FACULTAD DE INGENIERÍA UDELAR

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

Procesamiento automático de respuestas abiertas en encuestas a usuarios de servicios del INAU

Nestor Andrés Moreira Quijano

Tesis de Licenciatura presentada a la Comisión de Carrera en Computación, Facultad de Ingeniería de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de Licenciado en Computación.

Directores:
Prof. Aiala Rosá
Prof. Juan José Prada

Montevideo – Uruguay
Julio de 2022

Moreira Quijano , Nestor Andrés

Procesamiento automático de respuestas abiertas en encuestas a usuarios de servicios del INAU / Nestor Andrés Moreira Quijano . - Montevideo: Universidad de la República, Facultad de Ingeniería, 2022.

XXIII, 83 p.: il.; 29, 7cm.

Directores:

Aiala Rosá

Juan José Prada

Tesis de Licenciatura – Universidad de la República, Programa en Computación, 2022.

Referencias bibliográficas: p. 69 – 71.

1. pln, 2. aprendizaje no supervisado, 3. modelado de tópicos, 4. LDA, 5. Clustering, 6. Sentence embeddings, 7. SentenceBERT, 8. HDBSCAN, 9. t-SNE, 10. UMAP. I. Rosá, Aiala, Prada, Juan José, . II. Universidad de la República, Programa de Posgrado en Computación. III. Título.

INTEGRANTES DEL TRIBUNAL DE DEFENSA DE TESIS

Prof. Adriana Marotta

Prof. Guillermo Moncecchi

Prof. Santiago Góngora

Montevideo – Uruguay
Julio de 2022

Agradecimientos

Quiero agradecer a mi esposa Gabriela y a mis hijos Faustino y Josefina por todas las noches y días (más noches que días) que me apoyaron y acompañaron en la construcción de este proyecto de tesis.

RESUMEN

En los primeros meses del comienzo de la pandemia de COVID-19 las familias uruguayas con hijos a cargo manifestaron estar sufriendo elevados niveles de afectación (Ares et al. 2020). Desde la Secretaría Ejecutiva de Primera Infancia de **INAU** surge la iniciativa de lanzar un programa de acompañamiento familiar en una modalidad virtual llamado *Parentalidades Comprometidas en Casa*.

Este programa fue realizado en una serie de entregas distribuidas en modo virtual usando el sistema de mensajería *Whatsapp*. Cada entrega consistía de un documento con propuestas lúdicas y educativas para padres y madres. Al final de la entrega, se les pedía a los usuarios que completaran un formulario con una opinión sobre esa entrega.

Este trabajo surge de un planteo del **INAU** con el objetivo de automatizar el procesamiento de las respuestas a las encuestas de los usuarios. Estas respuestas son generalmente breves y usando lenguaje natural. El volumen a procesar se puede seguir incrementando (si el programa se continua más allá de la pandemia), lo cual plantea un desafío para procesar todas las respuestas de forma manual y con pocos recursos.

Uno de los objetivos del **INAU** es clasificar las respuestas en una jerarquía de categorías dada, donde cada respuesta pueda pertenecer a una o más de una de estas categorías.

En este documento de tesis se aborda el problema con una primera fase exploratoria de los datos, así como la aplicación de un conjunto de técnicas de aprendizaje automático no supervisado con los objetivos de (a) entender la naturaleza de los datos, (b) identificar características, tópicos y grupos de datos (c) reportar al INAU posibles caminos a seguir para un abordaje del problema a futuro.

Palabras claves:
pln, aprendizaje no supervisado, modelado de tópicos, LDA, Clustering, Sentence embeddings, SentenceBERT, HDBSCAN, t-SNE, UMAP.

Lista de figuras

2.1 Ejemplo del formato de datos y las categorías en el proceso de clasificación del INAU 6
2.2 Diagrama que ilustra como se conectan las diferentes etapas del proceso. 9

3.1 Ejemplo de árbol de recubrimiento mínimo para un set de datos. Fuente: https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html 16
3.2 Dendograma de la jerarquía de clusters. Fuente: https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html 16
3.3 Árbol condensando y los clusters identificados. Fuente: https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html . . . 17
3.4 La intuición detrás del modelado de tópico. Probabilistic topic models communications of the ACM, Blei, D.M. (2010) 19
3.5 Ejemplo del modelo *Bag of Words*. Imagén del libro Applied Text Analysis with Python. Bengfort et al. 2018 21
3.6 Arquitectura de SentBERT (con pesos compartidos entre las dos redes) 25
3.7 Arquitectura de SentBERT en el momento de inferencia. 25
3.8 Comparación de una UMAP vs t-SNE para un conjunto de datos denominado Fashion MNIST. Fuente: Google PAIR 28

4.1 Diagrama de la solución, desde los datos «raw» hasta la etapa de su visualización 31
4.2 DataFrame de Pandas con los datos del Excel luego de haber sido procesados. 33
4.3 Representación del documento número 9 de nuestro corups como una distribución de tópicos. 36

4.4 Representación de cada tópico como una distribución de palabras de nuestro corpus con su correspondiente probabilidad. . . 36
4.5 Curva de perplejidad. 38

5.1 Distribución de las respuestas contando los caracteres (izq) y las palabras (der). 43
5.2 Balance de clases 44
5.3 Correlación entre las clases. 45
5.4 Distribución de las respuestas por género y utilidad. 46
5.5 Nube de palabras de todas las respuestas (se usa Tf-Idf para darle el peso a cada palabra) 48
5.6 Nube de emojis de todas las respuestas (usando frecuencia). La estampita es un emoji que no se tradujo correctamente. 49
5.7 Tópicos encontrados por LDA (5 tópicos con 10 palabras cada uno). 50
5.8 Respuestas codificadas usando LDA, mapeadas en un espacio 2D usando t-SNE. 52
5.9 Respuestas codificadas usando LDA y agrupadas con HDBScan. Se pueden ver todas las respuestas, N/A es que no pudieron ser asignadas a un *cluster*. 53
5.10 Respuestas codificadas usando LDA y agrupadas con HDBScan. Se puede ver solo las respuestas que fueron asignadas a un *cluster* y además, el tópico predominante de cada agrupación (usando el nombre amigable). 54
5.11 Relación entre las categorías y los tópicos. 55
5.12 Representación de las respuestas usando *SentBERT* y UMAP para su visualización en 2D. 57
5.13 Respuestas agrupadas con HDBScan y codificadas con SentenceBERT. Cada «cluster» tiene las palabras más relevantes de sus respuestas. Se resaltan algunos *clusters* que van a ser explorados en detalle más adelante. 58
5.14 En este ejemplo se inspecciona el *cluster* de "Burbujas" donde se pueden ver dos respuestas seleccionadas que mencionan "jugar con burbujas". Este *cluster* se corresponde con una de las actividades propuestas de «divertirse con burbujas» mediante la construcción de un «Burbujero». 59

5.15 Se identifican dos *clusters* que tratan de «canciones» y «música». Las respuestas que se seleccionan sirven como ejemplo para notar que en ambas las canciones para dormir ó para lavarse las manos son mencionadas. Se puede interpretar que estos dos *clusters* puedan ser uno solo que trate de “Canciones y música”. 60
5.16 Se visualizan las respuestas usando *SentenceBERT*, características artificiales y *HDBScan*. Se colorea con naranja (valor 1) si la respuesta (o punto) pertenece a la super categoría determinada por el título. En la izquierda se ve las respuestas que pertenecen a la super categoría «Utilidad». En el centro, las respuestas que pertenecen a «Valoración positiva del dispositivo» y a la derecha, las respuestas que pertenecen a «Observaciones/Sugerencias». El objetivo de esta visualización es mostrar que en esta nueva iteración de la representación de las respuestas, no hay una clara correlación entre las agrupaciones y las super categorías. 62
5.17 Visualización de los datos colorizando por el campo «En caso afirmativo, marque las opciones que reflejan la utilidad». 63
5.18 Nombre dado a los grupos de acuerdo a la opción de utilidad, y su contenido. 64
5.19 Dos instancias que nos sirven como ejemplo para mostrar el contenido del cluster «Juegos y Actividades», asociado con la opción “Hice algo con los niños”. 64
5.20 Visualización de los clusters & palabras claves para el mapeo de «Sentence embeddings» con «Feature Engineering». 65

Lista de tablas

4.1 Algunos términos encontrados en las respuestas, y su correspondiente mapeo (o reemplazo). 34
4.2 Mejores hiperparámetros para el modelo de LDA. 37

5.1 Características del primer conjunto de datos (llamémosle INAU-2022-03-23) 42
5.2 Estadísticas de la columna respuesta 42
5.3 Quintiles y respuestas. 43
5.4 Características del segundo conjunto de datos (llamémosle INAU-2022-03-23-segunda-entrega) 46
5.5 Opciones de la pregunta: "En caso afirmativo, marque las opciones que reflejan la utilidad." 46
5.6 Perplejidad de los modelos de LDA entrenados con o sin preprocesamiento del texto. 50

Lista de siglas

Lista de siglas

INAU Instituto del Niño y Adolescente del Uruguay 1

NLP Natural language processing 11

PLN Procesamiento del Lenguaje Natural 11

UdelaR Universidad de la República 1

Tabla de contenidos

Agradecimientos ix

Lista de figuras xiii

Lista de tablas xvii

Lista de siglas xix

1 Introducción 1

1.1 Motivación . . . 1
1.2 Objetivo . . . 2
1.3 Cronograma de trabajo . . . 3
1.4 Organización del documento . . . 3

2 Definición del problema 5

2.1 Etapas del proceso . . . 8

3 Conceptos preliminares y Estado del arte 11

3.1 Text Clustering . . . 13
3.1.1 HDBScan . . . 14
3.2 Modelado de tópicos . . . 17
3.3 Representación de texto . . . 20
3.3.1 Sentence Embeddings . . . 22
3.3.2 SentBERT: Sentence Embeddings usando redes BERT y siamesas . . . 23
3.4 Reducción de dimensionalidad y Visualización . . . 26
3.4.1 t-SNE y UMAP . . . 26
3.5 Detección de palabras relevantes . . . 30

4 Diseño e implementación de la solución **31**
4.1 Preparación de los datos 32
4.1.1 Conversión Excel a Pandas 32
4.1.2 Tildes y errores tipográficos 33
4.1.3 Enriqueciendo los datos con nuevos datos de las encuestas 34
4.2 Exploración y análisis de datos 34
4.2.1 Modelado de tópicos 35
4.2.2 Clustering y Sentence embeddings 38

5 Análisis de Resultados **41**
5.1 Exploración de los datos 41
5.2 Análisis de frecuencia de palabras y *emojis* 47
5.3 Modelado de tópicos 49
5.4 Modelado con sentence embeddings 55

6 Conclusiones y trabajo a futuro **67**

Referencias bibliográficas **69**

Glosario **76**

Apéndices **77**
Apéndice 1 Apendice A. Repositorio & Datos 79

Anexos **81**
Anexo 1 Programa INAU & Materiales 83

Capítulo 1

Introducción

El presente trabajo surge como una iniciativa del Instituto del Niño y Adolescente del Uruguay (INAU), a través del *Departamento de Asuntos Internacionales y Cooperación*, y se realiza en colaboración con el *Grupo de Procesamiento del Lenguaje Natural* de la *Facultad de Ingeniería*, Universidad de la República (UdelaR).

1.1. Motivación

La *Secretaría Ejecutiva de Primera Infancia del INAU* desarrolló en el año 2020 el programa de *Parentalidades Comprometidas en Casa* como una respuesta a la situación de crisis generada por la pandemia de COVID-19. El objetivo de este programa fue acompañar y apoyar a las familias uruguayas con hijos a cargo en este contexto de pandemia, luego de conocerse reportes (Ares et al. 2020) de cambios importantes en las dinámicas familiares, y cómo estos tenían un impacto directo en ellas.

Este programa se implementó como una serie de cuatro entregas por medio del sistema de mensajería *Whatsapp*. Estas entregas o dispositivos –como le llama el INAU– presentan de forma muy gráfica y amena un conjunto de contenidos enfocados en temas de interés para el programa del INAU. Al final de este documento se presenta un formulario que el INAU creó y distribuyó para relevar la utilidad de estas entregas. Este formulario permite conocer la opinión de los usuarios mediante una serie de preguntas, entre las cuales una de sus respuestas es abierta y es el foco de análisis de este trabajo.

Actualmente el INAU procesa los resultados de estos formularios de forma manual, clasificando las respuestas de los usuarios, que son un texto libre en lenguaje natural, en una jerarquía de categorías de su interés. El INAU planea seguir con este dispositivo y las encuestas más allá de la situación de pandemia. Este trabajo es realizado por los integrantes de la *Secretaría Ejecutiva de Primera Infancia*. Sin embargo, este formato les presenta varias dificultades al equipo del INAU por dos motivos (a) la cantidad de datos obtenidos de las encuestas va en aumento (b) los recursos que disponen son limitados.

1.2. Objetivo

El objetivo del INAU es poder automatizar este proceso para obtener una retroalimentación más efectiva del éxito o no del programa, así como poder dar respuesta a las inquietudes planteadas por los mismos usuarios del sistema a través de modificaciones del contenido o nuevos programas.

En este documento de tesis se aborda el problema con una primera fase donde se exploran y analizan los datos obtenidos, así como también la aplicación de un conjunto de técnicas de aprendizaje automático no supervisado. Los objetivos definidos dentro del alcance de este proyecto se puede dividir en dos grupos, los objetivos para el INAU y los objetivos propios de la tesis. Dentro de los objetivos en relación al INAU están,

(a) entender la naturaleza de los datos,

(b) identificar características, patrones, agrupaciones de datos en el espacio y tópicos o palabras claves más relevantes,

(c) generar un reporte con posibles caminos a seguir para una segunda fase (en caso de que exista), y

(d) analizar relaciones entre tópicos/clusters y categorías definidas.

.

Y dentro de los objetivos propios a la tesis tenemos,

(a) estudiar los datos recibidos,

(b) estudiar técnicas para el modelado de los textos, y

(c) estudiar técnicas para detección de tópicos y agrupamientos (*clusters*)

1.3. Cronograma de trabajo

A continuación se detalla el cronograma de tareas con la duración tentativa, y las fechas en que fueron llevadas a cabo.

Tareas	**Duración**	**Fechas**
Análisis primario de los datos: cantidad de datos anotados, estudio de balance y correlaciones entre categorías, limpieza y formateo de datos	3 semanas	Abril 2022
Experimentos basados en agrupamientos (clustering) y detección de tópicos para analizar su correspondencia con las categorías de análisis propuestas	5 semanas	Mayo a Junio 2022
En base a los resultados anteriores, definir un alcance acotado para el marco de esta tesis y hacer experimentos de clasificación	3 semanas	Junio a Agosto 2022
Evaluación final de resultados	1 semana	Agosto a Septiembre 2022
Escritura de documento final	4 semanas	Septiembre a Diciembre 2022

1.4. Organización del documento

A continuación se presenta la organización del trabajo y una breve descripción del contenido de cada una de las secciones que forman este documento.

Se comienza con el capítulo de **Definición del problema** donde se define y presenta el problema. A continuación se introducen las diferentes técnicas y algoritmos usados en el capítulo **Conceptos preliminares y Estado del arte**. En la siguiente capítulo **Diseño e implementación de la solución** se detallan las decisiones tomadas, las técnicas utilizadas y se hace un breve resumen y análisis de los resultados obtenidos. Seguido, el capítulo de **Análisis de Resultados** donde se presentan los resultados obtenidos, se analizan en detalle y se discute qué nos permitió encontrar cada técnica usada. Al final se abordan las **Conclusiones y trabajo a futuro** profundizando en las conclusiones que se encontraron y se sugieren un conjunto de enfoques a tomar para seguir avanzando dado los objetivos planteados. En los **Apéndices y Anexos** se agregan recursos provistos por el INAU, así como los conjuntos de datos y los recursos técnicos que se fueron construyendo durante la ejecución de está tesis.

Capítulo 2

Definición del problema

El INAU –como anteriormente se detalló– presentó el problema de entender mejor y clasificar las respuestas de los usuarios del sistema desarrollado en el proyecto de *Parentalidades Comprometidas en Casa*, que actualmente se procesan de forma manual. Las respuestas son generalmente textos cortos, expresadas en lenguaje natural, y cuyo contenido puede contener errores ortográficos, de tipeo ó caracteres especiales como los «emojis»[1]. Asimismo, durante el desarrollo del programa se generó una buena cantidad de respuestas que hacen que un tratamiento manual sea muy engorroso y que se requiera un proceso más ágil y eficiente que no dependa –solo– de un conjunto de personas para su procesamiento.

El objetivo del INAU es clasificar de forma automática las respuestas de los usuarios en la siguiente jerarquía de categorías (en la Figura 2.1 se muestra una tabla con su proceso y las categorías),

[1]Según la RAE, un EMOJI es una pequeña imagen o icono digital que se usa en las comunicaciones electrónicas para representar una emoción, un objeto, una idea, etc.

Categoría	**Número de Categoría**
Categorías	
└ **Utilidad**	
└ └ Impulsó un proceso reflexivo en torno a la temática de la entrega	(1)
└ └ Manifiestan haber realizado alguna actividad o su intención de hacerla	(2)
└ └ Manifiestan que no les resultó útil	(3)
└ **Valoración positiva del dispositivo/actividad**	
└ └ Valoración del dispositivo	(4)
└ └ Valoración de actividades concretas	(5)
└ **Sugerencias/Observaciones**	(6)
└ **No clasificables**	(7)

B	C	D	E	F	G	H	I
Respuestas: ENTREGA 1	**Categorías**						
	Utilidad			Valoración positiva del dispositivo/actividad		6- Sugerencias/observaciones	7- No clasificables
	1-Impulsó un proceso reflexivo entorno a la temática de la entrega	2- Manifiestan haber realizado alguna actividad o su intención de hacerla	3 - Manifiestan que no le resultó útil	4- Valoración del dispositivo	5- Valoración de actividades concretas		
Total de apariciones de la subcategoría	72	35	0	188	43	30	46
% de aparición de la subcategoría	17,39%	8,45%	0,00%	45,41%	10,39%	7,25%	11,11%
% de aparición de la categoría		25,85%		55,80%		7,25%	11,11%
Excelente				4			
Muy buena la información				4			
Ecxelente material				4			
Está actividades siempre las realizamos en casa. Cuando guardamos los juguetes. Al bañarse y al dormir. Les gusta mucho cantar ebinteractuar en familia		2			5		
Me gusto mucho la propuesta,ya q hay cosas q me sirven para aplicarlas a nuestras rutinas diarias como lo importante que es el sentarnos juntos a la mesa y q nuestros niños aprendan la importancia de estar la familia reunida a la hora del almuerzo.	1			4	5		
Me gusto mucho la propuesta,tambien para aplicar algunas cosas como la importancia d la comida y el almuerzo en familia	1			4			
me gusta q los niños pepan y qle tengan pasiencia							7

Figura 2.1: Ejemplo del formato de datos y las categorías en el proceso de clasificación del INAU

Las categorías siguen una jerarquía de 2 niveles y con 4 super categorías (Utilidad, Valoración positiva del dispositivo, No clasificables, Sugerencias y Observaciones). Las sub categorías corresponden a la super categoría «Utilidad», con 3 sub categorías y la categoría «Valoración positiva del dispositivo/actividad», con 2 sub categorías.

Las respuestas pueden pertenecer a más de una categoría (o clase, normalmente se usa indistintamente clase o categoría) como se muestra en la Figura 2.1, haciendo que el problema sea un problema de clasificación de las respuestas en múltiples clases (clasificación multiclase). Para ilustrar lo anterior mostramos algunos ejemplos extraídos de los datos de la planilla del INAU:

Respuesta	**Categorías (número de categoría)**
Está actividades siempre las realizamos en casa. Cuando guardamos los juguetes. Al bañarse y al dormir. Les gusta mucho cantar ebinteractuar en familia	Manifiestan haber realizado alguna actividad o su intención de hacerla (2), Valoración de actividades concretas (5)
Me gusto mucho la propuesta,tambien para aplicar algunas cosas como la importancia d la comida y el almuerzo en familia	Impulsó un proceso reflexivo entorno a la temática de la entrega (1), Valoración del dispositivo (4)
Muy interesante la propuesta	Valoración del dispositivo (4)
El contenido de la historia es importante para los niños, al hablar de plantas de naturaleza y de crecer en la vida,	No clasificables (7)
Muy lindas las canciones y el cuento.	Valoración de actividades concretas (5)

Se puede apreciar que la jerarquía de categorías diseñada por el INAU tiene cierta ambigüedad ya que existen sub clases complejas como "Valoración del dispositivo" y "Valoración de actividades concretas". Además, otras sub-categorías parecen confusas o poco intuitivas como "Sugerencias/Observaciones".

