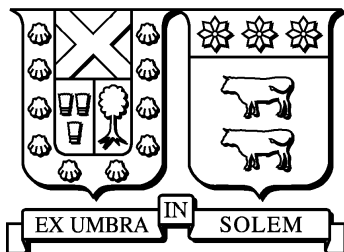


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE INFORMÁTICA

SANTIAGO – CHILE



MINERÍA DE DATOS EN LINKEDIN

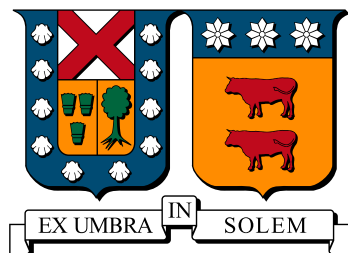
FELIPE ANDRÉS CARMONA BUSTOS

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: RICARDO ÑANCULEF

ABRIL 2017

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO – CHILE



MINERÍA DE DATOS EN LINKEDIN

FELIPE ANDRÉS CARMONA BUSTOS

**MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO**

PROFESOR GUÍA: RICARDO ÑANCULEF

PROFESOR CORREFERENTE: MARCELO MENDOZA

ABRIL 2017

MATERIAL DE REFERENCIA, SU USO NO INVOLUCRA RESPONSABILIDAD DEL AUTOR O DE LA INSTITUCIÓN

Agradecimientos

En primer lugar me gustaría agradecer a mi familia que tanto me apoyó y soportó por estos 6 años que estuve dentro de la universidad. Agradecer encarecidamente a mis padres que siempre estuvieron ahí cuando tuve algún problema, siempre haciendo las cosas tal que mi situación en la universidad no se complicara. Agradecer a mi madre que siempre está ahí cuando paso por momentos difíciles y a mi padre que siempre puedo contar con sus consejos cuando tengo algún problema. También agradecer a mis 3 hermanos que me ayudaron a forjar mi carácter, mi personalidad y mi sentido del humor. Agradecer a mi hermana América que tiene un logro extra por soportarme en mis momentos de máximo estrés y a mis hermanos Mauricio y Juan los cuales cada vez que los veo siempre me sacan una sonrisa.

Agradecer también a mis grandes amigos de la vida ya que sin ellos yo no estaría escribiendo este documento de agradecimiento. Los **fabulosos** amigos de la educación media con los cuales comparto historias (muchas historias) que siempre recuerdo con alegría y los **magníficos** amigos de la universidad que fueron unos de los pilares fundamentales en mi estadía en la universidad.

Finalmente dedicar este documento a todas las personas que hicieron posible su realización.

sld2.

Resumen

El nacimiento de redes sociales como *Facebook* o *Twitter* ha estimulado significativamente la utilización de la WEB por parte de las personas, creando un medio en el cual se intercambian creencias, juicios y opiniones que de alguna forma reflejan los eventos del mundo real. Ese enorme flujo de comunicaciones representa una fuente de datos en constante expansión y actualización, que puede ser utilizada por diversos actores en la toma de decisiones.

En esta memoria se aborda el problema de extraer información o patrones sobre el conjunto de competencias profesionales declaradas por las personas en la red social *LinkedIn*. A diferencia de *Facebook* o *Twitter*, *LinkedIn* es una plataforma orientada a la creación de contactos profesionales y de negocios, lo que la transforma en una fuente interesante de datos para estudios sobre la fuerza de trabajo. En efecto, *LinkedIn* es ampliamente utilizada por las personas para buscar empleo, y por las empresas para reclutar personal calificado, por lo que se puede asumir que las competencias allí declaradas reflejan aquellas que la industria considera relevantes.

El objetivo específico de esta memoria es la definición y aplicación de una metodología que ayude a entender y a visualizar el sistema de competencias profesionales (*skills*) declaradas por ingenieros informáticos de cuatro universidades chilenas en *LinkedIn*. La hipótesis de trabajo es que existen grupos de competencias que puedan ser identificados como diferentes líneas de especialización laboral y que la distribución de éstas es diferente entre profesionales de distintas universidades. Analizar esta hipótesis resulta relevante, por ejemplo, para orientar políticas sobre perfil de egreso y promoción en instituciones de educación superior.

Una contribución de este trabajo, es el levantamiento de una base de datos, correspondiente

a una muestra de 2830 perfiles informáticos de *LinkedIn* y sus respectivas listas de competencias, además de otros atributos que podrían ser de interés en futuros estudios. Después de limpiar y someter este conjunto de datos a un análisis exploratorio preliminar, se propone la utilización de técnicas de *clustering* para determinar la existencia de clases de competencias y usuarios. Se estudia el desempeño de diferentes métodos, incluyendo algoritmos jerárquicos, espectrales y de variable latente (*Topic Models*), abordando el problema de la representación más adecuada y la validación de los grupos obtenidos mediante diferentes métricas. Se propone un método para visualizar cada uno de los grupos encontrados usando *Word Clouds* y comparar grupos de usuarios en términos de competencias. Una contribución final de este trabajo es la definición y evaluación de un sistema de auto-completación de competencias, basado en métodos de filtrado colaborativo, que muestra una precisión superior al 90 %. Esto último sugiere que las estructuras encontradas sobre los datos están lejos de ser meramente casuales.

Palabras Clave: Minería de datos, *Scraping* , Agrupación de *skills*, Agrupación de usuarios, Sistema recomendador, *LinkedIn*.

Abstract

The birth of social networks like Facebook or Twitter has stimulated significantly the use of the WEB by people, creating a medium in which beliefs, judgment and opinions that in some way reflect the real world events are exchanged.

This paper addresses the problem of extracting information or patterns about the set of professionals skills declared by the people in the social network LinkedIn. Unlike Facebook or Twitter, LinkedIn is a platform aimed at creating business and professional contacts, making it an interesting source of data for studies on the workforce. In fact, LinkedIn is widely used by people to seek employment, and by companies to recruit qualified personnel, so it can be assumed that the skills stated there reflect those that the industry considers relevant.

The specific objective of this report is the definition and application of a methodology that helps to understand and to visualize the system of professional competences declared by computer engineers of four Chilean universities in LinkedIn. The working hypothesis is that there are groups of skills that can be identified as different lines of work specialization and that the distribution of these is different among professionals from different universities. Analyzing the hypothesis is relevant, for example, to orient policies on graduation and promotion profile in institutions of higher education.

A contribution of this work is the collection of a database, corresponding to a sample of 2830 computer profiles of LinkedIn and their respective list of skills, in addition to other attributes that could be of interest in future studies. After cleaning and submitting this dataset to a preliminary exploratory analysis, we propose the use of clustering techniques to determine the existence of classes of competencies and users. The performance of different methods,

including hierarchical, spectral and latent variable Topic Models, is studied, addresses the problem of the most adequate representation and the validation of the groups obtained by different metrics. We propose a method to visualize each of the groups found using Word Clouds and to compare groups of users in terms of competencies. A final contribution of this work is the definition and evaluation of a system of self-completeness of competences, based on collaborative filtering methods, which shows an accuracy greater than 90 %. The latter suggests that the structures found on the data are far from merely random.

Keywords: *Data mining, Scraping, Clustering of skills, Clustering of users, Recommender System, LinkedIn.*

Índice de Contenidos

Agradecimientos	III
Resumen	IV
Abstract	VI
Índice de Contenidos	VIII
Lista de Tablas	XIII
Lista de Figuras	XV
Glosario	XX
1. Introducción	1
1.1. Descripción del Problema	3
2. Marco Teórico y Trabajo Relacionado	6
2.1. Descripción plataforma <i>LinkedIn</i>	6
2.1.1. ¿Qué es <i>LinkedIn</i> ?	6
2.1.2. Estructura de la Red	7
2.1.3. Información que contienen los perfiles	8

2.1.4.	Cómo funciona la declaración de <i>Skills</i> y los <i>Endorsement</i>	9
2.1.5.	Cómo funciona la publicación de noticias, artículos , <i>links</i> , etc.	11
2.1.6.	Empleos dentro de <i>LinkedIn</i>	13
2.2.	Investigaciones Anteriores	14
2.2.1.	Análisis de Área de Trabajo de Perfiles de <i>LinkedIn</i>	14
2.2.2.	<i>Endorsement</i> y Ranking de Perfiles de <i>LinkedIn</i>	14
2.2.3.	Resolviendo Problemas Semánticos de <i>Skills</i>	18
2.2.4.	<i>Clustering</i> en perfiles <i>LinkedIn</i>	23
2.3.	Distintos algoritmos <i>Clustering</i>	25
2.3.1.	Métodos de Partición	28
2.3.1.1.	<i>K-Means</i> : Basado en el Centroide	28
2.3.1.2.	<i>K-Medoids</i>	30
2.3.2.	Métodos Jerárquicos	31
2.3.2.1.	Métodos Jerárquicos Aglomerativos y Divisivos	32
2.3.2.2.	Medidas de similitud entre <i>Clusters</i>	32
2.3.2.3.	<i>Chameleon: Multiphase Hierarchical Clustering Using Dynamic Modeling</i>	34
2.3.2.4.	<i>Ward</i>	36
2.3.2.5.	Métricas de Evaluación para Métodos Jerárquicos	37
2.3.2.6.	Métodos Jerárquicos Probabilistas	38
2.3.3.	Métodos Basados en Densidad	39
2.3.3.1.	<i>DBSCAN: Density-Based Clustering Based on Connected Regions with High Density</i>	40
2.3.4.	Criterios de evaluación de un <i>cluster</i>	42
2.3.4.1.	Evaluación de la tendencia de <i>clustering</i>	42
2.3.4.2.	Determinar el número de <i>clusters</i>	43

2.3.4.3.	Determinando la calidad de los <i>clusters</i>	44
2.4.	<i>Topic Models</i>	46
2.4.1.	<i>Latent Semantic Analysis</i>	46
2.4.2.	<i>Latent Dirichlet Allocation</i>	49
2.4.3.	Métricas para evaluar <i>Topic Models</i>	51
2.5.	Técnicas de Recomendación	53
2.5.1.	<i>Non-Negative Matrix Factorization</i>	53
2.5.1.1.	Regularización NMF	55
2.5.2.	<i>Singular Value Decomposition</i>	56
2.5.2.1.	Regularización SVD	57
2.5.3.	Determinando la calidad de los predictores/recomendadores	59
3.	Metodología	61
3.1.	Recolección semi-automática de datos	62
3.2.	Limpieza y representación	64
3.2.1.	Selección de atributos y perfiles a estudiar	64
3.2.2.	Obtención y limpieza de <i>skills</i>	64
3.2.3.	Descripción de los datos obtenidos	65
3.2.4.	Representación de los datos	65
3.2.5.	Normalizaciones	67
3.3.	Minería	68
3.3.1.	Agrupamiento de <i>Skills</i>	68
3.3.2.	Agrupamiento de Perfiles	70
3.3.3.	Predicción de Competencias No Declaradas	72
4.	Experimentos	73

4.1.	Recolección semi-automática de datos	73
4.2.	Limpieza y análisis exploratorio de datos	74
4.2.1.	Selección de atributos y perfiles a estudiar	74
4.2.2.	Obtención y limpieza de <i>skills</i>	76
4.2.3.	Descripción de los datos obtenidos	77
4.3.	Aplicación de técnicas de minería de datos	81
4.3.1.	Agrupamiento de <i>Skills</i>	82
4.3.2.	Agrupamiento de Perfiles	85
4.3.3.	Predictor de Competencias	86
5.	Resultados	87
5.1.	Agrupaciones de <i>skills</i> resultantes	87
5.1.1.	<i>K-Means</i>	88
5.1.2.	<i>DBSCAN</i>	92
5.1.3.	<i>Ward</i>	100
5.1.4.	<i>Topic Models</i>	110
5.1.4.1.	LSA	110
5.1.4.2.	NMF	112
5.1.4.3.	LDA	113
5.1.4.4.	LSA + <i>K-Means</i>	114
5.1.4.5.	LDA + <i>K-Means</i>	117
5.1.5.	Agrupación de <i>Skills</i> : <i>Word Clouds</i>	121
5.1.6.	Grupos Aleatorios	127
5.2.	Agrupaciones de Usuarios	128
5.2.1.	<i>K-Means</i>	128
5.2.2.	<i>DBSCAN</i>	132

5.2.3.	Topic Models	140
5.2.3.1.	LSA	140
5.2.3.2.	NMF	142
5.2.3.3.	LDA	143
5.2.3.4.	LSA + <i>K-Means</i>	144
5.2.3.5.	LDA + <i>K-Means</i>	148
5.2.4.	Agrupación de Usuarios: <i>Word Clouds</i>	152
5.2.5.	Grupos Aleatorios	158
5.3.	Recomendador de <i>skills</i>	159
5.3.1.	NMF	159
5.3.2.	SVD	164
5.3.3.	Recomendador de <i>skills</i> : Demostraciones	167
6.	Conclusiones	173
6.1.	Recolección semi-automática de datos	173
6.2.	Análisis Exploratorio	174
6.3.	Agrupación de <i>skills</i>	174
6.4.	Agrupación de usuarios	175
6.5.	Predictor que permita inferir <i>skills</i> no declaradas en los perfiles	175
6.6.	Conclusiones Generales	176
	Bibliografía	179

Índice de cuadros

2.1. Proporción de estudiantes que trabajan en distintos sectores de la industria según el año de egreso.	14
2.2. Resultados del experimento con un grafo de baja densidad de <i>Endorsement</i>	17
2.3. Resultados del experimento con un grafo de alta densidad de <i>Endorsement</i> .	18
3.1. Datos extraídos de los perfiles de <i>LinkedIn</i>	63
3.2. Representación matricial <i>skills</i> v/s <i>users</i>	66
3.3. Representación matricial <i>users</i> v/s <i>skills</i>	66
4.1. Agrupación realizadas con <i>Wikipedia</i>	76
4.2. Cantidad de <i>skills</i> con poca frecuencia.	79
4.3. Cantidad de usuarios que declaran pocas <i>skills</i>	79
5.1. Número de <i>clusters</i> variando el valor de <i>Epsilon</i> , considerando datos sin normalizar.	96
5.2. Número de <i>clusters</i> variando el valor de <i>Epsilon</i> , considerando datos con normalización L2.	97
5.3. Número de <i>clusters</i> variando el valor de <i>Epsilon</i> , considerando datos con normalización TF-IDF.	97

5.4. Número de <i>clusters</i> variando el valor de distancia de unión mínima, considerando datos sin normalizar.	107
5.5. Número de <i>clusters</i> variando el valor de distancia de unión mínima, considerando datos con normalización L2.	107
5.6. Número de <i>clusters</i> variando el valor de distancia de unión mínima, considerando datos con normalización TF-IDF.	108
5.7. Detalle de técnicas y grupos de <i>skills</i> encontrados.	125
5.8. Número de <i>clusters</i> variando el valor de <i>Epsilon</i> , considerando datos sin normalizar.	137
5.9. Número de <i>clusters</i> variando el valor de <i>Epsilon</i> , considerando datos con normalización L2.	137
5.10. Número de <i>clusters</i> variando el valor de <i>Epsilon</i> , considerando datos con normalización TF-IDF.	138
5.11. Detalle de técnicas y grupos de usuarios encontrados.	156

Índice de figuras

2.1. Agregación de <i>skills</i> en <i>LinkedIn</i>	10
2.2. Visualización de <i>skills</i> en <i>LinkedIn</i>	11
2.3. Opciones de compartir publicación.	12
2.4. Visualización publicación compartida.	12
2.5. Recomendación de usuario en <i>LinkedIn</i>	13
2.6. Ejemplo de oferta de empleo.	13
2.7. Ejemplo de método de <i>Elbow</i>	44
2.8. Factorización realizada en SVD.	48
3.1. Ejemplo de <i>Word Cloud</i>	70
4.1. Interfaz de diagrama de flujo y extracción.	74
4.2. Top 10 de <i>skills</i> con mayor frecuencia en los perfiles.	77
4.3. Frecuencia explicada a través de la integración de <i>skills</i>	78
4.4. Cantidad de <i>skills</i> declaradas por las universidades.	80
4.5. Varianza acumulada en función del número de componentes.	81

5.1. <i>K-Means</i> : Coeficiente de <i>Silhouette</i> en función de la cantidad de <i>clusters</i> considerando la agrupación de <i>skills</i>	88
5.2. Distintas configuraciones de <i>K-Means</i> sin normalizar los datos considerando la agrupación de <i>skills</i>	89
5.3. <i>K-Means</i> : Promedio <i>Coherence</i> con distintas normalizaciones de datos considerando la agrupación de <i>skills</i>	90
5.4. Distribución de las distintas normalizaciones de <i>K-Means</i> considerando la agrupación de <i>skills</i>	91
5.5. <i>DBSCAN</i> : Coeficiente de <i>Silhouette</i> con distintas normalizaciones de datos considerando la agrupación de <i>skills</i>	92
5.6. <i>DBSCAN</i> : Coeficiente de <i>Silhouette</i> con distintas normalizaciones de datos con rango de <i>Epsilon</i> disminuido y considerando la agrupación de <i>skills</i> . . .	93
5.7. <i>DBSCAN</i> : Promedio <i>Coherence</i> con distintas normalizaciones de datos considerando la agrupación de <i>skills</i>	94
5.8. <i>DBSCAN</i> : Promedio <i>Coherence</i> con distintas normalizaciones de datos con rango de <i>Epsilon</i> disminuido y considerando la agrupación de <i>skills</i>	95
5.9. Distintas configuraciones de <i>DBSCAN</i> con distintas normalizaciones considerando la agrupación de <i>skills</i>	99
5.10. Coeficiente de Correlación <i>Cophenetic</i> con distintas normalizaciones de datos.	100
5.11. Dendrograma de 9 niveles.	101
5.12. Grupos obtenidos del dendrograma de 9 niveles.	102
5.13. <i>Ward</i> : Coeficiente de <i>Silhouette</i> con distintas normalizaciones de datos. . .	103
5.14. <i>Ward</i> : Coeficiente de <i>Silhouette</i> con distintas normalizaciones con valor de distancia mínima de unión disminuida.	104

5.15. <i>Ward</i> : Promedio <i>Coherence</i> con distintas normalizaciones de datos.	105
5.16. <i>Ward</i> : Promedio <i>Coherence</i> con distintas normalizaciones de datos con valor de distancia mínima de unión disminuida.	106
5.17. Distintas configuraciones de <i>Ward</i> con distintas normalizaciones.	109
5.18. LSA: Promedio <i>Coherence</i> con distintas normalizaciones de datos.	110
5.19. NMF: Promedio <i>Coherence</i> con distintas normalizaciones de datos.	112
5.20. LDA: Promedio <i>Coherence</i> con distintas normalizaciones de datos.	113
5.21. LSA + <i>K-Means</i> : Coeficiente de <i>Silhouette</i> con distintas normalizaciones de datos.	114
5.22. LSA + <i>K-Means</i> : Promedio <i>Coherence</i> con distintas normalizaciones de datos considerando.	115
5.23. LSA: Distribución de las <i>skills</i> en los distintos <i>clusters</i>	116
5.24. LDA + <i>K-Means</i> : Coeficiente de <i>Silhouette</i> con distintas normalizaciones de datos.	118
5.25. LDA + <i>K-Means</i> : Promedio <i>Coherence</i> con distintas normalizaciones de datos.	119
5.26. LDA: Distribución de las <i>skills</i> en los distintos <i>clusters</i>	120
5.24. Word Clouds de agrupación de <i>skills</i>	124
5.25. Distribución de usuarios en los grupos de <i>skills</i> según universidad.	126
5.26. Métricas con agrupación de <i>skills</i> aleatoria.	127
5.27. <i>K-Means</i> : Coeficiente de <i>Silhouette</i> con distintas normalizaciones de datos considerando agrupación de usuarios.	129
5.28. <i>K-Means</i> : Promedio <i>Coherence</i> con distintas normalizaciones de datos considerando agrupación de usuarios.	130

5.29. Distribución de las distintas normalizaciones de <i>K-Means</i> considerando agrupación de usuarios.	131
5.30. DBSCAN: Coeficiente de <i>Silhouette</i> con distintas normalizaciones de datos considerando agrupación de usuarios.	133
5.31. DBSCAN: Coeficiente de <i>Silhouette</i> con distintas normalizaciones de datos con rango de <i>Epsilon</i> disminuido y considerando agrupación de usuarios. . .	134
5.32. DBSCAN: Promedio <i>Coherence</i> con distintas normalizaciones de datos considerando agrupación de usuarios.	135
5.33. DBSCAN: Promedio <i>Coherence</i> con distintas normalizaciones de datos con rango de <i>Epsilon</i> disminuido considerando agrupación de usuarios.	136
5.34. Distintas configuraciones de DBSCAN con distintas normalizaciones considerando agrupación de usuarios.	139
5.35. LSA: Promedio <i>Coherence</i> con distintas normalizaciones de datos.	140
5.36. NMF: Promedio <i>Coherence</i> con distintas normalizaciones de datos.	142
5.37. LDA: Promedio <i>Coherence</i> con distintas normalizaciones de datos.	143
5.38. LSA + <i>K-Means</i> : Coeficiente de <i>Silhouette</i> con distintas normalizaciones de datos.	144
5.39. LSA + <i>K-Means</i> : Promedio <i>Coherence</i> con distintas normalizaciones de datos.	146
5.39. LSA: Distribución de los usuarios en los distintos <i>clusters</i>	148
5.40. LDA + <i>K-Means</i> : Coeficiente de <i>Silhouette</i> con distintas normalizaciones de datos.	149
5.41. LDA + <i>K-Means</i> : Promedio <i>Coherence</i> con distintas normalizaciones de datos.	150
5.41. LDA: Distribución de los usuarios en los distintos <i>clusters</i>	152

5.41. <i>Word Clouds</i> de agrupación de usuarios	154
5.42. <i>Word Cloud</i> de usuarios con <i>skills</i> de distintas áreas.	155
5.43. Distribución de usuarios en los grupos según universidad.	157
5.44. Métricas con agrupación de usuarios aleatoria.	158
5.45. NMF: <i>Precision</i> del conjunto de entrenamiento y pruebas.	160
5.46. NMF: <i>Recall</i> del conjunto de entrenamiento y pruebas.	161
5.47. NMF: <i>F-Measure</i> del conjunto de entrenamiento y pruebas.	162
5.48. SVD: <i>Precision</i> del conjunto de entrenamiento y pruebas.	164
5.49. SVD: <i>Recall</i> del conjunto de entrenamiento y pruebas.	165
5.50. SVD: <i>F-Measure</i> del conjunto de entrenamiento y <i>test</i>	166
5.51. Distribución de <i>skills</i> recomendadas en función de <i>skills</i> declaradas.	168

Glosario

A

Algoritmo: Conjunto de operaciones sistemáticas que permite hallar una solución a un problema.

Apriori: Se refiere a algo que se realiza con anterioridad.

C

Cluster: Conjunto o agrupaciones de datos que poseen una similitud con datos de la misma agrupación pero diferencias entre datos de otras agrupaciones.

Clustering: Algoritmo o procedimiento de agrupación de datos.

Corpus: Conjunto de texto o documentos que refleja un lenguaje.

D

Data mining: Proceso de extracción, manejo y visualización de datos a través de distintos algoritmos.

Data set: Conjunto de datos.

E

Endorsement: Persona que valida una *skill* de otra persona.

N

Normalización: Estandarización de datos.

O

Outliers: Datos que presentan un comportamiento anómalo en relación de los demás.

Overfitting: Comportamiento de algunos algoritmos de aprendizaje de adaptarse demasiado a los datos conocidos perdiendo la capacidad de tener un comportamiento óptimo con datos nuevos.

S

Skills: Traducción de aptitud o habilidad que posee una persona.

U

Users: Traducción de usuario.

Siglas

DBSCAN: *Density-Based Spatial Clustering of Applications with Noise.*

LDA: *Latent Dirichlet Allocation.*

LSA: *Latent Semantic Analysis.*

NMF: *Non-negative Matrix Factorization.*

SVD: *Singular Value Decomposition.*

TF-IDF: *Term Frequency – Inverse Document Frequency.*

Capítulo 1

Introducción

En el conocido texto introductorio “*Data Mining: Practical Machine Learning Tools and Techniques*”, *Witten* y coautores definen la Minería de Datos como “la extracción de información implícita, previamente desconocida, y potencialmente útil a partir de datos” [25]. La idea es sencilla: una masiva colección de datos acerca del mundo, no garantiza que lo entendamos. Usualmente será necesario transformar esa simple colección de registros en una estructura o modelo que permita su entendimiento total o parcial con la finalidad hacer inferencias y tomar decisiones. A la definición de *Witten* se debe agregar el rol preponderante que hoy en día tiene en este proceso la informática y la inteligencia artificial. Si hace algunas décadas, el análisis de datos podía ser realizado con un altísimo grado de intervención humana, el tamaño de las bases de datos actuales hace necesario el diseño de algoritmos capaces de detectar automáticamente patrones y estructuras, probablemente inexactas, pero valiosas por su carácter explicativo o predictivo.

Ejemplos claros de bases de datos suficientemente grandes como para hacer inviable su análisis por simple inspección humana son aquellas generadas en redes sociales como *Facebook* o *Twitter*. El nacimiento de redes sociales como *Facebook* o *Twitter* ha estimulado significativamente la utilización de la WEB por parte de las personas, creando un medio en el cual se intercambian creencias, juicios y opiniones que de alguna forma reflejan los eventos del mundo real. Ese enorme flujo de comunicaciones representa una fuente de datos en constante expansión y actualización, que puede ser utilizada por diversos actores en la toma de

decisiones.

En esta memoria se aborda el problema de extraer información o patrones sobre el conjunto de competencias profesionales declaradas por las personas en la red social *LinkedIn*. A diferencia de *Facebook* o *Twitter*, *LinkedIn* es una plataforma orientada a la creación de contactos profesionales y de negocios, lo que la transforma en una fuente interesante de datos para estudios sobre la fuerza de trabajo. En efecto, *LinkedIn* es ampliamente utilizada por las personas para buscar empleo, y por las empresas para reclutar personal calificado, por lo que se puede asumir que las competencias allí declaradas reflejan aquellas que la industria considera relevantes.

El objetivo específico de esta memoria es la definición y aplicación de una metodología que ayude a entender y a visualizar el sistema de competencias profesionales (*skills*) declaradas por ingenieros informáticos de cuatro universidades chilenas en *LinkedIn*. La hipótesis de trabajo es que existen grupos de competencias que puedan ser identificados como diferentes líneas de especialización laboral y que la distribución de éstas es diferente entre profesionales de distintas universidades. Analizar esta hipótesis resulta relevante, por ejemplo, para orientar políticas sobre perfil de egreso y promoción en instituciones de educación superior.

Una contribución de este trabajo, es el levantamiento de una base de datos, correspondiente a una muestra de 2830 perfiles informáticos de *LinkedIn* y sus respectivas listas de competencias, además de otros atributos que podrían ser de interés en futuros estudios. Las distintas competencias obtenidas a través de la declaración de los perfiles fueron sometidas a procesos que permitieron eliminar en gran medida los problemas de duplicación existentes.

Después de limpiar y someter este conjunto de datos a un análisis exploratorio preliminar, se propone la utilización de técnicas de *clustering* para determinar la existencia de clases de competencias y usuarios.

Por otro lado, se estudia el desempeño de diferentes métodos, incluyendo algoritmos jerárquicos, espectrales y de variable latente (*Topic Models*), abordando el problema de la representación más adecuada y la validación de los grupos obtenidos mediante diferentes métricas. El rendimiento estos métodos fueron medidos a través de métricas intrínsecas, ya que no se conoce la agrupación óptima de los datos.

Luego, se propone un método para visualizar cada uno de los grupos encontrados usando *Word Clouds* y comparar grupos de usuarios en términos de competencias. Este método de visualización se aplica de distinta manera dependiendo del algoritmo que encontró la agrupación, por ejemplo, para los algoritmos que generan agrupaciones planares y en cuyos datos internos no existe algún tipo de jerarquía, se utilizó información obtenida a partir del *dataset* con el que se está trabajando. Considerando lo anterior es que también se varió el criterio para agregar palabras a los *Word Clouds*.

Una contribución final de este trabajo es la definición y evaluación de un sistema de auto-completación de competencias, basado en métodos de filtrado colaborativo, que muestra una precisión superior al 90 %. Esto último sugiere que las estructuras encontradas sobre los datos están lejos de ser meramente casuales.

1.1. Descripción del Problema

Caracterizar el perfil profesional, es decir, el conjunto de conocimientos y competencias que poseen o debiesen poseer los egresados de un determinado programa de estudios, resulta fundamental para evaluar la coherencia de éste con las expectativas y necesidades de sus beneficiarios, es decir, los estudiantes mismos, la academia, el sector productivo y la sociedad en general. Debido a la diversidad de los actores involucrados, la elaboración y evaluación de un perfil profesional atractivo y realista, es una tarea compleja que requiere explorar, entender e integrar diferentes criterios e intereses, entre los que se encuentran: las necesidades actuales y futuras de diversos sectores de la industria, el grado de diferenciación, la tradición y estándares académicos internacionales, etc.

El objetivo general de esta memoria consiste en explorar la utilización de herramientas modernas de análisis de datos sobre la red social *LinkedIn* que permitan obtener información acerca de las competencias que los profesionales de un determinado plan de estudios dicen tener. Más concretamente, esta memoria se propone estudiar los perfiles de alumnos del departamento de informática de la Universidad Técnica Federico Santa María y otras instituciones de educación superior.

LinkedIn es una red social orientada específicamente a la creación de contactos profesionales y oportunidades de negocio. Si bien, en términos de usuarios netos, *LinkedIn* está muy por debajo de otras redes sociales como *Facebook* o *Twitter*, estudios recientes sugieren que en Chile y Latinoamérica en general *LinkedIn* es la red social más utilizada con fines profesionales por personas y empresas^{1 2}, situación que la transforma en una fuente interesante de datos para estudios sobre la fuerza de trabajo. En efecto, *LinkedIn* es muy utilizada por los profesionales como una vitrina para darse a conocer, encontrar o cambiar de trabajo, así como por las compañías para reclutar personal altamente calificado. Debido a esta dinámica, los perfiles de esta red social contienen datos explícitos acerca de las competencias de una persona, su experiencia laboral, sus estudios y sus redes de contacto; datos que organizados y procesados adecuadamente pueden dar luces importantes acerca de las competencias que efectivamente tienen los profesionales de un determinado sector y/o aquellas que la industria considera altamente relevantes actualmente.

La utilización de métodos y herramientas que permitan recoger y analizar de modo eficiente grandes volúmenes de datos es una habilidad que se ha demostrado fundamental para la competitividad y el éxito de las organizaciones modernas. Esta necesidad ha gatillado la convergencia de muchas disciplinas (estadística, bases de datos, inteligencia artificial, computación de alto desempeño, visualización, procesamiento de lenguaje natural, etc.) en un área denominada ciencia de datos, cuyo objetivo es, al igual que en minería de datos, la utilización de modelos, algoritmos y plataformas de cómputo en la transformación de datos en información útil para la comprensión de un determinado fenómeno, pero que a diferencia de otras, pone gran énfasis en la recolección de datos desde fuentes heterogéneas; la combinación de diferentes modelos en la consecución de un objetivo; y la interpretación de los resultados a la luz de las necesidades del dominio de aplicación. En este contexto, esta memoria se puede entender como un caso de estudio en que estas herramientas y metodología se pondrán al servicio del descubrimiento de patrones no evidentes acerca de las competencias evidenciadas por alumnos de este departamento, siendo la comprensión de la fuente de datos y la interpretación de los resultados, aspectos cruciales del proyecto.

Para comenzar este estudio se partirá por estudiar el sistema de declaración y validación

¹<http://www.gointegro.com/wp-content/uploads/2015/02/GOintegro-Insights-Cultura-ESP.pdf>

²<http://www.iab.cl/primer-informe-social-recruiting-chile/>

de competencias en *LinkedIn*, analizando el potencial de los datos disponibles. También se ejecutará un análisis exploratorio de datos que permita describir las competencias más populares en *LinkedIn*.

Por otro lado, también se desea diseñar y ejecutar una metodología de extracción de datos (*WEB scraping*) de *LinkedIn* que permita generar una base de datos de perfiles correspondientes a alumnos DI UTFSM y de otras instituciones de educación. También se desea detectar grupos de competencias relacionadas entre sí y distinguibles del resto, afrontando problemas de sinonimia y de contexto. Otro punto importante es determinar si existen grupos de perfiles relacionados entre sí y distinguibles del resto en términos de competencias. Por otro lado se construirá un predictor que permita inferir competencias no declaradas por usuarios a partir de aquellas declaradas.

Capítulo 2

Marco Teórico y Trabajo Relacionado

2.1. Descripción plataforma *LinkedIn*

Para contextualizar el problema, se procederá a definir qué es *LinkedIn* y cómo funciona la plataforma social orientada a profesionales.

2.1.1. ¿Qué es *LinkedIn*?

Según el sitio web *LinkedIn*¹, “*LinkedIn* es un servicio *online* que ayuda a los profesionales a buscar y conectarse con otros profesionales más efectivamente. Ya sea en busca de trabajo o buscar un experto en la industria, usuarios pueden conectarse con miles de profesionales a través de una cadena de conexiones.”

Otra definición que engloba lo más destacable, es la brindada por *iTunes*²:

“Todo comienza con tu perfil de *LinkedIn*. Es mucho más que tu curriculum: muestra al mundo quien eres y lo que has hecho en tu trayectoria profesional.

LinkedIn permite:

¹https://www.linkedin.com/help/linkedin?trk=neptune_help

²<https://itunes.apple.com/cl/app/linkedin/id288429040?mt=8>

- Buscar personas, empleos empresas y grupos.
- Ver actualizaciones de las personas, editores y empresas que pueden ayudarte a alcanzar el éxito profesional.
- Ampliar tu red profesional y mantener el contacto.
- Poner en contacto distintas personas y ver sus perfiles profesionales.
- Compartir artículos y posicionarse como líder de opinión en el sector profesional.
- Seguir a empresas para ver anuncios de empleo, actualizaciones y sugerencias para conectar con personas que conoces.”

En pocas palabras, *LinkedIn* es una red social de profesionales, en la cual se pueden crear distintos tipos de perfiles con el fin de poder conectarse con distintos profesionales. En esta red social cada usuario posee la libertad de ingresar toda su carrera laboral, sus competencias (**desde ahora skills**), sus intereses, compartir noticias de distinta índole, etc. La finalidad de esta red es crear una gran malla de profesionales para así poder compartir de manera más expedita todo lo que tenga que ver con el ámbito profesional.

2.1.2. Estructura de la Red

LinkedIn funciona con el principio básico de las redes sociales, donde las personas poseen un perfil y éstos pueden seguir o ser seguidos por otros usuarios.

Haciendo un análisis más profundo y en relación a la plataforma, existen 2 formas de generar un contacto con otros perfiles: El usuario puede conectar a otro usuario o se le puede enviar un mensaje directo. Respecto a la conexión de los usuarios estas tienen distintos grados de profundidad:

- Primer Grado: En este grupo se encuentran los usuarios que están conectados directamente, ya sea aceptando la invitación de alguien o que alguien acepte su invitación.

- Segundo Grado: Son los usuarios que están conectados (en primer grado) con los contactos de primer grado.
- Tercer Grado: Son los usuarios que están conectados con los usuarios de segundo grado. En este caso existen 2 tipos de usuarios de tercer grado: Los que tienen nombre completo los cuales se pueden conectar y los que tienen solo la primera letra del nombre los cuales solo se les puede enviar mensajes.
- Compañero de Grupos: Estos son usuarios que pertenecen a un mismo grupo, por ejemplo, usuarios pertenecientes al grupo **Universidad Técnica Federico Santa María**. Un grupo en *LinkedIn* son usuarios profesionales que tienen los mismos intereses en común.
- Fuera de la red: Son usuarios que se pueden contactar solo enviando un mensaje privado.

En resumen la red de conexiones de *LinkedIn* es un grafo, donde los nodos son los usuarios y las aristas son el tipo de conexión que tienen entre éstos.

2.1.3. Información que contienen los perfiles

Como se mencionó en la sección 2.1.2 existen distintos tipos de conexiones, por ende, cierta información no estará disponible para cierto tipo de conexiones. En esta ocasión se mostrará la información disponible en los perfiles públicos de primer grado de conexión.

Información que se puede ingresar al perfil:

- Información Personal: Nombre Completo, Fecha de Nacimiento, País, Estado Civil y *Mail* de Contacto.
- Educación: Fecha de estudio, Titulación, Disciplina Académica, Nota, Actividades y Grupos.
- Experiencia: Nombre de la empresa, Título, Ubicación y Período.

- *Skills y Endorsement*: Son las *skills* que creen tener los usuarios. En la sección 2.1.4 se dará más detalle sobre esto.
- Grupos para seguir.
- A continuación se darán a conocer otras secciones que no dan cabida en los ítems anteriores, esta sección será llamada como Otros: Extracto, Experiencia voluntariado, Oportunidades de voluntariado, Organizaciones, Reconocimientos y premios, Calificaciones de pruebas, Cursos, Patentes, Causas benéficas que importan, Organizaciones que apoya, Publicaciones, Certificaciones, Intereses y Consejos para ponerse en contacto.

Información que se puede obtener de un perfil público con una conexión de primer grado:

- Información Personal: Nombre, Profesión y País.
- Experiencia: Cargo, Ubicación, Fecha y Descripción.
- Educación: Fecha de estudio, Titulación, Disciplina Académica y Nota.
- *Skills y Endorsement*.
- Idiomas.
- Otros.

En los perfiles de segundo y tercer grado también es posible encontrar las características más importantes, tales como nombre, educación y *skills*.

2.1.4. Cómo funciona la declaración de *Skills* y los *Endorsement*

En esta sección se verá cómo los usuarios pueden ingresar sus *skills* para luego poseer más *endorsement*. También se verá cómo desde un perfil público se puede observar las *skills* y cómo formar parte de los *endorsement* de esa *skill*.

Ingresar *Skills* y poseer *Endorsement*:

Al ingresar las *skills*, éstas pueden ser seleccionadas de un listado recomendado o ser inventadas por el mismo usuario. La figura 2.1 muestra cómo el sistema despliega la información de las *skills*.

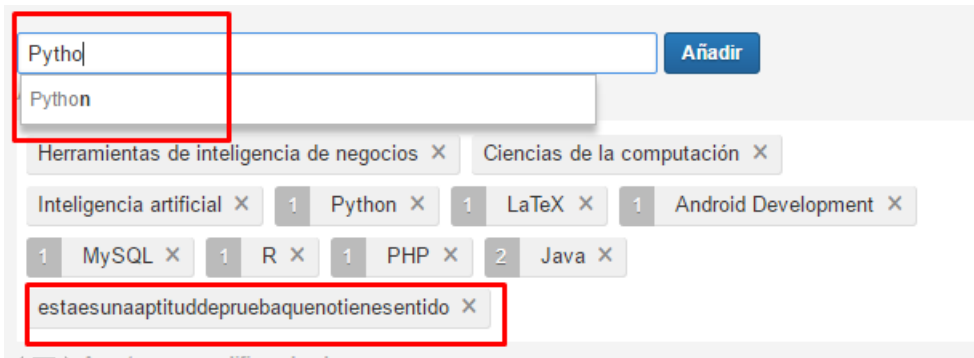


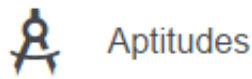
Figura 2.1: Agregación de *skills* en *LinkedIn*.

En el primer recuadro de la figura 2.1 se puede observar una *skill* recomendada por *LinkedIn*. En el segundo recuadro se puede observar que se agregó una *skills* generada por el usuario. Cada usuario solo puede agregar a lo más 50 *skills*.

Estos usuarios también tienen la opción de ser o no ser validados (sistema *endorsement*) por sus conexiones, incluirse en las sugerencias de validación de sus contactos y mostrar las sugerencias para validar las *skills* de sus contactos. Solo los contactos de primer grado pueden validar las aptitudes.

Ver *skills* de contactos y validar sus *skills*:

Las *skills* se encuentran dentro del perfil de un usuario y se visualizan como se muestra en la figura 2.2



Aptitudes principales

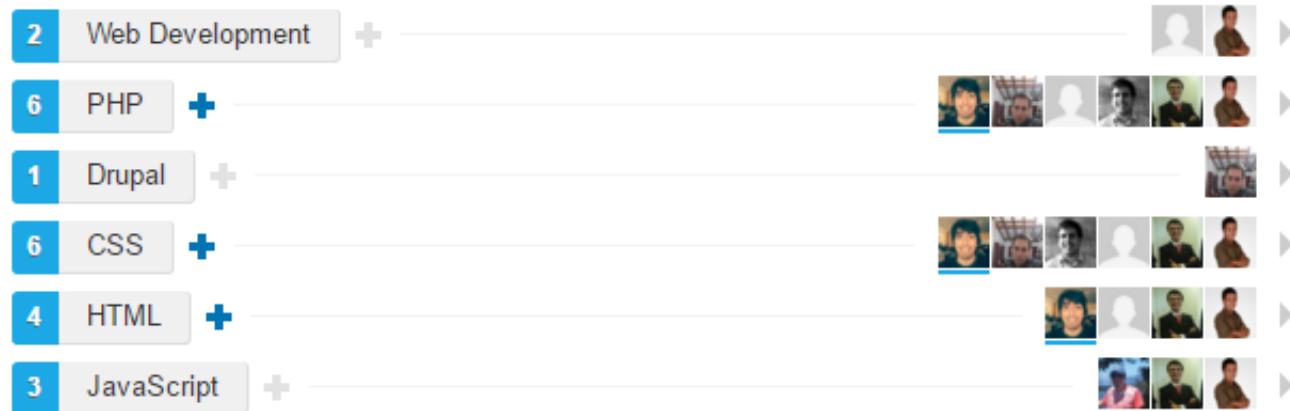


Figura 2.2: Visualización de *skills* en *LinkedIn*.

Las *skills* puede ser vistas por todos los grados de usuarios, es decir, desde contactos de primer grado hasta usuarios de tercer grado.

Para validar las *skills* simplemente basta con ser contacto de primer grado y validar la *skills* a través de un botón (no es necesario llenar formulario o descripciones).

Las personas que validarán las competencias serán llamadas *endorsement*.

2.1.5. Cómo funciona la publicación de noticias, artículos , *links*, etc.

Los perfiles públicos tienen distintas acciones para utilizar en *LinkedIn* respecto a la manera de compartir información. Las más básicas son: publicar, recomendar y compartir.

- **Publicar:** En esta opción la persona puede escribir alguna noticia, mostrar alguna URL o subir alguna fotografía. Estas publicaciones pueden ser vistas por distintos usuarios así lo desee el que comparte la publicación. Las opciones para compartir son 3: público, solo tus contactos o público + *Twitter*.

- **Compartir:** En *LinkedIn* también se pueden compartir publicaciones de otros usuarios. Estas publicaciones compartidas se visualizan como si el usuario mismo hubiera realizado la publicación, es decir, no existe relación con el usuario del cual fue compartida la noticia. En las figuras 2.3 y 2.4 se puede observar cómo se publica la noticia y el cómo se visualiza.

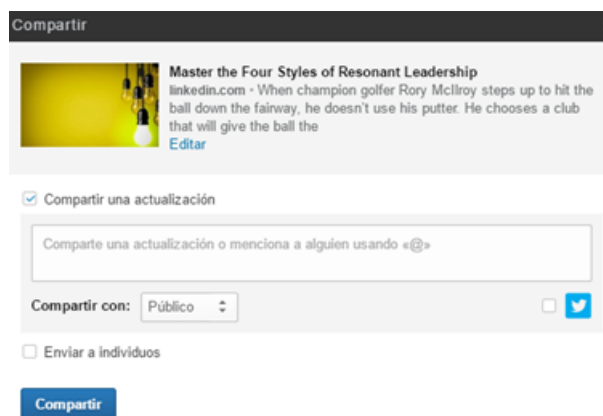


Figura 2.3: Opciones de compartir publicación.



Figura 2.4: Visualización publicación compartida.

- **Recomendar:** En *LinkedIn* existen distintos tipos de recomendaciones. Las recomendaciones hechas a los artículos son análogos a los *likes* que existen en *Facebook*, es decir, es una aprobación de dicho artículo por parte de algún usuario. También existe la acción de recomendar usuarios, que al igual que un Curriculum Vitae, es una opinión (generalmente positiva) de parte del empleador hacia el profesional y/o al trabajo que realizó. En la figura 2.5 se observa una recomendación de la plataforma *LinkedIn*

Coordinador de Emprendimiento y Capital Humano

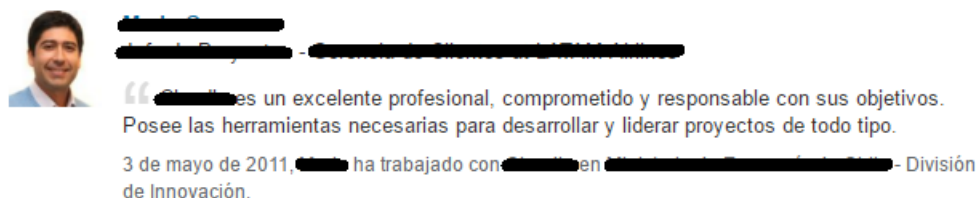


Figura 2.5: Recomendación de usuario en *LinkedIn*.

