UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

TRABAJO DE FIN DE MÁSTER

MÁSTER UNIVERSITARIO EN INGENIERÍA Y CIENCIA DE DATOS

MONTAJES DE EQUIPOS DE TRABAJO UTILIZANDO TÉCNICAS DE APRENDIZAJE AUTOMÁTICO NO SUPERVISADO

Tatiana Guanangui Guamán

Dirigido por: Dr. Antonio Rodríguez Anaya

Curso: 2023-2024 – 1ra Convocatoria

Resumen

Los equipos de trabajo son la unidad básica organizada para afrontar retos en cualquier ámbito. Actualmente se dispone de datos sobre las conexiones sociales que los individuos desarrollan en las redes, misma que es aprovechada para optimizar y personalizar las soluciones al problema de formación de equipos (Team Formation Problem TFP).

Hasta el momento, las soluciones basadas en redes sociales han considerado a la comunicación como elemento central para la formación de equipos; sin embargo, en varios estudios se menciona indirectamente a la cohesión (familiaridad, química, sinergia, cooperación interpersonal) como un elemento significativo, por lo tanto, la propuesta que se plantea incluye a la comunicación y cohesión como criterios para formar equipos y asimismo bosqueja una forma de evaluar la cohesión mediante el contenido y características de los mensajes intercambiados.

El contexto de uso de esta propuesta es el entorno académico en línea, específicamente estudios de maestría. Los mensajes compartidos en los foros virtuales sirvieron como insumo, y se utilizó análisis de redes sociales junto a una técnica de aprendizaje automático no supervisado (K-means), para dar lugar a una propuesta de tipo exógena que permita al tutor formar equipos optimizados que puedan cumplir un proyecto durante el curso. El clustering efectuado con cada criterio posibilitó hallar grupos; según la comunicación pueden existir estudiantes Comunicativos y Poco Comunicativos; y según la cohesión: Unificadores, Diplomáticos e Indiferentes. A continuación, se programan dos estrategias para la formación de equipos: a) estrategia armoniosa: busca maximizar la comunicación y cohesión; y b) estrategia extrema: pretende reunir estudiantes discordantes.

Los resultados de este trabajo permiten concluir que el TFP debe solucionarse atendiendo al contexto, los objetivos del trabajo, las habilidades requeridas, el tipo de tarea y criterios específicos a ser optimizados, de modo que, una estrategia adaptable a distintos contextos y criterios no es del todo factible. Además, se promueve el uso de la cohesión como un criterio relevante a considerar en la formación de equipos.

Palabras clave: Formación de equipos, redes sociales, clustering, K-means, comunicación, cohesión.

Abstract

Teams are the basic organized unit to face challenges in any field. Data is now available on the social connections that individuals develop in networks, which is used to optimize and customize solutions to the Team Formation Problem (TFP).

So far, solutions based on social networks have considered communication as a central element for team formation; however, in several studies cohesion (familiarity, chemistry, synergy, interpersonal cooperation) is indirectly mentioned as a significant element, therefore, this proposal includes communication and cohesion as criteria for team formation and also outlines a way to evaluate cohesion through the content and characteristics of the messages exchanged.

The context of use of this proposal is the online academic environment, specifically master's degree studies. The messages shared in the virtual forums served as input, together with the use of unsupervised automatic learning techniques (K-means) and social network analysis, to give rise to an exogenous type proposal (the tutor chooses the strategy) that allows the tutor to form optimized teams that can complete a project during the course. The clustering carried out with each criterion made it possible to find groups; according to communication there can be Communicative and Poorly Communicative students; and according to cohesion: Unifying, Diplomatic and Indifferent. Two strategies for team formation are then programmed: a) harmonious strategy, which seeks to maximize both communication and cohesion; and b) extreme strategy, which seeks to bring together discordant students.

The results of this work allow us to conclude that the TFP must be solved according to the context, the objectives of the work, the skills required, the type of task and specific criteria to be optimized, so that a strategy adaptable to different contexts and criteria is not entirely feasible. In addition, the use of cohesion is promoted as a relevant criterion to be considered in team building.

Keywords: Team Formation Problem, social networks, clustering, k-means, communication, cohesion.

Agradecimientos

Agradezco a Dios por la oportunidad de estudiar y prepararme para servir mejor. También expreso gratitud a mi familia por su apoyo incondicional. Y presento un reconocimiento especial al tutor de este trabajo por cumplir a cabalidad su rol con excelente disposición, ánimo y profesionalismo de principio a fin.

Soli Deo gloria – Solo a Dios sea la gloria.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

1. Introducción, motivación y objetivos

Este primer capítulo tiene como objetivo presentar al lector el propósito de este Trabajo de Fin de Máster (TFM). Por este motivo, se inicia con la exposición del tema principal y los objetivos de este trabajo, a continuación, se explican los métodos usados para su desarrollo y la organización de la memoria que resume el trabajo realizado.

1.1. Motivación

El trabajo en equipo constituye el mecanismo a través del cual las empresas deben operar para la consecución de objetivos más grandes y complejos, que se caracterizan principalmente por el uso importante de tecnología, la convergencia de múltiples disciplinas en los proyectos y la ubicación geográfica de los participantes. De forma general, la formación de un equipo estará subordinada a los objetivos del proyecto y a las habilidades requeridas en los participantes.

Los estudios de las ciencias psicológicas por una parte avanzan y afinan sus análisis sobre características particulares para formación de equipos especializados: deportivos, militares, estudio, cirugía, etc. A la par, las tecnologías proveen insumos importantes para probar las nuevas teorías: datos de redes sociales para validar la interacción entre miembros de los equipos y variadas técnicas para resolver el problema de formar un equipo capaz de llevar a cabo el objetivo asignado.

No es difícil apreciar que la formación de un equipo de trabajo es un problema no trivial en constante evolución y que ofrece un amplio abanico de características a considerar en su solución. Durante la última década las herramientas para la formación de equipos han evolucionado y son usadas de forma activa por ciertas comunidades, por ejemplo, en el área educativa (CATME y marco de trabajo Team-Maker[1]). Las mejoras en estas aplicaciones se concentran en su adaptabilidad y uso amigable a través de la flexibilidad en los tipos de restricciones a considerar durante la formación del equipo.

[1] https://www.catme.org/

El contexto académico es el escenario donde se ubica este trabajo de investigación, específicamente en los ambientes educativos en línea. En este tipo de entorno, el ensamblaje de equipos es un asunto importante porque las actividades colaborativas y cooperativas que realizan los miembros del equipo son primordiales dentro del proceso de aprendizaje. En los ambientes educativos en línea, la estrategia para formación de equipos debe considerar las interacciones entre los estudiantes, porque estas pueden ayudar a valorar cómo se lleva a cabo el proceso de aprendizaje colaborativo. (Anaya A.R., Letón E., Luque M., 2023)

El estudio de Anaya, Letón y Luque (2023) plantea que el ensamblaje de equipos podría basarse en el modelamiento del comportamiento social de los estudiantes y ser conducido por el criterio de expertos. (Anaya A.R., Letón E., Luque M., 2023) Dentro de este estudio, se menciona el trabajo realizado por Liu Z., Kang, Domanska, Liu S., Sun y Fang (2018), el cual reveló una correlación positiva significativa entre la centralidad de red de un estudiante y sus resultados, por ejemplo: estudiantes de alto desempeño tenían alto prestigio[2] e influencia[3] en la red. (Liu Z., Kang L., Domanska M., Liu S., Sun J., & Fang C., 2018) Estos estudios dan luces sobre la importancia del análisis del comportamiento social de los estudiantes en provecho de una mejor experiencia de aprendizaje.

Retomando el tema de la educación en línea, es necesario mencionar que la eficacia de la enseñanza en línea en niveles superiores está relacionada a varios aspectos, entre ellos, ofrecer posibilidades de colaboración y trabajo cooperativo entre los alumnos, además de proponer actividades relevantes para la práctica. (Castro M., Tumibay G., 2019)

Profundizando en el tema de las interacciones entre estudiantes, en la enseñanza superior, se subraya la importancia de formar una comunidad de aprendizaje a través de la creación de grupos pequeños y facilitar canales de comunicación efectivos entre alumnos moderados por los docentes. (Tallent-Runnels M., Thomas J., Lan W., Cooper S., Ahern T., Shaw S., Liu X., 2006)

En cuanto a los factores que afectan al desempeño y satisfacción de los alumnos en experiencias de enseñanza en línea en el nivel superior, se encuentra que los

[2] Prestigio se refiere a recibir muchos enlaces, en este caso mensajes.
[3] Influencia se refiere a que tiene muchas conexiones directas con otros.

alumnos aprecian los cursos interactivos, que apuestan por una participación activa y ofrecen oportunidades para la colaboración entre pares e intercambio de ideas. En cuando al desempeño, los cursos que favorecen mejores aprendizajes son interactivos, estimulan la colaboración y facilitan la reflexión durante todo el proceso de aprendizaje. (Kauffman H., 2015)

Como se puede apreciar, el tema de la formación de equipos es un tema relevante en el desarrollo de la educación en línea. Para la enseñanza en el nivel superior, el trabajo colaborativo es un aspecto constantemente señalado como una herramienta que propicia tanto la satisfacción como el desempeño de los estudiantes. Por tanto, la motivación de este trabajo es aportar con una propuesta para la formación de equipos de trabajo a partir de las interacciones que los estudiantes desarrollan en los foros virtuales en estudios de maestría.

El incentivo de este trabajo es proveer al tutor de: 1) criterios para formación de los equipos, medidos a través de la caracterización de los estudiantes y los mensajes intercambiados, y 2) poner a su consideración un par de estrategias para ensamblar equipos. Se estima que con este trabajo se podría aportar al mismo tiempo, al desempeño y satisfacción de los alumnos.

En este trabajo se plantea la realización de estrategias de montajes de equipos de utilizando técnicas de aprendizaje automático no supervisado y análisis de redes sociales y se presume que las estrategias pueden ser adaptables a diferentes contextos y criterios. De aquí que una pregunta que se intenta responder en este trabajo es: ¿Es posible crear una propuesta adaptable a diferentes contextos y criterios de ensamblaje?

1.2. Objetivos

Para llevar a cabo la propuesta de montaje de los equipos de trabajo usando técnicas de aprendizaje no supervisado, se identificaron los siguientes objetivos y subobjetivos, que constan en la Tabla 1:

Objetivo 1: Documentar técnicas de montaje de equipos en entornos informáticos. **Subobjetivos:** 1.1 Comprender las características del proceso de formación de equipos. 1.2 Conocer las taxonomías y criterios para formar equipos. 1.2 Conocer los enfoques computacionales para la formación de equipos.
Objetivo 2: Plantear una propuesta de montaje de equipos en entornos informáticos. **Subobjetivos:** 2.1. Identificar la taxonomía y criterios a emplear para la formación de equipos. 2.2. Analizar la técnica supervisada a emplear. 2.3. Analizar los conjuntos de datos disponibles.
Objetivo 3: Diseñar una solución informática que pueda adaptarse a diferentes contextos de trabajo o criterios de agrupación. **Subobjetivos:** 3.1. Definir las variables asociadas a los criterios para la formación de equipos. 3.2. Plantear la(s) estrategia(s) para la formación de equipos. 3.3. Señalar ventajas y desventajas de las estrategias propuestas.

Tabla 1 Objetivos y subobjetivos del TFM

1.3. Métodos y materiales

El primer objetivo del TFM consistirá en documentar las técnicas de montaje de equipos que actualmente se emplean para formar equipos. Se hará uso de la bibliografía propuesta por el director, más la investigación de trabajos relacionados a este tema, que se encuentren disponibles a la fecha. El objetivo es identificar las características más relevantes a considerar en el proceso de montaje de equipos de trabajo, así como las técnicas de aprendizaje más apropiadas para solucionar el problema.

Los estudios analizados dentro del marco teórico en el área computacional fueron considerados desde el año 2019 en adelante. Fueron excluidos artículos relacionados a gestión de equipos, reemplazo de miembros de un equipo, sistemas multiagente

(multi-agent systems MAS) o equipos robóticos o de Inteligencia Artificial (Artificial Intelligence IA), si estos se orientaban a desarrollar su estudio sobre un equipo preexistente o formado previamente. También se descartaron aquellos estudios que concentraban su estudio exclusivamente en agentes IA. Al final se revisaron cerca de 23 artículos.

Además, en la mayoría de los artículos revisados el contexto de formación de equipos tuvo lugar en campos de la ingeniería, desarrollo de software y ciencias computacionales. En la mayoría de los estudios revisados prima la formación de equipos de trabajo con enfoques basados en redes.

Debido a que el problema de la formación de equipos es un tema multidisciplinario, no se puede dejar de lado la revisión de material de las ciencias organizacionales para comprender las taxonomías, los criterios para formar equipos y los factores sociales que intervienen en este problema. Por lo tanto, en esta área, se han considerado referencias bibliográficas desde el año 2000 en adelante.

A continuación, para cumplir con el segundo objetivo, se busca, esquematizar los elementos y relaciones que deben existir para el montaje de equipos y añadir la técnica de aprendizaje no supervisada que permita constituir equipos de trabajo a partir de los datos de individuos en un entorno específico. La propuesta debe ser adaptable a distintos contextos de trabajo y flexible al considerar diferentes criterios para la formación de equipos.

Finalmente, el tercer objetivo, es aterrizar y poner a prueba la propuesta obtenida en el punto anterior. La propuesta será implementada usando herramientas de análisis de datos como Gephi y Python. Como insumo para el análisis de datos se consideran los conjuntos de datos (datasets) de mensajes intercambiados en los foros virtuales por estudiantes de las materias Aprendizaje Automático (AA) y Minería de Datos en las Redes Sociales (MDMS) para los períodos 2019-2020; 2020-2021 y 2021-2022. Estos datos se encuentran anonimizados y fueron provistos por el director del TFM.

1.4. Estructura de la memoria

La memoria de este trabajo se estructura en los siguientes capítulos:

1. Introducción, motivación y objetivos. Expone la motivación y objetivos del presente trabajo.
2. Estado del arte. Desarrolla el marco teórico de este trabajo, dando a conocer las ideas y conceptos analizados en este trabajo.
3. Desarrollo de la propuesta. Presenta las actividades de diseño y desarrollo para llevar a cabo la propuesta de formación de equipos. Además, se realiza la evaluación con los conjuntos de datos disponibles.
4. Conclusiones y trabajos futuros. Señala las conclusiones a las que se arriba luego de la realización de este trabajo, e indica posibles trabajos futuros relacionados al mismo.

1.5. Relación entre los objetivos del trabajo y la estructura de la memoria

En la Tabla 2 se esquematizan los objetivos asociados a las secciones que componen la memoria:

Objetivo	**Capítulo**
#1 Documentar técnicas de montaje de equipos en entornos informáticos.	# 2 Estado del arte
#2 Plantear una propuesta de montaje de equipos en entornos informáticos. #3 Diseñar una solución informática que pueda adaptarse a diferentes contextos de trabajo o criterios de agrupación. Subobjetivos	#3 Análisis, diseño, implementación y evaluación de la propuesta.

Tabla 2 Objetivos del TFM y estructura de la memoria

Capítulo 2

2. Estado del arte

A medida que las organizaciones crecen y el avance de la tecnología añade complejidad y a la vez dinamismo al quehacer de estas, se hace muy necesario particionar los problemas y enfrentarlos con un enfoque de trabajo de equipo en lugar de trabajo individual.

En la actualidad, no se podría discutir que la capacidad de resolver problemas y aportar soluciones es siempre mayor en un equipo que en una sola persona (Ander-Egg E., Aguilar, M., 2001). Sin embargo, es importante observar que la reunión de varias personas con una meta no siempre produce un resultado de equipo de trabajo, por lo tanto, cabe una apreciación: el trabajo en equipo debe ser eficaz.

2.1. Equipo de trabajo

Un equipo puede ser definido como un grupo de individuos con roles y responsabilidades específicas que interactúan de forma adaptativa, interdependiente y dinámica (identidad del equipo) hacia un resultado que valoran en común (trabajo en equipo); y que están integrados en un sistema organizativo (contexto del equipo) con límites y vínculos hacia un contexto más amplio del sistema y del entorno de la tarea. (Salas E., Sims D., Burke Sh., 2005) (Hoegl M., Gemuenden H., 2001)

Un equipo de trabajo tiene una meta común y para llevarla a cabo durante un tiempo determinado, ocurre un proceso gradual donde sus miembros se comunican; comparten criterios, ideas, opiniones; realizan un trabajo colaborativo donde las personas se ayudan entre sí; y resuelven divergencias.

2.1.1. Trabajo en equipo

El trabajo en equipo se refiere a los procesos conductuales que los miembros del equipo utilizan para lograr el trabajo dentro del equipo, por ejemplo: comunicación, colaboración, intercambio de experiencias. (Salcinov B., Drew M., Dijkstra P., Waddington G., Serpell B., 2022)

2.1.2. Ventajas

Los equipos de trabajo son usados en organizaciones, diversos sectores e industrias debido a que alcanzan un mejor desempeño que los individuos, particularmente cuando se necesitan múltiples habilidades y criterios. Los equipos pueden ser usados para alcanzar las necesidades humanas de afiliación descritas en la teoría de jerarquía de necesidades de Maslow (Frager, 1987).

Los equipos de trabajo tienen el potencial para incrementar la productividad, que frecuentemente resulta en un aumento considerable porque se puede obtener mejores resultados que cuando los individuos actúan en roles definidos. Dentro de los equipos tiende a existir una variedad de conocimiento y habilidades que pueden ser agrupadas con información y recursos que pueden ser compartidos. Los equipos de trabajo posibilitan el uso complementario de habilidades disponibles para alcanzar una productividad más alta. (Baiden B., Price A., 2011)

2.2. Formación del equipo de trabajo

Un modelo, de entre algunos, que ayuda a entender la evolución social de un equipo es el propuesto por el psicólogo Bruce Tuckman. En 1965, Tuckman publicó un ensayo titulado “Developmental sequence in small groups”; el mencionado artículo describe cómo un grupo de extraños atraviesan cuatro etapas para convertirse en un equipo que hace un trabajo eficaz[4]. En 1977, la investigación de Tuckman fue mejorada con la colaboración de Mary Ann Conover (Tuckman, B.W., and M.A. Jensen., 1977), para añadir una quinta etapa para la formación de equipos, quedando las cinco etapas definidas de la siguiente manera: 1) Formación; 2) Conflicto; 3) Normalización; 4) Desempeño; 5) Disolución. A continuación, se describen brevemente las cuatro etapas: (Jones A., 2019)

1) **Formación**. El equipo se forma de acuerdo con las necesidades del proyecto. Los integrantes intentan ser aceptados y conocen al resto de participantes. Pueden aparecer comportamientos individualistas, intentando destacar. Existe

[4] https://alaimolabs.com/es/self-learning/agile-coaching/el-modelo-de-tuckman-para-equipos-aplicado-a-facilitacion

inseguridad y falta de confianza. Existe un nivel de comunicación bajo y alta dependencia del líder.

2) **Conflicto**. Los miembros del equipo tienen una mayor confianza y conocen la forma de trabajo y personalidad de cada participante. Se expresan diferencias o pujas de interés por las acciones y opiniones del resto e incluso del líder. Es una etapa agitada, pero necesaria para la consolidación del equipo.

3) **Normalización**. Se conoce el propósito del equipo con claridad y existe una forma establecida para el trabajo. Cada miembro desarrolla su responsabilidad para contribuir al éxito del equipo. Las normas establecidas son respetadas por todos los miembros; el equipo está altamente integrado y comprometido. El equipo se encuentra cohesionado.

4) **Desempeño**. Los miembros del equipo trabajan como un reloj, tienen alto rendimiento y bajo desperdicio. El equipo ha aprendido a tomar decisiones incluso sin requerir supervisión externa, el trabajo se desarrolla sin problemas. Los niveles de motivación y moderación son altos, así como el orgullo y sentido de pertenencia al grupo.

5) **Disolución**. El equipo se disuelve y sus miembros pueden desarrollar nuevas tareas en otros proyectos. El líder debe ayudar en la gestión de la pérdida, que pueden desarrollar algunos miembros.

El modelo de Tuckman permite apreciar que los equipos atraviesan varias etapas en las que se presentan ciertas características en las relaciones sociales de los participantes y la actitud que cada uno demuestra al cumplir sus tareas. Cabe mencionar que el modelo de Tuckman muestra el proceso que se cumple dentro del equipo cuando los miembros ya están designados y arrancan para cumplir una o varias metas.

2.3. Factores sociales que considerar para la formación de un equipo

Usualmente cuando se forma un equipo de trabajo, las habilidades técnicas (hard skills) son las características más consideradas por los jefes de proyectos. Sin

embargo, estudios recientes han argumentado que el éxito o fracaso de un equipo frecuentemente incluye otros factores como: a) la interdependencia de las competencias del equipo; b) integración; c) confianza; y d) las habilidades técnicas de cada miembro. Además, un nivel apropiado de cohesión es necesario para reunir a la gente y hacerlos colaborar para construir la base de un trabajo de calidad. (Hoegl M., Gemuenden H., 2001)

Otros estudios señalan la importancia del trabajo en equipo (Teamwork) para el éxito de proyectos innovadores. Para los autores, Hoegl y Gemuenden (Hoegl M., Gemuenden H., 2001), un equipo puede ser definido como un sistema social de tres o más personas que están inmersas en una organización (contexto), cuyos miembros se perciben a sí mismos como tales y son percibidos como miembros por otros (identidad), y juntos colaboran en una tarea común (trabajo en equipo). El trabajo en equipo es una construcción teórica que proponen los autores, como un concepto integral de la calidad de las interacciones en equipo. El constructo trabajo en equipo captura la naturaleza de los miembros del equipo trabajando juntos mediante seis dimensiones del proceso colaborativo en equipo y que se integran para dar lugar al concepto calidad del trabajo en equipo (Team Work Quality - TWQ)

Las dimensiones capturan las interacciones sociales del equipo y también aquellas relacionadas a la tarea. La calidad de las interacciones con partes externas no es parte del TWQ. Las dimensiones del TWQ son: comunicación, coordinación, balance de las contribuciones de los miembros, soporte mutuo, esfuerzo y cohesión. En resumen, el aspecto relevante es la calidad del trabajo colaborativo del equipo, más allá del contenido de sus tareas y actividades y cuán bien pueden transmitir información para desarrollar sus asignaciones. Por lo tanto, las medidas del proceso de las tareas del equipo, estrategia usada y procesos de liderazgo no son parte del constructo TWQ. A continuación, se describe un breve concepto de cada dimensión:

- **Comunicación**. El componente más elemental del TQW. La comunicación provee un medio para el intercambio de información entre miembros del equipo. La calidad de la comunicación puede ser descrita en términos de frecuencia, formalización, estructura y apertura al intercambio de información.

- **Coordinación**. El grado de entendimiento común respecto de la interrelación y el estado actual de las contribuciones individuales determinan la calidad del trabajo realizado en equipo.
- **Balance de las contribuciones de los medios**. Es importante para la calidad del trabajo en equipo que todos los miembros del equipo sean capaces de aportar con su conocimiento y experiencia para las tareas relevantes.
- **Soporte mutuo**. La colaboración intensiva de los individuos descansa en un estado de ánimo cooperativo en lugar de uno competitivo.
- **Esfuerzo**. Las normas son definidas como experiencias compartidas relacionadas al comportamiento de los miembros del equipo (Levine J., Moreland R., 1990).
- **Cohesión**. Se refiere al grado de anhelo que manifiesta cada miembro por ser parte del equipo.

En el estudio, los autores concluyen que el TWQ es un factor de éxito importante para los proyectos innovadores. Cuando las tareas tienden a ser más innovadoras el TQW cobra más importancia para el desempeño del equipo. Mientras que, cuando las tareas son rutinarias, la calidad del trabajo en equipo puede tener menor efecto en el desempeño, en tanto que otros factores se hacen más relevantes, por ejemplo: relaciones externas, contexto organizacional y procesos del conocimiento.

En algunos estudios de la teoría sociológica se señala a la cohesión como una variable importante en la aparición del consenso entre los miembros del equipo. Aún más relevante, la cohesión es directamente responsable del impacto positivo en el ambiente de trabajo, influenciando la motivación del equipo, moral, coordinación de esfuerzos, productividad, satisfacción y cooperación. (Dwivedula R., Bredillet N., 2010). La teoría de la identidad social señala que mientras más una persona se identifica a sí misma como miembro de un equipo, más activamente contribuirá a alcanzar la meta común. (Maurer, 2010)

La cohesión está definida como el grado en el cual los individuos se sienten aceptados o rechazados por un grupo. (Beal D. J., Cohen R. R., Burke, M. J., & McLendon, C. L., 2003). El resultado alcanzado por el equipo depende grandemente de la manera en la cual los individuos desarrollan sus relaciones e interacciones. En

consecuencia, es posible maximizar la cohesión de algunos grupos si se reúne a ciertas personas y se puede esperar que el proyecto muestre un óptimo desempeño. (Esgario J., Da Silva I., Krohling R., 2019)

El grado de cohesión de un grupo puede ser obtenido a través de varios métodos, tales como: pruebas sociométricas y estudios de ambiente de trabajo. En la prueba sociométrica cada miembro acepta o rechaza al resto de los miembros. Los resultados de la prueba sociométrica pueden ser obtenidos mediate las siguientes preguntas: 1) ¿Con qué empleados te gustaría trabajar? 2) ¿Con qué empleados no te gustaría trabajar? (Esgario J., Da Silva I., Krohling R., 2019)

2.4. Criterios para la formación de equipos

La formación de equipos puede entrañar diversos criterios para su ejecución. A partir de los artículos analizados se desarrolló la Tabla 3, que resume los hallazgos en cuanto a los criterios propuestos por diversos autores:

Criterio	Descripción
Participación externa/interna para la formación	Exógena (agentes externos escogen los miembros del equipo); o endógena (los miembros deciden participar o no en un equipo).
Contexto	Donde se requiere formar un equipo: desarrollo de software, unidades militares, equipos de cirugía.
Atributos críticos para el desempeño	Cualidades primordiales que debe poseer un equipo (habilidades duras o suaves). Por ejemplo: la cohesión para los líderes del ejército de los Estados Unidos es fundamental para construir equipos. (Boyce-Jacino C., Harrington N., 2022).
Cantidad de equipos a formar	Uno solo equipo o múltiples equipos.
Optimizar ciertas variables del equipo.	Ejemplos: minimizar costo de comunicación; minimizar el costo de constituir el equipo; minimizar la distancia geográfica; minimizar el costo de cohesión; minimizar el potencial de falla; optimizar el balance de la carga de

	trabajo; maximizar la heterogeneidad; maximizar el ajuste del candidato al rol; maximizar el desempeño del equipo de trabajo.
Maximizar las salidas o resultados del equipo	Promover mayor efectividad o desempeño de las salidas que entrega el equipo.
Miembros del equipo en puestos específicos.	Participantes en posiciones específicas dentro del equipo con o sin posibilidad de redundancia.
Presencia de un líder o mediador	Identificación o no de un líder o mediador en el equipo.
Participación de un miembro en uno o varios equipos	Los miembros deben asignar una fracción de tiempo a cada equipo en el que fueron asignados. Este problema se conoce como la formación de múltiples equipos (Multiple Team Formation Problem - MTFP), (Gutiérrez H., Astudillo C., Ballesteros-Pérez P., Mora-Meliá D., Candia-Vejar A., 2016))
Rasgos de la personalidad y género.	Características de pruebas de personalidad (como el indicador de tipo Myers-Briggs; Myers-Briggs Type Indicator MBTI)
Naturaleza de la tarea	Creativa, repetitiva; estructurada o abierta.

Tabla 3 Posibles criterios para formar un equipo de trabajo

Cabe anotar que cada uno de los criterios señalados en la tabla pueden verse matizados por las restricciones que cada autor decide añadir a su estudio de la formación de equipos. Por ejemplo, cuando Dorn y Dustdar (Dorn C., Dustdar S., 2010). extrajeron un grupo de expertos de una red, buscaron cumplir el criterio de las habilidades requeridas para cada miembro, pero hicieron una variación con respecto al estudio de Lappas, Liu y Terzi (Lappas T., Liu K., & Terzi E., 2009), el cual además exigía establecer un umbral para las habilidades de los miembros. La variación de Dorn et al. (2010) consistió en equilibrar la compatibilidad social y la habilidad. Por otra parte, a diferencia del trabajo de Lappas et al. (2009) que requería enfocarse en el vínculo más fuerte, fue demandado que cada miembro del equipo resultante estuviera conectado uno con otro.

Otro estudio que toma prestada la definición del costo de comunicación de Lappas et al. (2009) es el realizado por Anagnostopoulos, Becchetti, Castillo, Gionis y Leonardi (2012), quienes consideraron individuos con habilidades binarias (cuenta o no con ella) y además realizaron un enfoque más realístico permitiendo que los miembros estén conectados por una relación de segundo o tercer nivel, y no necesariamente por una relación directa para tener una buena coordinación. (Anagnostopoulos A., Becchetti L., Castillo C., Gionis A., & Leonardi S., 2012)

Es importante considerar que cada estudio sobre la formación de equipos puede definir otros criterios además de los anteriormente señalados, e incluso introducir nuevas formas para definir una variable, o el costo de la variable, considerando los datos que tiene disponibles.

Cuando un tomador de decisiones (ente exógeno) considera las conexiones sociales y de comunicación, puede reunir diferentes individuos que cohesionan más rápidamente, crean puntos en común y construyen modelos mentales compartidos. Por otro parte, sin tener en cuenta los vínculos sociales, podría encontrarse con equipos que tienen costos iniciales sociales más altos.

Los autores Bahargam, Goshan, Lappas y Terzi (2019) plantearon que, al formar equipos bajo una perspectiva endógena, se puede examinar el potencial de colaboración de un equipo a través de la minimización del potencial de línea de falla (faultline potential). (Bahargam S., Golshan B., Lappas T., & Terzi E., 2019) Las líneas de falla del equipo son descritas como líneas divisorias que fraccionan al grupo en subgrupos homogéneos basados en diferencias de atributos (por ejemplo: la edad). Las líneas de falla han sido observadas por su efecto en la cohesión y desempeño del equipo, pero han probado ser difíciles de medir en alguna forma que pueda aprovecharse para la formación del equipo. (Boyce-Jacino C., Harrington N., 2022).

El estudio de Bahargam et al. (2019) propone minimizar dentro del equipo los conflictos triangulares (Conflict Triangles CT); su trabajo consideró características superficiales (que se miran a simple vista) como: raza, género o profesión. Por ejemplo, un conflicto de triángulo captura la intuición de que dos personas del mismo país tienen más probabilidades de interactuar entre sí que con una tercera persona de otra nacionalidad, y esto justamente permite la creación de una línea de falla potencial. Por otra parte, una línea de falla nunca podría ocurrir por la característica

de género, si las tres personas presentan el mismo valor (todos hombres o mujeres). La teoría de las líneas de falla indica que estas no pueden emerger en la presencia de homogeneidad o diversidad perfectas. El algoritmo propuesto por Bahargam et al. (2019) inicia con una partición aleatoria de la población en un número de grupos y luego reasigna de forma iterativa a los individuos dentro de los grupos hasta que el potencial de línea de falla de las particiones no mejore a través de las iteraciones (similar al algoritmo K-means). (Bahargam S., Golshan B., Lappas T., & Terzi E., 2019)

Por otra parte, algunos estudios han documentado los beneficios de la diversidad en un equipo, por ejemplo, Horwitz y Horwitz (2007) hallaron evidencia del impacto positivo de la diversidad relacionada con tareas en el desempeño del equipo. Además, encontraron que la diversidad demográfica no estaba significativamente relacionada con el desempeño del equipo, así también, no hallaron un efecto apreciable de la diversidad del equipo en la integración social. (Horwitz S., Horwitz I., 2007). Reuniendo las ideas de la teoría de las líneas de falla y la diversidad en los equipos, se podría considerar que los algoritmos diseñados por los investigadores podrían complementarse para dividir una gran población en equipos donde se valore la diversidad, pero se minimicen las líneas de falla. (Boyce-Jacino C., Harrington N., 2022)

Aunque Bahargam et al. (2019) usó propiedades externas para modelar las líneas de falla, los investigadores de la Psicología sugieren que las características superficiales son solamente de particular importancia en las etapas tempranas de formación de un equipo. Es más probable que los atributos de nivel profundo (por ejemplo: personalidad, preferencias de trabajo en equipo, valores) continúen interactuando e influenciando los procesos y la efectividad del equipo a lo largo del tiempo. (Bell S., Brown S., Colaneri A., & Outland N., 2018)

2.5. Integrando perspectivas de las ciencias psicológicas

2.5.1. Habilidades, atributos individuales

Se mencionó previamente que una forma común de optimizar la asignación de individuos a un equipo es considerando sus habilidades. Un equipo óptimo tiene

redundancia en las habilidades de sus miembros para poder completar la tarea. Sin embargo, la habilidad individual para contribuir a un equipo o a una tarea individual va más allá del nivel de sus habilidades e incluye atributos tales como sus destrezas cognitivas, personalidad y género. (Lykourentzou I., Antoniou A., Naudet Y., & Dow S. , 2016) En el caso de la personalidad, uno de los retos que enfrentan las ciencias computacionales es la dificultad en la obtención de las mediciones.