Además del conjunto inicial de datos, se nos brindó un conjunto de datos extra que corresponde a la salida capturada de los formularios de *Google* (datos crudos). Estos nuevos datos se entregaron avanzado el proyecto, y se usaron para complementar la exploración y el análisis de datos en la etapa final. El objetivo de obtener estos nuevos datos fue contar con nuevos atributos que nos posibilitaran profundizar el análisis, para así afinar las conclusiones del proyecto. El conjunto de datos tiene como atributos:

- Marca temporal
- Te resultó útil el contenido? En caso afirmativo, marque las opciones que reflejan la utilidad.

- Identidad de género
- ¿Querés comentarnos o sugerirnos algo? Nos encantaría leerte! (este atributo corresponde al campo "respuesta" del primer set de datos)

y se entregó en cuatro (4) documentos de *Google Sheets* que corresponden a las cuatro entregas que el INAU llevó a cabo durante el programa.

2.1. Etapas del proceso

El problema presentado por el INAU requiere de un conjunto de etapas e iteraciones que permitan explorar y analizar posibles caminos para obtener las respuestas buscadas. En este proyecto de ciencia de datos se ejecutan las siguientes etapas,

1. Obtener los datos que nos permitan dar respuesta al problema planteado.
2. Preprocesar los datos (corregir inconsistencias, rellenar valores que faltan, etc.).
3. Explorar y analizar los datos.
4. Construir un modelo de aprendizaje automático no supervisado.
5. Evaluar el modelo.

Una parte importante del proceso es –una vez planteado el problema– obtener los datos necesarios que nos permitan encontrar la respuestas que estamos buscando (o no). En este proyecto ya se cuenta con los datos dado que el INAU fue el encargado de recogerlos como parte de las encuestas del programa. Por lo tanto se enfoca en las tareas de la número 2 a la número 5 de los puntos anteriores.

En este trabajo se organizan las tareas teniendo en cuenta que se van a aplicar dos enfoques distintos; el primero es analizar nuestros datos usando el modelado de tópicos, y así poder visualizar las asociaciones entre estos, y sus características (como se forman, como se agrupan, etc.), y en el segundo enfoque se representa el texto usando SENTENCE EMBEDDINGS y se aplica un ALGORITMO DE CLUSTERING con el fin de encontrar y analizar las asociaciones que se dan entre las respuestas –de las encuestas– apoyándonos en el poder de las representaciones de *embeddings*[1].

[1]Se hace referencia a los SENTENCE EMBEDDINGS

A continuación se describen las tareas y los dos enfoques que se mencionaron anteriormente, y en la Figura 2.2 se muestra visualmente como estas interactúan,

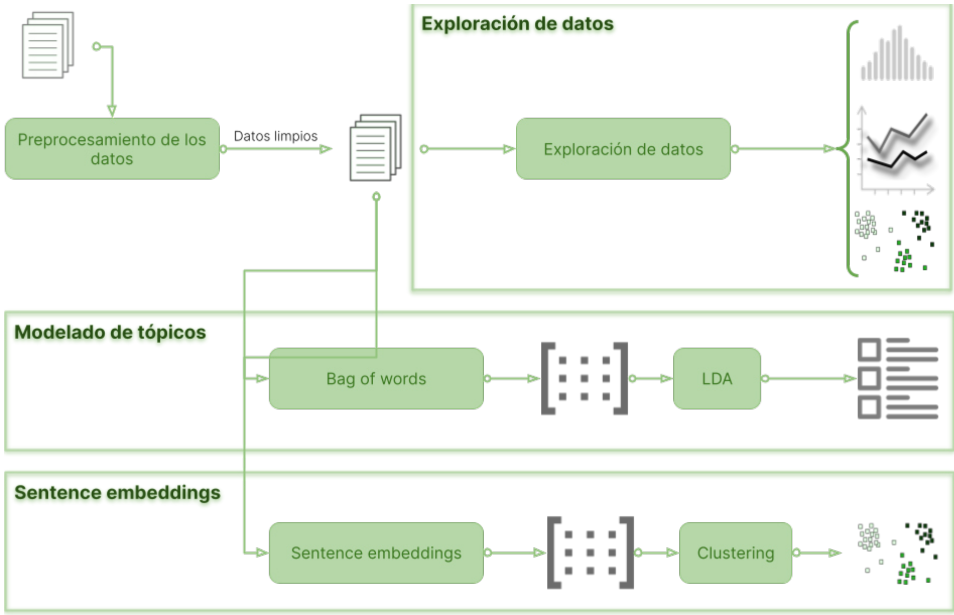


Figura 2.2: Diagrama que ilustra como se conectan las diferentes etapas del proceso.

1. **Preparar los datos.** En esta etapa se realiza un preprocesamiento de los datos para que puedan ser procesados desde los diferentes algoritmos y métodos de visualización.
2. **Explorar y analizar los datos.**

   a) Primer acercamiento. Se intenta determinar como se distribuyen las respuestas en las categorías previamente clasificadas (balance de clases), identificar caracteres especiales (como «emojis»), identificar palabras más comunes, así como otras tareas de identificación y exploración de los datos que permitan identificar patrones.

   b) Modelado de tópicos (4.2.1), Mediante el uso de modelos de Aprendizaje automático no supervisado (ver 3) se busca encontrar patrones no visibles en los datos. En particular, se intenta buscar en los textos cuales son los tópicos (conceptuales, ya que están formado por un conjunto de palabras) que nos permitan entender cómo se relacionan y agrupan las respuestas de los usuarios.

c) Agrupar las respuestas usando SENTENCE EMBEDDINGS (ver 3.3.1). De igual forma que el punto anterior, se usa otra técnica para explorar las relaciones entre los datos, así como patrones que puedan surgir de ellos.

3. **Evaluar y analizar los resultados obtenidos**. Finalmente, se evalúa el resultado de los puntos anteriores, se analiza las relaciones y patrones obtenidos. Además, se vincula estas relaciones con las categorías dadas, y se concluye si es posible responder o no al problema planteado.

No se incluye el proceso de evaluación, en nuestro caso la evaluación surge de la interpretación de los resultados para generar recomendaciones. El objetivo de estas etapas es entender mejor la naturaleza de los datos para responder si es posible automatizar la categorización de las respuestas en la jerarquía dada, o en caso de que no sea posible qué camino tomar. De esto pueden surgir interrogantes como, ¿son suficientes los datos que se tienen para construir un modelo predictivo (un clasificador por ejemplo)? ¿el problema como se plantea tiene una solución o se necesita revisar el problema y dar opciones alternativas?.

En el siguiente capítulo se introducen los conceptos teóricos usados en este proyecto. Luego, en el capítulo siguiente de **Diseño e implementación de la solución** se detalla cómo se abordó el problema aquí definido y se exploran las decisiones tomadas así como los enfoques que se usaron en las diferentes etapas.

Capítulo 3

Conceptos preliminares y Estado del arte

En esta capítulo se introducen los conceptos teóricos, y se comenta el estado del arte de las diferentes técnicas utilizadas.

Para comenzar se necesita hablar de Procesamiento del Lenguaje Natural (PLN) –también puede usarse Natural language processing (NLP) por su sigla en inglés– dado que es la columna vertebral de esta tesis. El PROCESAMIENTO DEL LENGUAJE NATURAL permite a las computadoras llevar a cabo tareas útiles que involucran al lenguaje humano. Tales tareas pueden mejorar las formas de comunicarnos con dispositivos (por ejemplo, con Siri en nuestros teléfonos, o el asistente de Google), mejorar la forma de comunicación entre personas (asistentes gramaticales en los procesadores de texto), o simplemente para cuando necesitamos procesar el lenguaje para obtener información de utilidad para cierta tarea que estemos emprendiendo (Jurafsky y Martin, 2009).

Procesar el lenguaje con el objetivo de resolver una tarea puntual requiere de cierto conocimiento del lenguaje que nos permita entender el contenido. Durante años se han desarrollado técnicas para procesar textos que nos permiten entender de qué tratan, resumirlos, obtener respuestas a preguntas, entre otras tareas. Generalmente, los procesos comienzan dividiendo (o segmentando) el texto en tiras de *tokens*[1] –se puede pensar como las palabras de la oración o del texto–. Posteriormente, sobre esta secuencia de *tokens* se aplican otras técnicas que nos permiten analizar sintácticamente los textos (la forma), así como semánticamente (el significado).

[1] El TOKEN se define como una secuencia de caracteres que están agrupados de tal forma que forman una unidad semántica usable para ser procesada.

Los *tokens*, para ciertas aplicaciones, necesitan ser transformados a una forma en especial para su procesamiento. Por ejemplo, hay ocasiones que en el texto que estamos analizando, nos puede interesar inspeccionar nuestra lista de *tokens* y encontrar las conjugaciones del verbo «correr» como *corro*, *corrí* y *corrieron*. En este contexto, la palabra «correr» se llama *lema*. En PLN existe un proceso para transformar los *tokens* a su respectivo *lema* y se conoce como LEMATIZACIÓN. Una forma más simple y menos compleja computacionalmente es la técnica de *stemming*, que mediante un algoritmo lleva la palabra a su raíz.

También, se recurre al uso de PART-OF-SPEECH TAGGING que es una técnica que permite etiquetar las palabras de una oración con la clase a la que pertenece (si es sustantivo, verbo, adjetivo, adverbio u otro tipo), esta clase se conoce como PART-OF-SPEECH (ó POS). El POS-TAGGING es una técnica de desambiguación ya que permite diferenciar palabras dado su contexto (por ejemplo, "presente" es un sustantivo, pero también es un verbo).

Por último, introducimos las técnicas de Aprendizaje Automático, que se usan de forma extensiva en el procesamiento del lenguaje natural. Como se puede ver más adelante en este trabajo se usan técnicas de Aprendizaje Automático no supervisado. Los algoritmos de aprendizaje automático no supervisado intentan encontrar las relaciones o patrones en los datos por sí solos (Wilmott, 2019). Para poder definir mejor estos algoritmos «no supervisados» deberíamos definir los «supervisados». Los algoritmos de aprendizaje automático supervisados necesitan ser guiados para que puedan encontrar los patrones en los datos. La forma de guiar estos algoritmos es mediante el uso de conjuntos de datos que hayan sido previamente revisados por un humano (o una máquina en ciertas ocasiones) y estos datos hayan sido «etiquetados». La entrada de estos algoritmos son conjuntos de datos donde sus elementos son tuplas $\langle x, y\rangle$ donde la x son los elementos de los cuales queremos «aprender un patrón» y la y es una etiqueta (o sea, un valor que un humano anteriormente etiquetó, ejemplo una clase o un número). Por ejemplo, un conjunto de fotos de animales: $\{\langle foto1, "perro"\rangle, \langle foto2, "gato"\rangle, \langle foto3, "caballo"\rangle, ...\}$ donde la $foto1$ es la foto de un perro, la $foto2$ es la foto de un gato, etc. (Wilmott, 2019). Estos algoritmos usan el conjunto de datos para aprender los patrones mediante un proceso que se llama «entrenar el algoritmo» (o *training* en inglés). Una vez que el proceso de entrenamiento se completa, se genera un «modelo» que se usa para responder a lo aprendido. Por ejemplo, predecir si la foto del perro es de un perro o de un gato, o predecir cual va a ser el valor de una casa en un año dado el barrio donde se encuentra.

Los algoritmos de aprendizaje automático no supervisado usan datos que no están etiquetados y buscan encontrar relaciones que existen entre los elementos del conjunto de datos (estas relaciones surgen de las características propias de esos datos). Dentro de este tipo de algoritmos se pueden encontrar (a) algoritmos de *clustering* (b) algoritmos para la detección de anomalías, y (c) enfoques para encontrar modelos de variable latente [1] (como el modelo de *Latent Dirichlet Allocation* o LDA).

A continuación describimos los conceptos teóricos comenzando con la técnica de «text clustering» ya que esta técnica se usa tanto en Modelado de tópicos como en representación con *Sentence embeddings*. Se sigue con el «modelado de tópicos» donde se trata el modelo de «Latent Dirichlet Allocation». Luego, se introduce técnicas de representación de texto para algoritmos de aprendizaje automático –con foco en las «Sentence embeddings»–, y se finaliza con técnicas de reducción de dimensionalidad y visualización de datos en múltiples dimensiones.

3.1. Text Clustering

Dentro de las técnicas de aprendizaje automático no-supervisado se encuentra lo que se conoce como técnicas de agrupamiento o *clustering*. Son algoritmos que nos permiten agrupar los datos usando características propias de ellos (por ejemplo, color, tamaño, etc.). El resultado del algoritmo son grupos de elementos de nuestro conjunto de datos que nos permiten identificar patrones, generalmente no visibles, que pueden ser relevantes o no para la tarea que estemos ejecutando. Estos grupos se denominan *clusters*, y es la denominación que vamos a usar en todo el documento.

Existen varios algoritmos de *clustering* pero los más conocidos son: *K-Means*, *Hierarchical clustering*, *Gaussian mixtures* y los basados en densidad como *Density-Based clustering based on Hierarchical density estimates* (más conocido como **DBScan**).

Para la agrupación de las respuestas, se usa una técnica basada en los algoritmos de densidad que se denomina **HDBScan**.

[1] Un modelo de variable latente es un modelo probabilístico de variables ocultas y observadas donde las variables ocultas codifican patrones escondidos en nuestros datos (Blei, 2014)

3.1.1. HDBScan

Este algoritmo de clustering fue desarrollado por Campello, Moulavi, y Sander que documentan su trabajo en el artículo «Density-Based Clustering Based on Hierarchical Density Estimates» (Campello et al. 2013). Es una extensión de *DBScan* pero usando elementos de agrupamiento jerarquizado (*hierarchical clustering*) y combinando técnicas de grafos para obtener los «clusters». El algoritmo HDBScan realiza los siguientes pasos,

1. **Se transforma el espacio de datos de acuerdo a la densidad o lejanía de los puntos**. La idea aquí es encontrar las «islas» en un «mar» de puntos. Una forma de identificar las «islas» o los «mares» es estimando la densidad de los puntos en el espacio de tal forma que las zonas con baja densidad son «mares» y las de alta densidad son «islas». Una premisa de este algoritmo es asumir que los datos tienen ruido (o sea, pueden existir datos corruptos o mediciones erróneas), entonces, dado que se usa un algoritmo (*single-linkage clustering*) sensible al ruido, la forma de estimar la densidad tiene que ser robusta. Se introducen dos métricas importantes, la distancia de un punto a su punto k-esimo más cercano (*kth nearest neighborhood*) denominada *core distance* que actúa como una medida económica de densidad y una nueva medida de distancia, denominada *mutual reachability distance*, que nos ayuda a separar a más distancia los puntos con baja densidad. Usando estas métricas, se prepara el conjunto de datos para usarse en el siguiente punto.

2. **Se construye el árbol de recubrimiento mínimo de acuerdo al grafo de distancia entre puntos**. Conceptualmente se arma el grafo usando como vértices los puntos, y la métrica de *mutual reachability distance* como peso de las aristas. Se usa el algoritmo de Prim para obtener el *árbol de recubrimiento mínimo*, como ejemplo en la Figura 3.1.

3. **Se construye una jerarquía de *clusters* de acuerdo a los componentes que están conectados en el grafo**. Ahora que tenemos el *árbol de recubrimiento mínimo* el paso siguiente es convertirlo en una jerarquía de componentes conectados. Gráficamente lo podemos visualizar en el dendograma [1] de la Figura 3.2. Para encontrar los *clusters* que se

[1]Un dendrograma es un tipo de representación gráfica o diagrama de datos en forma de

están buscando se puede trazar una linea en el dendograma tal que a ese nivel exista una densidad por *cluster* adecuada. Sin embargo, hacer esto nos desvía del espíritu original del algoritmo ya que lo que realmente se busca es usar la densidad como una medida para obtener las mejores agrupaciones de puntos, a un nivel más particular y menos general.

4. **Se condensa la jerarquía encontrada anteriormente usando como parámetro el tamaño menor de *cluster* especificado (*minimum cluster size*)**. Esto permite encontrar un árbol más pequeño y condensando, y usando el parámetro anterior nos permite identificar los grupos finales. En la Figura 3.3 podemos ver el diagrama de árbol condensando, así como los *clusters* ya identificados, que es el siguiente paso.

5. **Se extraen los *clusters* más estables del árbol condensado**. Se usa una medida de estabilidad para encontrar los *clusters* y además, intuitivamente se buscan *clusters* «chatos», o sea, *clusters* que en el dendograma no tengan *clusters* que desciendan de él.

Con esté último paso se identifican los *clusters* que el algoritmo encontró.

Los algoritmos de aprendizaje automático como el anterior (sea supervisado o no supervisado) van a necesitar trabajar sobre texto (en este caso, las respuestas de las encuestas). El texto necesita representarse de una forma que estos algoritmos lo puedan entender, y para esto recurrimos a usar una notación matemática que nos permita calcular y encontrar estas relaciones entre los datos. Se recurre al uso de vectores para representar el texto, y generalmente, un texto se representa como un conjunto de vectores donde cada vector es una palabra, o una oración ó un párrafo.

árbol

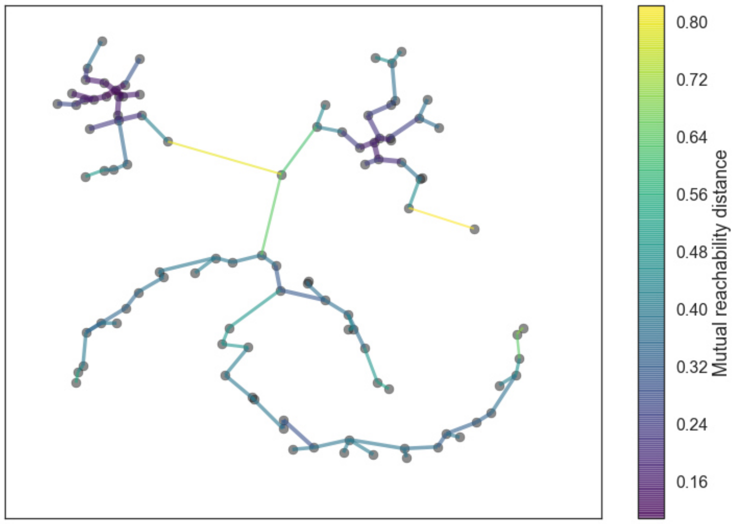


Figura 3.1: Ejemplo de árbol de recubrimiento mínimo para un set de datos. Fuente: https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

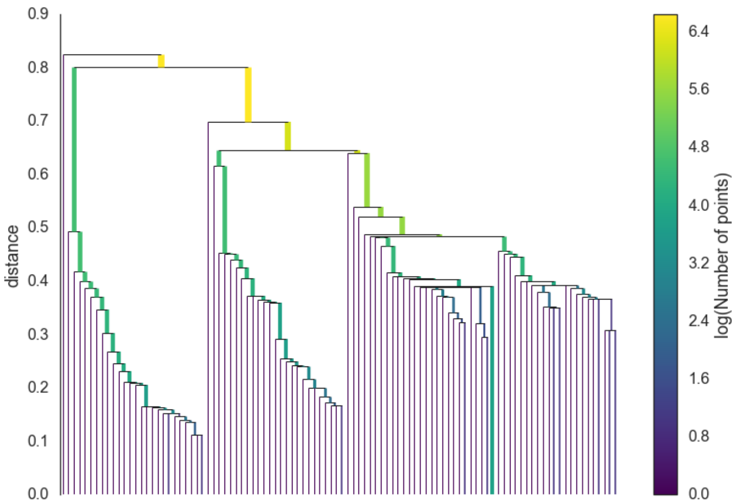


Figura 3.2: Dendograma de la jerarquía de clusters. Fuente: https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

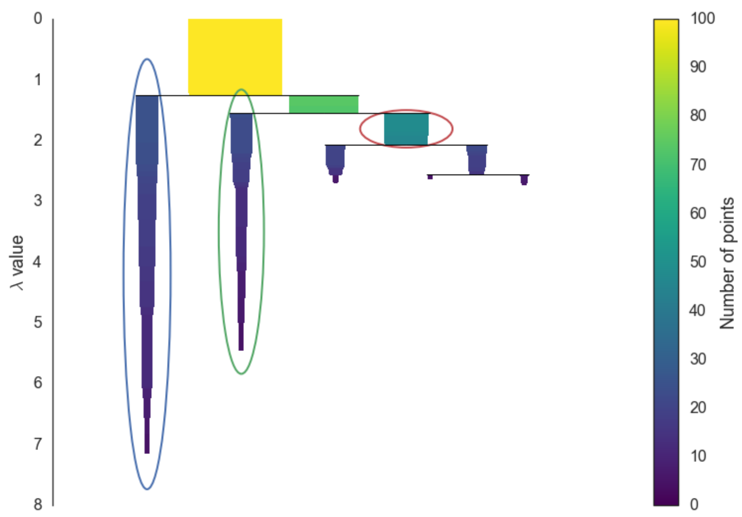


Figura 3.3: Árbol condensando y los clusters identificados. Fuente: https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

3.2. Modelado de tópicos

Cuando se necesita procesar grandes cantidades de texto las primeras preguntas que nos hacemos –normalmente– son ¿de qué tratan estos documentos? ¿cuál es su temática?. Leer la colección de textos de principio a fin puede resultar una tarea extensa y de mucha dedicación. En estas situaciones, se necesita disponer de una técnica que permita darnos una idea de las temáticas o tópicos que tratan los documentos de una forma eficiente. El modelado de tópicos nos permite organizar, entender y resumir grandes colecciones de texto. El modelo en cuestión es un modelo estadístico que nos permite extraer tópicos abstractos de la colección de textos. Una forma de minería de texto para detectar patrones de las ocurrencias de las palabras (Kamath et al. 2019).