2.1.6. Empleos dentro de *LinkedIn*

Los empleos que se brindarán en *LinkedIn* son una gran fuente de información para observar cómo está funcionando el mercado actual respecto a los profesionales que se requieren en un cierto periodo de tiempo. En una oferta de empleo que cada empresa utiliza cuando busca futuros empleados, explicita qué habilidades son necesarias para cierto trabajo. En *LinkedIn* las publicaciones de empleo tienen una estructura definida al ser publicadas. En la figuras 2.6 se observa una oferta de empleo que contiene los siguientes campos: Descripción, Aptitudes y Conocimientos deseados, Sector, Tipo de Empleo, Experiencia y Función Laboral.

Descripción de empleo Empresa líder en el rubro de la construcción, requiere incorporar a Subgerente de Innovación e Industrialización, quien será el encargado de liderar los procesos de construcción, mediante el desarrollo de propuestas innovadoras, las cuales sean altamente eficientes, de vanguardia y estén relacionadas con las necesidades del. Es importante resaltar, que estamos en la búsqueda de un profesional el cual cuente con experiencia en el rubro de la construcción y tenga las competencias esenciales para el puesto, las cuales serán evaluadas en un proceso de selección que se realizará en la IV región.	Sector Recursos humanos
Aptitudes y conocimientos deseados Los requisitos excluyentes para el cargo son: 1) De 3- 5 años de experiencia liderando procesos de mejoramiento continuo en el área de construcción inmobiliaria. 2) Post título en área de construcción. 3) Manejo de inglés fluido (Deberá participar en ferias de internacionales de construcción) 4) Dominio avanzado de AutoCAD Y MS Project 5) Título de Ingeniero Civil, Ingeniero en Construcción Civil o carrera afin 6) Residencia IV región	Tipo de empleo Jornada completa Experiencia Mando intermedio Función laboral Otro

Figura 2.6: Ejemplo de oferta de empleo.

2.2. Investigaciones Anteriores

2.2.1. Análisis de Área de Trabajo de Perfiles de *LinkedIn*

Case y compañía [2] decidieron investigar sobre el cómo los años de egresado en cierta disciplina correspondiente al área de Sistemas de Información (*Information System*), afecta al progreso de su carrera profesional en el ámbito laboral. Para la recolección de datos fue necesario utilizar la base de datos de la universidad que realiza el estudio, para así, con estos mismos datos, realizar una búsqueda de perfiles en la red social *LinkedIn*. Con este método se logró obtener 175 perfiles. Los datos principales que se utilizaron fue el año de graduación de cada persona y el empleo actual de éste. Considerando esto, la descripción del *data set* arrojó que 85 personas se graduaron entre el 2000 y el 2008, mientras que las personas restantes lo hicieron entre 1982 y el 2008. Los resultados de la investigación se resumen en la tabla 2.1.

Cuadro 2.1: Proporción de estudiantes que trabajan en distintos sectores de la industria según el año de egreso.

	1° Año	5° Año	10° Año	15° Año
	Cantidad	Cantidad	Cantidad	Cantidad
Técnicos	86 - 55 %	46 - 36,8 %	57 - 83,38 %	28 - 82,4 %
Gestión	69 - 44,5 %	79 - 63,2 %	11 - 16,2 %	69 - 17,6 %
Total	175	125	68	34

Como conclusión respecto a estos resultados, se puede decir que los años de egresado afectan directamente a las *skills* que posee la persona, ya que éstas están directamente ligadas con el papel desempeñado en el mundo laboral.

2.2.2. *Endorsement* y Ranking de Perfiles de *LinkedIn*

En el año 2015, los investigadores *Hebert Pérez* y *Josep Ma. Ribó* [17] realizaron una investigación basada en las *skills* que declaran los usuarios en la plataforma *LinkedIn*. Ellos en

su investigación realizan 2 aportes importantes: El primero consiste en enriquecer la información del grafo de *endorsement* correspondiente a una habilidad en particular. El segundo aporte fue aplicar un algoritmo de *ranking* al grafo obtenido con el primer acercamiento.

A continuación se explicará el cómo se realizó cada acercamiento de esta investigación:

Para la primera investigación hay que considerar que para cada *skill* habrá un grafo de conexiones de *endorsement*. El grafo que se construirá a continuación no solo contiene la información de qué persona recomendó la *skill* de otra persona, sino que también se considerará la información de las *skills* relacionadas a éstas y cómo han sido recomendadas por otros usuarios.

Para esto hay que considerar los siguientes elementos:

- $S = s_0, s_1, \dots, s_l$: conjunto de todas las habilidades posibles.
- D_k : grafo de *endorsement* de la *skill* s_k .
- M_k : matriz de adyacencia que representa D_k .

Por último se define la matriz $\Pi = (\pi_{kt})$ como lo siguiente: Dado un par de *skills* s_k y s_t , π_{kt} representa la probabilidad que una persona con la *skill* s_k también sea poseedora de la *skill* s_t . En otras palabras se puede decir que desde s_k se puede inferir s_t con una confianza de π_{kt} . Entonces, si un usuario recomienda una *skill* s_k pero no realiza la recomendación para s_t , se puede deducir que esa persona debió haber recomendado esa *skill* con probabilidad π_{kt} . El propósito de esto es poder unir la información otorgada por Π y D_k . La unión de ambas representaciones se denominará D_k^{we} . La construcción del grafo se realiza de la siguiente manera, considerando que el grafo se está realizando en base a la *skill* s_0 :

1. Si un usuario v_i recomienda directamente la *skill* s_0 (*skill* en estudio) de un usuario v_j , entonces en el grafo D_0^{we} el arco (v_i, v_j) tendrá el valor 1, es decir, máxima confianza.
2. Si el usuario v_i no recomienda la *skill* s_0 de un usuario v_j , pero existe solo una recomendación de otra *skill* disponible del usuario v_i al usuario v_j , entonces el grafo D_0^{we}

tendrá un peso igual a π_{k0} , es decir, se tendrá un peso igual a la probabilidad de que el usuario v_i también considere recomendar la *skill* s_0 al usuario v_j .

3. Finalmente, si el usuario v_i no recomienda la *skill* s_0 de un usuario v_j , pero existe más de una recomendación de otras *skills* disponibles del usuario v_i al usuario v_j , entonces el peso del arco D_0^{we} será la unión de todas las probabilidades otorgadas por Π (se considera que cada recomendación son eventos independientes y a la vez son sucesos compatibles).

A continuación se construye iterativamente la matriz de adyacencia del grafo D_0^{we} .

$$Q_0 = M_0 \tag{2.1}$$

$$Q_k = Q_{k-1} + \pi_{k0}((J^{(n)} - Q_{k-1}) \circ M_k). \quad \forall k=1, \dots, l.$$

Donde $J^{(n)}$ será la matriz identidad de orden n y el símbolo $\circ$ representa el producto de *Hadamard* de matrices.

En la iteración k , la matriz Q_k corresponderá al grafo de conexiones de s_0 luego de haber agregado la información deducida de las *skills* $s_1, \dots, s_k$. Finalmente la matriz Q_l corresponderá a la matriz de adyacencia del grafo D_0^{we} .

Una vez que se tiene la matriz Q_l se construye la matriz normalizada P , donde:

$$p_{ij} = \begin{cases} \frac{q_{ij}}{\sum q_{i*}} & \text{si } \sum q_{i*} > 0 \\ \frac{1}{n} & \text{si } \sum q_{i*} = 0 \end{cases} \tag{2.2}$$

Donde n corresponde a la cantidad de usuarios que habrán en el estudio.

Luego a partir de la matriz P se procede a crear la matriz P_α :

$$P_\alpha = \alpha P + (1 - \alpha) \frac{1}{n} J^{(n)} \tag{2.3}$$

Donde α es una probabilidad que se conoce como *damping factor*, en esta investigación se utilizó un $\alpha = 0,85$.

Respecto a los experimentos y resultados obtenidos por los investigadores, ellos utilizaron una red social artificial y aleatoria que es muy similar a *LinkedIn*. Ellos generaron un *data set* de 1493 usuarios y 2489 conexiones entre éstos. Además, las *skills* que consideraron fueron las siguientes: 1. Programación, 2. C++, 3. Java, 4. Modelos Matemáticos y 5. Estadística. También al *data set* se agregó un grupo especial de usuario que se denominan *spammers* los cuales se dedican a promocionar a una persona denominada líder. Se decide agregar este tipo de usuario ya que afecta directamente en el *ranking* de toda la red generada.

Las principales comparaciones que ellos realizaron se basaron en 2 redes, una con baja densidad de *endorsement* y otra con alta densidad de *endorsement*. Se observa cómo la deducción de *endorsement* afecta a la posición de los ya descritos líderes en el *ranking* y cómo afecta en el número de relaciones que se observa en el grafo generado por cada *skill*.

Cuadro 2.2: Resultados del experimento con un grafo de baja densidad de *Endorsement*

Skill	Número de Endorsements	Número de lazos			Posición del líder		
		Sin Deducción	Con Deducción	% Reducción	Sin Deducción	Con Deducción	% Reducción
Programación	220	1460	1316	10 %	1	48	3 %
C++	140	1478	1304	12 %	4	48	3 %
Java	137	1486	1292	13 %	1	48	3 %
Modelos Matemáticos	134	1483	1318	11 %	1	45	3 %
Estadística	128	1486	1304	12 %	1	45	3 %

Cuadro 2.3: Resultados del experimento con un grafo de alta densidad de *Endorsement*

Skill	Número de Endorsements	Número de lazos			Posición del líder		
		Sin Deducción	Con Deducción	% Reducción	Sin Deducción	Con Deducción	% Reducción
Programación	427	1428	625	56 %	1	175	12 %
C++	1793	1005	575	43 %	66	178	7 %
Java	1856	1005	566	44 %	63	180	8 %
Modelos Matemáticos	1406	1130	652	42 %	56	168	7 %
Estadística	1447	1113	580	48 %	58	169	7 %

Lo que se puede observar de los resultados expuestos en las tablas 2.2 y 2.3, es cómo el sistema de deducción de *skills* propuestos por los investigadores, puede manejar adecuadamente el problema de los *spammers* y su líder. Acá se evidencia que la posición de líder en los *rankings* decae drásticamente (a diferencia de *PageRank*, en este *ranking* un mayor valor indica peor posición) utilizando la deducción de *skills*.

2.2.3. Resolviendo Problemas Semánticos de *Skills*

Bastian y compañía [1], presentan una técnica para la inferencia y recomendación de *skills* basados en un perfil de *LinkedIn*. Para esto, los temas que ellos proponen se dividen en 2 partes, la primera hace referencia a la extracción y la calidad de los datos, mientras que la segunda parte hace referencia hacia la técnica utilizada para realizar el recomendador en base a los datos obtenidos del trabajo anterior. A continuación se precede a describir los temas descritos por los autores.

En la primera parte de esta investigación los autores optan por realizar una **folksonomía** de *skills*, es decir, tratan de crear relaciones no jerárquicas entre las distintas *skills* presentes entre los usuarios. Para la creación de esta **folksonomía** los investigadores analizaron los datos y se encontraron con 2 problemas graves dentro de este conjunto de *skills*. El primer problema consiste en el significado duplicado de cada *skill*, por ejemplo, un usuario puede declarar la *skill* “Programación Java” y a su vez declarar “Desarrollador Java”. El problema

que surge de esto es que ambas *skills* se consideran como distintas pero en el fondo pertenecen a una misma *skill* denominada “Java”. El otro problema importante es la ambigüedad de *skills*, por ejemplo, un usuario puede declarar la habilidad “Python” pero esta puede referirse tanto como un desarrollador del lenguaje programación “Python” como de un entrenador de serpientes.

Siguiendo la linea investigadora de los autores, ellos para crear la **folksonomía** dividieron el primer trabajo en 3 partes:

1. Descubrir: Para esta investigación se trabajó en párrafos de texto libre, por ende, para obtener las posibles *skills* fue necesario un proceso de *parcing* de las *skills* que estén dentro de estos párrafos. Luego se realiza un contador de frecuencias para desechar las *skills* que tienen una frecuencia baja respecto a las demás. Realizando solo este paso se recuperaron 150.000 *skills* donde el 6 % de éstas representa el 80 % de las ocurrencias.
2. Desambiguación: Como se discutió anteriormente, una *skill* puede tener distintos significados igualmente válidos dependiendo del área de experticia. Para solucionar este problema se consideró la relación existente entre una *skill* y sus posibles vecinos. En este caso los vecinos serán el conjunto de *skills* declaradas en conjunto a la *skill* que se desea analizar. Finalmente se aplica una técnica *cluster* para distinguir las distintas áreas o grupos que se pueden generar en base a estas *skills*. A continuación se entrará más en detalle sobre el cómo se construyó esta fase.

Considerar 2 conjuntos distintos: $P = \{p_1, p_2, \dots, p_t\}$ y $D = \{d_1, d_2, \dots, d_{t'}\}$. Donde p representan las *skills* estudiadas y d representan a los usuarios dentro de la red en estudio. Considerar también $f_{i,k}$ como la ocurrencia de la *skill* p_i en el usuario d_k .

La cantidad de ocurrencias de p_i en el conjunto de usuarios D se define como:

$$F_i = \sum_{k=1}^{t' \in D} f_{i,k} \quad (2.4)$$

Ahora, $f_{i,j,k}$ será la co-ocurrencia de las *skills* p_i y p_j en el mismo perfil d_k .

La cantidad de co-ocurrencias p_i y p_j en el conjunto de usuarios D se define como:

$$F_{i,j} = \sum_{k=1}^{r' \in D} f_{i,j,k} \quad (2.5)$$

Y la similitud de *Jaccard* entre 2 *skills* se calcula como:

$$S_{i,j} = \frac{F_{i,j}}{F_i + F_j - F_{i,j}} \quad (2.6)$$

Luego por cada *skill* p_i se genera un vector R_i de largo n , donde n será la cantidad de *skills* más relacionadas a la *skill* p_i en base a la similitud de Jaccard, mencionada anteriormente. Luego para cada *skill* p_i se genera la matriz M_i donde las columnas y filas corresponden a los elementos de la matriz R_i y el valor de la celda será la similitud de Jaccard entre las *skills* más relacionadas de la *skill* p_i . Luego de generar la matriz M_i se procede a calcular la similitud de coseno en cada celda de la matriz M_i , resultando así la matriz N_i donde cada celda contiene la similitud de coseno entre cada *skill* p_j más relacionadas de la *skill* general p_i . Cabe destacar que en esta investigación, la generación de la matriz M_i no es necesaria ya que solo con los vectores R_i se es capaz de generar la matriz N_i .

Finalmente se procede a realizar una técnica *clustering* llamada *K-Means* a la matriz N_i para realizar una agrupación de cada *skill* que se tiene disponible. El mejor K fue seleccionado en base a la relación entre *clusters* y la penalización de poseer muchos *clusters*. Luego de obtener los mejores *clusters* en base a la relación mencionada anteriormente, se procede a etiquetar cada *cluster* generado, para eso se etiquetó el *cluster* con la industria más frecuente dentro de los usuarios que tienen la *skill* perteneciente a ese *cluster*.

3. Duplicación: Como se vio anteriormente una *skill* puede ser escrita de distintas maneras, pero significar lo mismo, por ejemplo, “Desarrollador Java” y “Programador Java”. En esta investigación para solucionar este problema, se utilizó como comparador la página *Wikipedia*. Si 2 habilidades son dirigidas a la misma página de *Wikipedia*, se podría considerar como la misma habilidad. Si ambas habilidades son la misma, se

procede a unir ambas y se etiqueta la nueva *skill* con el nombre de la *skill* que tuviera mayor frecuencia dentro de los usuarios en estudio.

Luego de solucionar los 3 problemas descritos, los experimentadores quedaron con aproximadamente 50.000 *skills* para analizar.

Como segundo punto importante se tiene la creación de un sistema de inferencia de competencias basado en la técnica *Naive Bayes*. A continuación se presenta el cómo se aplicó esta técnica de clasificación para poder crear un sistema de inferencia (o recomendación).

Lo primero que realizan los investigadores es seleccionar los atributos con los cuales se realizarán las recomendaciones, en este caso los atributos fueron los siguientes: Compañía, Título, Industria, Área de estudio o grupos a los cuales pertenece. Luego de la selección se procede a la estandarización de estos atributos, esto con el fin de acotar un poco el dominio de estos mismos. Luego, dado un perfil con identificador único y atributos $a_1...a_n$, se define como probabilidad de tener una *skill* como: $P(s|a_1...a_n)$ donde $s \in S$ y $S = \{s_1, s_2, ..., s_n\}$ es la **folksonomía** de *skills* obtenidas en la sección anterior. Por otro lado se tiene $n(a_i, s)$ el número de co-ocurrencias donde un usuario posee el atributo a_i y la *skill* s , por otro lado, $n(s)$ será el número total de usuarios que poseen esa *skill*. A continuación se define la probabilidad de tener una habilidad s dado que se tienen ciertos atributos:

$$P(s|a_1, ..., a_n) = \frac{1}{Z} P(s) \prod_{i=1}^n P(a_i|s) \quad (2.7)$$

Donde $Z = P(a_1, ..., a_n)$, $P(s)$ es la probabilidad **apriori** de la *skill* s y $P(a_i|s)$ es la probabilidad de tener un atributo a_i dada la *skill* s , esta probabilidad se define como:

$$P(a_i|s) = \frac{n(a_i, s)}{n(s)} \quad (2.8)$$

Finalmente, para calcular la *skill* más probable de un miembro, dado ciertos atributos, se

utiliza:

$$\operatorname{argmin}_{s \in S} P(s) \prod_{i=1}^n P(a_i|s) \quad (2.9)$$

Cabe destacar que para la definición anterior de $P(a_i|s)$ existe la posibilidad que se produzca una probabilidad 0 dado que $n(a_i, s)$ puede ser 0. Por esta razón se procede a realizar un suavizado de *Laplace*, por ende la nueva definición de $P(a_i|s)$ será:

$$P(a_i|s) = \frac{1 + n(a_i, s)}{|W_i| + n(s)} \quad (2.10)$$

Donde $|W_i|$ es la cantidad de palabras que tiene el atributo i .

Para la evaluación del algoritmo recomendador, se predijeron las *skills* que ya están declaradas utilizando un *data set* interno. La clasificación de *skills* resultante para este usuario (que tiene ciertos atributos) es ordenada en una lista de *skills* inferidas y ésta es ordenada para realizar el sistema de recomendador.

Respecto a los resultados obtenidos con datos no etiquetados, se aplicó este algoritmo de recomendación para sugerir *skills* a los perfiles de los usuarios de *LinkedIn*. Para la evaluación del algoritmo con datos no etiquetados se utilizaron 2 métricas distintas: Conversión y Aceptación. La conversión hace referencia a cuántos usuarios agregaron a lo menos una *skill* recomendada por el algoritmo. La Aceptación hace referencia al porcentaje de *skills* agregadas respecto a las *skills* que entrega el algoritmo. Para las pruebas se realizaron 2 configuraciones distintas que varían la cantidad de *skills* que muestra el recomendador, una prueba consiste en mostrar 5 *skills* y la otra 10 *skills*. Se observa que el 57 % de los miembros agregan las 5 *skills* que son mostradas con el primer experimento y el 45 % de los miembros agregan las 10 *skills* que son mostradas por el segundo experimento. Respecto a la cantidad de miembros que agregan todas las *skills* presentadas como sugerencia no se muestra un gran cambio.

2.2.4. *Clustering en perfiles LinkedIn*

La investigación [11] también realizó importantes avances respecto a la minería de datos en la red social *LinkedIn*. Lo que propusieron ellos fue generar un *data set* de perfiles de *LinkedIn* para luego poder aplicar distintas técnicas *cluster* en base a distintas características que consideraron relevantes, por ejemplo, la categoría educacional de los perfiles. A continuación se dará más detalle del cómo fue obtenido el *data set* y cómo fue trabajado para luego aplicar una técnica *cluster*.

Para la obtención de los distintos perfiles, en primera instancia se pensó en una técnica *Crawling*, el cual permitía obtener variada cantidad de información sobre los perfiles como: *skills*, publicaciones, número de conexiones, patentes, descripción, etc. El problema de esta técnica es que la información desplegada que se podía obtener dependía mucho del grado de conexión que se tenía con el perfil al cual se le deseaba obtener la información, en consecuencia se tendrían muchos perfiles incompletos para la investigación.

Como alternativa del problema anterior se decidió realizar una estrategia *Scraping* la cual trabajó en un directorio de perfiles públicos que tiene disponible *LinkedIn*.

Luego para no violar las políticas de privacidad de *LinkedIn* fue necesario anonimizar todos los perfiles del *data set* obtenido.

Por otro lado, los campos extraídos de los perfiles públicos fueron los siguientes:

- Experiencia profesional: Todos los cargos que ha tenido el usuario.
- Descripción: Un campo en texto libre el cual el usuario se describe a si mismo.
- Grado 1: El último título obtenido.
- Grado 2: El penúltimo título obtenido.
- Grado 3: El antepenúltimo título obtenido.

A continuación los investigadores decidieron elegir solo los perfiles que tuvieran al menos una experiencia profesional y algún grado educativo (que en este caso se expresa en tener

grado 1). También se consideraron solo los perfiles que estuvieran escritos en Inglés y que además poseyeran la experiencia profesional descrita en Inglés. La cantidad de perfiles existentes antes de aplicar el filtro era de 5.702.544 perfiles. Luego de aplicar el filtro la cantidad de perfiles existentes en el *data set* disminuyó a 217.930.

Luego de obtener el *data set* con el cual se trabajará, los investigadores descubrieron un problema relacionado con la distinta semántica que se les da a los grados educacionales en distintos países. Para eso se categorizaron las distintos grados educacionales en las siguientes secciones:

- PhD o equivalente: Acá estarán todos los grados que contienen términos como Ph.D, PHD, doctorado, etc.
- Master o equivalente: Esta sección contendrá todos los grados que posean términos como: master, postgraduate, postgrado, etc.
- Bachelor o equivalente: Esta sección contiene términos como bachelor, licenciatura, bachillerato, etc.
- Secondary o equivalente: Contendrá los términos con grado de escuela secundaria.

El siguiente paso que realizan los investigadores trata sobre el manejo del texto libre en los campos que tiene cada perfil en el *data set*. Para esto, los investigadores construyeron un *corpus* en base a los campos Experiencia profesional y Descripción de cada perfil. El primer paso para la construcción del *corpus* fue transformar todo el texto a minúscula, para así facilitar la comparación entre los términos. El segundo paso fue eliminar los términos que carecen de sentido o como se llaman en inglés, eliminar las *stop-words*. El tercer paso fue separar todas las frases por espacios, para así eliminar términos anormales. En esta investigación no se utilizó la técnica *Stemming* ya que no demostró mejor separación de términos con este *data set*.

La cantidad de términos que se obtuvieron utilizando 217.390 documentos (perfiles) fue una cantidad demasiado grande, tanto así que el costo de memoria y de tiempo al procesar la matriz de frecuencias era muy grande. Este problema es más conocido como la maldición

de la dimensionalidad. Debido al problema anterior se decidió trabajar solamente con los perfiles que poseyeran el grado de Phd, utilizar términos de a lo menos largo 4 y eliminar los términos que tuvieran una frecuencia menor a 50. Con esto se logró construir un *corpus* de 1.338 términos considerando 14.650 documentos.

Finalmente se utiliza la técnica *K-Means* aplicada a la matriz de frecuencia de los términos en cada documento. Para encontrar cuál es el *K* indicado para representar el problema, se utilizó el método de *Elbow*, el cual consiste en minimizar la diferencia entre la varianza explicada con la cantidad de *clusters* y la diferencia al cuadrado entre *clusters*. Considerando esta técnica los investigadores consideraron que 9 *clusters* es lo más óptimo. Luego para interpretar los *clusters* se observaron los términos más frecuentes de cada *cluster* para luego aplicar la técnica *Word Cloud* para visualizar de mejor manera la información.

El resultado destacable de esta investigación fue la obtención de distintos grupos de perfiles, desde grupos que representan perfiles asociados al sector público administrativo hasta grupos que describen perfiles más técnicos como desarrollador o ingeniero. Se puede observar que de los 9 grupos generados, solo un grupo pequeño posee perfiles cuyos trabajos no pueden ser categorizados en algo interpretable como los grupos anteriores.

2.3. Distintos algoritmos *Clustering*

Clustering es el proceso de particionar un conjunto de datos dentro de subconjuntos. Cada subconjunto creado se define como un *cluster* y cada dato dentro de un *cluster* es similar de alguna manera a los otros datos que están dentro del mismo. Considerando que existen distintas métricas para definir cómo un dato es similar a otro y que también existen distintas formas de realizar particiones de datos, es que distintos algoritmos de *clustering* pueden resultar con conjuntos totalmente distintos y que en la práctica pueden significar distintas cosas para el problema al que se está aplicando.

Dentro de las aplicaciones en *data mining*, estos métodos pueden ser usados para observar la distribución de los datos, observar las características de cada *cluster* o para concentrar la investigación en un *cluster* en particular. Además se puede utilizar los *cluster* como apoyo a

distintos algoritmos tales como, caracterización, selección de atributos y clasificación.

Por otro lado, los algoritmos *clustering* son conocidos como métodos no supervisados, ya que suponiendo que los datos pertenecen a un grupo, estos datos no están previamente etiquetados con ese grupo. Esta etiqueta se conoce en la literatura como clase. Considerando lo anterior, los algoritmos de *clustering* son una forma de aprendizaje a partir de ejemplos.

Kamber y Pei discuten en [8] distintos requerimientos para que una técnica o algoritmo de *clustering* pueda ser utilizada eficazmente en el ambiente de *data mining*:

- Escalabilidad: Se requiere que los algoritmos de *clustering* trabajen eficientemente tanto con *datasets* de pequeña magnitud hasta *datasets* que contengan millones o billones de datos.
- Manejo de distintos tipos de atributos: Muchos algoritmos son diseñados para agrupar datos con características de tipo numérico. En aplicaciones de la vida real éstos atributos pueden ser del tipo binario, nominales, ordinales o una mezcla de estos tipos de datos.
- Encontrar *clusters* con distintas formas: Muchos algoritmos para encontrar *clusters* se basan en las medidas de distancia *Euclidiana* o de *Manhattan*. Generalmente estos algoritmos encuentran *clusters* con forma esférica con igual densidad entre ellos. Un buen algoritmo *clustering* debería ser capaz de encontrar *clusters* de distintas formas, tamaños y densidades.
- Dependencia del dominio del problema: La mayoría de los algoritmos *clustering* requieren del conocimiento del usuario para fijar los parámetros que determinan completamente el comportamiento del algoritmo. Es requisito tener conocimiento del problema que se trabaja ya que así se le podrá dar parámetros al algoritmo que ayudarán a aumentar la calidad de los resultados.
- Habilidad para manejar el ruido en los datos: La mayoría de los problemas en el mundo real contienen *outliers* (es decir, observaciones atípicas), con atributos nulos, atributos erróneos, etc. Esta situación provoca que los resultados no sean realmente representativos. Por esta razón se requiere que el algoritmo *clustering* ser robusto frente a este

tipo de datos, es decir, detectarlos y manejarlos de manera adecuada.

- Algoritmos de *clustering* incrementales e invariables al orden de ingreso de los datos: En muchas aplicaciones las actualizaciones incrementales (ingreso de nuevos datos) pueden suceder en cualquier momento. Los algoritmos de *clustering* deben ser capaces de incorporar estos datos sin la necesidad de recalculer los *clusters* nuevamente, además el orden de ingreso de los datos no debe afectar en la creación de éstos *clusters*.
- Capacidad de crear *clusters* con una alta dimensión de datos: Los algoritmos de *clustering* deben ser capaces de manejar datos con alta dimensionalidad, por ejemplo, la generación de datos a partir de documentos de texto.
- *Clusters* basados en restricciones: El requerimiento es encontrar buenos *clusters* considerando satisfacer restricciones específicas.
- Interpretabilidad y usabilidad: Los *clusters* generados deben tener relación con el contexto del problema que se está trabajando. Esto es importante ya que personas con un conocimiento no técnico deben ser capaces de interpretar los grupos o *clusters* que se generen.

Por otro lado existen diferentes aspectos que pueden ser utilizados para describir los distintos algoritmos de *clustering*:

- Criterio de partición: Existen algoritmos cuyos *clusters* están en un mismo nivel conceptual, es decir, no existe jerarquía entre los *clusters* generados. En cambio existen algoritmos cuyos *clusters* si poseen una relación jerárquica, es decir, están en un distinto nivel semántico.
- Separación de *clusters* : Algunos algoritmos crean *clusters* donde los datos son incluidos de manera exclusiva, es decir, si pertenece a un *cluster* no puede pertenecer a otro. Por otro lado, existen algoritmos cuyos datos pueden pertenecer a más de un *cluster*.
- Medida de similitud: Existen algoritmos cuya medida de similitud entre los datos está basada en la distancia entre ellos, por ejemplo, la distancia *Euclidiana*. En cambio

existen otros algoritmos que definen su similitud en base a la densidad o qué tan contiguos están los datos. Los algoritmos que basan su similitud en la distancia con los datos ayudan a la optimización del algoritmo. Por otro lado, los algoritmos que basan su similitud en densidad o contigüidad ayudan a encontrar *clusters* con formas arbitrarias.

- Espacio de los *clusters* : Algunos algoritmos de *clustering* generan *clusters* considerando todo el espacio dimensional de los datos. Esto es conveniente solo para datos con baja dimensionalidad. Por lo anterior existen algoritmos que generan los *clusters* considerando solo los atributos más relevantes de los datos.

A continuación se presentarán distintos algoritmos de *clustering* utilizados en la literatura y que servirán de una u otra manera para definir las técnicas que se utilizarán en el transcurso de esta memoria.

2.3.1. Métodos de Partición

Esta es la estrategia más simple para la creación de *clusters* , ya que éste organiza los datos en estudio dentro de grupos o *clusters* exclusivos, es decir, si un dato pertenece a un *cluster*, es imposible que pertenezca a algún otro *cluster*. En esta ocasión la cantidad de *clusters* a crear o descubrir será definida previamente por el usuario.

Para definir formalmente este tipo de *clusters*, considere un *dataset* D que contiene n datos. Además considere la variable k que será la cantidad de *clusters* a formar. Los *clusters* serán formados para optimizar un objetivo dependiente del criterio de partición, en este caso la función de similitud basada en distancia. En general, se busca minimizar la distancia entre los datos dentro de un *cluster* y maximizar la distancia entre los datos de distintos *clusters*.

2.3.1.1. *K-Means*: Basado en el Centroide

Esta técnica utiliza el centroide para representar algún *cluster* C_i donde $1 \leq i \leq k$. Conceptualmente el centroide de C_i se define como el punto central entre todos los datos que

componen el *cluster* y se denominará con c_i . Luego se define la diferencia entre un dato p perteneciente a C_i con el centroide c_i como $dist(p, c_i)$, donde $dist(a, b)$ es la distancia *Euclidiana* entre a y b .

A continuación se presenta el algoritmo de *K-Means* como lo presenta [8]:

Data:

k : Número de *clusters*.

D : *dataset* que contiene n datos.

Result: Un conjunto de k *clusters*.

- 1 Elegir aleatoriamente k datos de D para ser elegidos como centroides;
- 2 **while** *Existan cambios* **do**
- 3 Reasignar cada objeto al *cluster* que tenga el centroide más cerca;
- 4 Actualizar el centroide considerando los datos actualizados;
- 5 **end**

Algorithm 1: Algoritmo *K-Means*.

Cabe destacar que este método no asegura la convergencia al óptimo global al tratar de encontrar los *clusters* más compactos y separados. Esto se debe a que los puntos iniciales se inicializan de manera aleatoria.

Otro punto importante es la complejidad de tiempo que tiene este algoritmo, el cual es $O(nkt)$ donde n es la cantidad de datos, k es la cantidad de *clusters* y t es el número de iteraciones. Hay que considerar que generalmente k es mucho menor que n y éste a su vez es mucho menor que t .

Una de las variantes más importantes de este algoritmo es *K-Modes* el cual extiende *K-Means* para ser utilizado considerando datos nominales. Para esto se toma en cuenta la moda de los datos a utilizar como el centroide de los *clusters*.

2.3.1.2. *K-Medoids*

Este algoritmo fue creado para evitar la sensibilidad de *outliers* que posee el algoritmo *K-Means*. Para esto, en vez de elegir como representación de un *cluster* la media de los elementos del mismo *cluster*, se elige un individuo de ese *cluster* como el elemento representativo de éste. La elección de los *clusters* se genera en base a la premisa de que se debe minimizar la suma de la disimilitud entre cada objeto p y su correspondiente objeto representativo. Esto es el criterio del error absoluto y se define como:

$$E = \sum_{i=1}^k \sum_{p \in C_i} \text{dist}(p, o_i) \quad (2.11)$$

Donde E es la suma absoluta del error de todos los datos p en el *dataset* y o_i que es el objeto representativo de C_i . El problema de encontrar la asignación óptima de objetos dentro de K *clusters* es un problema *NP-Hard*.

Este algoritmo aborda el problema de asignación de *clusters* aplicando un algoritmo *Greedy*, es decir, iterativamente se trata de mejorar la función objetivo. Para comenzar el algoritmo, al igual que *K-Means*, los objetos representativos se eligen aleatoriamente. Para esta ocasión se considera cualquier reemplazo de un objeto representativo a uno no representativo que mejore la calidad de los *clusters*. Los objetos a probar son los que estén más cercanos al objeto representativo a cambiar. Este proceso iterativo continúa hasta que no exista ningún reemplazo que mejore la calidad de los *clusters*. La medida de calidad de los *clusters* será el promedio del error absoluto de los objetos con el elemento representativo de su *cluster*. Para la reasignación de los puntos a los nuevos *clusters*, para cada punto se asigna el *cluster* cuyo objeto representativo esté más cercano a dicho punto.

A continuación se presenta el algoritmo *K-Medoid*:

Data:

k : Número de *clusters*

D : *dataset* que contiene n datos.

Result: Un conjunto de k *clusters*.

```
1 Elegir aleatoriamente  $k$  datos de  $D$  para ser elegidos como objetos representativos;
2 while Existan cambios en la calidad de los clusters do
3   Reasignar cada objeto al cluster que tenga el objeto representativo más cerca;
4   for cada cluster  $C_i$  do
5     Elegir de manera aleatoria otro objeto representativo  $o_{rand}$  que esté dentro de
         $C_i$ ;
6     if La calidad de los clusters aumenta then
7       Elegir como nuevo objeto representativo del cluster  $C_i$  el objeto  $o_{rand}$ ;
8     end
9   end
10  Actualizar los puntos considerando los objetos representativos actualizados;
11 end
```

Algorithm 2: Algoritmo *K-Medoids*.

2.3.2. Métodos Jerárquicos

Estos métodos son utilizados cuando se desean encontrar *clusters* exclusivos y con distintos niveles de significancia, es decir, es posible encontrar grupos de *clusters* organizados como una jerarquía o un “árbol” de *clusters*.

La dificultad de utilizar este método radica en la selección del método con el cual se realizará la separación o unión de los distintos *clusters*. Esta decisión es crítica ya que una vez que los *clusters* se unan o se separen, la siguiente iteración considerará el nuevo *cluster* resultante. También cabe destacar que deshacer el proceso de juntar/separar algún *cluster* no permite que los datos se puedan cambiar de *clusters*. Esto implica que la decisión de qué se juntará o separará tenga que examinar muchos datos o *clusters*, haciendo de este proceso algo muy costoso.

Para solucionar este problema y aumentar la calidad de los *clusters* generados, se deben integrar los métodos jerárquicos con otras técnicas *clustering*, resultando así otro método llamado *Multiple-phase clustering*.

Los métodos jerárquicos a su vez se pueden dividir en 2 tipos: Algorítmicos (Aglomerativos y divisivos) y Probabilistas. A continuación se dará detalles de cada uno:

2.3.2.1. Métodos Jerárquicos Aglomerativos y Divisivos

Los métodos Algorítmicos pueden ser separados en aglomerativos y divisivos dependiendo del cómo se realicen los *clusters*.

Los métodos aglomerativos utiliza una estrategia *bottom-up*, es decir, comienzan de las hojas del árbol a construir hasta llegar a la raíz. Al comienzo de esta estrategia cada dato es considerado como un *cluster* que luego se van uniendo a medida que proceden las iteraciones. El algoritmo se detiene cuando todos los datos están en un mismo *cluster* o cuando se llegue a una cierta cantidad de *clusters*. En cada iteración los 2 *clusters* más cercanos (de acuerdo a la medida de similitud elegida) se unen para formar uno nuevo que luego se utilizará como un único *cluster* en la siguiente iteración.

Los métodos divisivos utilizan una estrategia *top-down*, es decir, comienzan desde la raíz del árbol a construir hasta llegar a las hojas. Al comienzo de la estrategia todos los datos están en un mismo *cluster* y a medida que avanzan las iteraciones el *cluster* se separa en unos más pequeños que a su vez en la siguiente iteración también se dividirán. El algoritmo se detiene cuando se llega a la unidad más pequeña de *cluster* (cada dato es un *cluster*) o hasta que se llegue a una cierta cantidad de *clusters* definida por el usuario anteriormente.

2.3.2.2. Medidas de similitud entre Clusters

Cuando se utiliza un método Jerárquico del tipo Algorítmico es necesario definir una medida de similitud que se utilizará para realizar las comparaciones de distancia entre 2 *clusters*. Para comprender de manera más simple las distintas métricas, considere que $|p - p'|$ es la distancia

entre los 2 datos o puntos p y p' , m_i será el centro de cada *cluster* C_i y n_i es el número de datos del *cluster* C_i . A continuación se presentan las distintas medida de similitud:

- **Mínima Distancia:** La distancia entre 2 *clusters* es considerada como la distancia entre los puntos más cercanos de cada *cluster*.

$$dist_{min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} |p - p'| \quad (2.12)$$

- **Máxima Distancia:** La distancia entre 2 *clusters* es considerada como la distancia entre los puntos más lejanos de cada *cluster*.

$$dist_{max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} |p - p'| \quad (2.13)$$

- **Distancia Central:** La distancia entre 2 *clusters* e considerada como la distancia entre los centros de ambos *clusters*.

$$dist_{mean}(C_i, C_j) = |m_i - m_j| \quad (2.14)$$

- **Distancia Promedio:** Es la distancia promedio entre todos los puntos de ambos *clusters*.

$$dist_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i, p' \in C_j} |p - p'| \quad (2.15)$$

Respecto a las distintas medidas de similitud es importante nombrar que el método de Mínima y Máxima distancia son muy sensibles a *outliers* que pueden presentarse en los distintos *datasets*, por lo que el algoritmo se torna inestable si se utilizan estas medidas. En cambio las medidas de Distancia Central y Promedio manejan en cierto modo los distintos *outliers* que puedan aparece, sin embargo, existe un costo de calcular tanto el centro del *cluster* como el promedio de los datos. En esta ocasión el cálculo más costoso pertenece a la medida del Promedio ya que se calcula la distancia entre todos los datos o puntos del nuevo *cluster*. En definitiva la medida de similitud que tiene control de *outliers* y que es relativamente más fácil de computar es la medida Central.

2.3.2.3. *Chameleon: Multiphase Hierarchical Clustering Using Dynamic Modeling*

Este algoritmo fue propuesto por [12] y trata sobre un algoritmo *clustering* aglomerativo que evita los inconvenientes de los algoritmos existentes hasta ese momento. *Chameleon* para crear los *clusters* se basa en qué tan conectados están los objetos dentro del *cluster* y qué tan cercanos están los *clusters*. Por lo tanto, 2 *clusters* son unidos si su inter-conectividad es alta y si ellos están cerca el uno con el otro. Este proceso de unión permite encontrar *clusters* homogéneos durante todo el proceso.

El primer paso de este algoritmo es crear un grafo de conexiones de k vecinos más cercanos, donde cada vértice representa un dato. La conexión entre vértices existirá si y solo si un vértice está dentro de los k vecinos más cercanos de otro. La etiqueta de cada arco será qué tan cercano está un vértice de otro con respecto a alguna métrica de distancia.

A continuación se procede a particionar este grafo utilizando una constante que en este caso se llamará *MinSize*, y que definirá cuándo el algoritmo de partición debe parar. El algoritmo de partición parará cuando el *cluster* con mayor número de datos tenga menos datos que *MinSize*. Respecto qué *clusters* surgirán debido a la partición de arcos, se considerará que cada vez que ocurra una partición, se debe minimizar una función objetivo denominada *Edge Cut*, y definida como la suma de todos los arcos que unen distintos *clusters*.

Luego se procede a la unión de los distintos *clusters* generados. Para esto se utilizan 2 medidas de similitud: *Relative interconnectivity* (RI) y *Relative closeness* (RC).

RI se define como la conectividad absoluta entre un *cluster* C_i y otro C_j , normalizado en términos de la conectividad interna de C_i y C_j . RI se calcula de la siguiente manera:

$$RI(C_i, C_j) = \frac{|EC_{(C_i, C_j)}|}{\frac{|EC_{(C_i)}| + |EC_{(C_j)}|}{2}} \quad (2.16)$$

Donde $EC_{(C_i, C_j)}$ corresponde al *Edge Cut* entre C_i y C_j . $EC_{(C_i)}$ (o $EC_{(C_j)}$) será la mínima suma de los *Edge Cuts* que separé C_i (o C_j) en 2 partes iguales (misma cantidad de datos).

RC se define como la cercanía relativa entre un *cluster* C_i y otro C_j , normalizado con respecto

a la cercanía interna de 2 *clusters* C_i y C_j . RC se calcula de la siguiente manera:

$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i, C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}} \quad (2.17)$$

Donde $\overline{S}_{EC_{\{C_i, C_j\}}}$ es el promedio de los arcos que conectan los *clusters* C_i y C_j y $\overline{S}_{EC_{C_i}}$ (o $\overline{S}_{EC_{C_j}}$) es el promedio de los pesos de los arcos que cortan a la mitad el *cluster* C_i (o C_j) considerando aquel que minimiza la suma de los arcos eliminados para separar el grafo.

Finalmente 2 *clusters* son unidos si entre ellos existe una gran cercanía relativa y una gran conectividad absoluta. En otras palabras se debe encontrar un par de *clusters* C_i, C_j tal que el siguiente producto sea mayor a un *threshold* definido como parámetro del algoritmo:

$$RI(C_i, C_j)^\alpha \times RC(C_i, C_j) \quad (2.18)$$

Donde α es un parámetro que define la importancia de la conectividad absoluta. Si α es mayor que 1 entonces se dará más importancia a la conectividad absoluta, si es menor a 1 entonces se le dará más importancia a la cercanía relativa. Si α es igual a 1 entonces ambos tendrán la misma importancia.

A continuación se presenta un algoritmo *Chameleon* que se basó en [4]:

Data: $D = o_1, \dots, o_n$: *dataset* que contiene n datos.

k : Cantidad de vecinos cercanos.

MinSize: Tamaño mínimo del *cluster* más grande.

α : Importancia de conectividad absoluta.

th: *Threshold* definido para la unión de *clusters*.

Result: Jerarquía de *clusters*.

```

1  Generar grafo  $G$  utilizando  $k - \text{Nearest} - \text{Neighbors}$  al dataset  $D$ ;
2  Generar  $m$  clusters particionando  $G$  considerando MinSize;
3  for  $i = 1$  hasta  $m$  do
4      for  $j = i + 1$  hasta  $m$  do
5          Calcular  $RI$  y  $RC$  para  $C_i$  y  $C_j$ ;
6          Calcular  $V = RI^\alpha \times RC$ ;
7          if  $V$  es mayor o igual que  $th$  then
8              Unir  $C_i$  y  $C_j$ ;
9          else
10             No unir  $C_i$  y  $C_j$ ;
11         end
12     end
13 end

```

Algorithm 3: Algoritmo Jerárquico *Chameleon*.

Se puede observar que este algoritmo posee 4 parámetros a definir por el usuario, siendo un gran problema de sintonización de parámetros, ya que se debe encontrar la combinación óptima de estos parámetros para obtener un rendimiento óptimo de este algoritmo.

2.3.2.4. *Ward*

El método de *clustering* jerárquico llamado *Ward* [15] [14] es un método del tipo Aglomerativo, ya que en cada iteración, éste va agrupando distintos *clusters* que existen en el espacio. Como muchos algoritmos de *clustering*, en esta ocasión para lograr generar los *clusters* de manera adecuada se utiliza un criterio de optimización. La función que se desea minimizar

en este caso se basa en la distancia entre *clusters*. Considerar el cluster A y B , el término que se desea minimizar en cada unión es el siguiente:

$$\Delta(A, B) = \sum_{i \in A \cup B} \|x_i - m_{A \cup B}\|^2 - \sum_{i \in A} \|x_i - m_A\|^2 - \sum_{i \in B} \|x_i - m_B\|^2 \quad (2.19)$$

$$\Delta(A, B) = \frac{n_A n_B}{n_A + n_B} \|m_A - m_B\|^2 \quad (2.20)$$

Donde m_j es el centro del *cluster* j y n_j es el número de puntos dentro de éste. Esta terminología también es utilizada en la investigación [15].

Lo primero que se puede evidenciar es que tratar de llegar al óptimo global minimizando Δ para todos los *clusters*, es un problema *NP*-Completo debido a la cantidad de combinaciones posibles, por ende, el algoritmo *Ward*, aplica un algoritmo *Greedy* el cual tratará siempre de juntar los *clusters* que se encuentren más cercanos el uno del otro.

Data: D : Matriz de distancias.

p : Nivel de jerarquía máxima.

Result: Jerarquía de *clusters*.

```

1 Crear un cluster  $C$  por cada fila (o columna) de la matriz  $D$ ;
2 for cada cluster existente do
3   | Encontrar un par de clusters  $C_i$  y  $C_j$  tal que  $\Delta(C_i, C_j)$  tenga el mínimo valor;
4   | if Nivel de la Jerarquía actual  $> p$  then
5   |   | Retornar Jerarquía.;
6   | end
7 end
```

Algorithm 4: Algoritmo Jerárquico *Ward*.