Aunque varias de las pruebas de personalidad han sido automatizadas y mejoradas, siempre será un requisito previo que los participantes deban completarlas para obtener información sobre sus rasgos de personalidad. Los esfuerzos para formalizar relaciones específicas entre personalidad y roles de equipo también carecen de herramientas y marcos de trabajo consensuados. Sin embargo, los estudios de la psicología sugieren que las características de un individuo lo motivan y habilitan para ocupar algunos roles más efectivamente que otros. (Boyce-Jacino C., Harrington N., 2022)

Más allá del ajuste que se da entre la persona y el rol, ciertas distribuciones de rasgos de personalidad son críticas para la efectividad de un grupo. Sin embargo, otros estudios añaden que no resulta del todo necesario balancear las personalidades, sino únicamente equilibrar ciertas características. Por ejemplo, el estudio de Halfhill, Sundstrom, Lahner, Calderone y Nielsen (2005) encontró que los rasgos de personalidad orientados a la relación, tales como la amabilidad, predijeron el desempeño del equipo más allá que los rasgos orientados a la tarea tales como la motivación por el logro. (Halfhill T., Sundstrom E., Lahner J., Calderone W., & Nielsen T., 2005) Otro autor, Bell (2007) señaló que particularmente en el caso de la amabilidad: un mínimo de amabilidad es más importante que tener amabilidad promedio al conformar un equipo; basta un solo miembro desagradable para alterar el equilibrio de un equipo, que de otro modo sería agradable. (Bell, 2007)

Cabe mencionar que los hallazgos para grupos en la esfera civil no necesariamente aplican para grupos en otros terrenos, como el marcial, por ejemplo. En el campo militar, las relaciones entre características de la personalidad y los resultados del equipo deben ser comprendidas y agregadas para crear equipos que se desempeñen de forma óptima. Por ejemplo, aunque la alta extroversión en un grupo es típicamente un predictor de alto desempeño del equipo, puede resultar contraproducente para

equipos operando en ambientes austeros o aislados. (Palinkas L., Gunderson E., Holland A., Miller C., & Johnson J. C., 2000).

La personalidad es una forma de representar la compatibilidad entre los miembros de un equipo, pero existen otros factores que impactan las relaciones interpersonales y que a su vez influyen en el desempeño del equipo. Algunos autores señalan el costo de la comunicación como una forma de capturar el potencial para la colaboración efectiva entre los miembros del equipo. El costo de la comunicación ha sido representado por eventos de colaboración (por ejemplo, Lappas et al. (2009) empleó una base de artículos científicos para buscar coautorías) y en otras ocasiones por métricas de las redes sociales como grado, cercanía, diámetro, suma de distancias, etc. El costo de comunicación podría ser además representado por la riqueza de otras fuentes de datos, por ejemplo: historia de trabajos realizados en equipo o ubicaciones geográficas de trabajo. (Boyce-Jacino C., Harrington N., 2022)

Entre los psicólogos existe un consenso general acerca de la cooperación interpersonal y la comunicación como piezas importantes para optimizar el trabajo en equipo. (Sheng C., Tian Y., & Chen M., 2010) La cooperación interpersonal es requerida en sí misma para resolver problemas y conflictos colaborativamente. (Korsgaard M. , Brodt S. , & Sapienza H., 2005).

Los enfoques basados en redes sociales ofrecen una arquitectura prometedora y natural para representar las relaciones, sin embargo, no puede dejarse de lado el alto costo computacional de los algoritmos de las redes actuales y que el enriquecimiento de la parte relevante de la red requiere de datos apropiados sobre los miembros individuales.

2.5.2. Tipo de tarea

Generalmente, diferentes tareas requieren diferentes tipos de equipos. Hollenbeck, Beersma y Schouten (2012). proponen un marco de trabajo conceptual para diferenciar equipos, el mismo que descansa sobre tres dimensiones: diferenciación de habilidades, diferenciación de la autoridad y estabilidad temporal. La estructura del equipo determina quien cumple determinadas tareas y esto se asocia a la dimensión de la diferenciación de habilidades. La diferenciación de la autoridad hace referencia

a conocer quién tiene autoridad para tomar decisiones cuando existen desacuerdos. Finalmente, el tiempo de duración de la estructura del equipo: corto, mediano o largo plazo, está asociado a la estabilidad temporal. (Hollenbeck J., Beersma B., & Schouten M., 2012)

Las dimensiones señaladas previamente dan lugar a varias categorías de equipos que constan en la Figura 1 y motivan a pensar en estrategias distintas para su formación.

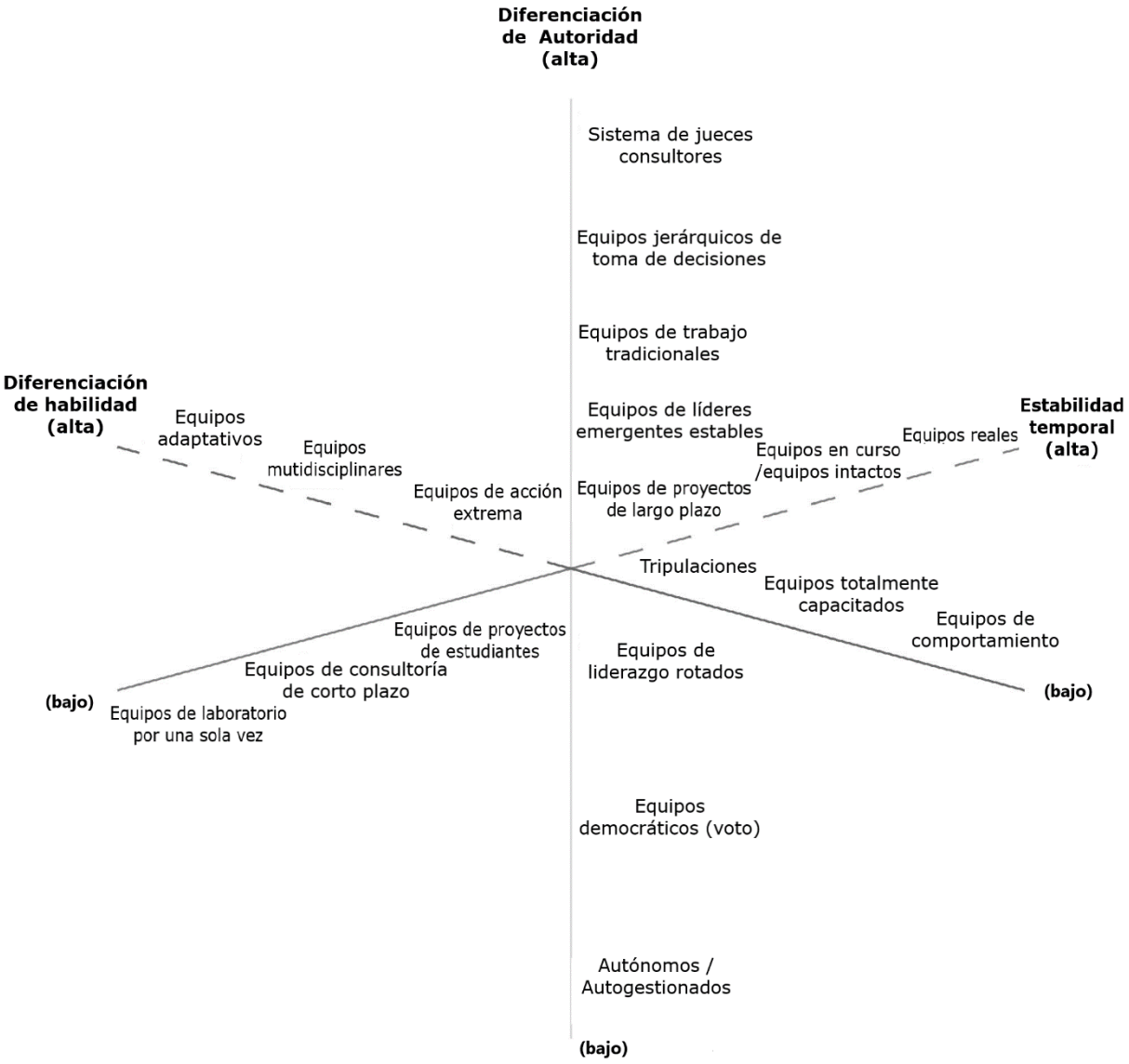

Figura 1 Dimensiones para describir equipos, Hollenbeck et al. (2012)

Se explican las dimensiones tomando un par de ejemplos del gráfico anterior: (1) “Sistema de jueces asesores” (Judge-adviser systems) es un tipo de equipo que tiene un nivel alto de autoridad en sus miembros para tomar decisiones; posee estabilidad

temporal alta y cuenta con una diferenciación de habilidades elevada. Del otro lado de las dimensiones existe un equipo “Democrático” (Democratic teams) donde la diferenciación de autoridad es baja (debe plantearse un conceso entre sus miembros); la diferenciación de habilidades es baja pues no es relevante para equipos que votan las decisiones y la estabilidad temporal también es baja, pues estos equipos se forman para acometer un fin específico considerando una votación.

La investigación sugiere que a medida que las tareas se hacen más complejas y los ambientes se tornan más inciertos, la demanda de equipos más maduros incrementa. La madurez de un equipo está asociada al nivel de desarrollo del equipo, lo cual puede ser identificado por: alta interrelación, alta identificación con el equipo, elevados niveles de coordinación y orientación hacia el logro de un objetivo. La descripción de la madurez de un equipo hace pensar que se requieren equipos más cohesivos ante tareas más complejas en un ambiente incierto. (Navarro J., Quijano S., Berger R., Meneses R., 2011). Es importante señalar que se requiere tiempo para que un equipo desarrolle madurez.

El impacto de los tipos de tareas puede ser apreciado en las decisiones de formación de equipo en distintos contextos: Academia: grupos de investigación y de estudiantes por materia; Ejército: grupos de estrategia, negociación y respuesta a crisis; Hospital: grupos de atención de emergencias, cuidados intensivos y atención ambulatoria. Los autores Baltos y Mitsopoulou (2007) hacen notar que el grado en que el potencial de colaboración y las relaciones personales importan para las decisiones de formación del equipo, cambia de acuerdo con el contexto externo de este; además, las relaciones interpersonales disminuyen como factor cuando el equipo enfrenta una situación de crisis. (Baltos G., & Mitsopoulou Z., 2007) Un ejemplo del planteamiento de Baltos et al. (2007) se presenta cuando una crisis sanitaria en un país requiere que distintos ministros trabajen juntos para dar solución al problema. El aspecto de las relaciones interpersonales se deja de lado a la vez que se demanda un alto potencial de colaboración para paliar la situación de inestabilidad.

2.6. Clasificación del Problema de Formación de Equipos

En esta sección se presentan algunas clasificaciones encontradas para el problema de la formación de equipos.

2.6.1. Taxonomía basada en tarea y comunidad

Los autores Juárez, Santos y Brizuela (2021) sugieren una clasificación que considera los modelos basados en tareas y en comunidad. Cuando el modelo se basa en la tarea, el objetivo es hacer un empate -que implica un score- entre el conjunto de candidatos y las posiciones del equipo; la meta en este caso es maximizar el score de las vinculaciones. Cuando el modelo se basa en la comunidad, se suele llamar Problema de Formación de equipos en Redes Sociales (Team Formation Problem Social Network TFP-SN). Bajo esta orientación, el problema consiste en encontrar un conjunto de candidatos que manifiesten interacciones sociales, además de poseer las habilidades requeridas para cumplir la tarea propuesta.

La Figura 2 muestra los tipos de equipos que pueden aparecer al clasificarlos basados en tareas o en comunidad. Si se trata de tareas: pueden formarse: a) un solo equipo, b) varios equipos (los participantes pueden ser asignado a uno o varios equipos) o c) un equipo afín (kindred team).

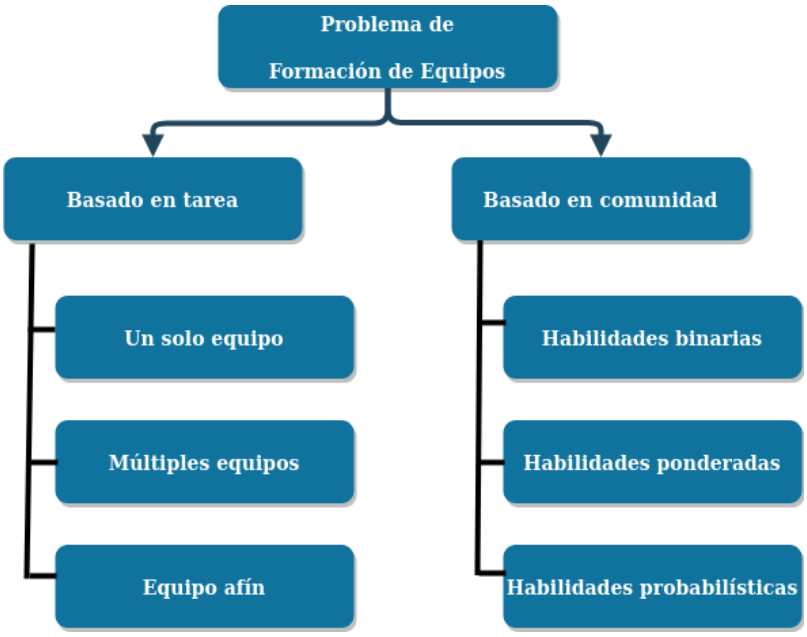

Figura 2 Taxonomía de Juárez et al. (2021)

Si se considera la clasificación por comunidad se tienen: a) Equipos con habilidades binarias (posee o no la habilidad); 2) Habilidades ponderadas; 3) Habilidades probabilísticas (el nivel de la habilidad está representado con una variable probabilística y una distribución). En las dos ramas de la clasificación, además de las habilidades técnicas se reconoce el efecto social de la conducta o preferencias de los individuos para trabajar en equipo.

Dentro del estudio de estos autores, también se proponen como componentes del problema de formación de equipos los siguientes elementos: habilidades técnicas (habilidades duras, hard skills), habilidades de trabajo en equipo y rasgos personales (estos dos últimos son considerados como soft skills o habilidades blandas). (Juárez, J., Santos, C., & Brizuela, C., 2021)

2.6.2. Taxonomía basada en la participación endógena o exógena

Los autores Gómez-Zará, Dechurch y Contractor (2020) señalan que el usuario debe percibir que, durante la formación de equipos, él tiene el control sobre los resultados, es decir sobre los equipos formados. Por este motivo plantean una clasificación donde se consideran dos dimensiones:

1) **Intervención del usuario (user's agency).** Representa el grado al cual el sistema permite que sus usuarios intervengan durante el ensamblaje de equipos.

2) **Participación el usuario (user's participation)**. Representa cuántos usuarios permite el sistema participar en el proceso de ensamblaje de equipos.

La intersección de estas dos dimensiones da lugar a cuatro tipos de equipos, según se puede apreciar en la Figura 3:

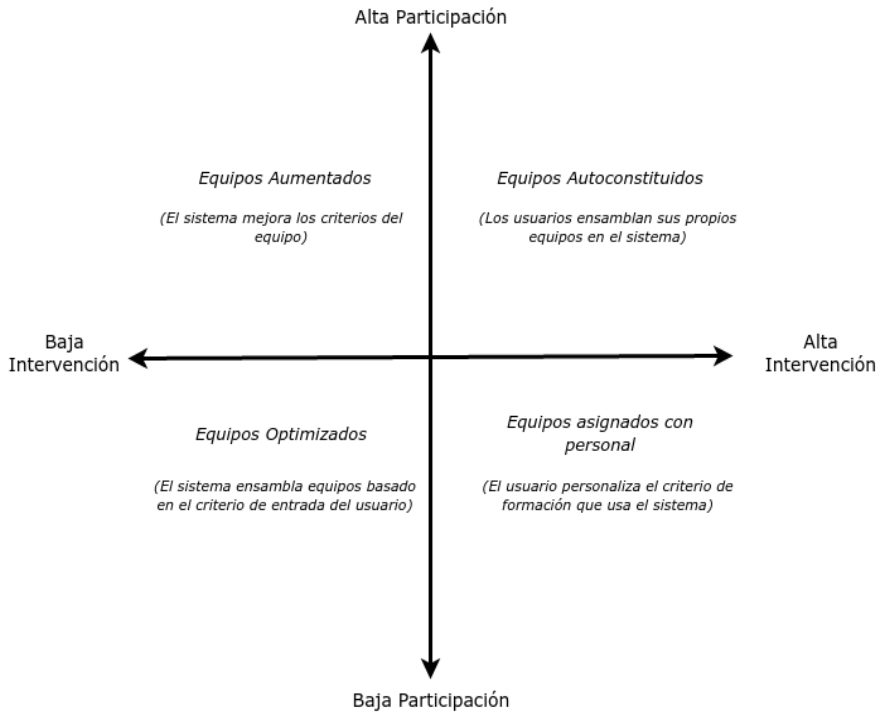

Figura 3 Taxonomía de Gómez-Zará, Dechurch y Contractor (2020)

A continuación, se describen los tipos de equipos formados según esta taxonomía:

- **Equipos autoconstituidos (Self-assembled teams)**. El sistema permite que los usuarios organicen su propio equipo.
- **Equipos asignados con personal (Staffed teams)**. El usuario personaliza el criterio usado por el sistema para simular y formar equipos.
- **Equipos optimizados (Optimized teams)**. El sistema crea equipos dado un criterio particular para la formación.
- **Equipos aumentados (Augmented teams)**. El sistema enriquece las acciones del usuario sugiriendo posibles compañeros de equipo.

Asociando esta taxonomía al presente trabajo, se puede señalar que “Equipos optimizados” es el tipo de equipo que se propone formar con las estrategias de este estudio. Este tipo de equipos se hallan en el cuadrante inferior izquierdo de la taxonomía propuesta y se caracteriza por tener una baja intervención del usuario y una baja participación del usuario. La formación del equipo está limitada a la aportación del usuario para proveer el criterio de formación de equipos. Este tipo de equipos son ensamblados por los algoritmos programados dentro del sistema.

La formación de equipos optimizados tiene ventajas y desventajas, las cuales se enlistan a continuación:

Ventajas

- El equipo se ensambla de forma rápida, objetiva y reproducible.
- Este enfoque es útil para producir equipos de forma masiva.
- El sistema encontrará combinaciones teniendo en cuenta la información de usuarios y tareas, que son manejables computacionalmente.
- El criterio de ensamblaje puede ser configurado previamente dentro del sistema.

Desventajas:

- Las soluciones son fijas y no existen otras alternativas, sino las propuestas.
- Los miembros del equipo están excluidos del proceso de formación.
- Falta de transparencia para los usuarios.
- Los usuarios no pueden proveer retroalimentación sobre el equipo.

Los otros tipos de equipos que surgen de la taxonomía de Gómez-Zará et al. (2020) también cuentan con ventajas y desventajas, sin embargo, se hizo hincapié en los equipos optimizados, por ser estos lo que serán formados con la propuesta a desarrollar. (Gómez-Zará D., Dechurch L., Contractor N., 2020)

2.7. Aspectos técnicos para formación de equipos

2.7.1. El problema de la formación de equipos

Los investigadores en el campo computacional formalizan el problema de crear un equipo (Team Formation Problem TFP) como una tarea en la cual se tiene un conjunto de candidatos, un conjunto de restricciones (por ejemplo: tamaño del equipo) y una función objetivo (por ejemplo: diversidad de miembros). La tarea del algoritmo es seleccionar un subconjunto del grupo de candidatos que satisfacen las restricciones y optimizan la función objetivo. El proceso de selección es engañosamente simple y su

complejidad da cabida a una amplia variedad de enfoques teóricos y algoritmos para su solución.

El TFP puede ser definido informalmente como: encontrar los miembros de un conjunto de candidatos hábiles que formarán un equipo de máxima efectividad para llevar a cabo una tarea.

El TFP ha generado mucho interés en varios campos de estudio. El TFP es considerado un problema NP-hard, cuya complejidad crece según el tamaño del problema.

El concepto de NP-Hardness proviene del estudio más amplio de la complejidad computacional y los algoritmos para la resolución de problemas. El término fue acuñado por Stephen Cook en 1971. El objetivo fue identificar y categorizar problemas que exhibían complejidad en términos de su resolución. Hasta la actualidad, el concepto de NP-Hardness ha sido esencial para entender las limitaciones y posibilidades de las soluciones algorítmicas en las ciencias computacionales.[5]

La dificultad en resolver problemas de tipo NP-hard viene del hecho que requieren tiempo superpolinomial para ser solucionados, y no se conoce de algoritmos que puedan resolverlos en tiempo polinomial. Esto es crucial porque los algoritmos de tiempo polinomial son considerados eficientes, mientras que los algoritmos de tiempo superpolinomial y exponencial son considerados ineficientes debido a su alto costo computacional para grandes entradas.

El TFP es un problema de tiempo Polinomial No determinístico y un problema de alta dimensionalidad con varios óptimos locales, que puede ser resuelto usando algoritmos de aproximación eficientes. (Nageh N., Elshamy A., Hassan A., Sami M., Salam M., 2022). Otros métodos que pueden ser empleados para la solución del TFP son: técnicas heurísticas, algoritmos de tiempo pseudopolinomial, algoritmos aleatorios, algoritmos exactos y algoritmos parametrizados. La selección del método depende del problema específico y los recursos disponibles (conjuntos de datos, software). En muchas aplicaciones del mundo real, es frecuentemente más práctico obtener una solución “suficientemente buena” que una “solución óptima absoluta”,

[5] https://www.larksuite.com/en_us/topics/ai-glossary/np-hard-definition-of-np-hardness

especialmente cuando se trata de problemas grandes y complejos. (Tkatek S., Bahti O., Lmzouari Y., & Abouchabaka J., 2020)

En términos sencillos se puede indicar que un problema NP-hard es aquel en que es improbable, usando cualquier método, producir soluciones óptimas en una cantidad razonable de tiempo.

2.7.2. Técnicas para resolver la formación de equipos

A continuación, se comentan algunas de las técnicas señaladas en los artículos revisados, que los autores emplearon para implementar la solución al TFM. Se realiza particular énfasis en los estudios en los que se realizó análisis de redes sociales.

2.7.3. Soluciones heurísticas

El TFP puede ser planteado como un programa de objetivos lineales enteros mixtos (Liang F., Lawrence S., 2007). Las estrategias heurísticas definen el problema mediante reglas generales que intentan trasladar el problema a un escenario más sencillo.

Las soluciones heurísticas ofrecen rápidamente buenas soluciones al problema y una solución óptima cuando es posible hallarla. Con este tipo de enfoque, las restricciones en la formación del equipo pueden ser limitadas. Matemáticamente, el problema puede ser visto más como un problema de satisfacción de restricciones que un problema de optimización.

2.7.4. Soluciones basadas en redes sociales

Es importante mencionar que los primeros autores en desarrollar el TFP con redes sociales fueron Lappas et al. (2009) definiendo que: dada una tarea T, un grupo de individuos X con diferentes habilidades, y una red social G que captura la compatibilidad entre esos individuos, el problema consiste en encontrar X', un subconjunto de X que lleve a cabo la tarea. Se requiere que los miembros de X' además de poseer las habilidades para cumplir la tarea, puedan trabajar efectivamente juntos como un equipo. La efectividad se mide usando el costo de

comunicación incurrido por el subgrafo en G que solo atañe a X'. (Lappas T., Liu K., Terzi E., 2009)

El artículo de los autores Berktas y Yaman (2019) se interesa por la formación de equipos ofrecida mediante un servicio: Team as a Service[6] – TAAS-. Las compañías que usan este modelo forman un equipo considerando las necesidades de un proyecto específico y proveen administración gerencial del mismo. Los autores estudiaron el TFP considerando la calidad y el costo de la comunicación; para ello, emplearon la proximidad de los miembros potenciales en una red social. Dado un conjunto de habilidades solicitadas, el TFP apunta a construir un equipo que pueda comunicarse y colaborar efectivamente. El estudio empleó dos medidas para la efectividad de la comunicación: la suma de distancias y el diámetro para conjuntos de datos medianos y pequeños; mientras que, para conjuntos de datos grandes, el problema se formuló como un problema de cobertura de conjunto cuadrático con restricciones. El trabajo de este artículo se apoyó en la investigación de los autores Hoegl y Gemuenden. (Berktas N., Yaman H., 2019)

Empleando también redes sociales, en su estudio, Nageh, Elshamy, Hassan, Sami y Salam (2022) postularon el TFP como formar el mejor equipo de expertos en una red social para completar una tarea con el menor costo. Para resolver el problema propusieron dos algoritmos swarm-based; este enfoque computacional está biológicamente inspirado y busca una solución óptima a un problema iniciando con un conjunto de soluciones aleatorias (llamadas partículas) que se mueven en el espacio de solución de acuerdo con reglas simples (una analogía de la naturaleza pueden ser las abejas buscando el polen). Los autores también hicieron un resumen de los algoritmos más populares para formación de equipos: técnicas heurísticas, algoritmos de aproximación, variable neighborhood local search (VNS), Brain Drain Optimization (BRADO, meta-heuristic swarm-based algorith), Improved African Buffalo algorithm (IABO, un nuevo concepto de swap sequence mejorado), Improved Particle Swarm Optimization with New Swap Operator (IPSONSO) y Search Space Reduction-Team Formation (SSR-TF). En la mayoría de los estudios que analizaron se buscó minimizar el costo de comunicación; en otros casos, se apuntó a minimizar el costo de

[6] https://www.teamasaservice.com

coordinación de una tarea o el costo del personal del equipo. (Nageh N., Elshamy A., Hassan A., Sami M., Salam M., 2022)

El estudio de Addanki y Durga (2022). emplea el grado de distribución de las redes sociales para la formación de equipos. La idea principal del algoritmo propuesto es muy similar al enfoque que se usa en los retos del mundo real: un líder escoge como miembros de su equipo a conocidos cercanos en su red que poseen las habilidades necesarias para la tarea propuesta. El algoritmo selecciona nodos con alto grado de la cola pesada de la distribución para fungir como líderes. A continuación, los líderes forman equipos de sus propias vecindades y finalmente el equipo con el mínimo costo de comunicación es escogido como el mejor. Este enfoque muestra que los expertos con alto grado y sus vecinos cubren un gran número de las habilidades requeridas para la tarea, reduciendo los costos computacionales y en consecuencia proveyendo un algoritmo escalable. (Addanki B., Durga B., 2022)

El estudio de Schall (2016)., en el contexto de ecosistemas de software a gran escala, tuvo como objetivo buscar un grupo de expertos dado un conjunto de habilidades, y para ello, empleó un algoritmo genético para evaluar la experiencia, al cual se sumó un componente de redes sociales para estimar la conectividad del equipo. El algoritmo genético buscó optimizar objetivos múltiples: a) Maximizar la calificación promedio de experiencia; b) Minimizar el costo promedio; y c) Minimizar la distancia promedio. El equipo con el mejor balance entre los objetivos obtendrá el puntaje más alto y será recomendado como el mejor. Además, introdujo un aspecto interesante, si el equipo tenía conectividad baja, el algoritmo sugiere un líder (persona con más alto grado en el grupo) que idealmente está conectado a todos los miembros del equipo, para cumplir las funciones de mediador en la comunicación y fortalecer la cohesión del equipo.

Además de mostrar una convergencia satisfactoria en el desempeño del algoritmo genético, una evaluación paralela reveló que a medida que más habilidades son requeridas en los individuos, incrementa la posibilidad de que los miembros del equipo estén desconectados del resto. En resumen, Schall (2016) señaló que el éxito de un equipo no depende solo de la experiencia de las personas involucradas sino de cuán efectivamente colaboran, se comunican y trabajan juntas como equipo. (Schall, 2016)

En una idea muy resumida, el problema de formación de equipos en las redes sociales explota los grafos para representar una posible colaboración entre los participantes. La colaboración puede expresarse a través de diversos criterios (que pueden contener una o muchas variables) y se requiere que su valor sea maximizado o minimizado. La mayoría de los artículos revisados consideran el costo de comunicación en sus algoritmos para la formación de equipos. Otros factores considerados son el costo personal, el balance de la carga de trabajo, la presencia de expertos únicos y la confiabilidad del equipo.

En varios de los artículos revisados se hace referencia a la cohesión como un factor importante para la formación de un equipo. Sin embargo, no se encontraron estudios donde la cohesión fuera medida a través de información provista por las redes sociales. Además, es importante señalar que todos los estudios efectuados con redes sociales parten de una premisa: que los miembros disponibles cuentan con las habilidades requeridas.

Capítulo 3

3. Desarrollo de la propuesta

Esta sección tiene como objetivo señalar los conceptos seleccionados para formular la propuesta de formación de equipos, cuya implementación será descrita en la sección 3.2. Se inicia con el diseño conceptual de la solución y luego se explica la implementación de esta.

3.1. Diseño Conceptual

El análisis de los artículos acerca de la formación de equipos permite apreciar que existen dos elementos que destacan por su aparición permanente: la comunicación y la cohesión. La comunicación es un concepto que se menciona de forma directa en los estudios, sin embargo, la cohesión puede ser inferida de conceptos que se mencionan como: familiaridad, cooperación interpersonal, integración, interrelación, sinergia o química presente en los equipos.

Sin importar las técnicas de algoritmos a utilizar, el contexto, o el número de equipos que se deben formar, los dos elementos citados son relevantes para la formación y operación de un equipo. Los autores Hoegl et al. (2021) definieron el concepto de calidad de trabajo en equipo (Teamwork Quality - TWQ) empleando seis elementos, dentro de los cuales constan la comunicación y la cohesión. Además, en la revisión de las perspectivas desde las ciencias psicológicas se pudo encontrar que existe un consenso general acerca de la cohesión y comunicación como dos factores clave para el desempeño de un equipo. (Sheng C., Tian Y., & Chen M., 2010)

Por los motivos expuestos, este estudio considera a la comunicación y cohesión como dos variables fundamentales para la propuesta de formación de equipos. Aunque para el presente estudio no se cuenta con información a priori de los estudiantes, por ejemplo, atributos como su nacionalidad, nivel académico, ubicación, edad o género, que pueda enriquecer el conocimiento de las posibles relaciones entre los mismos, se intentará aprovechar la mayor cantidad de información disponible a partir de las conversaciones en los foros, identificando métricas que permitan medir de forma directa o indirecta (indicador proxy) las variables.

3.2. Declaración del problema

"Formar un equipo *con* la *habilidad* para desarrollar una *tarea* encargada, considerando la *maximización* de la *comunicación* y la *cohesión,* variables que se modelan a través de las *interacciones en una red social*."

Es importante desarrollar algunos conceptos inmersos en la declaración del problema:

- **Formar**. Existe un agente exógeno que establece ciertos criterios para la formación del equipo.
- **Habilidad**. Se refiere a la capacidad del estudiante para aprobar una materia.
- **Tarea**. Para el caso específico de este estudio, la tarea es desarrollar un trabajo grupal dentro del contexto de una materia específica.
- **Metas de optimización.** Se pretende maximizar las variables de comunicación y cohesión entre los miembros del equipo.
- **Interacciones en la red social.** Estas interacciones permiten generar métricas que faciliten la evaluación de la comunicación y cohesión.

Para el presente estudio, en la Tabla 4, se detalla el concepto que se asocia a cada variable seleccionada (Gómez S., 2019) (Furht B., 2010):

Comunicación	Cohesión
Concepto Es la medida de cuán cercanamente relacionados se encuentran dos miembros de la red, considerando las interacciones establecidas en la red social, en este caso, el espacio de los foros virtuales.	**Concepto** Es la medida del interés o el agrado que una persona manifiesta al interactuar con otra en los foros virtuales.
Objeto al que se vincula: estudiantes	Objeto al que se vincula: mensajes
Métricas para medir: Grado; Centralidad de intermediación Centralidad de cercanía	Métricas para medir: Probabilidad afectiva; Tiempo de respuesta

Tabla 4 Variables usadas para la formación de equipos.

3.3. Restricciones

Debido a que el problema de formación de equipos es un problema de tipo NP-hard, cuya complejidad crece según el tamaño del problema, se hace necesario definir algunas restricciones para acotar la propuesta de la formación de equipos y a su vez delimitar la operación del algoritmo que genera los equipos. Por lo tanto, se consideran las restricciones descritas en la Tabla 5.