En este trabajo se decide usar el modelo de Latent Dirichlet Allocation o *LDA* introducido por Blei et al. 2003. Este modelo es un modelo probabilístico también llamado «modelo probabilístico generativo». Se asume que nuestros datos surgen de un proceso generativo que incluye *variables ocultas*. Este proceso define una distribución de probabilidad conjunta (*joint probability distribution* por sus siglas en ingles) sobre las variables aleatorias observadas (esto son los documentos, el conjunto de textos) y las ocultas, y se calcula la distribución condicional de las variables ocultas dadas las observadas. Esta distribución se llama «distribución a posteriori». En el caso de LDA las variables observadas son las palabras de los documentos, y las variables ocultas es la estructura de los tópicos y el proceso generativo es el que describimos. El problema a resolver en LDA es calcular la *distribución a posteriori*, o sea, la probabilidad condicional de las variables ocultas (los tópicos) dados los documentos.

A modo de introducir el proceso generativo de LDA, simplemente para dar un marco teórico más detallado, la siguiente es la distribución conjunta de las variables observadas y ocultas,

$$\begin{aligned} p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) \\ = \prod_{i=1}^{K} p(\beta_i) \prod_{d=1}^{D} p(\theta_d) \left(\prod_{n=1}^{N} p(z_{d,n}|\theta_d) p(w_{d,n}|\beta_{1:K}; z_{d,n}) \right) \end{aligned}$$

donde,

- $\beta_{1:K}$ donde β_k es una distribución sobre el vocabulario (el conjunto de todas las palabras de todos los documentos)
- θ_d es la proporción de tópicos para el documento d-ésimo, donde $\theta_{d,k}$ es la proporción del tópico k en el documento d.
- La asignación de tópicos para el documento d-ésimo es z_d donde $z_{d,n}$ es la asignación de tópicos para la n-ésima palabra en el documento d.
- Las palabras observadas en el documento d son w_d donde $w_{d,n}$ es la n-ésima palabra en el documento d donde es un elemento del vocabulario (fijo).

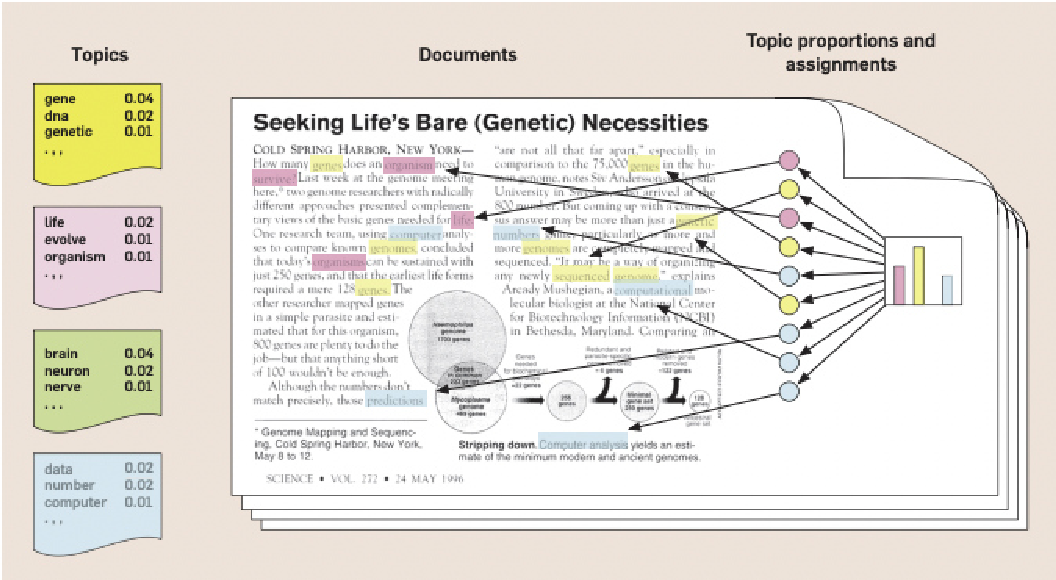


Figura 3.4: La intuición detrás del modelado de tópico.
Probabilistic topic models communications of the ACM, Blei, D.M. (2010)

Para terminar, se puede encontrar que el algoritmo de LDA funciona balanceando dos objetivos,

- En cada documento, el algoritmo asigna las palabras a un número reducido de tópicos.
- En cada tópico, el algoritmo asigna probabilidades grandes a un conjunto reducido de palabras.

El lugar donde el algoritmo encuentra el mejor resultado es en el equilibrio de ambos puntos, prestemos atención,

- Si asignamos un documento a 1 solo tópico, entonces hacemos que el punto 2 se haga muy complejo. Todas las palabras deberían tener una probabilidad en este tópico.
- Por el contrario, si asignamos pocas palabras por tópico hacemos muy difícil alcanzar el punto 1. Ya que para cubrir las palabras de un documento, tendríamos que asignar muchos tópicos a el.

Balanceando estos dos puntos, encontramos tópicos con alta cohesión de co-ocurrencia de palabras entre tópicos y documentos.

Si bien el modelo de LDA sirve para extraer los tópicos del conjunto de respuestas y explorarlas (para entender de que tratan), también este algoritmo nos brinda una representación para cada respuesta dada por su composición de tópicos (recordemos que LDA nos da una distribución de probabilidad de los tópicos del documento). En este trabajo, estas representaciones se usan para agrupar las respuestas e identificar relaciones de sus tópicos y las categorías dadas. Sin embargo, existen otras formas de representar el texto –posiblemente más poderosas– que se exploran a continuación.

3.3. Representación de texto

Los algoritmos de aprendizaje automático se apoyan en cálculos matemáticos para así obtener los patrones y poder tomar decisiones sobre los datos. Por este motivo estos algoritmos requieren que la entrada sea una representación numérica, generalmente se usan vectores o matrices (tensores[1] si queremos ser más generales). Entonces, es necesario, codificar nuestros datos de entrada (textos, imágenes, audio, etc.) en vectores.

Se han desarrollado varias técnicas para representar texto como un vector numérico, desde el modelo de *Bag of words* (siendo uno de los primeros) hasta la actualidad usando *Distributed representations*, como *Word embeddings* ó *Sentence embeddings*.

Para introducir correctamente los conceptos que siguen, se necesita dar dos definiciones importantes:

- CORPUS –del latín cuerpo–, refiere a una colección de textos o documentos textuales. En este contexto, un «documento» es una respuesta de los usuarios, y el «corpus» sería el conjunto de todas las respuestas.

- VOCABULARIO, es el conjunto de todas las palabras (únicas) que aparecen en todo el corpus (en este caso, en todas las respuestas).

Se describe a continuación el modelo de *Bag of words* que es sencillo en su implementación y es claro para enmarcar el resto de la sección, y así explicar mejor los modelos siguientes.

El modelo de BAG OF WORDS codifica el texto como un vector de $0s$ y $1s$ de largo fijo –la cardinalidad del vocabulario– (Jurafsky y Martin, 2009) y se coloca un 1 si la palabra del vocabulario pertenece al texto (a la respuesta en

[1] Un tensor es una generalización de vectores y matrices y es entendido como vector multidimensional.

este caso) o un 0 si no. En la Figura 3.5 se da un ejemplo de como funciona el modelo de *Bag of words* (está en inglés, pero de igual aplicación en español).

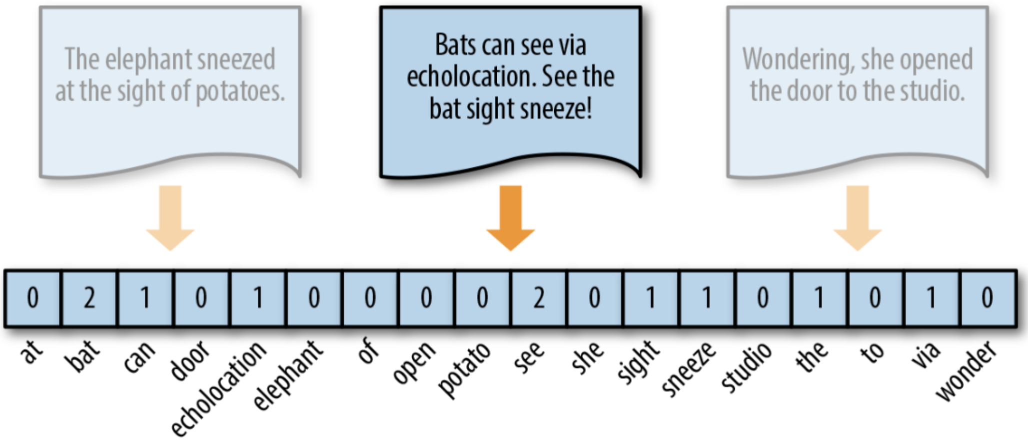


Figura 3.5: Ejemplo del modelo *Bag of Words*. Imagén del libro Applied Text Analysis with Python. Bengfort et al. 2018

Este modelo se lo conoce como el modelo de espacio vectorial (*Vector Space Model* o *VSM*) ya que representa los documentos del corpus como puntos o vectores en un espacio vectorial, de tal forma que podemos usar propiedades matemáticas de los Espacios vectoriales –como la distancia entre puntos– para calcular la similitud entre dos textos. Podemos ver que si tenemos dos vectores correspondientes a documentos de nuestro corpus, y la distancia entre ellos nos muestra que están muy cerca, entonces su contenido debería ser el mismo o muy parecido, de lo contrario, son documentos que no se parecen.

Sin embargo, el modelo de *Bag of words* tiene varias limitaciones. Una de ellas es que los vectores generados por este modelo tienen una alta dimensionalidad (recordar que el largo de los vectores es el tamaño del vocabulario, generalmente de miles de palabras). Además, muchas de sus entradas son nulas o “0” lo cual genera que los operaciones entre ellos tengan un elevado coste computacional (este problema se conoce como la «maldición de las dimensiones» o por su nombre en inglés, Curse of dimensionality la cual prefiero para su uso de aquí en más). Otra limitación de este modelo es que no mantiene el orden de las palabras del texto al codificarlo como un vector (perdiendo así información del contexto). Por ejemplo, para el modelo de *Bag of Words* la oración «Los rascacielos de Nueva York» y la oración «York rascacielos Nueva Los» generan el mismo vector pero claramente no son iguales (además de la segunda no tener sentido).

Se han desarrollado otros modelos para mejorar las limitaciones del anterior como el modelo de Tf-Idf (Term frequency-Inverse document frequency) que evoluciona la representación vectorial, teniendo en cuenta las frecuencias de las palabras en los documentos, y en el corpus.

En los últimos años, con el resurgimiento de las Redes Neuronales[1] y el Aprendizaje profundo[2] han aparecido un conjunto de técnicas para representar texto conocidas como Representaciones distribuidas o *Distributed representations* siendo una de las más conocidas la que se conoce como *Word embeddings*.

Los Word embeddings son representaciones vectoriales de las palabras de un texto, que a diferencia de las representaciones anteriores, producen vectores de menor largo y más densos (generalmente de 200 a 500 elementos). Este nuevo modelo surgió como un efecto secundario de un trabajo de unos investigadores en Google (Mikolov, Chen et al. 2013 y Mikolov, Sutskever et al. 2013) que usaban redes neuronales para entrenar un modelo que predecía las palabras del contexto. Este modelo introdujo un conjunto de nuevas características que generaron una rápida adopción de estas técnicas y una aceleración en el área de representaciones de texto para el Procesamiento del Lenguaje Natural.

Las características que se destacan son (a) reduce la dimensionalidad de los vectores y por ende ataca el problema de *curse of dimensionality* (b) los vectores generados contienen información sobre el contexto de la palabra, pudiendo utilizarse operaciones de vectores sobre los mismos, por ejemplo, $Vector(Reina) + Vector(Hombre) = Vector(Rey)$ (c) se pueden combinar estos vectores –mediante concatenación u otras operaciones como suma o promedio– para representar oraciones o textos más largo (Jurafsky y Martin, 2009).

3.3.1. Sentence Embeddings

Las *Sentence Embeddings* son una representación vectorial de una oración en un espacio de vectores reales. Estas surgen como una extensión razonable de las *Word Embeddings* (Mikolov, Chen et al. 2013) –que inicialmente fueron propuestas con el nombre de *Paragraph Vectors*– inmediatamente después por

[1]Las redes neuronales son un método de aprendizaje automático que toma su diseño y nombre de un modelo de neurona de nuestro cerebro, Jurafsky y Martin, 2009.

[2]Se refiere al uso de redes neuronales para el aprendizaje automático pero profundas, o sea, con más capas, Jurafsky y Martin, 2009.

el mismo autor (Mikolov, Sutskever et al. 2013).

Actualmente, si queremos representar una oración como un *embedding* existen dos formas,

- usando los *Word embeddings* de las palabras que forman la oración y agregando una operación de suma, promedio o concatenación, para poder obtener un vector único, ó
- usando *Sentence embeddings* en alguna de sus diferentes implementaciones.

Al momento de escribir este documento, las implementaciones más populares de *Sentence embeddings* son,

- **InferSent** de Facebook, usando como tarea la inferencia del lenguaje y que produce *embeddings* de dimensión 4096 (Conneau et al. 2017)
- **Universal Sentence Encoder** de Google, que usan transformers y transfer learning, produciendo *embeddings* de dimensión 512 (Cer et al. 2018).
- **SentBERT** de la Technische Universitat at Darmstadt, que toma el modelo BERT y lo combina con redes siamesas produciendo *embeddings* de dimensión 300 (Reimers y Gurevych, 2019).

En este trabajo se decide usar **SentBERT** por dos razones (a) la publicación está acompañada de una biblioteca con su correspondiente documentación que hace muy sencillo el uso (b) tiene un uso más extendido y existen mayores recursos disponibles respecto a su uso respecto a las otras implementaciones.

3.3.2. SentBERT: Sentence Embeddings usando redes BERT y siamesas

Uno de los objetivos que se busca es encontrar oraciones que son semánticamente parecidas (Reimers y Gurevych, 2019). Con esto nos referimos a oraciones que hablen de lo mismo, por ejemplo, las oraciones «El perro de Catalina es muy bonito» y «Catalina tiene un perro que es realmente precioso» son oraciones semánticamente parecidas (están hablando de lo mismo). En particular, cuando hablamos de «semánticamente parecidas» en el contexto de la representación vectorial nos referimos a vectores que estén "cerca" en el espacio.

Para poder introducir correctamente *SentBERT* necesitamos definir tres conceptos importantes,

- **Transformers**, es una nueva arquitectura de redes neuronales profundas que descienden de un tipo de redes neuronales llamado Redes Neuronales Recurrentes[1] (RNN por su sigla en inglés). El concepto fundamental en los *Transformers* es lo que se llama *Attention*, este mecanismo de «atención» (Vaswani et al. 2017) permite capturar información de contexto cuando se procesa la secuencia de texto, y actúa como una memoria que recuerda que es lo importante en una secuencia (para luego procesarlo en la tarea que sea).

- **Bidirectional Encoder Representations from Transformers (BERT)**, es un modelo pre-entrenado desarrollado por Google AI (Devlin et al. 2018). La idea detrás de este modelo (y de los *transformers*) es que se puede reusar partes del modelo y acoplar a nuevas tareas. Google (y empresas de este porte) entrenaron un modelo basado en *Transformers* con billones de datos, y publicaron los modelos pre-entrenados para que otros lo usen y sigan agregando mejoras.

- **Redes neuronales siamesas** consiste en 2 redes (gemelas) que aceptan entradas distintas pero que la salida de ambas se une usando una función especifica. La red es simétrica y los parámetros de la red son compartidos (Koch et al. 2015).

Ahora definido los conceptos, podemos introducir *SentBERT*. *SentBERT* es un modelo que utiliza la arquitectura de *Transformers*, en particular las redes denominadas Bidirectional Encoder Representations from Transformers (BERT) pre-entrenadas y modificadas con redes siamesas.

SentBERT también es pre-entrenado como *BERT* pero para entrenar el modelo usa una tarea que se basa en la similitud de oraciones. BERT puede hacer la misma tarea, pero es más complejo y requiere más tiempo y más recursos.

[1]Las redes neuronales recurrentes son un tipo de red que permiten conexiones cíclicas entre salidas de una neurona a ella misma o alguna otra, fueron originalmente introducidas en el trabajo de Rumelhart et al. 1986 y posteriormente evolucionadas en el trabajo de Schmidhuber, Hochreiter et al. 1997

A continuación se muestran dos figuras que intentan dar una intuición de la arquitectura de *SentBERT*,

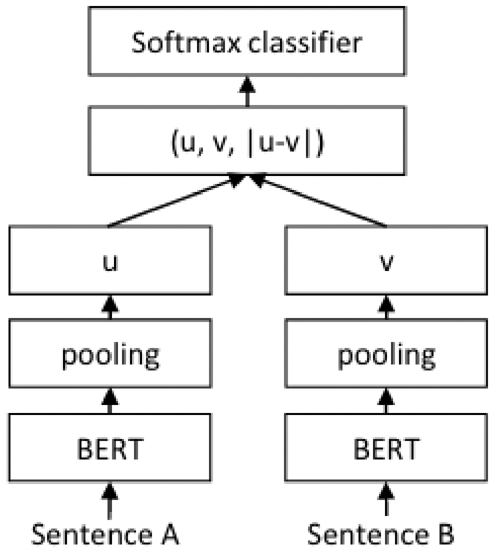


Figura 3.6: Arquitectura de SentBERT (con pesos compartidos entre las dos redes)

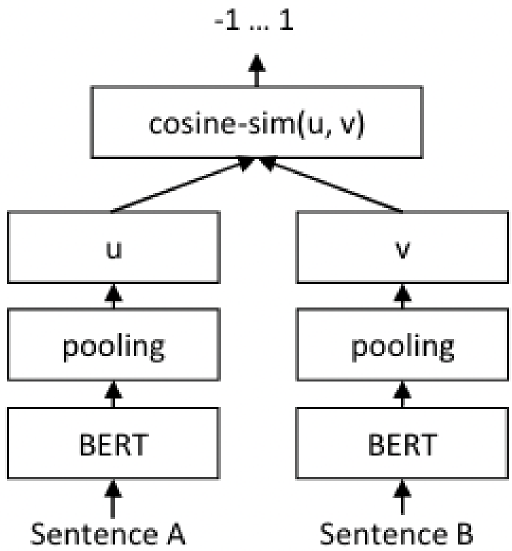


Figura 3.7: Arquitectura de SentBERT en el momento de inferencia.

En resumen, *SentBERT* codifica oraciones en vectores de largo 384 elementos y este modelo provee modelos pre-entrenados (*pre-trained models* en inglés) para varias tareas y multilenguaje. Los modelos pre-entrenados para mutiples lenguajes usan los beneficios del modelo de Transformers y la intuición por detrás es que las oraciones codificadas en diferentes lenguajes deben estar cerca en el espacio de vectores. O sea, la oración «Me gusta el helado» y «I love icecream» deberían estar muy cerca en el espacio de vectores. Dada estás características se elige el modelo por la conveniencia de su uso para las respuestas de las encuestas.

3.4. Reducción de dimensionalidad y Visualización

Parte del trabajo de exploración de datos es visualizar los datos para identificar patrones que a veces no se ven de otra forma. Un buen soporte visual es lo que permite al usuario observador encontrar más y mejores ideas en el menor tiempo posible, citando a Tufte, 2001.