2.3.2.5. Métricas de Evaluación para Métodos Jerárquicos

Luego de observar las distintas técnicas para construir una partición jerárquica, se procederá a ver una métrica que evalúa el conjunto de *clusters* obtenidos. Esta métrica es conocida como *Cophenetic Correlation Coefficient* (CPCC) y ha sido utilizada en distintas investigaciones [20][19]. Este coeficiente indica cuán fielmente un dendrograma presenta la distancia

entre pares, comparado con las distancias de los puntos originales. Este coeficiente se define de la siguiente manera:

$$c = \frac{\sum_{i < j} (D(i, j) - \bar{D})(DT(i, j) - \bar{DT})}{\sqrt{\sum_{i < j} (D(i, j) - \bar{D})^2 \sum_{i < j} (DT(i, j) - \bar{DT})^2}} \quad (2.21)$$

Donde X_i es el punto i de los datos en estudio, T_i corresponde a la ubicación del punto X_i dentro del dendrograma generado, $D(i, j)$ es la distancia *Euclidiana* entre los puntos X_i y X_j , $\bar{D}$ corresponde al promedio de las distancias de los puntos, $DT(i, j)$ corresponde a la altura del nodo donde T_i y T_j se unen por primera vez y $\bar{DT}$ corresponde al promedio de las distancias de los nodos.

Por último, no está demás mencionar que entre más cercano a 1 es el el valor de CPCC, mejor es el dendrograma generado.

2.3.2.6. Métodos Jerárquicos Probabilistas

Se puede observar que los métodos Algorítmicos poseen grandes desventajas a la hora de ser implementados y que afectarán directamente a la calidad de la solución entregada. En este tipo de algoritmos se debe escoger una medida de similitud adecuada al contexto del problema, ya que en cada iteración se asignan los *clusters* tratando siempre de encontrar el mejor *cluster* hasta ese momento, por lo tanto, se podría considerar como un algoritmo *Greedy* que no asegura llegar a la solución óptima global.

En esmero de evitar las desventajas de la creación de *clusters* Jerárquicos utilizando algoritmos tipo Algorítmicos, se propone la creación de *clusters* Jerárquicos tipo Probabilistas. Estos tipos de *clusters* Jerárquicos utilizan modelos probabilistas como medida de distancia entre *clusters*. Una forma de ver el problema de creación de *clusters* es considerar el *dataset* que se desea encontrar *clusters* como la idea del mecanismo de generación de datos, es decir, un modelo generativo.

Para trabajar con el modelo generativo se debe realizar una suposición muy fuerte y es que los datos generados poseen una distribución *Gaussiana* o *Bernoulli*. La finalidad del modelo

generativo es encontrar los parámetros libres de la distribución utilizada a modo de crear un modelo que represente más fielmente el *dataset*.

A continuación se presenta el algoritmo jerárquico probabilista presentado por [8]:

Data: $D = o_1, \dots, o_n$; *dataset* que contiene n datos.

Result: Jerarquía de *clusters*.

```

1 Crear un cluster por cada dato  $C_i = o_i, 1 \leq i \leq n$  ;
2 for  $i = 1$  hasta  $n$  do
3   Encontrar un par de clusters  $C_i$  y  $C_j$  tal que  $C_i, C_j = \text{argmax}_{i \neq j} \log \frac{P(C_i \cup C_j)}{P(C_i)P(C_j)}$ ;
4   if  $\log \frac{P(C_i \cup C_j)}{P(C_i)P(C_j)} > 0$  then
5     Unir  $C_i$  y  $C_j$ ;
6   else
7     Parar;
8   end
9 end

```

Algorithm 5: Algoritmo Jerárquico Probabilista.

2.3.3. Métodos Basados en Densidad

Los métodos de Partición y Jerárquicos están diseñados para encontrar *clusters* con formas esféricas, limitando así el encontrar *clusters* con formas arbitrarias. Para encontrar este tipo de *cluster*, se pueden modelar los *clusters* como regiones densas en el espacio, separadas por una región sin densidad. Esta es la estrategia tras los métodos basados en densidad, que pueden descubrir *clusters* que no necesariamente tengan formas esféricas. A continuación se presenta *DBSCAN* un algoritmo que está basado en la creación de *clusters* en regiones que tienen una alta densidad de puntos.

2.3.3.1. *DBSCAN: Density-Based Clustering Based on Connected Regions with High Density*

El algoritmo *DBSCAN* realiza *clustering* analizando la densidad de datos en torno a los puntos que se desean agrupar. La densidad de un punto se calcula en base a la cantidad de datos que estén a su alrededor, es decir, la cantidad de vecinos que tenga. Este algoritmo procura buscar puntos con densidad alta, para luego conectarlo con sus vecinos y así generar zonas con alta densidad de puntos.

El parámetro utilizado para saber si un punto es cercano a otro punto se define como ϵ y se puede interpretar como el radio que tendrán los puntos para considerar si existe vecindad entre otros puntos. Ahora, para saber si un punto es denso o no, se utiliza un parámetro denominado *MinPts*, el cual se define como el límite mínimo de vecinos que debe tener un punto para considerarse denso. Finalmente, se define más formalmente un punto denso como un punto que tiene al menos *MinPts* vecinos dentro del radio ϵ .

A continuación se definen 2 conceptos importantes y que servirán para definir de mejor manera el funcionamiento del algoritmo:

- **Densamente-Aceptable:** Considerar 2 puntos denominados q y p , tal que q es un punto denso considerando un cierto ϵ y *MinPts*. Se dice que p tiene una relación Densamente-Aceptable con q si p está dentro del radio de vecindad (definido por ϵ) de q . No está demás mencionar que la relación establecida no es simétrica ya que si se considera el caso contrario, que q tenga una relación Densamente-Aceptable con p , p tendría que ser obligatoriamente un punto denso.
- **Densamente-Conectado:** Considerar 3 puntos denominados q , p y p' , tal que q es un punto denso considerando un cierto ϵ y *MinPts*. Se dice que p está Densamente-Conectado con p' si y solo si p y p' tienen una relación Densamente-Aceptable con q . Esta relación se establece si y solo si q es un punto denso.

A continuación se presenta el algoritmo de *DBSCAN* presentado por [8].

Data: *D*: *dataset* que contiene *n* datos.

ϵ : El radio de los puntos.

MinPts: Cantidad mínima de puntos.

Result: Un conjunto de *clusters* basados en densidad.

```
1  Marcar todos los datos como no visitados;
2  while existan datos no visitados do
3      | Selecccionar de manera aleatoria un punto no visitado p;
4      | Marcar p como visitado;
5      | if Si dentro del radio de p existen al menos MinPts puntos then
6      |     | Crear un nuevo cluster C y agregar p a C;
7      |     | Considerar N como la cantidad de puntos dentro del radio de P;
8      |     | for Por cada punto p' en N do
9      |     |     | if p' no está visitado then
10     |     |     |     | Marcar p' como visitado;
11     |     |     |     | if Si dentro del radio de p' existen al menos MinPts puntos then
12     |     |     |     |     | Agregar estos puntos a N;
13     |     |     |     | else
14     |     |     |     | end
15     |     |     | else
16     |     |     | end
17     |     |     | if p' aún no es miembro de algún cluster then
18     |     |     |     | Agregar p' a C;
19     |     |     | else
20     |     |     | end
21     |     | end
22     | else
23     |     | Marcar p como ruido
24     | end
25 end
```

Algorithm 6: Algoritmo DBSCAN.

Respecto al desempeño del algoritmo, usando un índice espacial de los datos, la complejidad de *DBSCAN* es $O(n \log n)$, donde n es el número de datos en el *dataset*. En el caso que no se utilice un índice la complejidad del algoritmo es $O(n^2)$. El problema de este algoritmo es definir los parámetros ϵ y *MinPts* para que se encuentren soluciones óptimas al problema.

2.3.4. Criterios de evaluación de un *cluster*

A continuación se verán distintas métricas para observar la calidad de los *clusters* y para observar si es posible aplicar algoritmos *clustering* a algún conjunto de datos en estudio.

2.3.4.1. Evaluación de la tendencia de *clustering*

La tendencia de *clustering* hace referencia a qué tan bueno son los datos para poder aplicar algún algoritmo de generación de *clusters*. Para evaluar la tendencia de *clustering* de algún *dataset* es necesario recurrir a métodos estadísticos para analizar si los datos pueden ser generados por alguna distribución de datos. Para esto se utilizará el estadístico de *Hopkins* [10] que analiza la aleatoriedad de los datos y cómo están distribuidos en el espacio. El estadístico se calcula de la siguiente manera:

1. Considere n puntos, $p_1, \dots, p_n$ de algún *dataset* D . Por cada punto se debe encontrar el vecino p_j más cercano y calcular su distancia que se denominará x_i

$$x_i = \min_{v \in D, v \neq p_i} \text{dist}(p_i, v) \quad (2.22)$$

2. Generar n cantidad de puntos a través de una distribución uniforme aleatoria que posea la misma varianza que el *dataset* D . Finalmente se tiene un nuevo *dataset* $D' : (q_1, \dots, q_n)$ con los nuevos puntos generados.
3. Por cada punto generado q_i se debe encontrar el vecino más cercano q_j y calcular su distancia que se denominará y_i .

$$y_i = \min_{v \in D', v \neq q_i} \text{dist}(q_i, v) \quad (2.23)$$

4. Luego se calcula el estadístico de *Hopkins*:

$$H = \frac{\sum_{i=1}^n y_i}{\sum_{i=1}^n x_i + \sum_{i=1}^n y_i} \quad (2.24)$$

La hipótesis nula es que los datos están distribuidos de forma homogénea, por lo tanto, no se tienen *clusters* significativos. La hipótesis alternativa dice que los datos no están distribuidos de forma homogénea, por lo tanto, existen *clusters* significativos.

Considerando el cálculo del estadístico 2.24 se observa que si existiera una distribución uniforme, entonces $\sum_{i=1}^n x_i$ y $\sum_{i=1}^n y_i$ tendrían un valor muy similar, por lo tanto el estadístico tendría un valor sobre 0,5. En cambio si existe una distribución tal que permita aplicar algún algoritmo de *clustering*, entonces $\sum_{i=1}^n x_i$ sería mucho más pequeño que $\sum_{i=1}^n y_i$ por lo que H tendría un valor cercano a 1. Considerando esto se tiene que la hipótesis alternativa se rechaza si H tiene un valor inferior 0,5.

2.3.4.2. Determinar el número de *clusters*

Saber cuántos *clusters* se crearán a la hora de analizar datos, es fundamental para generar un equilibrio entre la precisión de los *clusters* y la facilidad de análisis de los datos que se estudiarán. Existen diversos métodos para calcular una “aproximación” de la cantidad óptima de *clusters* a considerar. Un método simple es considerar una cantidad de *clusters* de $\sqrt{\frac{n}{2}}$ donde n es la cantidad de datos que se tienen. Cada *cluster* podría contener $\sqrt{2n}$ datos.

Otro método interesante es el método visual de *Elbow*, el cual se utiliza en [11] para definir un número óptimo de *clusters* a utilizar. Este método se basa en el cálculo de la suma del error cuadrático de los puntos de un *cluster* con respecto al centroide o el punto representativo del éste. El cálculo mencionado anteriormente se realiza para K número de *clusters* para luego graficar el error cuadrático en función de la cantidad de *clusters*. Una vez se posee el gráfico que presenta cómo se comporta el error cuadrático en función del número de *clusters*, se procede a observar la curva resultante. En el caso que exista un cambio brusco en la tendencia de la línea presentada, ese será el punto en el cuál habrá que elegir la cantidad de *clusters*. Este método es bastante subjetivo y no siempre se encuentra un punto de inflexión marcado como se presenta en la figura 2.7

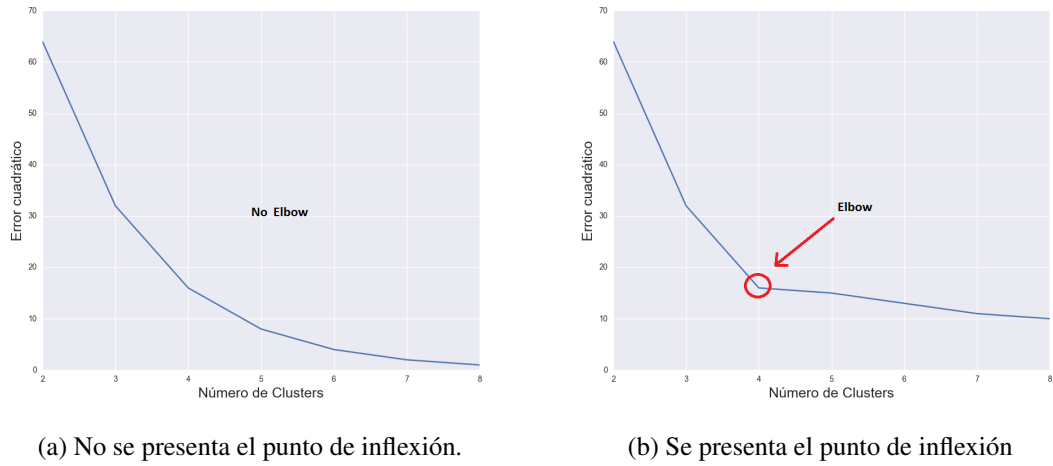


Figura 2.7: Ejemplo de método de *Elbow*.

Este método es bastante flexible con respecto a la métrica utilizada para observar la evolución de los *clusters*, es decir, en vez de utilizar la suma del error cuadrático, se puede utilizar el Coeficiente de *Silhouette* o alguna otra métrica que se estime conveniente.

2.3.4.3. Determinando la calidad de los *clusters*

A continuación se presentarán algunos métodos para observar qué tan buenos son los *clusters* que se han generado. Para esto existen 2 métodos para medir la calidad, un método extrínseco y un método intrínseco.

Los métodos extrínsecos son utilizados cuando se conoce la agrupación de los datos que se tienen disponible, es decir, sirven para probar algoritmos de *clustering* sobre datos para los que ya se conoce la agrupación óptima. En cambio en los métodos intrínsecos se evalúa la calidad considerando solo qué tan bien están separados los *clusters*. Para esta memoria solo se hará uso de métodos intrínsecos, ya que no se conoce una agrupación explícita de los datos que se analizarán en este estudio.

El método intrínseco más conocido es el llamado Coeficiente de *Silhouette*[21]. Considerar un *dataset* D de n datos. Suponer que se generan K *clusters*, $C_1, \dots, C_k$. Por cada dato $o \in D$

se calcula $a(o)$ como qué tan compacto está el *cluster*, C_i , que contiene a o , y se define de la siguiente manera:

$$a(o) = \frac{\sum_{o' \in C_i, o \neq o'} \text{dist}(o, o')}{|C_i| - 1} \quad (2.25)$$

Luego se define $b(o)$ como el grado de separación del dato o respecto a los demás *clusters*:

$$b(o) = \min_{C_j: 1 \leq j \leq k, j \neq i} \frac{\sum_{o' \in C_j} \text{dist}(o, o')}{|C_j|} \quad (2.26)$$

Finalmente el coeficiente de *Silhouette* de o está definido como:

$$s(o) = \frac{b(o) - a(o)}{\max\{a(o), b(o)\}} \quad (2.27)$$

Cuando el valor de *Silhouette* de un dato o es muy cercano a 1, quiere decir que $b(o)$ posee un valor mucho más grande que $a(o)$, es decir, el dato o está alejado de otros *clusters* y a su vez está muy cercano a los datos que están dentro de su mismo *cluster*, por lo tanto esta sería la situación ideal para todos los datos. En el caso contrario, cuando el coeficiente de *Silhouette* es cercano a -1, quiere decir que $b(o)$ posee un valor muy pequeño a comparación de $a(o)$, esto significa que el dato está muy cerca de otros *clusters* y a su vez alejado de los datos que están dentro de su mismo *cluster*, por ende, esto sería el peor caso en la evaluación de *clusters*.

Finalmente se puede apreciar que el Coeficiente de *Silhouette* también puede ser utilizado para obtener la cantidad óptima de *clusters* a generar. Esto se puede hacer considerando el método de *Elbow* pero cambiando la suma de la distancia al cuadrado entre *clusters* por el promedio del Coeficiente de *Silhouette* de todos los puntos dentro del *dataset*.

2.4. *Topic Models*

Finalmente, se presentarán algunas técnicas que son muy conocidas en el ámbito de minería de datos cuando se trabajan con documentos de texto. Estas técnicas también son conocidas como modelos estadísticos que se encargan de descubrir distintos tópicos en una ocurrencia de documentos. En esta ocasión, los tópicos pueden ser vistos como grupos implícitos que se encuentran en la colección de documentos y que sirven para clasificarlos. A diferencia de la agrupación generada por los algoritmos *clustering* de partición y basados en densidad, estos grupos no consideran los datos de forma excluyente, es decir, un dato puede pertenecer, por ejemplo, un 80 % a un tópico y un 20 % a otro tópico. Los algoritmos que se describirán en esta sección son *Latent Semantic Analysis* y *Latent Dirichlet Allocation*. Las demás técnicas como *Non-Negative Matrix Factorization* y *Singular Value Decomposition* se analizarán con más detalle en la sección [2.5](#)

2.4.1. *Latent Semantic Analysis*

Latent Semantic Analysis (LSA) [24] es una técnica utilizada para comparar distintas palabras usando una representación basada en vectores que es obtenida a través de distintos documentos que contienen las palabras que se analizarán.

El cómo trabaja LSA se basa en una matriz de ocurrencia de palabras dentro de distintos documentos, a esta matriz generalmente se aplica un proceso de *Term Frequency Inverse Document Frequency* [18], el cual consiste en transformar la información de la matriz de ocurrencia, considerando qué tan relevante es la palabra para la colección de documentos. Luego con esta estructura en mente, se procede a reducir el espacio dimensional con el que se trabajará, para esto se utiliza la técnica de descomposición llamada SVD. Considerando la matriz de dimensiones cantidad de documentos por cantidad de tópicos requeridos, se tendrá una matriz que representará qué tan relacionado está el documento i al tópico j .

De manera matemática considere una matriz X donde el elemento (i, j) describe la ocurrencia del término i en el documento j . Ahora la fila de una matriz será un vector que corresponde

al término i :

$$t_i^T = [X_{i,1} \dots X_{i,n}]$$

Asimismo, una columna de una matriz será un vector correspondiente a un documento j :

$$d_j^T = [X_{1,j} \dots X_{m,j}]$$

Luego considerando el producto punto entre $t_i^T t_p$ se obtiene la correlación entre 2 términos distintos ($t_p^T t_i$ entrega el mismo resultado de correlación para los términos i, j). La matriz resultante del producto punto entre XX^T contiene todas las correlaciones entre todos los términos. De la misma manera, la matriz $X^T X$ contendrá todas las correlaciones existentes entre todos los documentos disponibles (al igual que en los términos, $d_j^T d_q = d_q^T d_j$ corresponden a la correlación entre los documentos j y q).

Ahora, de la teoría del álgebra lineal, existe una descomposición de X tal que, U y V son matrices ortogonales y Σ es una matriz diagonal.

$$X = U\Sigma V^T \quad (2.28)$$

Esta descomposición es llamada *Singular Value Decomposition*, de esta descomposición se dará más detalle en la sección 2.5.2.

Considerando la descomposición 2.28 se tiene que los productos de las matrices de correlación de términos y documentos es la siguiente:

$$XX^T = U\Sigma^2 U^T \quad (2.29)$$

$$X^T X = V\Sigma^2 V^T \quad (2.30)$$

Teniendo en cuenta que Σ^2 son matrices diagonales, se observa que U y V debe contener los vectores propios de XX^T y $X^T X$ respectivamente.

Ahora a la factorización realizada por SVD se le puede dar el siguiente significado:

$$\begin{array}{c} X \\ \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix} \end{array} = \begin{array}{c} U \\ \begin{bmatrix} \begin{bmatrix} \vdots \\ \mathbf{u}_1 \end{bmatrix} \dots \begin{bmatrix} \vdots \\ \mathbf{u}_l \end{bmatrix} \end{bmatrix} \end{array} \cdot \begin{array}{c} \Sigma \\ \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix} \end{array} \cdot \begin{array}{c} V^T \\ \begin{bmatrix} \vdots & \mathbf{v}_1 \\ \vdots & \vdots \\ \vdots & \mathbf{v}_l \end{bmatrix} \end{array}$$

Figura 2.8: Factorización realizada en SVD.

De la figura 2.8 se observa que los valores σ pertenecen a los valores singulares de la matriz X , los vectores de u y v son llamados vectores singulares izquierdos y derechos. Ahora, la única parte de U que contribuye al término t_i es la i -ésima fila. De otro modo, la parte de V^T que contribuye al documento d_j es la j -ésima columna.

Por otro lado, si se eligen los k valores singulares con sus correspondientes vectores singulares de U y V , se tiene una aproximación de X de *rank* k con el mínimo error considerando la norma de *Frobenius*. Lo más importante de esta aproximación es que ahora cada término y documento se pueden ver como “espacios semánticos”. De la matriz aproximada, la fila $\hat{t}_i^T$ representará al término i . A si mismo, para los documentos, la columna $\hat{d}_j$ representará el documento j .

La aproximación de la matriz se escribirá de la siguiente manera:

$$X_k = U_k \Sigma_k V_k^T \quad (2.31)$$

Finalmente, con la reducción de espacio dimensional se puede realizar lo siguiente:

- Ver qué tan relacionados están los documentos j y q en el espacio reducido, comparando los vectores aproximados $\Sigma_{k_j} \hat{d}_j$ y $\Sigma_{k_q} \hat{d}_q$.
- Comparar los términos i y p comparando los vectores aproximados $\Sigma_{k_i} \hat{t}_i$ y $\Sigma_{k_p} \hat{t}_p$.
- Los términos y documentos con la representación vectorial pueden ser agrupados utilizando algoritmos de *clustering* y usando medidas de similitud como la de coseno.

- Por último, es posible asignar una agrupación o *cluster* a cada término utilizando el espacio reducido, para esto hay que considerar la matriz U_k resultante de la descomposición 2.31. Esta matriz es de dimensiones *cantidad de términos* $\times k$. Ahora, un término será asignado al grupo que posea mayor peso en la matriz U_k , esto quiere decir, una palabra tendrá k dimensiones, por lo que dentro de esos k valores, se elegirá como grupo el que tenga mayor valor dentro de esa fila. Análogamente, para los documentos se puede realizar el mismo proceso pero utilizando la matriz V_k .

2.4.2. *Latent Dirichlet Allocation*

Latent Dirichlet Allocation (LDA) [5] al igual que LSA, también es un método para relacionar palabras o documentos a través de grupos implícitos presentados en el conjunto de datos. A diferencia de LSA, LDA es un modelo probabilístico generativo que ayuda a encontrar estos diferentes grupos.

La idea básica que está detrás de LDA, es que un documento está representado por un conjunto de variados tópicos, donde cada tópico está caracterizado por una distribución de palabras.

Al igual que LSA, las soluciones entregadas por este modelo se pueden interpretar como qué tan relacionado está la palabra o documento respecto a un tópico.

A continuación se explicará la interpretación matemática y estadística del modelo.

Considerar el siguiente proceso generativo para cada documento w en el corpus D :

1. Elegir $N \sim \text{Poisson}(\zeta)$.
2. Elegir $\theta \sim \text{Dir}(\alpha)$.
3. Por cada palabra w_n :
 - Elegir un tópico $z_n \sim \text{Multinomial}(\theta)$.
 - Elegir una palabra w_n de $p(w_n|z_n, \beta)$, una probabilidad multinomial condicionada por el tópico z_n .

Para simplificar el modelo, se asume que la dimensión k de la distribución Dirichlet, se conoce y se puede elegir de antemano. También se asume que la probabilidad de las palabras están representadas por una matriz β cuyas dimensiones son $k \times V$, donde k es la cantidad de tópicos y V es el tamaño del vocabulario del corpus. El término i, j de la matriz β representará la probabilidad de una palabra se presente dado un tópico. Finalmente, la distribución de Poisson no es crítica y se puede elegir cualquier otra distribución que se necesite o se encuentre apropiada.

La probabilidad de densidad de la variable θ de k -dimensiones se presenta como lo siguiente:

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^k \alpha_i)}{\prod_{i=1}^k \Gamma(\alpha_i)} \theta_1^{\alpha_1-1} \dots \theta_k^{\alpha_k-1} \quad (2.32)$$

Donde α es un k -vector con componentes $\alpha_i > 0$ y $\Gamma(x)$ es la función Gamma.

Luego, dado los parámetros α y β , la probabilidad conjunta de la mezcla de tópicos θ , de un conjunto de N tópicos z , y un conjunto de N palabras w está dado por:

$$p(\theta, z, w|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^N p(z_n|\theta) p(w_n|z_n, \beta) \quad (2.33)$$

Luego sumando sobre z e integrando sobre θ se obtiene la probabilidad marginal de un documento:

$$p(w|\alpha, \beta) = \int p(\theta|\alpha) \left(\prod_{n=1}^N \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta) \right) d\theta \quad (2.34)$$

Finalmente, tomando el producto de las probabilidades marginales de los documentos, se obtiene la probabilidad de un corpus:

$$p(D|\alpha, \beta) = \prod_{d=1}^M \int p(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\theta_d) p(w_{dn}|z_{dn}, \beta) \right) d\theta_d \quad (2.35)$$

Observando la inferencia anterior y considerando los supuestos, existen 2 hiperparámetros presentes: α y β . El parámetro α está relacionado con la generación de los tópicos y el parámetro β está relacionado con la generación de los términos.

Finalmente, considerando la probabilidad de que un término pertenezca a un t3pico, es posible la agrupaci3n de 3stos de la misma manera que se hace en LSA, es decir, asignar la palabra al t3pico o grupo en el cu3l tenga m3s peso.

2.4.3. M3tricas para evaluar *Topic Models*

Seg3n lo revisado en la literatura, existen muchos m3todos para poder evaluar qu3 tan bueno es la distribuci3n de t3picos encontrados. A continuaci3n se presentan un par de estas m3tricas que servir3n para evaluar los t3picos encontrados en este trabajo.

1. *Perplexity*: Una de las m3tricas m3s conocidas para evaluar *Topic Models* se denomina *Perplexity*. Su c3lculo se obtiene a partir de la funci3n de verosimilitud del modelo.

Como una breve introducci3n, se tiene que la funci3n de verosimilitud del modelo hace referencia a la productoria de todas las probabilidades de explicar una palabra o documento, dado ciertos t3picos encontrados. La funci3n de log-verosimilitud aplica una escala logar3tmica a la funci3n de verosimilitud. Asumiendo que las palabras que conforman un documento son independientes entre s3, condicionalmente a los t3picos encontrados por el modelo, esta 3ltima toma la siguiente forma:

$$L(D) = \log p(D|\Phi, \alpha) = \sum_d \log p(w_d|\Phi, \alpha) \quad (2.36)$$

Donde D corresponde al conjunto de documentos, w_d corresponde a un documento en particular perteneciente a D , Φ es la matriz que explica el modelo a trav3s de t3picos y por 3ltimo α es un hiperpar3metro que describe la distribuci3n de los documentos en los t3picos.

Esta funci3n tambi3n sirve para comparar modelos. Entre m3s alto es su valor, mejor es el modelo encontrado.

Por otro lado se define la m3trica *Perplexity*:

$$Perplexity(D_t) = e^{\frac{-L(D)}{C_D}} \quad (2.37)$$

Donde D_t corresponde a documentos no vistos por los modelos pero que eventualmente podrían ser explicados por éstos y C_D corresponde a la cantidad de palabras en el documento D .

Perplexity, al contrario de la función de verosimilitud, entre más bajo es su valor, mejor es el modelo encontrado.

Si bien estas métricas tienen sustento matemático basado en las probabilidades de descripción de tópicos, la literatura [3] ha demostrado que estas métricas no están fuertemente relacionadas con la interpretación humana de los tópicos. Esto quiere decir que los modelos pueden describir muy bien los documentos de manera matemática, pero si se observan los documentos dentro de estos tópicos, el contenido de la agrupación no tiene buena interpretabilidad dentro de lo conocido. Por esta misma razón es que [13] define otra métrica basada en la co-ocurrencia de documentos y que mide la coherencia de los documentos dentro del mismo tópico.

2. *Coherence*: Esta métrica fue propuesta por [13] y corresponde a qué tan coherente son los tópicos que se encontraron. Para esto se eligen las palabras más representativas dentro de cada tópico (la cantidad puede variar entre los estudios desde 5 hasta 20). Luego se mide la ocurrencia y la co-ocurrencia de cada palabra dentro de las colección de documentos. A continuación se presenta el cálculo de esta métrica considerando un tópico:

$$Coherence(t) = \sum_{m=2}^M \sum_{l=1}^{m-1} \log \frac{D(w_m, w_l) + 1}{D(w_l)} \quad (2.38)$$

Donde M es la lista de las M palabras más representativas del tópico t , $D(w_l)$ representa la ocurrencia de la palabra w_l dentro de los documentos y $D(w_m, w_l)$ representa la co-ocurrencia de la palabra w_m con la palabra w_l dentro de todos los documentos. Cabe destacar que el término $\frac{D(w_m, w_l) + 1}{D(w_l)}$ de la ecuación 2.38, trata de estimar $P(W_n|W_l)$, es decir, que el documento W_n se presente dado que está el documento W_l . La única diferencia de este término con la probabilidad de co-ocurrencia, es la existencia del término “+1” que se utiliza en 2.38 para evitar valores iguales a 0 y así no causar problemas con la función logaritmo.

2.5. Técnicas de Recomendación

Los sistemas de recomendación son técnicas de análisis de información que proporciona a los usuarios una serie de recomendaciones personalizadas considerando los gustos de éste y los demás usuarios. Estos algoritmos se basan en la gran cantidad de información que generan los usuarios al plasmar sus gustos por algún tipo de tema de interés, por ejemplo, películas, artículos de compra, lugares para viajar, etc. En la literatura existen muchos tipos de sistemas de recomendación, pero en esta ocasión solamente se centrará en los tipos de filtrado colaborativo.

En [23] [9] se realiza un estudio de los distintos algoritmos de filtrado colaborativo, y se asegura que en el caso que se desee recomendar una lista de ítems, es conveniente aplicar técnicas de factorización matricial, más específicamente, la técnica *Non-Negative Matrix Factorization*. En comparación con otras técnicas de factorización de matrices, NMF tiene la ventaja de comportarse de mejor manera con matrices de forma *Sparse* y a su vez, el entrenamiento de los modelos es relativamente más rápido que las otras técnicas de factorización. La desventaja de NMF es que posee variados parámetros que deben ser ajustados. Pese a lo anterior NMF es una buena técnica para aplicar filtrado colaborativo. Otra ventaja importante que posee NMF, es que los resultados obtenidos son de fácil interpretación en el caso que se desee agrupar los ítems encontrados. Considerando que NMF es la mejor factorización para analizar matrices *Sparse*, también se introducirá la factorización SVD como una de las técnicas más básicas.

2.5.1. *Non-Negative Matrix Factorization*

Para comenzar a utilizar esta factorización como una técnica de filtrado colaborativo es necesario definir una matriz $R_{|U| \times |S|}$, que sea no negativa (U se considera como el conjunto de usuarios y S como los ítems en estudio), es decir, no tendrá valores negativos dentro de ésta. Ahora se define la siguiente factorización de R :

$$R \approx P \times Q^T \quad (2.39)$$

Donde P y Q serán matrices de dimensiones $|U| \times K$ y $|S| \times K$ respectivamente. El valor de K debe ser definido previamente. P representará la asociación entre un usuario, el ítem y su atributo (se nombrará así para mayor comprensión) y Q representará la asociación entre un ítem y sus atributos.

Luego para tener la predicción de un ítem S_j a partir del usuario U_i , se debe calcular el producto punto entre S_j y U_i :

$$\hat{r}_{ij} = p_i q_j = \sum_{w=1}^K p_{iw} q_{wj} \quad (2.40)$$

Donde p_i corresponde a la fila i de la matriz P y q_j corresponde a la columna j de la matriz Q .

Ahora que se sabe el cómo se puede obtener la predicción de un ítem, surge otro problema importante y es cómo se realizará la construcción de las matrices P y Q . Para solucionar este problema se debe considerar que al elegir una construcción de Q y P se debe minimizar la diferencia entre la predicción realizada ($\hat{r}_{ij}$) y su valor real (r_{ij}). La diferencia entre estos valores se denominará como e_{ij} y se calculará de la siguiente manera:

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{w=1}^K p_{iw} q_{wj})^2 \quad (2.41)$$

Para minimizar el error descrito anteriormente existe una técnica llamada se utiliza una técnica de optimización denominada gradiente descendente [22]. En esta ocasión se desea minimizar e_{ij}^2 , por ende, se debe modificar la elección de p_{ik} y q_{kj} ya que a partir de estos términos se genera e_{ij}^2 . Para definir cuánto se deben modificar estos términos, se debe calcular el gradiente de e_{ij}^2 en términos de p_{ik} y q_{kj} .

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2e_{ij} q_{kj} \quad (2.42)$$

$$\frac{\partial}{\partial q_{kj}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2e_{ij} p_{ik} \quad (2.43)$$

Obteniendo el gradiente, ahora se puede encontrar iterativamente p_{ik} y q_{kj} tal que se minimice e_{ij} .

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + 2\alpha e_{ij}^2 q_{kj} \quad (2.44)$$

$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + 2\alpha e_{ij}^2 p_{ik} \quad (2.45)$$

Donde p'_{ik} y q'_{kj} serán p_{ik} y q_{kj} respectivamente en la próxima iteración. α corresponde cuánto será la magnitud que se irá agregando del gradiente en la variable que se desea modificar.

Cabe mencionar como punto importante, que lo que se desea hacer en esta ocasión es encontrar una combinación de P y Q tal que la predicción sea muy cercana al valor real y no que la matriz $P \times Q$ sea idéntica a la matriz R . Por esta razón, es que las combinaciones para aumentar p_{ik} y q_{kj} estarán limitadas a los datos que se tienen disponibles. Estos datos serán llamados datos de entrenamiento y servirán para modelar el recomendador. En palabras más técnicas, hay que considerar que se tiene limitados s_i, d_j, r_{ij} que pertenecerán al conjunto T de entrenamiento.

2.5.1.1. Regularización NMF

A continuación se introduce la regularización con el fin de evitar el *overfitting* [16] del modelo. Para esto se introduce un nuevo parámetro denominado β en el cálculo del error cuadrático de las predicciones. Este parámetro es utilizado para generar una matriz P y Q que sirva para aproximar R sin utilizar números con grandes magnitudes. El cálculo del nuevo error considerando β es el siguiente:

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{w=1}^K p_{iw} q_{wj})^2 + \frac{\beta}{2} \sum_{w=1}^K (\|P\|^2 + \|Q\|^2) \quad (2.46)$$

Finalmente el cálculo iterativo de p_{ik} y q_{kj} queda de la siguiente manera:

$$p'_{ik} = p_{ik} + \alpha(2e_{ij}^2 q_{kj} - \beta p_{ik}) \quad (2.47)$$

$$q'_{kj} = q_{kj} + \alpha(2e_{ij}^2 p_{ik} - \beta q_{kj}) \quad (2.48)$$

Se puede evidenciar que el método de filtrado colaborativo aplicando NMF posee 3 parámetros importantes que se deben regular antes de generar el modelo:

- K : Cantidad que regula la dimensionalidad del subespacio correspondiente a las matrices a generar.
- α : Parámetro correspondiente al tamaño de paso en el algoritmo de gradiente descendente.
- β : Parámetro que determina la importancia relativa del regularizador en el modelo.

A continuación se presenta un algoritmo que se basa en el presentado por [26]

Data: K : Parámetro para dimensión de P y Q .

α : Adición de Gradiente.

β : Regularizador.

M : Datos de entrenamiento (Matriz).

Result: P y Q , matrices utilizadas para las predicciones.

```

1 while No se supere error ( $e$ ) mínimo do
2   for por cada fila  $i$  en  $M$  do
3     for por cada columna  $j$  en  $M$  do
4       Calcular el error  $e_{ij}$ ;
5       for  $k=1$  hasta  $K$  do
6         Calcular  $p_{ik} = p_{i(k-1)} + \alpha(2e_{ij}^2 q_{(k-1)j} - \beta p_{i(k-1)})$ ;
7         Calcular  $q_{kj} = q_{(k-1)j} + \alpha(2e_{ij}^2 p_{i(k-1)} - \beta q_{(k-1)j})$ ;
8       end
9     end
10  end
11  Calcular nuevo  $M$  a partir de los nuevos  $P$  y  $Q$ ;
12  Calcular  $e$  considerando regularización;
13 end

```

Algorithm 7: Algoritmo NMF Filtrado Colaborativo.

2.5.2. Singular Value Decomposition

Al igual que NMF la técnica *Singular Value Decomposition* (SVD) puede ser utilizada también como base para realizar una técnica de filtrado colaborativo. Esto se realiza en base a la

descomposición matricial que genera esta técnica:

$$A \approx U\Sigma V^T \quad (2.49)$$

Donde A es una matriz de R filas y C columnas.

Se aprecia que las columnas de U y V son vectores ortonormales que definen los vectores singulares de A . Σ es una matriz diagonal que contiene los valores singulares de la matriz A . U corresponde a una matriz de $R \times K$ que representa los vectores singulares correspondientes a las columnas de la matriz original. V es una matriz de $C \times K$ que representa los vectores singulares correspondientes a las filas de la matriz original. El valor de K debe ser definido previamente para la descomposición.

La premisa de esta técnica es que los vectores singulares representan la varianza de la matriz original, por lo tanto, si se eligen los vectores singulares más grandes para representar A , se generará una aproximación de matriz más fidedigna de la matriz A . Por lo tanto uno de los desafíos de esta descomposición es encontrar un K tal que se elijan los vectores singulares más representativos y su vez las matrices U y V tengan las menores dimensiones posibles.

Por otro lado, el objetivo de utilizar esta descomposición es encontrar una configuración de K vectores singulares tal que las filas y las columnas (para este caso, usuarios y *skills*) puedan ser representados por puntos en un espacio K dimensional.

2.5.2.1. Regularización SVD

Al igual que NMF es posible realizar una regularización para encontrar las matrices U y V tal que se evite el fenómeno de *overfitting*. Para esto se debe minimizar el error residual, definido como:

$$r_{ij} = A_{ij} - \hat{A}_{ij} \quad (2.50)$$

donde $\hat{A}$ corresponde a la aproximación generada por U y V .

Para encontrar las matrices U y V para minimizar el error residual se utiliza un método

iterativo similar al que se utilizó en NMF:

$$u'_{ik} = u_{ik} + \alpha(r_{ij}v_{jk} - \lambda u_{ik}) \quad (2.51)$$

$$v'_{jk} = u_{ik} + \alpha(r_{ij}u_{ik} - \lambda v_{jk}) \quad (2.52)$$

Donde u_{ik} y v_{jk} corresponden a elementos de las matrices U y V respectivamente. u'_{ik} y v'_{jk} serán u_{ik} y v_{jk} en la próxima iteración. α y λ son parámetros que definirán en qué cantidad irán cambiando u_{ik} y v_{jk} .

A continuación se presenta un algoritmo de Filtrado Colaborativo basado en SVD:

Data: K : Parámetro para dimensión de U y V .

α : Regularizador.

λ : Regularizador.

M : Datos de entrenamiento (Matriz).

Result: U y V , matrices utilizadas para las predicciones.

```

1 while No se supere error ( $r$ ) mínimo do
2   for por cada fila  $i$  en  $M$  do
3     for por cada columna  $j$  en  $M$  do
4       Calcular el error  $r_{ij}$ ;
5       for  $k=1$  hasta  $K$  do
6         Calcular  $u_{ik} = u_{i(k-1)} + \alpha(r_{ij}v_{j(k-1)} - \lambda u_{i(k-1)})$ ;
7         Calcular  $v_{jk} = v_{j(k-1)} + \alpha(r_{ij}u_{i(k-1)} - \lambda v_{j(k-1)})$ ;
8       end
9     end
10  end
11  Calcular nuevo  $M$  a partir de los nuevos  $U$  y  $V$ ;
12  Calcular  $r$ ;
13 end

```

Algorithm 8: Algoritmo SVD Filtrado Colaborativo.

2.5.3. Determinando la calidad de los predictores/recomendadores

En esta sección se presentarán algunas métricas que sirven para evidenciar qué tan bueno es el recomendador que se está construyendo. Estas métricas también se han utilizado en [7]. Antes de presentar estas métricas es necesario definir conceptos previos que hacen referencia a los estados de predicción:

- Verdaderos Positivos (VP): Son la cantidad de datos que el algoritmo selecciona como relevantes y que realmente son relevantes.
- Falsos Negativos (FN): Son la cantidad de datos que el algoritmo no selecciona como relevantes y que son relevantes.
- Falsos Positivos (FP): Son la cantidad de datos que el algoritmo selecciona como relevantes y que realmente no son relevantes.
- Verdaderos Negativos (VN): Son la cantidad de datos que el algoritmo no selecciona como relevantes y que no son relevantes.

Considerando como relevantes, los datos etiquetados que se desean seleccionar.

A continuación se presentan las distintas métricas:

1. *Precision*: Es la proporción de predicciones correctas con respecto al total de predicciones realizadas.

$$Precision = \frac{VP}{FP + VP} \quad (2.53)$$

2. *Recall*: Es la proporción de predicciones correctas con respecto a las que debiesen haber sido realizadas.

$$Recall = \frac{VP}{FN + VP} \quad (2.54)$$

3. *F-measure*: Esta métrica mezcla *Precision* y *Recall*. Esta medida es importante ya que la información obtenida por *Precision* y *Recall* por si solas no entrega información tajante de qué tan bueno está el predictor. Por ejemplo, considerar un *dataset* de 100 datos, los cuales 50 están como positivos y 50 como negativo. Si el modelo predice que los 100 datos son positivos entonces el *Recall* será de un 100 %, pero la *Precision* será un de 50 %, por lo que no se podría decir tajantemente que el modelo es un buen predictor.

F-Measure se encarga de combinar la información de ambas métricas y dar un valor más fidedigno sobre la calidad del predictor. Esta medida se define como la media armónica entre *Precision* y *Recall*:

$$F\text{-Measure} = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (2.55)$$

Finalmente no está demás mencionar que la técnicas más conocida para realizar validación de modelos de recomendación es la técnica llamada validación cruzada [6], la cual consta en dividir el *dataset* de trabajo en m partes, utilizando así $m - 1$ partes para la construcción del recomendador. Luego con la parte restante se procede a evaluar la calidad de los recomendadores considerando que son datos que no se conocen en realidad (datos de *test*). Este proceso se realiza iterativamente cambiando los grupos que se utilizan para entrenar los recomendadores.

Capítulo 3

Metodología

En esta sección se presentará las distintas etapas que se desarrollarán para poder cumplir con el objetivo general de esta memoria.

La primera etapa consiste en la recolección semi-automática de datos y trata sobre la extracción de distintos perfiles de informática desde la plataforma *LinkedIn*. Este proceso es semi-automático ya que en ocasiones el método propuesto comienza a capturar perfiles que no son de interés para el estudio, por ejemplo, profesionales de la USM que no son informáticos.

Luego procede la limpieza y representación de datos. En este paso se procesa la información obtenida en la parte anterior para luego generar la limpieza de los datos (eliminar perfiles que no sirven para el estudio, procesar la información para que sea más legible, solucionar problemas semánticos) y luego realizar su respectiva descripción, es decir, realizar un análisis exploratorio de los datos.

Por último se realiza la minería de datos, es decir, a los datos previamente procesados se les aplica distintas técnicas para encontrar información que a simple vista no se puede analizar. En esta ocasión se realizaron 3 procesos: agrupamiento de *skills*, agrupamiento de usuarios y la creación de un predictor de *skills*.

3.1. Recolección semi-automática de datos

La recolección de los datos a trabajar es una parte importante de esta memoria, por lo mismo se dedicó tiempo y esfuerzo en obtener datos correspondientes a distintos perfiles que estuvieran relacionados con el Departamento de Informática tanto de la Universidad Técnica Federico Santa María, como de las universidades que se consideran que están a un nivel similar respecto al área de informática.

En esta ocasión la obtención de los datos está concentrado en ex-estudiantes del Departamento de Informática de distintas universidades que podían ser localizados gracias a las herramientas que la misma plataforma *LinkedIn* contiene. Las búsquedas realizadas se concentraron en 4 universidades específicas: Universidad Técnica Federico Santa María, Universidad de Santiago de Chile, Pontificia Universidad Católica de Chile y Universidad de Chile. Cabe mencionar que el área de informática en otras universidades es nombrada también como Computación. Otro punto importante son los derivados de los nombres de las distintas universidades, siendo la Universidad Técnica Federico Santa María la que tiene más derivados. A continuación se presentan los distintos derivados de las Universidades:

- Universidad Técnica Federico Santa María: UTFSM, USM.
- Universidad de Santiago de Chile: USACH.
- Pontificia Universidad Católica de Chile: PUC, UC.
- Universidad de Chile: UCH.

Al momento de buscar perfiles se realizó una búsqueda de todas las universidades considerando sus derivados y las 2 áreas de especialidad nombradas anteriormente: Informática y Computación. En total se generaron 17 combinaciones entre universidades y especialidad para realizar las consultas al buscador de *LinkedIn*. Por otro lado, como alumno de la UTFSM se obtuvieron privilegios para ingresar a 2 grupos para la obtención de perfiles: DI Alumni-Universidad Técnica Federico Santa María y Extensión Informática UTFSM.