Restricción	Valor
Participación para formar el equipo	Exógena
Intervención del usuario (user's agency)	Baja (solo proporciona el criterio)
Participación del usuario (user's participation)	Nula (los miembros del equipo no participan)
Contexto	Academia: equipos de estudio virtual
Números de equipos	Varios equipos
Participantes únicos	Los miembros pertenecen solo a un equipo
Tamaño del equipo	Equipo de tamaño fijo (4 miembros)
Tipo de tareas	Un solo tipo de tarea
Habilidades de los integrantes	Una sola habilidad que todos cumplen
Conexión entre los integrantes	Conectados en la red social

Tabla 5 Restricciones consideradas en la propuesta de formación de equipos.

El número de miembros en el equipo es fijo en esta propuesta y es igual a 4, dicho valor fue considerado al buscar indicios sobre el número ideal de participantes en un equipo. El valor encontrado en las revisiones oscila entre 4 y 10, siendo el tamaño perfecto 4.6, número propuesto por Hackman y Vidmar en 1970. Los sujetos de estudio fueron alumnos universitarios reunidos en grupos de 2 a 7 miembros. Después de terminar la tarea asignada se les hizo dos preguntas: 1) ¿El grupo fue muy pequeño para obtener mejores resultados? 2) ¿El grupo fue muy grande para obtener mejores resultados? (Hackman J., Vidmar N., 1970) Las respuestas permitieron que se obtenga la Figura 4, donde las líneas de los puntajes estandarizados de satisfacción y el tamaño del grupo cruzan para exhibir que el número ideal de miembros en un grupo es 4.6.

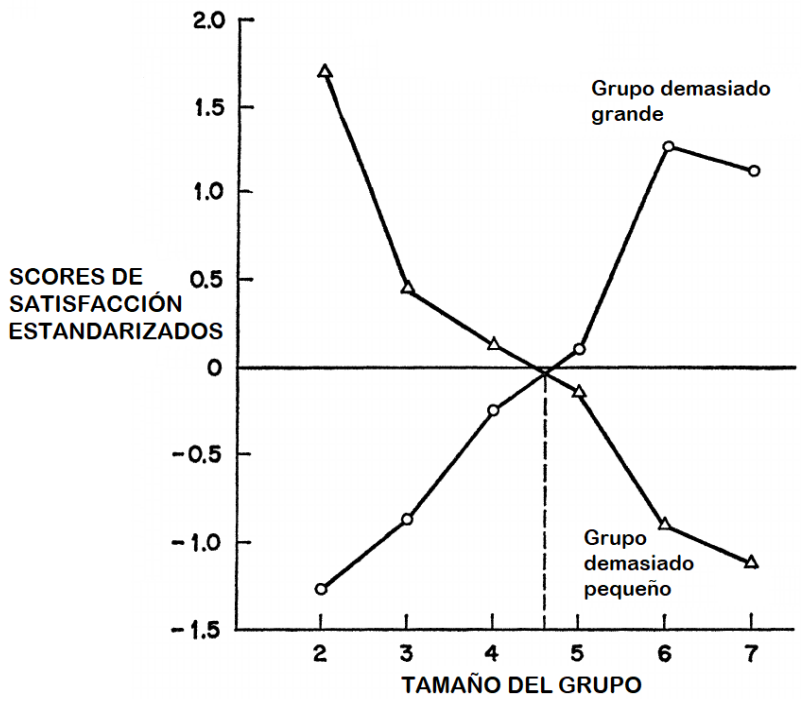

Figura 4 Satisfacción con el tamaño del grupo. Hackman and Vidmar (1970)

Buscando una referencia más cercana al tipo de problema que desarrolla este trabajo, se revisó el tamaño de los equipos de software, encontrándose como un rango ideal de miembros entre 3 y 7. Este rango fue propuesto por Putnam y Myers en el año 2000, luego de analizar 491 proyectos de software de tamaño medio que tenían entre 35.000 y 95.000 líneas de código. Al comparar el esfuerzo de desarrollo de estos proyectos, encontró que los equipos que tenían entre 3 y 7 personas (1.5-3 personas; 3-5 personas; 5-7 personas) les ocupó un cuarto del esfuerzo de los grupos más grandes (9-11 personas; 15-20 personas) construir la misma funcionalidad de software. De forma particular, a los equipos de tamaño 3-5 personas les tomó un tercio menos de esfuerzo que a los equipos de 5-7 personas. (Putnam L., Myers W., 2000). De estas cifras se puede estimar que el equipo más eficiente fue el que tuvo entre 3 y 5 miembros.

Los dos estudios revisados mencionan que la comunicación es uno de los factores que se ve afectado al aumentar los miembros el equipo, ya que eleva la carga de comunicaciones que debe gestionar el equipo. Las cifras encontradas 4.6 y un rango entre 3-5 hacen que este estudio se decante por usar un número intermedio completo: 4. Esta selección de un número pequeño de miembros es congruente con las recomendaciones encontradas para el trabajo colaborativo en la educación virtual a nivel superior, que sugiere trabajar en grupos pequeños.

Para esta propuesta en particular, se asume que se requiere equipos de estudiantes que están cercanos unos a otros. Lo contrario también puede cierto para otros entornos, por ejemplo, si el objetivo es crear un equipo de jueces expertos con opiniones independientes, probablemente se requiera que los mismos se encuentren apartados en la red social. (Anagnostopoulos A., Becchetti L., Castillo C., Gionis A., & Leonardi S., 2012)

3.4. Estrategia

Formar equipos teniendo en cuenta la maximización de dos variables: comunicación y cohesión, dirige el problema a resolver un objetivo bicriterio. Este tipo de problemas, usualmente se atacan, limitando una de las variables y optimizando la restante, es decir, fijar la variable comunicación y optimizar la cohesión, o a la inversa: limitar la cohesión y optimizar la comunicación. En este estudio ambas consideraciones son válidas y pueden resultar naturales para el tutor.

3.5. Herramientas

Se mencionó previamente que los conjuntos de datos disponibles para el análisis de este TFM corresponden a los mensajes intercambiados por los estudiantes en los foros de las materias Aprendizaje Automático y Minería de Datos en los Medios Sociales para los períodos 2019-2020; 2020-2021 y 2021-2022. Estos datasets tienen datos no etiquetados, es decir no poseen categorías o grupos asignados, de modo que el uso de una técnica no supervisada es apropiado para descubrir grupos en los datos. En la Tabla 6 se muestra el tamaño de las observaciones para cada dataset.

Materia y período	AA1920	AA2021	AA2122	MDMS1920	MDMS2021	MDMS2122
# estudiantes	27	55	45	10	28	22
# mensajes	302	192	222	132	240	82

Tabla 6 Datasets originales de trabajo

3.5.1. Técnica No Supervisada: K-means

K-means es un algoritmo no supervisado de tipo geométrico con el cual se busca encontrar un número K (previamente definido) de grupos o clústeres generados considerando las similitudes entre los atributos del dataset. Para medir la similitud se utiliza la clásica distancia euclidiana, que se muestra en las Figuras 5 y 6:

$$d(x,y)^2 = \sum_{j=1}^{m} (x_j - y_j)^2 = \|x - y\|_2^2$$

Figura 5 Cálculo de la distancia Euclidiana

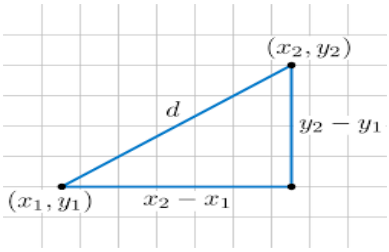

Figura 6 Medición entre dos puntos[7]

El proceso de generación de los clústeres consiste en iniciar inventando K observaciones (centroides) hipotéticas y luego ir ajustándolas hasta que sean las más representativas, es decir serán aquellas que posean los atributos más característicos de K agrupaciones. Las observaciones de un clúster deben ser lo más homogéneas posible entre sí y lo más heterogéneas posible en relación con las de otros clústeres. Si se considera que la similitud entre los centroides y las demás observaciones es proporcional a la distancia que los divide, entonces el objetivo del algoritmo es minimizar las distancias de las observaciones con respecto a sus centroides relacionados.

El proceso de formación de clústeres se menciona a continuación:

1) Después de la asignación inicial aleatoria de K centroides (vectores en plano), de forma iterativa, el conjunto de datos se asocia al centroide más cercano.

2) Tras cada iteración, los centroides se ajustan para ubicarse en el centro de todas las observaciones asociadas a ellos, esto es, en el punto más equidistante con respecto a las observaciones que han encajado en su clúster.

Estos pasos se repiten hasta que se cumple un criterio establecido: el número máximo de iteraciones se alcanza o no existe variación en los centroides. (Ahmed M., Seraj R., Islam S., 2020)

[7] https://es.wikipedia.org/wiki/Distancia_euclidiana

3.5.2. Búsqueda del número ideal de clústeres

Debido a que por adelantado no se conoce el número ideal de clústeres en los que se deben agrupar los datos, porque no existen clases atribuidas a cada observación, es necesario considerar las distancias: cohesión intraclúster (distancias entre las observaciones del mismo clúster) y cohesión interclúster (distancias de las observaciones de un clúster con respecto a las observaciones de los clústeres restantes). Se presentan dos métodos asociados a las distancias:

Coeficiente de Silhouette (Silhouette score) cuyo valor oscila entre -1 y 1 indica lo siguiente:

- Un valor próximo a -1 indica que la mínima distancia media interclúster es muy inferior a la distancia media intraclúster. Esto es un indicativo de que existen muchas observaciones que son más parecidas a las de otros clústeres que a las de su propio clúster.

- Un valor cercano a 1 indica que el algoritmo logró un buen nivel de cohesión interna y separación entre clústeres puesto que la mínima distancia interclúster es muy superior a la distancia media intraclúster. Es decir, existen muy pocas o casi ninguna observación que sea más parecida a una observación de un grupo distinto, que a una observación de su propio grupo.

Para encontrar el número óptimo clústeres es necesario entrenar el modelo un número determinado de veces, calculando el score de Silouhette para cada valor diferente de grupos. (Shahapure K., Nicholas C., 2020)

Error de Inercia. Cuando más variadas son las observaciones del dataset, mayor serán sus distancias en relación con sus centroides asociados. En K-means la inercia es la suma de todas las distancias en las observaciones de un clúster a su centroide. Dado que el objetivo es minimizar la suma de las distancias (cuadrados) de puntos con sus correspondientes centroides, cuanto menor sea esta suma total será mejor porque denotará mayor homogeneidad en las observaciones concernientes a los clústeres creados. (Géron A., 2019)

Al graficar el error de inercia asociado a cada uno de los valores de número de clústeres probados, se obtiene el gráfico del codo (elbow plot), que visualmente ayuda

a identificar el número de clústeres óptimo al buscar el punto en el que la inercia empieza a decrecer a una tasa más lenta.

En este trabajo se utilizará la implementación en Python (3.10) de K-means a través de la librería scikit-learn 1.4.0.

3.5.3. Análisis de redes sociales: Gephi

La herramienta empleada para realizar el análisis de redes sociales es Gephi. Esta es una herramienta consolidada como líder en la visualización y exploración de todos los tipos de grafos y redes. Gephi es una herramienta de software abierto (open-source) y gratuita. Este software es muy versátil para el análisis de redes ofreciendo una potente funcionalidad para la visualización de los elementos de las redes y por defecto incorpora las métricas más usadas en la sociología. Gephi es una aplicación multiplataforma y ofrece un alto desempeño[8].

Una vez resumidas las consideraciones del análisis y las herramientas a utilizar, en la Figura 7 se presenta de forma resumida el enfoque general adoptado para la elaboración de las estrategias para formación de equipos de trabajo.

[8] https://gephi.org/

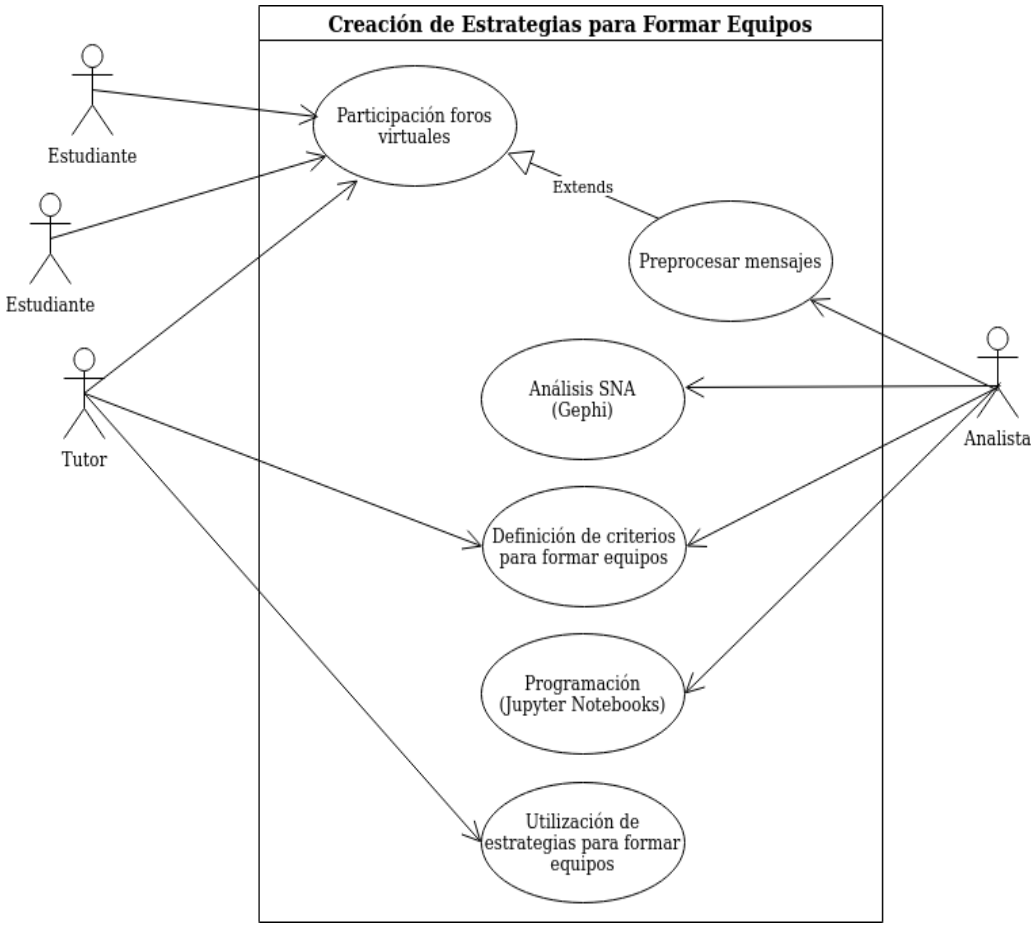

Figura 7 Esquema general del enfoque a emplear

3.6. Implementación

El análisis de datos tiene que ver con la exploración metódica e interpretación de los datos; es la base para la toma de decisiones en el panorama dinámico de datos que existe en la actualidad. La recolección, transformación, análisis y presentación de los datos para elaborar conclusiones y hacer predicciones son las tareas que en resumen se cumplen al realizar análisis de datos.

Para el desarrollo práctico de este TFM a continuación se presenta un diagrama donde de forma resumida se señalan las etapas cumplidas en la construcción de la propuesta para formación de equipos usando análisis de redes sociales y técnicas no supervisadas, esto se aprecia en la Figura 8:

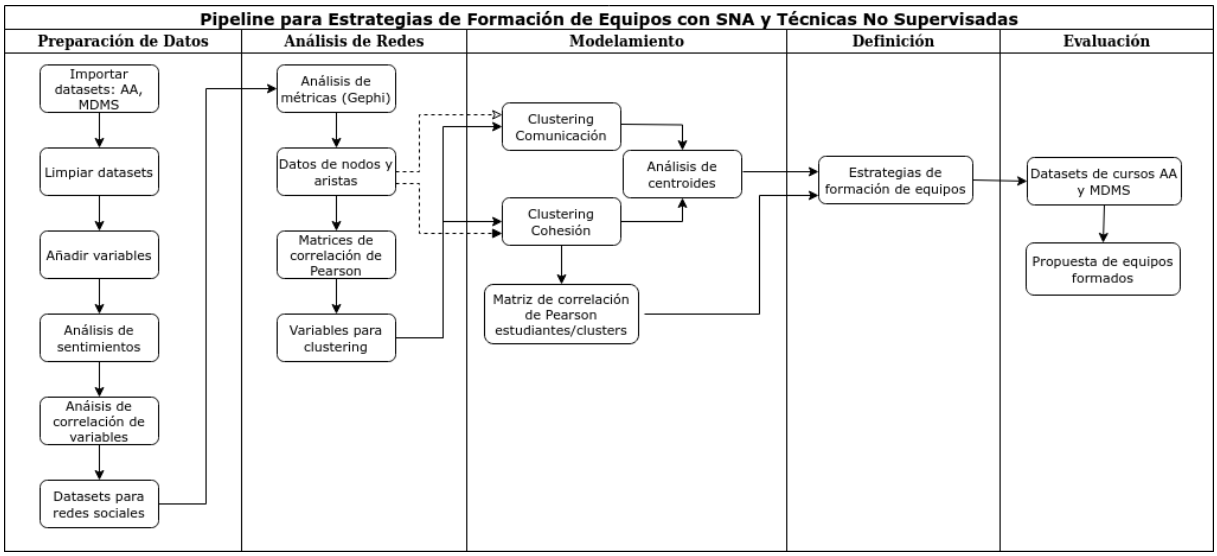

Figura 8 Pipeline para la implementación de las estrategias

Como se puede ver en la figura anterior, las cinco etapas definidas para la construcción de la estrategia de formación de equipos se encuentran vinculadas, y la culminación de cada etapa, genera un insumo para la siguiente. Por ejemplo, la etapa (1) provee los datos para generar el análisis de redes sociales descrito en la etapa (2). La etapa (2) genera archivos de nodos, aristas, además de las variables para clustering que sirven para la etapa (3). La etapa (3) entrega el análisis de centroides y las matrices de correlación de Pearson (estudiantes/clústeres) para definir las estrategias en la etapa (4). La etapa (4) formula las estrategias y las mismas son probadas en la etapa (5).

Para apreciar la correspondencia entre las etapas con las herramientas utilizadas se expone un Diagrama de componentes en la Figura 9:

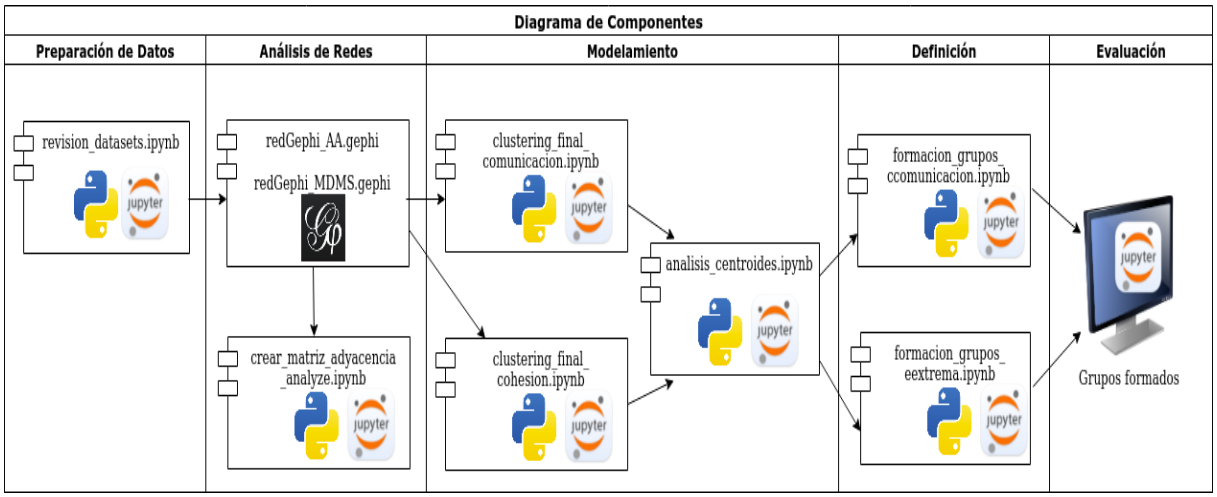

Figura 9 Diagrama de Componentes

A continuación, se pasa a describir en detalle el trabajo realizado en cada una de las etapas.

3.6.1. Etapa 1: Preparación de datos

Entrada	Datasets provistos por el docente
Salida	Archivos para desarrollar análisis de redes sociales.
Archivos	revision_datasets.ipynb

Tabla 7 Entradas y salidas en Etapa 1

Principales actividades realizadas:

El objetivo de esta actividad fue depurar y transformar los datasets para obtener archivos que puedan ser empleados en el análisis de redes sociales. Las acciones llevadas a cabo fueron:

1. Reducción de variables (retiro de columnas de poca utilidad para el análisis). Véase el contenido del archivo original en la Figura 10:

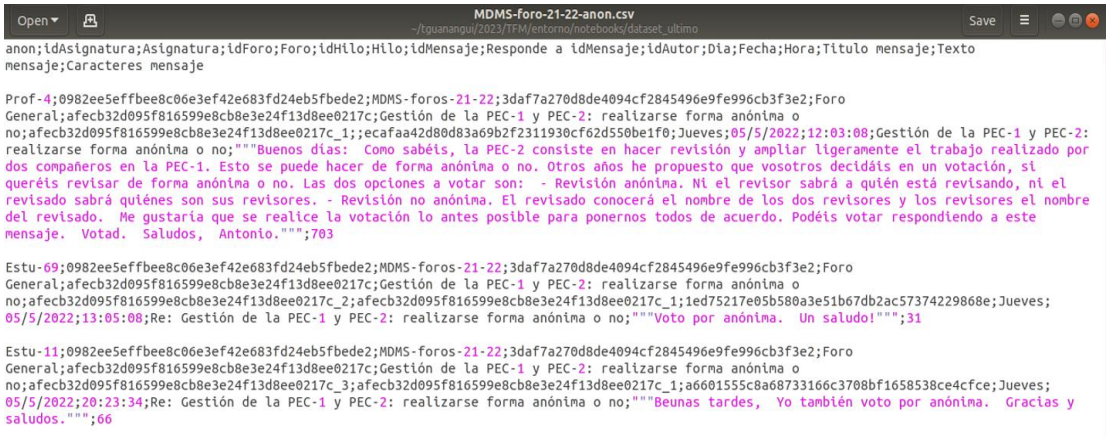

Figura 10 Dataset original para materia MDMS 2122

2. Establecer los estudiantes origen y destino para cada mensaje.

3. Añadir variables para explorar las métricas de medición para comunicación y cohesión: número de hilos creados por el estudiante (iniciativa); número de mensajes enviados por un estudiante; respuesta en horas como el tiempo que demoró el estudiante en contestar a cada mensaje.

4. Realizar un análisis de sentimientos empleando una librería basada en métodos Bayesianos (sentiment_analysis_spanish[9]) para obtener la probabilidad de que la cadena de texto sea positiva. Bajas probabilidades significan que el texto es negativo (números cercanos a cero), altas probabilidades (números cercanos a 1) indican que el texto es positivo. El espacio intermedio corresponde a textos neutros. A esta variable se le denominó probabilidad afectiva.

5. Realizar un análisis de correlación entre las variables disponibles.

6. Generar los datos para el análisis de redes sociales. Véase el contenido del archivo final en la Figura 11:

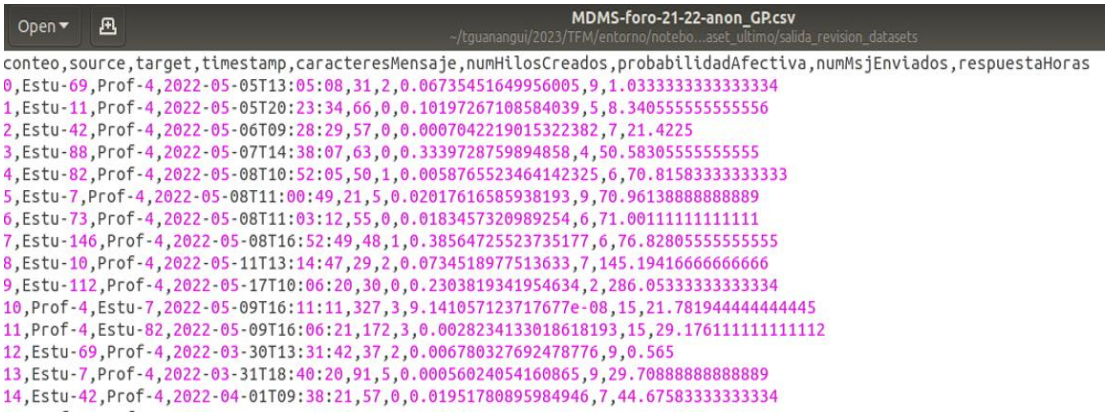

Figura 11 Dataset transformado para análisis de redes sociales

3.6.2. Etapa 2: Análisis de redes sociales

Entrada	Archivos preprocesados de mensajes de estudiantes
Salida	Archivos de nodos y aristas con métricas de redes sociales
Archivos	(Archivos Gephi) redGephi_AA.gephi y redGephi_MDMS.gephi; (Jupyter Notebook) crear_matriz_adyacencia_analyze.ipynb

Tabla 8 Entradas y salidas en Etapa 2

9

Principales actividades realizadas

Con la herramienta Gephi se desarrolló el análisis de redes sociales para obtener las métricas a partir de los nodos (estudiantes) y las aristas (mensajes intercambiados). El análisis visual realizado con Gephi es de gran utilidad puesto que a simple vista aparecen elementos y relaciones importantes en la red. Por ejemplo, en el dataset de Aprendizaje Automático para el período 20-21 lucen con texto de mayor tamaño los nodos que poseen grado más alto dentro de la red, esto se aprecia en la Figura 12:

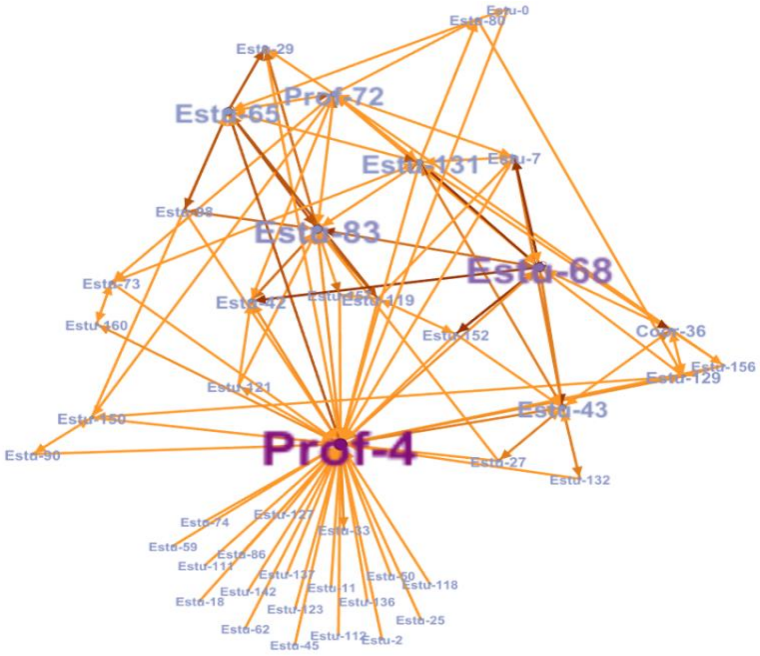

Figura 12 Red social para materia Aprendizaje Automático 20-21

Por defecto se generaron las métricas de uso más generalizado: degree (Grado), closnesscentrality (centralidad de cercanía), betweenesscentrality (centralidad de intermediación), Authority (autoridad), Pageranks. Un ejemplo de estas métricas se aprecia en la Figura 13:

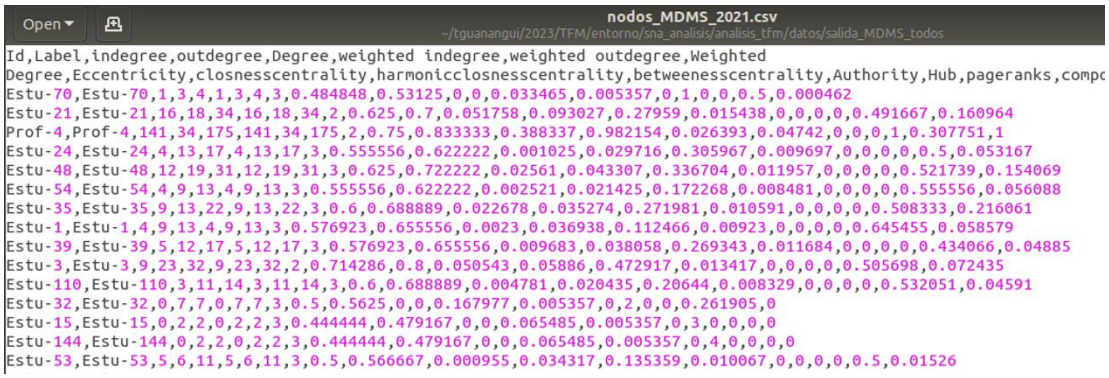

Figura 13 Métricas de redes sociales para dataset MDMS 20-21

Además, con la disponibilidad de los nodos y aristas se creó una matriz de adyacencia para analizar el comportamiento de las variables: CaracteresMensaje, ProbabilidadAfectiva y RespuestaHoras, entre cada par de nodos, es decir entre cada par de estudiantes.

Los gráficos a continuación permiten apreciar que efectivamente existen distintos grados de correlación entre los estudiantes tanto en sentido (positivo/negativo) como en fuerza (valor entre -1 y 1). La coloración blanca indica un valor de correlación positiva (hasta 1), mientras que el color negro muestra un valor de correlación negativa (hasta -1). La Figura 14 muestra la correlación para la variable RespuestaHoras; nótese que existe correlación positiva entre los estudiantes y en distinta fuerza.