Por este motivo se eligen un conjunto de técnicas que nos permiten condensar la información para visualizarla de mejor forma en un espacio de dos dimensiones.

3.4.1. t-SNE y UMAP

Una de las características –que ya hemos mencionado– de representar texto es que las representaciones son normalmente de alta dimensionalidad, o sea, los vectores son de 200 o más dimensiones. Este problema también aparece en otros tipos de datos como por ejemplo, la representación de imágenes ó la representación de nucleótidos celulares para el estudio de detección de cáncer.

Para poder visualizar estos datos que tienen alta dimensionalidad necesitamos reducir la dimensión de los vectores a 2D o 3D porque no se tiene forma de visualizar datos en más dimensiones. Las técnicas de reducción de dimensionalidad convierten conjuntos de datos de la forma $\mathcal{X} = \{x_1, x_2, x_3, ..., x_n\}$ en conjuntos de 2 a 3-dimensional $\mathcal{Y} = \{y_1, y_2, y_3, ..., y_n\}$ que pueden ser visualizados en una gráfica típica de puntos. En la nomenclatura de *t-SNE* se le llama mapa al conjunto $\mathcal{Y}$, y a los puntos de baja dimensionalidad y_i se les llama puntos del mapa.

Uno de los métodos usados para explorar datos con alta dimensionalidad es T-SNE (de sus iniciales en ingles *t-Distributed Stochastic Neighbor Embedding*, que fue introducido por Van der Maaten y Hinton, 2008. Recientemente, se ha convertido en una de las técnicas más usadas en el campo del aprendizaje automático ya que tiene capacidades muy interesantes para crear mapas 2-dimensionales de datos de cientos a miles de dimensiones (aunque puedan parecer impresionantes estos mapas pueden ser leídos incorrectamente).

La técnica de «reducción de la dimensionalidad» tiene como objetivo transformar los datos manteniendo las estructuras más significativas en su dimensión original, a un mapa de menor dimensionalidad. Tradicionalmente estas técnicas (como PCA, *Principal Component Analysis*) son técnicas lineales que se enfocan en distanciar los puntos que no son similares, sin embargo, para el caso de alta dimensionalidad donde se asume que existe cierta estructura presente en dimensiones más bajas las técnicas no-lineales dan mejores resultados. Generalmente, estás técnicas se enfocan en preservar (a) la distancia, (b) la topología, y/ó (c) la información.

t-SNE es capaz de capturar la estructura local de los datos multidimensionales mientras que además revelan estructuras a nivel global, como agrupaciones de puntos en varios niveles. *t-SNE* preserva la distancia de puntos pero a su vez intenta preservar la estructura topológica [1]. Este algoritmo está basado en un algoritmo anterior llamado SNE (*Stochastic Neighbor Embedding*) que convierte distancias euclidianas en similitudes (que pueden interpretarse como probabilidades) respecto a los puntos del conjunto de datos, y se calculan las distribuciones de cada «vecindad». Estas distribuciones se comparan usando la medida de divergencia de «Kullback-Leiber» y el problema se reduce a minimizar esta medida de divergencia. El algoritmo de *t-SNE* se enfoca en la geometría local (por lo que se decía de preservar la estructura topológica) y en modelar el concepto de cercano con probabilidades (el concepto de "cerca" en un espacio multidimensional puede ser ficticio, por lo tanto, se usan probabilidades dada la falta de certezas). Este algoritmo tiene un hiper-parámetro que es la perplejidad (se usa como estimador de la varianza). La perplejidad puede ser interpretada como una medida «suave» de la cantidad efectiva de vecinos que puede tener un punto. Sin embargo, un desafío de *t-SNE* es encontrar el mejor valor del hiper-parámetro de perplejidad para los datos que se están procesando como se explora en Wattenberg et al. 2016.

Durante la exploración de datos y el modelado de tópicos se uso *t-SNE*, y para la segunda parte de *Clustering* usando *Sentence embeddings* se uso

[1] Intuitivamente es la forma un conjunto (subconjunto)

UMAP.

UMAP es una nueva técnica por McInnes et al. 2018 que ofrece un conjunto de mejoras respecto a *t-SNE*, en particular, mejora la performance –reduce el tiempo de cómputo– para volúmenes de datos grandes, y mejora la fidelidad al preservar la estructura global de los datos.

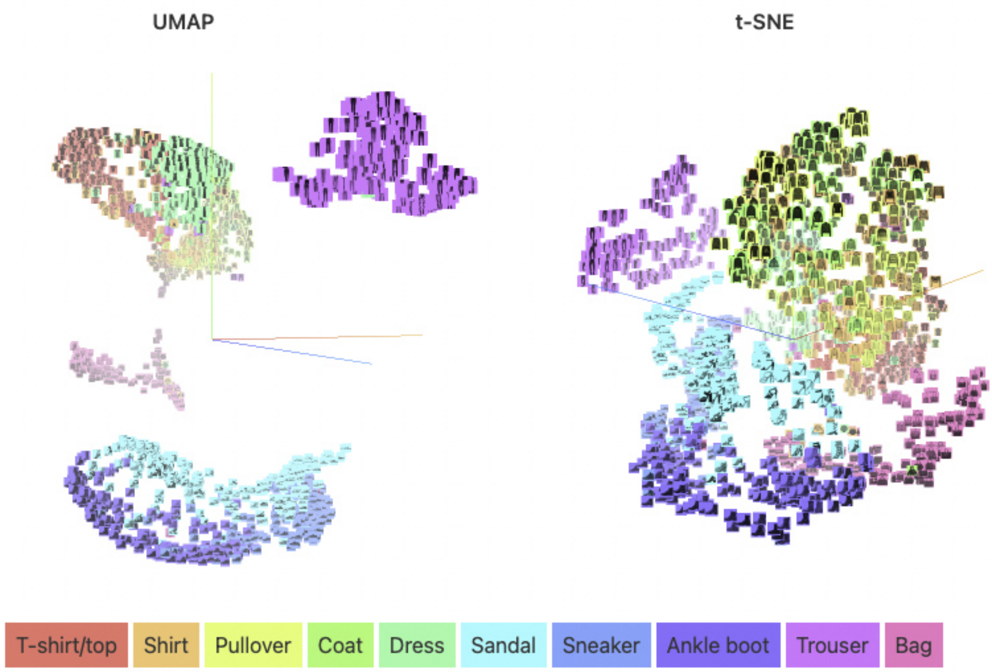


Figura 3.8: Comparación de una UMAP vs t-SNE para un conjunto de datos denominado Fashion MNIST. Fuente: Google PAIR

Ambos algoritmos generan agrupaciones que tienen sentido, sin embargo, UMAP separa los grupos de mejor forma haciendo más fácil la detección de categorías.

La intuición detrás de UMAP es similar a la de t-SNE ya que ambos se apoyan en buscar grafos en el espacio multi-dimensional para plasmarlos en un espacio menor (2 a 3 dimensiones). La matemática detrás de UMAP es compleja pero la idea detrás del algoritmo es muy simple: se busca una estructura de grafo y se optimiza hasta encontrar una representación que se asemeje lo más posible a la representación original.

UMAP construye el grafo inicial en el espacio altamente dimensional usando lo que se llama *Fuzzy simplicial complex*, o un grafo con pesos donde los pesos de las aristas es la probabilidad de que esos puntos estén conectados. Para determinar si están conectados se usa el radio con respecto al punto y se conectan cuando hay sobreposición de 2 radios. La elección del radio es crítica dado que sí es muy pequeño genera muchos *clusters* individuales, y en caso contrario, puede generar un solo *cluster* (muy grande) que conecta todos los puntos. Este problema se soluciona eligiendo el radio de forma local usando la distancia de cada punto a su vecino más cercano. Luego, se realizan un conjunto de correcciones para asegurar que la estructura local se preserve en un buen balance con la estructura global. Una vez construido la estructura de grafo en el modelo multidimensional, se realizan otra serie de optimizaciones (similares a lo que usa t-SNE) para llegar a un grafo en un espacio de menos dimensiones.

Es importante conocer los parámetros que necesita *UMAP* (recordemos qué *t-SNE* usaba la perplejidad `perplexity`). Los más importantes son `n_neighbors` y `min_dist`. `n_neighbors` –cantidad de vecinos cercanos– controla la estructura de grafo inicial, más pequeño el valor favorece a estructuras locales, y más grande, va a favorecer la estructura global. `min_dist` –distancia minima entre puntos– controla que tan apretado se pueden dar las representaciones, a menor valor genera representaciones más apretadas, a más grandes permite representaciones más separadas.

El algoritmo de *UMAP* tiene varias mejores sobre *t-SNE*, sin embargo, no es la panacea. Se tiene que tener en cuenta algunas consideraciones para interpretar mejor los resultados generados por *UMAP*, que se detallan a continuación,

- Los hiperparámetros importan, por ende es necesario probar con varias iteraciones y varias particiones de datos para obtener los de mejor resultado.

- El tamaño de los *clusters* no significa nada.

- La distancia entre los *clusters* no significa nada.

- Se necesitan más de una visualización, dado la naturaleza estocástica del algoritmo, distintas corridas nos van a generar resultados distintos. Es bueno compararlos y quedarnos con más de una visualización para contrastar y presentar resultados.

3.5. Detección de palabras relevantes

En la tarea de visualización de los datos se necesito encontrar las palabras más relevantes de cada *cluster* para poder visualizarlas como parte del gráfico, y así entender mejor los resultados. Se probaron un par de técnicas simples, como usar la frecuencia de las palabras, o la frecuencia más una normalización. Sin embargo, se decidió usar una técnica de extracción de palabras no supervisada llamada *YAKE!*.

YAKE! (Campos et al. 2020) es un algoritmo de extracción de palabras claves no-supervisado que utiliza las características estadísticas de las palabras en el texto para identificar y ordenar las palabras dado un cierto criterio de relevancia. El algoritmo tiene 6 etapas (a) preprocesamiento del texto, (b) extracción de características, (c) puntuación individual por palabras (d) generación de palabras candidatas (e) eliminación de duplicados y (f) ordenar por relevancia las palabras finales. Estas etapas se ejecutan sobre el texto a medida que se va procesando (de forma ≪*online*≫, lo cual lo hace útil cuando queremos una rápida respuesta.

Capítulo 4

Diseño e implementación de la solución

En este capítulo se detallan las diferentes etapas, y tareas que se llevaron acabo durante el desarrollo del proyecto de tesis; desde los primeros pasos dados en la etapa de exploración de datos, hasta los análisis finales que nos permitieron dar respuestas a algunas de las interrogantes que se plantearon (en la Figura 4.1 se pueden ver un diagrama de tareas y flujos). Se empieza con la preparación de los datos ya que es una etapa fundamental para poder obtener buenos resultados con los algoritmos de aprendizaje automático.

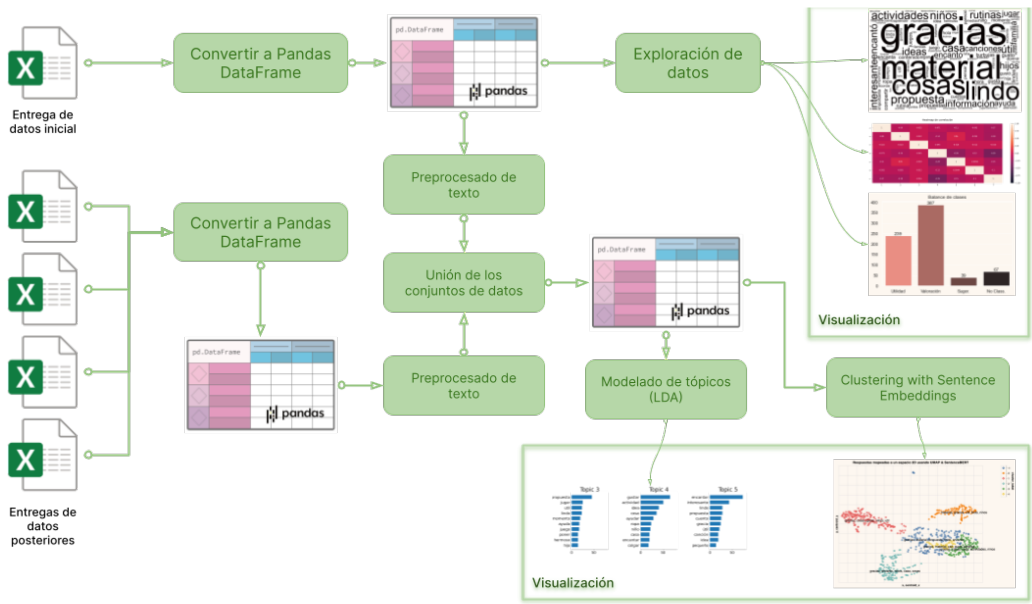


Figura 4.1: Diagrama de la solución, desde los datos «raw» hasta la etapa de su visualización

4.1. Preparación de los datos

En esta etapa es importante transformar los datos en un formato que sea fácilmente procesable para ser usados en las diferentes tareas que vamos a realizar tanto de análisis como de visualización.

Los datos fueron capturados en una planilla de *Microsoft Excel* y se brindaron siguiendo el formato que se muestra en la Figura 2.1. El archivo tiene cuatro hojas de cálculo y cada hoja representa una de las 4 entregas que el INAU ejecutó durante su programa.

Se optó por usar una biblioteca para *Python* llamada *openpyxl* para leer el archivo de *Excel* y así poder procesar las respuestas. Se elige guardar los datos en un *DataFrame*[1] –de la biblioteca de manipulación de datos *Pandas* (McKinney, 2010 y pandas development team, 2020)– de tal forma que nos facilite su procesamiento y manipulación.

4.1.1. Conversión Excel a Pandas

En el primer paso, se leen los datos del archivo *Excel* a un *DataFrame* de *Pandas*, donde se mapea el texto de las respuestas en una columna «respuesta», y las categorías se transforman en columnas numéricas que siguen este orden,

1. Impulsó un proceso reflexivo entorno a la temática de la entrega
2. Manifiestan haber realizado alguna actividad o su intención de hacerla
3. Manifiestan que no le resultó útil
4. Valoración del dispositivo
5. Valoración de actividades concretas
6. Sugerencias/observaciones
7. No clasificables

.

[1] Los DataFrame es una estructura de datos para guardar los datos en memoria de forma tabular, que además nos permite operar sobre ellos con varias operaciones

	respuesta	1	2	3	4	5	6	7
0	Excelente	0	0	0	4	0	0	0
1	Muy buena la información	0	0	0	4	0	0	0
2	Ecxelente material	0	0	0	4	0	0	0
3	Está actividades siempre las realizamos en cas...	0	2	0	0	5	0	0
4	Me gusto mucho la propuesta,ya q hay cosas q m...	1	0	0	4	5	0	0

Figura 4.2: DataFrame de Pandas con los datos del Excel luego de haber sido procesados.

4.1.2. Tildes y errores tipográficos

Las palabras se suelen representar como un secuencia de caracteres (letras, números, puntuación) y las computadoras suelen usar dos caracteres distintos para una letra con o sin tilde. Esto, puede generar un problema cuando se representan las palabras como vectores (como se mencionó en el capítulo anterior) ya que una palabra con tilde y una palabra sin tilde generan distintos vectores. Si bien en ocasiones este es el resultado esperado, ya que por ejemplo «papa» y «papá» son palabras distintas, en nuestro caso puntual es beneficioso. Puntualmente, se analizaron las respuestas y se notó que muchas de las respuestas fueron escritas en dispositivos móviles que cuentan con autocorrectores, generando así palabras sin tildes además de errores tipográficos. Otro motivo que llevó a la eliminación de los tildes fue su poco uso en el lenguaje de chat ya que se prefiere la velocidad más que la calidad, como se describe en el trabajo de Martínez, 2016 «La escritura de los chats equivale a una conversación: los textos escritos se convierten en textos orales, las conversaciones se transcriben y las normas lingüísticas se rompen.». Por lo tanto, se toma la decisión de eliminar los tildes de los textos de las respuestas.

Además, se trabajó en buscar todas las palabras de nuestro corpus que fueran «Out of Vocabulary» (OOV) –son palabras que no pertenecen a un vocabulario conocido o diccionario, generalmente, pueden ser errores tipográficos o palabras específicas a un dominio– y se analizan de forma manual para determinar si es necesario corregirlas o no. Las palabras *OOV* se identifican usando la biblioteca *Spacy* (Honnibal y Montani, 2017) que nos provee un método para chequear si una palabra es parte o no de un vocabulario. Para un conjunto de ellas se procede a corregirlas como se puede ver en la Tabla 4.1.

Tabla 4.1: Algunos términos encontrados en las respuestas, y su correspondiente mapeo (o reemplazo).

Termino	**Reemplazo**
xq, Xq, pq	porque
q, Q	que
xa	para
d	de
bb	bebé
flia	familia
abia	había
aserlo	hacerlo

4.1.3. Enriqueciendo los datos con nuevos datos de las encuestas

Se mencionó en el capítulo 2 que una vez comenzado el proyecto se obtuvo un conjunto de datos que disponía de nuevos atributos como el género, si la encuesta fue útil o no, y algunas reflexiones sobre una lista dada. Si bien este nuevo conjunto de datos se recibe casi terminado el modelado de los tópicos con LDA, y comenzado el trabajo con el enfoque de *Sentence embeddings* se decide incorporar esos nuevos atributos.

Se incorporan los nuevos datos realizando un procedimiento similar al descripto anteriormente donde se convierten los datos de archivos de *Excel* a objetos *Dataframe* de *Pandas* con el objetivo de una manipulación más sencilla. Se hacen conversiones de tipo de datos como texto "Si/No" a columnas del tipo «booleanas» (Si a "1" No a "0") ó «númericas» para una mejor manipulación por parte de los algoritmos usados.

Además, se conecta este nuevo conjunto de datos con el conjunto inicial usando la columna de «respuesta» como el campo de unión, enriqueciendo el conjunto inicial con los nuevos atributos.

4.2. Exploración y análisis de datos

La exploración de los datos se centra en entender mejor la naturaleza de los datos con los cuales vamos a trabajar. Entenderlos nos habilita a determinar si con los datos que tenemos podemos responder a las preguntas, y además, nos permite direccionar los procesos posteriores de análisis y construcción de modelos.

Se comienza cuantificando los datos que se tienen: (a) cantidad total, (b) cantidad de datos clasificados (c) cantidad de datos sin clasificar (d) balance de las clases (e) correlaciones entre los datos; así como otras cuestiones similares.

Para esta tarea, se usan un conjunto de herramientas que se detallan a continuación: el lenguaje de programación `Python`, documentos computacionales donde se puede ejectuar código llamados `Jupyter Notebooks`, bibliotecas de visualización de datos como `matplotlib`, `seaborn`, `altair` y, como mencionamos antes, la biblioteca de manipulación de datos `Pandas`.

4.2.1. Modelado de tópicos

El modelado de tópicos es el siguiente paso en nuestro proceso de análisis de datos. El objetivo es entender de qué tratan las respuestas extrayendo sus tópicos más relevantes. Los tópicos –como se detalla en la sección 3.2– constituyen una distribución de probabilidad sobre el conjunto de palabras del vocabulario que puede no tener un sentido lógico o natural, pero, que evaluando las palabras y los documentos de ese tópico permite deducir un nombre más representativo.

Está técnica se usa de dos formas en este trabajo,

- para encontrar los tópicos que se deducen de las respuestas, y así poder organizarlas y entenderlas

- para obtener una representación de un documento, y luego agruparlas con un algoritmo de *clustering*, y buscar otros patrones. La representación surge porque generalmente la distribución de tópicos es un vector (ver Figura 4.3), que representa a un documento en el espacio de vectores (cada tópico tiene una probabilidad, y a su vez, cada tópico se representa como una distribución de probabilidad sobre el conjunto de palabras del vocabulario, Figura 4.4). A modo de ejemplo, en la Figura 4.3 se representa el vector para el documento «Muy interesante la propuesta» y como se ve la mayor probabilidad se asigna al `tópico 5`. Si ahora exploramos la distribución de palabras que contiene este tópico (Fig. 4.4) vemos que aparecen dos de las palabras del documento: “interesante” y “propuesta”.