Luego de obtener las combinaciones de perfiles y especialidad, se procede a realizar la obtención de datos utilizando el software *Octoparse*¹. El funcionamiento del software se basa en un diagrama de flujos que se debe configurar para programar qué sitios visitará. En esta ocasión el flujo de trabajo consistía en una lista de búsquedas, ingresar al perfil, extraer los datos del perfil y volver a la lista para visitar el siguiente perfil.

La tabla 3.1 presenta los datos que se pudieron obtener con *Octoparse* y su breve descripción.

Cuadro 3.1: Datos extraídos de los perfiles de *LinkedIn*

Dato	Descripción
Url Out	Identificador único de cada perfil
Cargo Actual	Cargo que desempeña en la empresa que trabaja.
Empresa Actual	Empresa en la que trabaja actualmente.
País	País donde reside actualmente.
Skills	Lista de Skills que ha declarado el usuario del perfil.
Sector	Sector de la industria en el cual trabaja.
Título	Último título obtenido.
Educación	Última universidad en la que se graduó.
Trayectoria	Toda la trayectoria profesional del perfil.
Grupos	Grupos a los cuales pertenece el perfil.
Educación_Histórica	Toda la educación del perfil. Contiene Título, Universidad y Fechas.
Universidades	Universidades en las cuales ha estado el usuario del perfil.
Recomendaciones	Recomendaciones que ha recibido el usuario.

Luego de definir qué datos serán obtenidos, considerando mantener siempre el anonimato de cada perfil, se procede a realizar la programación de *Octoparse* para la obtención de los perfiles. Es necesario mencionar que hubieron perfiles cuyos datos no podían ser obtenidos ya que eran usuarios de tercer grado o simplemente los perfiles no tenían el campo solicitado. Otra situación importante y que afecta a la calidad del *dataset* adquirido, son los perfiles que no tienen relación con la búsqueda solicitada, generalmente estos perfiles están en las

¹<http://www.octoparse.com/> Visitado última vez: 30-08-2016

últimas páginas de búsquedas por lo que fue necesario definir un *threshold* para terminar la búsqueda en el caso de que los perfiles extraídos no tuvieran sentido con lo requerido. Por la razón anterior es que el proceso de *Scraping* no fue netamente automático sino que también fue necesario supervisar periódicamente en el caso de que se comenzara a recolectar perfiles que no son del interés del estudio. Considerando los 2 puntos anteriores se puede decir que el *dataset* obtenido tendrá muchos campos en blanco y perfiles sin información importante, por lo que será necesario realizar distintas limpiezas de éste al momento de terminar con el proceso de *Scraping*.

3.2. Limpieza y representación

3.2.1. Selección de atributos y perfiles a estudiar

Antes de comenzar el trabajo de *data mining* en los datos recolectados, es necesario realizar una limpieza de éstos y seleccionar qué atributos serán los que se utilizarán para continuar el estudio.

El filtrado de los perfiles es necesario ya que como se dijo anteriormente existen muchos registros con atributos vacíos o con contenido que no es de interés para el estudio, por ejemplo, informáticos de otras universidades, estudiantes de otras carreras que no son del Departamento de Informática de la Universidad Técnica Federico Santa María, etc.

En este estudio se utilizarán solo los siguientes atributos: Educación Histórica (Universidad y Título obtenidos durante toda su trayectoria educacional) y *Skills*. Se decidió de esta manera ya que considerar más variables al problema agregará más dificultad al estudio que podría atrasar los resultados de éste.

3.2.2. Obtención y limpieza de *skills*

Luego de obtener todos los perfiles de las universidades que se estudiarán, se procede a obtener todas las *skills* disponibles a trabajar. A continuación se procede a realizar un análisis

exploratorio en los datos para luego observar en qué grado se deben solucionar los problemas de *skills* descritos por [1] (Desambiguación y Duplicación).

La Desambiguación en este estudio no se encuentra presente de manera importante, ya que la obtención de los perfiles (por ende, las *skills*) se realizó concentrándose solo en un sector de la industria, que en este caso, es la industria informática.

La Duplicación si está presente en este estudio y es de suma importancia solucionar este problema. Para esto se consideraron 2 subproblemas de duplicación: Una misma *skill* escrita con distinto idioma y 2 *skills* que poseen el mismo significado. Para solucionar el subproblema del idioma se decidió trasladar todas las *skills* al idioma Español, por la sencilla razón de que los perfiles en estudio están en este idioma. Para el segundo subproblema se decidió utilizar la misma solución otorgada por [1] y que es utilizar como comparador *Wikipedia* para observar si una *skill* tiene el mismo significado que otras *skills*. Más específicamente, 2 o más términos se buscan en la plataforma *Wikipedia*, si éstos son dirigidos al mismo resultado (misma página de destino de *Wikipedia*), quiere decir que poseen el mismo significado.

3.2.3. Descripción de los datos obtenidos

Finalmente se procede a realizar un estudio de los datos que finalmente se tienen disponibles, observando así la cantidad de perfiles que se tienen dependiendo de la universidad, la cantidad de *skills* disponibles, observar la frecuencia de estas *skills* dependiendo de la universidad, etc. En el capítulo 4 se dará más detalle sobre la información obtenida de los datos.

3.2.4. Representación de los datos

La representación de los datos es algo fundamental para el rendimiento de las distintas técnicas que se aplicarán en este estudio. Típicamente, un método de minería de datos recibe como entrada una matriz X donde las filas corresponden a los ítems, casos o registros a analizar y las columnas corresponden a los atributos que describen esos registros. Como se mencionó al inicio de este capítulo, es posible buscar grupos de *skills* a partir del agrupamiento directo

de éstas o a partir de agrupamiento de perfiles. Por lo tanto, se considerarán 2 casos posibles de matriz de entrada X a un método de *clustering*:

El primer caso corresponde a la representación en la cual una *skill* está presente en distintos perfiles, lo que implica que cada fila represente una *skill* y cada columna represente un perfil en estudio. La tabla 3.2 muestra una representación matricial de este caso con n usuarios y p *skills*.

Cuadro 3.2: Representación matricial *skills* v/s *users*.

<i>Skills / Users</i>	id_user_1	id_user_2	...	...	id_user_n
id_skill_1	0	1	...	...	1
id_skill_2	0	1	...	...	0
...	...	...	...	...	...
...	...	...	...	...	...
id_skill_p	1	0	...	...	1

El segundo caso corresponde a la representación de que un perfil posea ciertas *skills*, esto implica que cada fila representa un perfil y cada columna representa una *skill* en estudio. La tabla 3.3 muestra una representación matricial de este caso considerando n usuarios y p *skills*.

Cuadro 3.3: Representación matricial *users* v/s *skills*.

<i>User/Skill</i>	id_skill_1	id_skill_2	...	...	id_skill_p
id_user_1	0	0	...	...	1
id_user_2	1	1	...	...	0
...	...	...	...	...	...
...	...	...	...	...	...
id_user_n	1	0	...	...	1

Luego se tiene que cada nuevo *dataset* puede ser trabajado de 2 maneras distintas. La primera

manera es utilizar una representación normal de una matriz, es decir, crear una estructura matriz que guarde en memoria tanto las ocurrencias y las no ocurrencias del *dataset* estudiado. El segundo caso corresponderá a la representación como matriz *Sparse*, donde solamente se guarda en memoria las ocurrencias del *dataset* estudiado. Se realiza esta diferenciación ya que algunos métodos implementados requieren específicamente alguna de las representaciones nombradas anteriormente.

3.2.5. Normalizaciones

En esta memoria se aplicaron distintas normalizaciones a los datos descritos en la sección anterior:

- Normalización L2: Considerar un vector X con n atributos, la normalización L2 consiste en generar un nuevo vector $y = \frac{x}{z}$, tal que:

$$z = \|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2}, \quad (3.1)$$

por lo que la norma del nuevo vector y , será 1.

Para ejemplificar de mejor manera, considerar la matriz presentada en la tabla 3.2, cada elemento de la fila (*skill*) de una matriz se dividirá por la raíz de la suma de los cuadrados de cada elemento de esa fila, este proceso se repite para todas las filas.

- Normalización TF-IDF: La sigla TF-IDF quiere decir *Term-Frequency Inverse Document-Frequency*, por lo que la función que se aplica a cada término deriva de este nombre. *Term-Frequency* hace alusión a la función que entrega como valor la cantidad de ocurrencias que se presenta un término en un documento. Si se considera la representación de la tabla 3.2, significa la cantidad de veces que una *skill* se presenta en un perfil, pero si se considera la representación de la tabla 3.3, significaría la cantidad de veces que un perfil declara cierta *skill*. De manera general $tf(t, d)$ significa:

$$tf(t, d) = \text{Cantidad de veces que se presenta el término } t \text{ en el documento } d. \quad (3.2)$$

El término *Inverse Document-Frequency* se presenta con la siguiente función:

$$idf(t) = \log \frac{1 + n_d}{1 + df(d, t)} + 1 \quad (3.3)$$

Donde n_d es el número total de documentos, y $df(d, t)$ es el número de documentos que contienen el término t . Si se considera la representación de la tabla 3.2, $df(d, t)$ será el número de perfiles que contienen la *skill* t , en cambio, si se utiliza la representación de la tabla 3.3, $df(d, t)$ será el número de *skills* declaradas por el perfil t .

Luego a cada término de una matriz X , se aplica la función $tf(t, d) \times idf(t)$.

Finalmente a los vectores (filas) resultantes después de realizar el proceso TF-IDF, se normalizan con la norma L2.

- No normalización: En este caso se dejan las entradas de las matrices sin modificar. Si bien esto no se considera como una normalización, acá se tratará como tal para efectos de redacción.

3.3. Minería

3.3.1. Agrupamiento de *Skills*

El primer método propuesto para encontrar grupos de *skills* relacionadas entre sí consiste en aplicar un algoritmo de *clustering* a las filas de la matriz *skill* v/s *users* definida en la sección anterior. Para determinar estos grupos se aplicarán distintos algoritmos de *clustering* y de *Topic Models* presentados en la sección 2.3 y 2.4 respectivamente.

Considerando que no existe una agrupación de *skills* definida, la calidad de los *clusters* y los *Topic Models* generados se tendrán que medir con método intrínsecos.

En esta ocasión los parámetros óptimos para cada algoritmo *clustering* se seleccionarán considerando aquella configuración que entregue mejor calidad de *clusters* y a su vez genere una agrupación que sea interpretable para el estudio. La calidad de *clusters* se medirá a través

de 2 métodos: Coeficiente de *Silhouette* y una adaptación del Promedio *Coherence* para algoritmos *clusterings*, presentados en la sección 2.3.4 y 2.4.3. La adaptación del Promedio *Coherence* consiste en calcular el Promedio *Coherence* considerando un *ranking* donde los pesos de las *skills* se basan en las frecuencias observadas en el *dataset*.

Para los *clusters* que poseen jerarquía se utilizarán 3 métricas distintas. La primera métrica se utilizará para medir el comportamiento interno del algoritmo. Esta métrica será el *Cophenetic Correlation Coefficient* presentado en la sección 2.3.2.5. Las otras 2 métricas se utilizarán con el ánimo de poder encontrar algún argumento para poder utilizar algún tipo de configuración distinta para este algoritmo. Las métricas nombradas anteriormente serán el Coeficiente de *Silhouette* y el Promedio *Coherence*.

Respecto a los *Topic Models*, los parámetros se definirán netamente considerando la métrica del Promedio *Coherence*. En estos algoritmos, el parámetro más importante a definir es la cantidad de tópicos que se generarán.

Otra técnica que se utilizará será la mezcla de la técnica *K-Means* con *Topic Models*, aplicando el algoritmo *K-Means* a la mejor configuración encontrada dentro de los *Topic Models*. Las métricas que se utilizarán para encontrar la mejor configuración para este algoritmo serán el Coeficiente de *Silhouette* y el Promedio *Coherence*.

Por otro lado, se realizará la agrupación aleatoria de *skills* con grupos de tamaño uniforme, para así observar cómo se comportan las métricas utilizadas en los distintos algoritmos empleados.

La representación de los *clusters* y *Topic Models* generados se realizará con una técnica gráfica llamada *Word Cloud*, la cual considerará como tamaño de las palabras la frecuencia de declaración de las *skills* para el caso de los algoritmos que no son *Topic Models*. Para los *Topic Models* el tamaño de las palabras será el peso de ésta dentro del tópico. En este caso, para la generación de las *Word Clouds*, se incluirán las *top 30 skills* que tengan frecuencia superior a 1 para los algoritmos que no son *Topic Models*. Para los algoritmos de *Topic Models* se integrarán las palabras cuyo peso sea superior al promedio de pesos de las *top 30* palabras del tópico. En la figura 3.1 se presenta un ejemplo de un *Word Cloud*.

Finalmente, considerando los distintos grupos de *skills* encontrados, se verá qué proporción de usuarios de distintas universidades declaran sus *skills* dependiendo de los grupos generados.



Figura 3.1: Ejemplo de *Word Cloud*.

3.3.2. Agrupamiento de Perfiles

El segundo método propuesto consiste en aplicar un algoritmo de *clustering* a las filas de la matriz *user* v/s *skills* definida en la sección anterior, de modo de encontrar grupos de perfiles, y luego determinar el conjunto de *skills* más relevantes para cada grupo encontrado. Para determinar estos grupos se aplicarán distintos algoritmos de *clustering* y de *Topic Models* presentados en la sección 2.3 y 2.4 respectivamente.

Respecto a las métricas sucede la misma situación que en el objetivo de agrupar *skills*. Como no existe una agrupación de usuarios definida, la calidad de los *clusters* y *Topic Models* generados se tendrán que medir con métodos intrínsecos.

La elección de parámetros para cada algoritmo se realizará en base a la calidad del *clusters*

que se medirá a través de las métricas seleccionadas y a su vez observado la distribución de los usuarios dentro de los distintos *clusters* generados. La calidad de los *clusters* se medirá a través de 2 métodos: Coeficiente de *Silhouette* y una adaptación del Promedio *Coherence* para algoritmos de *clustering* presentados en la sección 2.3.4 y 2.4.3. La adaptación del Promedio *Coherence* consiste en calcular el Promedio *Coherence* considerando un *ranking* donde los pesos de los usuarios se basa en la cantidad de declaraciones de *skills* realizadas en el *dataset*.

En lo *Topic Models* los parámetros se definirán netamente considerando la métrica del Promedio *Coherence*. En estos algoritmos el parámetro más importante a elegir es la cantidad de tópicos que se generarán.

Otra de las técnicas que se utilizará será la mezcla de la técnica *K-Means* con *Topic Models*, aplicando el algoritmo *K-Means* a la mejor configuración encontrada dentro de los *Topic Models*. Las métricas que se utilizarán para encontrar la mejor configuración serán el Coeficiente de *Silhouette* y la variación del Promedio *Coherence*.

Por otro lado, se realizará la agrupación aleatoria de usuarios con grupos de tamaño uniforme, para así observar cómo se comportan las métricas utilizadas en los distintos algoritmos empleados.

La representación de los *clusters* y *Topic Models* generados se realizará con la técnica gráfica llamada *Word Cloud*. Para la descripción de estos *clusters* se utilizarán las *skills* que fueron declaradas por los grupos de usuarios encontrados. El tamaño de las *skills* presentadas será la cantidad de veces que ocurre la *skill* en las declaraciones de los usuarios en el *cluster*. Para la generación de las *Word Clouds*, se incluirán las *top 40 skills*, y a su vez que tengan frecuencia superior a 1 para los algoritmos que no son *Topic Models*. Para los algoritmos de *Topic Models* se integrarán las palabras cuyo peso sea superior al promedio de pesos de las *top 30* palabras del tópico.

Finalmente, considerando que los grupos de usuarios obtenidos se pueden también interpretar como grupos de *skills*, se verá qué proporción de usuarios de distintas universidades declaran sus *skills* dependiendo de los grupos generados.

3.3.3. Predicción de Competencias No Declaradas

Para la creación de un predictor que infiera las *skills* no declaradas por los perfiles, se deben aplicar distintas técnicas de recomendación que se presentaron en la sección 2.5.

Para realizar el entrenamiento de los predictores generados será necesario considerar un nuevo *dataset* generado a partir del datos originales. En este nuevo *dataset* será necesario ocultar un cierto porcentaje de declaraciones de *skills* existentes en los datos originales. Luego para validar el modelo, se comparará las *skills* recomendadas con las que efectivamente posee la persona.

Luego de encontrar los parámetros óptimos de cada algoritmo considerando las métricas *Precision*, *Recall* y *F-Measure* presentadas en la sección 2.5.3, se analizará cada resultado, para luego realizar las respectivas interpretaciones.

Finalmente se mostrarán algunos casos de recomendación de *skills* para distintos usuarios de *LinkedIn* que se tengan disponibles.

Capítulo 4

Experimentos

4.1. Recolección semi-automática de datos

Como se mencionó en la sección 3.1, para la recolección de datos se utilizó un *software* llamado *Octoparse*.

El cómo funciona este *software* se basa en la utilización de un diagrama de flujo que simula la navegación de una página WEB como si fuera un usuario de la vida real. Al momento de configurar el diagrama, el usuario del programa puede especificar qué campos de información quiere obtener para luego ser llevados a una base de datos, que en este caso fue una base de datos SQL. Los campos de información más variables eran guardados como sección de código HTML. Se decidió esto ya que la información de cada perfil variaba respecto a cantidad y a la información que se obtenía de éstos. En la figura 4.1 se presenta la interfaz de *Octoparse*.

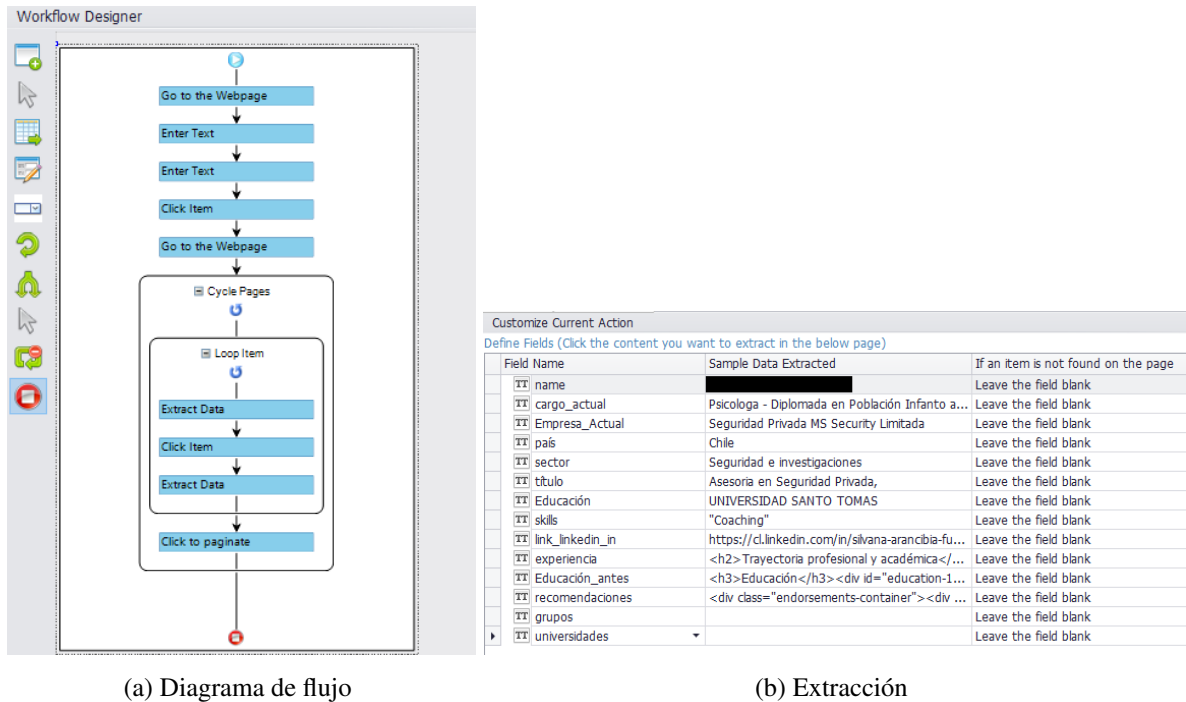


Figura 4.1: Interfaz de diagrama de flujo y extracción.

Utilizando el método de búsqueda que se describió en la sección 3.1 se lograron obtener **7890** perfiles de *LinkdIn*.

4.2. Limpieza y análisis exploratorio de datos

4.2.1. Selección de atributos y perfiles a estudiar

Luego de tener los **7890** perfiles, se procede a la obtención de la carrera profesional de cada individuo, es decir, del campo Educación.Histórica, se tendrá el Título profesional y la Universidad de la cual obtuvo ese título. Cabe mencionar que éste es uno de los campos que está extraído como sección HTML, debido a la variabilidad de su contenido. Por ejemplo, un usuario puede tener 3 títulos identificando la Universidad y fecha de obtención, en cambio, otro usuario puede poseer solo 1 título mencionando solamente el nombre de éste, sin rellenar los demás campos.

Respecto al desarrollo, para obtener la información relevante del segmento HTML extraído fue necesario utilizar conocimientos en **Expresiones Regulares**. La herramienta que se utilizó para este propósito fue **Python 2.7** con la librería *Regular expression operations*¹ y *CSV File Reading and Writing*².

Luego de obtener todos Títulos de los perfiles, se procede a obtener los perfiles de las 4 universidades en estudio considerando solamente títulos que tengan que ver con los departamentos de informática. Luego de procesar la selección se obtuvieron la siguiente cantidad de perfiles:

1. Universidad Técnica Federico Santa María: 857
2. Pontificia Universidad Católica de Chile: 456
3. Universidad de Santiago de Chile: 463
4. Universidad de Chile: 665

En total se obtuvieron **2441** perfiles para el estudio.

Para representar cada conjunto de datos de cada universidad se generaron 5 archivos, 4 de éstos contendrán los perfiles separados por universidad y el último tendrá la totalidad de todos los perfiles. El archivo está estructurado de la siguiente manera:

```
id_perfil_1: nombre_skill_1, nombre_skill_3, nombre_skill_5, . . .
id_perfil_2: nombre_skill_2, nombre_skill_3 . . .
.
.
id_perfil_n: nombre_skill_2, nombre_skill_1...
```

Por lo tanto, cada fila representará un perfil en estudio. Esta fila contendrá el *id* del perfil que se está estudiando junto con las *skills* que posee, separadas por ','.

¹<https://docs.python.org/2/library/re.html>

²<https://docs.python.org/2/library/csv.html>

4.2.2. Obtención y limpieza de *skills*

Luego de obtener todos los perfiles en estudio, se procede a obtener todas las *skills* que están presentes dentro de los perfiles. En total se obtuvieron **6237** *skills*, asociada cada una con su respectivo *id*.

A continuación se procede a la traducción de todas las *skills* obtenidas desde inglés a español, para esto se utilizó un proceso no automático, el cual fue la traducción manual desde la página de traducción de *Google*³. Se decidió utilizar este motor de traducción ya que la traducción de muchas *skills* posee contexto, por ejemplo, la traducción de **Microsoft Office** es traducida como **Microsoft Office** y no como *Oficina Microsoft*. Por otro lado se eligió la traducción manual ya que las librerías gratuitas disponibles están restringidas por tiempo y número de consultas, mientras que librería oficial de Google cobra dinero por carácter traducido y además también tiene un límite de caracteres traducidos.

Luego de relacionar la *skill* traducida con su respectiva *id*, se procede a realizar la búsqueda de la página de *Wikipedia* de cada *skill* traducida. Para esto se utilizó la librería **Wikipedia**⁴ compatible con Python 2.7. Esta librería permite obtener el *link* de la página de *Wikipedia* sobre algún término el particular. En la tabla 4.1 se aprecia un par de ejemplos de la agrupación resultante.

Cuadro 4.1: Agrupación realizadas con *Wikipedia*.

Grupo	<i>Skills</i> agrupadas
Microsoft PowerPoint	Powerpoint, Power Point, Microsoft PowerPoint
PMO	PMO, Gestión de Proyectos, Dirección de Proyectos
COBOL	Cobol Microfocus, Lenguaje COBOL, Cobol
Relaciones Públicas	Relaciones Públicas, Publicidad y relaciones públicas, Relación con los medios.

Respecto al desarrollo, se utilizó un diccionario de listas para agregar los identificadores de las *skills* que tuvieran el mismo resultado identificado con *Wikipedia*. En el caso que existiera

³<https://translate.google.cl/>

⁴<http://wikipedia.readthedocs.io/en/latest/>

más de una *skill* asociada a ese grupo, se nombraba al grupo como la *skill* que tuviera más frecuencia dentro de todos los perfiles en estudio. En el caso que no existiera algún resultado esta *skill* se dejaría con el mismo nombre. Finalmente con las *skills* ya desambiguadas se procede a asociar estos grupos con el nombre original de las *skills* (nombres sin traducir) y eliminar la repetición de *skills* dentro del mismo perfil.

Considerando el proceso anterior la cantidad de *skills* a analizar bajó a un total de **4350**.

4.2.3. Descripción de los datos obtenidos

Luego de obtener todas las *skills* que se analizarán, se procede a realizar un análisis cuantitativo de las *skills* disponibles.

En la imagen 4.2 se puede apreciar las *skills* que son más declaradas por los usuarios informáticos de las distintas universidades en la plataforma *LinkedIn*

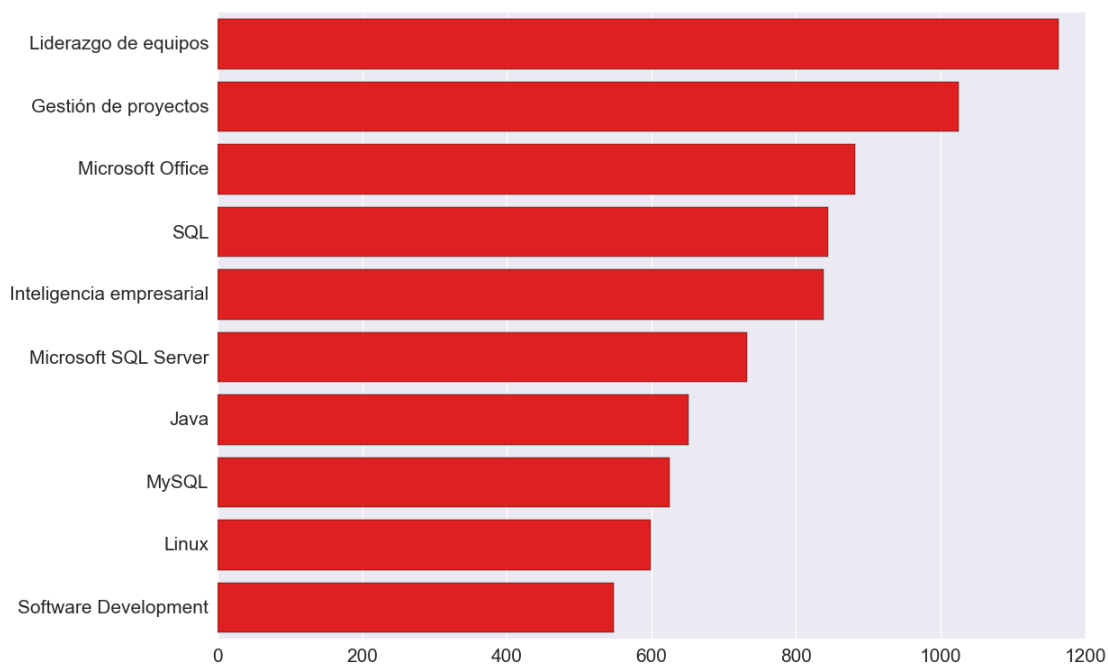


Figura 4.2: Top 10 de *skills* con mayor frecuencia en los perfiles.

En el gráfico 4.3, se puede apreciar cuántas *skills* son necesarias considerar para explicar cierto porcentaje de ocurrencia de *skills* dentro de los perfiles. Para realizar este gráfico las *skills* se ordenaron de manera descendente respecto a su frecuencia en los perfiles. Se evidencia que agregando 2225 *skills* se puede explicar un 95 % de las ocurrencias dentro de los perfiles.

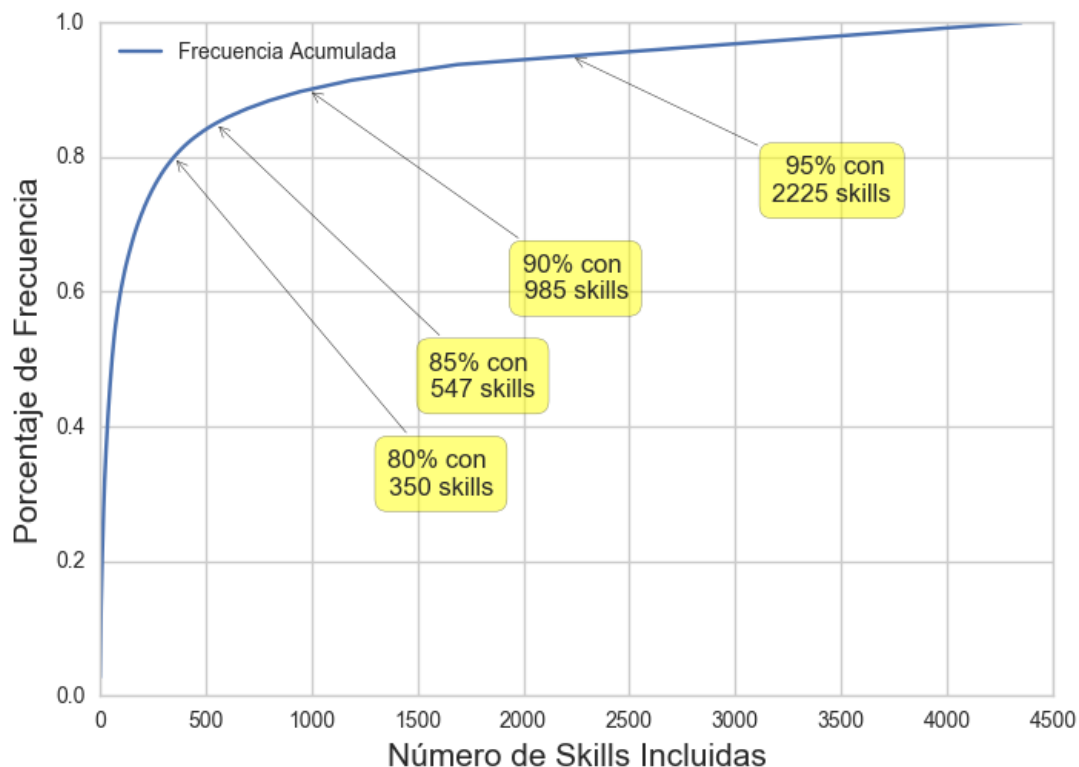


Figura 4.3: Frecuencia explicada a través de la integración de *skills*.

Por otro lado, se descubrió que existen 3669 *skills* cuya frecuencia es menor a 5. En la tabla 4.2 se puede observar la cantidad de *skills* que tienen solo esa frecuencia y el porcentaje en relación a la cantidad de *skills* en total.

Cuadro 4.2: Cantidad de *skills* con poca frecuencia.

Frecuencia	Cantidad	Porcentaje
5	107	2 %
4	146	3 %
3	241	6 %
2	504	12 %
1	2671	61 %

Ahora, observando los datos desde el punto de vista de los perfiles, se puede evidenciar en la tabla 4.3 existen 355 usuarios que declaran menos de 5 *skills* en sus perfiles.

Cuadro 4.3: Cantidad de usuarios que declaran pocas *skills*.

Skills declaradas	Cantidad	Porcentaje
5	270	12 %
4	15	1 %
3	16	1 %
2	20	1 %
1	34	2 %

Por otro lado, se presenta el gráfico 4.4 que representa la proporción de *skills* declaradas por los perfiles de distintas universidades divididas en todas las declaraciones realizadas en el *dataset*.

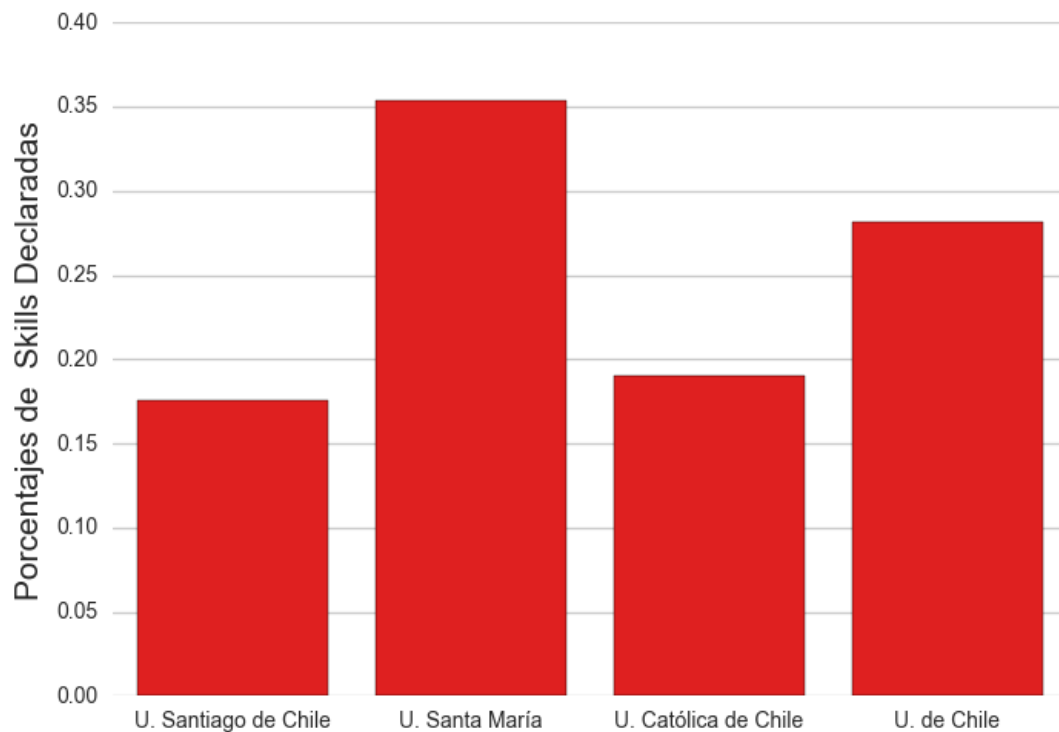


Figura 4.4: Cantidad de *skills* declaradas por las universidades.

Se evidencia claramente que la Universidad Técnica Federico Santa María y la Universidad de Chile, poseen más declaraciones de *skills* dentro sus perfiles. Si se realiza una comparación más equitativa considerando la cantidad de perfiles en cada universidad, se evidencia que el promedio de declaración de *skills* por persona de todas las universidades fluctúa entre 17 y 19.

Finalmente, considerando las componentes principales de la matriz de ocurrencia, se presenta el gráfico 4.5 que muestra la explicación de varianza de las primeras 500 componentes. Se evidencia que después de las 430 componentes, se comienza a presentar una acumulación de varianza superior al 90 %.

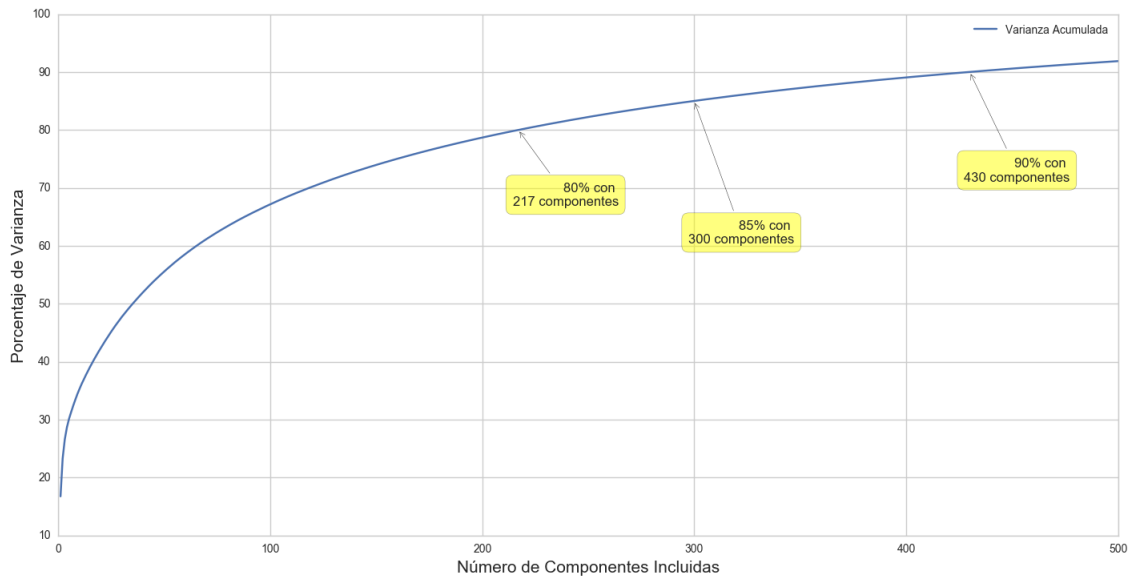


Figura 4.5: Varianza acumulada en función del número de componentes.

4.3. Aplicación de técnicas de minería de datos

A continuación se presentarán los procesos que se consideraron para realizar los experimentos en los distintos objetivos que se tienen en esta memoria. Para desarrollar todos los experimentos que tienen relación con los objetivos de esta memoria, se utilizó el lenguaje de programación **Python 2.7** en conjunto a distintas librerías. A continuación se nombrarán las librerías más destacables:

- **Scikit-learn**⁵: Librería que implementa muchos algoritmos de *Machine Learning*.
- **SciPy**⁶: Librería que contiene muchas implementaciones de algoritmos para disciplinas como las matemáticas, ciencia e ingeniería.
- **Numpy**⁷: Librería con implementaciones que ayudan al soporte de algoritmos relacionados a la ciencia de datos.

⁵<http://scikit-learn.org/>

⁶<https://www.scipy.org/scipylib/index.html>

⁷<http://www.numpy.org/>

- **SeaBorn**⁸: Librería de visualización de datos.
- **Pandas**⁹: Librería cuya utilidad es destinada hacia el fácil manejo de análisis de datos.

4.3.1. Agrupamiento de *Skills*

Para encontrar algún tipo de relación entre las distintas *skills* que se disponen, se optó por utilizar el *dataset* que contiene las *skills* (filas) y su ocurrencia en distintos perfiles (columnas).

Para encontrar algún tipo de agrupación de estas *skills* se utilizaron 3 tipos de algoritmos: Los algoritmos de *clustering*, los algoritmos asociados con los *Topic Models* y la mezcla de los algoritmos *Topic Models* y *K-Means*.

Dentro de los algoritmos de *clustering* que se implementaron se encuentran los siguientes: *K-Means*, *DBSCAN* y *Ward*.

Respecto a los algoritmos relacionados a los *Topic Models* que se implementaron en este trabajo fueron los siguientes: LDA, LSA y NMF. Para la mezcla de algoritmos se utilizó LSA y LDA en conjunto con *K-Means*, debido a que estos algoritmos distribuyen los pesos de manera más equitativa dentro de los tópicos encontrados. En el caso de NMF los pesos siempre apuntaban a que una palabra pertenecía sólo a ese tópico, es decir, el peso de esa palabra en ese tópico era muy cercano a 1.

A continuación se detallarán los experimentos que se realizaron para encontrar la configuración óptima de parámetros de cada algoritmo.

Para la aplicación a todos los algoritmos se realizaron 3 modificaciones a la matriz de ocurrencias (*skills* v/s usuarios):

- Normalizaciones: Los datos antes de ser trabajados se normalizaban de 3 maneras distintas. La primera es la no normalización, que se contará como una normalización

⁸<https://seaborn.pydata.org/>

⁹<http://pandas.pydata.org/>

para efectos de redacción. La segunda normalización es la norma L2 y por último la tercera normalización corresponde a la normalización TF-IDF (*Term frequency – Inverse document frequency*). Para mayor comprensión de la normalización TF-IDF, en este caso se considera las *skills* como los términos y los perfiles como documentos.

- Reducción de dimensionalidad: Existe la opción de reducir la dimensionalidad de la matriz original considerando 430 dimensiones, lo que representa un 90 % de la varianza total. La técnica utilizada para realizar la reducción de dimensionalidad fue SVD.
- Eliminar *skills* con frecuencia menor a X : A la matriz original se eliminan las *skills* que tengan frecuencia menor a X .

Cabe mencionar que las configuraciones anteriores eran compatibles entre sí, realizando variadas representaciones para el mismo *dataset*.

Para encontrar la mejor configuración, se variarán los parámetros de los distintos algoritmos para luego aplicar la técnica a las distintas estructuras descritas anteriormente. A continuación se presenta la variación de parámetros para cada algoritmo:

K-Means

- K se varió desde 2 hasta $\sqrt{\frac{n}{2}}$, donde n es la cantidad de *skills* disponibles, que en este caso es 4350, por lo que finalmente K se variará desde 2 hasta 46.

DBSCAN

- **Cantidad Mínima de Puntos** se varió desde 1 hasta 3.
- **Epsilon** se varió desde 1 hasta 30, realizando saltos de 0,1.

Ward

- **Distancia mínima de unión** se varió desde 1 hasta 100, realizando saltos de 0,1.

Topic Models

- **Cantidad de tópicos** se varió desde 2 hasta 14, realizando 30 mediciones por configuración debido a que los resultados obtenidos no eran deterministas.

LSA+K-Means y LDA+K-Means

- **Cantidad de tópicos** se eligió el que tuviera mejores resultados considerando las 30 mediciones de *Topic Models*.
- **K** se varió de 1 hasta 25.

No está demás mencionar que los hiperparámetros del algoritmo LDA no fueron modificados, por lo que solo se utilizaron los que venían por defecto en la implementación utilizada.

Como se mencionó en la sección 3.3.1 se utilizarán distintas métricas para los algoritmos descritos anteriormente. Para *K-Means*, *DBSCAN*, *Ward*, *LSA+K-Means* y *LDA+K-Means* se utilizará tanto el Coeficiente de *Silhouette* como la variación del Promedio *Coherence*. Para los *Topic Models* se utilizará solo el Promedio *Coherence*.

Por otro lado, para la configuración de tópicos en los algoritmos *Topic Models* se optó por utilizar solamente la variación de estructura que consiste en eliminar todas las *skills* que tengan declaraciones inferiores a 1. Respecto a la cantidad máxima de tópicos a generar solo se eligieron 14 ya que se consideró un buen número para poder analizar los grupos resultantes de forma manual.

Luego, considerando las agrupaciones de *skills* encontradas en esta sección, se observará por cada universidad en estudio qué proporción de usuarios declara al menos una *skill* dado un cierto *cluster*.

Finalmente, analizando los algoritmos que encontraron agrupaciones, se generarán grupos aleatorios uniformes para luego comparar los distintos valores de las métricas utilizadas y verificar que los algoritmos empleados poseen mejor rendimiento que la agrupación aleatoria.

4.3.2. Agrupamiento de Perfiles

En esta ocasión se utilizó el *dataset* que contiene las declaraciones de *skills* (columnas) de cada usuario (filas).

Para encontrar algún tipo de agrupación de los distintos usuarios se utilizaron 3 tipos de algoritmos: Los algoritmos *clustering*, los algoritmos asociados con los *Topic Models* y la mezcla de algunos algoritmos *Topic Models* con *K-Means*.

Dentro de los algoritmos de *clustering* que se implementaron se encuentran los siguientes: *K-Means* y *DBSCAN*.

Los algoritmos relacionados a los *Topic Models* que se implementaron fueron los siguientes: LDA, LSA y NMF. Para la mezcla de algoritmos se utilizó LSA y LDA en conjunto con *K-Means* por la misma razón que se mencionó en el caso de agrupación de *skills*.

Respecto a las modificaciones al *dataset*, a la sintonización de parámetros de los algoritmos y las distintas métricas para evaluar los algoritmos, se utilizarán lo mismo que en la sección de agrupación de *skills* 4.3.1, con la diferencia que para las modificaciones al *dataset* se invertirá la representación de la matriz y se eliminarán usuarios que tengan menos de X declaraciones. Otra diferencia es la interpretación de términos en la normalización TF-IDF, en este caso se considera los usuarios como los términos y las *skills* como documentos.

Por otro lado, para la configuración de tópicos en los algoritmos *Topic Models* se optó por utilizar solamente la variación de estructura que consiste en eliminar todos los usuarios que hayan realizado menos de 5 declaraciones. Respecto a la cantidad de tópicos a generar solo se eligieron 14 ya que se consideró un buen número para poder analizar los grupos resultantes de forma manual.

Luego, considerando las agrupaciones de usuarios descritas por las declaraciones de *skills* de estos mismos, se observará por cada universidad en estudio qué proporción de usuarios declara al menos una *skill* dado un cierto *cluster*.

Finalmente, analizando los algoritmos que encontraron agrupaciones, se generarán grupos aleatorios uniformes para luego comparar los distintos valores de las métricas utilizadas y

verificar que los algoritmos empleados poseen mejor rendimiento que la agrupación aleatoria.

4.3.3. Predictor de Competencias

Para la creación de un predictor que infiera *skills* se utilizó el *dataset* que contiene las declaraciones de *skills* (columnas) de cada usuario (filas).

Para la creación de este recomendador, se utilizaron 2 algoritmos descritos en la sección 2.5, NMF y SVD.

Respecto a la validación de los modelos creados con estos algoritmos, es necesario tener 2 *datasets* importantes: El conjunto de entrenamiento y el conjunto de pruebas.