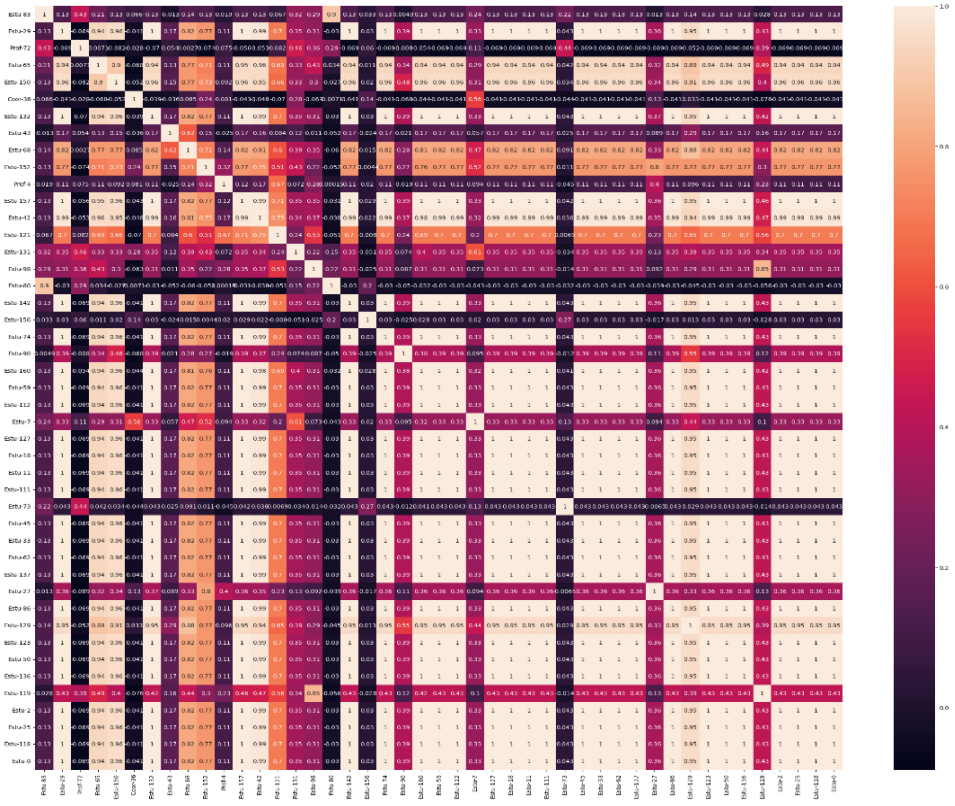

Figura 14 Matriz de correlación de Pearson - variable RespuestaHoras

Para la variable ProbabilidadAfectiva se apreciaron pocas correlaciones positivas (color claro) entre los pares de estudiantes. Esto demuestra que existen estudiantes que dirigen mensajes más positivos a unas personas que a otras, lo cual podría traducirse en que a un estudiante posiblemente le gustaría trabajar más con ciertos compañeros y menos con otros. La Figura 15 muestra la correlación de Pearson para la variable ProbabilidadAfectiva:

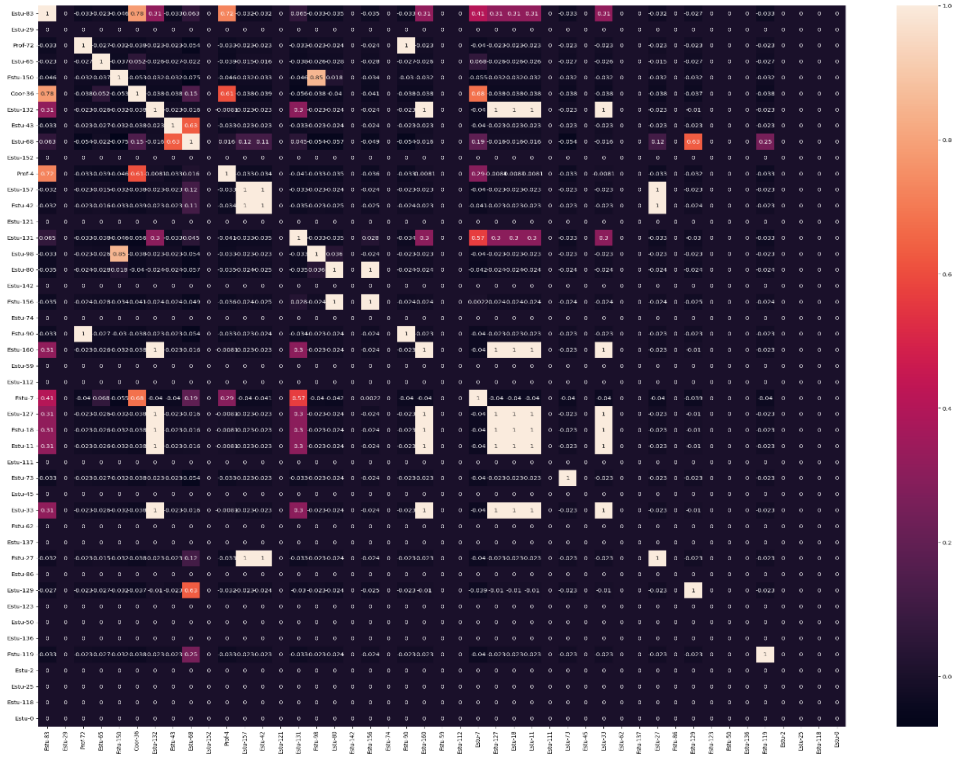

Figura 15 Matriz de correlación de Pearson - variable ProbabilidadAfectiva

Para la variable CaracteresMensaje se generó la matriz de correlación de Pearson, que consta en la Figura 16, y permite distinguir que algunos estudiantes intercambiaron mensajes largos y cortos con distintas personas.

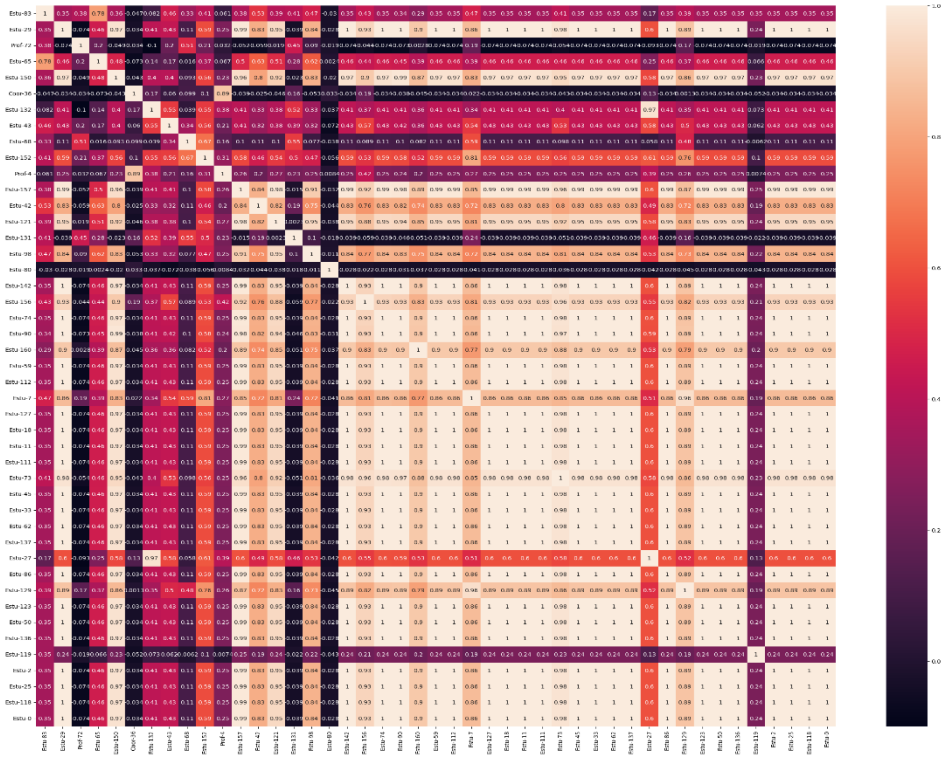

Figura 16 Matriz de correlación de Pearson - variable CaracteresMensaje

Selección de métricas para las variables

Los estudios analizados en el Capítulo 2, “Estado del arte”, permiten identificar a la comunicación y cohesión como dos de los atributos que más destacan en la formación de equipos de trabajo. Al analizar los artículos que incluyeron redes sociales, aparecen diversas métricas seleccionadas para el estudio (diámetro, camino más corto, autoridad, grado), mismas que están asociadas al contexto del problema, los objetivos de optimización y también la disponibilidad de los datos. Esto motiva a considerar que cada estudio debe escoger las métricas de acuerdo con sus características particulares.

El análisis de redes sociales se concentra en el descubrimiento de patrones de interacción entre los actores sociales en las redes sociales, y permite hacer uso de medidas de centralidad para descubrir actores con una posición más central, es decir que tienen un acceso más fácil y rápido a los demás actores de la red (valioso para acceder a recursos como información) y una mayor capacidad para ejercer control del flujo entre ellos. En otras palabras, se busca hallar a los individuos más influyentes, prestigiosos o centrales.

Existen varias medidas de centralidad: 1) Grado (degree), 2) Intermediación (betweeness), 3) Cercanía (closeness), 4) Excentricidad (eccentricity), 5) Centralidad de vector propio, 6) Coeficiente de clustering. Las tres primeras fueron propuestas por Freeman (Freeman L., 1979) para redes no ponderadas y Brin y Page (creadores de Google) las extendieron para redes ponderadas. La cuarta medida fue propuesta por Hage y Harary. (Hage P., Harary F., 1995) La quinta medida fue propuesta por Bonacich en 1987 y la sexta medida fue propuesta por Sokal y Michener[10].

Los datos hallados en el análisis de redes sociales y las matrices de adyacencia elaboradas, permitieron definir métricas a nivel de actores, es decir, medidas que permiten identificar actores clave de la red que tienen un poder social. (Gómez S., 2019). A continuación, en la Tabla 9, se describen las métricas usadas para cada variable con la justificación respectiva:

Métricas para comunicación

Métrica	Definición	Aporte
Grado (Degree)	Número de conexiones para un nodo.	Tienen influencia o protagonismo en la red; pueden acceder a más oportunidades y alternativas. Pueden intercambiar información rápidamente con muchos otros.
Cercanía (Closeness)	Longitud de los caminos hacia otros nodos.	Realiza un intercambio directo de mensajes con otros estudiantes. Estos estudiantes son capaces de difundir información de forma eficiente en una red.
Intermediación (Betweeness)	Mide el grado al cual un nodo descansa en las rutas hacia otros nodos.	Los estudiantes con intermediación alta ocupan una posición particular en la red, ya

[10] https://sci2s.ugr.es/sites/default/files/files/Teaching/GraduatesCourses/RedesSistemasCompejos/Tema01-IntroduccionalasRedes-13-14.pdf

		que les permite trabajar como interfases entre subgrupos de estudiantes. Además, poseen influencia considerable dentro de una red debido a su control sobre la información que pasan entre unos y otros.

Tabla 9 Métricas para medir la variable Comunicación

Métricas para Cohesión

Los estudios revisados señalan que la forma de medir la cohesión es mediante pruebas psicométricas y estudios de ambiente de trabajo. La cohesión generalmente puede ser medida a posteriori, ya que se cuantifica una vez que las personas han trabajado juntas. La evaluación puede ser sencilla y consiste en preguntar al individuo con quién le gustaría trabajar o con quién no le gustaría trabajar. Considerando los datasets de redes sociales disponibles, no existe directamente una métrica que ayude a evaluar esta variable, sin embargo, se pueden emplear variables proxy que permitan predecir con qué personas le gustaría trabajar o interactuar a un estudiante. Para contestar la pregunta anterior, este estudio proponer dos elementos que pueden ser obtenidos a través de las características de los mensajes intercambiados por los estudiantes: La Tabla 10 contiene los elementos propuestos para medir la Cohesión:

Elemento	Definición	Aporte
Probabilidad afectiva	La probabilidad de que una comunicación escrita sea expresada en términos positivos. (1 mayor probabilidad, 0 menor probabilidad)	Indica el grado de aceptación o rechazo que un estudiante muestra hacia otro.
Respuesta en horas	El tiempo en horas que una persona demoró en contestar un mensaje.	Puede revelar el anhelo de un estudiante por iniciar un intercambio de ideas con otro.

Tabla 10 Elementos para medir la variable Comunicación

Definidas las variables con sus respectivas formas de evaluación se procede a aplicar el clustering con los datasets generados.

3.6.3. Etapa 3: Modelamiento

Entrada	Archivos de nodos y aristas
Salida	Resultados del clustering usando las variables comunicación y cohesión. Matriz de correlación de Pearson para estudiantes
Archivos	clustering_final_comunicacion.ipynb; clustering_final_cohesion.ipynb; analisis_centroides_v4.ipynb

Tabla 11 Entradas y salidas en Etapa 3

Algunos aspectos comunes para los dos ejercicios de clustering se describen a continuación. Se empleó la técnica no supervisada de clustering K-means con los siguientes parámetros[11]:

- **K:** número de clústeres, de 1 hasta 12 grupos. Definido por defecto para este ejercicio.

```
kmeansArgs = {"init": "random", "n_init": 10, "max_iter": 300, "random_state":42}
```

Donde:

- **init:** método de inicialización. Random: escoge varias observaciones (filas) de datos de forma aleatoria para los centroides iniciales.
- **n-init:** el número de veces que el algoritmo K-means es ejecutado con diferentes semillas de centroides.
- **max_iter:** número máximo de iteraciones del algoritmo K-means para una sola ejecución.

[11] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

- **random_state:** determina la generación del número aleatorio para la inicialización del centroide. Se fija un número para que los resultados sean reproducibles.

Para determinar el número óptimo de clústeres a formar se empleó el Método del codo y el Coeficiente de Silhouette. Cabe mencionar que no se realizó una optimización de los hiperparámetros de K-means, por no ser motivo de este estudio.

Clustering usando la variable Comunicación

El objetivo de este clustering es obtener grupos de estudiantes que sean similares de acuerdo con los atributos definidos para la comunicación. Antes de desarrollar el clustering se realizó un escalamiento del dataset con las tres variables: grado, cercanía e intermediación.

Previamente se realizó un clustering con todo el grupo (coordinador/profesor y estudiantes) de la materia para cada período de estudios, con el objetivo de analizar los atributos identificados en el análisis de redes sociales. Uno de los primeros hallazgos fue confirmar que el coordinador o profesor modela generalmente el comportamiento de comunicador principal en el grupo de estudios. El coordinador registró el grado más alto dentro de la red y viene a ser el arquetipo por emular en cuanto al comportamiento solicitado a los estudiantes: participar y aportar en la red. El coordinador también mostró valores de cercanía e intermediación mayores a los de los estudiantes. En todos los ejercicios de clustering por materia y período, el coordinador apareció dentro de un clúster con un elemento único.

A continuación, se retiró al coordinador del grupo y nuevamente se aplicó el clustering con las métricas para la variable Comunicación. Los resultados se muestran en la Tabla 12:

Materia y período	AA1920	AA2021	AA2122	MDMS1920	MDMS2021	MDMS2122
# estudiantes	26	54	44	9	27	21
# clústeres formados	4	3	3	3	3	4
# miembros por clúster	2,1,13,10	5,9,40	6,15,23	3,3,3	3,16,8	9,8,3,1
Inercia	14.91	37.78	20.88	5.36	12.73	4.51

Tabla 12 Resultados de clustering con variable Comunicación

Se aprecia que los clústeres formados oscilan entre 3 y 4 grupos. Cabe mencionar que en un ejercicio previo fueron consideradas cuatro métricas (grado, intermediación, centralidad y pageranks) y posteriormente al emplear las tres métricas definidas (se realizaron pruebas retirando una métrica una a la vez) disminuyó la inercia para todos los datasets excepto para AA2021, de modo que utilizar solo tres métricas resultó en un beneficio. La métrica Pageranks asigna un puntaje de importancia a cada nodo, y la suposición que hace es que los nodos importantes son aquellos que tienen muchos enlaces de entrada de otros nodos importantes. (Gómez S., 2019) Sin embargo, este concepto no era del todo consistente con la medición de la variable comunicación y, además, al retirar Pageranks mejoró la inercia para algunos datasets.

Pese a que los clústeres formados ya no contienen al Coordinador o Profesor se formaron clústeres con 1 elemento, sin embargo, los únicos elementos de esos clústeres corresponden a estudiantes con situaciones particulares: un estudiante que solo recibió mensajes, pero nunca contestó alguno; y un estudiante que escribió pocos mensajes, pero recibió numerosas contestaciones sobre un mismo mensaje.

Los valores de métricas de redes sociales de los clústeres formados estuvieron por debajo de los valores del coordinador, denotando que efectivamente el profesor es la figura prototipo de la comunicación que se espera en el foro.

Durante esta etapa se generó un archivo único para almacenar todos los estudiantes en todos los foros de todas las materias y también se guardaron los centroides para análisis posterior.

Clustering usando la variable Cohesión

El objetivo de este clustering es obtener grupos de mensajes intercambiados por un estudiante origen y otro destino, que sean similares de acuerdo con los atributos definidos para la cohesión. Los estudiantes pueden repetirse en varios grupos de mensajes, por lo cual en un paso más adelante, se identificará en qué grupos un estudiante fue asignado. Esto indica en qué grupos un estudiante podría participar por la cohesión que demuestra con distintos individuos.

Previo a la ejecución de este clustering se realizó un escalamiento de las variables ProbabilidadAfectiva y RespuestaHoras. El escalamiento tuvo como objetivo permitir

una correcta interpretación de las variables dentro del algoritmo de clustering. La ProbabilidadAfectiva contaba con valores muy pequeños y se decidió mapear el valor más alto a 1 y el valor más bajo a 0. De modo, que todos los valores oscilen en el rango de 0 a 1 y mantengan el significado de la probabilidad original. En cuanto a la variable RespuestaHoras se decidió aplicar una equivalencia inversa al tiempo: mientras más horas tardó en contestar, el nuevo valor se acercaba a 0, y si tardó menos horas en responder, el nuevo valor se acercaba a 1. Este escalamiento puede interpretarse más precisamente como la "rapidez al contestar": mientras más rapidez se acerca a 1 y mientras más demora a 0. El resultado del clustering con la variable cohesión se muestra en la Tabla 13:

Materia y período	**AA1920**	**AA2021**	**AA2122**	**MDMS1920**	**MDMS2021**	**MDMS2122**
# estudiantes	27	55	45	10	28	22
# mensajes	302	192	222	132	240	82
# clusters formados	3	4	3	3	3	3
# miembros por clúster	37; 11; 254	3; 23; 3; 163;	15; 16; 191	17; 104; 11	198; 20; 22	13; 4; 65
Inercia	4.05	0.67	1.66	1.55	3.21	1.65

Tabla 13 Resultados de clustering con variable Cohesión

En la mayoría de datasets se formaron 3 clústeres. Los datos muestran que los valores de inercia son más bajos que aquellos obtenidos en el clustering por comunicación. Esto nos permite concluir que existe mayor homogeneidad en los clústeres formados con la variable cohesión.

Posterior a la ejecución del clustering se almacenan en un archivo general todos los mensajes intercambiados de todas las materias y períodos.

Creación de matriz de correlación de estudiantes y clústeres formados

El clustering por cohesión arrojó grupos de mensajes que tienen similitudes por los factores de cohesión: Probabilidad Afectiva y Tiempo de Respuesta. Sin embargo, se requiere trasladar el concepto de cohesión a los individuos, para conocer su disposición para interactuar con otros estudiantes. En primer lugar, se creó una matriz de adyacencia para identificar la presencia o ausencia de los estudiantes asignados en cada clúster. A continuación, se creó la matriz de correlación de Pearson para evaluar la correlación de cada par estudiantes. Se calcula la correlación para evaluar

hasta qué punto (linealmente) están relacionados dos estudiantes. La salida del cuaderno Jupyter Notebook consta en la Figura 17:

```
In [604]: matrizCorrPearson
```

Out[604]:

	Estu-83	Estu-29	Prof-72	Estu-65	Estu-150	Coor-36	Estu-132	Estu-43	Estu-68	Estu-152	...	Estu-86	Estu-129
Estu-83	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
Estu-29	0.0	1.0	1.0	0.5	0.5	0.5	1.0	0.5	0.0	1.0	...	1.0	0.5
Prof-72	0.0	1.0	1.0	0.5	0.5	0.5	1.0	0.5	0.0	1.0	...	1.0	0.5
Estu-65	0.0	0.5	0.5	1.0	-0.5	-0.5	0.5	-0.5	0.0	0.5	...	0.5	1.0
Estu-150	0.0	0.5	0.5	-0.5	1.0	1.0	0.5	1.0	0.0	0.5	...	0.5	-0.5
Coor-36	0.0	0.5	0.5	-0.5	1.0	1.0	0.5	1.0	0.0	0.5	...	0.5	-0.5
Estu-132	0.0	1.0	1.0	0.5	0.5	0.5	1.0	0.5	0.0	1.0	...	1.0	0.5
Estu-43	0.0	0.5	0.5	-0.5	1.0	1.0	0.5	1.0	0.0	0.5	...	0.5	-0.5
Estu-68	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
Estu-152	0.0	1.0	1.0	0.5	0.5	0.5	1.0	0.5	0.0	1.0	...	1.0	0.5

Figura 17 Cálculo de la matriz de correlación de Pearson

A continuación, se muestra en la Figura 18, la matriz de correlación para la materia Aprendizaje Automático 21-22: La coloración de la matriz permite reconocer que existen distintos niveles de correlación entre cada par de estudiantes, por ejemplo, las secciones de color rojo indican que esos pares de estudiantes tienen afinidad por la cohesión demostrada en sus mensajes y que hizo que fueran asignados en distintos clústeres donde coincidieron como individuos. Las zonas hacia el color azul demuestran menor cohesión entre cada par de estudiantes.

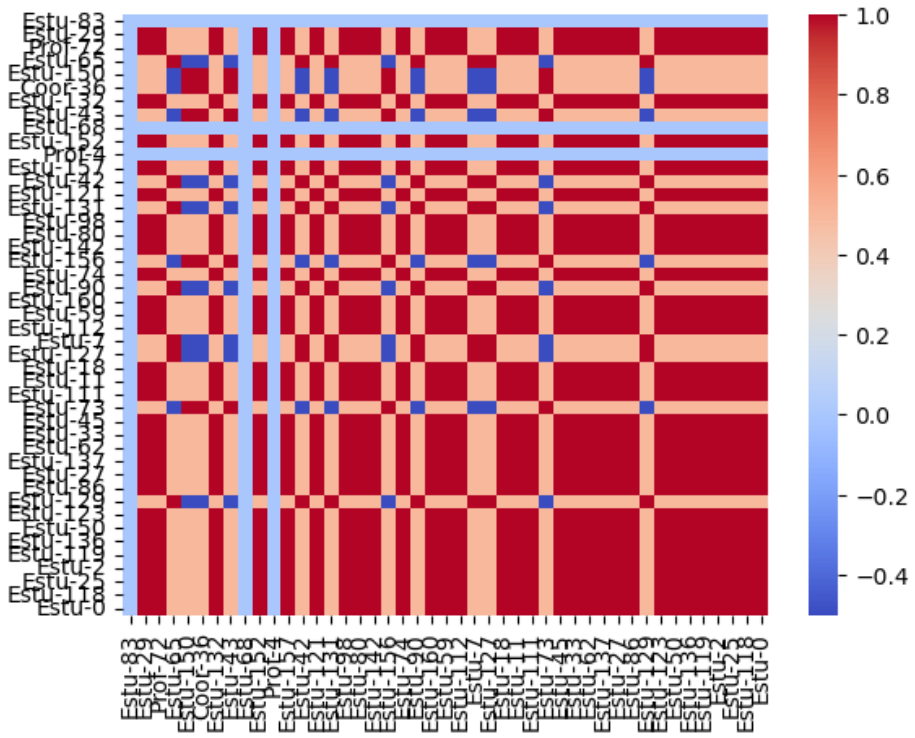

Figura 18 Matriz de correlación de Pearson para la cohesión - AA 21-22

Esta matriz será usada más adelante para seleccionar estudiantes con los cuales la cohesión es mayor. Los datos de esta matriz se almacenan en archivos de acuerdo con la materia y el periodo de estudio.

Análisis de centroides

Se procede a analizar los centroides obtenidos al realizar el clustering con los atributos de comunicación y cohesión. El objetivo es revisar las similitudes o diferencias de los centroides para estimar posibles rangos para los atributos y con estos umbrales ofrecer opciones para la formación de grupos.

Por cada materia y año se aplicó el clustering K-means al dataset correspondiente, de modo que se obtuvieron 3 o 4 centroides por materia y por año (ver tablas 12 y13: cuadros de resultado del clustering para las variables cohesión y comunicación). La Figura 19 muestra los centroides graficados para cada materia y período para la variable Comunicación:

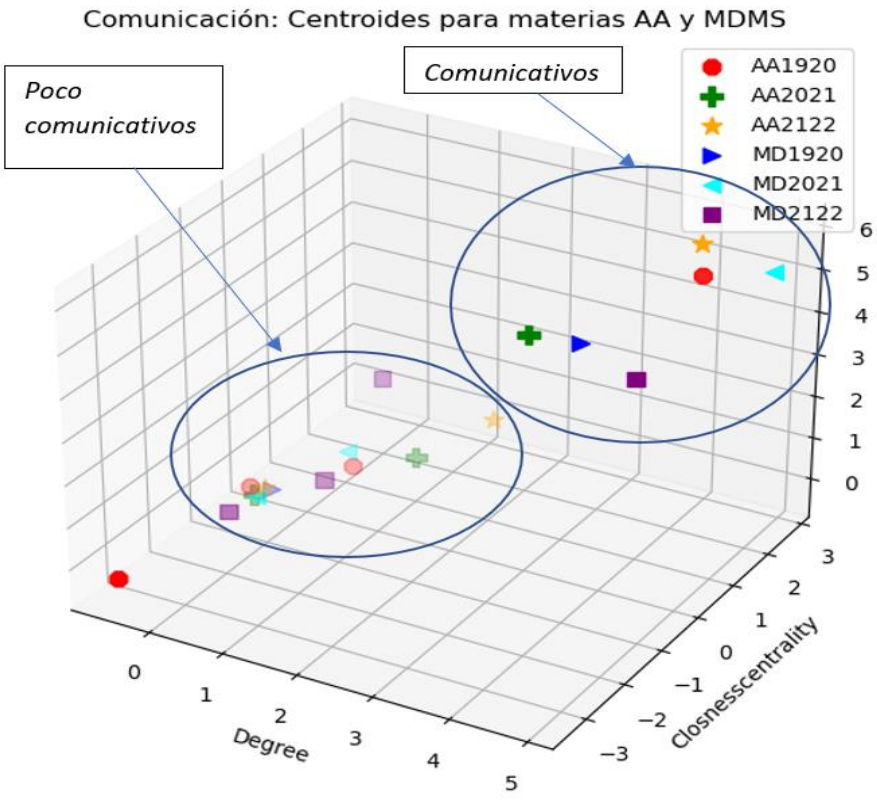

Figura 19 Centroides para datasets agrupados por Comunicación

Para la variable Cohesión de igual manera fueron graficados los centroides obtenidos para todos los datasets, esto se muestra en la Figura 20:

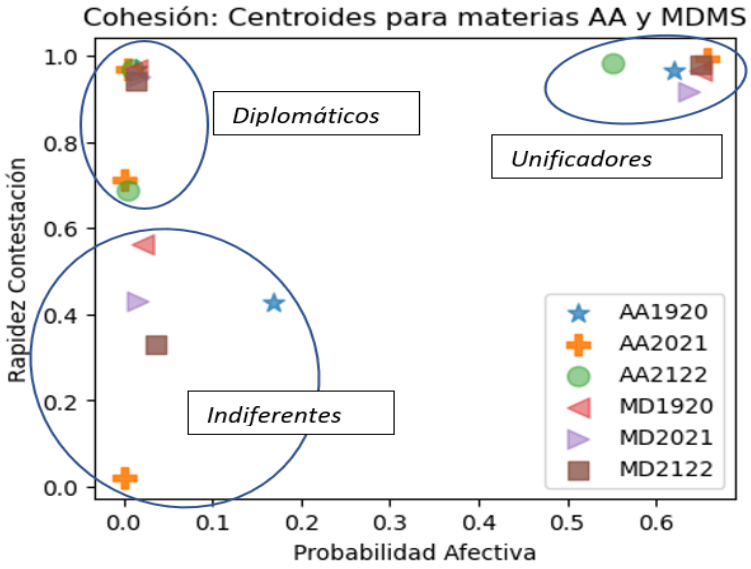

Figura 20 Centroides para datasets agrupados por Cohesión

El análisis de los clústeres para los grupos formados en cada dataset permite distinguir ciertas separaciones en los centroides de los clústeres formados al emplear las variables Comunicación y Cohesión.

Centroides de Comunicación. Se aprecian dos separaciones a las cuales se les asigna una denominación:

- **Comunicativos**. Los centroides que muestran valores mayores para grado, positivos en cercanía y mayores para centralidad de intermediación. Este grupo, engloba a estudiantes que intentan generar una comunicación fluida en los foros, haciendo aportes propios y comunicándose con los demás.
- **Poco comunicativos**. Los centroides que muestran menores valores para grado, positivos en cercanía y menores para centralidad de intermediación. Esto grupo abarca aquellos que quieren comunicar lo necesario, posiblemente solo para cumplir con la tarea.

Las coordenadas mínimas y máximas de cada grupo de clústeres ayudan a definir los límites para cada variable, y esto a su vez constituye el punto de partida para dividir el número total de estudiantes en los dos grupos: Comunicativos y Poco Comunicativos.

En el gráfico de los centroides de comunicación se aprecia que el grado (Degree) viene a ser el criterio que marca de forma más clara la división entre los grupos Comunicativos y Poco Comunicativos. Para este análisis específicamente un grado mayor o igual que 2 distingue a los estudiantes “Comunicativos”; mientras que aquellos que posean un menor grado serán catalogados como “Poco Comunicativos”. Las métricas intermediación y cercanía no ayudan a distinguir claramente la división entre los grupos. Sin embargo, es consistente que el grado ayude a marcar los grupos porque es una medida de cuán conectado está un estudiante a otros estudiantes en la red.

Centroides de Cohesión. Se aprecian tres separaciones:

- **Unificadores**. Estudiantes que brindan una contestación pronta y a través del contenido de sus mensajes muestran un tono de comunicación positivo con sus pares. Contestan y buscan aportar al grupo.

- **Diplomáticos**. Aquellos que contestan pronto, pero establecen una comunicación muy básica con sus pares. Contestan en respuesta a la tarea, pero no aportan significativamente al grupo.
- **Indiferentes**. Aquellos que no contestan pronto y establecen una comunicación muy básica con sus pares. Responden porque la tarea les obliga y su aporte al grupo es limitado.

En el caso de la cohesión, ambas variables aportan a la división de los grupos, por lo tanto, se definieron los siguientes límites: Unificadores (Probabilidad afectiva >= 0.5 & Rapidez de Contestación >= 0.6); Diplomáticos (Probabilidad afectiva < 0.5 & Rapidez de Contestación >= 0.6); Indiferentes (Rapidez de contestación < 0.6).

Es importante apreciar que no existe un grupo que cubra el lado inferior derecho del cuadrante de centroides de cohesión, y existe algún sentido para esta situación porque resultaría un tanto contradictorio intentar comunicar un discurso positivo a otro estudiante y contestar de forma tardía o poco oportuna. Por esta situación, el grupo de Indiferentes podría cubrir los dos cuadrantes inferiores (lado derecho e izquierdo).

Los datos de los centroides fueron guardados en un archivo de texto, cuando se realizó la ejecución respectiva.

3.6.4. Etapa 4. Definición de estrategias

Se proponen dos estrategias para la formación de equipos. En las dos estrategias participa un agente externo: el tutor o profesor. Las dos estrategias consideran fijar primero una de las variables: la comunicación, y luego trabajar con la variable cohesión.

Primera estrategia: formar un equipo armonioso

Se escogen miembros que se complementen, según las agrupaciones definidas en este estudio, para garantizar la existencia tanto de la comunicación y la cohesión. En este caso, el criterio primario es la Comunicación y el criterio secundario es la Cohesión. Los participantes se combinan según se indica en la Tabla 14:

Grupo base	Adicional	Adicional	Adicional
Comunicativos (2)		Diplomáticos (1)	Indiferentes (1)
Poco comunicativos (2)	Unificadores (1)	Diplomáticos (1)	

Tabla 14 Formación de equipos - 1ra estrategia

Operativamente la estrategia consiste en organizar a los estudiantes por el nivel de comunicación (Grado) desde el mayor al menor y luego formar pares. Para cada par formado se buscan dos compañeros según la composición de la tabla para la primera estrategia. Se hace un intento para encontrar los dos miembros adicionales. De no encontrar los adicionales, el equipo podría llegar a tener al menos dos miembros por ser parte del grupo base.

Segunda estrategia: formar un equipo extremo

Esta es una estrategia enfocada en el mundo real, pero de forma extrema; en la misma se considera que personas diversas pueden participar en un equipo y que además poseen valores extremos en las variables analizadas.

En este caso se organizan a los participantes de acuerdo con su nivel de comunicación (Grado), de menor a mayor. Luego se juntan dos miembros opuestos según la variable Comunicación, es decir una persona comunicativa y otra no comunicativa. Para completar el grupo, se busca en el resto de los candidatos a aquellos con los que el grupo base minimiza la cohesión. El detalle de la estrategia consta en la Tabla 15:

Grupo base (2)	Adicionales (2)
Comunicativo (1) Poco Comunicativo (1)	Del resto de posibles miembros, buscar aquellos con los cuales el grupo base minimice la cohesión.

Tabla 15 Formación de equipos - 2da estrategia

Para minimizar la cohesión en esta segunda estrategia se emplea la matriz de correlación de Pearson desarrollada en la etapa 3 (Modelamiento), puesto que esta matriz permite identificar la correlación entre pares de estudiantes, de manera que se obtiene la suma de correlaciones del grupo base más cada posible candidato. Se eligen los candidatos con los cuales la suma es menor.

3.6.5. Etapa 5. Evaluación

Entrada	Archivos de centroides; archivo de estudiantes; archivo de mensajes; archivos de matrices de correlación por cada materia y período.
Salida	Grupos formados
Archivos	formacion_grupos_ccomunicacion.ipynb; formacion_grupos_eextrema.ipynb

Tabla 16 Entradas y salidas en Etapa 4

Para evaluar las estrategias planteadas de forma ideal se debería obtener información etiquetada a posteriori, es decir de los equipos formados en un cuatrimestre para alguna de las materias. Los grupos realmente formados en el cuatrimestre podrían contrastarse con los que son generados por alguna de las propuestas.

Sin embargo, para poner a prueba la programación realizada y el funcionamiento de las propuestas se hizo la evaluación de las estrategias con los datasets originales proporcionados. Al ejecutar las dos estrategias se encontraron ventajas y desventajas para cada una de las estrategias.