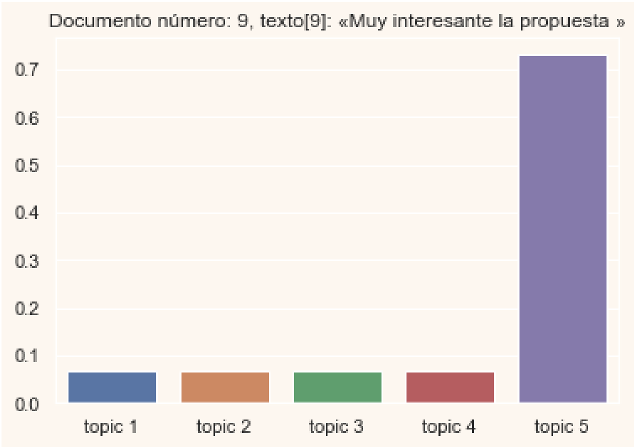


Figura 4.3: Representación del documento número 9 de nuestro corups como una distribución de tópicos.

Figura 4.4: Representación de cada tópico como una distribución de palabras de nuestro corpus con su correspondiente probabilidad.

En esta tarea se decide usar la biblioteca `scikit-learn`, en particular se usa el módulo de `CountVectorizer` para pasar los textos a vectores del tipo *Bag of Words* (ya sea contando solo la frequencia o usando *TF-IDF*).

Para la aplicación de LDA (3.2) se usa el módulo `sklearn.decomposition.LatentDirichletAllocation`. Puntualmente para LDA se probaron dos bibliotecas, la de `scikit-learn` y además la de `gensim`. Ambas bibliotecas usan una implementación más eficiente de LDA que se basa en el paper ≪Online Learning for Latent Dirichlet Allocation≫ de Hoffman et al. 2010, la cual permite al algoritmo ser entrenando en etapas, y con conjuntos de datos pequeños. Esto permite entrenar grandes conjuntos de datos de forma eficiente. La razón de usar `scikit-learn` por sobre `gensim` fue porque la última no tiene forma de brindarle como parámetro la semilla aleatoria para poder replicar las pruebas y corroborar los resultados. Asimismo, se usan las técnicas de visualización de datos de `t-SNE` (detalladas en 3.4.1) así como una herramienta específica para ver los tópicos de LDA llamada `pyLDAvis`.

También se usa la biblioteca hdbscan para la agrupación de los documentos (representados como vectores de tópicos).

El modelo de *LDA* se debe entrenar sobre un conjunto de datos, y además, se necesita encontrar los mejores valores de los hiperparámetros para el problema específico. Para encontrar los mejores hiperparámetros se usa el método de GridSearch[1] que prueba combinaciones de los hiperparámetros sobre un conjunto reducido de los datos.

Los hiperparámetros que se prueban son:

- n_components (entre 5 y 20),
- topic_word_prior (entre 1/5 y 0.5),
- doc_topic_prior (entre 1/5 y 0.5),
- learning_decay (entre 0.5 y 0.9), y
- learning_offset (entre 1 y 256)

El mejor modelo que se encontró se entrenó usando los hiperparámetros de la Tabla 4.2.

Tabla 4.2: Mejores hiperparámetros para el modelo de LDA.

Hiperparámetro	Valor
n_components	5
topic_word_prior	0.5
doc_word_prior	0.5
learning_decay	0.5
learning_offset	1

El último paso es usar estos valores hallados y variar el número máximo de iteraciones para obtener el mejor modelo posible. Se elige como medida de evaluación la PERPLEJIDAD (a menor perplejidad, mejor es el modelo, y lo opuesto en caso contrario). Se eligen 300 iteraciones para el hiperparámetro max_iter ya que a partir de él no mejora el modelo de forma significativa.

[1] La técnica de Grid Search es un técnica de optimización que intenta encontrar los valores óptimos para los hiperparametros dados. El algoritmo realiza una búsqueda exhaustiva sobre cada hiperparametro usando como referencia un rango o lista de valores dada

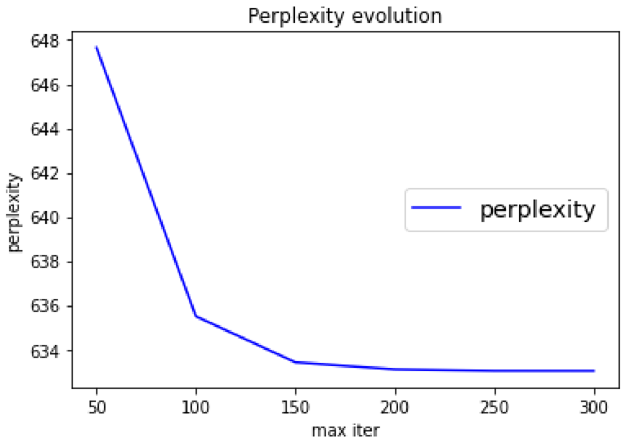


Figura 4.5: Curva de perplejidad.

4.2.2. Clustering y Sentence embeddings

El último enfoque abordado en esta fase exploratoria es buscar patrones en las agrupaciones de las respuestas. En la fase de modelado de tópicos se usó los vectores de tópicos de cada documento para agruparlos y visualizarlos. En está etapa se decide usar lo que hoy es el estado del arte en PLN, que son las representaciones vectoriales distribuidas (ó, *embeddings*), en particular, los *Sentence embeddings* (Sección 3.3.1).

Las herramientas usadas son las mismas que se vienen describiendo, y se agregan las biblioteca `sentence-transformers` y `umap`.

En el caso de la biblioteca `sentence-transformers` se usa un modelo pre-entrenado y multi-lenguaje [1] que se le brinda como parametro a la biblioteca y ella misma se encarga de descargarlo.

Una vez que se tiene el modelo de `SentBERT` cargado, se convierten las documentos y se comienza el proceso de agrupamiento de datos usando `hdbscan`. Se reduce la dimensionalidad de los datos con `umap` y luego se los visualiza usando la biblioteca de visualización `Altair`[2].

Asimismo, usando los *Sentence embeddings* por sí solo no fue suficiente para obtener los resultados esperados, y se explora el uso de lo que se conoce como Ingeniería de características ó por sus siglas en ingles «Feature Engineering». La técnica de «Feature Engineering» es el proceso de encontrar y formular las mejores características dado los datos que tenemos, el modelo y

[1] El modelo pre-entrenado que se usa es **paraphrase-multilingual-MiniLM-L12-v2**

[2] Altair, una biblioteca de visualización declarativa para Python, https://altair-viz.github.io.

la tarea. Una característica es generalmente un representación numérica de los datos (como por ejemplo el género se puede representar como un vector dado $\langle masculino, femenino, nobinario \rangle$).

Las características se agregaron en dos etapas (a) al comenzar el proyecto se adicionaron: largo de la respuesta, similitud de respuesta con el vector de verbos de reflexión, similitud de respuesta con frases de «utilidad» (útil, no útil), similitud de respuesta con verbos o nombres de actividades (actividad, propuesta, canción, etc.), (b) la siguiente etapa fue cuando el INAU proporcionó los datos de las encuestas adicionales, de esta forma se incluyó el género, la utilidad como un campo propio, y si un nueva característica sobre si existió un proceso reflexivo o no.

Capítulo 5

Análisis de Resultados

Hasta aquí hemos hablado de teoría, de procesos y de formas de explorar, analizar y sacar conclusiones sobre los datos. Si bien es necesario para dar contexto al trabajo, es el momento de pasar a la acción. En este capítulo se detallan las tres etapas desarrolladas durante este proyecto, la «Exploración de los datos», el «Modelado de tópicos» y por último se aborda «Clustering usando Sentence Embeddings».

El proceso de exploración y análisis sucedió en dos etapas. El INAU inicialmente nos brindó una sola planilla de *Excel* con datos, y a la mitad del proyecto se dio acceso a una segunda planilla con más datos que no habían sido aclarados en la primera etapa. Por ende se tuvieron en cuenta para la fase de exploración, y para el último punto de «Clustering con Sentence Embeddings» y no así para la parte de modelar los tópicos.

5.1. Exploración de los datos

El primer paso en la etapa de exploración es entender qué campos tiene el conjunto de datos, cuál es el tipo de los datos y ciertas estadísticas sobre los datos (en el caso de datos numéricos, como promedio, mediana, mínimo, máximo, etc.). El primer conjunto de datos y sus características se muestran en la Tabla 5.1.

En la misma tabla se puede ver la cantidad total de instancias, y cuántas instancias fueron etiquetadas –clasificadas en una o varias categorías por parte del INAU– y cuantas no.

La única columna de la cual se pueden obtener características es la de

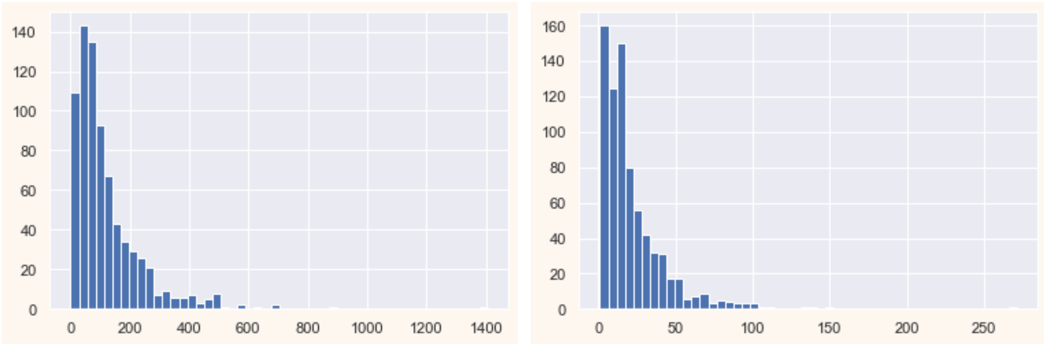
Tabla 5.1: Características del primer conjunto de datos (llamémosle INAU-2022-03-23)

Columna	**Super categoría**	**# instancias**
`respuesta`	-	759
1. Impulsó un proceso reflexivo entorno a la temática de la entrega	Utilidad	119
2. Manifiestan haber realizado alguna actividad o su intención de hacerla	Utilidad	119
3. Manifiestan que no le resultó útil	Utilidad	1
4. Valoración del dispositivo	Valoración positiva del dispositivo/actividad	298
5. Valoración de actividades concretas	Valoración positiva del dispositivo/actividad	89
6. Sugerencias/Observaciones	Sugerencias / Observaciones	39
7. No clasificables	No clasificables	67
# total de instancias		759
# instancias etiquetadas		557
# instancias sin etiquetar		202

las «respuestas». Generalmente nos interesa saber el largo promedio de los textos, y cómo se distribuyen estos textos (si son mayoritariamente largos, o mayoritariamente cortos). Se puede ver estos resultados en la Tabla 5.2 y en las Figuras 5.1a y 5.1b.

Tabla 5.2: Estadísticas de la columna respuesta

-	**Mínimo**	**Mediana**	**Promedio**	**Máximo**
`palabras`	1	15	22	271
`caracteres`	1	84	121	1405



(a) Histograma contando caracteres. **(b)** Histograma contando palabras.

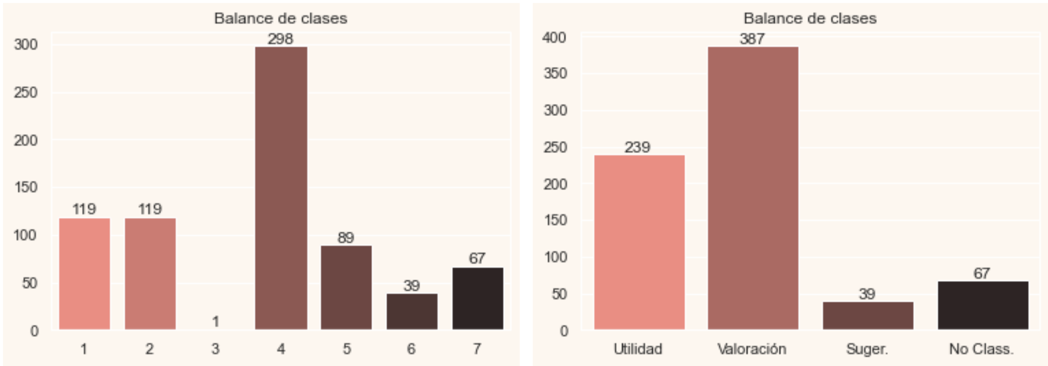
Figura 5.1: Distribución de las respuestas contando los caracteres (izq) y las palabras (der).

Es interesante notar en la Tabla 5.3 que el 75 % de las respuestas tienen menos de 27 palabras, o sea que mayoritariamente las respuestas son cortas.

Tabla 5.3: Quintiles y respuestas.

Quintil	**Largo en palabras**	**Ejemplo**
25%	7	*Muy linda toda la información recibida gracias*
50%	15	*Agradezco el apoyo en la crianza de nuestros hijos desde el amor y la contención*
75%	27	*Buen día , nosotros nos levantamos. Nos lavamos la carita tomamos la leche con chocolate nos cepillamos los dientes jugamos un poco con los juguetes del baúl*
$<$ 25%	más de 27 palabras	*Tiene muy buenas técnicas para la vida diaria cuando uno tiene hijos, a veces estamos tan ocupados que ellos no reciben la atención que realmente necesitan , esto te hace reflexionar y está muy bueno ,porque son formas distintas que se pueden utilizar en las rutinas del día*

Continuando con el análisis se puede ver cómo están distribuidos las instancias en cada categoría (Fig. 5.2a). Se puede apreciar dos gráficas, en la (a) se muestra el balance de clases al nivel de las subcategorías como se describieron en la Tabla 2, y en la (b) al nivel de las super categorías. Las categorías de **Utilidad** y de **Valoración del dispositivo/actividad** son las que tienen la mayor cantidad de respuestas clasificadas como tal.



(a) A nivel de subcategorias

(b) A nivel de categorías

Figura 5.2: Balance de clases

En cambio, si miramos al nivel de subcategorías se ve un desbalance entre las clases donde la categoría (4) (referente a ≪Valoración del dispositivo≫) contiene al 54 % de respuestas, y por el contrario, la categoría número (3) (referente a ≪Manifiestan que no le resultó útil≫) tiene una sola instancia.

Una vez analizado como las respuestas de las encuestas se distribuyen de acuerdo a las categorías dadas, usamos las subcategorías (que es el nivel más granular) para determinar si existe alguna correlación entre ciertas categorías (con el objetivo de agrupar en ≪super categorías≫ o poder usar esa información para analizar la construcción de un modelo predictivo). Para esto se usa una gráfica que se conoce como *Heatmap*[1] que nos permite ver las correlaciones entre los datos.

No existen correlaciones fuertes entre las columnas ya que como se puede apreciar la mayoría de los valores del se ubica alrededor de 0. El único valor que es ligeramente superior al resto corresponde al par $\langle 4, 7 \rangle$ del *heatmap* –con un valor absoluto de 0.39–. Esta correlación más fuerte se presenta entre las categorías ≪Valoración del dispositivo≫ y ≪No clasificables≫, lo cual nos dice que es posible que se trate de un error ó que la categoría ≪No clasificables≫ tenga una semántica distinta a la que se puede deducir.

[1]El *Heatmap* usa los colores más claros (valores cercanos a 1.0) y los más oscuros (valores cercanos a -1.0) para resaltar una alta correlación –la diagonal en la figura, por ejemplo– y el color rojo (valores cercanos a 0) para denominar que no existe correlación.

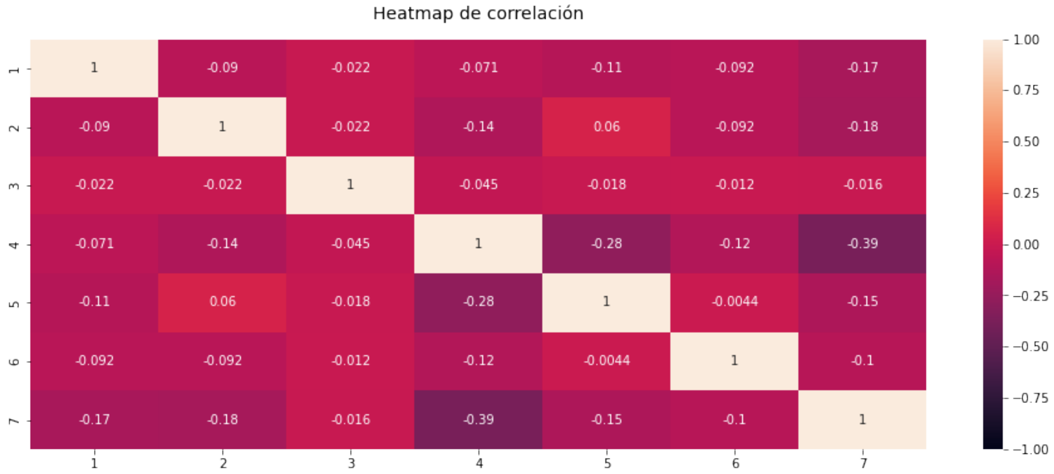


Figura 5.3: Correlación entre las clases.

Los datos de género y utilidad fueron brindados en una etapa posterior al comienzo del proyecto, por ende algunas de las visualizaciones y exploraciones realizadas no usan estos datos. Sin embargo, cuando se obtuvieron si se realizó el proceso de exploración y análisis de los nuevos datos ya que enriquecían el trabajo realizado.

Como se mencionó anteriormente a la mitad del proyecto se agregó un nuevo conjunto de datos que enriquece el conjunto inicial de datos brindados al comienzo del proyecto. Se exploran los datos nuevos y se detalla su esquema en la Tabla 5.4.

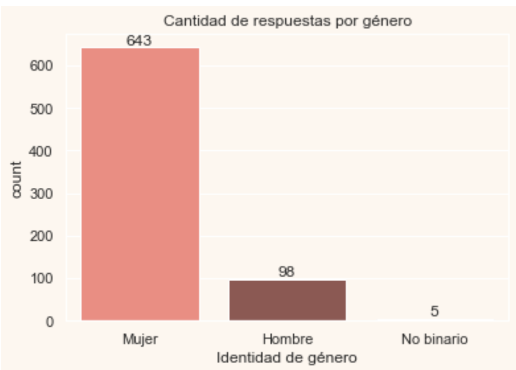
Dado los nuevos conjuntos de datos, se determina si estos pueden darnos nueva información para poder modelar mejor el problema. Como se ve en la Figura 5.4a el género se distribuye de forma desigual, siendo las mujeres las que más respondieron a la encuesta, seguido de un 13 % los hombres, y menos de un 1 % el género «No binario». También se consulta si el contenido les resultó útil (fig 5.4b) –siendo esta pregunta redundante con la categoría que el INAU diseñó como parte de su jerarquía– y como se ve, el 96 % encontró el contenido útil.

Tabla 5.4: Características del segundo conjunto de datos (llamémosle INAU-2022-03-23-segunda-entrega)

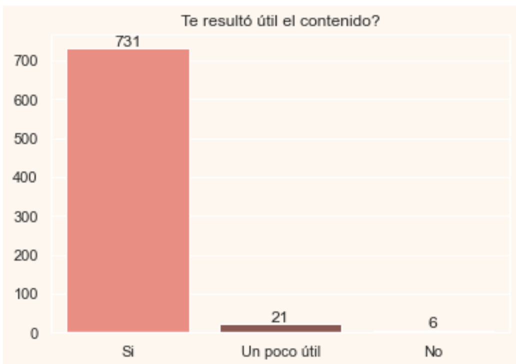
Columna	**Tipo**	**# elementos**
Marca temporal	`Fecha y tiempo`	759
Te resultó útil el contenido?	`Lista` Opciones: Si, No, Un poco útil	759
En caso afirmativo, marque las opciones que reflejan la utilidad.	`Lista` Opciones: ver Tabla 5.5	759
Identidad de género	`Lista` Opciones: Mujer, Hombre, No binario.	759
¿Querés comentarnos o sugerirnos algo? Nos encantaría leerte!	`Texto`	759

Tabla 5.5: Opciones de la pregunta: “En caso afirmativo, marque las opciones que reflejan la utilidad.”

Opción
`'Hice algo con los niños/as'`
`'Lo comentamos en casa'`
`'Lo compartí con alguien más, fuera de mi familia'`
`'Me hizo pensar en algunas cosas'`



(a) Distribución por género.



(b) Distribución por utilidad.

Figura 5.4: Distribución de las respuestas por género y utilidad.

5.2. Análisis de frecuencia de palabras y *emojis*

Generalmente cuando se tienen datos textuales nos interesa visualizar de qué se tratan esos textos usando un método rápido y heurístico –sin invertir mucho tiempo–. Las nubes de palabras[1] (o *wordclouds*) son una herramienta visual que muestra palabras como en forma de «nube». Estás palabras tienen tamaños distintos y el tamaño es fijado en función del peso de cada palabra en nuestro conjunto. En los problemas de «PLN» se estila usar la frecuencia de las palabras en nuestro corpus, o la medida de «Tf-Idf» (Sección 3.3) para asignarles un «peso». En nuestro problema usamos esta última dado que tiende a encontrar las palabras más relevante en cada respuesta (y no las más frecuentes, que generalmente son las palabras como pronombres, artículos, preposiciones, etc.).