Como se explicó en la sección 3.3.3, se necesita esconder un cierto porcentaje de ocurrencias de *skills* en los perfiles. Para la creación de este *dataset* se consideraron todas las ocurrencias existentes y se seleccionó un cierto porcentaje para ser transformadas a no ocurrencias.

Para esta ocasión el porcentaje elegido para realizar el conjunto de entrenamiento fue de un 30 %.

Por otro lado, los parámetros que se variarán serán 2:

- **N:** La cantidad de dimensiones que tendrán los vectores que aproximarán las matrices. Este parámetro se variará desde 2 hasta 481, realizando saltos de 20 dimensiones.
- **Tolerancia:** Este valor indicará con qué valor aproximado de la matriz aproximada se considera como una recomendación o no. Este valor variará desde 0,1 hasta 0,9, realizando saltos de 0,1.

Las métricas que se considerarán para evaluar los modelos recomendadores son las siguientes: *Precision*, *Recall* y *F-Measure*. Estas validaciones se realizarán considerando el conjunto de pruebas (datos sin ocurrencias ocultas) y el conjunto de entrenamiento (datos con ocurrencias ocultas).

Capítulo 5

Resultados

5.1. Agrupaciones de *skills* resultantes

En la sección 4.3.1 se nombraron distintas técnicas para poder encontrar grupos que están implícitos en los datos. A continuación se muestran los resultados de los experimentos realizados. Como conocimiento previo, se utilizó una notación para simplificar la lectura de la información de los gráficos generados en esta sección, a continuación se presenta la codificación de cada leyenda:

- $df > X$ indica que se consideraron todas las *skills* que fueran declaradas a lo menos X veces.
- NN, L2 y TF-IDF, corresponde a la normalización de los datos utilizados. NN indica que no hubo normalización en los datos, L2 indica que los datos fueron normalizados con norma L2 y por último TF-IDF indica que se utilizó la normalización TF-IDF. En este caso, para mayor comprensión de TF-IDF, se considera las *skills* como los términos y los perfiles como documentos.
- $D0$ indica que a los datos obtenidos no se le redujo la dimensión, en cambio, $D430$ indica que los datos fueron reducidos hasta 430 dimensiones.

5.1.1. *K-Means*

La primera técnica que se utilizó para agrupar las *skills* de los usuarios a través de sus declaraciones, fue la técnica *K-Means*.

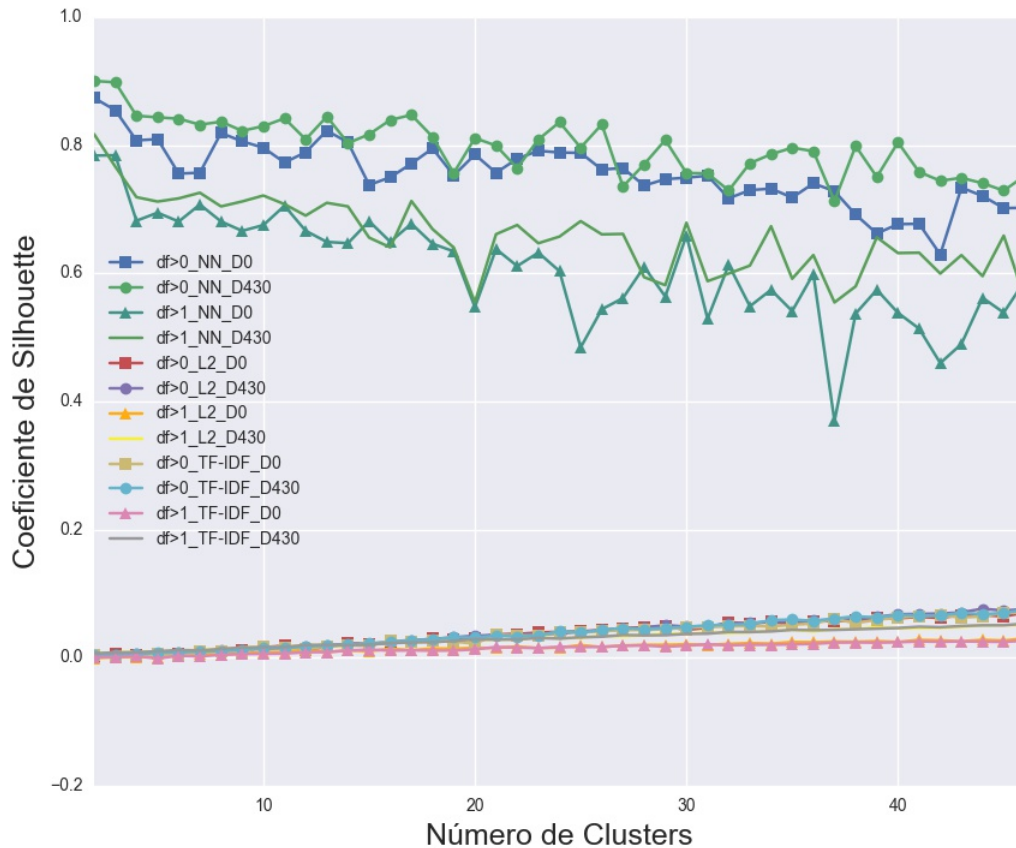
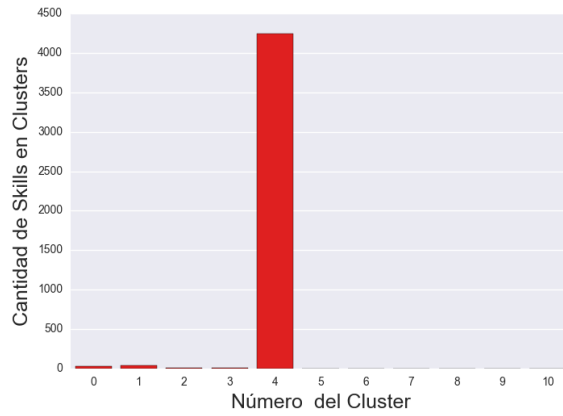


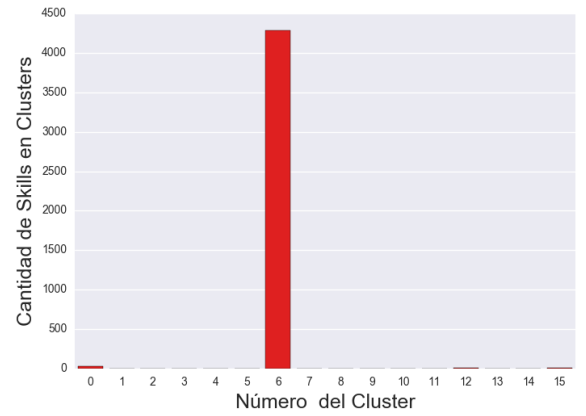
Figura 5.1: *K-Means*: Coeficiente de *Silhouette* en función de la cantidad de *clusters* considerando la agrupación de *skills*.

En el gráfico 5.1 se presenta cómo evoluciona el Coeficiente de *Silhouette* considerando distintas cantidades de *clusters* y distintas normalizaciones de datos teniendo en cuenta también el cambio en las dimensiones de éstos. Como se puede apreciar, la normalización L2 y TF-IDF, no genera datos cuyos *clusters* puedan ser considerados buenos tomando en cuenta la métrica de *Silhouette*. Por otro lado, se evidencia que cuando no se normalizan los datos y

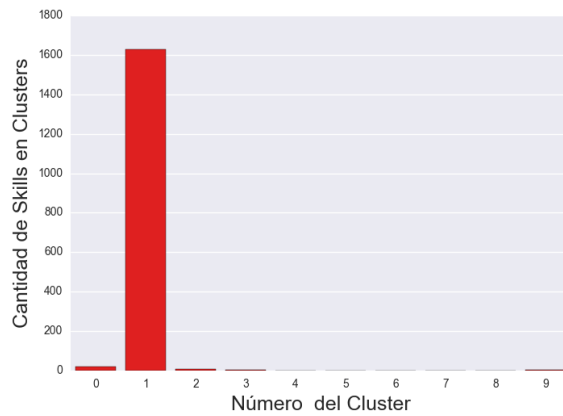
se consideran toda las *skills*, el Coeficiente de *Silhouette* obtenido considerando las distintas agrupaciones es bastante cercano a 1, tomando así valores entre 0,7 y 0,9. Sin embargo, al momento de considerar las cantidades de *skills* que tiene cada *cluster*, se evidencia que no se puede interpretar de buena manera lo encontrado.



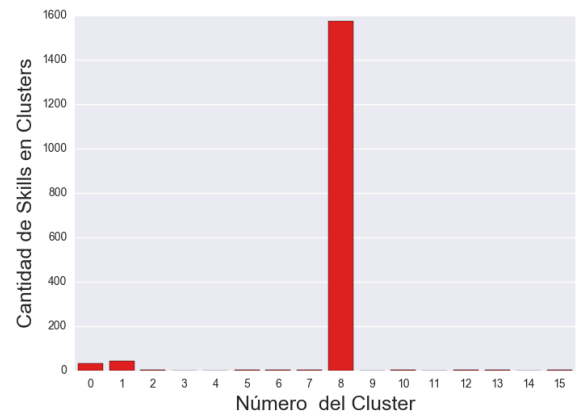
(a) $df > 0$ y sin reducción de dimensionalidad



(b) $df > 0$ y con reducción de dimensionalidad



(c) $df > 1$ y sin reducción de dimensionalidad



(d) $df > 1$ y con reducción de dimensionalidad

Figura 5.2: Distintas configuraciones de *K-Means* sin normalizar los datos considerando la agrupación de *skills*.

En las figuras de 5.2 se muestra la cantidad de *skills* que tiene cada *cluster* generado a través de la técnica *K-Means*, considerando los datos sin normalizar, pero variando las dimensiones y la frecuencia mínima de cada *skill*. Se puede notar que en estos casos, cuyas agrupaciones

presentan un Coeficiente de *Silhouette* alto, siempre existe un *cluster* que posee la mayoría de las *skills* o una cantidad muy grande de éstas en comparación con los demás *clusters*. Esta situación no es favorable para el estudio, ya que lo que se requiere, son *clusters*, que posean un alto Coeficiente de *Silhouette* y a su vez, la distribución de la cantidad de *skills* en cada *cluster* sea homogénea, es decir, que cada grupo pueda ser representado con una misma cantidad de *skill*.

Por otro lado, también se aplicó la métrica Promedio *Coherence* considerando las 30 *skills* que tuvieran más frecuencia dentro de cada *cluster*.

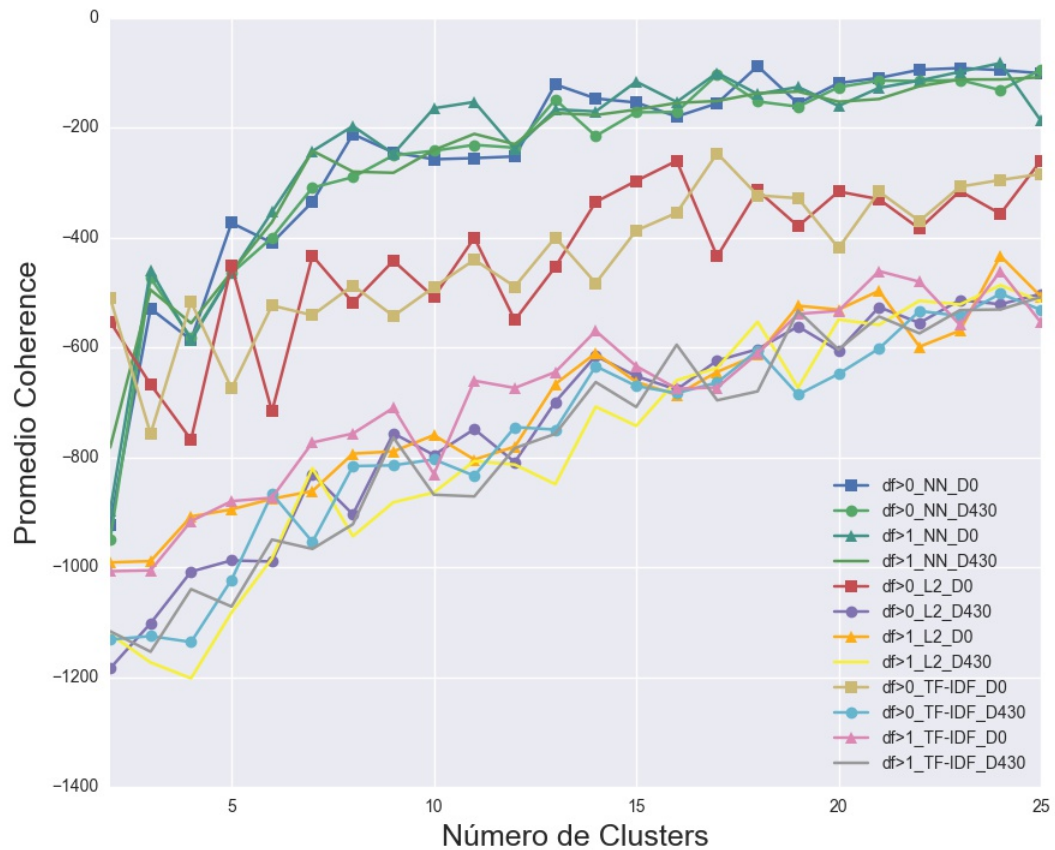
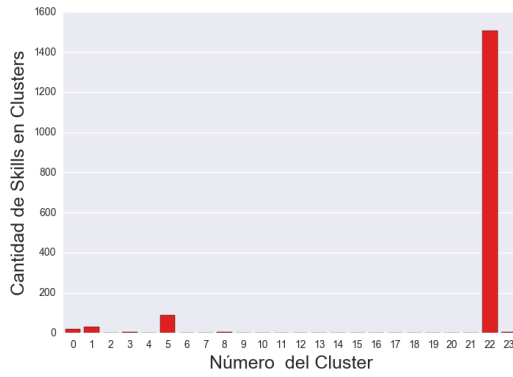
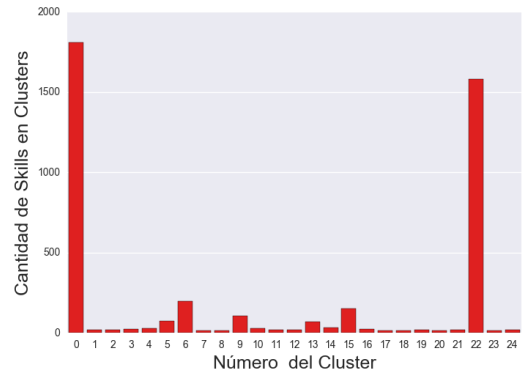


Figura 5.3: *K-Means*: Promedio *Coherence* con distintas normalizaciones de datos considerando la agrupación de *skills*.

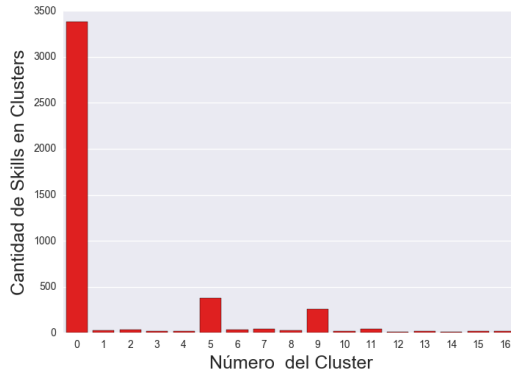
En la gráfico 5.3 se evidencia que a medida que aumentan la cantidad de *clusters* también aumenta el Promedio *Coherence* para las distintas configuraciones de normalización. Considerando el gráfico 5.1 se tiene que el comportamiento es totalmente contrario para la no normalización. Para las normalizaciones L2 y TF-IDF, se presenta un crecimiento más notorio que en el Coeficiente de *Silhouette*. Debido a que ya se sabe cómo se distribuyen las *skills* en los casos no normalizados, ahora se observará la distribución de *skills* en los distintos *clusters* considerando la mejor configuración de cada normalización, es decir, la configuración que otorgue mejor valor del Promedio *Coherence*.



(a) Sin normalizar, $df > 1$ y con reducción de dimensionalidad.



(b) Normalización L2, $df > 0$ y sin reducción de dimensionalidad.



(c) Normalización TF-IDF, $df > 0$ y sin reducción de dimensionalidad.

Figura 5.4: Distribución de las distintas normalizaciones de *K-Means* considerando la agrupación de *skills*.

En los gráficos 5.9 se evidencia que las normalizaciones de las mejores configuraciones considerando el Promedio *Coherence* para L2 y TF-IDF presentan una distribución de *skills* similar que las presentadas para el caso de no normalización, por lo tanto, se determina que en *K-Means* no se presenta una agrupación de *skills* analizable considerando las métricas de Coeficiente de *Silhouette* y el Promedio *Coherence*.

Hay que destacar que a diferencia de la investigación [11], acá no fue posible encontrar agrupaciones de *skills* utilizando la técnica *K-Means* en la matriz de ocurrencias.

5.1.2. DBSCAN

La segunda técnica que se utilizó para agrupar las *skills* encontradas en los distintos usuarios en estudio, fue la técnica *DBSCAN*.

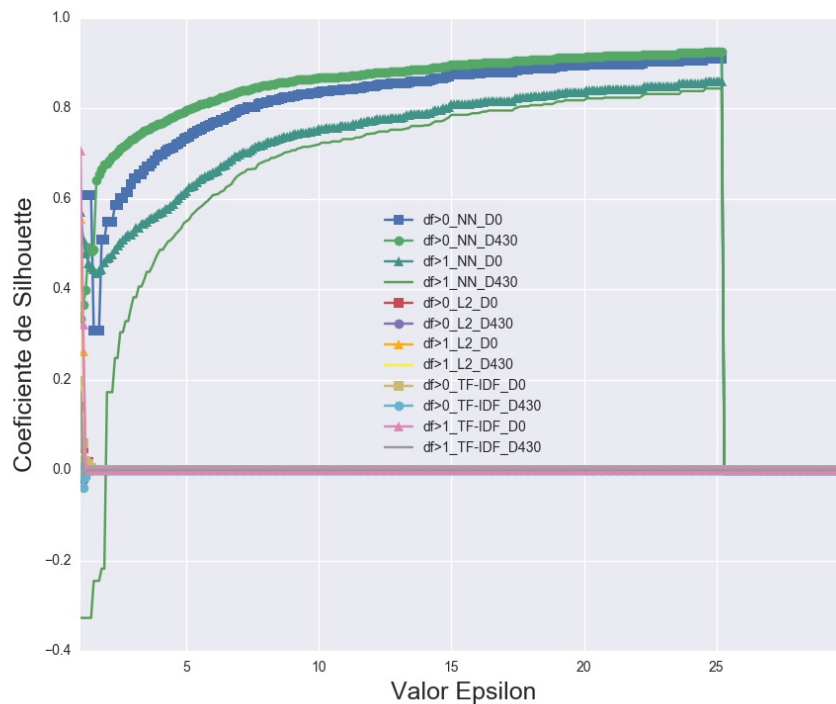


Figura 5.5: *DBSCAN*: Coeficiente de *Silhouette* con distintas normalizaciones de datos considerando la agrupación de *skills*.

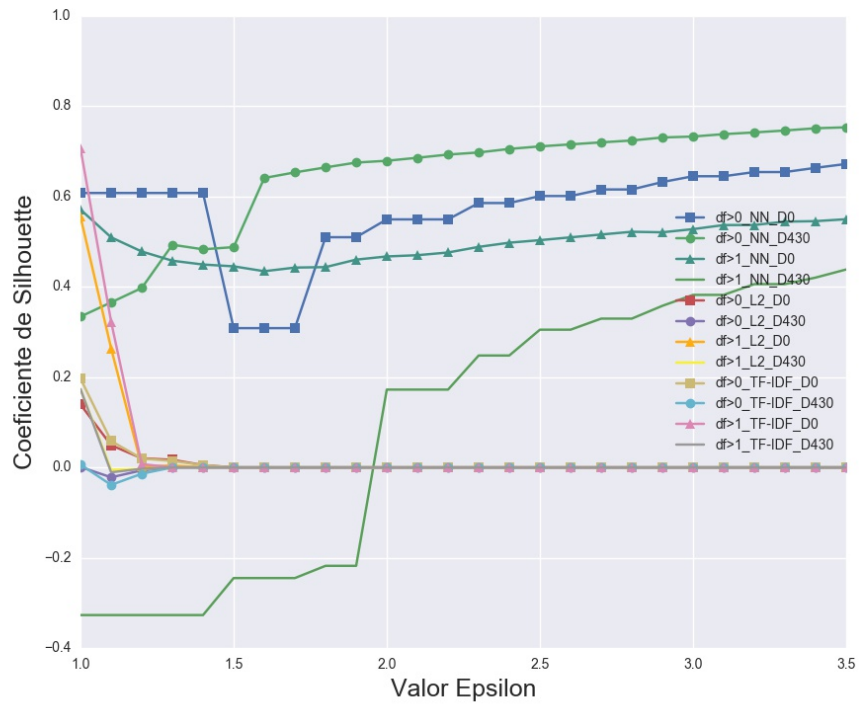


Figura 5.6: *DBSCAN*: Coeficiente de *Silhouette* con distintas normalizaciones de datos con rango de *Epsilon* disminuido y considerando la agrupación de *skills*.

En el gráfico 5.5 se presenta la evolución del Coeficiente de *Silhouette* considerando los distintos valores de *Epsilon* y tomando en cuenta los mínimos puntos que den mejor valor de esta métrica. Se evidencia que luego del valor de *Epsilon* igual a 25,3, las curvas correspondientes a los datos sin normalizar, tienden abruptamente a 0. Lo anterior es a causa que el radio donde se toman en cuenta los puntos centrales es muy alto, por lo tanto considera a toda la *data* como un solo *cluster* por lo que no se puede calcular el Coeficiente de *Silhouette*. En el gráfico 5.6 se aprecia el mismo fenómeno pero en mayor detalle con respecto a las curvas de las normalizaciones L2 y TF-IDF, éstas suelen tender a 0 en el valor de *Epsilon* igual a 1,5 a causa del mismo fenómeno descrito para las curvas no normalizadas.

Luego con el fin de tener más métricas para realizar los futuros análisis se procedió a la medición del Promedio *Coherence* variando el valor de *Epsilon* y considerando los mínimos puntos que den mejor valor de este promedio.

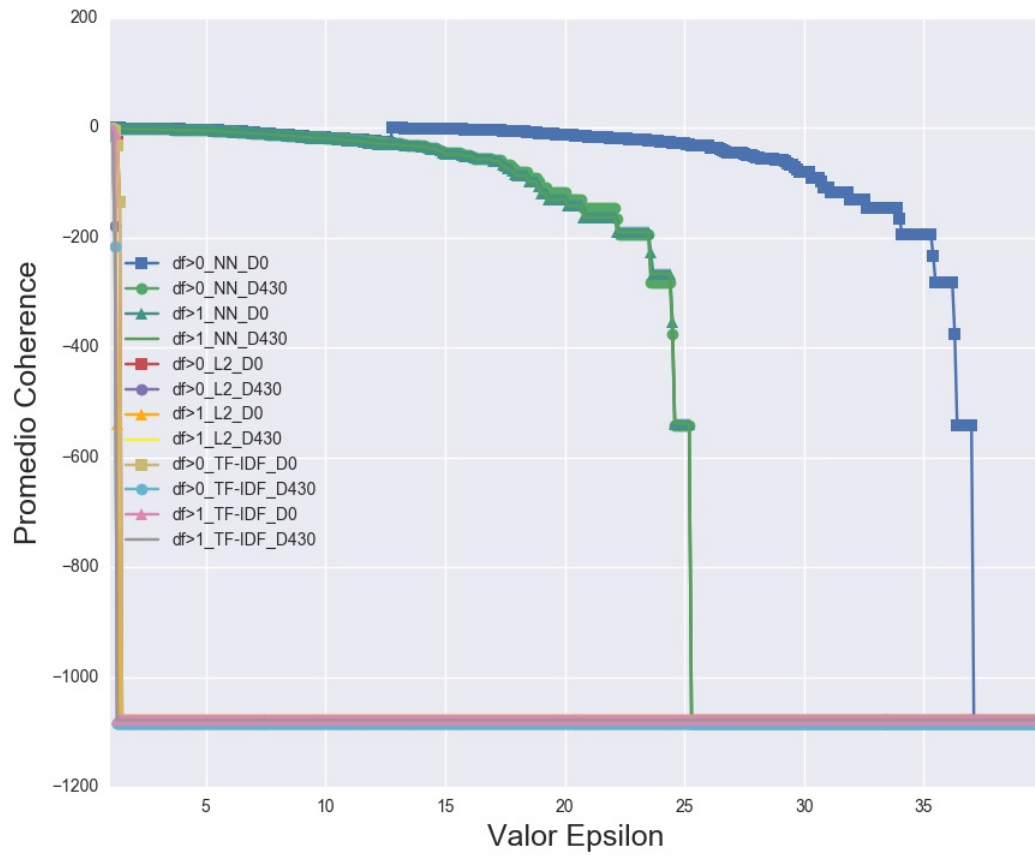


Figura 5.7: *DBSCAN*: Promedio *Coherence* con distintas normalizaciones de datos considerando la agrupación de *skills*.

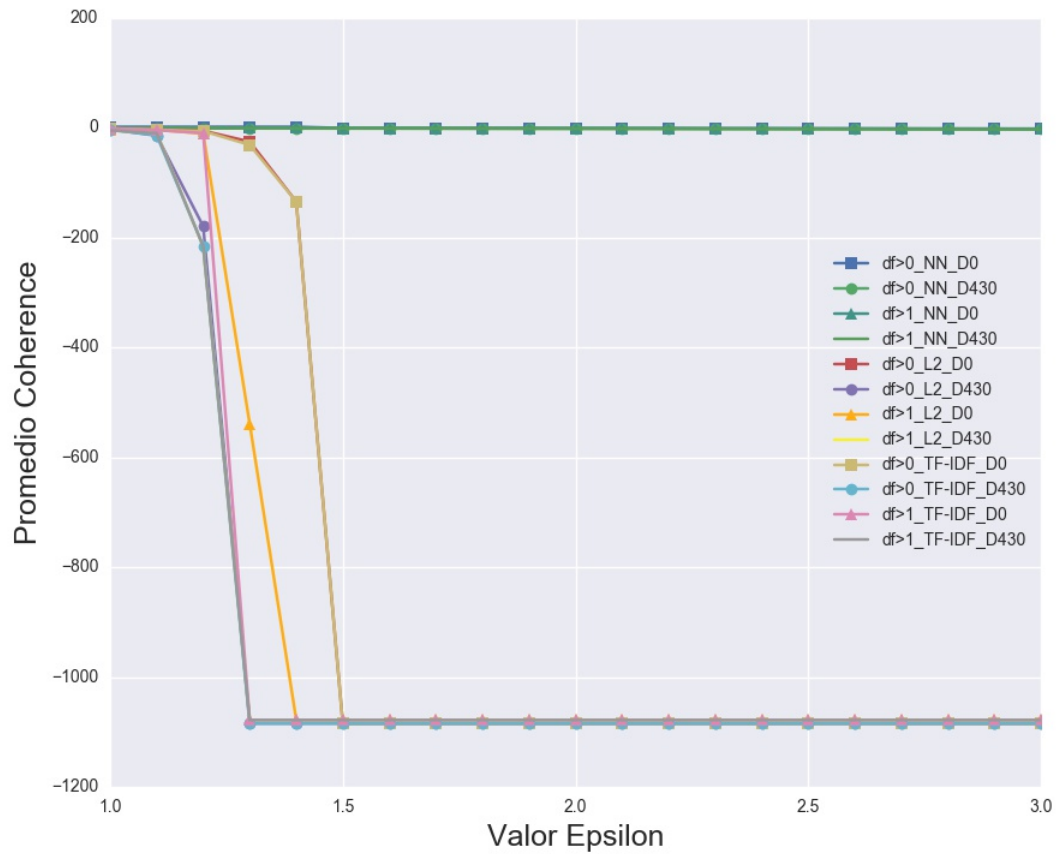


Figura 5.8: *DBSCAN*: Promedio *Coherence* con distintas normalizaciones de datos con rango de *Epsilon* disminuido y considerando la agrupación de *skills*.

Se aprecia en el gráfico 5.7 que el Promedio *Coherence* decrece a medida que *Epsilon* aumenta. Esto quiere decir, que entre más aumente *Epsilon*, menos *clusters* habrá por lo que esta métrica se hará más negativa debido a que habrán menos grupos por los cuales dividir y obtener un menor promedio. El otro comportamiento interesante, es que nuevamente las curvas que no fueron normalizadas poseen un comportamiento distinto a las normalizadas. Las curvas no normalizadas comienzan a decrecer abruptamente luego del valor de *Epsilon* igual a 1.5. En cambio para las normalizaciones L2 y TF-IDF, este decaimiento ocurre rápidamente entre el valor de *Epsilon* 1,2 y 1,4. Esto ocurre fundamentalmente por la cantidad de *clusters* que se encuentran dependiendo de las normalizaciones.

Debido a que la limitación en esta técnica es que la cantidad de *clusters* encontrados sean analizables, también se verá cómo se distribuyen las *skills* dentro de los *clusters* encontrados.

A continuación se presenta unas tablas donde se muestra el valor de *Epsilon* y la cantidad de *clusters* encontrados considerando las distintas configuraciones y normalizaciones:

Cuadro 5.1: Número de *clusters* variando el valor de *Epsilon*, considerando datos sin normalizar.

Epsilon	df>0_NN_D0	df>0_NN_D430	df>1_NN_D0	df>1_NN_D430
1,0	1806	895	1031	30
1,1	1806	829	912	30
1,2	1806	766	823	30
1,3	1806	711	762	30
1,4	1806	660	708	30
1,5	931	626	656	74
1,6	931	588	607	74
1,7	931	558	562	74
1,8	762	531	543	28
1,9	762	506	519	28
10,1	72	72	72	2
16,0	27	27	27	2
16,1	26	26	26	2
22,5	7	7	7	2
22,6	6	6	6	2
25,2	2	2	2	2
25,3	1	1	1	1
25,4	1	1	1	1

Cuadro 5.2: Número de *clusters* variando el valor de *Epsilon*, considerando datos con normalización L2.

Epsilon	df>0_L2_D0	df>0_L2_D430	df>1_L2_D0	df>1_L2_D430
1,0	909	500	1079	530
1,1	539	83	636	159
1,2	185	5	145	5
1,3	42	1	2	1
1,4	8	1	1	1
1,5	1	1	1	1
1,6	1	1	1	1
1,7	1	1	1	1

Cuadro 5.3: Número de *clusters* variando el valor de *Epsilon*, considerando datos con normalización TF-IDF.

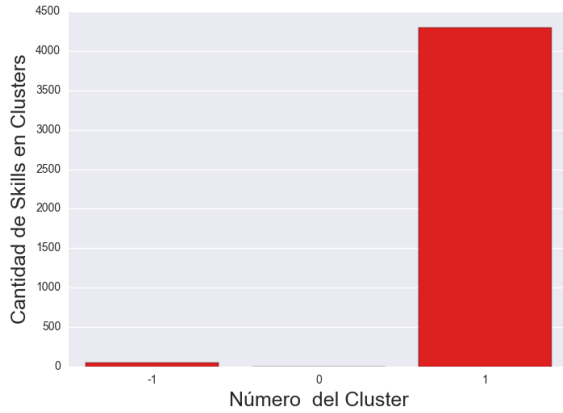
Epsilon	df>0_TF-IDF_D0	df>0_TF-IDF_D430	df>1_TF-IDF_D0	df>1_TF-IDF_D430
1,0	1026	498	1291	531
1,1	548	87	734	157
1,2	167	5	152	5
1,3	34	1	1	1
1,4	8	1	1	1
1,5	1	1	1	1
1,6	1	1	1	1
1,7	1	1	1	1

Analizando las tablas [5.1](#), [5.2](#), [5.3](#) se observa que en los casos de normalización, existen pocas configuraciones que otorguen una buena cantidad de *clusters* a analizar. En el caso de la no normalización existe más diversidad con respecto a la cantidad de *clusters* generados, en la tabla [5.1](#) se observan los casos más representativos de cómo evoluciona esta cantidad en función de *Epsilon* y considerando la mejor cantidad de puntos que diera el Coeficiente de *Silhouette*.

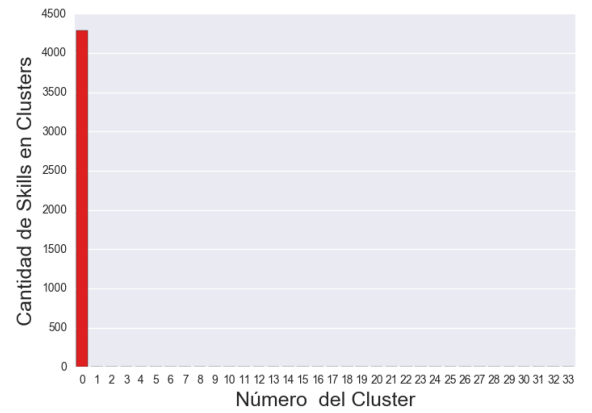
Para la elección de la mejor configuración obtenida, en los casos de no normalización no es complejo elegir, ya que existen pocas opciones por donde realizar la elección. Para la normalización L2 se optó por elegir la configuración *Epsilon* igual a 1,4, cantidad mínima de puntos igual a 1, resultando así 8 *clusters*. Respecto a los resultados de las métricas que se tienen para esta configuración se tiene lo siguiente: 0,00516 Coeficiente de *Silhouette* y -134,33 de Promedio *Coherence*. Para la normalización TF-IDF se eligió la configuración *Epsilon* igual a 1,4, cantidad mínima de puntos igual a 1, resultando así 8 *clusters*. En las métricas se obtiene lo siguiente considerando estos 8 *clusters*: 0,00510 de Coeficiente de *Silhouette* y -134,33 de Promedio *Coherence*.

Del último extracto se puede obtener que ambos poseen casi la misma estructura de *clusters*, lo único que los diferencia es la métrica de *Silhouette* que difiere en una cantidad mínima. Por otro lado el Coeficiente de *Silhouette* indica que los datos no están separados como *clusters* aislados, por lo que no sería una buena opción elegir esta configuración para el análisis. Sin embargo, la métrica de Promedio *Coherence* indica un buen valor comparado con las agrupaciones de otras técnicas, por lo que valdría la pena poder observar cómo se agrupan las *skills* en estos *clusters*.

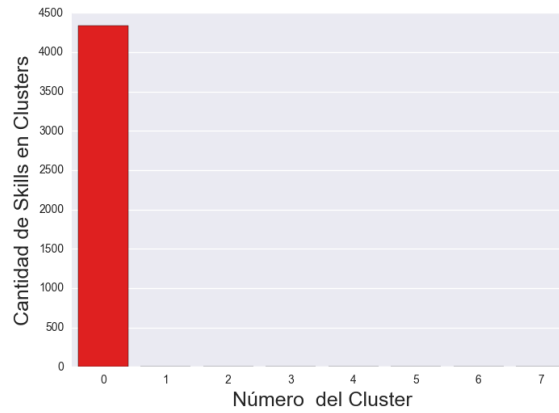
A continuación se presentan distintas distribuciones encontradas en los distintos *clusters* considerando las normalizaciones con las respectivas configuraciones discutidas anteriormente:



(a) Sin normalizar, $df > 0$ y sin reducción de dimensionalidad



(b) Normalización L2, $df > 0$ y sin reducción de dimensionalidad



(c) Normalización TF-IDF, $df > 0$ y sin reducción de dimensionalidad

Figura 5.9: Distintas configuraciones de *DBSCAN* con distintas normalizaciones considerando la agrupación de *skills*.

Se puede observar en las figuras 5.9 que las distintas distribuciones siempre poseen solo 1 *cluster* que tiene la mayoría de las *skills*, por lo que no se podrían obtener agrupaciones de *skills* analizables para el estudio, resultando así, que *DBSCAN* no es una técnica apropiada para completar este objetivo.

5.1.3. Ward

Otra técnica que se utilizó para la agrupación de *skills* fue el algoritmo jerárquico *Ward*.

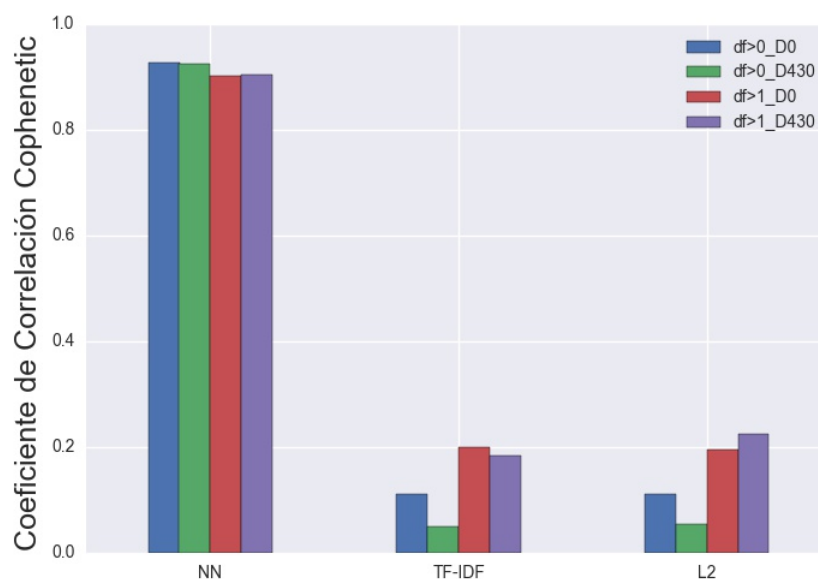


Figura 5.10: Coeficiente de Correlación *Cophenetic* con distintas normalizaciones de datos.

En el gráfico 5.10 se aprecia que la no normalización tiene un mejor rendimiento que las normalizaciones L2 y TF-IDF considerando la métrica del Coeficiente de Correlación *Cophenetic*. Por lo anterior es que se presentará un dendrograma a 9 niveles en la figura 5.11 realizado con los datos sin normalizar, con *skills* que hayan sido declaradas por lo menos 2 veces y sin reducir la dimensionalidad de los datos.

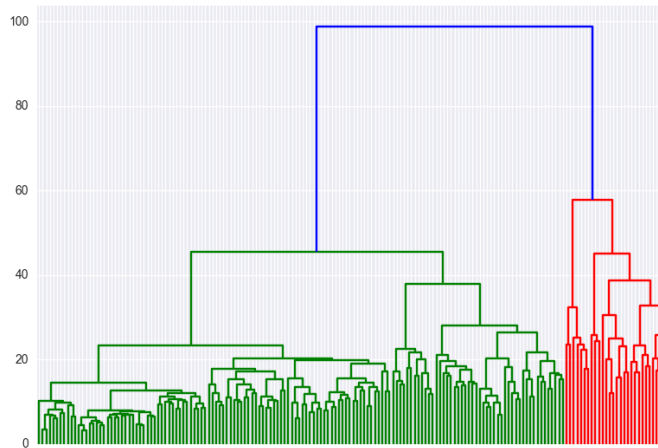
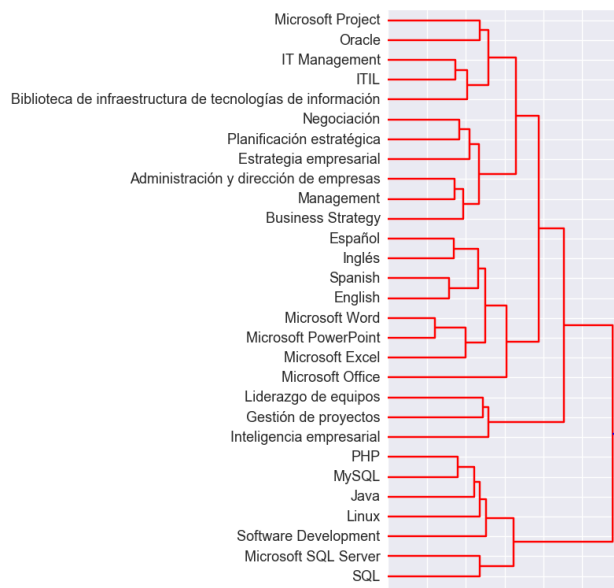


Figura 5.11: Dendrograma de 9 niveles.

Observando el gráfico 5.11 se evidencia una tendencia de agrupar muchas *skills* en un solo grupo (grupo color verde), tendencia que se evidencia también en las técnicas *K-Means* y *DBSCAN*.

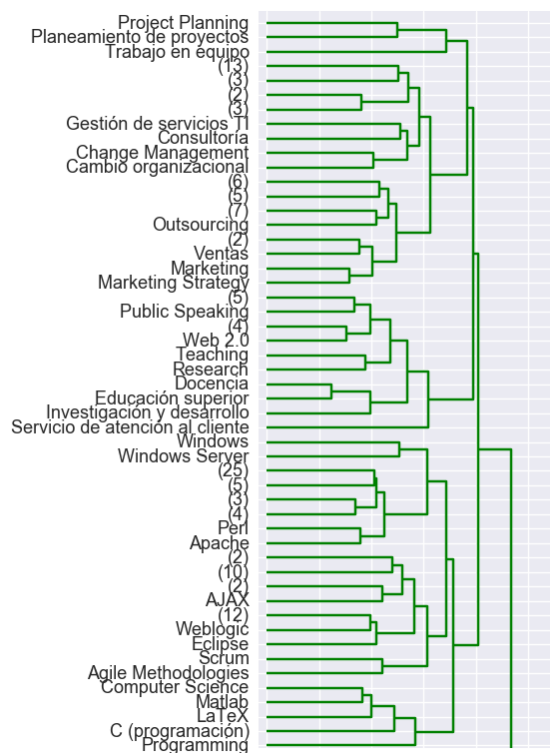
Luego, en las figuras 5.12, se presentan los detalles de los grupos encontrados en el dendrograma a 9 niveles. Considerando las distintas agrupaciones, se observa que las *skills* que se encuentran en éstas tienen bastante sentido entre sí, por ejemplo, en la agrupación roja se presenta un *cluster* o agrupación que posee *IT Management*, *ITIL* y Biblioteca de infraestructura de tecnologías de información. Si se analiza estas 3 *skills* rápidamente se puede apreciar que están relacionadas entre sí. Otro ejemplo que evidencia la coherencia de los grupos también se presenta en el grupo rojo, existe un *cluster* que posee *Microsoft Word*, *Microsoft Excel* y *Microsoft Excel*, luego este *cluster* se junta con su generalización que es *Microsoft Office*.



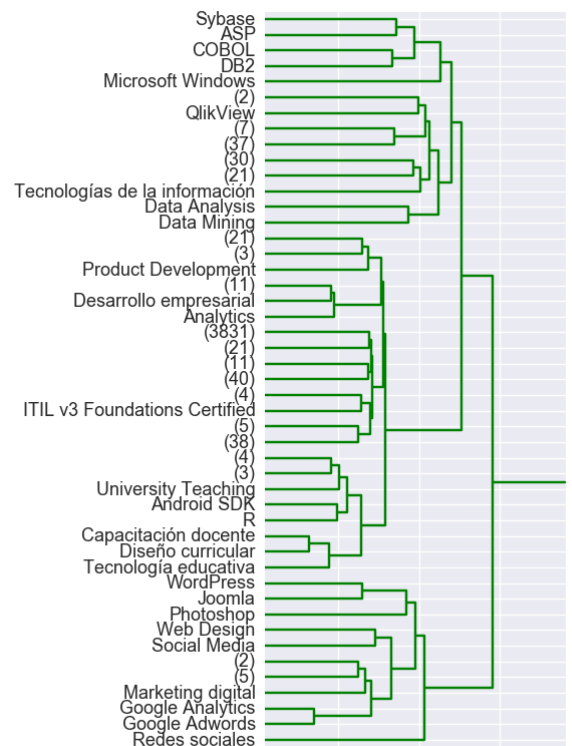
(a) Grupo rojo



(b) Distintos agrupaciones del grupo verde



(c) Distintos agrupaciones del grupo verde



(d) Distintos agrupaciones del grupo verde

Figura 5.12: Grupos obtenidos del dendrograma de 9 niveles.

Otras métricas que se aplicaron con el fin de tener una métrica en común con los otros métodos, son las siguientes: Coeficiente de *Silhouette* y Promedio *Coherence*.

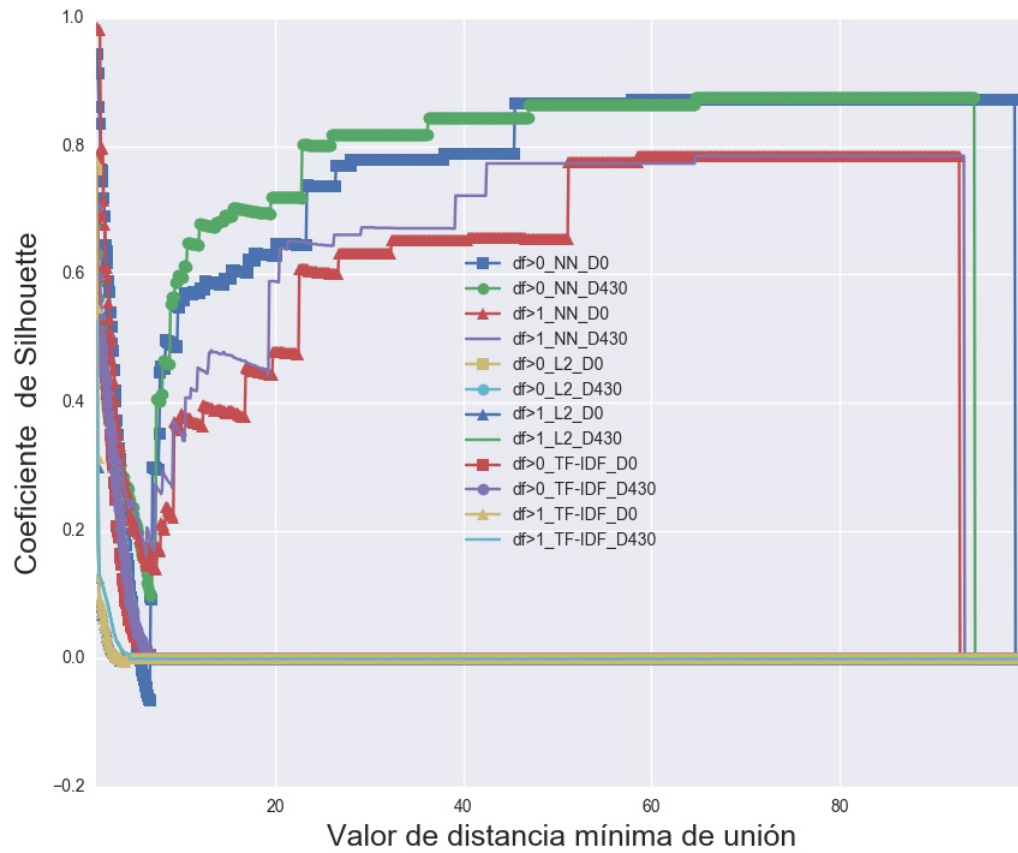


Figura 5.13: Ward: Coeficiente de *Silhouette* con distintas normalizaciones de datos.