La primera estrategia -armoniosa- hace un solo intento para encontrar dos miembros adicionales, y de no encontrarlos podría dejar al equipo con solo dos elementos e incluso hasta 3. En la Figura 21 se aprecia la salida del algoritmo de formación de equipos para la primera estrategia:

Grupos obtenidos por 1er Criterio: COMUNICACION, 2do: COHESION

```
dfFinalGrupos.sort_values(['Grupo', 'Tipo']).head(10)
```

	Degree	estudiante	Disponible	Grupo	Tipo
145	-0.960425	Estu-144	1	0	0
134	2.345579	Estu-21	0	1	C
141	2.138954	Estu-3	0	1	C
140	0.589265	Estu-39	0	1	D
136	2.035641	Estu-48	0	1	I
138	1.105828	Estu-35	0	2	C
152	0.899203	Estu-89	0	2	C
154	-0.857112	Estu-153	0	2	D
137	0.176014	Estu-54	0	2	I
149	0.589265	Estu-26	0	3	C

Figura 21 Detalle de grupos formados luego de ejecutar la primera estrategia

La columna “Disponible” con valor 1 y “Grupo” con valor 0, marca a los estudiantes que no fueron incorporados a un grupo, por lo tanto, el tutor los podrá unir al grupo que considere más conveniente. La columna “Grupo” indica el número de grupo al que fue asignado por el algoritmo. La columna “Tipo” indica la categorización del elemento del grupo (Comunicativo, No Comunicativo, Unificador, Diplomático, Indiferente). Nótese que en la primera estrategia solamente se incluirán miembros Diplomáticos (D) e Indiferentes (I), los Unificadores (U) no son considerados en esta primera estrategia. Los grupos finales formados aparecen en la Figura 22.

```
dfFinalGrupos.groupby(by=['Grupo'])['estudiante'].count()
```

```
Grupo
0    1
1    4
2    4
3    4
4    3
5    4
6    4
7    3
```

Figura 22 Grupos formados y número de participantes – 1ra estrategia

Por otra parte, la segunda estrategia -extrema- completa más equipos de 4 elementos, porque siempre buscará un par de miembros que minimicen la cohesión hasta que se agoten los candidatos disponibles. El resultado de la ejecución de la segunda estrategia se muestra en la Figura 23:

```
dfFinalGrupos.sort_values(['Grupo', 'Tipo']).head(10)
```

	Degree	estudiante	Disponible	Grupo	Tipo
0	3.843343	Estu-68	0	1	C
1	2.939027	Estu-83	0	1	NC
5	1.294816	Estu-43	0	1	NC
42	-0.596026	Estu-112	0	1	PC
2	1.952500	Prof-72	0	2	C
3	1.952500	Estu-65	0	2	NC
10	-0.102763	Estu-150	0	2	NC
41	-0.596026	Estu-45	0	2	PC
4	1.870290	Estu-131	0	3	C
11	-0.184974	Estu-73	0	3	NC

Figura 23 Detalle de grupos formados con la segunda estrategia

Nótese en este caso que en la columna “Tipo” de la figura anterior constan los dos elementos extremos en la comunicación: Comunicativo (C) y Poco Comunicativo (PC); del resto de elementos no se conoce su tipo (NC) puesto que el algoritmo simplemente busca dos elementos cualesquiera con los que se minimice la cohesión. Estos resultados constan en la Figura 24:

```
dfFinalGrupos.groupby(by=['Grupo'])['estudiante'].count()
Grupo
1     4
2     4
3     4
4     4
5     4
6     4
7     4
8     4
9     4
10    4
11    3
Name: estudiante, dtype: int64
```

Figura 24 Grupos formados y participantes – 2da estrategia

Ambas estrategias permiten que queden candidatos “sueltos” que no lograron incorporarse en un grupo de cuatro personas. Estos candidatos pueden ser identificados y el tutor puedo asignarlos a su criterio a otro equipo. El tutor puede decidir esa asignación como un refuerzo o como un elemento disruptivo en el equipo.

Capítulo 4

4. Conclusiones y trabajos futuros

4.1. Conclusiones

(Subobjetivo 1.1) Este trabajo permitió documentar las características del proceso de formación de equipos mediante una tabla de resumen de los criterios comúnmente considerados para la formación de equipos.

Las perspectivas desde las Ciencias Organizacionales y Psicológicas permitieron comprender el proceso de formación de equipos y considerar los factores sociales que influencian la formación de equipos, así como los factores más relevantes para su fundación y permanencia: comunicación y cohesión.

(Subobjetivo 1.2) Este trabajo también permitió identificar al menos tres taxonomías propuestas para la formación de equipos: a) Participación endógena y exógena; b) Basada en la comunidad y basada en la tarea; 3) Basada en tres dimensiones de los equipos: diferenciación de habilidades, diferenciación de la autoridad y estabilidad temporal.

(Subobjetivo 1.3) Este trabajo permitió identificar las técnicas computacionales empleadas para la solución del TFP; entre las más recientemente usadas se encuentran: programación lineal, técnicas heurísticas, algoritmos genéticos, algoritmos aproximados y enfoques basados en redes sociales.

(Subobjetivo 2.1) La propuesta creada para el montaje de equipos considera la taxonomía de tipo endógena/exógena (Juárez et al. 2021), y para este caso resulta ser exógena porque un agente externo, el profesor/tutor forma los equipos. La propuesta genera equipos optimizados (Gómez-Zara et al. 2020), ya que el tutor señala la estrategia a utilizar. Los criterios empleados para formar los equipos están asociados a la maximización de las variables: comunicación y cohesión.

La variable comunicación fue representada por métricas obtenidas a partir del análisis de redes sociales: grado, intermediación y cercanía. La variable cohesión se midió a través de una variable proxy que valoró la cohesión mediante el contenido de los mensajes intercambiados por los estudiantes, empleando dos variables: probabilidad afectiva del mensaje y tiempo de respuesta.

(Subobjetivo 2.2) La técnica usada para la formación de grupos es una de tipo no supervisado, debido a que los datos proporcionados no tienen su salida etiquetada, es decir no existen grupos definidos a priori. El algoritmo K-means fue escogido para obtener clústeres considerando las variables comunicación y cohesión, representadas por las métricas explicadas anteriormente.

La ventaja de usar K-means es que su uso es sencillo y fácil de entender. Tanto las observaciones como sus centroides pueden ser graficados para tener una mejor comprensión de los grupos formados. Sin embargo, también se debe tomar en consideración que, al tratase de un método basado en distancias, el desempeño del algoritmo puede ser pobre al contar con muchas características en el dataset. En estos casos se recomienda una reducción de dimensiones.

(Subobjetivo 2.3) Este trabajo buscó aprovechar además de las conexiones establecidas por medio de los foros, las características de los mensajes para conocer el tono (positivo, negativo o neutro) del contenido y el tiempo que tomó en ser contestado.

Este trabajo cumplió con el Objetivo No. 3, relacionado a la definición de estrategias para la formación de equipos, puesto que formuló dos tácticas para ensamblar equipos: la primera apuntado al equilibro del equipo al buscar miembros que balanceen la comunicación y la cohesión. La segunda estrategia fue de tipo discordante, buscando elementos opuestos de cada variable; posiblemente esta segunda estrategia se acerca más a los escenarios de la vida real. La primera estrategia puede producir equipos incompletos cuando no encuentre algún tipo de miembro, sin embargo, la segunda estrategia permite completar todas las veces equipos de 4 miembros.

La propuesta realizada en este trabajo busca brindar una alternativa sencilla para la formación de equipos, particularmente, que su operación pueda ser entendida por usuarios no expertos en redes sociales o algoritmos.

La propuesta inicial fue generar un algoritmo para la formación de equipos que se adapte a varios entornos y que pueda considerar diversos criterios, es decir proveer una herramienta general. Sin embargo, esto no es del todo posible, porque según se reconoce en la bibliografía estudiada y en la práctica realizada, es necesario considerar varias características del entorno donde serán formados los equipos:

contexto, habilidades obligatorias, tipo de tareas a ejecutar, formación de uno o múltiples equipos y de forma muy relevante el criterio que se pretende optimizar en el equipo.

La creación de una propuesta que se adapte a varios entornos no es del todo factible, sin embargo, se pueden crear iniciativas que se enfoquen en atender áreas específicas que requieran la formación de equipos, como el proyecto CATME que está orientado a la Academia. La propuesta presentada en este trabajo puede ser una referencia para los profesores/tutores que imparten materias en la UNED.

La propuesta de estrategias para formación de equipos desarrollada se puede aplicar para el ensamblaje de equipos orientado a los entornos virtuales de estudios, específicamente los de maestría.

Las herramientas para formación de equipos deben contar con características que hagan amigable su uso y otorguen flexibilidad para escoger los criterios a considerar durante la formación del equipo. La propuesta hecha tiene fijados dos criterios y la entrada que facilita el tutor es el tipo de estrategia que desea emplear. Este trabajo podría extenderse para desarrollar una interfaz más amigable.

La información de las redes sociales permite descubrir las conexiones sociales y de comunicación que existen entre sus miembros. Un sistema de formación de equipos, que toma en cuenta este insumo, podría reducir los costos iniciales de comunicación del equipo y hacia adelante facilitar su cohesión.

El tipo de equipo que resulta de la propuesta realizada en este trabajo es “Optimizado”, según la taxonomía de Gómez-Zará et al. (2020). Las ventajas y desventajas al formar este tipo de equipos son:

a) Los equipos formados pueden ser reproducidos de manera automática y sencilla.
b) Las propuestas de formación requieren que el tutor brinde mínimos insumos al sistema, por ejemplo: criterio a usar y número de alumnos a procesar.
c) La desventaja principal es que el usuario formador de equipos no tiene comprensión total del procesamiento que realiza el algoritmo; el tutor se halla frente a una caja negra.

4.2. Trabajos futuros

Las posibilidades para desarrollar trabajos futuros se extienden al mejorar o añadir nuevos criterios para la formación de equipos, enfocándose en áreas específicas. Por este motivo se plantean las siguientes ideas como posibles líneas para continuar trabajos futuros, y que pueden apoyarse en datos provenientes de redes sociales:

- **Proponer nuevas características para la formación de equipos**. Por ejemplo, usando el estudio de Hoegl y Gemuenden (2001) pueden incorporarse o combinarse otras características de la calidad de trabajo en equipo TWQ. Nótese que estas características se ajustan a equipos innovadores y sin lugar a duda aquellos equipos que funcionan en el mundo de la educación virtual deben serlo.

- **Proponer que en la formación del equipo se defina un posible líder de inicio.** Dentro del equipo recién formado se podría proponer un líder para dirigir el grupo; por ejemplo, se podría proponer a aquel miembro que tenga más alto grado.

- **Medir la calidad de la entrega en los equipos de trabajo.** Luego de la formación de equipos de trabajo, es necesario validar el desempeño de este. Uno de los autores revisados, (Huckman R., Bradley D., 2009) propone medir el desempeño operacional de cara al cliente (en este caso sería el profesor), y podrían ser considerados dos aspectos: (1) la calidad del resultado y (2) el cumplimiento del cronograma. Esto podría realizarse mediante el seguimiento que el docente efectúe a los grupos formados y los trabajos entregados.

- **Realizar un análisis de formación de equipos con redes dinámicas.** Las fuerzas de las conexiones pueden cambiar con el tiempo porque los individuos ingresan o salen de la red, de manera que este fenómeno puede ser considerado al formar equipos.

- **Explorar otras redes sociales**. Se puede examinar cuán bien los equipos pueden ser creados en diferentes redes sociales que exhiban distintos grados de homofilia (la disposición de los individuos a relacionarse con otros que son similares en varios atributos). Por ejemplo, si las únicas conexiones se dan entre expertos con las mismas habilidades, cubrir un conjunto más amplio de

habilidades con un grupo estrechamente conectado puede ser difícil. (Anagnostopoulos A., Becchetti L., Castillo C., Gionis A., & Leonardi S., 2012)

- **Identificar mediadores en el equipo formado**. Proponer el descubrimiento de potenciales mediadores en los equipos si las conexiones sociales en el equipo son pobres.

Bibliografía

Addanki B., Durga B. (2022). *Algorithms for team formation based on the degree distribution of the social networks.*

Ahmed M., Seraj R., Islam S. (2020). The k-means Algorithm: A Comprehensive Survey and Performance Evaluation. *Electronics*.

Anagnostopoulos A., Becchetti L., Castillo C., Gionis A., & Leonardi S. (2012). Online team formation in social networks. *In Proceedings of the 21st international conference*, 839-848.

Anaya A.R., Letón E., Luque M. (2023). Team assembly approach based in social modelling. *In press*.

Ander-Egg E., Aguilar, M. (2001). *El trabajo en Equipo.* México: Progreso.

Andrejczuk E., Berger R., Rodriguez-Aguilar J., Sierra C., & Marín-Puchades V. (2018). The composition and formation of effective teams: computer science meets organizational psychology. *The Knowledge Engineering Review*.

Bahargam S., Golshan B., Lappas T., & Terzi E. (2019). A team-formation algorithm for faultline minimization. *Expert Systems with Applications*, 441-455.

Baiden B., Price A. (2011). The effect of integration on project delivery team effectiveness. *International Journal of Project Management*, 129-136.

Baltos G., & Mitsopoulou Z. (2007). Team formation under normal versus crisis situations: leaders' assessments of task requirements and selection of team members. *Naval Postgraduate School Monterey CA*.

Beal D. J., Cohen R. R., Burke, M. J., & McLendon, C. L. (2003). Cohesion and Performance in Groups: A Meta-Analytic Clarification of Construct Relations. *Journal of Applied Psychology*, 989-1004.

Bell S., Brown S., Colaneri A., & Outland N. (2018). Team composition and the ABCs of teamwork. *American Psychologist*, 349-362.

Bell, S. (2007). Deep-level composition variables as predictors of team performance: a meta-analysis. *Journal of Applied Psychology*, 595.

Berktas N., Yaman H. (2019). Team as a Service: Team Formation on Social Networks. Obtenido de Optmization Online.

Boyce-Jacino C., Harrington N. (2022). Optimizing Team Staffing: A Review of Computational Approaches to Team Formation. *Consortium of Universities Washington DC*, 70.

Castro M., Tumibay G. (2019). A literature review: efficacy of online learning courses for higher education institution using meta-analysis. *Education and Information Technologies*, 1-19.

Dorn C., Dustdar S. (2010). Composing near-optimal expert teams: A trade-off between skills. *Proceedings of the International Conference on Cooperative*.

Dwivedula R., Bredillet N. (2010). Profiling Work Motivation of Project Workers. *International Journal of Project Management*, 158-165.

Esgario J., Da Silva I., Krohling R. (2019). Application of Genetic Algorithms to the Multiple Team Formation Problem.

Frager, R. (1987). *The Influence of Abraham Maslow. Motivation and Personality, Abraham H. Maslow.* Harper and row.

Freeman L. (1979). Centrality in Social Networks. Social Networks. 215-239.

Furht B. (2010). *Handbook of social network technologies and applications.* Florida: Springer.

Géron A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition.* O'Reilly.

Gómez S. (2019). Centrality in Networks: Finding the Most Important Nodes. *Business and Consumer Analytics: New Ideas*, 401-433.

Gómez-Zará D., Dechurch L., Contractor N. (2020). A Taxonomy of Team-Assembly Systems: Understanding How People Use Technologies to Form Teams. *Association for Computing Machinery*, 1-36.

Gutiérrez H., Astudillo C., Ballesteros-Pérez P., Mora-Meliá D., Candia-Vejar A. (2016). The multiple team formation problem using sociometry. *Computers & Operations Research*, 150-162.

Hackman J., Vidmar N. (1970). Effects of size and task type on group performance and member reactions. *Sociometry*, 37-54.

Hage P., Harary F. (1995). ccentricity and centrality in networks. . *Social Networks,*, 57-63.

Halfhill T., Sundstrom E., Lahner J., Calderone W., & Nielsen T. (2005). Group personality composition and group effectiveness: An integrative review of empirical research. *Small Group Research*, 83-105.

Hoegl M., Gemuenden H. (2001). Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence. *Organization Science*, 435-449.

Hollenbeck J., Beersma B., & Schouten M. (2012). Beyond team types and taxonomies: A dimensional scaling conceptualization for team description. *Academy of Management Review*, 82-106.

Horwitz S., Horwitz I. (2007). The effects of team diversity on team outcomes: A meta-analytic review of team demography. *Journal of Management*, 987-1015.

Huckman R., Bradley D. (2009). Team Familiarity, Role Experience, and Performance: Evidence from Indian Software Services. *Management Science*, 85-100.

Jones A. (2019). The Tuckman's model implementation, effect, and analysis & the new development of Jones LSI model on a small group. *Journal of Management*, 23-28.

Juárez, J., Santos, C., & Brizuela, C. (2021). A comprehensive review and a taxonomy proposal of team formation problems. *ACM Computing Surveys (CSUR)*, 1-33.

Kauffman H. (2015). A review of predictive factors of student success in and satisfaction with. *Research in Learning Technology*, 23.

Korsgaard M. , Brodt S. , & Sapienza H. (2005). Trust, identity, and attachment:Promoting individuals' cooperation in groups. *The essentials of teamworking: International perspectives*, 37-54.

Lappas T., Liu K., & Terzi E. (2009). Finding a team of experts in social networks. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery*, 467-476.

Lappas T., Liu K., Terzi E. (2009). Finding a team of experts in social networks. *Association for Computing Machinery*, 467–476}.

Levine J., Moreland R. (1990). Progress in small group research. *Annual Review of Psychology*, 585-634.

Liang F., Lawrence S. (2007). *A Goal Programming Approach to the Team Formation Problem.* Obtenido de Leeds School of Business- University of Colorado.

Liu Z., Kang L., Domanska M., Liu S., Sun J., & Fang C. (2018). Social Network Characteristics of Learners in a Course Forum and Their Relationship to Learning Outcomes. *CSEDU*, 15-21.

Lykourentzou I., Antoniou A., Naudet Y., & Dow S. . (2016). Personality matters: Balancing for personality types leads to better outcomes for crowd teams. *In Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, 260-273.

Maurer, I. (2010). How to Build Trust in Inter-organizational Projects: The Impact of Project Sataffing and Project Rewards on the Formation of Trust, Knowledge Acquisition and Product Innovation. *International Journal of Project Management*, 629-637.

Nageh N., Elshamy A., Hassan A., Sami M., Salam M. (2022). Enhanced Heap-Based Optimizer Algorithm for Solving Team Formation Problem. *Computers, Materials \& Continua*, 5245-5268.

Navarro J., Quijano S., Berger R., Meneses R. (2011). Work-groups in organizations: A basic tool to manage increasing complexity and ambiguity. *Papeles del Psicologo*, 17-28.

Palinkas L., Gunderson E., Holland A., Miller C., & Johnson J. C. (2000). Predictors of behavior and performance in extreme environments: The Antarctic space analogue program. *Aviation, space, and environmental medicine*, 619-25.

Putnam L., Myers W. (2000). Lessons learned from studying the SEI core metrics and process productivity (PI) - What We Have Learned. *Crosstalk The Journal of Defense Software Engineering* , 22-24.

Salas E., Sims D., Burke Sh. (2005). Is there a "Big Five" in Teamwork? *Small Group Research*, 55-99.

Salcinov B., Drew M., Dijkstra P., Waddington G., Serpell B. (2022). Factors Influencing Team Performance: What Can Support Teams in High-Performance Sport Team from Other Industries? A Systematic Scoping Review. *Springer Open*.

Sapienza A., Goyal P., Ferrara E. (2019). Deep Neural Networks for Optimal Team Composition. *Frontiers in Big Data*.

Schall, D. (2016). Skill-based team formation in software ecosystems. *In Proc. Int. Workshop Qual. Assurance Comput. Vis. Int. Workshop Digit*, 35-41.

Shahapure K., Nicholas C. (2020). Cluster Quality Analysis Using Silhouette Score. *IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, 747-748.

Sheng C., Tian Y., & Chen M. (2010). Relationships among teamwork behavior, trust, perceived team support, and team commitment. *Social Behavior and Personality:an international journal*, 1297-1305.

Tallent-Runnels M., Thomas J., Lan W., Cooper S., Ahern T., Shaw S., Liu X. (2006). Teaching courses online: A review of the research. *Review of educational research*, 93-135.

Tkatek S., Bahti O., Lmzouari Y., & Abouchabaka J. (2020). Artificial intelligence for improving the optimization of NP-hard problems: a review. *International Journal of Advanced Trends Computer Science and Applications*, 7411-7420.

Tuckman, B.W., and M.A. Jensen. (1977). Stages of small-group development revisited. *Group & Organization Studies*, 419-427.

Anexos

A) Cuadernos programados en Python con Jupyter Notebooks

1_revision_datasets_v8

February 16, 2024

0.0.1 Descripción

Este notebook permite hacer una limpieza de los dataset para preparlos para el análisis de redes sociales con Gephi; posteriormente servirán para generar un dataset inicial para clustering.

La entrada para este notebook son los archivos originales de foros provistos por el profesor.

La salida es un archivo para cargarlo a Gephi; y otro para trabajar con técnicas de clustering.

[1]:
```
import pandas as pd
#print(pd.__version__)
```

[2]:
```
#check python version
from platform import python_version
#print(python_version())
```

[3]:
```
#Para realizar el análisis de sentimientos (una un modelo bayesiano)
from sentiment_analysis_spanish import sentiment_analysis
```

[4]:
```
import matplotlib.pyplot as plt
import seaborn as sn
```

0.0.2 1. Leer archivos csv del dataset

Para la mayoría de datasets el separador de campos fue ";", sin embargo existieron dos archivos con separador ",".

[8]:
```
#antes solo usamos data
directorio = 'dataset_ultimo/'
#entrada = 'AA-I-foro-19-20-anon.csv'
#salida = 'AA-I-foro-19-20-anon_GP1.csv'

#va con , !! falla 'AA-I-foro-21-22-anon.csv'; MDMS-foro-20-21-anon.csv
entrada = 'MDMS-foro-21-22-anon.csv'
salida = 'MDMS-foro-21-22-anon_GP.csv'
salidaModelo = 'MDMS-foro-21-22-anon_MK.csv'

nombreArchivoIn = directorio + entrada
```

```
nombreArchivoOut = directorio + salida
nombreArchivoOutModelo = directorio + salidaModelo

#df = pd.read_csv(nombreArchivoIn, sep=';')
df = pd.read_csv(nombreArchivoIn, sep=';')
```

[9]:
```
#Ver el contenido del dataframe
df.head(3)
```

[10]:
```
#Analizar las estadísticas de las columnas con formato numérico
df.describe()
```

0.0.3 2. Extraer las columnas que interesan del dataframe

Algunas columnas no tendrán utilidad en el análisis. A continuación se descartan algunas columnas: idAsignatura, Asignatura, idForo, idHilo, idMensaje,Responde a idMensaje, Titulo mensaje.

Inicialmente no tendrían utilidad en el análisis de redes sociales, porque únicamente necesitamos los participantes en la comunicación, el tiempo en el que establecieron contacto y la información que intercambiaron, es decir el texto del mensaje.

[11]:
```
df_columns=df.columns.values.tolist()
print(df_columns)
```

2.1 Verificar tipos de columnas

[12]:
```
df.dtypes
```

0.0.4 3. Crear un nuevo dataframe con las columnas seleccionadas y prepararlo

[13]:
```
dfsna = df[['anon', 'Foro', 'Hilo', 'idMensaje', 'Responde a idMensaje',␣
 ↪'idAutor', 'Fecha', 'Hora', 'Titulo mensaje', 'Texto mensaje', 'Caracteres␣
 ↪mensaje']].copy()
dfsna.head(3)
```

[14]:
```
dfsna[dfsna['Responde a idMensaje']==␣
 ↪'9da654d4187f8b42789e861c6fc2a474cb5941d5_1']
```

[15]:
```
#Cuàntos registros tenemos?
#Resultado obtenido: 374
dfsna.count()
```

3.1 Unir los campos fecha y hora

[16]:
```
#Importante: enviar como parámetro el formato de la fecha
dfsna['fechaMensaje'] = pd.to_datetime(dfsna['Fecha'] + ' ' + dfsna['Hora'],␣
↪format='%d/%m/%Y %H:%M:%S')
dfsna.head(3)
```

[17]:
```
#eliminar las columnas Fecha y hora
dfsna.drop(['Fecha', 'Hora'], axis = 1, inplace = True)
```

[18]:
```
dfsna.head(3)
```

3.2 Cambiar el nombre de las columnas para facilitar el análisis

[19]:
```
#Cambiar los nombres de las columnas para que sea màs fácil usar las columnas
dfsna.rename(columns = {'Foro' : 'foro',
                        'Hilo' : 'hilo',
                        'Responde a idMensaje':'respondeaIdMensaje',
                        'Titulo mensaje' : 'tituloMensaje',
                        'Texto mensaje' : 'textoMensaje',
                        'Caracteres mensaje' : 'caracteresMensaje'
                       }, inplace = True)
dfsna.head(3)
```

[20]:
```
dfsna.dtypes
```

3.3 Validar que todos los mensajes tengan un id

[21]:
```
#Verificar que todos los mensajes tengan un valor.
#Resultado esperado: 0
#Resultado obtenido: 0
dfsna['idMensaje'].isnull().sum()
```

3.4 Validar si todos los mensajes tienen una respuesta. Resultado esperado: es posible que algunos mensajes no tengan una respuesta, pues fueron el inicio de un hilo. Entonces no son respuesta a algún mensaje.

[22]:
```
# Tenemos 72 filas que tienen nulo; estos corresponden a algùn autor que inició␣
↪uno o varios hilos
dfsna['respondeaIdMensaje'].isnull().sum()
```

[23]:
```
#Verificar a quièn corresponden los mensajes que estàn nulos
dfsna[dfsna['respondeaIdMensaje'].isnull()].head(3)
```

3.5 Añadir la variable número de mensajes enviados por cada usuario Obtener el número de mensajes enviado por cada usuario anon.

[24]:
```
#Obtener el número de mensajes enviado por usuario y luego crear un diccionario␣
↪con esos valores
varMensajesEnviados=dfsna[dfsna['idMensaje'].notnull()].groupby('anon')['anon'].
↪count()
dictMsjEnviados=varMensajesEnviados.to_dict()
dictMsjEnviados
```

[25]:
```
#Crear una funcion para devolver el número de mensajes que envió un usuario
def devolverMsjEnviadosDadoUsuario(idUsuario):
    numMensajes=dictMsjEnviados.get(idUsuario)
    if numMensajes is None:
        numMensajes = 0
    return numMensajes

#pruebas de la funciòn: Prof-130; Coor-36
devolverMsjEnviadosDadoUsuario('Coor-36')
```

[26]:
```
#asignar a una columna esta variables
#aplicar la funcion al dataframe:
dfsna['numMsjEnviados']= dfsna['anon'].apply(lambda x:␣
↪devolverMsjEnviadosDadoUsuario(x))
```

3.6 Añadir una variable que indique la iniciativa de los usuarios al crear nuevos hilos

Para cada usuario del foro se contará el número de veces que inició un hilo. Este número será colocado en una columna llamada numHilosIniciados. Esta variables un indicativo de la iniciativa de los usuarios para empezar una comunicación.

Se identificarán los usuarios con su número de hilos en un diccionario, siendo la clave, el identificador del usuario.

[27]:
```
varIniciativaHilo= dfsna[dfsna['respondeaIdMensaje'].isnull()].
↪groupby('anon')['anon'].count()
dictIniciativa=varIniciativaHilo.to_dict()
dictIniciativa
```

[28]:
```
#Crear una funcion para devolver el número de veces que inició un hilo
def devolverHilosDadoUsuario(idUsuario):
    numHilos=dictIniciativa.get(idUsuario)
    if numHilos is None:
        numHilos = 0
    return numHilos
```

[29]:
```
#probar la funcion
numHilosUsuario= devolverHilosDadoUsuario('Estu-8')

if numHilosUsuario is None:
    print('0 hilos')
```

```
else:
    print(numHilosUsuario)
```

[30]:
```
#aplicar la funcion al dataframe:
dfsna['numHilosCreados']= dfsna['anon'].apply(lambda x:␣
↪devolverHilosDadoUsuario(x))
```

[31]:
```
dfsna.head(3)
```

[32]:
```
#buscar usuarios que no crearon hilos
dfsna[dfsna['numHilosCreados']==0].groupby('anon')['anon'].count()
```

[33]:
```
dfsna[dfsna['numHilosCreados'] == 0].head(3)
```

[34]:
```
#Ver los hilos creados por el usuario Estu-115
dfsna[dfsna['anon'] == 'Estu-115'][['anon', 'numHilosCreados']].head(10)
```

[35]:
```
#conocer los usuarios y cuántos hilos crearon
dfsna[['anon', 'numHilosCreados']].groupby(['anon', 'numHilosCreados']).count()
```

Es importante observar los casos distintos que iniciaron mensajes. Estos casos no constarán en los datos para análisis de redes sociales, porque no implican una comunicación origen y destino. Sin embargo estos casos ayudan a identificar la iniciativa de los estudiantes en iniciar comunicaciones. Resultado obtenido: 19 personas que iniciaron mensajes. Es decir estos no fueron respuesta.

[36]:
```
dfsna[dfsna['respondeaIdMensaje'].isnull()]['anon'].value_counts()
```

[39]:
```
dfsna.head(3)
```

3.7 Realizar análisis de sentimientos sobre los comentarios del foro Mediante el análisis de sentimientos, se intenta determinar la actitud de un interlocutor o usuario con respecto a algún tema. La actitud puede ser su juicio o evaluación, estado afectivo (o sea, el estado emocional del autor al momento de escribir), o la intención comunicativa emocional (o sea, el efecto emocional que el autor intenta causar en el lector).

En este caso se evalúa la actitud de la persona que escribe el comentario ante el foro; es decir su actitud al comunicar sus ideas y pareceres al foro de alumnos.

[40]:
```
#llamar a la función que hará el análisis de sentimientos
sentiment = sentiment_analysis.SentimentAnalysisSpanish()
```

[41]:
```
#print(sentiment.sentiment("me gusta la tombola es genial"))
dfsna['probabilidadAfectiva']=dfsna['textoMensaje'].apply(lambda x: sentiment.
↪sentiment(x))
```

[42]:
```
dfsna[['anon', 'textoMensaje','probabilidadAfectiva', 'caracteresMensaje']].
↪head(5)
```

[43]:
```
dfsna[['probabilidadAfectiva', 'caracteresMensaje', 'numHilosCreados']].
 ↪describe()
```

[44]:
```
dfsna['probabilidadAfectiva'].max()
```

[45]:
```
#Observar cuál es el mensaje más positivo
dfsna[dfsna['probabilidadAfectiva']==dfsna['probabilidadAfectiva'].max()]
```

[46]:
```
#Observar el mensaje más negativo
dfsna[dfsna['probabilidadAfectiva']==dfsna['probabilidadAfectiva'].
 ↪min()]['idMensaje']
```

[47]:
```
dfsna[dfsna['idMensaje']=='8479f73023af665434c5ef4857270046b1f073ec_6']['textoMensaje']
```

3.8 Validar si son correctos los códigos anonimizados con los ids largos. El resultado esperado es que al código anonimizado le corresponda un solo id largo. Y que a la inversa ocurra lo mismo: a un id de Autor, le corresponde un solo código anonimizado.

[48]:
```
len(dfsna.groupby(['anon', 'idAutor'])['idAutor'].count())
```

[49]:
```
#Obtener los sujetos distintos tengo en los autores:
len(dfsna.groupby(['idAutor', 'anon'])['anon'].count())
```

[50]:
```
#Obtener cuàntos sujetos distintos tengo en los autores:
dfsna['idAutor'].value_counts().count()
```

[51]:
```
#Los nombres de los anonimizados deberían ser igual 28
#Resultado obtenido: 28
dfsna['anon'].value_counts().count()
```

3.9 Identificar el tiempo que cada usuario se demoró en responder a un mensaje

[52]:
```
#1. Identificar el mensaje con su fecha original
#2. Tomar los ids de mensaje de la columna respondeaIdMensaje y obtener la␣
 ↪fecha del mensaje
#3. Y colocar la fecha del punto 2, en una nueva columna
dfFechas=dfsna[['idMensaje', 'fechaMensaje']]
dictFechaMsj=dfFechas.set_index('idMensaje').to_dict()['fechaMensaje']
dictFechaMsj

#funcion para devolver la hora del mensaje
def devolverFechaMensaje(idMensaje):
    fecha=dictFechaMsj.get(idMensaje)
    ##print(type(idAutor))
    return fecha
```

```
devolverFechaMensaje('da0034b635ce018799348f72ee20b81441468497_5')
```

[53]:
```
dfsna['fechaRespondeaIdMensaje']=dfsna['respondeaIdMensaje'].apply(lambda p:
 ↪devolverFechaMensaje(p))
dfsna.head(3)
```

[54]:
```
#Obtener las diferencias entres las fechas del mensaje original y las␣
 ↪respuestas, para saber cuánto demoró
#calcular la diferencia entre las fechas
dfsna['respuestaHoras']=0
dfsna['respuestaHoras'] = ((dfsna['fechaMensaje'] -␣
 ↪dfsna['fechaRespondeaIdMensaje']).dt.total_seconds())/3600
```

3.10 Hacer un análisis de correlación entre las variables numéricas del dataframe obtenido

[55]:
```
dfsna.head(3)
```

[56]:
```
dfsna.describe()
```

[57]:
```
corrMatrix= dfsna[['caracteresMensaje', 'numHilosCreados',␣
 ↪'probabilidadAfectiva', 'numMsjEnviados', 'respuestaHoras']].corr()
#Se aprecia que no existe una correlación entre las variables.
corrMatrix
```

Al revisar la correlación entre las tres variables numéricas no se aprecia que exista una correlación positiva o negativa entre las variables analizadas.

