se recomienda:

- Mejorar el diseño visual de la interfaz para lograr una mayor aceptación de los usuarios
- Agregar una opción que permita al investigador obtener una vista previa de la configuración XML dentro de la interfaz
- Hacer que *VisualJCLAL* sea compatible con versiones anteriores y posteriores a la 2.0 del *framework* JCLAL

ANEXO 1

REFERENCIAS BIBLIOGRÁFICAS

[1] A. Suárez, Aprendizaje Automático, 2005.

[2] E. P. Perdomo, Propuesta de un framework para el Aprendizaje Activo, Holguín, 2013.

[3] B. Settles, Active Learning Literature Survey, Wisconsin, 2010.

[4] Z. Zhou, M. Zhan, S. Huang y Y. Li, Multi-instance multi-label learning, 2011.

[5] O. Reyes, E. Pérez, S. Ventura, M. d. C. Hernandez y H. Fardoun, «JCLAL: A Java Framework for Active Learning,» 2016.

[6] H. Hoi, R. Jin y M. Lyu, Large-Scale Text Categorization by Batch Mode Active Learning, International World Wide Web Conference, 2006.

[7] D. Lewis y W. Gale, «A Sequential Algorithm for Training Text Classifiers,» 1994.

[8] R. Liere y P. Tadepalli, «Active Learning with Committees for Text Categorization,» 1997.

[9] Y. Zhang, J. Callan y W. Xu, «Exploration and Exploitation in Adaptive Filtering Based on Bayesian Active Learning,» 2003.

[10] A. Cullota, T. KRISTJANSSON, A. MCCALLUM y P. Viola, «Corrective Feedback and Persistent Learning for Information Extraction,» 2006.

[11] M. Becker, B. Hachey, B. Alex y C. Grover, «Optimising Selective Sampling for Bootstrapping Named Entity Recognition,» 2005.

[12] S. Argamon-Engelson y I. Dagan, «Committee-Based Sample Selection For Probabilistic Classifiers,» 1999.

[13] I. Dagan y S. P. Engelson, «Committee-Based Sampling For Training Probabilistic Classifiers,» 1995.

[14] E. Ringger y P. MCCLANAHAN, «Active Learning for Part-of-Speech-Tagging: Accelerating Corpus Annotation,» 2007.

[15] M. Becker y M. Osborne, «A Two-Stage Method for Active Learning of Statistical Grammars,» 2005.

[16] B. Hachey, B. Alex y B. Becker, «Investigating the Effects of Selective Sampling on the Annotation Task,» 2005.

[17] Y. Chan y H. NG, «Domain Adaptation with Active Learning for Word Sense Disambiguation,» 2007.

[18] G. Tur, D. Hakkani-T y R. Schapire, «Combining active and semi-supervised learning for spoken language understanding,» 2004.

[19] J. Kuo, H. Li y Y. Yang, « Learning Transliteration Lexicons from the Web,» 2006.

[20] M. Sassano, «An Empirical Study of Active Learning with Support Vector Machines for Japanese Word Segmentation,» 2002.

[21] E. Chang, S. Tong, K. Goh y C. Chang, «Support Vector Machine Concept-Dependent Active Learning for Image Retrieval,» 2005.

[22] B. Angluin, « Queries and Concept Learning,» 1988.

[23] D. Chon, «Neural Network Exploration Using Optimal Experiment Design,» Massachusetts, 1994.

[24] S. Tong y D. Koller, «Support Vector Machine Active Learning with Applications to Text Classification.,» 2001.

[25] H. Hoi, R. Jin y M. Lyu, «Batch Mode Active Learning with Applications to Text Categorization and Image Retrieval,» 2009.

[26] N. Roy y A. Mc Callum, Toward Optimal Active Learning through Sampling Estimation of Error Reduction, 2001.

[27] B. Settles y M. Craven, An Analysis of Active Learning Strategies for Sequence Labeling Tasks, 2008.

[28] C. Thompson, V. Hall y R. Mooney, Active Learning for Natural Language Parsing and Information Extraction, 1999.

[29] H. Riccardi y D. Hakkani, Active Learning: Theory and Applications to Automatic Speech Recognition, 2005.

[30] Y. Liu, Active Learning with Support Vector Machine Applied to Gene Expression Data for Cancer Classification, 2004.

[31] H. Seung, M. Opper y H. Sompolinsky, Query by Committee, 1992.

[32] F. Yoav, H. Seung, E. Shamir y N. Tishby, Selective Sampling Using the Query by Committee, 1997.

[33] D. Mackay, Information-Based Objective Functions for Active Data Selection, 1992.

[34] I. Witten y E. Frank, Data Mining, Nueva Zelanda, 2005.

[35] J. Álcala Fernandez, A. Férnandez, J. Luengo, J. Derrac, S. García, L. Sánchez y F. Herrera, KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework, 2010.

[36] M. C. Albornoz, Diseño de interfaz gráfica de usuario, 2014.

[37] I. Jacobson, G. Booch y J. Rumbaugh, El proceso Unificado de Software, 2000.

[38] D. Rosenberg y M. Stephens, Use Case Driven Object Modeling with UML, New York, 2007.

ANEXOS

Anexo 1

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<experiment>
<process evaluation-method-type="net.sf.jclal.evaluation.method.HoldOut">
   <rand-gen-factory seed="9871234" type="net.sf.jclal.util.random.RanecuFactory"/>
   <file-dataset type="net.sf.jclal.dataset.WekaDataset">
            <path>iris.arff</path>
           <class-attribute>1</class-attribute>
   </file-dataset>
    <percentage-split>66</percentage-split>
   <sampling-method type="net.sf.jclal.sampling.unsupervised.Resample">
     <percentage-to-select>5</percentage-to-select>
     <no-replacement>true</no-replacement>
    </sampling-method>
 <algorithm type="net.sf.jclal.activelearning.algorithm.ClassicalALAlgorithm">
      <listener type="net.sf.jclal.listener.GraphicalReporterListener">
               <report-title>Default test</report-title>
               <report-frequency>1</report-frequency>
               <report-directory>reports/default test</report-directory>
               <report-on-console>false</report-on-console>
               <report-on-file>true</report-on-file>
               <show-passive-learning>false</show-passive-learning>
               <show-window>false</show-window>
        </listener>
       <stop-criterion type="net.sf.jclal.activelearning.stopcriteria.MaxIteration">
               <max-iteration>2</max-iteration>
        </stop-criterion>
   <stop-criterion type="net.sf.jclal.activelearning.stopcriteria.UnlabeledSetEmpty"/>
   <scenario type="net.sf.jclal.activelearning.scenario.PoolBasedSamplingScenario">
   <batch-mode type="net.sf.jclal.activelearning.batchmode.QBestBatchMode">
```

```
                    <batch-size>1</batch-size>
   </batch-mode>
   <oracle type="net.sf.jclal.activelearning.oracle.SimulatedOracle"/>
    <query-strategy type="...singlelabel.querystrategy.EntropySamplingQueryStrategy">
       <wrapper-classifier type="net.sf.jclal.classifier.WekaClassifier">
             <classifier type="weka.classifiers.functions.SMO"/>
        </wrapper-classifier>
    </query-strategy>
    </scenario>
  </algorithm>
 </process>
</experiment>
```