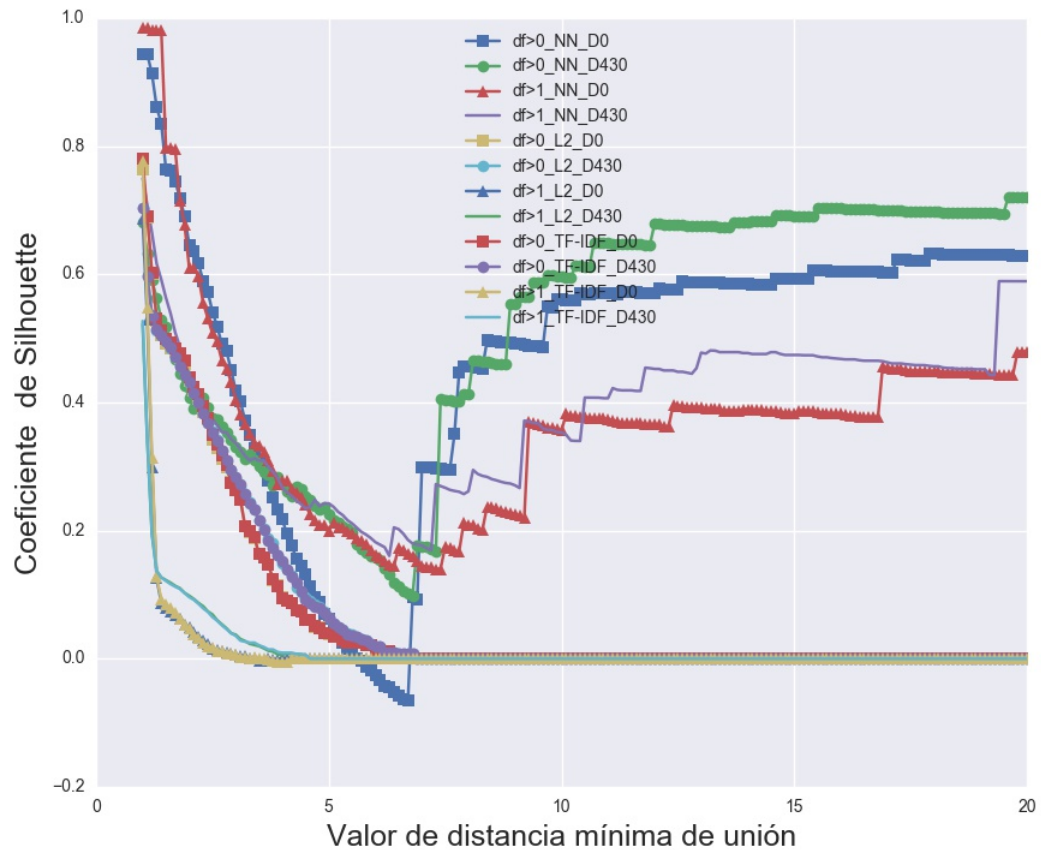


Figura 5.14: Ward: Coeficiente de *Silhouette* con distintas normalizaciones con valor de distancia mínima de unión disminuida.

Respecto al Coeficiente de *Silhouette* se observa la misma situación que en las distintas técnicas, las curvas no normalizadas tienen un comportamiento muy diferente a las curvas con normalización L2 y TF-IDF. En este caso, las curvas no normalizadas disminuyen a medida que la distancia mínima de unión comienza a crecer, es decir, cuando los *clusters* comienzan a unirse entre sí. Esta disminución se frena entre valor de distancia mínima de unión 5 y 7, luego de esto, las curvas comienzan a ascender. Este fenómeno se puede explicar por la forma de ejecución del algoritmo Ward, recordar que éste comienza contando cada punto como un *cluster* por separado, por lo tanto el Coeficiente de *Silhouette* será cercano a 1. Luego el ascenso se explica por la formación de *clusters* más grandes que tienen relación

entre ellos.

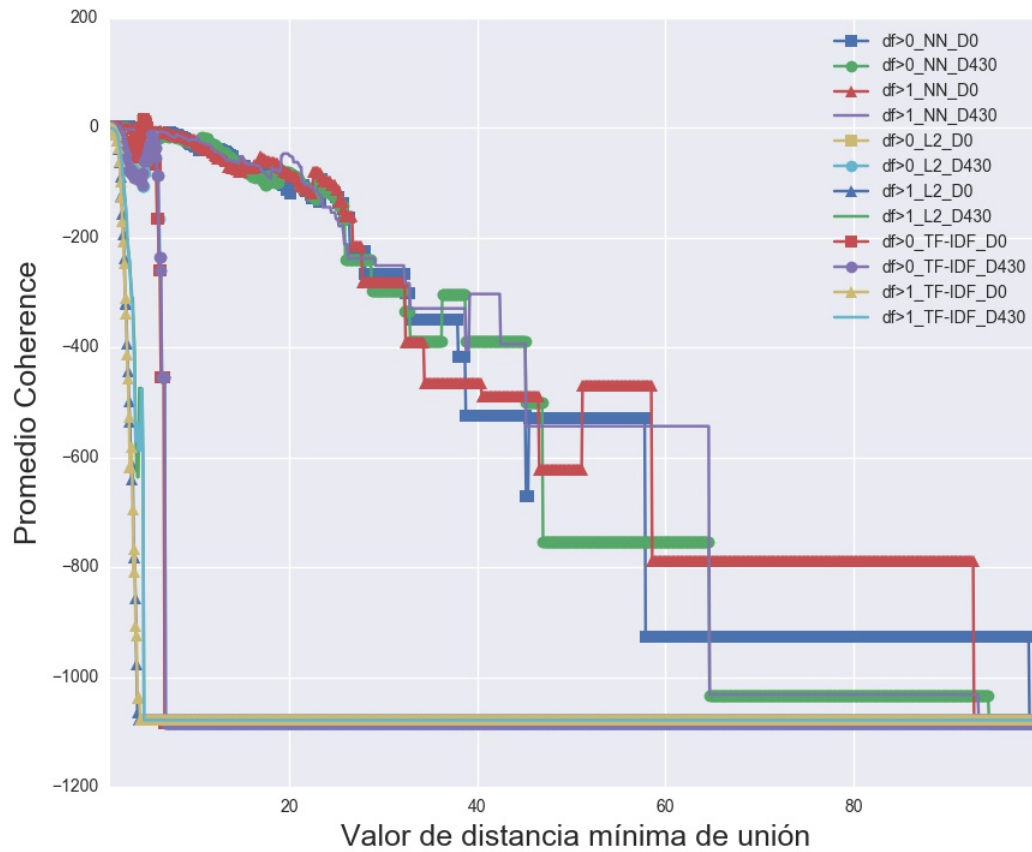


Figura 5.15: Ward: Promedio *Coherence* con distintas normalizaciones de datos.

Luego con referente a la métrica del Promedio *Coherence*, se observa en el gráfico 5.15 un comportamiento de disminución para todas las curvas presentadas. En la figura 5.16 se nota claramente que las curvas normalizadas presentan un decaimiento más abrupto en comparación a las curvas no normalizadas.

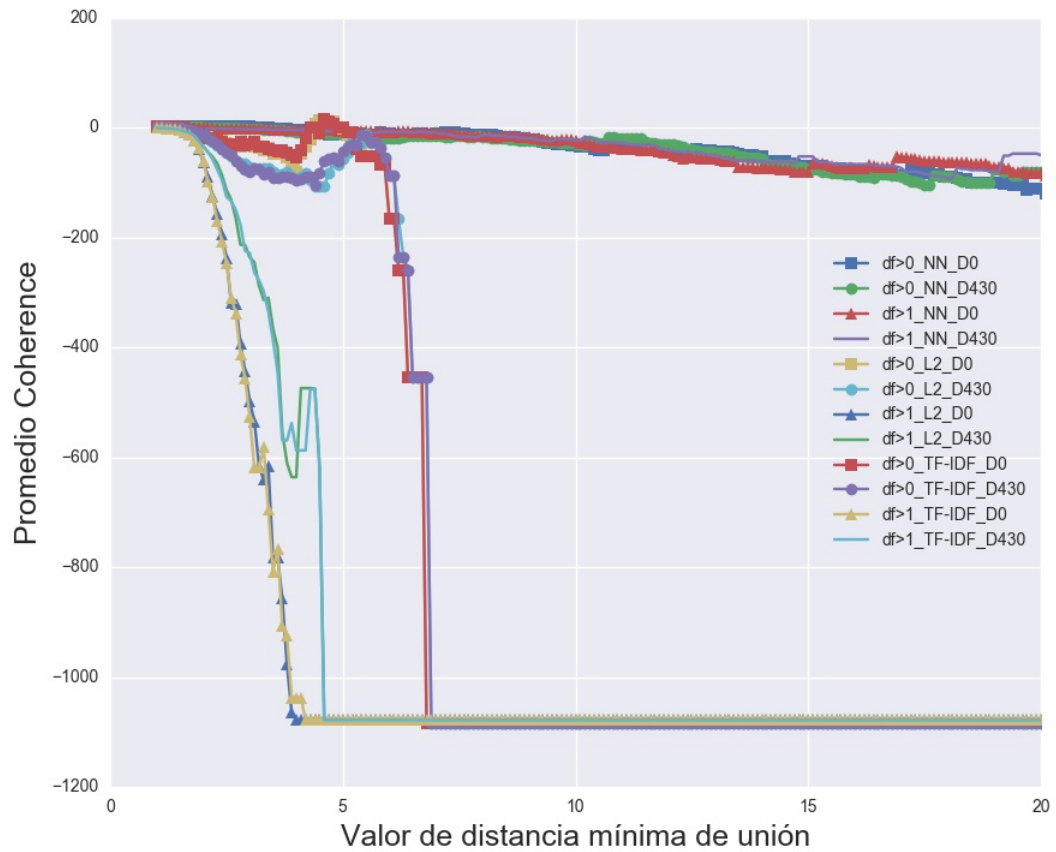


Figura 5.16: Ward: Promedio *Coherence* con distintas normalizaciones de datos con valor de distancia mínima de unión disminuida.

Por otro lado, al igual que en *DBSCAN*, se tiene que existe una variable que controla la cantidad de *clusters* que encontrará el algoritmo, esta es la distancia mínima de unión. Por esto mismo se presentará las cantidades más representativas de *clusters* encontrados según su normalización.

Cuadro 5.4: Número de *clusters* variando el valor de distancia de unión mínima, considerando datos sin normalizar.

t	df>0_NN_D0	df>0_NN_D430	df>1_NN_D0	df>1_NN_D430
1	2463	1616	1644	1332
1,1	2463	1494	1644	1264
1,2	2381	1391	1641	1195
1,3	2224	1311	1638	1130
1,4	2136	1230	1638	1071
1,5	1879	1163	1431	1017
22,2	25	26	25	24
22,3	24	24	24	23
22,4	24	23	24	23
31,7	9	9	9	9
31,8	9	9	9	9

Cuadro 5.5: Número de *clusters* variando el valor de distancia de unión mínima, considerando datos con normalización L2.

t	df>0_L2_D0	df>0_L2_D430	df>1_L2_D0	df>1_L2_D430
1	1979	1680	1328	1077
1,1	1767	1378	1169	846
1,2	1507	1171	889	639
1,3	1255	1054	631	517
1,4	1117	967	503	440
1,5	1003	868	404	366
5,1	11	25	1	1
5,2	11	22	1	1
5,3	10	20	1	1
5,4	8	18	1	1

Cuadro 5.6: Número de *clusters* variando el valor de distancia de unión mínima, considerando datos con normalización TF-IDF.

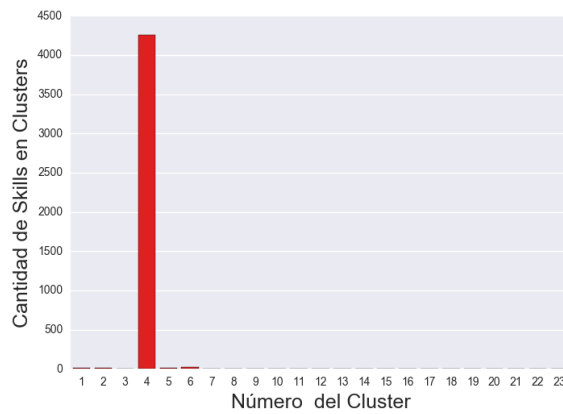
t	df>0_TF-IDF_D0	df>0_TF-IDF_D430	df>1_TF-IDF_D0	df>1_TF-IDF_D430
1	2010	1684	1428	1083
1,1	1756	1374	1185	848
1,2	1493	1174	916	644
1,3	1257	1059	648	526
3,7	92	133	4	11
3,8	72	121	3	11
3,9	63	110	2	7
4	47	101	2	6
4,1	44	89	2	6
4,2	41	80	1	6
5,7	7	12	1	1
5,8	6	11	1	1
5,9	6	9	1	1
6	4	7	1	1

Observando las tablas 5.4 5.5 5.6 se procede a realizar una búsqueda de la configuración que se acerque a 15 o 20 *clusters*, ya que esta cantidad facilita realizar un análisis manual de los *clusters* generados. A continuación se presentan las mejores configuraciones elegidas:

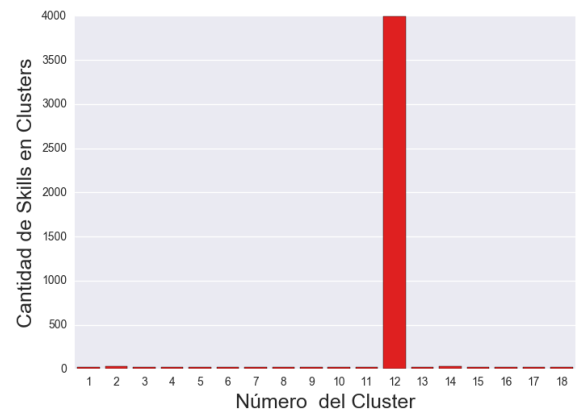
- No normalizado con todas las *skills* y reducción de dimensionalidad, se escogió una distancia mínima de unión de 22,4 cuyo valor para el Coeficiente de *Silhouette* es 0,7191 y el Promedio *Coherence* es -123,691. La cantidad de *clusters* generados es 24.
- Para la normalización L2 con todas las *skills* y reducción de dimensionalidad, se escogió una distancia mínima de unión de 5,4 cuyo valor para el Coeficiente de *Silhouette* es 0,0406 y el Promedio *Coherence* es -27,015. La cantidad de *clusters* generados es 18.

- Para la normalización TF-IDF con todas las *skills* y reducción de dimensionalidad, se escogió una distancia mínima de unión de 5,7 cuyo valor para el Coeficiente de *Silhouette* es 0,0310 y el Promedio *Coherence* es -25,684. La cantidad de *clusters* generados es 12.

A continuación se presentan el cómo se distribuyen las *skills* considerando las configuraciones anteriores.



(a) Sin normalizar, $df > 0$ y con reducción de dimensionalidad



(b) Normalización L2, $df > 0$ y con reducción de dimensionalidad



(c) Normalización TF-IDF, $df > 0$ y con reducción de dimensionalidad.

Figura 5.17: Distintas configuraciones de *Ward* con distintas normalizaciones.

De las figuras 5.17 se evidencia una distribución de *skills* similar a las técnicas *K-Means* y *DBSCAN*. Como se ha dicho anteriormente estas distribuciones no aportan información al estudio, por lo que analizar el algoritmo *Ward* observando por separado los niveles de jerarquía no presenta resultados analizables que puedan ser sustentados por métricas.

5.1.4. Topic Models

En esta sección se analizará la búsqueda de grupos de *skills* a través de distintos algoritmos: LSA, NMF y LDA.

5.1.4.1. LSA

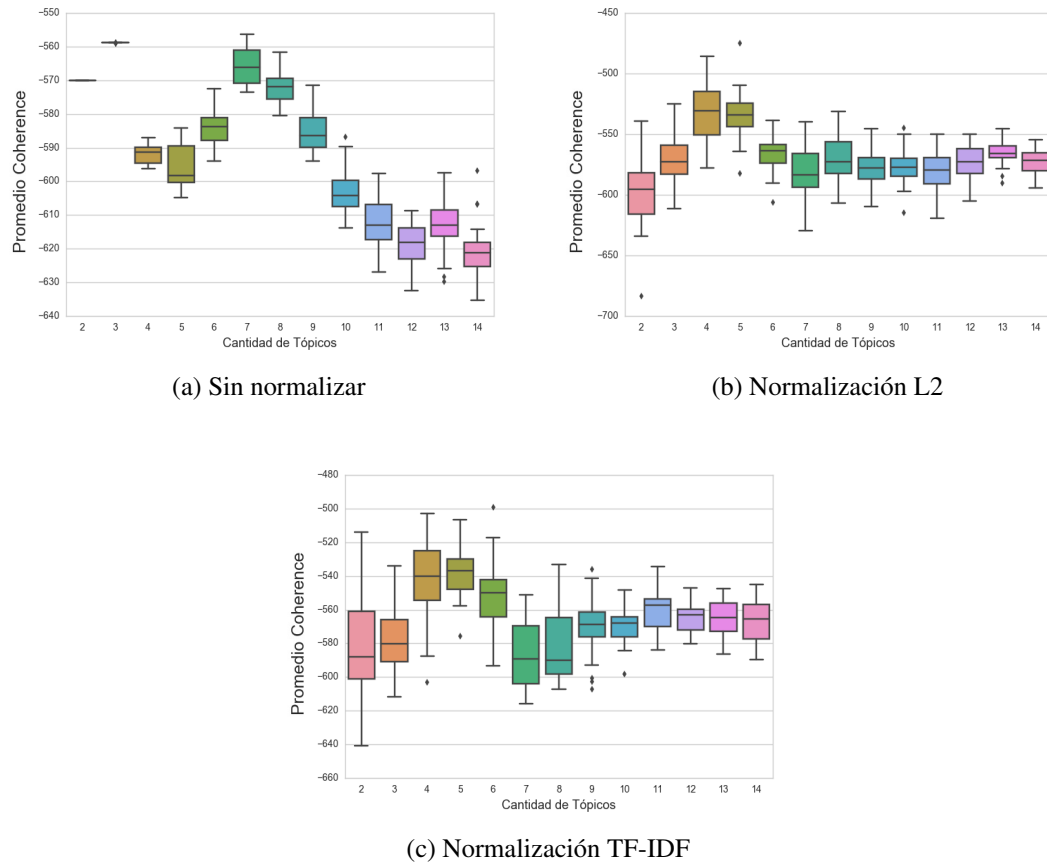


Figura 5.18: LSA: Promedio *Coherence* con distintas normalizaciones de datos.

En los gráficos 5.18 se puede evidenciar que la técnica LSA presenta un comportamiento similar para las 3 normalizaciones, en referente al mejor valor correspondiente al mayor Promedio *Coherence*. Se evidencia que para la no normalización, la mejor cantidad de tópicos se presenta en 7 tópicos, esto se puede asegurar ya que tiene menos dispersión y la mediana se presenta en un valor superior que las demás. Para L2 se tiene que el mejor valor de tópicos es 5, esto se debe a que en los experimentos se tuvo mucho menos dispersión que con 4 tópicos. Finalmente para la normalización TF-IDF se tiene que el mejor valor se presenta en 5, por las mismas razones que L2.

Por otro lado se puede observar que la tendencia de estas normalizaciones son distintas, esto se puede evidenciar en donde hacia convergen los *boxplots*. En la no normalización se presenta una tendencia a un valor muy inferior a las otras 2 normalizaciones, en cambio, L2 y TF-IDF convergen a un valor muy similar, con la diferencia que la normalización L2 posee menor varianza en los *boxplots*.

5.1.4.2. NMF

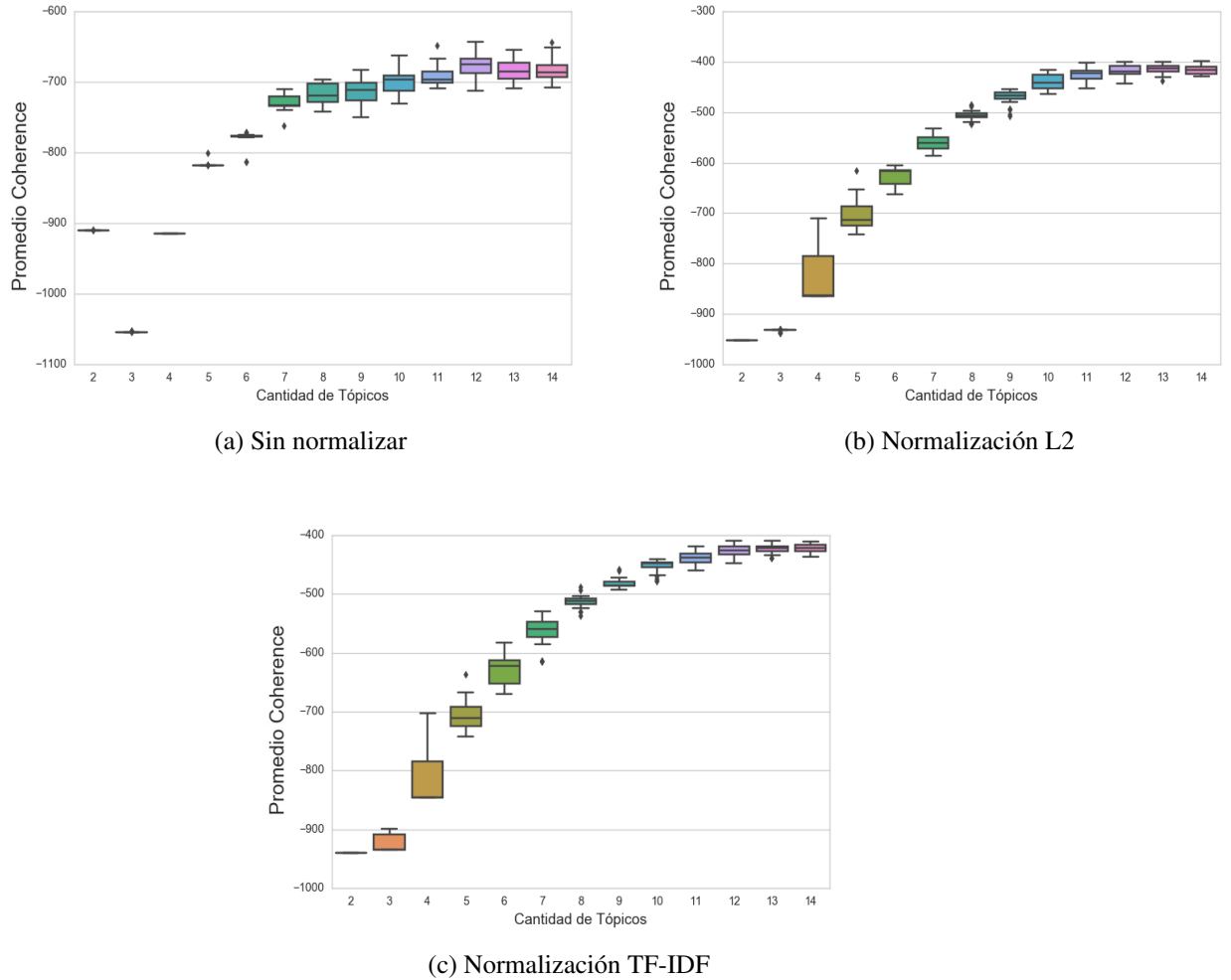
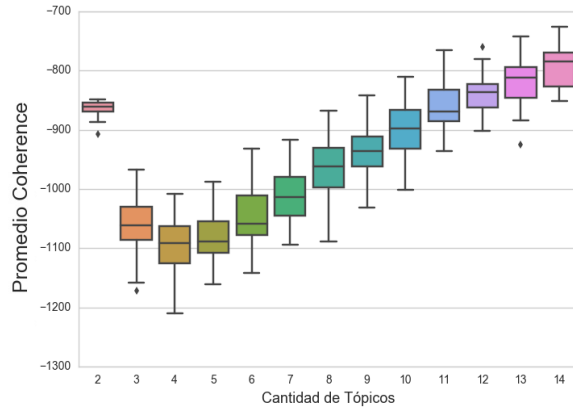


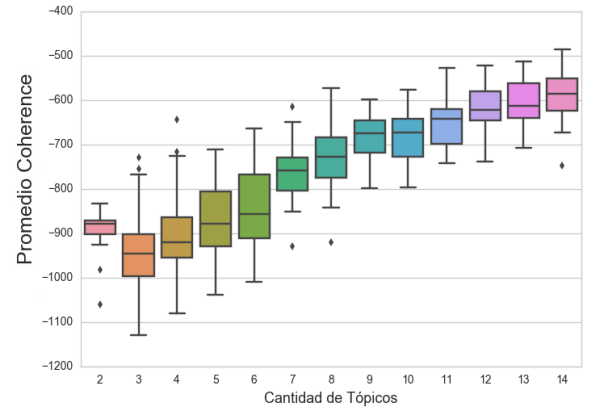
Figura 5.19: NMF: Promedio *Coherence* con distintas normalizaciones de datos.

Se aprecia en los gráficos 5.19 que a diferencia de la técnica LSA, en los *boxplots* el máximo se presenta en el último tópico en las 3 normalizaciones. También otra diferencia significativa es que NMF no presenta tanta dispersión en los datos. Para la no normalización se eligió 6 tópicos, debido a que se sopesó tanto el valor del Promedio *Coherence* como la dispersión de los datos. En el caso de L2 y TF-IDF se optó por elegir 10 tópicos para representar los grupos, considerando la mejor menor cantidad de tópicos en relación a los demás *boxplots*.

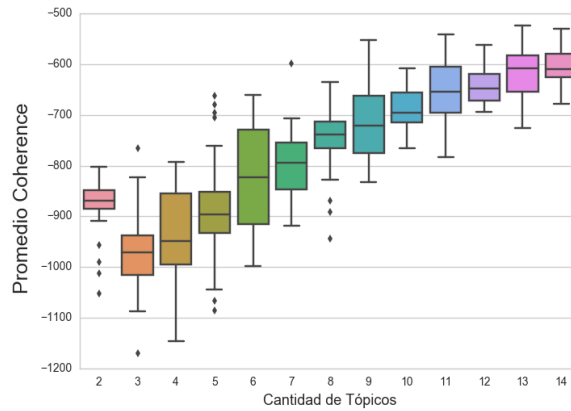
5.1.4.3. LDA



(a) Sin Normalizar



(b) Normalización L2



(c) Normalización TF-IDF

Figura 5.20: LDA: Promedio *Coherence* con distintas normalizaciones de datos.

Los resultados obtenidos con LDA son presentados en los gráficos 5.20. En éstos se puede evidenciar que al igual que NMF, los datos convergen a un valor de Promedio *Coherence*. En esta ocasión, por la dispersión de los datos y el valor de las medianas, se eligió para los 3 casos (no normalizar, L2 y TF-IDF) seleccionar 12 Tópicos para analizar los resultados.

5.1.4.4. LSA + *K-Means*

En esta ocasión, a la mejor configuración de normalizaciones en LSA, se le aplicará el algoritmo *K-Means* para observar cómo se agruparán los datos resultantes.

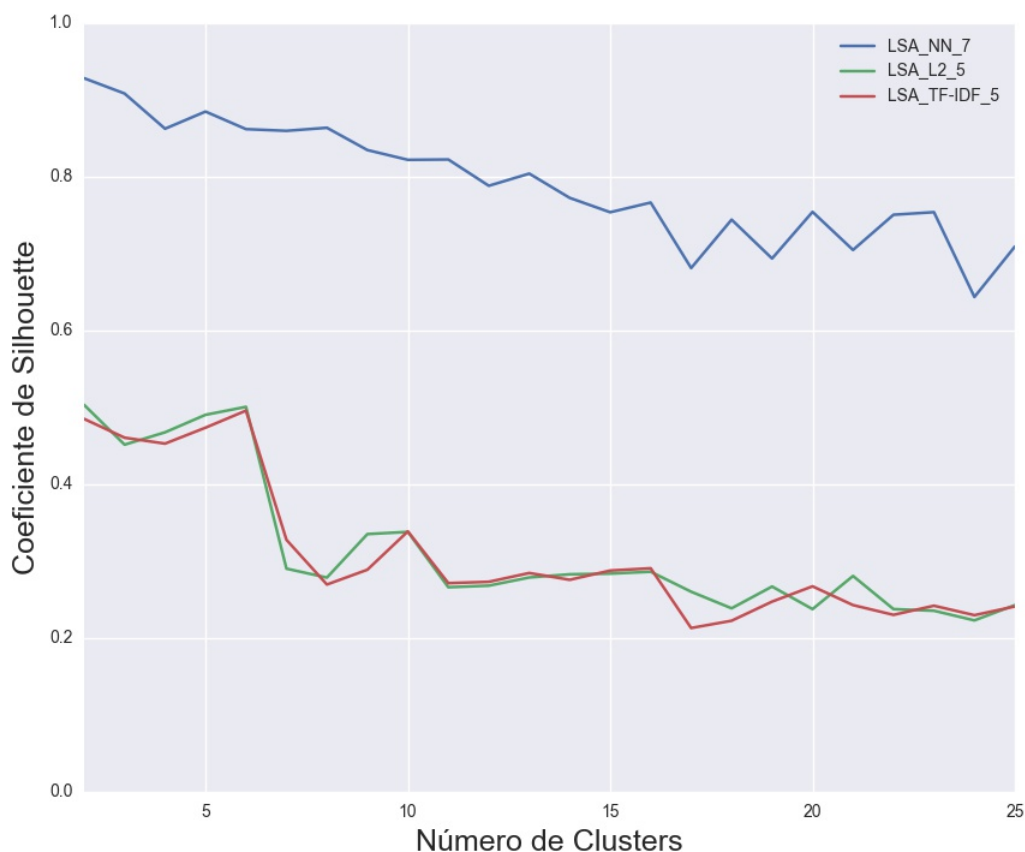


Figura 5.21: LSA + *K-Means*: Coeficiente de *Silhouette* con distintas normalizaciones de datos.

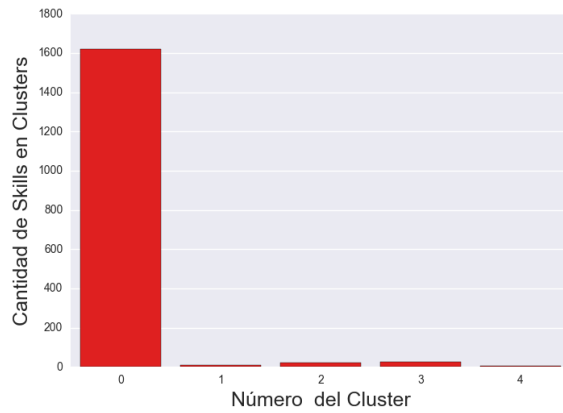
En el gráfico 5.21 se observa que entre más *clusters* existan, menor será el Coeficiente de *Silhouette*. Se aprecia que en el caso de la no normalización, los Coeficientes de *Silhouette* son altos, fluctuando entre 0,62 hasta 0,91. El número de *clusters* más convenientes en este caso es 5, ya que es necesario que sea un número razonable de grupos y a su vez tenga un buen Coeficiente de *Silhouette* que en este caso es 0,884. Para el caso de L2 y TF-IDF, se

tienen que sus curvas son muy similares, donde los valores del Coeficiente de *Silhouette* rondan entre 0,20 y 0,50. Para ambas normalizaciones el mejor número de *clusters* será 6, donde el Coeficiente de *Silhouette* es 0,500 para L2 y 0,495 para TF-IDF.



Figura 5.22: LSA + *K-Means*: Promedio *Coherence* con distintas normalizaciones de datos considerando.

Ahora, observando el gráfico 5.22 se evidencia que la curva de Promedio *Coherence* es creciente en las 3 normalizaciones. En este caso, el máximo valor de Promedio *Coherence* de lo no normalizado se encuentra en el número de *clusters* 16 con -153,613, en cambio, para L2 y TF-IDF se encuentra al final del gráfico, en el número de *clusters* 25 con valores de -465,579 y -449,935 respectivamente.



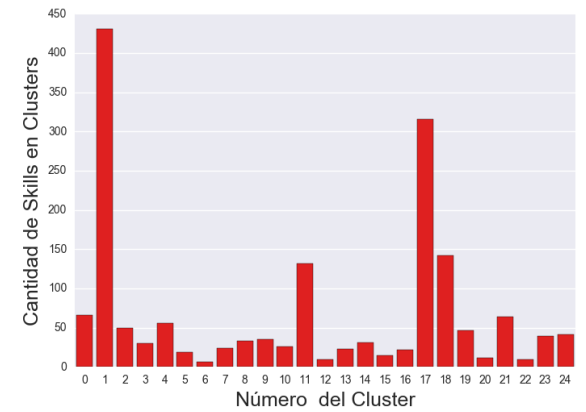
(a) Sin normalizar con 5 clusters



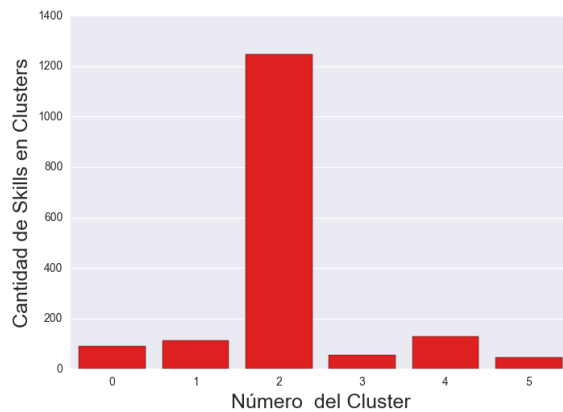
(b) Sin normalizar con 16 clusters



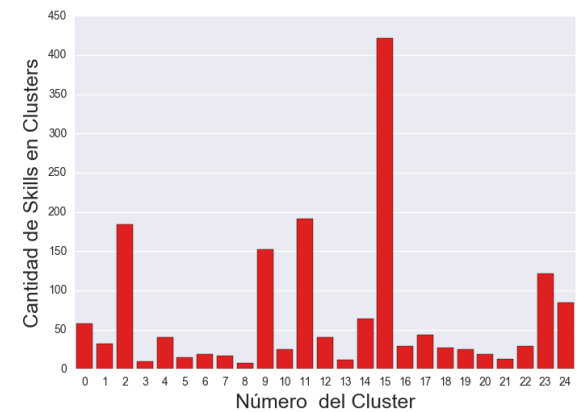
(c) Normalización L2 con 6 clusters



(d) Normalización L2 con 25 clusters



(e) Normalización TF-IDF con 6 clusters



(f) Normalización TF-IDF con 25 clusters

Figura 5.23: LSA: Distribución de las *skills* en los distintos *clusters*.

En las figuras 5.23 se observan distintas distribuciones de las configuraciones escogidas en base al Coeficiente de *Silhouette* y el Promedio *Coherence*. Se observa que en la no normalización, la distribución de las *skills* en los *clusters* se centra más en un solo grupo, siendo así, algo no favorable para el futuro análisis. Por otro lado, en las normalizaciones L2 y TF-IDF, se observa que cuando se presentan pocos *clusters* se tiene una “mejor” distribución de las *skills* en los *clusters*. Cuando se posee una mayor cantidad de *clusters* se observa que las *skills* se distribuyen de mejor manera a comparación de la no normalización con mayor cantidad de *clusters*.

En este caso, para realizar los *Word Clouds* se utilizarán las normalizaciones L2 y TF-IDF con 6 grupos formados gracias a *K-Means*. Si bien con 25 *clusters* se presenta una distribución más balanceada de *skills*, se optó por elegir 6 grupos debido a que 25 dificulta la evaluación manual de los grupos.

5.1.4.5. LDA + *K-Means*

En esta sección, a la mejor configuración de normalizaciones en LDA, se le aplicará el algoritmo *K-Means* para observar cómo se agruparán los datos resultantes.



Figura 5.24: LDA + *K-Means*: Coeficiente de *Silhouette* con distintas normalizaciones de datos.

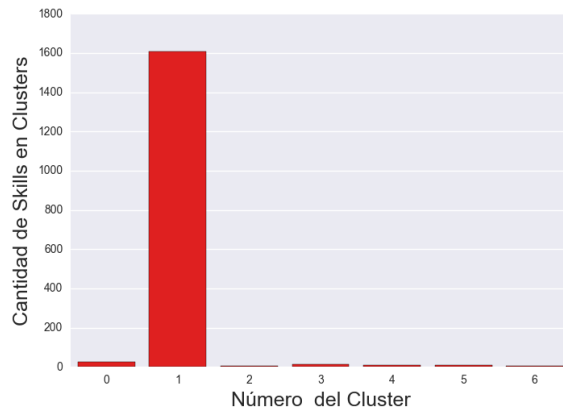
Se aprecia en el gráfico 5.24, que la curva de no normalizada presenta saltos con tendencia a la disminución del valor del Coeficiente de *Silhouette*, presentando así su máximo valor en el número de *clusters* 2, no está demás mencionar que los valores de esta curva oscilan entre 0,63 y 0,90. Si bien el valor más alto de la curva no normalizada se encuentra en el número de *clusters* 2, en esta ocasión se optó por elegir la cantidad de *clusters* 7, ya que si bien posee menos valor en el Coeficiente de *Silhouette*, esta agrupación podría entregar más información sobre las distintas agrupaciones de *skills*. El valor del Coeficiente de *Silhouette* para la curva no normalizada con 7 *clusters* es 0,878. Para las normalizaciones L2 y TF-IDF, se observa que el máximo valor del Coeficiente de *Silhouette* se encuentra en 21 y 16

respectivamente. Los valores del Coeficiente de *Silhouette* para 21 *clusters* con L2 es 0,486, mientras que para la normalización TF-IDF con 16 *clusters* es 0,507.

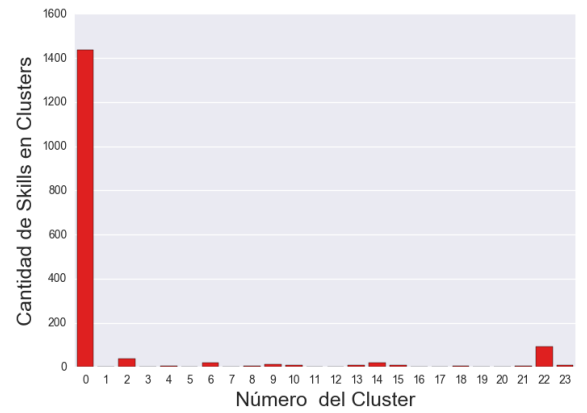


Figura 5.25: LDA + *K-Means*: Promedio *Coherence* con distintas normalizaciones de datos.

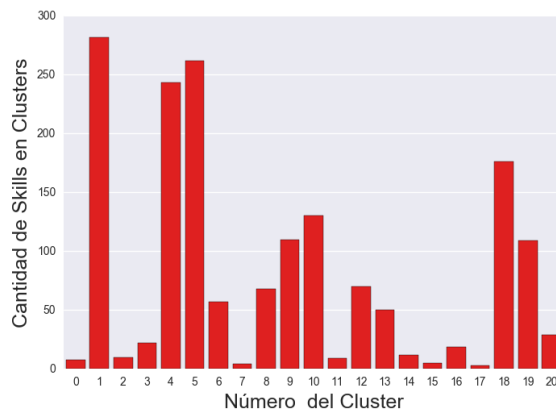
Por otro lado, en el gráfico 5.25, se puede observar que las 3 curvas presentadas presentan un crecimiento a medida que crece la cantidad de *clusters*. En los 3 casos se tiene que el mejor valor de número de *clusters* es el 24, donde los Promedios *Coherence* son -162,447 para la no normalización, -401,824 para L2 y -442,395 para TF-IDF.



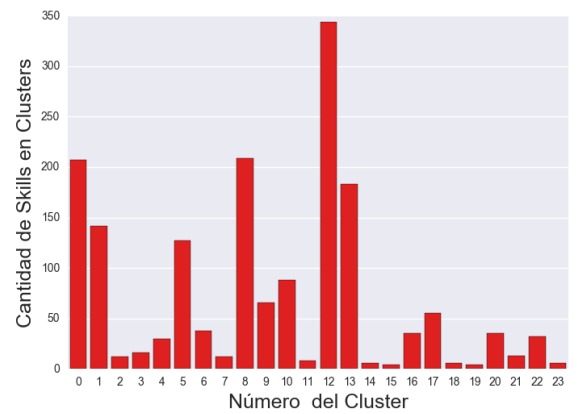
(a) Sin Normalizar con 7 clusters



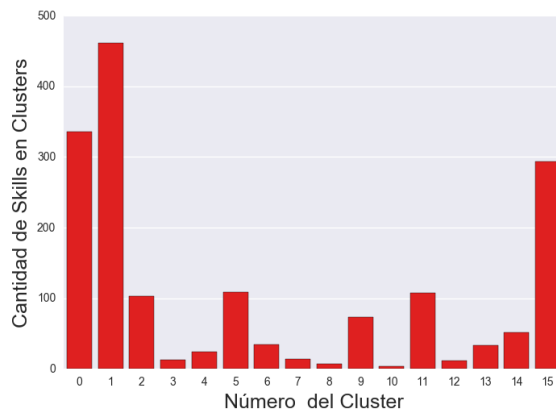
(b) Sin Normalizar con 24 clusters



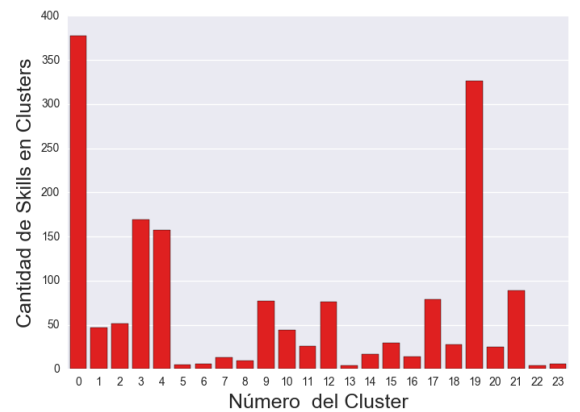
(c) Normalización L2 con 21 clusters



(d) Normalización L2 con 24 clusters



(e) Normalización TF-IDF con 16 clusters



(f) Normalización TF-IDF con 24 clusters

Figura 5.26: LDA: Distribución de las *skills* en los distintos *clusters*.

Finalmente en los gráficos 5.26 se observan las distribuciones de las distintas configuraciones elegidas para las normalizaciones. Se evidencia claramente que las normalizaciones L2 y TF-IDF poseen mejor comportamiento a la no normalización, con respecto a la distribución de las *skills* en los *clusters* generados. Por esta misma razón y considerando las métricas obtenidas, se tiene que los grupos que serán escogidos para realizar los *Word Clouds* serán las normalizaciones L2 y TF-IDF con el número de *clusters* 21 y 16 respectivamente.

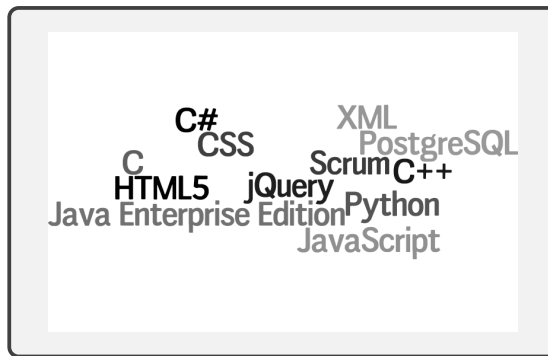
5.1.5. Agrupación de *Skills*: *Word Clouds*

Para la agrupación de *skills* no está demás recordar cómo se generaron los distintos *Word Clouds* que se verán a continuación.

Considerando que en las técnicas de *clustering* no se presentan pesos de las *skills* dentro de los *clusters*, se optó por asignar como peso a una *skill* la frecuencia de declaración que tiene ésta dentro de todo el *dataset*. Con esto el tamaño de la *skill* dentro del *Word Cloud* será la frecuencia descrita anteriormente. Respecto a los *Topic Models*, se eligió como tamaño de las *skills* el peso relativo que tiene ésta dentro del grupo asignado. Para ambos casos se integraron solo las *top 30 skills* y luego se eliminó las que tuvieran frecuencia menor a 1.