[58]:
```
plt.figure(figsize = (4, 3))
sn.heatmap(corrMatrix, annot=True)
#sn.set(font_scale=1.4)
plt.show()
```

3.11 Hacer un análisis de la distribución de la probabilidad afectiva de los mensajes

[59]:
```
dfsna['probabilidadAfectiva'].describe()
```

[60]:
```
#figsize: 3 ancho x 5 alto
plt.figure(figsize = (6, 2))
dfsna['probabilidadAfectiva'].plot(kind='kde')
#sn.displot(data=dfsna, x='probabilidadAfectiva', kind='kde')
#plt.show()

#sns.displot(penguins, x="flipper_length_mm", kind="kde", bw_adjust=.25)
#sn.distplot(dfsna['probabilidadAfectiva'], kde=True, hist=True)

#sns.distplot(df['lifeExp'], hist = False, kde = True, label='Americas')
```

[61]:
```
dfsna[dfsna['probabilidadAfectiva']<0]
```

Generar un histograma de la probabilidad Afectiva Se aprecia que la mayoría de comentarios tiene afectividad negativa, es decir, cercanos al valor cero.
Existen muy pocos comentarios escritos con una actitud positiva; sin embargo, los comentarios con afectividad neutra (situados en torno al 0.5) son mayores que los comentarios con afectividad negativa.

Se debe recordar que este dataset corresponde al año de la pandemia: 2019-2020.

[62]:
```
plt.figure(figsize = (6, 2))
sn.histplot(data=dfsna, x='probabilidadAfectiva', bins=100)
```

3.12 Revisar la distribución de la variable numHilosCreados

[63]:
```
plt.figure(figsize = (6, 2))
sn.histplot(data=dfsna, x='numHilosCreados', bins=100)
#dfsna['numHilosCreados'].hist(bins=100)
#dfsna['numHilosCreados'].plot(kind='kde')
```

[64]:
```
#Revisar la distribución de la variable numHilos
#fig, ax = plt.subplots(figsize=(2, 1))
#fig, ax = plt.subplots(figsize=(40, 5))
#plt.figure(figsize = (6, 2))
sn.displot(data=dfsna, x="numHilosCreados", kind="kde", height=2, aspect=2.5)␣
↪#, bw_adjust=.25
plt.show()
```

[65]:
```
#Revisar la variable caracteresMensaje
plt.figure(figsize = (6, 2))
sn.histplot(data=dfsna, x='caracteresMensaje', bins=500)
```

[66]:
```
sn.displot(data=dfsna, x='caracteresMensaje', kind='kde', bw_adjust=0.25,
           height=2, aspect=2.5)
```

[67]:
```
plt.figure(figsize = (6, 2))
sn.histplot(data=dfsna, x='numMsjEnviados', bins=10)
```

0.0.5 4. Obtener el dataframe para el análisis de redes sociales

Para analizar estos datos con Gephi es necesario identificar las columnas origen y destino en las relaciones de comunicación. 1. El origen se identifica por la columna idAutor. 2. El destino será identificado por el autor del mensaje(Responde a idMensaje) al que se contesta; es decir, con el id del mensaje, se busca a su autor.

[68]:
```
dfsna.head(3)
```

4.1 Identificar al autor del mensaje al que un estudiante responde. Es necesario identificar el id del mensaje que responde (columna 'Responde a idMensaje'), y ese valor buscarlo en la columna idMensaje, y a continuación identificar la persona mediante la columna anon

[69]:
```
#Obtener los ids de autores con los ids de mensajes hechos
dfam = dfsna[['idMensaje', 'idAutor']]
dfam.head()
```

[70]:
```
dfam.count()
```

Extraer los ids de los mensajes con su autor respectivo y transformarlo en un diccionario. Así se consultará el mensajeid y se obtendrá el id del autor.

[71]:
```
#'dict', list, 'series', 'split', 'records', 'index'
dictMsjAut=dfam.set_index('idMensaje').to_dict()['idAutor']
dictMsjAut
```

[72]:
```
claves = dictMsjAut.keys()
#print(claves)
```

[73]:
```
valores= dictMsjAut.values()
#print(valores)
```

[74]:
```
#Probar que se recuperar con una clave, un valor
#86009cf7267cb95e867e2fad29b0d95a214c02f0_1␣
 ↪        740c82d452a323d17333f927e2cf5259ee21b321
clave = '86009cf7267cb95e867e2fad29b0d95a214c02f0_1'
print(dictMsjAut.get(clave))
```

Obtener los valores anonimizados con su id de autor. Se usarà el valor anon porque es más fácil de manejar que un texto largo de caracteres.

[75]:
```
dfanid = dfsna[['anon', 'idAutor']]
dfanid.head()
```

[76]:
```
#Se forma un diccionario
dictAnonId=dfanid.set_index('idAutor').to_dict()['anon']
len(dictAnonId)
```

Funciones para devolver un id corto (anon) dado un id largo(valor anonimizado) y para devolver el id corto (anon) dado el id del mensaje.

[77]:
```
#Funcion para devolver el texto corto "anon" (identificación del usuario) dado␣
 ↪el id largo
def devolverAnonDadoIdUsuario(idUsuario):
    nickAnon=dictAnonId.get(idUsuario)
    return nickAnon
```

```
id = '740c82d452a323d17333f927e2cf5259ee21b321'
resultId=devolverAnonDadoIdUsuario(id)
print(resultId)
```

[78]:
```
#Crear una funcion para que devuelva el autor dado un id de mensaje
def devolverAutorDadoMensaje(idMensaje):
    idAutor=dictMsjAut.get(idMensaje)
    ##print(type(idAutor))
    return idAutor

#'86009cf7267cb95e867e2fad29b0d95a214c02f0_1'
claveEj= '86009cf7267cb95e867e2fad29b0d95a214c02f0_1'
resultado=devolverAutorDadoMensaje(claveEj)
print(resultado)
```

[80]:
```
dfsna.head(3)
```

[81]:
```
#crear una función para buscar el id del autor

dfsna['usuarioDestino']= dfsna['respondeaIdMensaje'].apply(lambda y:␣
 ↪devolverAutorDadoMensaje(y))
dfsna['anonDestino']= dfsna['usuarioDestino'].apply(lambda z:␣
 ↪devolverAnonDadoIdUsuario(z))
```

[82]:
```
dfsna.head(3)
#final
```

4.2 Extraer los datos que interesan para el análisis de redes sociales

[83]:
```
#dfsna[['anon', 'anonDestino', 'tituloMensaje', 'textoMensaje', 'fechaMensaje',␣
 ↪'probabilidadAfectiva', 'numHilosCreados', 'numMsjEnviados',␣
 ↪'respuestaHoras' ]].head(4)
dfsna[['anon', 'anonDestino', 'fechaMensaje', 'probabilidadAfectiva',␣
 ↪'numHilosCreados', 'numMsjEnviados', 'respuestaHoras' ]].head(4)
```

[84]:
```
#validar algunos mensajes:
#Mensajes que salen desde Prof-4 hacia otros: 15
#Mensajes que contestan a Prof-4 hacia otros: 28

#dfsna[dfsna['anon'] == 'Prof-4'].count()
dfsna[dfsna['anonDestino'] == 'Prof-4'].count()
```

4.3 Verificar si el usuario al que se contesta es nulo. Existen 72 nulos, es decir estos 72 mensajes fueron el inicio de un hilo

[85]:
```
dfsna['anonDestino'].isnull().sum()
```

4.4 Extraer los datos para el SNA

[86]:
```
dfsna.count()
```

[87]:
```
#cuantas filas tienen el destino no null, es decir existe un autor al que se le␣
↪responde
dfsna[dfsna['anonDestino'].notnull()].count()
```

Resetear el índice para que los 302 registros se numeren desde 0. Estos 302 registros tienen un origen y un destino, es decir cada participante contestó a otro participante.

[88]:
```
dfsna_final= dfsna[dfsna['anonDestino'].notnull()].reset_index(drop=True)
dfsna_final.head(5)
```

[89]:
```
#cambiar la fecha a formato ISO8601
#dfsna_final['timestamp'] =
dfsna_final['timestamp'] = dfsna_final['fechaMensaje'].apply(lambda x: x.
↪isoformat())
```

[90]:
```
dfsna_final.head(3)
```

Es necesario cambiar los nombres de las columnas para poder importar desde Gephi. Deben tener los textos source, target.

[91]:
```
cabecera = ['source', 'target', 'timestamp', 'caracteresMensaje',␣
↪'numHilosCreados', 'probabilidadAfectiva',
           'numMsjEnviados', 'respuestaHoras']
dfsna_final.to_csv(nombreArchivoOut,
            columns=['anon', 'anonDestino', 'timestamp', 'caracteresMensaje',
                     'numHilosCreados', 'probabilidadAfectiva',␣
↪'numMsjEnviados',
                     'respuestaHoras'], index_label='conteo',
            header=cabecera)
```

0.0.6 5. Extraer los datos para aplicar una técnica no supervisada: algoritmo de k-means¶

[92]:
```
dfsna.head(3)
```

[93]:
```
dfsna_final_modelo= dfsna[dfsna['anonDestino'].notnull()].reset_index(drop=True)
dfsna_final_modelo.head(3)
```

[108]:
```
dfsna[['anon', 'anonDestino','textoMensaje', 'idMensaje', 'respondeaIdMensaje',␣
↪'tituloMensaje' ]][(dfsna['anonDestino'] == 'Estu-7') | (dfsna['anon'] ==␣
↪'Estu-7')]
#dfsna['anon', 'nonDestino', 'textoMensaje']
```

[445]:
```
#Datos para Kmeans
#cabeceraModelo = ['source', 'target', 'start', 'textoMensaje',␣
↪'caracteresMensaje',
cabeceraModelo = ['source', 'target', 'start', 'caracteresMensaje',
            'numHilosCreados', 'probabilidadAfectiva', 'numMsjEnviados',␣
↪'respuestaHoras']
dfsna_final_modelo.to_csv(nombreArchivoOutModelo,
             columns=['anon', 'anonDestino', 'fechaMensaje',
                      'caracteresMensaje',␣
↪'numHilosCreados','probabilidadAfectiva',
                     'numMsjEnviados', 'respuestaHoras'], index_label='conteo',
             header=cabeceraModelo)
```

2_crear_matriz_adyacencia_analyze_v9

February 16, 2024

0.0.1 Cuaderno para generar la matriz de adyacencia a partir de los nodos y de las relaciones.

Se crea una matriz de adyacencia por cada una de las siguientes variables:

caracteresmensaje

probabilidadafectiva

respuestahoras

La matriz de adyacencia muestra una relación entre dos nodos adyacentes, marcando un "1" entre estos dos nodos si existe adyacencia; si no existe adyacencia coloca un "0". Para la construcción de la matriz de adyacencia para cada una de las variables señaladas arriba, se hará uso del valor promedio de la variable que se presente entre los dos nodos. Por ejemplo: Si el nodo "A" envió 4 mensajes al nodo "B", para la variable "caracteresmensaje", el valor de adyacencia entre A y B será el promedio del número de caracteres de los mensajes que A envió a B.

De la misma forma se opera con el resto de variables. Para la variable "probabilidadafectiva", la matriz de adyacencia considera la probabilidad afectiva promedio entre los mensajes intercambiados entre dos estudiantes. Si existen varios mensajes se obtiene el promedio del valor de probabilidad afectiva.

0.0.2 Parte I. Crear las matrices de adyacencia

1. Creación de la matriz de adyacencia

[268]:
```
#Leer los nodos: a través de Gephi se obtiene un archivo de nodos y otros de␣
↪aristas.
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
```

[269]:
```
import seaborn as sn
```

[270]:
```
directorio = "dataset_ultimo/matrices_adyacencia_var_msj/"
nombreArchivoNodos = "nodos_AA_2122.csv"
nombreArchivoAristas = "aristas_AA_2122.csv"
dfNodos = pd.read_csv(directorio + nombreArchivoNodos)
```

```
dfNodos.head(3)
```

1.1 Obtener los nodos

[271]:
```
#Obtener solo los nodos para formar la matriz
listaNodos = dfNodos['Label'].tolist()
```

[272]:
```
#Variable global nùmero de nodos
NUM_NODOS = len(listaNodos)
```

[273]:
```
listaNodos
```

1.2 Identificar las aristas entre los nodos

[274]:
```
#dfAristas = pd.read_csv("dataset_ultimo/matriz_adyacencia_AA_1920/
↪aristas_AA_1920.csv")
dfAristas = pd.read_csv(directorio + nombreArchivoAristas)
dfAristas.head(3)
```

[275]:
```
dfAristas.shape
```

[276]:
```
dfAristas[['Source', 'Target']].head(3)
```

[277]:
```
#Realizar algunas consultas con los estudiantes origen y destino
dfAristas[(dfAristas['Source']=='Estu-115') & (dfAristas['Target']=='Estu-39')]
```

[278]:
```
#Buscar quiénes escribieron al estudiante Estu-92 (èl fue el destino-Target).
#Estamos revisando la columna Target
#Resultado obtenido: 3 personas escribieron a Estu-92 (Estu-113, Estu-1,␣
↪Coor-36)
#dfAristas[(dfAristas['Target']=='Estu-92')]

#Buscar a quiénes les escribió el estudiante Estu-92 (él fue el origen-Source)
#Estamos revisando la columna Source
#Resultado obtenido: a 0 personas escribió Estu-92
dfAristas[(dfAristas['Source'] == 'Estu-92')]
```

[279]:
```
#Crear una columna con las relaciones en el dataframe,juntando Source y Target
dfAristas['relacion'] = (dfAristas['Source'] + '_' + dfAristas['Target'])
dfAristas['relacion'].head()
```

[280]:
```
#obtener una lista de las aristas
listaAristas = dfAristas['relacion'].tolist()
listaAristas[0]
```

[281]:
```
#funciòn para crear una matriz poblada con ceros
#Crear y encerar la matriz con valores cero.  La matriz està representada por␣
↪un dataframe
def crearMatriz(lNodos):
    matriz=pd.DataFrame(0, index=lNodos, columns= lNodos)
    return matriz
```

1.3 Cálculo de matrices de adyacencia para otras variables (valor promedio)

[282]:
```
#Con el dataframe original de aristas procedemos a pasar las variables para␣
↪calcular el promedio
def calcularVarPromEntreNodos(dfDeAristas, colOrigen, colDestino,
                                        nodoOrigen, nodoDestino,␣
↪variableCalcular, nmAgregado):

    resultado = dfDeAristas[variableCalcular][(dfDeAristas[colOrigen]==␣
↪nodoOrigen)
                                    & (dfDeAristas[colDestino]== nodoDestino)].
↪aggregate([nmAgregado])

    #el resultado es una serie, pedir el primer elemento para evitar el texto␣
↪"count" o "mean"
    return resultado[0]


#probar la función con promedio (mean)
resultado = calcularVarPromEntreNodos(dfAristas, "Source", "Target", "Coor-36",␣
↪"Coor-36", "caracteresmensaje", "mean")
print("Prueba funcion calcularVariablePromedioEntreNodos: " , resultado)
```

[283]:
```
#probar la función con conteo (count)
resultado1 = calcularVarPromEntreNodos(dfAristas, "Source", "Target",␣
↪"Coor-36", "Estu-115", "Source", "count")
resultado1
```

La siguiente funciòn calcula la matriz de Adyacencia con la variable y agregados enviados como parámetros.

[284]:
```
def poblarMatrizAdyacenciaConVariable(matrizCeros, lNodos, lAristas, dffAristas,
                                      colSource, colTarget, nmVariable,␣
↪nmAgregado):

    #cuenta las veces que el valor en la celda de la matriz es mayor que cero␣
↪para imprimir abajo
    contador=0
```

```
    print("=======>> Calculando matriz de adyacencia para variable: ",␣
↪nmVariable)
    for nodo in lNodos:
        for arista in lAristas:
            source, target = arista.split("_",1)
            if source == nodo:
                valorActual = calcularVarPromEntreNodos(dffAristas, colSource,␣
↪colTarget,
                                                        source, target,␣
↪nmVariable, nmAgregado)
                matrizCeros.loc[source, target] = valorActual

                #imprimir los valores que va colocando en la matriz: promedios␣
↪de la variable
                if valorActual > 0:
                    contador = contador + 1
```

0.0.3 Matriz de adyacencia propiamente

[285]:
```
#Crear una matriz de adyacencia en blanco
matrizAdy1 = crearMatriz(listaNodos)
matrizAdy1.head()
```

[286]:
```
#Poblar la matriz de adyacencia, que tiene pesos
#Se coloca la columna Source, ùnicamente con el objetivo de que cuente;␣
 ↪promediar no aplica
poblarMatrizAdyacenciaConVariable(
    matrizAdy1, listaNodos, listaAristas, dfAristas, "Source", "Target",␣
 ↪"Source", "count")
```

[287]:
```
matrizAdy1.head()
```

[288]:
```
#Verificar la fila 'Estu-92'
#Resultado obtenido: la fila de Estu-92 está llena de ceros.  Este es un nodo␣
 ↪aislado
#matrizAdy1.loc['Estu-92', :]
```

Matriz de adyacencia con el promedio de la longitud del mensaje entre los nodos

[289]:
```
#Crear la matriz con los promedios de la longitud del mensaje entre nodos
matrizPromCaracteresMensaje = crearMatriz(listaNodos)
matrizPromCaracteresMensaje.head(3)
```

[290]:
```
#Poblar la matriz
poblarMatrizAdyacenciaConVariable(matrizPromCaracteresMensaje,
```

```
                                    listaNodos, listaAristas, dfAristas,␣
↪"Source", "Target", "caracteresmensaje", "mean")
```

[293]:
```
matrizPromCaracteresMensaje.head(3)
```

Matriz de adyacencia con el promedio de la probabilidadAfectiva entre los nodos

[294]:
```
matrizPromProbAfectiva = crearMatriz(listaNodos)
poblarMatrizAdyacenciaConVariable(matrizPromProbAfectiva,
                                    listaNodos, listaAristas, dfAristas,␣
↪"Source", "Target", "probabilidadafectiva", "mean")
matrizPromProbAfectiva.head()
```

Matriz de adyacencia con el promedio de la variable respuestahoras

[295]:
```
matrizPromTiempoRespMsj = crearMatriz(listaNodos)
poblarMatrizAdyacenciaConVariable(matrizPromTiempoRespMsj,
                                    listaNodos, listaAristas, dfAristas,␣
↪"Source", "Target", "respuestahoras", "mean")
matrizPromTiempoRespMsj.head()
```

0.0.4 Parte II. Calcular la matrices de correlaciòn de Pearson

[296]:
```
#obtener el nùmero de nodos total, es una variable global
NUM_NODOS
```

[297]:
```
matrizAdy1.head(3)
```

[298]:
```
matrizAdyPearson = crearMatriz(listaNodos)
matrizAdyPearson.head(3)
```

[300]:
```
# Función para realizar el calculo de la correlación de Pearson y llenar la␣
↪matriz creada.
# Al colocar range(NUM_NODOS) realizará iternaciones desde 0 hasta␣
↪(NUM_NODOS-1): de 0 - 26

def calcularCorrPearson(pMatrizAdy, pMatrizCorr, numNodos):

    for i in range(numNodos):
        for j in range(numNodos):
            results = stats.pearsonr(pMatrizAdy.iloc[i],pMatrizAdy.iloc[j])
            pMatrizCorr.iat[i,j] = results[0]

    #devolver la matriz con los coeficientes de Pearson
    return pMatrizCorr
```

[301]:
```
matrizAdyPearson = calcularCorrPearson(matrizAdy1, matrizAdyPearson, NUM_NODOS)
matrizAdyPearson.head(3)
```

[306]:
```
#funcion para cambiar ceros por Nan en una matriz

def cambiarNanPorCeros(pMatriz):
    pMatriz = pMatriz.fillna(0.0)
    return pMatriz

#Probar funcion
matrizAdyPearson = cambiarNanPorCeros(matrizAdyPearson)
```

II.1 Matriz de correlación de Pearson para matriz de adyacencia con pesos

[307]:
```
#Se grafica la matriz de correlaciòn de Pearson.
#Resultado obtenido: se aprecia que uno de los estudiantes no se relacionó con␣
 ↪nadie:
#Tiene una fila y columna de ceros.
plt.subplots(figsize=(30,25))
sn.heatmap(matrizAdyPearson, annot=True)
plt.show()
```

[308]:
```
#calculamos la media de las correlaciones
str(matrizAdyPearson.mean().mean())
# '0.2772875621992368'
```

[309]:
```
#guardar la matriz de Pearson
matrizAdyPearson.to_csv('dataset_ultimo/matrizCPearson_AA1920.csv', sep=",")
```

Utilizar solo la mitad de la matriz. Mejor: utilizar solamente el triàngulo superior de la matriz, ya que sed trata de una matriz simétrica. Usando la mitad de la matriz de Adyacencia completa, se ordenan los valores de la matriz para mirar cuáles son los usuarios más relacionados.

[310]:
```
matrizAdyPearson.shape
```

Crear una función para extraer la parte superior de la matriz simétrica de coeficientes de Pearson

[311]:
```
def obtenerDfParesCorrelacionados(matriz):
    #obtener la parte superior de la matriz
    superiorMatriz = matriz.where(np.triu(np.ones(matriz.shape), k=1).
 ↪astype(bool))
    paresUnicos = superiorMatriz.unstack().dropna()

    #generar un dataframe a partir de la series
    dfParesUnicos = paresUnicos.reset_index()
```

```
    #modificar el dataframe para hacerlo más legible
    dfParesUnicos.rename(columns = {'level_0' : 'Source', 'level_1' : 'Target',␣
 ↪0 : 'CorrPearson'}, inplace=True)

    return dfParesUnicos

#Probar la función
dfParesCorr=obtenerDfParesCorrelacionados(matrizAdyPearson)
dfParesCorr.head(3)
```

[312]:
```
#Verificar si quedaron los pares de la diagonal
#Resultado obtenido: 0 (antes eran 27 elementos)
dfParesCorr[dfParesCorr['Source'] == dfParesCorr['Target']].count()
```

[313]:
```
#Ordenar los pares correlaciones, pero no considerar los de la diagonal␣
 ↪principal
dfParesCorr[dfParesCorr['Source'] != dfParesCorr['Target']].
 ↪sort_values(by='CorrPearson', ascending=True).head(7)
```

[314]:
```
#Cuántos tiene el df
len(dfParesCorr)
```

Verificar si existe ausencia de relación

[315]:
```
#AA1920: 26
#AA2021: 0
#M
dfParesCorr[dfParesCorr['CorrPearson'] == 0].count()
```

[316]:
```
#Contar pares con correlación positiva
#AA1920 235
#AA2021 1248
dfParesCorr[dfParesCorr['CorrPearson'] > 0].count()
```

[317]:
```
#Contar pares con correlación negativa
#AA1920: 90
#AA2021: 237
dfParesCorr[dfParesCorr['CorrPearson'] < 0].count()
```

[318]:
```
#Obtener valores mínimo y màximo de correlaciòn negativa
#AA1920 cuando es negativa: -0.227 hasta -0.004
dfParesCorr[dfParesCorr['CorrPearson'] < 0]['CorrPearson'].aggregate(['min',␣
 ↪'max'])
```

[319]:
```
#Obtener valores mínimo y màximo de correlaciòn positiva
#AA1920 cuando es negativa: 0.014 hasta 0.963
```

```
dfParesCorr[dfParesCorr['CorrPearson'] > 0]['CorrPearson'].aggregate(['min',␣
↪'max'])
```

Verificar si existe correlación negativa

[320]:
```
#Verificar si existe correlación negativa
#Si consideramos -0.7 como un indicativo de que existe una correlaciòn muy alta.
#Resultado obtenido: se aprecia que no existen valores de correlación negativos␣
↪altos.
dfParesCorr[dfParesCorr['CorrPearson'] < 0].sort_values(by='CorrPearson',␣
↪ascending=True).head(7)
```

Verificar si existe correlación positiva

[321]:
```
#Verificar si existen correlaciones positivas:
#Resultado obtenido: se aprecian estudiantes que tienen una correlación mayor a␣
↪0.7
#Estos estudiantes tienen una correlación positiva: 30 interacciones entre␣
↪estudiantes
dfParesCorr[dfParesCorr['CorrPearson'] >= 0.7 ].sort_values(by='CorrPearson',␣
↪ascending=False).head(7)
```

[322]:
```
dfParesCorr[dfParesCorr['CorrPearson'] >= 0.7 ].count()
```

II.2 Matriz de correlación de Pearson para matriz de adyacencia variable caracteres-mensaje Crear una matriz para calcular la correlaciòn de Pearson matrizPromCaracteresMensaje es la matriz de adyacencia con el valor promedio de caracteres en los mensajes intercambiados por el origen y el destino

[323]:
```
mProCarMsjCorrPea = crearMatriz(listaNodos)
mProCarMsjCorrPea = calcularCorrPearson(matrizPromCaracteresMensaje,␣
↪mProCarMsjCorrPea, NUM_NODOS)
mProCarMsjCorrPea = cambiarNanPorCeros(mProCarMsjCorrPea)
mProCarMsjCorrPea.head()
```

[324]:
```
plt.subplots(figsize=(30,25))
sn.heatmap(mProCarMsjCorrPea, annot=True)
plt.show()
```

[325]:
```
str(mProCarMsjCorrPea.mean().mean())
```

[326]:
```
dfParesCarMsj = obtenerDfParesCorrelacionados(mProCarMsjCorrPea)
dfParesCarMsj.head()
```

[348]:
```
#AA1920 dfParesCarMsj: 0 cero; mayor que cero 169; menor que cero: 156
#AA2021: igual a cero: 0; mayor que cero: 1247 ;menor que cero 238
#AA2122: igual a cero: 0; mayor que cero: 857 ; menor que cero: 113

#MDMS1920: igual a cero: 0; mayor que cero: 40; menor que cero: 5
#MDMS021: igual a cero: 0 ; mayor que cero: 263 ; menor que cero: 115
#MDMS2122: igual a cero: 0; mayor que cero: 174; menor que cero: 57
dfParesCarMsj[dfParesCarMsj['CorrPearson'] == 0].count()
```

[349]:
```
#AA1920 dfParesCarMsj: positivo 0,0001 hasta 0.99; negativo -0.377 hasta -0.003
#AA2021 positivo: 0.009 hasta 1 ; negativo: -0.072 hasta -0.0009
#AA2122 0.001 hasta 1 ; -0.099 hasta -0.006

#MDMS1920 positivo: 0.07 hasta 0.99; negativo: -0.180 hasta -0.018
#MDMS2021 0.002 hasta 1 ; -0.118 hasta -0.001
#MSMS2122 0.068 hasta 1; -0.136 hasta -0.016
dfParesCarMsj[dfParesCarMsj['CorrPearson'] > 0]['CorrPearson'].
 ↪aggregate(['min', 'max'])
```

[329]:
```
#Verificar correlaciones negativas
dfParesCarMsj[dfParesCarMsj['CorrPearson'] < 0].sort_values(by='CorrPearson',␣
 ↪ascending=True).head(7)
```

[330]:
```
#Verificar correlaciones positivas
dfParesCarMsj[dfParesCarMsj['CorrPearson'] >= 0.7 ].
 ↪sort_values(by='CorrPearson', ascending=False).head(7)
```

II.3 Matriz de correlación de Pearson para matriz de adyacencia variable probabilidadafectiva

[331]:
```
mProProAfeCorrPea = crearMatriz(listaNodos)
mProProAfeCorrPea = calcularCorrPearson(matrizPromProbAfectiva,␣
 ↪mProProAfeCorrPea, NUM_NODOS)
mProProAfeCorrPea = cambiarNanPorCeros(mProProAfeCorrPea)
mProCarMsjCorrPea.head()
```

El siguiente gráfico permite apreciar que existen estudiantes que se correlacionan positivamente en cuanto a la probabilidad afectiva promedio que demostraron en los mensajes intercambiados.

[332]:
```
plt.subplots(figsize=(30,25))
sn.heatmap(mProProAfeCorrPea, annot=True)
plt.show()
```

[333]:
```
str(mProProAfeCorrPea.mean().mean())
```

[334]:
```
dfParesProAfe = obtenerDfParesCorrelacionados(mProProAfeCorrPea)
dfParesProAfe.head(3)
```

[351]:
```
#contar los pares de positivos, negativos y cero
#AA2122: igual a cero: 665; positivos: 58 ; negativos: 267

#MDMS1920: igual a cero: 0; positivos: 18; negativos: 27
#MDMS2021: igual a cero: 78; positivos: 241; negativos:59;
#MDMS2122: igual a cero: 95; positivos: 88; negativos: 48
dfParesProAfe[dfParesProAfe['CorrPearson'] == 0].count()
```

[352]:
```
#ver valores mìnimo y màximo de correlaciòn
#AA1920 0.001 hasta 1; -0.099 hasta - 0.001
#AA2021 0.017 hasta 1 ; -0.044 hasta -0.003
#AA2122 0.002 hasta 1 ; -0.074 hasta -0.008

#MSMS1920 0.033 hasta 0.99 ; -0.236 hasta -0.049
#MDMS2021 0.006 hasta 1  ; -0.059 hasta -0.007
#MDMS2122 0.005 hasta 1; -0.068 hasta -0.008
dfParesProAfe[dfParesProAfe['CorrPearson'] > 0]['CorrPearson'].
 ↪aggregate(['min', 'max'])
```

[337]:
```
#Verificar correlaciones negativas
dfParesProAfe[dfParesProAfe['CorrPearson'] < 0].sort_values(by='CorrPearson',␣
 ↪ascending=True).head(7)
```

Verificar correlaciones positivas

Resultado obtenido: considerando el promedio de la probabilidad afectiva de los mensajes que intercambaron los alumnos encontramos que hay una correlación perfecta entre un par Estu-26 y Estu-103.

[338]:
```
#Verificar correlaciones negativas
dfParesProAfe[dfParesProAfe['CorrPearson'] >= 0.7 ].
 ↪sort_values(by='CorrPearson', ascending=False).head(7)
```

II.4 Matriz de correlación de Pearson para matriz de adyacencia variable respuesta-horas (el tiempo que demoró en contestar al mensaje)

[339]:
```
mProResHorCorrPea = crearMatriz(listaNodos)
mProResHorCorrPea = calcularCorrPearson(matrizPromTiempoRespMsj,␣
 ↪mProResHorCorrPea, NUM_NODOS)
mProResHorCorrPea = cambiarNanPorCeros(mProResHorCorrPea)
mProResHorCorrPea.head(3)
```

[340]:
```
plt.subplots(figsize=(30,25))
sn.heatmap(mProResHorCorrPea, annot=True)
plt.show()
```

[341]:
```
str(mProResHorCorrPea.mean().mean())
```

[342]:
```
dfParesResHor = obtenerDfParesCorrelacionados(mProResHorCorrPea)
dfParesResHor.head(3)
```

[354]:
```
#AA1920: igual a cero: 26; positivos 184; negativos: 141
#AA2021: cero: 0; positivos: 1136 ; negativos: 349
#AA2122: cero: 0; positivos: 872; negativos: 118

#MDMS1920: cero: 0 ; positivos: 42 ; negativos: 3
#MDMS2021: cero: 0; positivos: 323; negativos: 55;
#MDMS2122: cero: 0; positivos: 167 ; negativos: 64
dfParesResHor[dfParesResHor['CorrPearson'] > 0].count()
```

[355]:
```
##AA1920: 0.0003 hasta 0.999; -0.165 hasta 0.002
#AA2021: positivo: 0.001 hasta 1; negativo: -0.082 hasta -0.004
#AA2122: 0.0001 hasta 1 ; -0.088 hasta -0.004

#MDMS1920: 0.096 hasta 0.998; -0.193 hasta -0.009
#MDMS2021: 0.001 hasta 1; -0.115 hasta -0.0007
#MDMS2122: 0.008 hasta 1; -0.095 hasta -0.010
dfParesResHor[dfParesResHor['CorrPearson'] > 0]['CorrPearson'].
 ↪aggregate(['min', 'max'])
```

[345]:
```
#Verificar correlaciones negativas
dfParesResHor[dfParesResHor['CorrPearson'] < 0].sort_values(by='CorrPearson',␣
 ↪ascending=True).head(7)
```

[346]:
```
#Verificar correlaciones negativas
dfParesResHor[dfParesResHor['CorrPearson'] >= 0.7 ].
 ↪sort_values(by='CorrPearson', ascending=False).head(7)
```

4.1_clustering_final_comunicacion_v5

February 16, 2024

0.0.1 Clustering de estudiantes usando como criterio de comunicación tres métricas de redes sociales

De los análisis efectuados previamente se aprecia que los atributos que pueden ser usados para medir la comunicación, son aquellos que el Coordinador o Profesor "modelan" en su interacción con los estudiantes. Se espera que en cada grupo, el coordinador se constituya en un facilitador "prototipo" para la comunicación: intercambiar ideas, consultas, diferencias en el grupo.