En particular se hace muy gráfico ver que las palabras que más sobresalen son «gracias», «material», «cosas» y «lindo», dando el indicio de que el contenido del programa fue positivamente recibido.

[1] Las nubes de palabras aparecen al comienzo de la década del 90s en un trabajo alemán «Tausend plateaus», y luego en un trabajo en inglés «Microserfs» de Douglas Coupland en 1995.

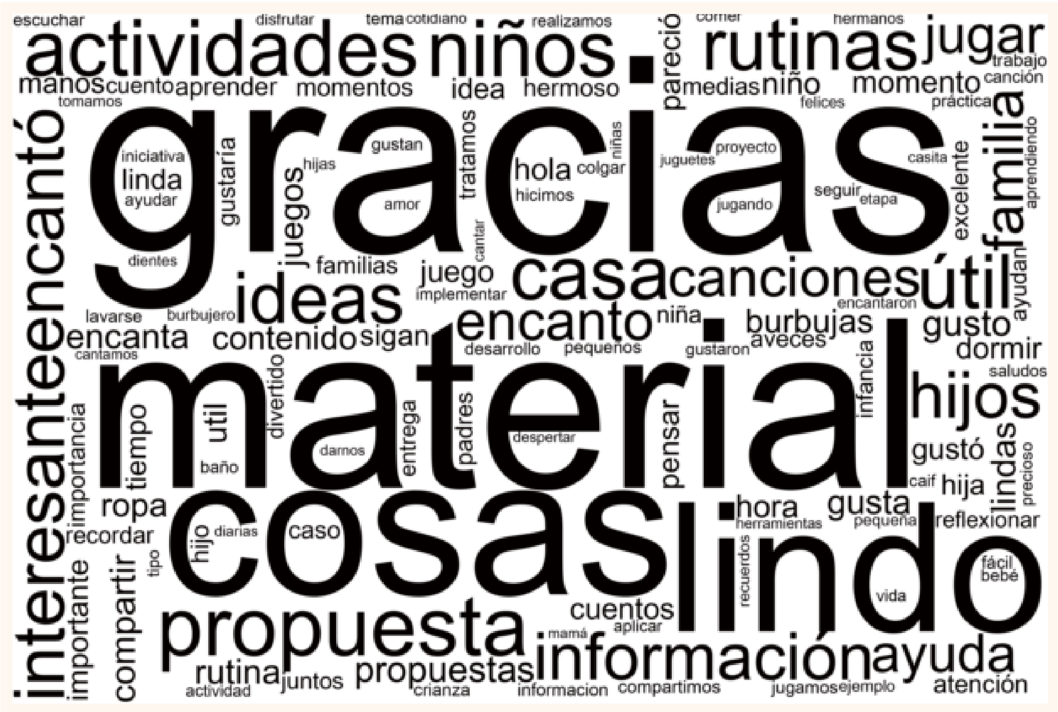


Figura 5.5: Nube de palabras de todas las respuestas (se usa Tf-Idf para darle el peso a cada palabra)

Otra idea fue visualizar como una «nube» los emojis –dado que estos aparecían de forma frecuente en las respuestas–, y como vemos a continuación son *emojis* positivos (abrazo, besos, corazones, ojos de corazones, etc.).

Figura 5.6: Nube de emojis de todas las respuestas (usando frecuencia). La estampita es un emoji que no se tradujo correctamente.

Este ejercicio visual nos permite acercarnos al contenido de las respuestas identificando palabras y «emojis» que en esta primera mirada nos indican un sentimiento positivo –en general–. En las siguientes secciones se trabaja en buscar mejores patrones en los datos –con nuevas herramientas– que nos permitan entender y visualizar mejor las respuestas de los usuarios.

5.3. Modelado de tópicos

Hasta aquí se han explorado los conjuntos de datos analizando características como el largo de las respuestas, frecuencias de palabras, emojis, etc. Es necesario avanzar en el análisis para determinar si existen características "escondidas" que podamos usar para agruparlas de forma tal de identificar patrones ocultos, o que sirvan para entrenar modelos predictivos (clasificadores) que nos permitan –a futuro– clasificar los textos de forma automática. Tal cual se detalló en la sección 4.2.1 –y en 3.2– se usa el modelo de LDA para obtener los tópicos más relevantes.

Se probaron dos enfoques en el modelado de LDA (a) un primer enfoque donde los textos se usan tal cual fueron brindados y (b) un segundo enfoque donde los textos se preprocesan[1] realizando lemmatización del texto, filtrado de stopwords[2], filtrado de *tokens* por su *pos-tag* (dejando sustantivos, adjetivos, verbos y adverbios), filtrado de símbolos de puntuación y excluyendo palabras de un solo carácter. Se analizó la perplejidad de ambos modelos para decidir cual era mejor para nuestro problema, y se obtuvieron los resultados de la Tabla 5.6.

Tabla 5.6: Perplejidad de los modelos de LDA entrenados con o sin preprocesamiento del texto.

Modelo	Perplejidad
Sin preprocesamiento	633
Con preprocesamiento	445

Se elige el modelo con menor perplejidad. Además, este modelo produjo tópicos más cohesivos y coherentes (analizando empíricamente los resultados). El mejor balance del algoritmo se obtuvo con 5 tópicos, y se usan las primeras 10 palabras de cada tópico (la literatura recomienda entre 5-20, ver Blei et al. 2003 y Hoffman et al. 2010).

Los tópicos que se encontraron son los que aparecen en la Figura 5.7, donde vemos las palabras en un gráfico de barras decreciente que corresponde a la distribución del tópico para ese conjunto de palabras.

Figura 5.7: Tópicos encontrados por LDA (5 tópicos con 10 palabras cada uno).

Si revisamos el contenido de cada uno de los tópicos se intenta asignar un nombre más familiar a estos de tal forma que cuando se visualicen, o se citen, se pueda asociar mejor el tópico con su contenido. Veamos el tópico (1) que contiene las palabras «niño, padre, hijo, momento, casa, etc.» puede ser lógico llamar a este tópico "Familia". Si repetimos este ejercicio con el resto, llegamos a la siguiente lista,

[1] Para esta etapa de preprocesamiento se usa la biblioteca *Spacy*, https://spacy.io/.

[2] Conjunto de palabras que son parte de una lista de exclusión. Generalmente son palabras de uso muy frecuente en el idioma como: el, de, la, los, las, etc.

- Topic 1, sería **Familia**
- Topic 2, sería **Entregas & rutina**
- Topic 3, sería **Propuesta de juegos**
- Topic 4, sería **Actividades**
- Topic 5, sería **Cuentos & canciones**

Es interesante notar que los tópicos encontrados se pueden caracterizar en grupos de acuerdo al contenido de las propuestas o sus consecuencias como "rutina" (mejoró o cambió algo en la rutina), o "compartir" o "aprender" (generó un aprendizaje o se compartió). El nombre que se les asigna es para simplificar las visualizaciones, pero es posible que iterando se puedan y se tengan que cambiar.

Una vez que se encuentran los tópicos más relevantes, y se tiene una primera visión del contenido de los documentos es necesario continuar avanzando para responder la pregunta de si es posible clasificar las respuestas en la jerarquía de clases brindada. El próximo paso es visualizar las respuestas usando la representación vectorial de estas y el algoritmo de *t-SNE* (sección 3.4.1). Recordemos que cada respuesta (o documento) se puede representar como un vector de tópicos (sección 3.2) donde cada tópico tiene una probabilidad asignada (ya que para el modelo de LDA, un documento está representado como una distribución de tópicos). Además, el algoritmo de *t-SNE* reduce la dimensionalidad de los vectores a un espacio 2D para poder ser visualizados como se ve en la gráfica de la Figura 5.8.

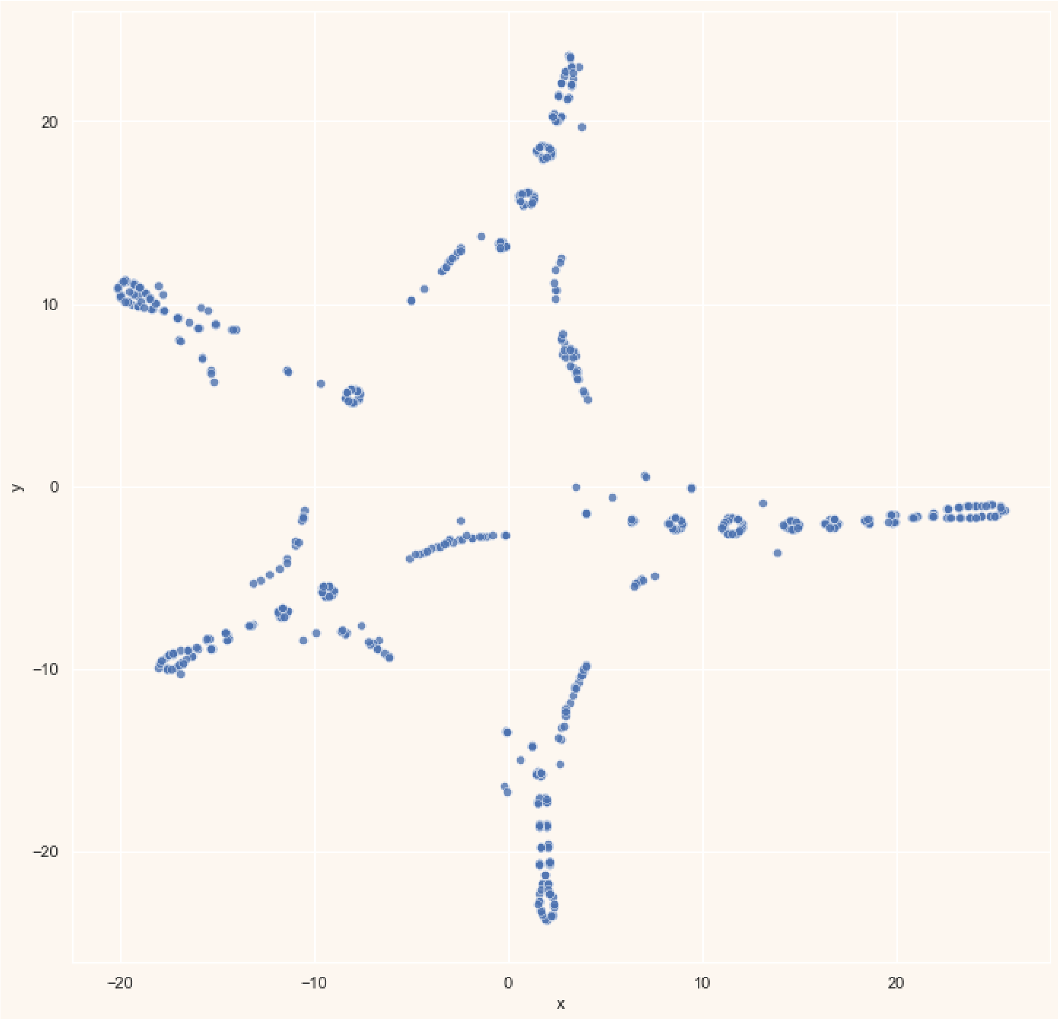


Figura 5.8: Respuestas codificadas usando LDA, mapeadas en un espacio 2D usando t-SNE.

Se puede ver que existe cierto patrón en los datos, ya que se genera una "estrella" con 5 puntas. Si visualizamos la estrella y se asocia con los cinco tópicos que fueron hallados nos surgen las preguntas ¿cada punta corresponderá a un tópico? ¿tendrá alguna correlación con las categorías que se brindaron?.

La forma de poder identificar si esos patrones aparecen también en el mundo multidimensional (recordemos que el algoritmo de *tSNE* puede mostrar agrupaciones de puntos que no necesariamente se corresponden en el espacio multidimensional, ver sección 3.4.1) se decide usar la técnica de *clustering HDBScan* (sección 3.1.1) con el objetivo de determinar si la representación visual producida por *t-SNE* tiene alguna relación con los tópicos generados por LDA. Tanto el algoritmo de *HDBScan* como el algoritmo de *tSNE* usan las respuestas representadas como vectores de tópicos. En la Figura 5.9 se puede ver la estrella con los puntos asignados a un cluster, además de los puntos no asignados (etiquetados como N/A).

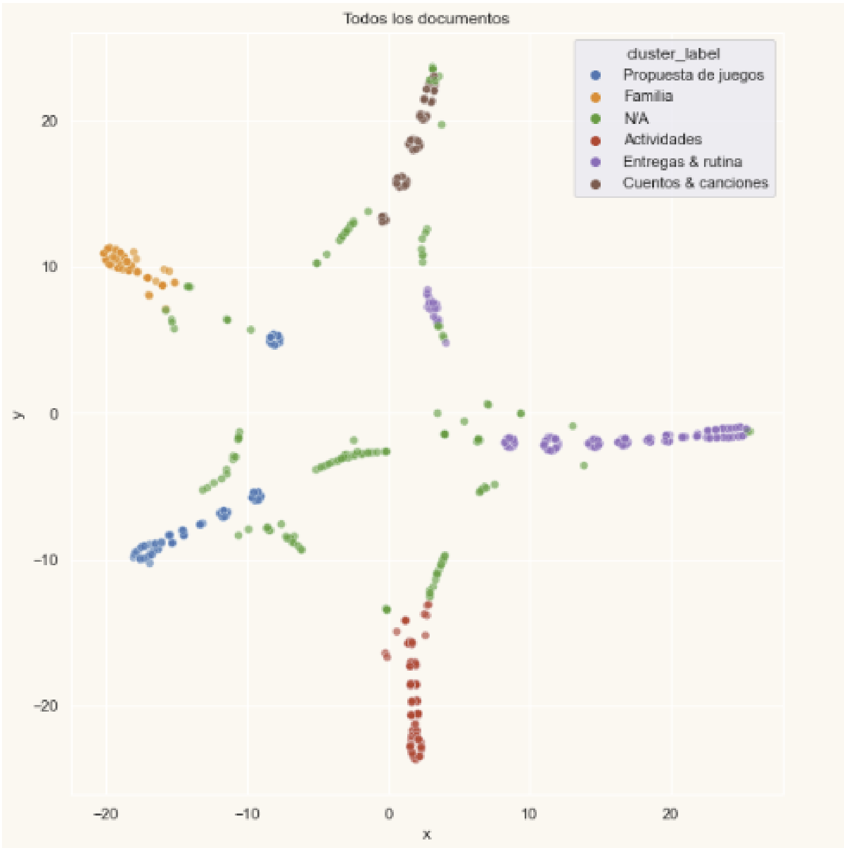


Figura 5.9: Respuestas codificadas usando LDA y agrupadas con HDBScan. Se pueden ver todas las respuestas, N/A es que no pudieron ser asignadas a un *cluster*.

En la Figura 5.10 se puede notar que efectivamente cada punta de las estrella corresponde a un tópico diferente, en la parte inferior tenemos las respuestas relativas a "Actividades", y en la superior a "Cantos & canciones". Nos queda por contestar si esta agrupación tiene alguna correlación con la jerarquía de categorías dadas o no (ya mencionamos que las categorías son demasiado abiertas y es posible que no exista).

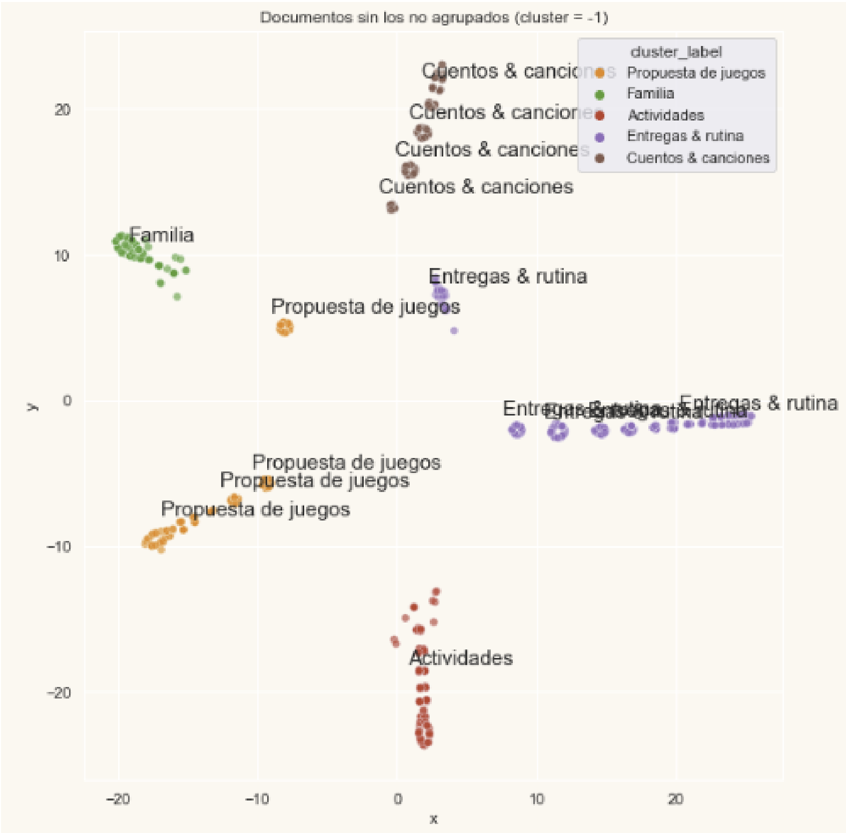


Figura 5.10: Respuestas codificadas usando LDA y agrupadas con HDBScan. Se puede ver solo las respuestas que fueron asignadas a un *cluster* y además, el tópico predominante de cada agrupación (usando el nombre amigable).

En la última gráfica de esta sección (Fig 5.11) –y además el último experimento– se grafica a cada una de las 7 sub-categorías descriptas en la Tabla 2 donde cada punto es una respuesta perteneciente a esa categoría, y su color es el tópico al cual pertenece. Se puede ver en detalle como en las gráficas de todas las categorías hay textos con diferentes tópicos. O sea, los tópicos son transversales a las categorías. Una primera conclusión es que las categorías puedan ser demasiado ambiguas para clasificar de forma automática las respuestas en ellas.

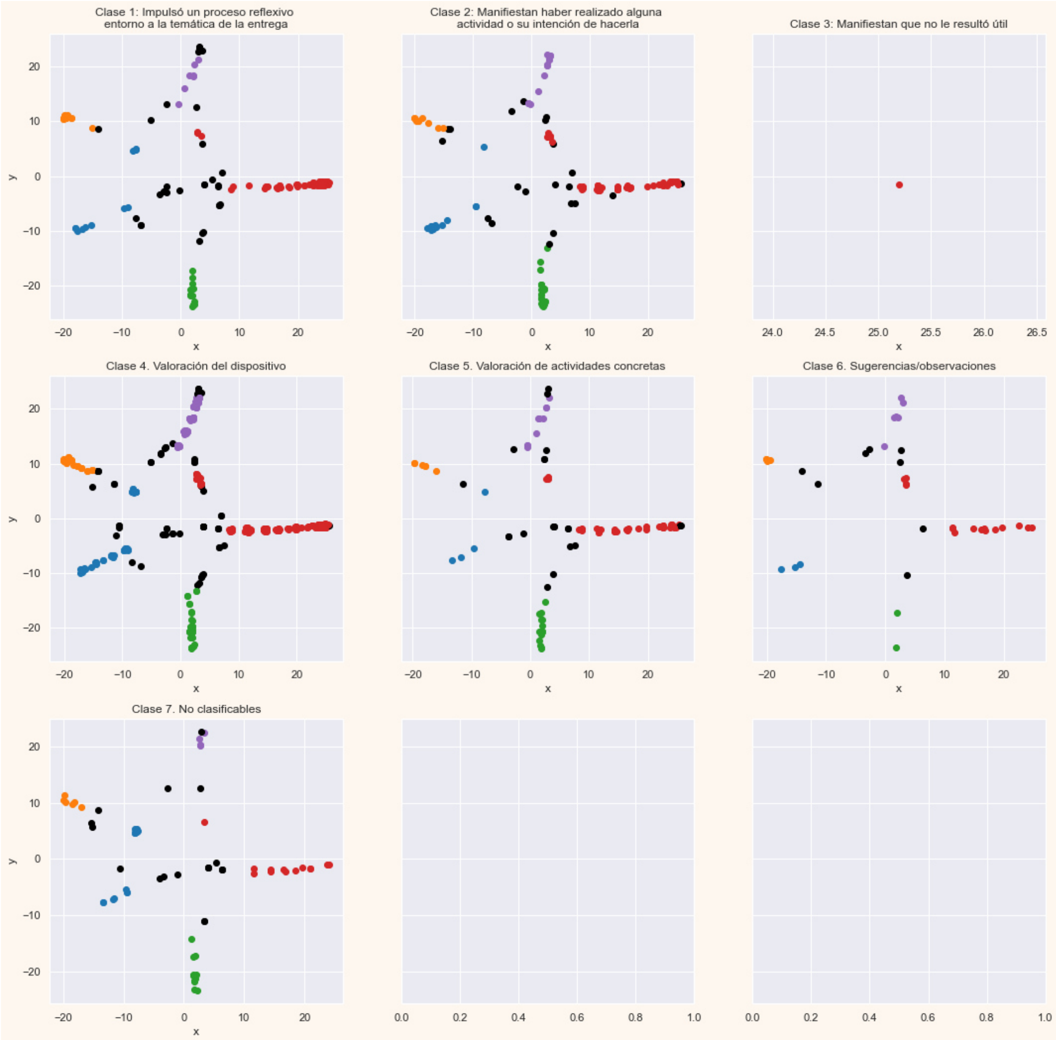


Figura 5.11: Relación entre las categorías y los tópicos.