Por otro lado, como hay que realizar un análisis manual de cada grupo generado, se considerarán que 2 grupos son iguales, siempre y cuando las primeras 10 *skills* del *top* sean las mismas sin importar el orden.

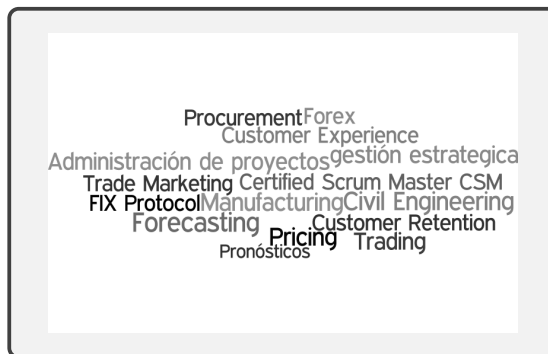
A continuación se presentan los grupos más coherentes encontrados con las distintas técnicas.



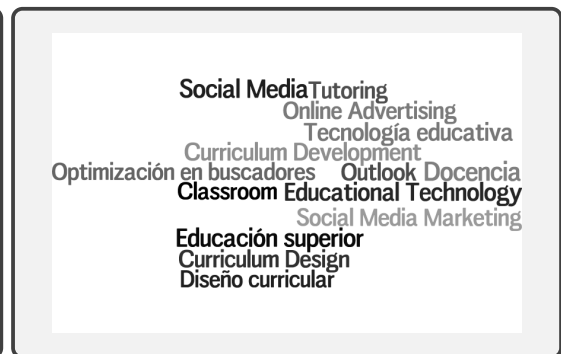
(a) Programación



(b) Gestión de recursos humanos



(c) Gestión de clientes



(d) Docencia



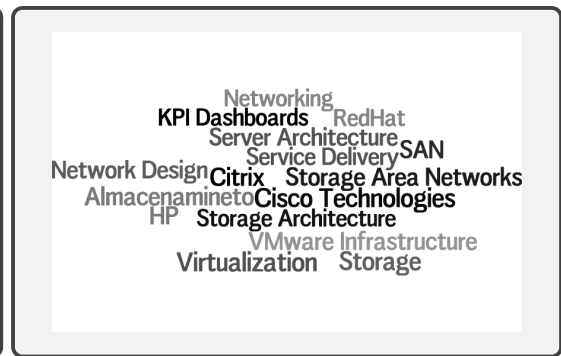
(e) Tecnología SAP



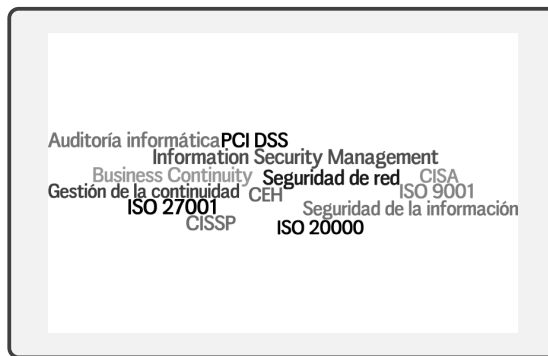
(f) Banca



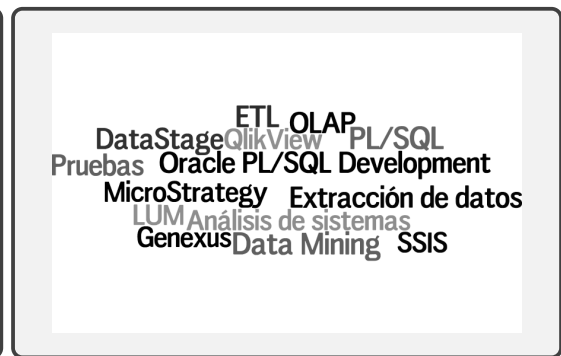
(g) Tecnologías de mercado



(h) Infraestructura computacional



(i) Seguridad informática



(j) Inteligencia de negocios



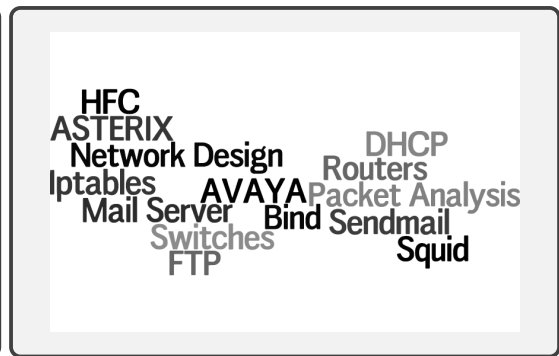
(k) Gestión de proyectos



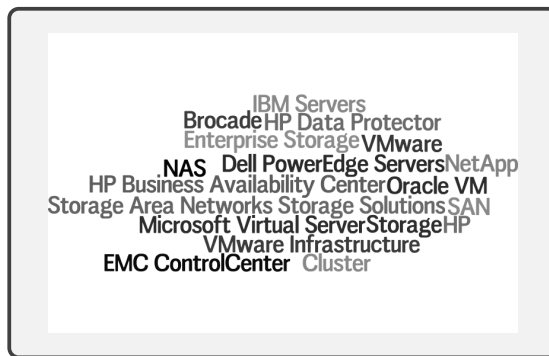
(l) Marketing



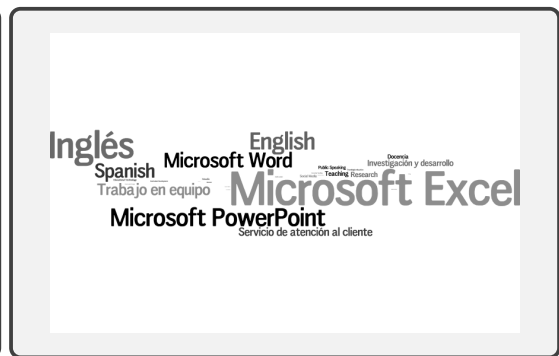
(m) Almacenamiento



(n) Redes



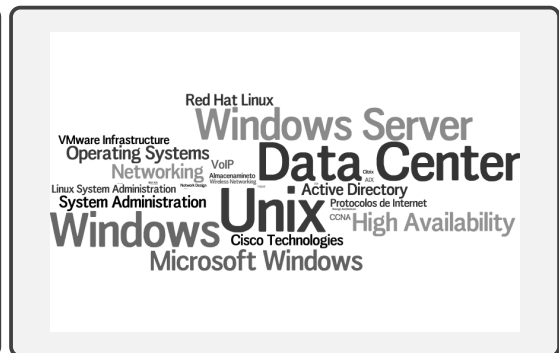
(ñ) Infraestructura servidores



(o) Ofimática



(p) Desarrollo WEB



(q) Configuración servidores

Figura 5.24: Word Clouds de agrupación de *skills*

Los distintos grupos encontrados en 5.24 no se pueden clasificar al 100 % por la etiqueta designada, ya que en algunas ocasiones posee *skills* que no necesariamente pertenecen realmente a esa disciplina, sin embargo se agregan de igual manera ya que los algoritmos

encontrados la consideran importante dentro de la agrupación. Esto se puede evidenciar por ejemplo en el grupo Gestión de clientes donde se presenta una *skill* que se nombra como Administración de Proyectos, esto no necesariamente va dentro de esta agrupación.

Por otro lado se presenta la tabla 5.7 que indica qué algoritmos encontraron los grupos mencionados en la figura 5.24:

Cuadro 5.7: Detalle de técnicas y grupos de *skills* encontrados.

Grupo / Técnica	LSA	NMF	LDA	LSA+ <i>K-Means</i>	LDA + <i>K-Means</i>
Programación		√	√		
Gestión de recursos humanos			√		
Gestión de clientes	√		√		
Docencia		√	√		
Tecnología SAP		√	√		
Banca			√		
Tecnologías de mercado			√		
Infraestructura computacional			√		
Seguridad informática		√	√		
Inteligencia de negocios	√	√	√		
Gestión de proyectos	√	√	√		
Marketing		√	√		
Almacenamiento		√			
Redes	√	√			
Infraestructura servidores		√			
Ofimática				√	√
Desarrollo WEB				√	√
Configuración servidores				√	√

En la tabla 5.7 solo se presentan las técnicas que encontraron agrupaciones coherentes.

Se observa que las técnicas que más aportaron a este objetivo fueron LDA y NMF. LSA solo encontró 4 grupos que también fueron encontrados con los algoritmos NMF y LDA. Las

técnicas de LSA+*K-Means* y LDA+*K-Means* encontraron los mismos grupos entre sí y a la vez grupos que no fueron encontrados con las técnicas LSA, NMF y LDA.

Por último se presenta la figura 5.25 que presenta la distribución de los usuarios según universidad en los distintos grupos encontrados.

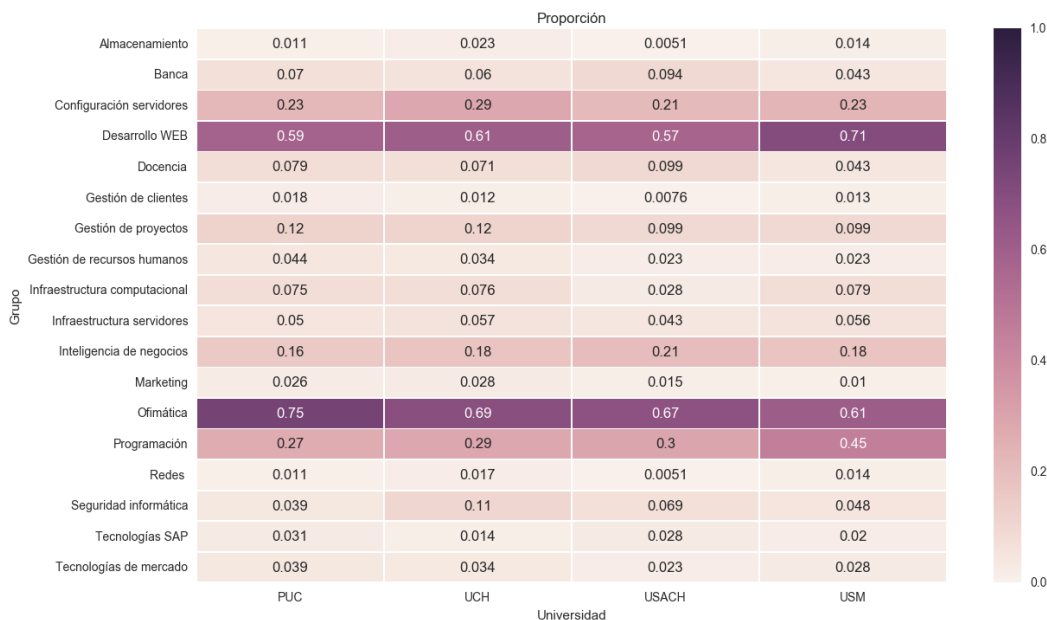


Figura 5.25: Distribución de usuarios en los grupos de *skills* según universidad.

Lo primero que se puede observar es que existen 2 grupos que poseen muchos más usuarios que declaran este tipo de *skills*, sin importar de qué universidad sean. Estos grupos son Desarrollo WEB y Ofimática, que más del 50 % de usuarios declara a lo menos 1 *skill* de este tipo de grupo. Luego existen 4 grupos cuyas proporciones superan el 10 %, estas son Configuración de servidores, Gestión de proyectos, Inteligencia de negocios y Programación.

Con respecto a las distribuciones de las universidades se observa que la USM presenta más habilidades de Programación y Desarrollo WEB que las otras 3 universidades. Ahora bien, se presenta también que los usuarios de la PUC declaran muchas más *skills* referentes a Ofimática en comparación con la USM. Por otro lado se observa que la UCH posee profesionales que declaran más *skills* referentes a Configuración servidores, a comparación con las demás universidades. Finalmente, observando el grupo de Inteligencia de negocios, se aprecia que

la USACH posee la mayor proporción a comparación con las otras universidades.

5.1.6. Grupos Aleatorios

Para verificar el comportamiento de las distintas métricas utilizadas, se generaron agrupaciones aleatorias de *skills* considerando distribución uniforme entre los *clusters* generados. Se generaron distintos grupos considerando todas las *skills* y eliminando las *skills* cuya frecuencia fueran mayores a 1.

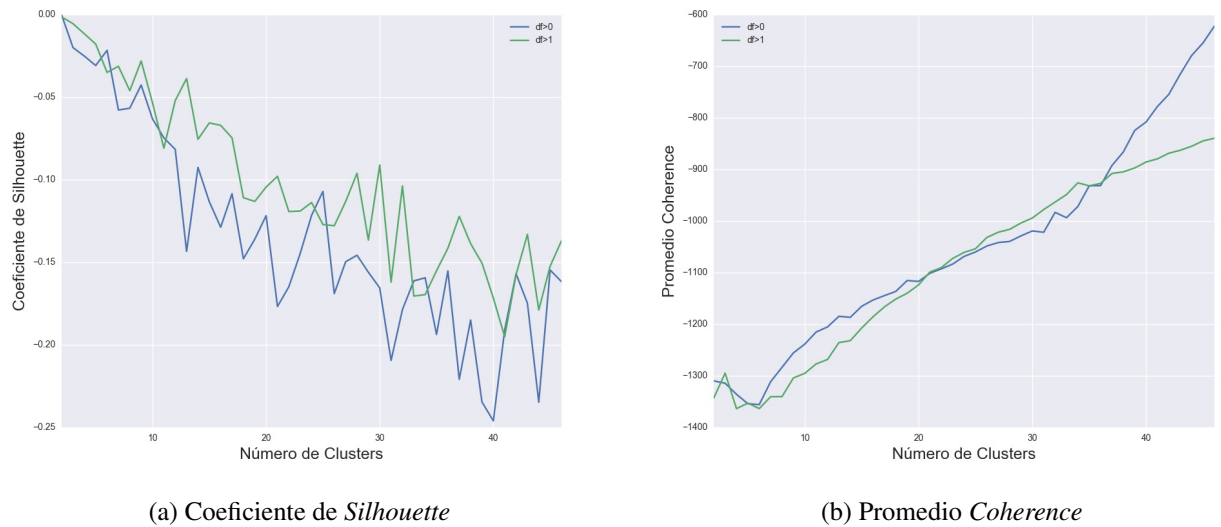


Figura 5.26: Métricas con agrupación de *skills* aleatoria.

Respecto al Coeficiente de *Silhouette* se observa que entre más *clusters* se consideren, peor será el valor de la métrica, llegando incluso a un valor de -0,20. Observando los distintos Coeficientes de *Silhouette* obtenidos a través de las técnicas que se utilizaron para encontrar las agrupaciones de *skills*, se observa que éstas poseen mejores valores a comparación de los valores de Coeficiente de *Silhouette* de la agrupación aleatoria.

Por otro lado, en el Promedio *Coherence* se evidencia que los primeros 10 *clusters* presentan un valor inferior a -1300, en cambio, en las técnicas utilizadas para agrupar *skills* comienzan con un valor de -1000 en el peor de los casos. Luego, si se observa el rango entre 10 y

20 *clusters* se observa que la agrupación aleatoria presenta valores inferiores a -1100, en cambio, con las técnicas utilizadas, presentan un valor inferior a -900 en el peor de los casos.

Finalmente se puede decir que las métricas obtenidas a partir de las distintas técnicas utilizadas son mucho mejores que las que se obtendrían vía un agrupamiento aleatorio.

5.2. Agrupaciones de Usuarios

En esta ocasión se realizará una estrategia alternativa de agrupación de *skills* utilizando la agrupación de usuarios. Para esto se utilizarán las técnicas nombradas en la sección 4.3.2 para poder agrupar los distintos usuarios y luego describir los grupos a través de las *skills* que declararon los usuarios dentro del grupo resultante. En esta ocasión se utilizará la misma leyenda que el objetivo anterior:

- $df > X$ indica que se consideraron todos los usuarios que declararon más de X *skills*.
- NN, L2 y TF-IDF, corresponde a la normalización de los datos utilizados. NN indica que no hubo normalización en los datos, L2 indica que los datos fueron normalizados con norma L2 y por último TF-IDF indica que se utilizó la normalización TF-IDF. En este caso, para mayor comprensión de TF-IDF, se considera los perfiles como los términos y las *skills* como documentos.
- $D0$ indica que a los datos obtenidos no se le redujo la dimensión, en cambio, $D430$ indica que los datos fueron reducidos hasta 430 dimensiones.

5.2.1. *K-Means*

La primera técnica que se utilizó para agrupar a los distintos usuarios fue la técnica *K-Means*.

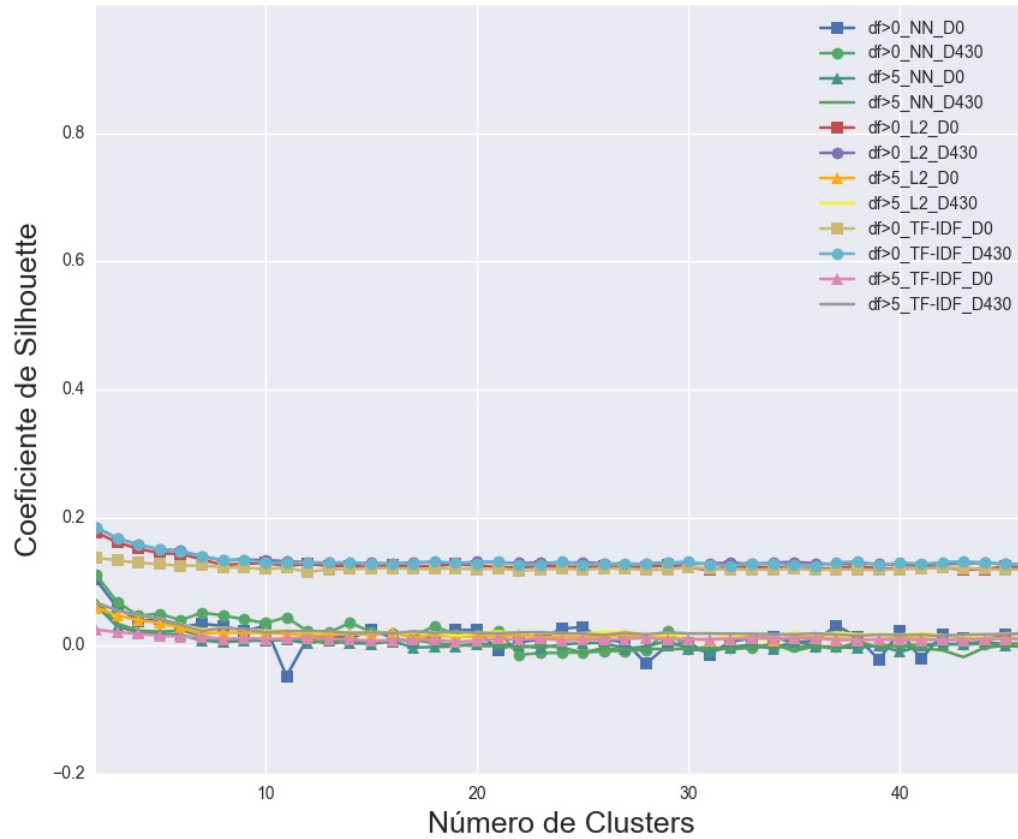


Figura 5.27: *K-Means*: Coeficiente de *Silhouette* con distintas normalizaciones de datos considerando agrupación de usuarios.

En el gráfico 5.27 se evidencia que la agrupación de datos sin normalizar y normalizados, no presentan la estructura idónea para ser calculada a través del Coeficiente de *Silhouette*. Las curvas sin normalizar están presentes entre 0,1 y 0,2 respecto al Coeficiente de *Silhouette*. Por otro lado, las curvas normalizadas se presentan entre -0,07 y 0,1, lo que significa que los datos no presentan estructura de *clusters* separados entre si y con sus datos juntos.

Otra métrica que se utilizó en este algoritmo fue el Promedio *Coherence* a continuación se presenta el gráfico relacionado a las mediciones:

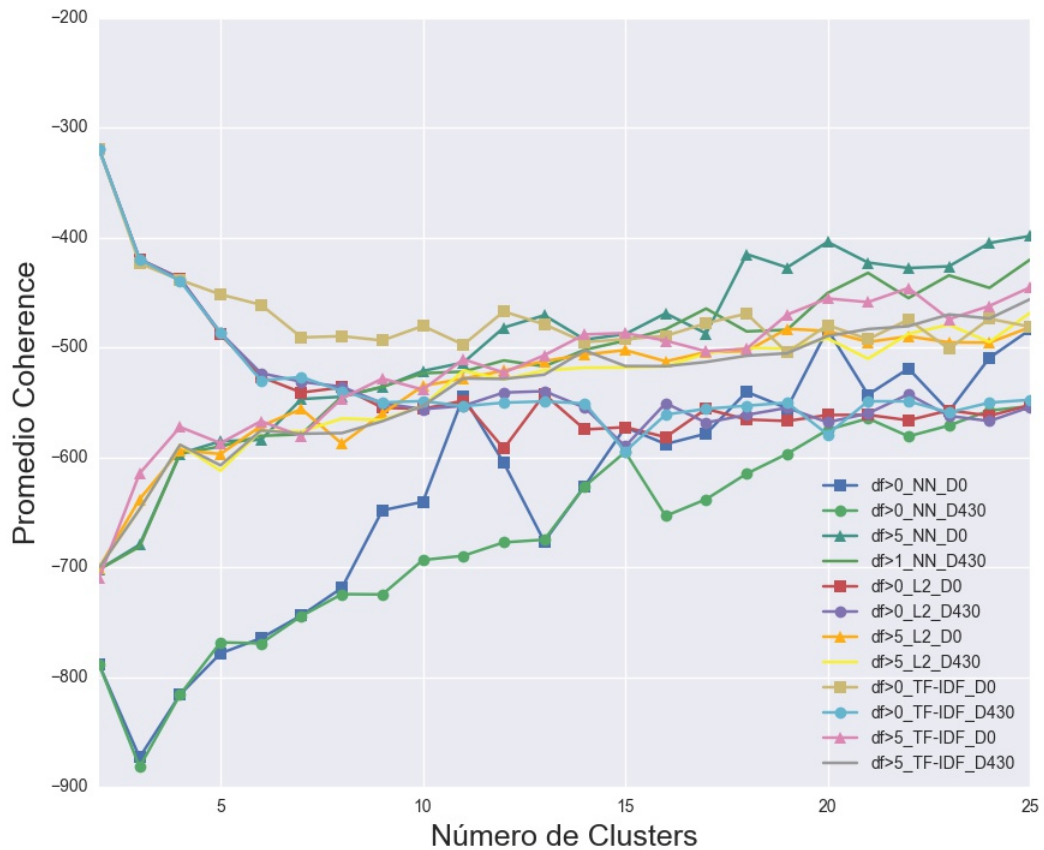
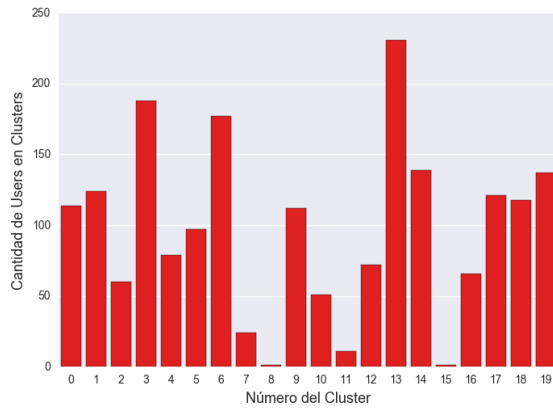


Figura 5.28: *K-Means*: Promedio *Coherence* con distintas normalizaciones de datos considerando agrupación de usuarios.

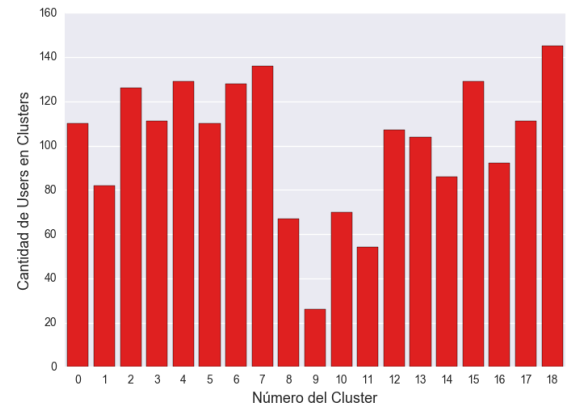
Lo primero que se puede observar en la figura 5.28, es que todas las curvas presentadas convergen a un rango de Promedio *Coherence* entre -470 y -580. A diferencia de otros gráficos, la no normalización, la normalización L2 y TF-IDF se comportan de distinta manera. En la no normalización las curvas ascienden con respecto al Promedio *Coherence*, cosa esperable gracias a los experimentos del objetivo correspondiente a la agrupación de *skills*. La normalización L2, presenta un decaimiento en las curvas en las cuales no se eliminaron los usuarios con cantidad de declaraciones inferiores a 5, en cambio, en la curvas L2 que se eliminaron estos usuarios se evidencia un aumento del Promedio *Coherence*. Todas las curvas de L2 se estabilizan cuando ya existen más de 10 *clusters*. Por último se observa que en las curvas

TF-IDF se presenta el mismo comportamiento que las curvas L2 con respecto al decaimiento y aumento de las curvas considerando las eliminaciones de usuarios.

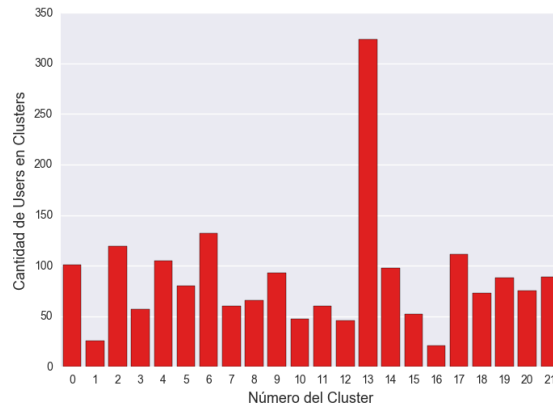
A continuación se procede a observar la distribución de los usuarios en los distintos *clusters* generados. No está demás mencionar que se elegirá la mejor configuración para cada normalización.



(a) Sin normalizar, $df > 5$ y sin reducción de dimensionalidad



(b) Normalización L2, $df > 5$ y sin reducción de dimensionalidad



(c) Normalización TF-IDF, $df > 5$ y sin reducción de dimensionalidad

Figura 5.29: Distribución de las distintas normalizaciones de *K-Means* considerando agrupación de usuarios.

Se observa en la figura 5.29 que existe una muy buena distribución de los usuarios dentro de los grupos generados, por lo tanto, se escogerán estas configuraciones para realizar la posterior descripción de los grupos encontrados.

Por último se tiene que las mejores configuraciones encontradas y justificadas por la métrica del Promedio *Coherence* y su distribución son las siguientes:

- No normalización: Se eliminan los usuarios que posean menos de 5 declaraciones y no se reducen dimensiones. Se eligen 20 *clusters*.
- Normalización L2: Se eliminan los usuarios que posean menos de 5 declaraciones y no se reducen dimensiones. Se eligen 19 *clusters*.
- Normalización TF-IDF: Se eliminan los usuarios que posean menos de 5 declaraciones y no se reducen dimensiones. Se eligen 22 *clusters*.

5.2.2. DBSCAN

Otra técnica utilizada para encontrar agrupaciones de usuarios en base a sus declaraciones de *skills* fue *DBSCAN*.

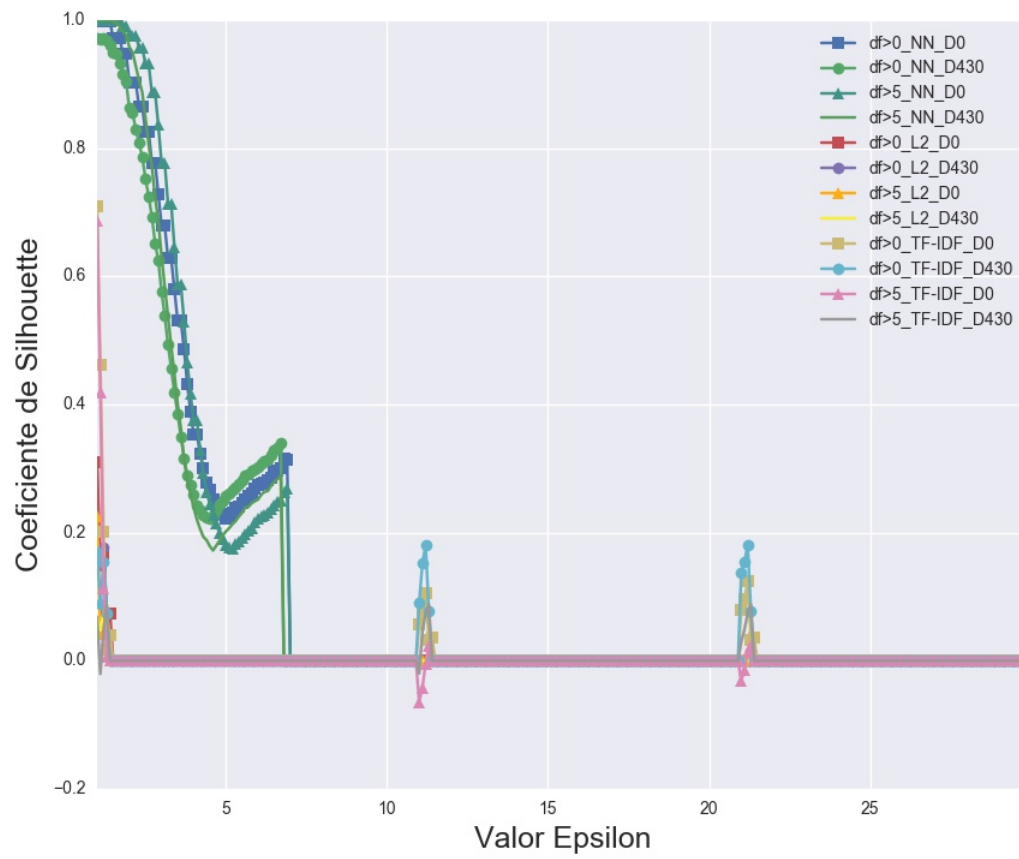


Figura 5.30: DBSCAN: Coeficiente de *Silhouette* con distintas normalizaciones de datos considerando agrupación de usuarios.

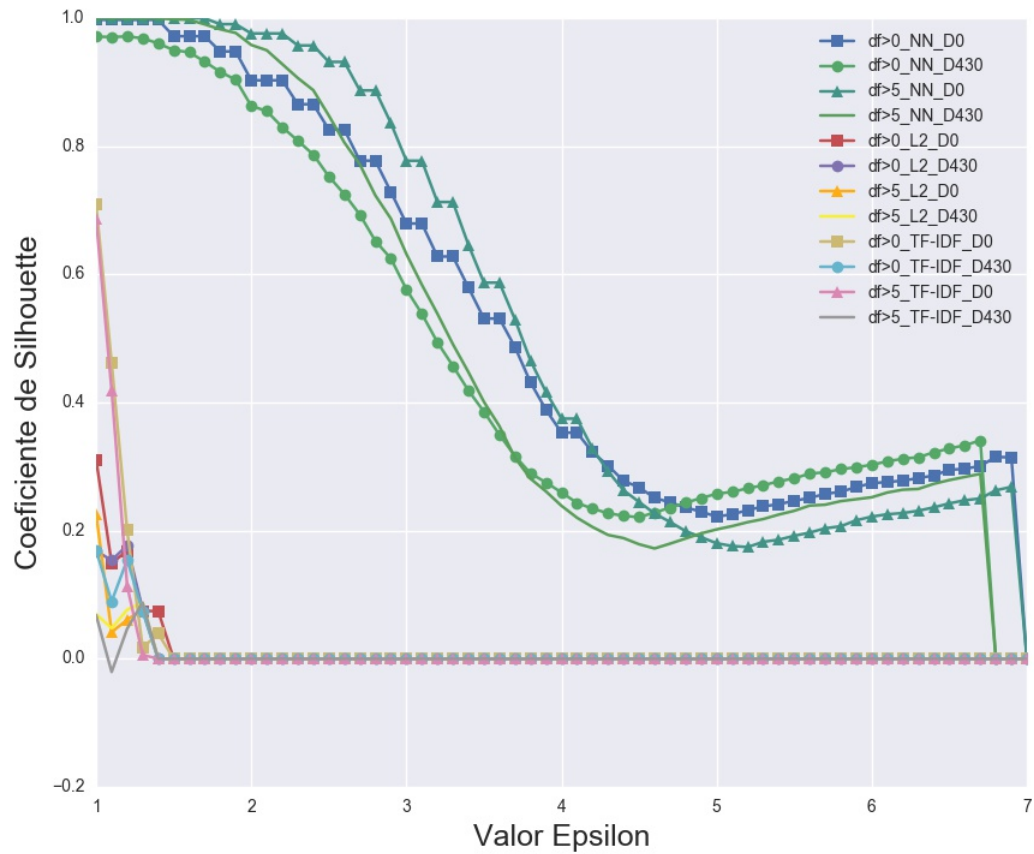


Figura 5.31: DBSCAN: Coeficiente de *Silhouette* con distintas normalizaciones de datos con rango de *Epsilon* disminuido y considerando agrupación de usuarios.

Se evidencia en los gráficos 5.30 y 5.31 una caída constante de todas las curvas presentadas en el gráfico. Estas curvas poseen un comportamiento distinto a las presentadas en el gráfico 5.5 correspondientes al objetivo de agrupación de *skills*. Recordar que a medida que crece *Epsilon* también disminuye la cantidad de *clusters* que va generando el algoritmo. En este caso cuando se generan más *clusters* el Coeficiente de *Silhouette* también disminuye.

Por otro lado también se obtuvo el Promedio *Coherence* para cada configuración de *Epsilon*.

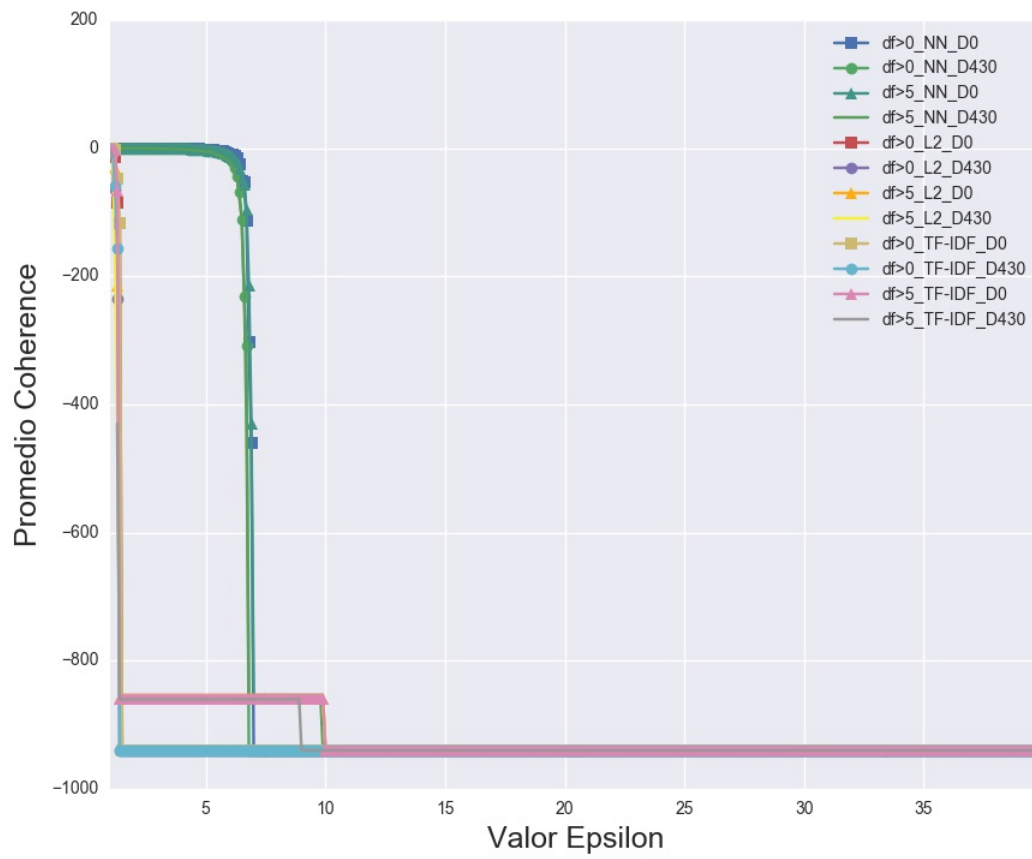


Figura 5.32: *DBSCAN*: Promedio *Coherence* con distintas normalizaciones de datos considerando agrupación de usuarios.

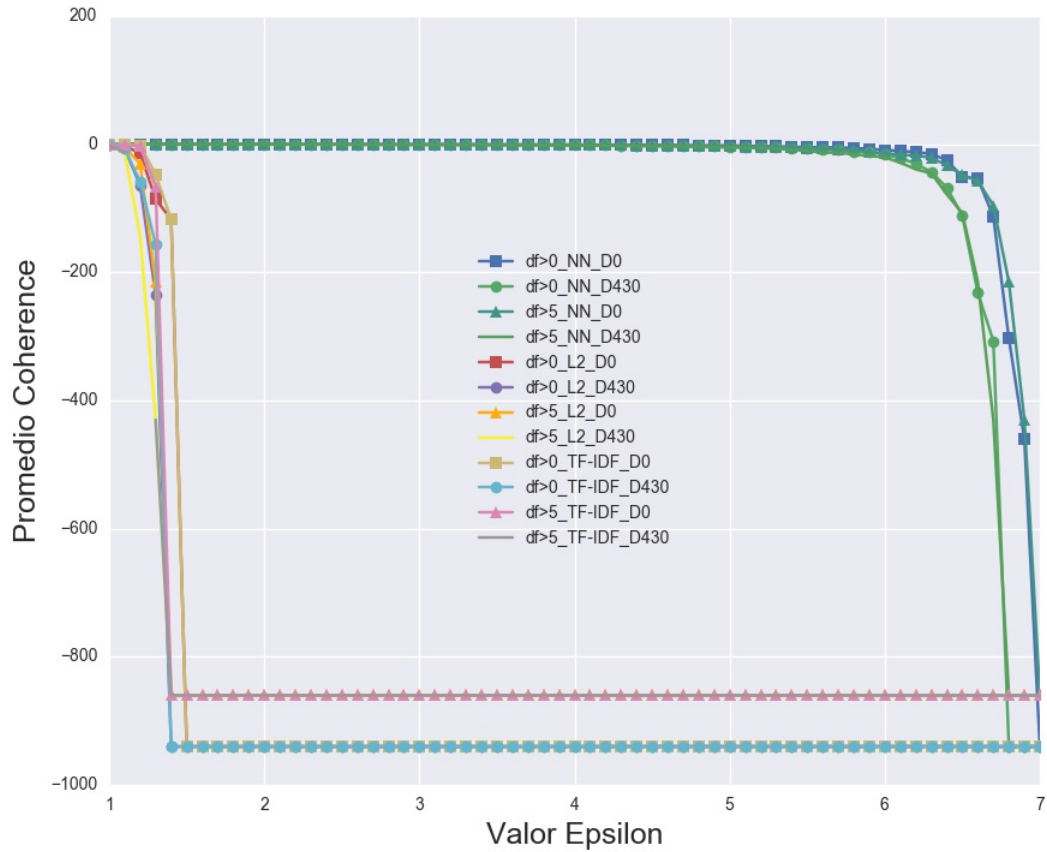


Figura 5.33: DBSCAN: Promedio *Coherence* con distintas normalizaciones de datos con rango de *Epsilon* disminuido considerando agrupación de usuarios.

En las figuras 5.32 y 5.31 se muestra una caída abrupta de todas las curvas de normalización presentadas. La no normalización decae después del valor *Epsilon* 6,8, en cambio, las normalizaciones L2 y TF-IDF decaen rápidamente luego del valor de *Epsilon* 1,3. Si bien al principio se presentan muy buenos Promedios *Coherence* esto se debe a que la cantidad de *clusters* están en la cantidad de los miles. Además, en algunos *clusters* solo se presentan 2 usuarios que si estos co-ocurren en los datos, *Coherence* dará un valor positivo, es por esta razón que este gráfico hay que complementarlo con la información de la cantidad de *clusters* generados.

Ahora para complementar los figuras 5.30 y 5.32, se procede a presentar las cantidades de

clusters más representativas que fueron generados en función de los *Epsilon* que presentan mejor Coeficiente de *Silhouette*.

Cuadro 5.8: Número de *clusters* variando el valor de *Epsilon*, considerando datos sin normalizar.

Epsilon	df>0_NN_D0	df>0_NN_D430	df>5_NN_D0	df>5_NN_D430
1,0	2012	1983	2012	1982
1,1	2012	1980	2012	1980
1,2	2012	1979	2012	1979
1,3	2012	1974	2012	1975
1,4	2012	1963	2012	1963
1,5	1983	1948	1983	1949
6,3	48	22	48	21
6,4	33	12	33	12
6,5	17	8	17	7
6,6	16	4	16	4
6,7	8	2	8	2
6,8	3	1	3	1
6,9	2	1	2	1
7,0	1	1	1	1

Cuadro 5.9: Número de *clusters* variando el valor de *Epsilon*, considerando datos con normalización L2.

Epsilon	df>0_L2_D0	df>0_L2_D430	df>5_L2_D0	df>5_L2_D430
1,0	670	378	670	380
1,1	226	84	226	85
1,2	45	11	45	10
1,3	11	4	11	3
1,4	8	1	8	1
1,5	1	1	1	1

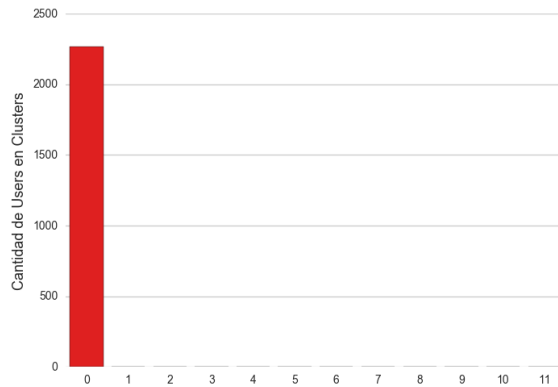
Cuadro 5.10: Número de *clusters* variando el valor de *Epsilon*, considerando datos con normalización TF-IDF.

Epsilon	df>0_TF-IDF_D0	df>0_TF-IDF_D430	df>5_TF-IDF_D0	df>5_TF-IDF_D430
1,0	1469	381	1469	380
1,1	950	89	950	88
1,2	333	12	333	12
1,3	20	4	20	6
1,4	8	1	8	1
1,5	1	1	1	1

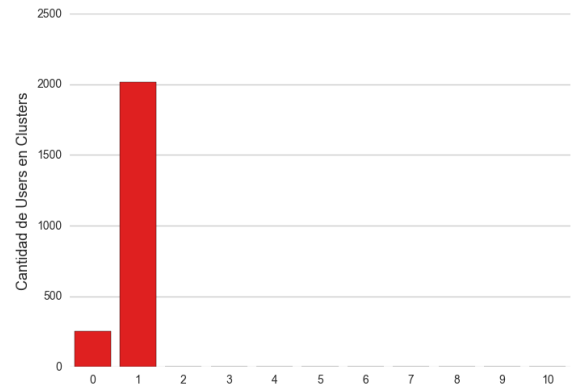
Observando las tablas 5.8 5.9 5.10 se evidencia que existen pocas configuraciones que posean una cantidad analizable de *clusters*. Considerando que una cantidad de *clusters* aceptable para analizar manualmente son menos de 15, se decidió por elegir cada normalización cuya configuración otorgue un número de *clusters* que esté cercano a este valor.

Lo obtenido fue lo siguiente:

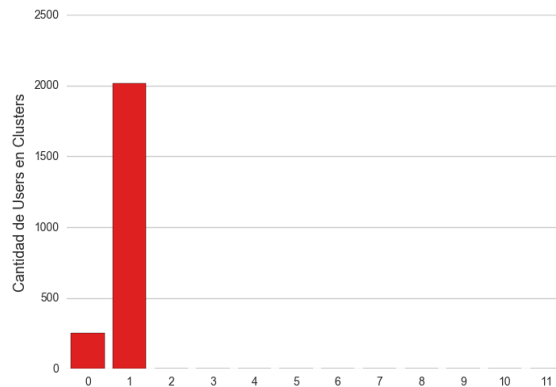
- No normalización: Se utilizó un *Epsilon* de 6,4 con cantidad mínima de puntos igual 1. El Coeficiente de *Silhouette* es 0,322 y el Promedio *Coherence* es -67,665. Con esta configuración se encontraron 12 *clusters*.
- Normalización L2: Se utilizó un *Epsilon* de 1,2 con cantidad mínima de puntos igual 1. El Coeficiente de *Silhouette* es 0,168 y el Promedio *Coherence* es -63,959. Con esta configuración se encontraron 11 *clusters*.
- Normalización TF-IDF: Se utilizó un *Epsilon* de 1,2 con cantidad mínima de puntos igual 1. El Coeficiente de *Silhouette* es 0,154 y el Promedio *Coherence* es -58,045. Con esta configuración se encontraron 12 *clusters*.