El análisis efectuado indica que tres métricas de redes sociales pueden servir para medir la comunicación: Grado (degree), centralidad de intermediación (Betweeness Centrality) y Centralidad de cercanía (Closness Centrality).

En este cuaderno, se hace un clustering para obtener grupos de estudiantes, considerando únicamente estos tres atributos para representar la comunicación. En este clustering se excluye al profesor.

En este notebook se seguirán los siguientes pasos:

Paso 1. Leer los estudiantes con su métricas de redes sociales.

Paso 2. Escalar los datos.

Paso 3. Realizar clustering.

Paso 4. Guardar los datos.

0.0.2 Paso 1. Leer los estudiantes con su métricas de redes sociales.

[659]:
```
#Importar librerías
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from kneed import KneeLocator
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
```

[660]:
```
import numpy as np
```

[661]:
```
#usar las funciones para clustering
%run funciones_clustering.ipynb
```

[662]:
```
#Leer el archivo de nodos con sus atributos de redes sociales
dfNodos = pd.read_csv("dataset_ultimo/matrices_adyacencia_var_msj/
 ↪nodos_MDMS_2122.csv")
#dfNodos = pd.read_csv("dataset_ultimo/matrices_adyacencia_var_msj/
 ↪nodos_AA_2122.csv")
nombreArchivo = 22122

dfNodos.head(3)
```

[663]:
```
#Obtener un dataframe con las medidas de centralidad que nos interesan
dfRedes = dfNodos[['Label', 'Degree', 'closnesscentrality',␣
 ↪'betweenesscentrality']].copy()
dfRedes.rename(columns = {'Label': 'estudiante'}, inplace = True)
dfRedes.head(3)
```

[664]:
```
#Antes, verificar el número de personas en el dataframe
len(dfRedes)
```

[665]:
```
#buscar el coordinador o profesor: Coor-36 o Prof-4
#solo en AA1920 tenemos Coor-36; el resto de dataframes tiene Prof-4
indiceCoordinador = dfRedes[(dfRedes['estudiante'] =='Coor-36') &␣
 ↪(nombreArchivo==11920)].index
#indiceProfesor = dfRedes[(dfRedes['estudiante'].str.contains('Prof-4',␣
 ↪na=False)) & (nombreArchivo != 11920)].index
indiceProfesor = dfRedes[(dfRedes['estudiante'] == 'Prof-4') & (nombreArchivo !
 ↪= 11920)].index

print('Indice coor',indiceCoordinador, len(indiceCoordinador))
print('Indice profe',indiceProfesor, len(indiceProfesor))
```

Buscar el coordinador que se debe eliminar

[666]:
```
#Excluir al profesor o coordinador
if len(indiceCoordinador) > 0:
    dfRedes.drop(index=[indiceCoordinador[0]], inplace=True, axis=0)

if len(indiceProfesor) > 0:
    dfRedes.drop(index=[indiceProfesor[0]], inplace=True, axis=0)
```

[667]:
```
dfRedes[(dfRedes['estudiante']=='Prof-4') | (dfRedes['estudiante']=='Coor-36')]
```

[668]:
```
#tenemos un dataset solo con estudiantes, se han retirado los coordinadores
dfRedes.head(3)
```

[669]:
```
#Después: Verificar el número de estudiantes en el dataframe
#num_estudiantes_final
```

```
len(dfRedes)
```

[670]:
```
#dfRedes.sort_values(by='Degree')
dfRedes= dfRedes.reset_index(drop=True)
dfRedes.head(3)
```

0.0.3 Paso 2. Escalar los datos.

[671]:
```
#Retirar la columna estudiante, cuyo seudónimo no es útil para el clustering
#Hasta aquí: dfKmeans es igual a dfRedes
dfKmeans = dfRedes.drop(columns=['estudiante'])
dfKmeans.head(3)
```

[672]:
```
dfKmeans.sort_values(by='Degree').head(3)
```

[673]:
```
dfKmeans.loc[[0]]
```

[674]:
```
dfKmeans.head(3)
```

[675]:
```
dfKmeans.index.values
```

[676]:
```
#Para obtener la fila asociada a la etiqueta del índice puedo usar: 2␣
 ↪alternativas:
# 1)buscar por la etiqueta del índice;
# 2) buscar por el índice como tal (empieza desde cero y con orden secuencial)
# loc[[2]] que es igual a loc[2].  Esto no es igual a iloc[2], que indexa desde␣
 ↪cero y devuelve
#dfKmeans.loc[[indice]]
print('Indice original coordinador: ', indiceCoordinador)
print('Indice original profesor', indiceProfesor)
print('Contenido indice original coordinador: ', dfKmeans.
 ↪iloc[indiceCoordinador])
print('Contenido indice original profesor', dfKmeans.iloc[indiceProfesor])
```

[677]:
```
#dfKmeans va a ser escalado porque ahora solo tiene nùmeros
scaler = StandardScaler()
scaler.fit(dfKmeans)
dataEscalada=scaler.transform(dfKmeans)
```

[678]:
```
dfEscalado=pd.DataFrame(dataEscalada, columns=['Degree', 'closnesscentrality',␣
 ↪'betweenesscentrality'])
dfEscalado.describe()
```

[679]:
```
#dfEscalado.sort_values(by='Degree')
```

[680]:
```
#sin escalar: descomentar
#con escalamiento: comentar
#dataEscalada = dfKmeans
```

0.0.4 Paso 3. Realizar clustering

[681]:
```
#iniciar el modelo Kmeans
kmeansArgs = {"init": "random", "n_init": 10, "max_iter": 300, "random_state":
 ↪42}
```

[682]:
```
#Para el dataset MDMS1920 solo tenemos 10 elementos, retirando el profesor␣
 ↪quedan 9
#Por ello hay que colocar màximo 9 elementos
MAXIMOK = 12
inercias, ks = encontrarMejoresClusters(dataEscalada, MAXIMOK)
clusterCodo=encontrarClustersCodo(inercias,MAXIMOK)
print("Número de cluster obtenido por el gráfico del codo: ", clusterCodo)

NUM_CLUSTERES = clusterCodo
```

[683]:
```
generarPlotCodo(inercias, ks)
```

[684]:
```
coeficientesSilhouette=calcularIndiceSilhouette(dataEscalada, MAXIMOK)
coeficientesSilhouette
```

[685]:
```
graficarScoresSilhouette(coeficientesSilhouette, MAXIMOK)
```

[686]:
```
#Aplicar el modelo con el mejor número de clusters encontrado
kmeans = KMeans(n_clusters=NUM_CLUSTERES, **kmeansArgs)
kmeans.fit(dataEscalada)
```

[687]:
```
#medir la inercia: cada vez el valor de la inercia va disminuyendo
#AA1920: con 4: 14.91;  con 5: 10.81
kmeans.inertia_
```

[688]:
```
#añadir las etiquetas considerando 4 grupos
dfKmeans["clusters"] = kmeans.labels_
```

[689]:
```
len(dfKmeans)
```

[690]:
```
dfKmeans.head()
```

[691]:
```
dfRedes["clusters"] = kmeans.labels_
dfRedes.head()
```

0.0.5 Paso 4. Guardar los datos

Guardar los estudiantes con sus clústeres, materia y año

[693]:
```
#añadir el curso al dataframe
dfEscalado['curso']= nombreArchivo
#dfMsjProAfe[['source', 'target', 'resHorasEscala','resProAfeEscala',␣
 ↪'clusters','curso']].header()

#abrir un archivo
fileEst = open('todos_estudiantes.csv','a')
#guardar el dataframe en el archivo abierto
dfEscalado.to_csv(fileEst, index=False, header=False)
fileEst.close()
```

Obtener los centroides

[694]:
```
#Los centroides son coordenadas (Degree, closnesscentrality,␣
 ↪betweenesscentrality) (datos escalados)
centroides = kmeans.cluster_centers_
centroides
```

[695]:
```
#guardar los centroides en un archivo
dfCentroides = pd.DataFrame(centroides)

dfCentroides['curso'] = nombreArchivo
dfCentroides['cluster'] = dfCentroides.index

arrayCentroides = dfCentroides.to_numpy()
print(arrayCentroides)

fcoh = open('centroides_comunicacion.csv','a')
np.savetxt(fcoh, arrayCentroides, delimiter = ",")
fcoh.close()
```

[696]:
```
#Verificar cuántos estudiantes tenemos en cada cluster
#En AA1920 hay un caso especial con el estudiante Estu-92 que no envió ningùn␣
 ↪mensaje, pero sì recibiò
dfKmeans.groupby('clusters')['clusters'].count()
```

[697]:
```
#En AA1920 aparece el cluster 1 con 1, porque Estu-92 es un caso especial
#En AA1920 el clustering distinguiò los grupos, por la mayor presencia de los␣
 ↪atributos (valores  mayores)
#grupo 0: màs alto grado; grupo 3 mediano grado; grupo 2: menor grado;
#grupo 1: podrìa anexar al grupo 2
dfKmeans[dfKmeans['clusters']== 3]
```

[698]:
```
#Buscar de acuerdo a la "etiqueta" del índice (distinto es buscar por el␣
↪índice, asociado al orden de las filas)
dfKmeans.loc[6]
```

[700]:
```
#observar a qué estudiante pertenece
dfNodos.iloc[6]
```

Analizando todos los datasets se tiene que el coordinador o profesor:

El coordinador tiene generalmente el grado más alto.

El coordinador tiene un valor de centralidad de cercanía superior al 0.5 y menor que el 0.9

El coordinador tiene un valor de centralidad de intermediación entre 0.31 y 0.48

[701]:
```
#dfKmeans.groupby('clusters')['clusters'].aggregate( {'Degree': ['mean',␣
↪'std'], 'closnesscentrality': ['mean', 'std'], 'betweenesscentrality':␣
↪['mean', 'std']})
dfKmeans.groupby('clusters').aggregate({'Degree' : ['count', 'min', 'max',␣
↪'mean', 'std'],
                                        'closnesscentrality': ['count', 'min',␣
↪'max', 'mean', 'std'],
                                       'betweenesscentrality': ['count', 'min',␣
↪'max', 'mean', 'std']}).reset_index()
```

4.2_clustering_final_cohesion_v6

February 16, 2024

0.0.1 Clustering de mensajes, considerando la probabilidad afectiva y la rapidez de contestación como criterio de cohesión

El objetivo de este notebook es realizar dos actividades principales:

I) un clustering de los mensajes que intercambiaron los estudiantes. Se obtienen grupos de mensajes y cada mensaje está asociado a un par de estudiantes origen y destino. Los estudiantes pueden repetirse en varios grupos de mensajes, por lo cual el siguiente paso será identificar en qué grupos un estudiante fue asignado. Esto indica en qué grupos un estudiante podría participar de acuerdo a la probabilidad afectiva.

II) En segundo lugar se elabora una matriz de adyacencia teniendo como filas y columnas a los estudiantes. A continuación se crea una matriz para calcular el coeficiente de Pearson, para evaluar la fuerza de la relación entre cada par de estudiantes.
Pasos a desarrollar en este notebook:
Paso 1. Leer los datos de mensajes
Paso 2. Hacer un escalamiento de las variables probabilidad afectiva y respuesta horas que representan la cohesión.
Paso 3. Realizar el clustering (se guardan los datos del clustering)
Paso 4. Construir la matriz de adyacencia
Paso 5. Construir la matriz de correlación de Pearson.

0.0.2 Primera parte: realizar el clustering con los atributos probabilidad afectiva y rapidez de contestación

Paso 1. Leer los datos de mensajes

[550]:
```
#importar librerías
import pandas as pd
from sklearn.cluster import KMeans
from kneed import KneeLocator
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from scipy import stats
import seaborn as sns
import numpy as np
```

[551]:
```
from pathlib import Path
```

[552]:
```
#usar las funciones para clustering
%run funciones_clustering.ipynb
```

[553]:
```
#archivoEntrada='dataset_ultimo/salida_revision_datasets/
 ↪MDMS-foro-21-22-anon_MK.csv'
archivoEntrada='dataset_ultimo/salida_revision_datasets/AA-I-foro-21-22-anon_MK.
 ↪csv'

nombreArchivo = 12122

dfMsj = pd.read_csv(archivoEntrada, sep=',')
dfMsj.head(3)
```

[554]:
```
len(dfMsj)
```

[555]:
```
#Casos particulares: Coor-36; Estu-113
dfMsj[dfMsj['source']=='Estu-21']
```

[556]:
```
dfMsjProAfe = dfMsj[['source', 'target', 'probabilidadAfectiva',␣
 ↪'respuestaHoras']].copy()
dfMsjProAfe.head(3)
```

Paso 2: Escalar variables Escalar variable Respuesta Horas

Transformar la respuesta en horas a una escala especìfica entre 0 y 1: el menor valor de horas será 1; el mayor valor de horas será cero. En otras palabras: se asigna un valor mayor (1) a quien contestó más rápido; y un valor cero (0) a quien demoró más en contestar.

[557]:
```
minHoras=dfMsjProAfe['respuestaHoras'].min()
maxHoras=dfMsjProAfe['respuestaHoras'].max()
denominador = minHoras - maxHoras
#print('minimo es: ', minHoras, 'maximo es: ', maxHoras, 'denominador es:',␣
 ↪denominador)
#aplicar a la columna del dataframe el valor:
dfMsjProAfe['resHorasEscala'] = dfMsjProAfe['respuestaHoras'].apply(lambda x:␣
 ↪((x-maxHoras)/denominador))
#cambiar un valor negativo cero por cero
dfMsjProAfe['resHorasEscala'] = dfMsjProAfe['resHorasEscala'].replace(-0, 0)
```

[558]:
```
dfMsjProAfe[dfMsjProAfe['resHorasEscala']==1]
```

[559]:
```
dfMsjProAfe[dfMsjProAfe['resHorasEscala']==0]
```

Escalar variable Probabilidad Afectiva

Se usa una escala entre 0 y 1. Se asigna un valor mayor (1) a quien demuestra un valor más grande de probabilidad afectiva; y se asigna un valor menor (0) a quien demuestra un valor más pequeño de probabilidad afectiva.

[560]:
```
minProAfe=dfMsjProAfe['probabilidadAfectiva'].min()
maxProAfe=dfMsjProAfe['probabilidadAfectiva'].max()
denominador = maxProAfe - minProAfe
dfMsjProAfe['resProAfeEscala'] = dfMsjProAfe['probabilidadAfectiva'].
↪apply(lambda x: ((x-minProAfe)/denominador))
```

[561]:
```
dfMsjProAfe.describe()
```

[562]:
```
dfMsjProAfe[dfMsjProAfe['resProAfeEscala']==1]
```

[563]:
```
dfMsjProAfe[dfMsjProAfe['resProAfeEscala']==0]
```

[564]:
```
#dataset para hacer clustering con el atributo que representa la cohesión
dfMsjKmeans = dfMsjProAfe[['resProAfeEscala', 'resHorasEscala']].copy()
dfMsjKmeans.head(3)
```

[565]:
```
dfMsjKmeans.describe()
```

Dataset para trabajar con el clustering

[566]:
```
#probar con escala y sin escala: se decide no escalar, pues se hizo un␣
↪escalamiento previo ya
dataEscalada = dfMsjKmeans
```

[567]:
```
dataEscalada
```

Paso 3. Realizar el clustering

[568]:
```
kmeansArgs = {"init": "random", "n_init": 10, "max_iter": 300, "random_state":
↪42}
```

[569]:
```
MAXIMOK = 12
inercias, ks = encontrarMejoresClusters(dataEscalada, MAXIMOK)
clusterCodo=encontrarClustersCodo(inercias,MAXIMOK)
print("Número de cluster obtenido por el gráfico del codo: ", clusterCodo)

#la variable NUM_CLUSTERES contiene el mejor número de clústeres encontrado
NUM_CLUSTERES = clusterCodo
```

[570]:
```
generarPlotCodo(inercias, ks)
```

[571]:
```
coeficientesSilhouette=calcularIndiceSilhouette(dataEscalada, 12)
coeficientesSilhouette
```

[572]:
```
graficarScoresSilhouette(coeficientesSilhouette, 12)
```

[573]:
```
#Aplicar el modelo con el mejor número de clusters encontrado
kmeans = KMeans(n_clusters=NUM_CLUSTERES, **kmeansArgs)
kmeans.fit(dataEscalada)
```

[574]:
```
#1920 3grupos: 145.688 escalamiento
kmeans.inertia_
```

Obtener las etiquetas

[575]:
```
#añadir las etiquetas considerando 3 grupos
dfMsjKmeans["clusters"] = kmeans.labels_
dfMsjKmeans.head()
```

[576]:
```
#Verificar cuántos estudiantes tenemos en cada cluster
#Clùsteres con la influencia del coordinador
dfMsjKmeans.groupby('clusters')['clusters'].count()
```

[577]:
```
dfMsjKmeans.head(3)
```

[578]:
```
#Al dataset original le añadimos los clusteres
dfMsjProAfe["clusters"] = kmeans.labels_
dfMsjProAfe.head()
```

[579]:
```
len(dfMsjProAfe)
```

[580]:
```
dfMsjProAfe.describe()
```

Guardar los mensajes con sus clústeres, materia y año

[581]:
```
#añadir el curso al dataframe
dfMsjProAfe['curso']= nombreArchivo
#dfMsjProAfe[['source', 'target', 'resHorasEscala','resProAfeEscala',␣
 ↪'clusters','curso']].header()

#abrir un archivo
fileMsj = open('todos_mensajes.csv','a')
#guardar el dataframe en el archivo abierto
dfMsjProAfe[['source', 'target', 'resHorasEscala','resProAfeEscala',␣
 ↪'clusters','curso']].to_csv(fileMsj, index=False, header=False)
fileMsj.close()
```

Obtener los centroides

[582]:
```
#Los centroides son coordenas (ProbabilidadAfectiva, HorasEscala) (datos␣
 ↪escalados)
```

```
centroides = kmeans.cluster_centers_
centroides
```

[583]:
```
#guardar los centroides en un archivo
dfCentroides = pd.DataFrame(centroides)
dfCentroides
```

[584]:
```
dfCentroides['curso'] = nombreArchivo
dfCentroides
```

[585]:
```
dfCentroides['cluster'] = dfCentroides.index
dfCentroides
```

[586]:
```
arrayCentroides = dfCentroides.to_numpy()
print(arrayCentroides)

fcoh = open('centroides_cohesion.csv','a')
np.savetxt(fcoh, arrayCentroides, delimiter = ",")
fcoh.close()
```

[587]:
```
#dibujar los clusters y sus centroides
kmeans.labels_
```

[588]:
```
plt.figure(figsize=(4,3))
plt.scatter(dfMsjKmeans['resProAfeEscala'], dfMsjKmeans['resHorasEscala'], c=␣
 ↪dfMsjKmeans['clusters'],
            s=50, cmap='viridis')
plt.scatter(centroides[:, 0], centroides[:, 1], c='black', s=200, alpha=0.5)

#colocar y configurar los titulos de los ejes
font2 = {'family':'serif','color':'black','size':10}
plt.xlabel("Probabilidad Afectiva (escalada)", fontdict = font2)
plt.ylabel("Interés en Responder (escalada)", fontdict = font2)
```

[589]:
```
#DIbujar puntos con cluster 1 y centroide 1
plt.figure(figsize=(4,3))
centroides

dfC1 = dfMsjKmeans[dfMsjKmeans['clusters'] ==1]

plt.scatter(dfC1['resProAfeEscala'], dfC1['resHorasEscala'], c=␣
 ↪dfC1['clusters'],
            s=50, cmap='viridis')
plt.scatter(centroides[1, 0], centroides[1, 1], c='black', s=200, alpha=0.5)
```

0.0.3 Segunda parte: crear la matriz de adyacencia (filas: alumnos; columnas: estudiantes)

Paso 4 Construir la matriz de adyacencia

[590]:
```
#Cargar los nodos
#Leer el archivo de nodos con sus atributos de redes sociales
dfNodos = pd.read_csv("dataset_ultimo/matrices_adyacencia_var_msj/nodos_AA_2122.
 ↪csv")
#dfNodos = pd.read_csv("dataset_ultimo/matrices_adyacencia_var_msj/
 ↪nodos_MDMS_2122.csv")
dfNodos.head(3)
```

[591]:
```
#Obtener la lista de nodos que serán las filas
listaNodos = dfNodos['Label'].tolist()
NUM_NODOS=len(listaNodos)
NUM_NODOS
```

[592]:
```
#Obtener los clústeres que serán las columnas
listaClusteres = []

for i in range(0, NUM_CLUSTERES):
    listaClusteres.append("Cluster_" + str(i))

listaClusteres
```

[593]:
```
#Crear una funcion para generar las filas de la matriz y las columnas
def crearMatrizDadasFilCol(lFilas, lColumnas):
    matriz=pd.DataFrame(0, index=lFilas, columns= lColumnas)
    return matriz

matrizAdy=crearMatrizDadasFilCol(listaNodos, listaClusteres)
matrizAdy.head(3)
```

[594]:
```
matrizAdy.shape
```

[595]:
```
#probar la consulta que obtiene los clùsteres para los estudiantes
dicClusters=dict(dfMsjProAfe[dfMsjProAfe['source'] == 'Estu-115'].
 ↪groupby(['clusters'])['clusters'].count())
#len(dicClusters)
dicClusters
```

[596]:
```
#crear una funcion que entregue los clùsteres en los que està cada estudiante
def getClustersEstudiante(dfClusters, estudiante, numClusters):
    #iniciar la lista a ceros con el número de clusteres
    listaClusters = [0 for i in range(0, numClusters)]
```

```
    #obtener los clústeres del sujeto
    dictClusters=dict(dfClusters[dfClusters['source'] == estudiante].
↪groupby(['clusters'])['clusters'].count())
    #print('hola')
    #rint(dictClusters)
    #print(len(dictClusters))
    #validar el número de clústeres
    if ( (len(dictClusters) > 0) and (len(dictClusters) != numClusters)):
        #completar los valores
        for k in dictClusters.keys():
            #print(dictClusters[k])
            listaClusters[k] = 1 #dictClusters[k]
    elif len(dictClusters) == 3:
        listaClusters = [1,1,1]

    #print('final')
    #print(listaClusters)
    return listaClusters

getClustersEstudiante(dfMsjProAfe, 'Estu-115', NUM_CLUSTERES)
```

[597]:
```
#crear una funcion que complete las filas de la matriz
#Función para llenar la matriz de estudiantes

def llenarMatrizFilColAdy(pMatrizLlenar, pListaNodos, dfMensajes):
    for nodo in pListaNodos:
        datos = getClustersEstudiante(dfMensajes, nodo, NUM_CLUSTERES)
        #print("nodo es: " , nodo , " -> " , datos)
        pMatrizLlenar.loc[nodo] = datos

    return pMatrizLlenar
```

[598]:
```
#Rellenar la matriz con la presencia o ausencia de los estudiantes en el cluster
matrizAdy = llenarMatrizFilColAdy(matrizAdy, listaNodos, dfMsjProAfe)
matrizAdy.head(3)
```

[599]:
```
matrizAdy.shape
```

Paso 5. Construir la matriz de correlación de Pearson.

[600]:
```
#crear una matriz con ceros para guardar las correlaciones de Pearson
matrizCorrPearson=crearMatrizDadasFilCol(listaNodos, listaNodos)
matrizCorrPearson.head(3)
```

[601]:
```
matrizCorrPearson.shape
```

[602]:
```
#funcion para calcular el coeficiente de Pearson
def calcularCorrPearson(pMatrizAdy, pMatrizCorr, numNodos, NUM_CLUSTERES):

    for i in listaNodos:
        for j in listaNodos:
            #results = stats.pearsonr(pMatrizAdy.iloc[i],pMatrizAdy.iloc[j])
            results = stats.pearsonr(pMatrizAdy.loc[i],pMatrizAdy.loc[j])
            pMatrizCorr.at[i,j] = results[0]
            #print("i-j: ", i,' - ' ,j)
            #print(pMatrizAdy.loc[i])
            #print(pMatrizAdy.loc[j])

    #devolver la matriz con los coeficientes de Pearson
    return pMatrizCorr

#prueba de la función
matrizCorrPearson=calcularCorrPearson(matrizAdy, matrizCorrPearson, NUM_NODOS,␣
 ↪NUM_NODOS)
```

[603]:
```
#podemos escoger los que están mejor correlacionados (1), y los menos␣
 ↪correlacionados(0.5)
#los que tienen correlación negativa, no se los podría considerar porque tienen␣
 ↪diferencias
matrizCorrPearson.fillna(0, inplace=True)
```

[604]:
```
matrizCorrPearson
```

Guardar la matriz de correlaciòn de Pearson para la cohesión

[605]:
```
#guardar la matriz de Pearson
matrizCorrPearson.to_csv('dataset_ultimo/matrices_coefPearson/
 ↪matrizCPearson_AA2122.csv', sep=",")
#matrizCorrPearson.to_csv('dataset_ultimo/matrices_coefPearson/
 ↪matrizCPearson_MDMS2122.csv', sep=",")
```

Matriz de correlación de Pearson

Se aprecia que existen relaciones entre cada par de estudiantes analizados.

La matriz muestra tanto relaciones fuertes positivas (1 rojo intenso) y pocas relaciones negativas (-1 color azul eléctrico).

También se aprecia relaciones cuyo valor de coefiente constra entre: 1 y -1.

Existen estudiantes que demuestran una ausencia de relación con todos los estudiantes (color beige).

[606]:
```
sns.heatmap(matrizCorrPearson, xticklabels=True, yticklabels=True, cmap=␣
 ↪'coolwarm')
```

[607]:
```
matrizCorrPearson.shape
```

[608]:
```
matrizCorrPearson.min().min()
```

[609]:
```
matrizCorrPearson.max().max()
```

[610]:
```
#Obtener estadísticas de la matriz de correlación
matrizCorrPearson.describe()
```

5_analisis_centroides_v4

February 16, 2024

0.0.1 Análisis de los centroides

En este notebook se analizarán los centroides obtenidos al hacer el clustering con los atributos de comunicación y cohesión.

El objetivo es revisar las similitudes o diferencias de los centroides para estimar posibles rangos para los atributos y con estos umbrales ofrecer opciones para la formación de grupos.

Por cada materia y año se realizó el clustering, de modo que existen 3 (o 4) centroides por materia y por año.

En los ejercicios de clustering se verificó que los datasets arrojaron de forma común 3 grupos para cada materia; con alguna excepción que generó 4 grupos.

Los datos de los centroides fueron guardados en un archivo de texto, cuando se realizó la ejecución respectiva.

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

0.0.2 1. Graficar los centroides de cohesión

[3]:
```
#df = pd.read_csv('centroides_cohesion.csv', header=None, names=['proAfeEsc',␣
↪'resHorasEsc', 'curso', 'cluster'])
df = pd.read_csv('centroides_cohesion.csv')
```

[4]:
```
df.head()
```

Extraer los datos para cada año de las materias AA y MDMS

[5]:
```
dfAA1920 = df[df['curso'] == 11920]
dfAA2021 = df[df['curso'] == 12021]
dfAA2122 = df[df['curso'] == 12122]
dfMD1920 = df[df['curso'] == 21920]
dfMD2021 = df[df['curso'] == 22021]
dfMD2122 = df[df['curso'] == 22122]
```

Graficar los centroides de las materias

[6]:
```
plt.figure(figsize=(5,4))
plt.scatter(dfAA1920['proAfeEsc'], dfAA1920['resHorasEsc'], alpha=0.7, s=90,␣
 ↪label='AA1920', marker='*')
plt.scatter(dfAA2021['proAfeEsc'], dfAA2021['resHorasEsc'], alpha=0.9, s=90,␣
 ↪label='AA2021', marker='P')
plt.scatter(dfAA2122['proAfeEsc'], dfAA2122['resHorasEsc'], alpha=0.5, s=90,␣
 ↪label='AA2122', marker='o')

plt.scatter(dfMD1920['proAfeEsc'], dfMD1920['resHorasEsc'], alpha=0.5, s=90,␣
 ↪label='MD1920', marker='<')
plt.scatter(dfMD2021['proAfeEsc'], dfMD2021['resHorasEsc'], alpha=0.5, s=85,␣
 ↪label='MD2021', marker='>')
plt.scatter(dfMD2122['proAfeEsc'], dfMD2122['resHorasEsc'], alpha=0.8, s=80,␣
 ↪label='MD2122', marker='s')

plt.xlabel('Probabilidad Afectiva')
plt.ylabel('Interés en responder')

plt.title('Cohesión: Centroides para clústeres en materias AA y MDMS')
plt.legend()
```

De la figura anterior se puede apreciar que los centroides de los clusters tienen coordenadas similares: se han concentrado en tres áreas.

1. En la esquina superior izquierda se aprecia la mayor coincidencia de los centroides, prácticamente se sobreponen unos a otros sobre la misma área. Esta área se puede denominar: baja probabilidad afectiva y contestación rápida. Los estudiantes que se relacionan poco en sus mensajes, pero contestan rápidamente.
2. En la esquina superior derecha se aprecia que existen coincidencias cercanas entre todas las materias y todos los años. Esta área tiene alta probabilidad afectiva y alta rapidez al contestar. Los estudiantes que mejor se relacionan a través de sus mensajes y contestan rápidamente.
3. En la esquina inferior izquierda se aprecia que existe poca cercanía entre los centroides, anque dos de ellos están cercanos (AA2021 y AA2122). Este grupo abarca a los estudiantes que se relacionan poco en sus mensajes y tardan más en contestar.

Obtención de los centroide similares Obtenemos los centroides para analizar los límites entre cada grupo.