5.4. Modelado con sentence embeddings

En esta etapa, el objetivo es el mismo, encontrar qué características nos permiten diseñar un modelo de utilidad para el INAU, pero en este caso modelamos los textos con *sentence embeddings*. Como se explicó en la sección 3.3.1, este método consiste en representar el texto con un vector que es codificado usando redes neuronales profundas con arquitecturas modernas. Se decide experimentar con este método porque es el tipo de enfoque que actualmente está dando mejores resultados para diferentes tareas de PLN.

El proceso comparte algunas etapas con el descripto en la sección de modelado de tópicos (sección 5.3). El proceso es el siguiente:

1. Se cargan los datos de los Excel a Pandas.
2. Se juntan los datos iniciales con los datos entregados posteriormente.
3. Se realiza un preprocesamiento del texto mínimo (se eliminan los tildes y se corrigen palabras usando un diccionario).
4. Se calculan los *sentence embeddings* para cada una de las respuestas usando el modelo pre-entrenado y multilenguaje (3.3.1).
5. Se corre el algoritmo de *clustering HDBscan*, y se visualizan los datos (en este punto se desarrollan varias iteraciones).

.

El primer resultado que se obtiene del proceso anterior se puede ver en la Figura 5.12. Esta visualización de las respuestas codificadas con el modelo de *SentBERT* y proyectadas en el plano usando el algoritmo de *UMAP* nos muestra puntos desordenados sin patrones claros. Sin embargo, se pueden identificar algunas agrupaciones débiles o de poco tamaño. El siguiente paso fue usar el algoritmo *HDBScan* para agrupar los puntos e intentar identificar patrones. Además, se adiciona que para cada *cluster* identificado se extraigan las palabras claves usando el algoritmo de *YAKE* (sección sobre YAKE! 3.5), como se puede ver en la Figura 5.13.

Figura 5.12: Representación de las respuestas usando *SentBERT* y UMAP para su visualización en 2D.

Si se inspecciona esta figura se nota un mayor grado de desorden de los puntos, se ve que los «clusters» resultantes son menos cohesivos (hay muchos y muy pequeños). También se ve una mayor granularidad que resalta características locales de los datos (por ejemplo, un «cluster» sobre burbujeros, o uno sobre «canciones para dormir»), y una conexión entre las palabras claves asociadas a cada «cluster» y los tópicos de la sección anterior.

Figura 5.13: Respuestas agrupadas con HDBScan y codificadas con SentenceBERT. Cada «cluster» tiene las palabras más relevantes de sus respuestas. Se resaltan algunos *clusters* que van a ser explorados en detalle más adelante.

Si nos detenemos en algunos *clusters* de la Figura 5.13 se encuentra que algunas de las palabras claves que aparecen son particularmente interesantes, como por ejemplo, se encuentra un *cluster* para los "burbujeros". Este caso motivó a que investigara en detalle el conjunto de datos para encontrar las respuestas que mencionaran términos relacionados a «burbujas» («burbujas», «burbujeros», etc.) con el objetivo de entender la naturaleza del *cluster* generado. En efecto, varias de las respuestas se refieren a un «burbujero» y «burbujas». Recurriendo a la documentación brindada por el INAU se comprueba que una de las actividades propuestas era «construir un burbujero». El contenido de este *cluster* nos muestra que el impacto de la propuesta fue muy positivo como se puede apreciar en la Figura 5.14.

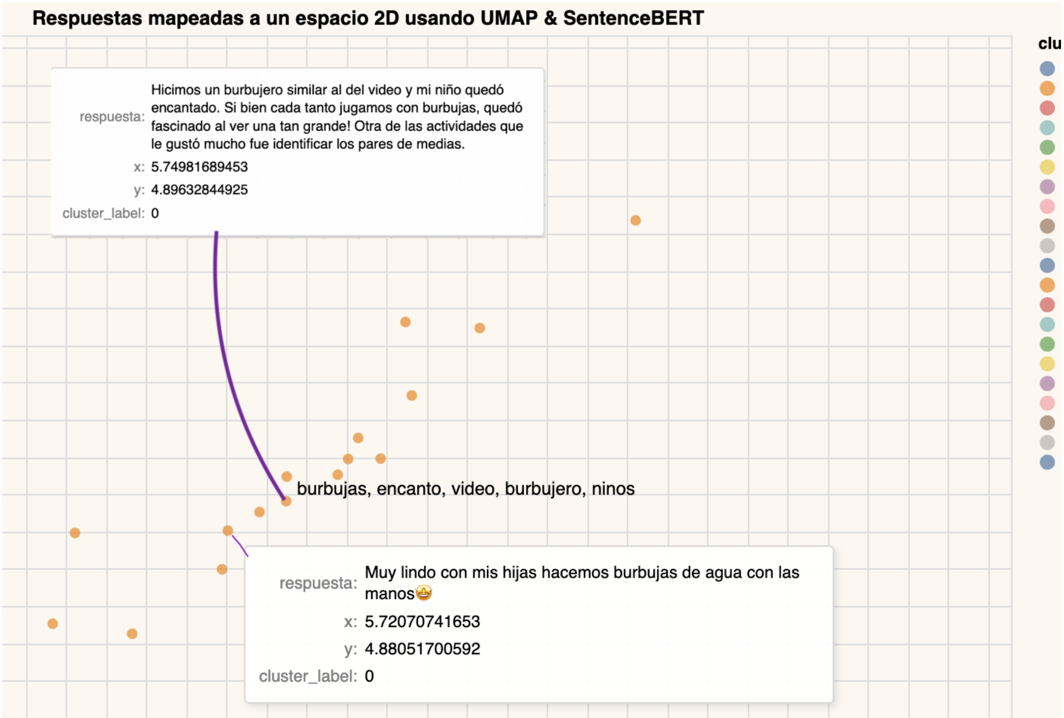


Figura 5.14: En este ejemplo se inspecciona el *cluster* de “Burbujas” donde se pueden ver dos respuestas seleccionadas que mencionan “jugar con burbujas”. Este *cluster* se corresponde con una de las actividades propuestas de ≪divertirse con burbujas≫ mediante la construcción de un ≪Burbujero≫.

Otro ejemplo con similares características fue la agrupación de las canciones –en la parte inferior derecha–, donde se hace referencia a canciones para ≪antes de dormir≫ y canciones para ≪lavarse las manos≫. Este último ejemplo, también es bien interesante de visualizar ya que esta agrupación, como se muestra en la Figura 5.15, se acompaña de palabras muy positivas como “encantó”, “lindo” y “gracias”.

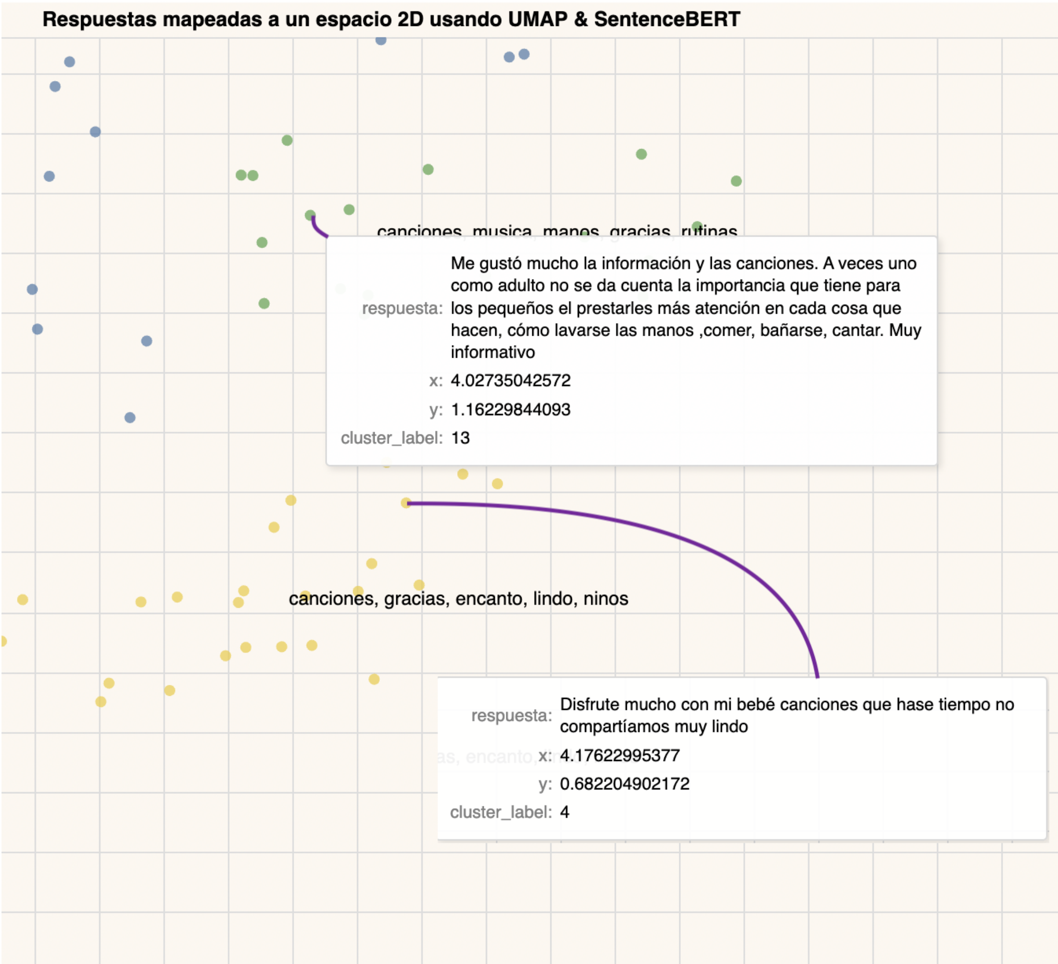


Figura 5.15: Se identifican dos *clusters* que tratan de «canciones» y «música». Las respuestas que se seleccionan sirven como ejemplo para notar que en ambas las canciones para dormir ó para lavarse las manos son mencionadas. Se puede interpretar que estos dos *clusters* puedan ser uno solo que trate de "Canciones y música".

Se ha visto que agrupar las respuestas usando solo las representaciones del modelo de «*Sentence embeddings*» no generan agrupaciones cohesivas, o al menos, que nos permitan identificar patrones claros para usar en una siguiente etapa de clasificación. Por este motivo se explora adicionar otras características para representar los vectores con el objetivo de ver los efectos que estos puedan tener a la hora de visualizarlos y agruparlos con el algoritmo de *clustering*.

Se concatenan al vector resultante del modelo *SentBERT* otros vectores que codifican características adicionales como utilidad, género, reflexión de utilidad, largo de la respuesta y nuevas características creadas en base a las otras. En particular, se usan las características de los nuevos datos brindados.

A continuación se detalla las características usadas y creadas,

- género, se codifica como un vector $\langle mujer, hombre, no\_binario \rangle$
- opciones que reflejan utilidad, se procesa la opción (que es una oración) usando SentBERT y se obtiene un vector del modelo de SentBERT.
- utilidad, se codifica como un vector $\langle si, no, poco\_util \rangle$ (vector de 0 o 1)
- indice de reflexión, característica artificial, creada usando el logaritmo del cantidad de *tokens* de la oración
- características artificiales para reflexión (`embeddings_reflexion`), utilidad (`embeddings_no_util`) y actividades (`embeddings_activity`), estás características se crean con el objetivo de identificar el grado de similitud de las respuestas a esas categorías. Se crea para cada caso un vector de palabras, por ejemplo para reflexión se usan las palabras "reflexionar", "pensar", "considerar" y se calcula la similitud de cada respuesta con estas palabras. De tal forma se obtiene un vector para cada respuesta. El proceso se repite para las actividades y la utilidad (puntualmente, se calcula la similitud con la palabra ≪inútil≫ o ≪no útil≫).

La primera visualización se muestra en la Figura 5.16 y se enfoca en mostrar las tres super categorías de la jerarquía dada (cada una de las gráficas es una super categoría), y en color naranja se ven las respuestas que pertenecen a esa categoría. Por ejemplo, en la gráfica de la izquierda se muestra la representación de las respuestas usando los nuevos vectores (*SentBert* más características adicionles) y cuales de esas respuestas fueron clasificadas como útiles de acuerdo al equipo del INAU. De la misma forma, para la super categoría Valoración Positiva y Observaciones. Se puede observar claramente que no existen agrupaciones claras que nos permitan identificar relaciones entre los puntos naranjas y la super categoría (Utilidad, Valoración del dispositivo positiva, Observaciones).

Se experimenta con otras características como el género (Figura 5.4a) sin obtener resultados que contribuyan a la detección de patrones útiles.

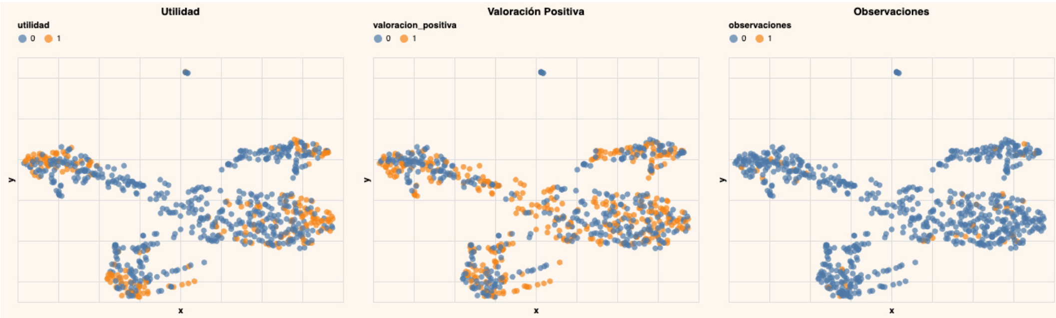


Figura 5.16: Se visualizan las respuestas usando *SentenceBERT*, características artificiales y *HDBScan*. Se colorea con naranja (valor 1) si la respuesta (o punto) pertenece a la super categoría determinada por el título. En la izquierda se ve las respuestas que pertenecen a la super categoría ≪Utilidad≫. En el centro, las respuestas que pertenecen a ≪Valoración positiva del dispositivo≫ y a la derecha, las respuestas que pertenecen a ≪Observaciones/Sugerencias≫. El objetivo de esta visualización es mostrar que en esta nueva iteración de la representación de las respuestas, no hay una clara correlación entre las agrupaciones y las super categorías.

Hasta ahora no se han generados resultados diferentes –incluyendo nuevas características en nuestros datos–. Sin embargo, probamos agregar como nueva característica el campo "En caso afirmativo, marque las opciones que reflejan la utilidad" del conjunto de datos brindado a mitad de proyecto, y resultó generar una interesante visualización que se ve en 5.17. Es claro que el campo tiene una incidencia muy importante en la generación de agrupaciones cohesivas determinando así cuatro grandes ≪clusters≫ (no se considera el ≪cluster≫ dado por el color rojo –Fig. 5.17– por no distinguirse del *cluster* que lo contiene). Estas agrupaciones corresponden a las opciones de utilidad (a) Hice algo con los niños (Celeste), (b) Lo comentamos en casa (Naranja), (c) Lo compartí con alguien más fuera de mi familia (Rojizo), (d) Me hizo pensar en algunas cosas (Turquesa) y (e) Varias opciones (Verde).

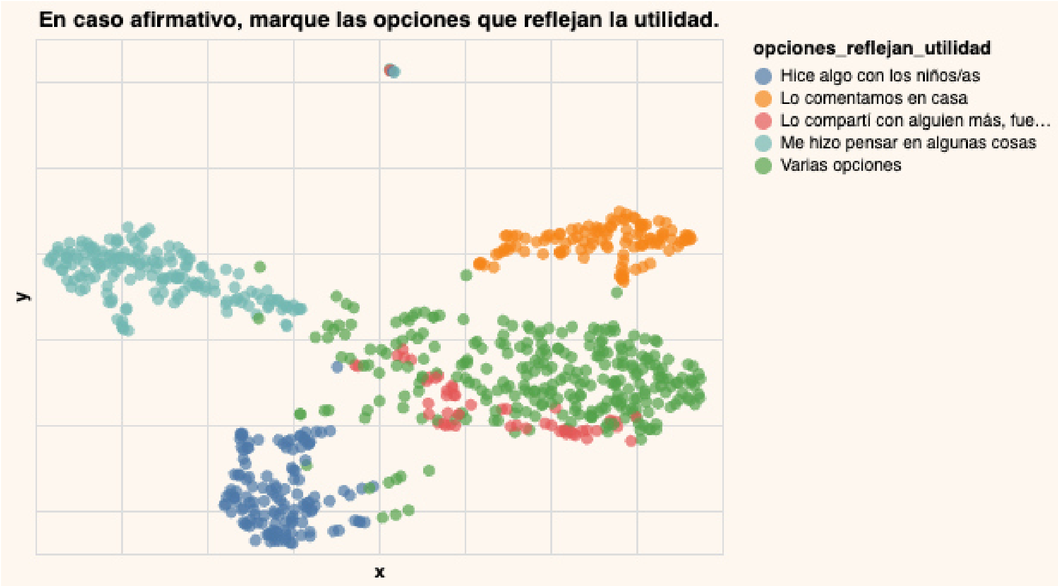


Figura 5.17: Visualización de los datos colorizando por el campo «En caso afirmativo, marque las opciones que reflejan la utilidad».

Explorando las instancias en cada agrupación, se puede observar que su contenido es cohesivo con la «opción de utilidad» que mencionamos anteriormente, lo cual nos permite asignarle un nombre a cada *cluster* más familiar. En la Figura 5.18 se asigna un nombre más amigable que nos permita visualizar, y entender mejor el contenido de estos. A modo de ejemplo, en la Figura 5.19 se muestran dos instancias que responden sobre juegos y actividades realizadas como parte de la propuesta, coincidiendo con el nombre del *cluster* «Juegos y actividades» al que pertenecen.

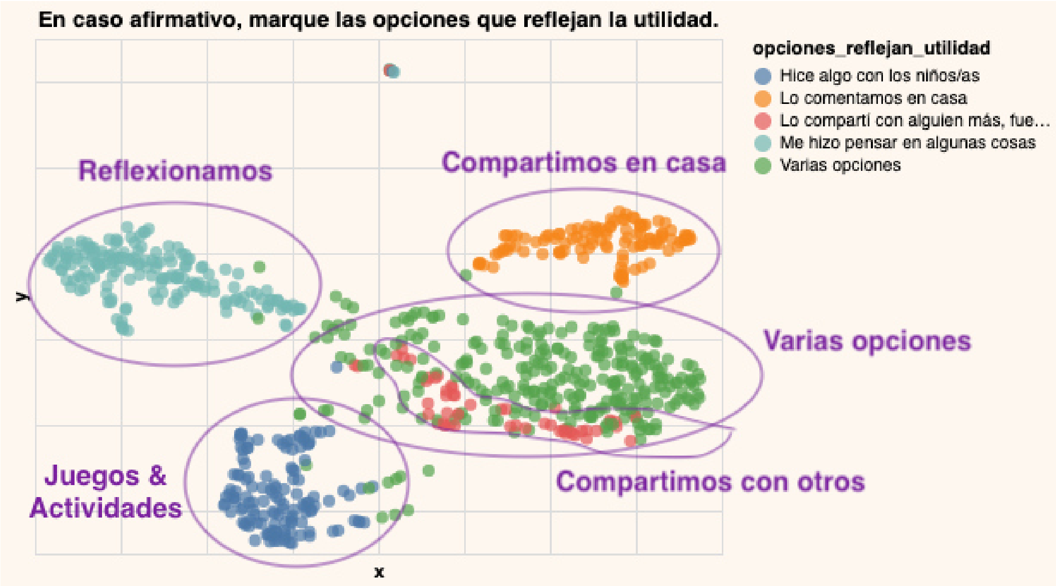


Figura 5.18: Nombre dado a los grupos de acuerdo a la opción de utilidad, y su contenido.

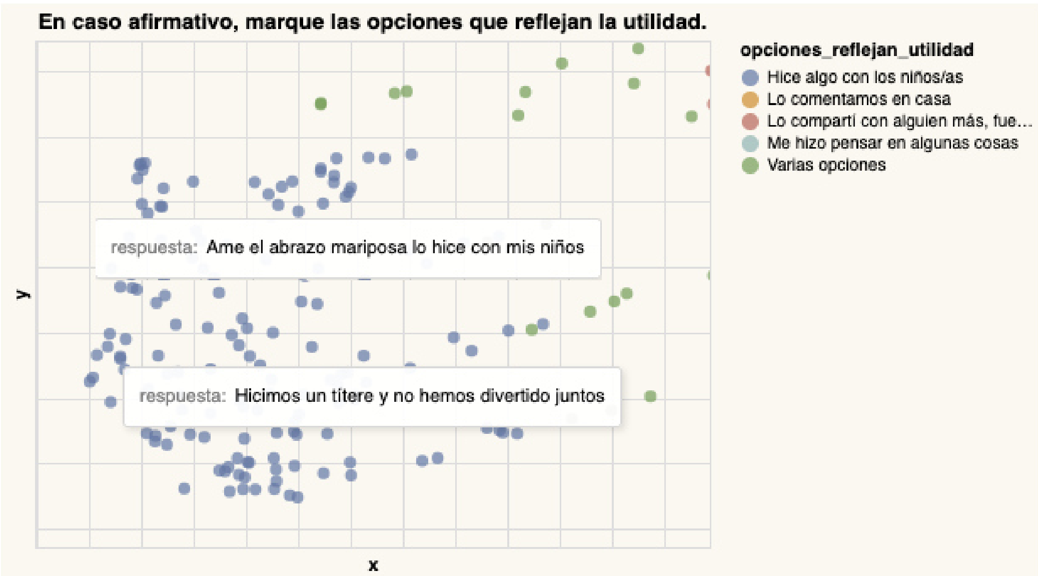


Figura 5.19: Dos instancias que nos sirven como ejemplo para mostrar el contenido del cluster ≪Juegos y Actividades≫, asociado con la opción "Hice algo con los niños".

El último experimento es observar las características de los *clusters* generados observando los cinco sustantivos y verbos más frecuentes que aparecen en cada una de las agrupaciones. En la Figura 5.20 se puede ver este experimento. Como se explicó anteriormente, las agrupaciones están influenciadas por el campo "En caso afirmativo, marque las opciones que reflejan la utilidad" pero en esta visualización los colores cambiaron. Por este motivo, y para mejorar el entendimiento visual se redondean los *clusters* asociándole la opción de acuerdo a la Figura 5.17. Si bien las agrupaciones no son exactamente las mismas, se puede visualizar asociaciones entre la «utilidad» y las palabras relevantes. Por ejemplo, cuando se observa la opción «Hice algo con los niños/as» se identifican los verbos "hacer" y "jugar", y los sustantivos "niños", "familia" y "cosas". Si miramos instancias (respuestas) dentro de este *cluster*encontramos respuestas como (en negrita se destaca el verbo o sustantivo),

- «**Realizamos** un **juego** colgamos ropaz y medias juntos»
- «Le **hice** escuchar los videos!!!»
- «En casa siempre aplicamos el **juego** y lo divertido que es para **aprender** cosas juntos»

.

El experimento es interesante pero no es útil ya que no brinda información adicional. Se observa que los verbos más usados son hacer, tener, jugar, gustar, encantar, etc. mientras que los sustantivos más frecuentes son relacionados a la propuesta, tales como material, propuesta, canciones, actividades, casa, etc..

Se destaca en todo el análisis de resultados la palabra **gracias** donde aparece como palabra clave, como sustantivo más frecuente y como término repetido de los tópicos encontrados en el sección anterior (ver 5.3).

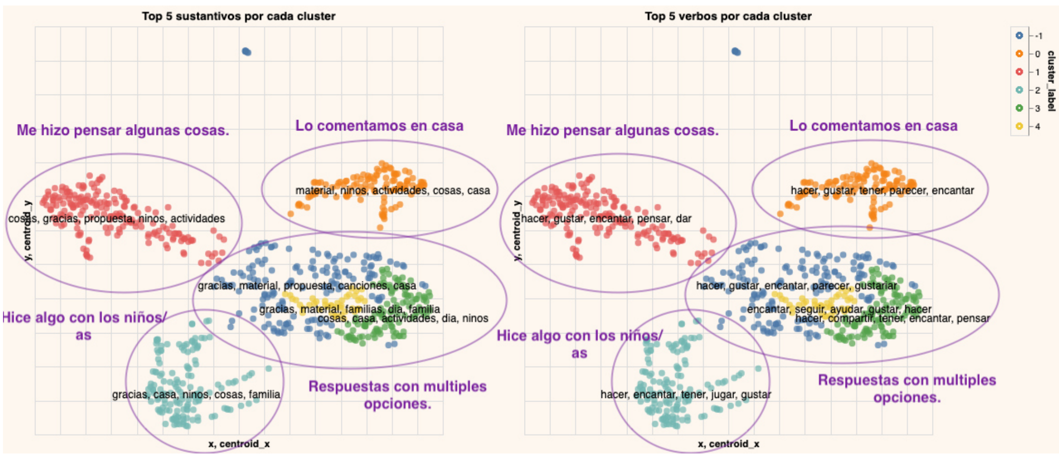


Figura 5.20: Visualización de los clusters & palabras claves para el mapeo de «Sentence embeddings» con «Feature Engineering».

Capítulo 6

Conclusiones y trabajo a futuro

El objetivo del INAU era responder si es posible automatizar el proceso de clasificación de los datos según la jerarquía diseñada. Sin embargo, desde el comienzo se descarta la opción de construir un clasificador ya que el primer conjunto de datos obtenidos no contiene la cantidad suficiente de datos para desarrollar e implementar un clasificador.

Lo que se expuso en este trabajo nos permite concluir que con los datos brindados se hace muy complejo encontrar patrones y/o características que nos permita separar los datos en agrupaciones tales que se puedan mapear a una categoría de la jerarquía dada. Desde la perspectiva de este trabajo, observamos que la jerarquía construida presenta cierta ambigüedad además de que el significado de algunas clases es complejo de razonar e implementar.

Durante gran parte del proyecto se contó solo con un conjunto de datos en el cual la entrada eran solo las respuestas de texto libre. En una primera etapa, se intentó buscar patrones y características que permitieran identificar agrupaciones o tópicos que se acercaran a las categorías dadas. Sin embargo, analizando los datos con las distintas técnicas no fue posible identificar formas de separar las respuestas en las categorías dadas. En una segunda etapa, se obtienen más datos donde se incluye un conjunto de características adicionales como utilidad del programa y género, y se exploran y analizan estos nuevos datos. A pesar de ello, se reafirma la conclusión de que –incluyendo los nuevos datos– con la jerarquía diseñada se hace muy difícil encontrar agrupaciones de datos que nos señalen posibles mapeos. Además, como parte del segundo conjunto de datos encontramos que existen columnas que pueden responder de forma directa a algunas de las categorías, por ejemplo, se puede ver que para la super categoría utilidad tenemos dos columnas con respuestas sobre la utilidad del programa. Del mismo modo, es importante notar que para las categorías «Sugerencias/Observaciones» y «No clasificables» se cuenta con muy poca cantidad de datos clasificados como tales, por lo que se hace muy difícil detectar patrones que nos habiliten a modelar un clasificador.

A pesar de no haber encontrado patrones que nos permitan construir un clasificador y automatizar el problema planteado, se obtienen dos resultados interesantes. Se identifica en los diferentes enfoques del trabajo el predominio de palabras con connotación positiva como «gracias» (en la nube de palabras y los tópicos), «encantar» (en las agrupaciones), «gustar» y «jugar». Se puede deducir un impacto positivo del programa dado estas valoraciones. El segundo resultado fue encontrar propuestas del programa con un gran impacto para los usuarios, como la construcción de un «burbujero» (que se puede ver como una agrupación en la sección 5.4) o las distintas canciones sugeridas (como la canción de «lavarse las manos»).

Esta conclusión abre nuevos caminos para poder enfocar el trabajo a futuro. Un primer camino es explorar el diseño de un clasificador que detecte el sentimiento de la respuesta (si es positivo, negativo, o neutro), y cuáles son los sustantivos o entidades asociados a esa valoración. Esto nos ayudaría a comprender qué propuestas o actividades gustaron, y cuáles no. De alguna forma este camino nos ayudaría a entender la valoración que hace el usuario en torno al «dispositivo» y/o las «actividades». También es posible modelar un sistema de recomendación que dado los usuarios, y actividades que gustaron o no, recomiende nuevas actividades que puedan ser útiles para proponer a los nuevos usuarios (como hacen los sistemas de streaming online, por ejemplo).

En lo personal, este proyecto de tesis me deja una mirada mucho más profunda y cercana sobre las realidades y dificultades que presentan las familias que son alcanzadas por los programas del INAU. Los desafíos que existen para llegar a estas familias con propuestas que enriquezcan y habiliten la construcción de nuevas realidades. A lo largo del proceso tuve la oportunidad de leer cada una de más de 700 respuestas y poder conectarme –desde mi ser padre– de los desafíos y dificultades que se expresan en ellas, reconociendo en las palabras de las mamás, papás o personas a cargo la alegría, los enojos, las lagrimas, y las posibilidades que se abren cuando se generan los encuentros con los hijos.

Referencias bibliográficas

Ares, G., Bove, M. I., Fuletti, D., Blanc, M. V., Brunet, G., y Vidal, L. (2020). Actitudes y comportamientos de las familias uruguayas en relación con el coronavirus (COVID-19) a cinco meses del comienzo de la emergencia sanitaria, 36. https://www.unicef.org/uruguay/informes/las-familias-uruguayas-en-relacion-con-el-coronavirus-covid-19

Bengfort, B., Bilbro, R., y Ojeda, T. (2018). *Applied Text Analysis with Python*. O'Reilly Media, Inc.

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77-84.

Blei, D. M. (2014). Build, Compute, Critique, Repeat: Data Analysis with Latent Variable Models, 203-232. http://arks.princeton.edu/ark:/88435/pr12b6p

Blei, D. M., Ng, A. Y., y Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993-1022.

Campello, R. J. G. B., Moulavi, D., y Sander, J. (2013). Density-Based Clustering Based on Hierarchical Density Estimates (J. Pei, V. S. Tseng, L. Cao, H. Motoda y G. Xu, Eds.), 160-172.

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Nunes, C., y Jatowt, A. (2020). YAKE! Keyword extraction from single documents using multiple local features. *Information Sciences*, *509*, 257-289. https://doi.org/https://doi.org/10.1016/j.ins.2019.09.013

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., y Kurzweil, R. (2018). Universal Sentence Encoder. https://doi.org/10.48550/ARXIV.1803.11175

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., y Bordes, A. (2017). Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. https://doi.org/10.48550/ARXIV.1705.02364

Devlin, J., Chang, M.-W., Lee, K., y Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://doi.org/10.48550/ARXIV.1810.04805

Hoffman, M., Bach, F., y Blei, D. (2010). Online learning for latent dirichlet allocation. *advances in neural information processing systems*, *23*.

Honnibal, M., y Montani, I. (2017). *spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing* [To appear].

Jurafsky, D., y Martin, J. H. (2009). *Speech and Language Processing (3rd Edition, draft)*. Prentice-Hall, Inc.

Kamath, U., Liu, J., y Whitaker, J. (2019). *Deep learning for NLP and speech recognition* (Vol. 84). Springer.

Koch, G., Zemel, R., Salakhutdinov, R., et al. (2015). Siamese neural networks for one-shot image recognition. *ICML deep learning workshop*, *2*, 0.

Martínez, S. S. (2016). La escritura de los jóvenes en los chats en el siglo XXI. *Didáctica. Lengua y Literatura*, *27*, 183-196. https://doi.org/10.5209/rev_DIDA.2015.v27.51298

McInnes, L., Healy, J., y Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. En S. van der Walt y J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56-61). https://doi.org/10.25080/Majora-92bf1922-00a

Mikolov, T., Chen, K., Corrado, G., y Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. https://doi.org/10.48550/ARXIV.1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., y Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. https://doi.org/10.48550/ARXIV.1310.4546

pandas development team, T. (2020, febrero). *pandas-dev/pandas: Pandas* (Ver. latest). Zenodo. https://doi.org/10.5281/zenodo.3509134

Reimers, N., y Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. https://doi.org/10.48550/ARXIV.1908.10084

Rumelhart, D. E., Hinton, G. E., y Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533-536.

Schmidhuber, J., Hochreiter, S., et al. (1997). Long short-term memory. *Neural Comput*, *9*(8), 1735-1780.

Tufte, E. R. (2001). *The Visual Display of Quantitative Information*. Graphics Press.

Van der Maaten, L., y Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, *9*(11).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., y Polosukhin, I. (2017). Attention Is All You Need. https://doi.org/10.48550/ARXIV.1706.03762

Wattenberg, M., Viégas, F., y Johnson, I. (2016). How to Use t-SNE Effectively. *Distill*. https://doi.org/10.23915/distill.00002

Wilmott, P. (2019). *MACHINE LEARNING: An Applied Mathematics Introduction*. Panda Ohana Publishing.

Glosario

Un glosario incluye una lista de términos y su explicación sucinta. El objetivo de este apartado es permitirle a un lector especializado en el área, aunque no necesariamente en la temática, comprender con mayor facilidad ciertos términos.

Se organiza en forma alfabética y en el cuerpo de la obra se lo puede señalar con VERSALITA la primera vez que se menciona, si este tipo de letra no fue utilizado con otro fin.

Algoritmo de clustering Un algoritmo de *clustering* es un método que particiona un conjunto de datos en grupos o *clusters* de documentos (e textos en nuestro caso) relacionados que pertenecen a ciertos tópicos o categorías. 8

Aprendizaje automático no supervisado Los algoritmos de aprendizaje automático no supervisado intentan encontrar las relaciones o patrones en los datos por sí solos (Wilmott, 2019). 9

Aprendizaje profundo Se refiere al uso de redes neuronales para el aprendizaje automático pero profundas, o sea, con más capas, Jurafsky y Martin, 2009. 22

Bag of words Modelo que codifica el texto como un vector de 0s y 1s de largo fijo –la cardinalidad del vocabulario– y se coloca un 1 si la palabra del vocabulario pertenece al texto o un 0 en caso contrario (Jurafsky y Martin, 2009). 20

Corpus Del latín cuerpo, refiere a una colección de textos o documentos textuales. 20

Curse of dimensionality Problema que sufren los espacios de vectores donde algunas representaciones vectoriales tienen una alta dimensionalidad

–el largo del vector es muy grande– y generalmente, estos vectores son muy poco densos –sus entradas son 0s en su mayoría–. Esto hace que se necesite un gran poder computacional, tanto en capacidad de computo como en recursos de memoria (Kamath et al. 2019). 21

emoji Según la RAE, pequeña imagen o icono digital que se usa en las comunicaciones electrónicas para representar una emoción, un objeto, una idea, etc. 5

HDBScan Algoritmo de clustering desarrollado por Campello et al. 2013, siendo una extensión de *DBScan* pero usando elementos de agrupamiento jerarquizado (*hierarchical clustering*) y combinando técnicas de grafos para obtener los «clusters». 13

Latent Dirichlet Allocation Modelo probabilístico también llamado «modelo probabilístico generativo» donde se asume que nuestros datos surgen de un proceso generativo que incluye *variables ocultas* (Blei et al. 2003). 17

Lematización Proceso por el cual se transforma un *token* en su respectivo *lema* 12

Modelado de tópicos El modelado de tópicos es un proceso por el cual se extraen los tópicos o temas más relevantes de un corpus o conjunto de textos, con el objetivo de entender de qué trata este conjunto sin la necesidad de inspeccionarlo en detalle (Blei, 2012). 9

Part-of-Speech Clase a la que pertenece una palabra –si es sustantivo, verbo, adjetivo, u otro tipo–. 12

Part-of-Speech tagging Técnica que permite etiquetar las palabras de una oración con la clase a la que pertenece (si es sustantivo, verbo, adjetivo, adverbio u otro tipo). 12

perplejidad La perplejidad en el contexto del Procesamiento del Lenguaje Natural es una forma de evaluar un modelo de lenguaje (un modelo de lenguaje es una probabilidad de distribución sobre oraciones o textos). Esta mide con respecto a un corpus de prueba la probabilidad del modelo de generar dicho corpus. 37

Procesamiento del lenguaje natural El procesamiento del lenguaje natural permite a las computadoras llevar a cabo tareas útiles que involucran al lenguaje humano. Tales tareas pueden mejorar las formas de comunicarnos con dispositivos (por ejemplo, con Siri en nuestros teléfonos, o el asistente de Google), mejorar la forma de comunicación entre personas (asistentes gramaticales en los procesadores de texto), o simplemente para cuando necesitamos procesar el lenguaje para obtener información de utilidad para cierta tarea que estemos emprendiendo (Jurafsky y Martin, 2009). 11

Redes Neuronales Las redes neuronales son un método de aprendizaje automático que toma su diseño y nombre de un modelo de neurona de nuestro cerebro, Jurafsky y Martin, 2009. 22

Sentence embeddings Es una representación vectorial de una oración o texto, donde una oración es mapeada a un vector de números reales. 8, 10

t-SNE Método de reducción de dimensionalidad introducido por Van der Maaten y Hinton, 2008 que tiene la capacidad de mantener las estructuras locales sin perder la estructura global, generando representaciones visuales muy ricas. 26

Tf-Idf Modelo de *Term frequency-Inverse document frequency* (o TF-IDF) es una mejora sobre el modelo de *Bag of Words* donde la representación vectorial tiene en cuenta las frecuencias de las palabras en los documentos y además, en el corpus (Jurafsky y Martin, 2009). 22

token El token se define como una secuencia de caracteres que están agrupados de tal forma que forman una unidad semántica usable para ser procesada. 11

UMAP Técnica que ofrece un conjunto de mejoras respecto a *t-SNE*, en particular, mejora la performance –reduce el tiempo de cómputo– para volúmenes de datos grandes, y mejora la fidelidad al preservar la estructura global de los datos McInnes et al. 2018. 28

Vocabulario Conjunto de todas las palabras (únicas) que aparecen en todo el corpus. 20

Word embeddings Representación vectorial de una palabra que a diferencia de otras representaciones, producen vectores de menor largo, más densos –generalmente de 200 a 500 elementos– y guardando información de contexto (Kamath et al. 2019). 22

YAKE! Algoritmo de extracción de palabras claves no-supervisado que utiliza las características estadísticas de las palabras en el texto para identificar y ordenar las palabras dado un cierto criterio de relevancia (Campos et al. 2020). 30

APÉNDICES

Apéndice 1

Apendice A. Repositorio & Datos

Durante todo el proyecto se trabajo usando el repositorio de Gitlab de la Udelar. Los datos también se incluyeron en el mismo repositorio.

- **Repositorio:** https://gitlab.fing.edu.uy/nestor.moreira/tesis-computacion-2022
- **Google Drive con los datos:** https://drive.google.com/drive/u/1/folders/1ttZSxd5l2z3NDRjOVvOy1JuWDGiVQykG

ANEXOS

Anexo 1

Programa INAU & Materiales

Entregas del **INAU** del programa *Parentalidades Comprometidas*,

- **Primera entrega**, “Nuestras rutinas”: https://bit.ly/PPCenCasa1
- **Segunda entrega**, “Para el encuentro y disfrute”: https://bit.ly/PPCenCasa2
- **Tercera entrega**, “Y yo, como me cuido”: https://bit.ly/PPCenCasa3
- **Cuarta entrega**, “Convivencia saludable”: https://bit.ly/PPCenCasa4

El material complementario del programa donde se detalla la propuesta y cada una de las entregas se puede encontrar aquí: https://drive.google.com/file/d/1ycA4LB18oEE7h41V6Ny_un71inafipf9/view?usp=sharing.