Visualmente podemos obtener los límites:

1er grupo: los unificadores: probabilidad afectiva >= 0.5 y rapidez de contestacion >= 0.6

2do grupo: los diplomáticos: probabilidad afectiva < 0.5 y rapidez de contestación >= 0.6

3er grupo: los indiferentes: rapidez de contestación < 0.6

[20]:
```
#total 6
#print('Los unificadores')
```

```
dfCentUnificadores = df[(df['proAfeEsc'] >= 0.5) & (df['resHorasEsc'] >= 0.6)]
dfCentUnificadores

#total 8
#print('Los diplomáticos')
dfCentDiplomaticos = df[(df['proAfeEsc'] < 0.5) & (df['resHorasEsc'] >= 0.6)]
dfCentDiplomaticos

#total 5; indices: 1,5,10,14,16
print('Centroides de los indiferentes')
dfCentIndiferentes = df[df['resHorasEsc'] < 0.6]
dfCentIndiferentes
```

[19]:
```
dfCentIndiferentes.head()
```

[24]:
```
#cargar todos los mensajes compartidos entre los participantes de todos los␣
 ↪grupos todos los años
dfTodosMsj = pd.read_csv('todos_mensajes.csv')
#las columnas resHorasEscala y resProEscala estàn escaladas
dfTodosMsj.describe()
```

Para obtener los estudiantes que intercambiaron mensajes, se cruzan los centroides (cluster y año) con los mensajes originales (cluster y año)

[22]:
```
#ver los centroides de los indiferentes
dfCentIndiferentes[dfCentIndiferentes['curso']==11920]
```

[23]:
```
#ver todos los mensajes
dfTodosMsj[(dfTodosMsj['curso']==11920) & dfTodosMsj['cluster']== 1]
```

[25]:
```
#cruzar los indiferentes con sus mensajes para saber quiènes son los que tienen␣
 ↪valores similares
#sin embargo, se debe aplicar el filtro de los limites en los centroides
dfMsjIndiferentes=pd.
 ↪merge(dfCentIndiferentes[dfCentIndiferentes['curso']==11920],
                           dfTodosMsj[dfTodosMsj['curso']==11920],␣
 ↪on=['cluster', 'curso'])
#dfMsjIndiferentes.describe()
len(dfMsjIndiferentes)
```

[27]:
```
dfMsjIndiferentes
#Aplicar el lìmite definido para loscentroides.
dfMsjIndiferentes= dfMsjIndiferentes[dfMsjIndiferentes['resHorasEscala'] < 0.6]
dfMsjIndiferentes
```

[28]:
```
dfMsjIndiferentes.describe()
```

[29]:
```
dfMsjIndiferentes['source'].unique()
```

Por esos centroides debo obtener los puntos asociados y obtener minimos y màximos para generar un lìmite para los pùntos. Ejemplo, los unificadores

[30]:
```
#Los diplomàticos: 0.00125; 0.016 (el lìmite està PA<5; R>=6)
dfMsjDiplomaticos=pd.
 ↪merge(dfCentDiplomaticos[dfCentDiplomaticos['curso']==11920]
                           ,dfTodosMsj[dfTodosMsj['curso']==11920],␣
 ↪on=['cluster', 'curso'])
len(dfMsjDiplomaticos)
```

[31]:
```
dfMsjDiplomaticos.describe()
```

[76]:
```
dfMsjDiplomaticos = dfMsjDiplomaticos[(dfMsjDiplomaticos['resProAfeEscala'] < 0.
 ↪5 )
                                      &
                                      (dfMsjDiplomaticos['resHorasEsc'] >= 0.6) ]
dfMsjDiplomaticos.describe()
```

[33]:
```
#obtener los estudiantes diplomàticos
dfMsjDiplomaticos['source'].unique()
```

[35]:
```
#Los unificadores: PA: 0.55; 0.656; IR: 0.91 - 0.99 (el lìmite està PA >= 5;␣
 ↪R>=6)
dfMsjUnificadores=pd.
 ↪merge(dfCentUnificadores[dfCentUnificadores['curso']==11920],
                           dfTodosMsj[dfTodosMsj['curso']==11920],␣
 ↪on=['cluster', 'curso'])
len(dfMsjUnificadores)
```

[36]:
```
dfMsjUnificadores = dfMsjUnificadores[(dfMsjUnificadores['resProAfeEscala'] >=␣
 ↪0.5)
                                      &
                                      (dfMsjUnificadores['resHorasEsc'] >= 0.6)]
dfMsjUnificadores.describe()
```

[37]:
```
#Obtener los estudiantes ùnicos
dfMsjUnificadores['source'].unique()
```

0.0.3 2. Graficar los centroides de comunicación

[38]:
```
#dfCom = pd.read_csv('centroides_comunicacion.csv', header=None,␣
 ↪names=['Degree', 'closnesscentrality', 'betweenesscentrality','curso',␣
 ↪'cluster'])
```

```
dfCom = pd.read_csv('centroides_comunicacion.csv')
dfCom.head()
```

[40]:
```
#Revisamos los valores mìnimos y màximos de las coordenadas de los centroides
dfCom.describe()
```

[43]:
```
dfAA1920Com = dfCom[dfCom['curso'] == 11920]
dfAA2021Com = dfCom[dfCom['curso'] == 12021]
dfAA2122Com = dfCom[dfCom['curso'] == 12122]
dfMD1920Com = dfCom[dfCom['curso'] == 21920]
dfMD2021Com = dfCom[dfCom['curso'] == 22021]
dfMD2122Com = dfCom[dfCom['curso'] == 22122]
```

[44]:
```
dfAA1920Com
```

[45]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.set_xlabel("Degree")
ax.set_ylabel("Closnesscentrality")
ax.set_zlabel("Betweeness")

ax.set_title("Comunicación: Centroides para clústeres en materias AA y MDMS")
ax.
 ↪scatter(dfAA1920Com['Degree'],dfAA1920Com['closnesscentrality'],dfAA1920Com['betweenesscenti
 ↪c="red", s=70, label='AA1920')
ax.
 ↪scatter(dfAA2021Com['Degree'],dfAA2021Com['closnesscentrality'],dfAA2021Com['betweenesscenti
 ↪c="green", s=90, label='AA2021')
ax.
 ↪scatter(dfAA2122Com['Degree'],dfAA2122Com['closnesscentrality'],dfAA2122Com['betweenesscenti
 ↪c="orange", s=90, label='AA2122')
ax.
 ↪scatter(dfMD1920Com['Degree'],dfMD1920Com['closnesscentrality'],dfMD1920Com['betweenesscenti
 ↪c="blue", s=60, label='MD1920')
ax.
 ↪scatter(dfMD2021Com['Degree'],dfMD2021Com['closnesscentrality'],dfMD2021Com['betweenesscenti
 ↪c="cyan", s=60, label='MD2021')
ax.
 ↪scatter(dfMD2122Com['Degree'],dfMD2122Com['closnesscentrality'],dfMD2122Com['betweenesscenti
 ↪c="purple", s=60, label='MD2122')
#ax.legend(loc="best")
ax.legend()
fig.set_size_inches(7, 7)
```

La figura anterior permite apreciar dos áreas que se forman con los centroides graficados:

1. Zona: Alto valor de Centralidad de Intermediación, alto valor de Centralidad de cercanía y

alto Grado (Degree)

2. Zona: Bajo valor de Centralidad de Intermediacion, bajo valor de Centralidad de cercanía y bajo Grado (Degree).

[48]:
```
#Cambiando los ejes de las variables, para validar la existencia de dos grupos
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.set_xlabel("Betweeness")
ax.set_ylabel("Degree")
ax.set_zlabel("Closnesscentrality") #Closnesscentrality

ax.set_title("Comunicación: Centroides para clústeres en materias AA y MDMS")
ax.
 ↪scatter(dfAA1920Com['betweenesscentrality'],dfAA1920Com['Degree'],dfAA1920Com['closnesscenti
 ↪c="red", s=70, label='AA1920')
ax.
 ↪scatter(dfAA2021Com['betweenesscentrality'],dfAA2021Com['Degree'],dfAA2021Com['closnesscenti
 ↪c="green", s=90, label='AA2021')
ax.
 ↪scatter(dfAA2122Com['betweenesscentrality'],dfAA2122Com['Degree'],dfAA2122Com['closnesscenti
 ↪c="orange", s=90, label='AA2122')
ax.
 ↪scatter(dfMD1920Com['betweenesscentrality'],dfMD1920Com['Degree'],dfMD1920Com['closnesscenti
 ↪c="blue", s=60, label='MD1920')
ax.
 ↪scatter(dfMD2021Com['betweenesscentrality'],dfMD2021Com['Degree'],dfMD2021Com['closnesscenti
 ↪c="cyan", s=60, label='MD2021')
ax.
 ↪scatter(dfMD2122Com['betweenesscentrality'],dfMD2122Com['Degree'],dfMD2122Com['closnesscenti
 ↪c="purple", s=60, label='MD2122')
#ax.legend(loc="best")
ax.legend()
fig.set_size_inches(7, 7)
```

Se identifican dos grupos de centroides: Comunicativos y poco comunicativos. Se aprecia que la variable que divide de forma más precisa los grupos es el grado. Por lo tanto se procede a usar el grado para dividir los dos grupos de centroides.

[51]:
```
#Extracción de los grupos: comunicativos
#Grupo comunicativos: total 6
dfCentroidesCom = dfCom[(dfCom['Degree'] >= 2 ) & (dfCom['curso']==11920)]
dfCentroidesCom
```

[54]:
```
#Grupo poco comunicativos
dfCentroidesPocoCom = dfCom[(dfCom['Degree'] < 2) & (dfCom['curso']==11920)]
#centroides: 13
dfCentroidesPocoCom
```

[55]:
```
#Obtener el grupo de los comunicativos
#Se debe recordar que este archivo tiene los valores escalados
dfTodosEstudiantes = pd.read_csv('todos_estudiantes.csv')
dfTodosEstudiantes.head()
```

[56]:
```
dfTodosEstudiantes.describe()
```

[62]:
```
#cruzar la lista de estudiantes originales con sus mètricas con los grupos␣
↪Comunicativo y no Comunicativo
dfEstudiantes11920 = dfTodosEstudiantes[dfTodosEstudiantes['curso'] == 11920]
len(dfEstudiantes11920)
```

[66]:
```
dfEstudiantes11920.head()
```

[63]:
```
dfCentroidesCom
```

Obtener los estudiantes comunicativos

[67]:
```
dfEstudCom11920 = pd.merge(dfEstudiantes11920, dfCentroidesCom, on=['cluster',␣
↪'curso'])
dfEstudCom11920 = dfEstudCom11920[dfEstudCom11920['Degree_x'] >= 2]
dfEstudCom11920
```

[70]:
```
dfEstudCom11920['estudiante'].unique()
```

Obtener los estudiantes poco comunicativos

[72]:
```
dfCentroidesPocoCom
```

[74]:
```
#Obtener los estudiantes comunicativos
dfEstudPocoCom11920 = pd.merge(dfEstudiantes11920, dfCentroidesPocoCom,␣
↪on=['cluster', 'curso'])
dfEstudPocoCom11920 = dfEstudPocoCom11920[dfEstudPocoCom11920['Degree_x'] < 2]
dfEstudPocoCom11920
```

[75]:
```
dfEstudPocoCom11920['estudiante'].unique()
```

6.1_formacion_grupos_ccomunicacion_v4

February 16, 2024

0.0.1 Estrategias para formar grupos considerando la comunicación y cohesión

El objetivo de este notebook es programar una estrategia para formación de grupos, considerando los criterios de comunicación y cohesión en el equipo.

Se forman por defecto equipos de 4 personas y la estrategia a emplear es la siguiente:

Se usa el criterio de comunicación para obtener 2 estudiantes comunicativos y se busca un diplomático y un indiferente.

Ventajas y desventajas:

El grupo podría llegar a tener solamente dos estudiantes, si no se encuentran miembros diplomáticos e indiferentes.

Pueden quedar más estudiantes disponibles al final. Estos pueden ser asignados a otros grupos, según considere el profesor.

[24]:
```
import pandas as pd
```

Parametrización Solicitar:

(1) número de participantes;
(2) Primer criterio para formar grupos

[25]:
```
#Indicar el número de participantes
NUM_ELEMENTOS = 4

#parametros de prueba còdigos de materias: 1 es AA; 2 es MDMS
CURSO = 2021
MATERIA = 2
DATASET = int(str(MATERIA) + str(CURSO))
DATASET
```

0.0.2 1ra Estrategia: Criterios COMUNICACION luego COHESIÓN

1.1 Estimar el número de grupos que se formarán Para facilidad en el manejo de datos se tiene un archivo donde constan todos los estudiantes.

[26]:
```
dfTodosEstudiantes = pd.read_csv('todos_estudiantes.csv')
numTotalEstudiantes = len(dfTodosEstudiantes[dfTodosEstudiantes['curso'] ==␣
↪DATASET])
print(numTotalEstudiantes)
```

1.2 Se retirna coordinadores de los grupos a formar

[27]:
```
dfTodosEstudiantes = pd.read_csv('todos_estudiantes.csv')
dfTodosEstudiantes = dfTodosEstudiantes[(dfTodosEstudiantes['curso'] == DATASET)
                    & (~dfTodosEstudiantes['estudiante'].
↪isin(['Coor-36','Prof-4']))]
NUM_PARTICIPANTES = len(dfTodosEstudiantes)
print(NUM_PARTICIPANTES)
```

[28]:
```
#El número ideal de sujetos en un grupo es 4.2
NUM_GRUPOS, RESTANTES = divmod(NUM_PARTICIPANTES, NUM_ELEMENTOS)
print("Número de grupos: ", NUM_GRUPOS)
print("Estudiantes restantes: ", RESTANTES)
```

1.3 Ordenar el grupo por el criterio Comunicación

[29]:
```
dfTodosEstudiantes.head()
```

[30]:
```
#obtener los estudiantes del grupo de prueba
dfEstudiantesDataset = dfTodosEstudiantes[dfTodosEstudiantes['curso'] ==␣
↪DATASET]
dfEstudiantesDataset.describe()
```

[31]:
```
#dfEstudiantesDataset[dfEstudiantesDataset['estudiante']=='Coor-36']
dfEstudiantesDataset.head(5)
```

Crear nuevas columnas para asignar los valores de los grupos y los tipos de estudiantes que forman el grupo. Los estudiantes se acuerdan de acuerdo al grado para tomar desde los más comunicativos hasta los menos comunicativos.

[32]:
```
#ordenar los estudiantes de acuerdo al grado,y asignar bandera de␣
↪disponibilidad para formar grupo
dfEstudComu = dfEstudiantesDataset[['Degree', 'estudiante']].
↪sort_values('Degree', ascending=False)
dfEstudComu['Disponible'] = 1
dfEstudComu['Grupo'] = 0
dfEstudComu['Tipo'] = '0'
dfEstudComu.head(5)
```

[33]:
```
dfEstudComu.describe()
```

1.4 Procesar grupos de 2 estudiantes Antes de procesar grupos de 2 estudiantes, es necesario crear algunas funciones para formar el grupo de 4 personas.

Creación de una función para obtener los grupos de estudiantes: Unificadores, Diplomáticos e Indiferentes, que se generan por el criterio de Cohesión.

[34]:
```
def obtenerGruposCohesion(DATASET, tipoGrupoCohesion, noDisponibles):

    #1. obtener el dataset de los centroides
    df = pd.read_csv('centroides_cohesion.csv')
    df = df[df['curso'] == DATASET]

    #2. obtener los centroides
    if tipoGrupoCohesion == 'Diplomaticos':
        #dfCentDiplomaticos = df[(df['proAfeEsc'] < 0.5) & (df['resHorasEsc']␣
↪>= 0.6)]
        dfCentroides = df[(df['proAfeEsc'] < 0.5) & (df['resHorasEsc'] >= 0.6)]
    elif tipoGrupoCohesion == 'Indiferentes':
        dfCentroides = df[df['resHorasEsc'] < 0.6]
    elif tipoGrupoCohesion == 'Unificadores':
        dfCentroides = df[(df['proAfeEsc'] >= 0.5) & (df['resHorasEsc'] >= 0.6)]

    #3. Cruzar con todos los mensajes
    dfTodosMsj = pd.read_csv('todos_mensajes.csv')
    #print('Imprimiendo todos los mensajes: print')
    dfTodosMsj = dfTodosMsj[~dfTodosMsj['source'].isin(['Coor-36','Prof-4'])]
    #print(len(dfTodosMsj))
    dfMsjSubgrupo = pd.merge(dfCentroides[dfCentroides['curso']== DATASET]
                             ,dfTodosMsj[dfTodosMsj['curso']==DATASET],␣
↪on=['cluster', 'curso'])

    #4. Ajustar los valores dentro del grupo de mensajes, considerando los␣
↪límites
    if tipoGrupoCohesion == 'Diplomaticos':
        dfMsjSubgrupo = dfMsjSubgrupo[(dfMsjSubgrupo['resProAfeEscala'] < 0.5)
                                      & (dfMsjSubgrupo['resHorasEsc'] >= 0.6) ]
    elif tipoGrupoCohesion == 'Indiferentes':
        dfMsjSubgrupo = dfMsjSubgrupo[dfMsjSubgrupo['resHorasEsc'] < 0.6 ]
    elif tipoGrupoCohesion == 'Unificadores':
        dfMsjSubgrupo = dfMsjSubgrupo[(dfMsjSubgrupo['resProAfeEscala'] >= 0.5)
                                      & (dfMsjSubgrupo['resHorasEsc'] >= 0.6)]

    #4. Obtener los sujetos distintos
    #Convertir a conjunto
    estuSubgrupo = set(dfMsjSubgrupo['source'].unique())
    #Hacer la operacion de conjuntos: substracciòn, y convertir el set a una␣
↪serie pandas
    grupoDisponible = pd.Series(list(estuSubgrupo - noDisponibles))
```

```
    return grupoDisponible
```

Función para obtener la mejor correlación considerando un grupo de candidatos

Se toma el grupo base y se prueba con cada candidato para verificar la suma mayor de correlaciones (tres estudiantes). Al identificar la suma mayor, se obtiene el candidato asociado para insertarlo en el grupo que se está fomando con la base.

[35]:
```
#Se le envìa como parâmetro un grupo de estudiantes separados por cohesiòn:
#Unificador; Indiferente, Diplomatico

def obtenerMejorCorrelacion(grupoBase, grupoCandidatos):

    #1. Levantar la matriz de correlaciòn del DATASET
    archivoPearson='dataset_ultimo/matrices_coefPearson/
↪matrizCPearson_'+str(DATASET)+'.csv'
    #print(archivoPearson)
    dfPearson = pd.read_csv(archivoPearson,  index_col=[0])
    #print(dfPearson)

    #2. Buscar la suma de correlaciones con distintos candidatos
    #print('Buscando la mejor suma de correlaciones')
    sumas = {}

    for i in range(0, len(grupoCandidatos)):
        #Encerando las sumas parciales, con cada candidato a prueba
        valor1 = 0
        valor2 = 0

        if (grupoBase[0][0] != grupoCandidatos[i]) & (grupoBase[1][0] !=␣
↪grupoCandidatos[i]):
            valor1 = dfPearson.loc[grupoBase[0][0],grupoCandidatos[i]]
            valor2 = dfPearson.loc[grupoBase[1][0],grupoCandidatos[i]]
            #obtener la suma total
            sumaTotal = valor1 + valor2
            sumas[grupoCandidatos[i]] = sumaTotal

    #3. Obtener el mayor valor
    #Si existen varios candidatos se toma el primero que devuelve el␣
↪diccionario.
    if(len(sumas) > 0):
        #print('largo sumas: ', len(sumas))
        candidatoMaximiza =max(sumas, key=sumas.get)
    else:
        candidatoMaximiza = None

    return candidatoMaximiza
```

La siguiente función actualiza el dataframe para que los miembros ya tomados en un grupo, no estén disponibles

[36]:
```
def marcarDisponibilidadGrupo(df, grupoBase, mDiplomatico, mIndiferente,␣
↪numGrupo):

    if mDiplomatico == None:
        mDiplomatico = 'NED'

    if mIndiferente == None:
        mIndiferente = 'NEI'

    #crear una lista vacia
    grupoFinal = []

    grupoFinal.append(grupoBase[0][0] + ';' + 'C')
    grupoFinal.append(grupoBase[1][0] + ';' + 'C')
    grupoFinal.append(mDiplomatico + ';' + 'D')
    grupoFinal.append(mIndiferente + ';' + 'I')

    detalle = None

    for i in grupoFinal:
        partes = i.split(';')
        print("Dentro funcion ", partes[0])
        df.loc[df['estudiante']== partes[0], ['Disponible']] = 0
        df.loc[df['estudiante']== partes[0], ['Grupo']] = numGrupo
        df.loc[df['estudiante']== partes[0], ['Tipo']] = partes[1]

    return df
```

1.5 Lazo general para iterar sobre todos los estudiantes del grupo

[37]:
```
#iterar sobre el dataset para obtener los grupos de personas comunicativas
banderaDisponibles = 1

contadorGrupos = 1
vueltas = 0

#Recuperar la primera vez los estudiantes disponibles
dfFinalGrupos = dfEstudComu[dfEstudComu['Disponible'] == 1]

while banderaDisponibles == 1:

    print('Iteración: ', vueltas)
    dfDisponibles = dfFinalGrupos[dfFinalGrupos['Disponible'] == 1]
    #transformar la serie a un conjunto
```

```
    setNoDisponibles =␣
↪set(dfFinalGrupos['estudiante'][dfFinalGrupos['Disponible'] == 0].unique())
    print('Cuántos disponibles: ?')
    print(len(dfDisponibles))

    if (len(dfDisponibles) > 0) & (len(dfDisponibles) > 2):
        #tomar 2 casos
        #grupoBase= dfEstudComu.iloc[0:2, 1:2]
        #grupoBase= dfEstudComu.iloc[0:2, 1:2].to_numpy()
        grupoBase= dfDisponibles.iloc[0:2, 1:2].values.tolist()
        print('Función principal: grupo base')
        print(grupoBase)
        #print(grupoBase[0])

        #Buscar un candidato del Diplomàtico
        candidatosDiplomaticos=obtenerGruposCohesion(DATASET, 'Diplomaticos',␣
↪setNoDisponibles)
        candidatosIndiferentes=obtenerGruposCohesion(DATASET, 'Indiferentes',␣
↪setNoDisponibles)

        #para este grupo de 2 buscar otros 2: 1 Diplomático y 1 Indiferente
        diplomatico = obtenerMejorCorrelacion(grupoBase, candidatosDiplomaticos)
        print('Miembro Diplomatico: ', diplomatico)
        #si el miembro diplomatico no existe,entonces intentar buscar un␣
↪indiferente
        indiferente = obtenerMejorCorrelacion(grupoBase, candidatosIndiferentes)
        print('Miembro Indiferente: ', indiferente)

        #marcar la disponibilidad
        dfFinalGrupos = marcarDisponibilidadGrupo(dfFinalGrupos, grupoBase,␣
↪diplomatico, indiferente, contadorGrupos)
        contadorGrupos = contadorGrupos + 1

        #banderaDisponibles = 3
        print('Cómo queda el df?')
        print(dfFinalGrupos)

        vueltas = vueltas + 1

    else:
        print('Termina la formación de grupos...')
        banderaDisponibles = 0
        print('Usuarios disponibles')
        print(dfFinalGrupos)
```

1.6 Grupos obtenidos por 1er Criterio: COMUNICACION, 2do: COHESION

[38]:
```
dfFinalGrupos.sort_values(['Grupo', 'Tipo']).head(10)
```

Alumnos que quedan restantes y pueden añadirse a otros grupos, segùn considere el profesor

[39]:
```
#Validar los estudiantes que quedan finalmente como disponibles
len(dfFinalGrupos[dfFinalGrupos['Disponible'] == 1])
```

Grupos finalmente formados y detalle

[40]:
```
#Indicar cuántos grupos quedaron finalmente formados
len(dfFinalGrupos[dfFinalGrupos['Grupo'] > 0].groupby(by=['Grupo'])['Grupo'])
```

Detalle de grupos formados. El grupo “0” es aquel que tiene los estudiantes restantes, no asignados a ningún grupo.

[41]:
```
dfFinalGrupos.groupby(by=['Grupo'])['estudiante'].count()
```

6.2_formacion_grupos_eextrema_v4

February 16, 2024

0.0.1 Estrategia para formar grupos de forma extrema: miembros discordantes

El objetivo de este notebook es programar una estrategia para formación de grupos, considerando los criterios de comunicación y cohesión en el equipo. La estrategia que se plantea es formar equipos con características cercanas a la vida real, donde las personas no se conocen y posiblemente se elevan las diferencias de los criterios: comunicativos con poco-comunicativos; y con un par de estudiantes cuya cohesión tiene una relación baja en fuerza con los dos primeros elementos del grupo. De este modo, se obliga a la formación de un equipo con menos similitudes, es decir más heterogéneo.

Se forman por defecto equipos de 4 personas y la estrategia a emplear es la siguiente:

1. Partir de grupos con 2 estudiantes: comunicativo y poco comunicativo.
2. Completar el grupo con dos estudiantes: uno por vez, cuya cohesión tenga el menor valor en la correlación de Pearson, es decir, el menor valor en la suma total de correlaciones entre los tres miembros probados.

   Ventajas y Desventajas:
   Con el algoritmo de esta estrategia, siempre se completan 4 miembros.
   Se minimiza el número de estudiantes que restan al final, y que podrían ser ubicados por el tutor en cualquier grupo.

[218]:
```
import pandas as pd
```

Parametrización Solicitar:

(1) número de participantes;
(2) Primer criterio para formar grupos

[219]:
```
#Indicar el número de participantes
#NUM_PARTICIPANTES = 27
NUM_ELEMENTOS = 4

#parametros de prueba: 1 es AA; 2 es MDMS
CURSO = 2122
MATERIA = 1
DATASET = int(str(MATERIA) + str(CURSO))
DATASET
```

0.0.2 Estrategia de la vida real: formar grupos disímeles en Comunicación y Cohesión

1.1 Estimar el número de grupos que se formarán El archivo de todos los estudiantes contiene todos los alumnos para recuperar la información con mayor facilidad.

[267]:
```
#se recupera toda la lista de estudiantes
dfTodosEstudiantes = pd.read_csv('todos_estudiantes.csv')
numTotalEstudiantes = len(dfTodosEstudiantes[dfTodosEstudiantes['curso'] ==␣
↪DATASET])
print(numTotalEstudiantes)
```

1.2 Se retiran coordinadores de los grupos a formar

[221]:
```
dfTodosEstudiantes = pd.read_csv('todos_estudiantes.csv')
dfTodosEstudiantes = dfTodosEstudiantes[(dfTodosEstudiantes['curso'] == DATASET)
                    & (~dfTodosEstudiantes['estudiante'].
↪isin(['Coor-36','Prof-4']))]
NUM_PARTICIPANTES = len(dfTodosEstudiantes)
print(NUM_PARTICIPANTES)
```

[237]:
```
#Estimar el número de grupos
NUM_GRUPOS, RESTANTES = divmod(NUM_PARTICIPANTES, NUM_ELEMENTOS)
print("Número estimado de grupos: ", NUM_GRUPOS)
print("Estudiantes restantes: ", RESTANTES)
```

[223]:
```
dfTodosEstudiantes.head()
```

1.3 Ordenar el grupo por el criterio Comunicación Se ordenan los estudiantes según el grado, para tomar grupos de dos en dos. Se toman los estudiantes de los extremos superior e inferior. Así tendremos un estudiante más comunicativo y otro menos comunicativo.

[225]:
```
#formar los pares reuniendo uno de grado mayor y uno de grado menor
dfOrdenar = dfTodosEstudiantes[['Degree', 'estudiante']].sort_values(by=␣
↪['Degree'], ascending=False).reset_index(drop=True)
dfOrdenar.head()
```

Añadir columnas para organizar los grupos e identificar el tipo de estudiante. Estas columnas se llenarán a medida que se formen los grupos.

[226]:
```
#ordenar los estudiantes para agruparlos de 2 en 2: uno de alto grado y otro de␣
↪menor grado-
#dfOrdenar.reset_index(drop=True)
dfOrdenar['Disponible'] = 1
dfOrdenar['Grupo'] = 0
dfOrdenar['Tipo'] = '0'
dfOrdenar.head()
```

1.4 Buscar los estudiantes con los cuales se obtenga la menor suma en la correlación de Pearson para todo el grupo Esta funcion, recibe un grupo de 2 estudiantes, para luego buscar la menor suma de correlaciones considerando el criterio de cohesiòn y proponer esos candidatos para completar el grupo. El objetivo es buscar estudiantes que tengan la mínima correlación entre sí.

[258]:
```
#La funcion recibe una lista
def obtenerPeorCorrelacion(grupoBase, p_setNoDisponibles):

    candidatosMinimizan = None

    #1. Levantar la matriz de correlaciòn del DATASET
    archivoPearson='dataset_ultimo/matrices_coefPearson/
↪matrizCPearson_'+str(DATASET)+'.csv'
    #print(archivoPearson)
    dfPearson = pd.read_csv(archivoPearson,  index_col=[0])
    #print(dfPearson)

    #Obtener la lista de candidatos y excluir los que están en el grupo base
    todos = pd.Series(dfPearson.index)
    potencialesCandidatos = todos.loc[(todos != 'Coor-36') & (todos !=␣
↪'Prof-4')]
    conjEstudiantes = set(potencialesCandidatos.unique())
    #los candidatos a revisar
    grupoDisponiblesPre = conjEstudiantes - set(grupoBase)
    grupoDisponiblesFin = grupoDisponiblesPre - p_setNoDisponibles

    #convertir el conjunto a lista
    grupoCandidatos = list(grupoDisponiblesFin)

    #2. Buscar la suma de correlaciones con distintos candidatos
    #print('Buscando la suma más baja de correlaciones')
    sumas = {}

    for i in range(0, len(grupoCandidatos)):
        #Encerando las sumas parciales, con cada candidato a prueba
        valor1 = 0
        valor2 = 0
        #Obtener las sumas para los tres candidatos: 2 del grupo base más el␣
↪propuesto
        valor1 = dfPearson.loc[grupoBase[0],grupoCandidatos[i]]
        valor2 = dfPearson.loc[grupoBase[1],grupoCandidatos[i]]
        #obtener la suma total
        sumaTotal = valor1 + valor2
        sumas[grupoCandidatos[i]] = sumaTotal

    #3. Obtener los dos valores menores
```

```
    #Si existen varios candidatos se toman los dos primeros que devuelve el␣
 ↪diccionario.
    if(len(sumas) > 0):
        #print('largo sumas: ', len(sumas))
        #ordenar diccionario por valores
        sumasOrdenadas = sorted(sumas.items(), key=lambda x:x[1])
        dicSumasOrdenadas = dict(sumasOrdenadas)
        if len(sumas) >= 2:
            dicCandidatos = dict(list(dicSumasOrdenadas.items())[0:2])
        else:
            #si solo existe un candidato, entonces devuelve el único candidato
            dicCandidatos = dict(list(dicSumasOrdenadas.items())[0:1])

        candidatosMinimizan = list(dicCandidatos.keys())

    return candidatosMinimizan
```

Función para marcar disponibilidad en los estudiantes: cuando ya se ocupa un estudiante, se lo marca como no disponible

[260]:
```
#Esta función actualiza el dataframe para que los miembros ya no estèn␣
 ↪disponibles
def marcarDisponibilidadGrupo(df, grupoBase, miembros, numGrupo):

    #crear una lista vacia
    grupoFinal = []

    #Los dos primeros son el comunicativo y el no comunicativo:
    grupoFinal.append(grupoBase[0] + ';' + 'C')
    grupoFinal.append(grupoBase[1] + ';' + 'PC')

    #iterar sobre la lista de miembros para anexarlos
    for j in miembros:
        grupoFinal.append(j + ';' + 'NC')

    #print(grupoFinal)
    #print(df)
    detalle = None

    for i in grupoFinal:
        partes = i.split(';')
        #print("Dentro funcion ", partes[0])
        df.loc[df['estudiante']== partes[0], ['Disponible']] = 0
        df.loc[df['estudiante']== partes[0], ['Grupo']] = numGrupo
        df.loc[df['estudiante']== partes[0], ['Tipo']] = partes[1]

    return df
```

1.5 Procesar grupos de 2 estudiantes: comunicativo y poco comunicativo

[263]:
```
#Lazo para iterar sobre el dataset para obtener los grupos de personas␣
↪comunicativas
banderaDisponibles = 1

contadorGrupos = 1
vueltas = 0

#Recuperar la primera vez los estudiantes disponibles
dfFinalGrupos = dfOrdenar[dfOrdenar['Disponible'] == 1]

while banderaDisponibles == 1:

    print('Iteración: ', vueltas)
    dfDisponibles = dfFinalGrupos[dfFinalGrupos['Disponible'] == 1]
    #transformar la serie a un conjunto
    setNoDisponibles =␣
↪set(dfFinalGrupos['estudiante'][dfFinalGrupos['Disponible'] == 0].unique())
    print('¿Cuántos disponibles? ')
    print(len(dfDisponibles))

    if (len(dfDisponibles) > 2):
        grupoBase= list(dfDisponibles.iloc[[0, -1],[1]].squeeze())#.values.
↪tolist()#.flatten()
        print('Grupo base:')
        #print(type(grupoBase))
        #print('Funcion principal, grupo base')
        print(grupoBase)

        #1. Buscar dos candidatos de entre todo el grupo
        candidatos = obtenerPeorCorrelacion(grupoBase, setNoDisponibles)
        #2. Marcar disponibilidad
        dfFinalGrupos = marcarDisponibilidadGrupo(dfFinalGrupos, grupoBase,␣
↪candidatos, contadorGrupos)
        contadorGrupos = contadorGrupos + 1

        #banderaDisponibles = 3
        print('Cómo queda el df')
        print(dfFinalGrupos)
        vueltas = vueltas + 1

    else:
        print('Termina la formación de equipos: ')
        banderaDisponibles = 0
        print('Usuarios que restan disponibles')
        print(dfFinalGrupos)
```

1.6 Grupos obtenidos por 1er Criterio: COMUNICACION, 2do: COHESION - vida real

[265]:
```
dfFinalGrupos.sort_values(['Grupo', 'Tipo']).head(10)
```

1.7 Alumnos que restan y pueden añadirse a otros grupos, según considere el tutor

[266]:
```
#Validar los estudiantes que quedan finalmente como disponibles
len(dfFinalGrupos[dfFinalGrupos['Disponible'] == 1])
```

1.8 Grupos finalmente formados y detalle

[256]:
```
#Indicar cuántos grupos quedaron finalmente formados
len(dfFinalGrupos[dfFinalGrupos['Grupo'] > 0].groupby(by=['Grupo'])['Grupo'])
```

Detalle de grupos formados. El grupo "0" es aquel que tiene los estudiantes restantes, no asignados a ningún grupo.

[257]:
```
dfFinalGrupos.groupby(by=['Grupo'])['estudiante'].count()
```