(a) Sin normalizar, $df > 0$ y con reducción de dimensionalidad



(b) Normalización L2, $df > 0$ y con reducción de dimensionalidad



(c) Normalización TF-IDF, $df > 0$ y con reducción de dimensionalidad

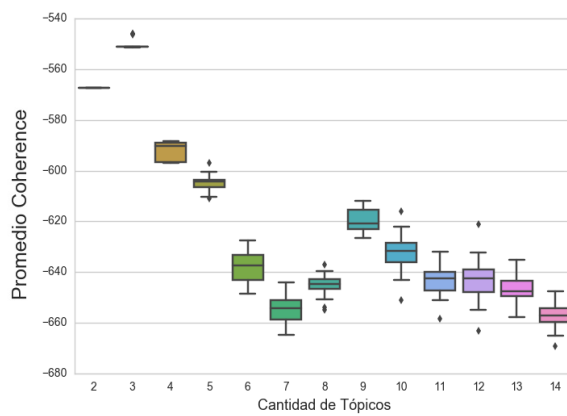
Figura 5.34: Distintas configuraciones de *DBSCAN* con distintas normalizaciones considerando agrupación de usuarios.

En la figura 5.34 se evidencia que existe una muy mala distribución de los *clusters*. En el caso de la no normalización, muchos puntos son ubicados en un solo *cluster*, por lo que sencillamente no sirve para realizar un análisis. Respecto a L2 y TF-IDF, sucede el mismo fenómeno que en la no normalización.

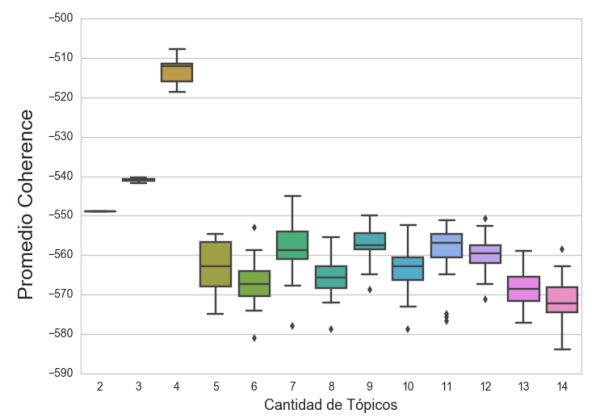
5.2.3. Topic Models

Otro tipo de algoritmo que se utilizó para encontrar agrupaciones de usuarios, son los algoritmos relacionados a los *Topic Models*. En este objetivo se utilizaron 3 técnicas: LSA, LDA y NMF.

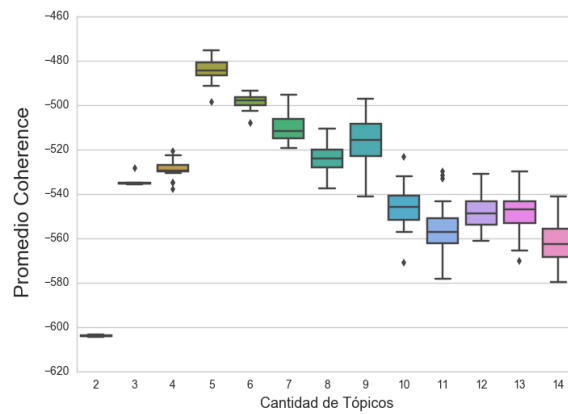
5.2.3.1. LSA



(a) Sin normalizar



(b) Normalización L2



(c) Normalización TF-IDF

Figura 5.35: LSA: Promedio *Coherence* con distintas normalizaciones de datos.

Se aprecia en los gráficos [5.35](#) que la técnica LSA se comporta de igual manera tanto para la no normalización como para las normalizaciones L2 y TF-IDF. En los 3 casos se puede observar que existe un punto donde el Promedio *Coherence* es superior al resto y luego comienza a decaer. Para el caso de no normalización este punto se eligió en la cantidad de tópicos 3, donde la mediana de su *boxplot* se encuentra entre el valor Promedio *Coherence* -540 y -560. Para la normalización L2 se tiene que la mejor cantidad de tópicos encontrados es 4 y la mediana se encuentra entre el valor de la métrica -510 y -520. En el caso de la normalización TF-IDF, se encontró que la mejor cantidad de tópicos es 5 y la mediana del *boxplot* se encuentra entre -480 y -490. Finalmente, con los datos recolectados anteriormente se realizarán las distintas agrupaciones de usuarios que luego serán descritas con las *skills* que presentan cada uno de ellos.

5.2.3.2. NMF

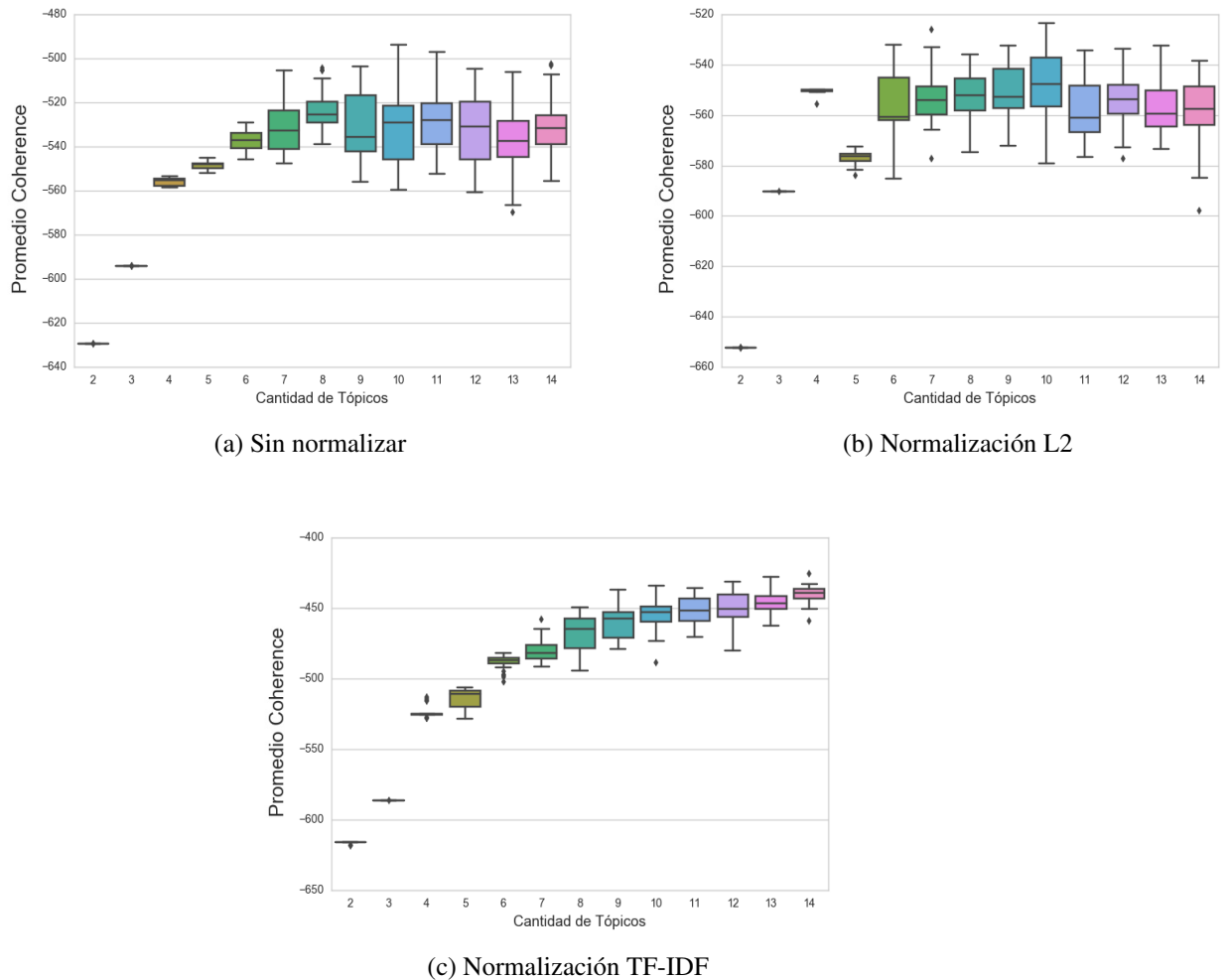
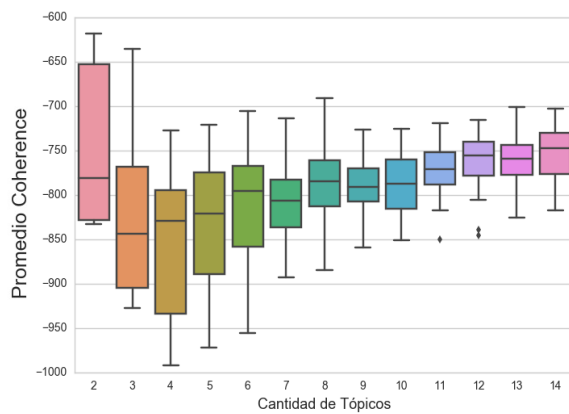


Figura 5.36: NMF: Promedio *Coherence* con distintas normalizaciones de datos.

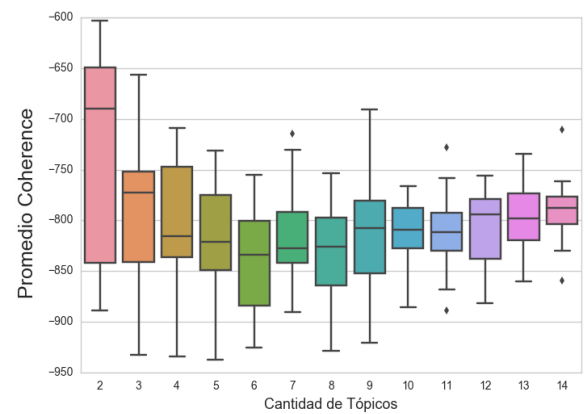
Como se evidencia en el gráfico 5.36 existe una tendencia de aumentar el Promedio *Coherence* a medida que aumentan la cantidad de tópicos. Otro punto destacable es que las primeras cantidades de tópicos presentan muy baja dispersión en la técnica NMF. Respecto a la no normalización se tiene que la mejor cantidad de tópicos se encuentra en el número 8 y la mediana del *boxplot* correspondiente se ubica entre -520 y -540. Respecto al gráfico que presenta L2, se evidencia que existe un máximo local entre todas las mediciones cuyo *boxplot* presenta una dispersión mínima, este se encuentra en el número de tópicos 4 y la mediana

del *boxplot* se encuentra entre -540 y -560. En el caso de TF-IDF se presenta un aumento del Promedio *Coherence* a medida que aumenta la cantidad de tópicos. A su vez, los *boxplots* generados presentan menos dispersión en los datos a comparación de los *boxplots* generados con la no normalización y la normalización L2. En este caso el mejor número de tópicos encontrados es 14 cuya mediana de *boxplot* se encuentra entre -425 y -450.

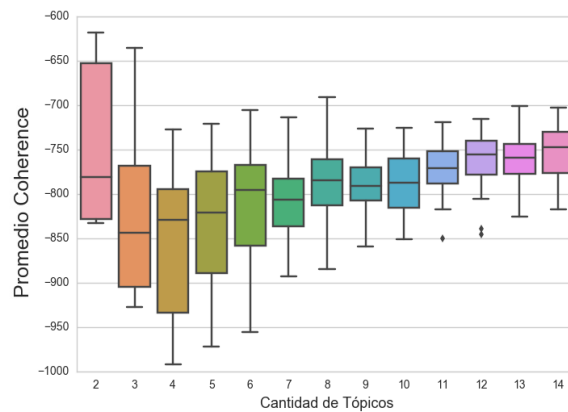
5.2.3.3. LDA



(a) Sin Normalizar



(b) Normalización TF-IDF



(c) Normalización L2

Figura 5.37: LDA: Promedio *Coherence* con distintas normalizaciones de datos.

En la figura presentada en 5.37 se evidencia que los *boxplots* de todas las normalizaciones tienen una gran cantidad de dispersión comparado con las figuras 5.35 5.36. Sin embargo se presenta una tendencia de converger a un punto en cada gráfico distinto. Para el caso de la no normalización se decidió que la mejor cantidad de tópicos a elegir es 5 cuya mediana del *boxplot* se encuentra entre -700 y -725. La normalización L2 presenta la mejor configuración en la cantidad de tópicos 9 donde la mediana del *boxplot* entre -750 y -800. Por otro lado, en la normalización TF-IDF se tiene que el mejor valor escogido del número de tópicos es 10 donde la mediana de las mediciones se encuentra entre -750 y -800.

5.2.3.4. LSA + *K-Means*

Otra técnica que se utilizará para encontrar agrupaciones de usuarios es aplicar el algoritmo *K-Means* a la mejor configuración de LSA encontrada.

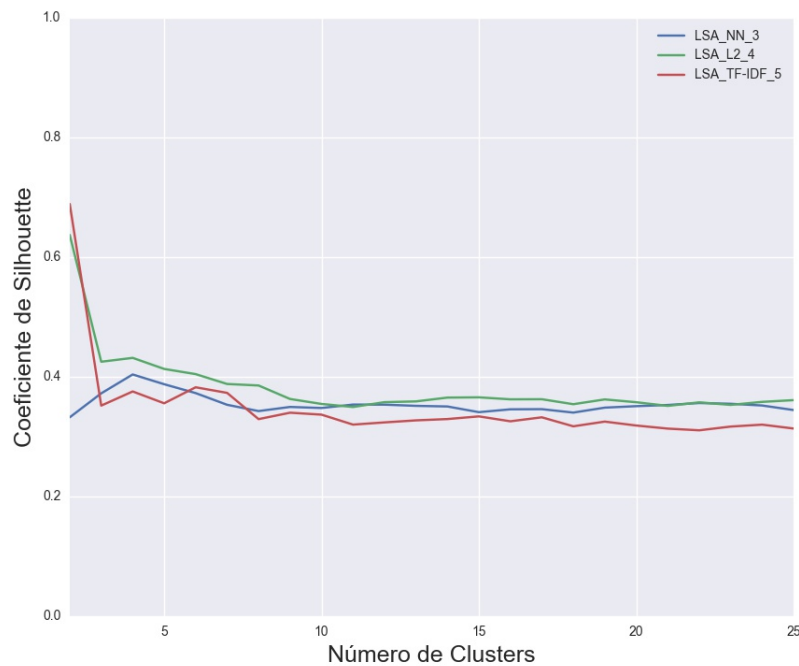


Figura 5.38: LSA + *K-Means*: Coeficiente de *Silhouette* con distintas normalizaciones de datos.

En el gráfico 5.38 se observa que las 3 curvas presentadas poseen un comportamiento similar a medida que aumenta la cantidad de *clusters* generados, la única diferencia de estas curvas se presenta entre los valores 1 y 5, que es donde las curvas se ajustan y siguen con un valor casi constante hasta el *cluster* 25. Cabe notar que estas curvas convergen a un valor aproximado del Coeficiente de *Silhouette* que oscila entre 0,4 y 0,3.

Por otro lado se tiene que el mejor valor para la no normalización se encuentra en la cantidad de grupos 2, pero como esta cantidad no otorga suficiente información de las agrupamientos resultantes, se optó por utilizar 4 *clusters* con un valor de Coeficiente de *Silhouette* igual a 0,405. Para la normalización L2 se tiene que el mejor valor se encuentra con cantidad de grupos igual a 4 y un valor del Coeficiente de *Silhouette* igual a 0,431. En el caso de la normalización TF-IDF, sucede el mismo caso que en la no normalización, salvo que acá el segundo mejor punto se tiene con 6 grupos con un Coeficiente de *Silhouette* igual a 0,382.

Para obtener más argumentos para las distintas selecciones de grupos también se consideró tomar la métrica del Promedio *Coherence*.



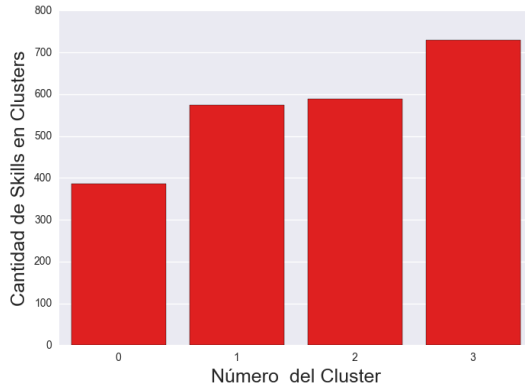
Figura 5.39: LSA + *K-Means*: Promedio *Coherence* con distintas normalizaciones de datos.

Se observa en la figura 5.39 que las normalizaciones L2 y TF-IDF se comportan de manera distinta a la no normalización, ya que este último a medida que aumentan la cantidad de *clusters* también aumenta el valor de la métrica obtenida, en cambio, las otras normalizaciones disminuyen su métrica a medida que la cantidad de *clusters* aumenta. Lo otro notable es que las normalizaciones poseen mejores valores del Promedio *Coherence* que los datos no normalizados.

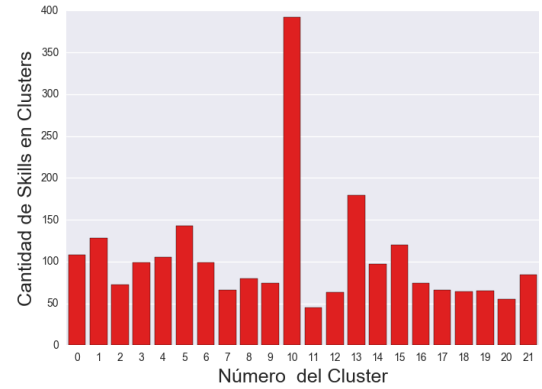
El mejor valor encontrado para la no normalización fue -669,959 que se encontraba en la cantidad de grupos 22. Para las normalizaciones L2 y TF-IDF se tiene que el mejor valor considerado no es necesariamente el que posee mayor valor de esta métrica ya que en este caso también hay que ponderar una cantidad de grupos que otorgue información valiosa

respecto al objetivo que se desea lograr. Para ambas normalizaciones la cantidad de grupos óptimos es 3 y el valor de las métricas es -419,616 y -423,396 respectivamente.

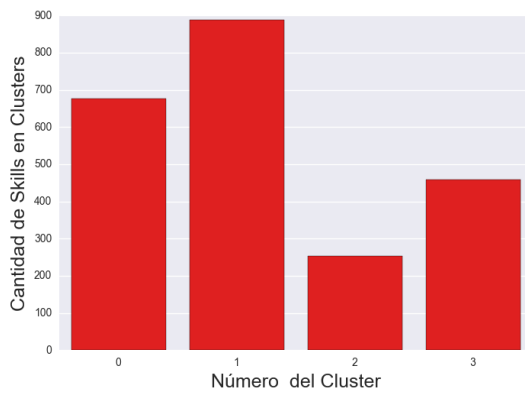
Luego de elegir cuáles serán los mejores grupos de esta técnica se procede a ver cómo se distribuyen los usuarios en los *clusters* generados.



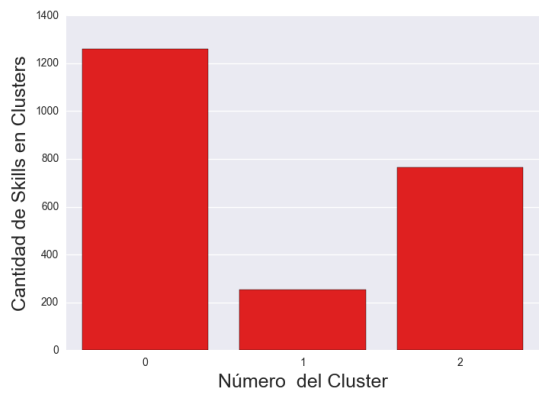
(a) Sin normalizar con 4 *clusters*



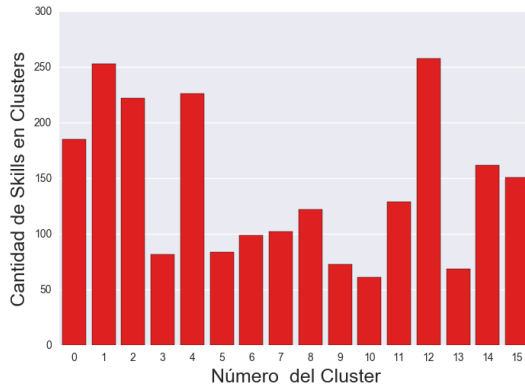
(b) Sin normalizar con 22 *clusters*



(c) Normalización L2 con 4 *clusters*



(d) Normalización L2 con 3 *clusters*



(e) Normalización TF-IDF con 16 clusters



(f) Normalización TF-IDF con 3 clusters

Figura 5.39: LSA: Distribución de los usuarios en los distintos *clusters*.

Para las distribuciones presentadas en la figura 5.39 se evidencia que existe una distribución analizable de usuarios en los *clusters*. En este caso se elegirán las configuraciones que den más facilidades para poder realizar un análisis manual, salvo por la normalización TF-IDF que se elegirán los 16 *clusters* debido a que presenta una mejor distribución que con 3 *clusters*.

5.2.3.5. LDA + *K-Means*

La última técnica que se utilizó fue aplicar el algoritmo *K-Means* a los resultados de la mejor configuración de LDA.

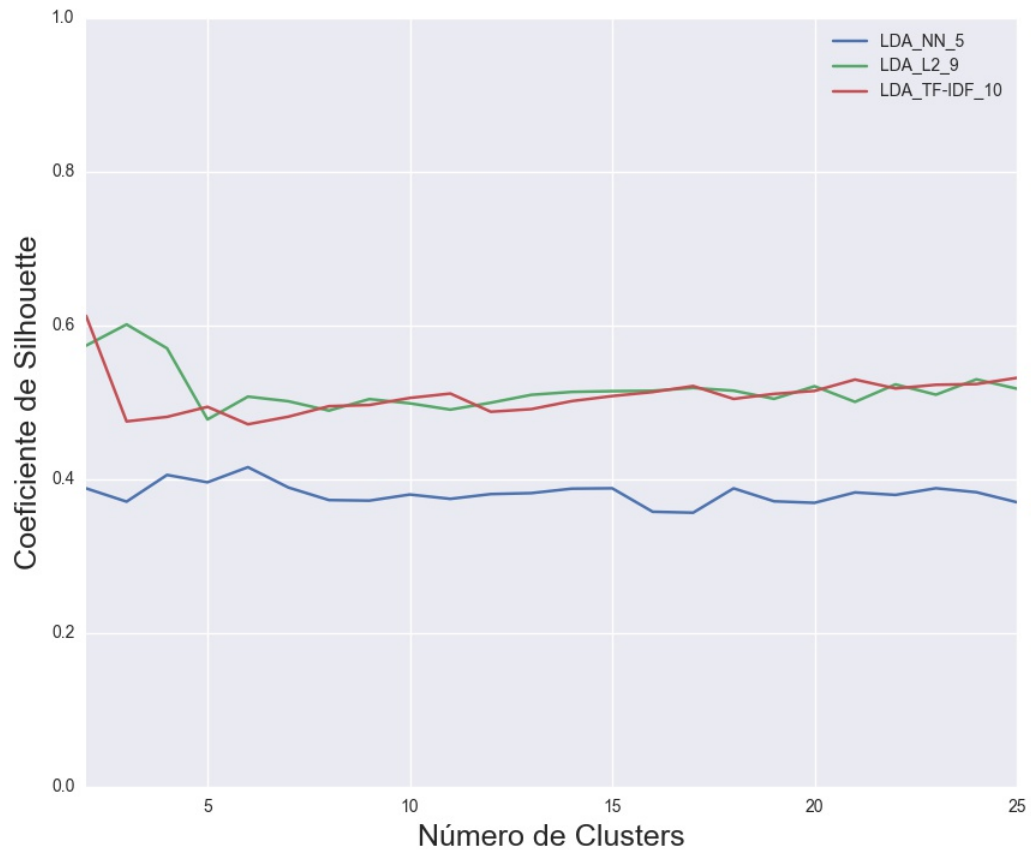


Figura 5.40: LDA + *K-Means*: Coeficiente de *Silhouette* con distintas normalizaciones de datos.

En la figura 5.40 se puede observar una tendencia a convergencia al igual que en la figura 5.38. La diferencia de éste es que se puede notar una gran diferencia entre los valores de las normalizaciones L2 y TF-IDF comparado con los datos sin normalizar. Ambas normalizaciones convergen a un valor del Coeficiente de *Silhouette* aproximado de 0,5, en cambio, la no normalización converge a un valor aproximado de 0,4.

Por otro lado, el mejor punto obtenido para la no normalización se presenta en 6 grupos con un valor del Coeficiente de *Silhouette* de 0,415. Para la normalización L2 se tiene que la mejor cantidad de grupos se presenta en 3 con un valor de la métrica de 0,601. Finalmente para TF-IDF se tiene que la mejor cantidad de grupos es 11 y el valor del Coeficiente de

Silhouette es 0,511

Otra métrica que se utilizó fue la del Promedio *Coherence* para así obtener más información de los *clusters* generados.

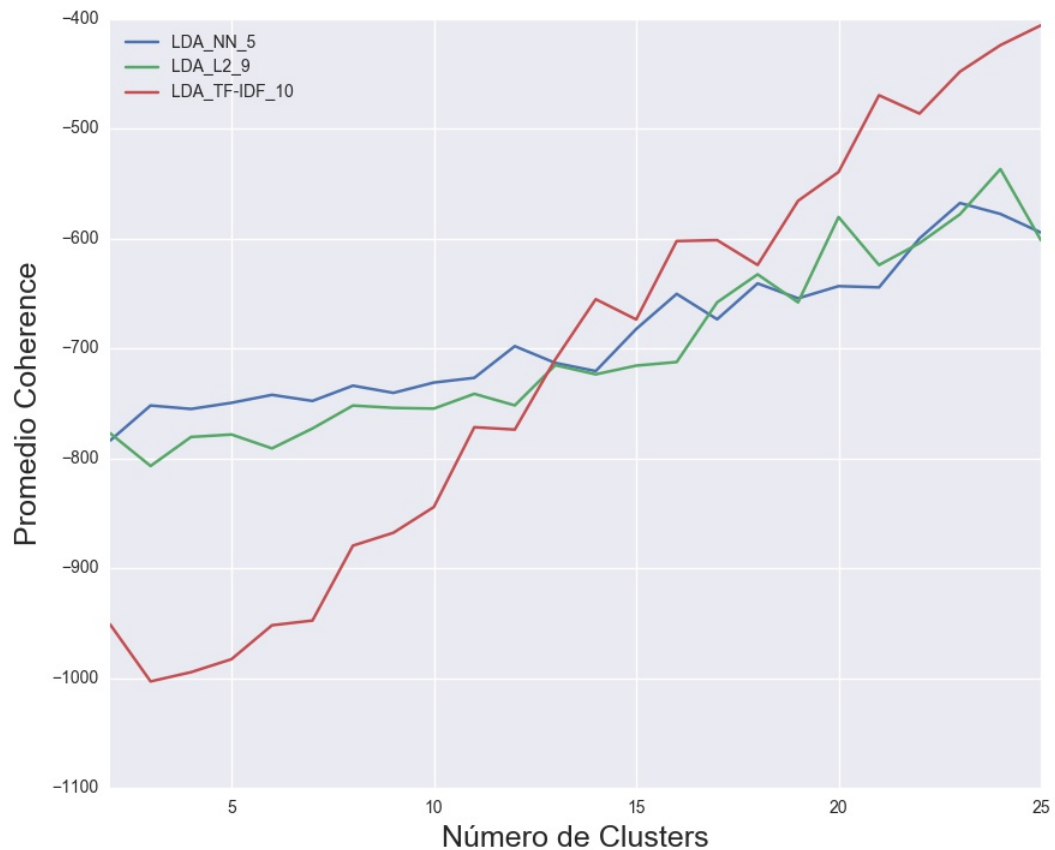


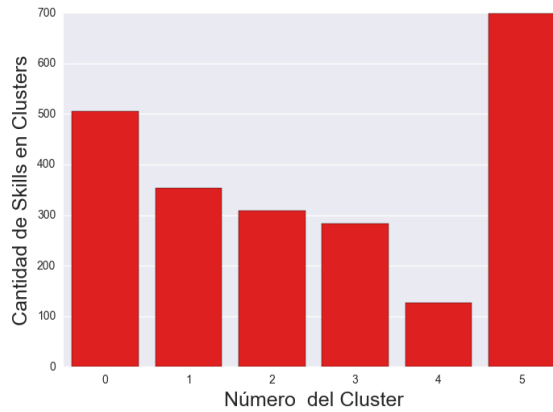
Figura 5.41: LDA + *K-Means*: Promedio *Coherence* con distintas normalizaciones de datos.

Lo que se puede observar de la figura 5.41 es una tendencia de aumento del Promedio *Coherence* a medida que aumenta la cantidad de *clusters*, esto se presenta para las 3 curvas de distinta normalización. Se evidencia que la normalización TF-IDF destaca por sobre las demás por el hecho de que su incremento es más abrupto que las otras 2 curvas.

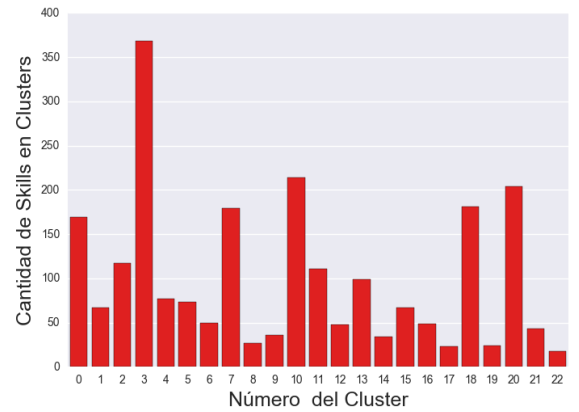
Debido a que estas curvas presentan gran incremento a medida que aumenta la cantidad de *clusters*, no es posible elegir una cantidad pequeña para realizar el análisis de los *clusters*

generados, por lo tanto se elegirán los puntos que son efectivamente los que poseen mejor valor de la métrica escogida. En este caso la no normalización presenta su mejor valor en la cantidad de *clusters* 23 con un valor del Promedio *Coherence* de -567,489. En la normalización L2 el mejor punto se ubica en la cantidad de *clusters* 24 con un valor de -536,608. Para TF-IDF se tiene que el mejor punto se ubica al final del gráfico, con un valor de número de *clusters* igual a 25 y un valor del Promedio *Coherence* de -405,699.

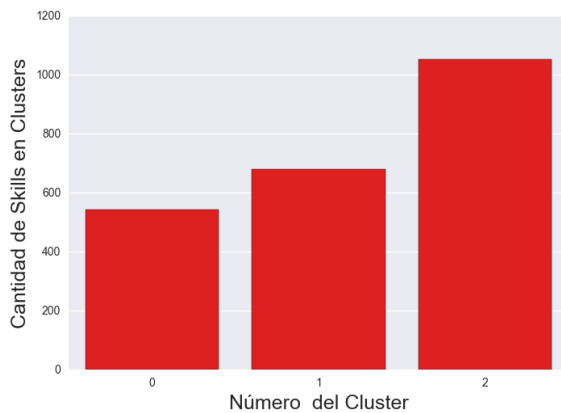
A continuación se procede a ver cómo se distribuyen los usuarios en los *clusters* generados, considerando las distintas configuraciones encontradas.



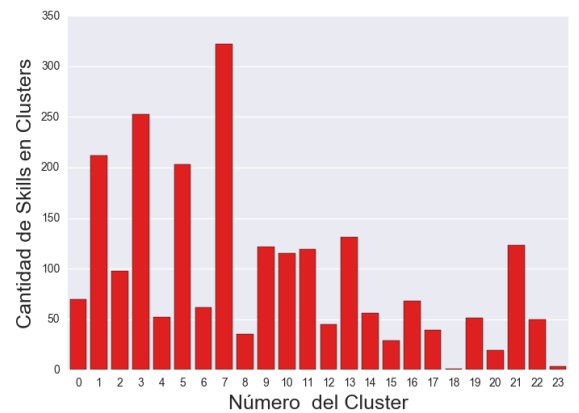
(a) Sin normalizar con 6 *clusters*



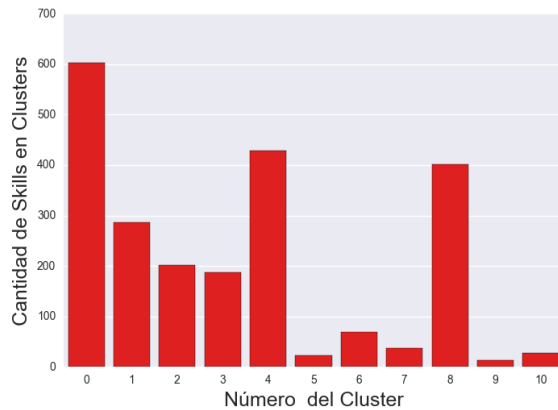
(b) Sin normalizar con 23 *clusters*



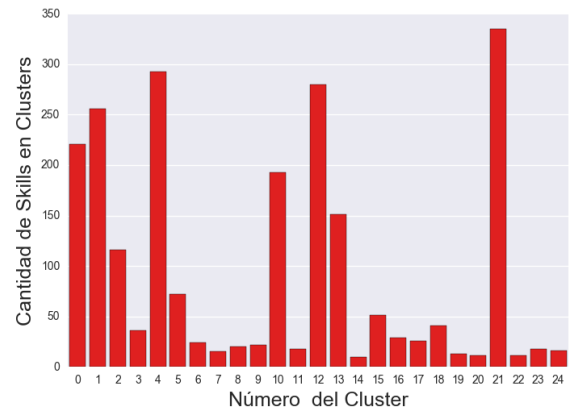
(c) Normalización L2 con 3*clusters*



(d) Normalización L2 con 24*clusters*



(e) Normalización TF-IDF con 11clusters



(f) Normalización TF-IDF con 25clusters

Figura 5.41: LDA: Distribución de los usuarios en los distintos *clusters*.

En los gráficos 5.41 se observa que todas las configuraciones poseen una buena distribución de usuarios en los distintos *clusters*. En este caso se elegirán las configuraciones que sea más fáciles de analizar de forma manual, por lo que se elegirá de cada normalización la menor cantidad de *clusters*.

5.2.4. Agrupación de Usuarios: *Word Clouds*

Recordar que para la agrupación de usuarios se utilizó la declaración de las *skills* para representar los distintos grupos. Considerando lo anterior se explicará el cómo se generaron los distintos *Word Clouds* que se obtuvieron de estas agrupaciones.

Considerando que los grupos de usuarios encontrados se explicarán a través de las declaraciones de *skills* de cada perfil dentro del *cluster* generado, cada *skill* tendrá como peso la cantidad de veces que se declara esa *skill* en un *cluster* en particular. El peso nombrado anteriormente también estará relacionado con el tamaño de la palabra dentro del *Word Cloud*. En este caso solo se agregaron las *top 40 skill* que tuvieran frecuencia mayor que 1. Respecto a los *Topic Models* hay que considerar que solamente los usuarios poseen peso, por lo que en esta ocasión se optó por utilizar el mismo método para pesar las *skills* dentro de los tópicos generados, considerando solo las declaraciones de los usuarios cuyos pesos estuvieran sobre



Figura 5.42: *Word Cloud* de usuarios con *skills* de distintas áreas.

Otro punto importante es que en la mayoría de los *Word Clouds* de la figura 5.41 se presentaban 3 *skills* dominantes: Gestión de proyectos, Inteligencia empresarial y Liderazgo de equipos. Estas *skills* se eliminaron de los *Word Clouds* para así presentar grupos que se pudieran distinguir entre sí. La razón por la cual estas *skills* son dominantes es porque pertenecen al *top 10* de *skills* declaradas.

Por otro lado se presenta la tabla 5.11 que indica qué algoritmos encontraron los grupos mencionados en la figura 5.41

Cuadro 5.11: Detalle de técnicas y grupos de usuarios encontrados.

Grupo / Técnica	K-Means	LSA+ K-Means	LDA + K-Means
ITIL	√		
Programación	√		
Base de datos	√		
SAP	√		
Estrategia empresarial	√		
Desarrollo WEB		√	√
Servidores			√

En la tabla 5.7 solo se presentan las técnicas que encontraron agrupaciones coherentes.

Si bien los *Topic Models* por sí solos no pudieron encontrar una agrupación que sea capaz de describirse a través de *skills*, esto no quiere decir que estas agrupaciones no tengan sentido, sino que el atributo *skills* que se está utilizando puede que no sea el óptimo para describir los grupos encontrados.

Por último se presenta la figura 5.43 que presenta la distribución de los usuarios según universidad en los distintos grupos encontrados.

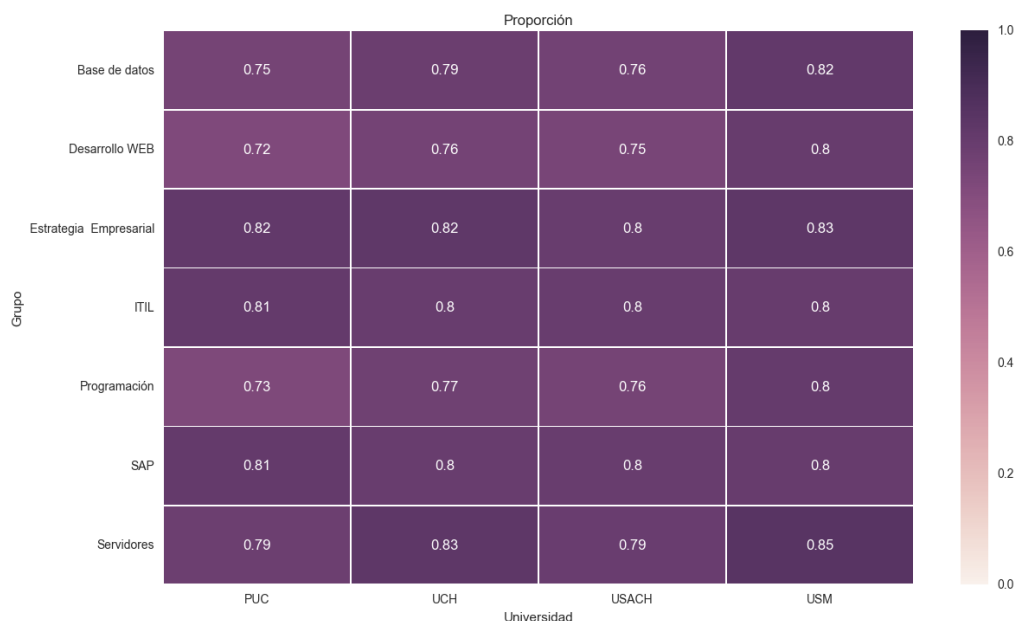


Figura 5.43: Distribución de usuarios en los grupos según universidad.

Lo primero que se puede observar es que no existe gran diferencia de proporciones entre las distintas universidades considerando los grupos propuestos. Por otro lado se evidencia que todas las proporciones superan el 70 %, lo que quiere decir que la mayoría de usuarios de todas las universidades declaran por lo menos 1 *skill* correspondiente a cada grupo. También se observa la discrepancia de proporciones existentes entre algunos grupos de la figura 5.25 y 5.43. El caso más destacable es el de Programación, en la figura 5.25 se observa que esta proporción no supera el 50 % en todas las universidades, sin embargo, en la figura 5.43 se evidencia que esta proporción es más del 70 %. La razón de esta diferencia puede ser debido a la diferencia de criterios existentes entre la generación de los *Word Clouds*. Recordar que los grupos de la figura 5.43 fueron creados a partir de las declaraciones de *skills* de los usuarios agrupados con las distintas técnicas, por lo que la posibilidad de que siempre exista un usuario que declare al menos una *skill* de ese *cluster* es muy alta. Otro punto importante es que no necesariamente ambos grupos de Programación sean los mismos considerando la opinión experta, ya que recordar que las etiquetas de los grupos fueron generadas solamente por el juicio y conocimiento del autor de esta memoria.

5.2.5. Grupos Aleatorios

Para verificar el comportamiento de las distintas métricas utilizadas, se generaron agrupaciones aleatorias de usuarios considerando distribución uniforme entre los *clusters* generados. Se generaron distintos grupos considerando todos los usuarios y los usuarios que hayan declarado por lo menos 5 *skills*.

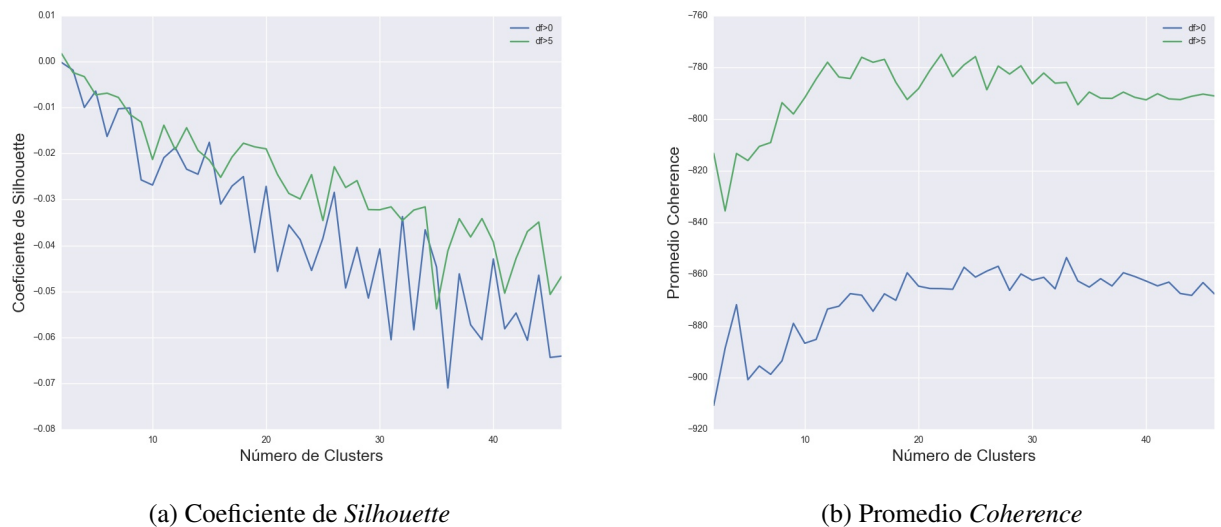


Figura 5.44: Métricas con agrupación de usuarios aleatoria.

Respecto al Coeficiente de *Silhouette*, se observa que los valores obtenidos si bien tienden a negativo, estos no sobrepasan el -0,1. Si se compara estos valores con las técnicas que se utilizaron para encontrar la agrupación de usuarios, se evidencia que estas últimas presentan mejores valores dependiendo del algoritmo. Para *K-Means* no presenta una gran diferencia ya que el mejor valor del Coeficiente de *Silhouette* encontrado ronda por el valor 0,2, sin embargo, para las técnicas *LSA + K-Means* y *LDA + K-Means*, existe una gran diferencia ya que estos últimos presentan valores que rondan los 0,4.

Para el Promedio *Coherence* se evidencia una convergencia para las distintas curvas. La curva en la cual se eliminan los usuarios tiende a un valor que está entre -780 y -800, en cambio, la curva que posee todos los usuarios converge a un valor entre -860 y -880. Comparando

estos valores con las técnicas que encontraron agrupaciones de usuarios se presenta que nuevamente estas últimas poseen mejores valores con creces, ya que considerando hacia donde converge cada técnica se encuentran diferencias entre -300 y -400. Considerando el rango entre 10 y 20 *clusters* se evidencia una diferencia de -100 con respecto al Promedio *Coherence*.

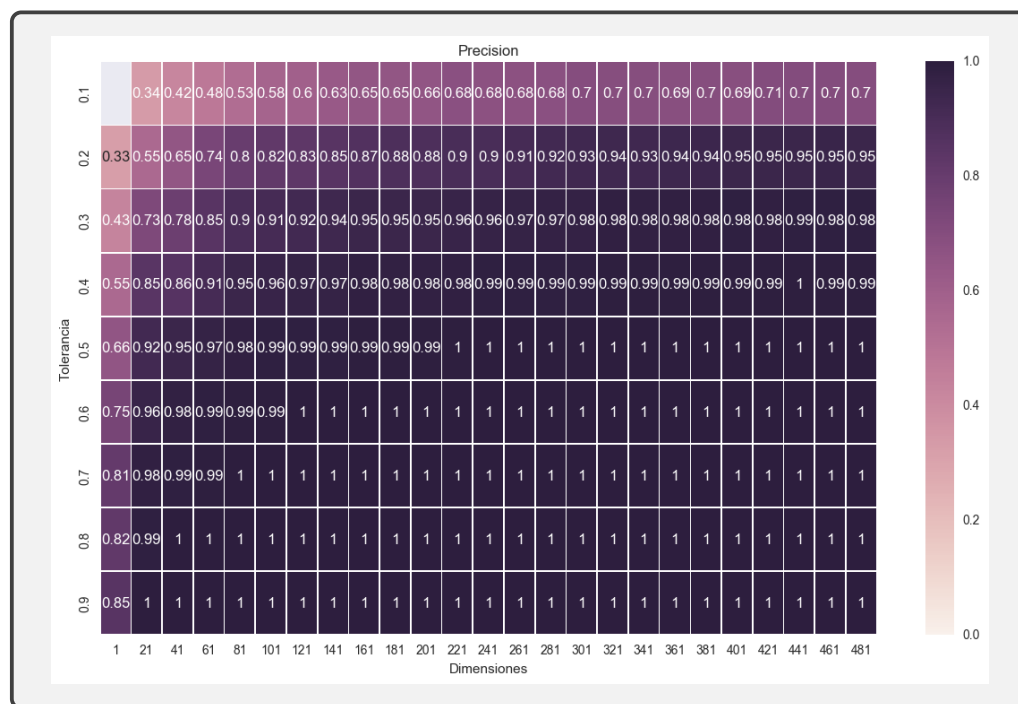
Finalmente se puede decir que las métricas obtenidas a partir de las distintas técnicas utilizadas son mucho mejores que las que se obtendrían vía un agrupamiento aleatorio.

5.3. Recomendador de *skills*

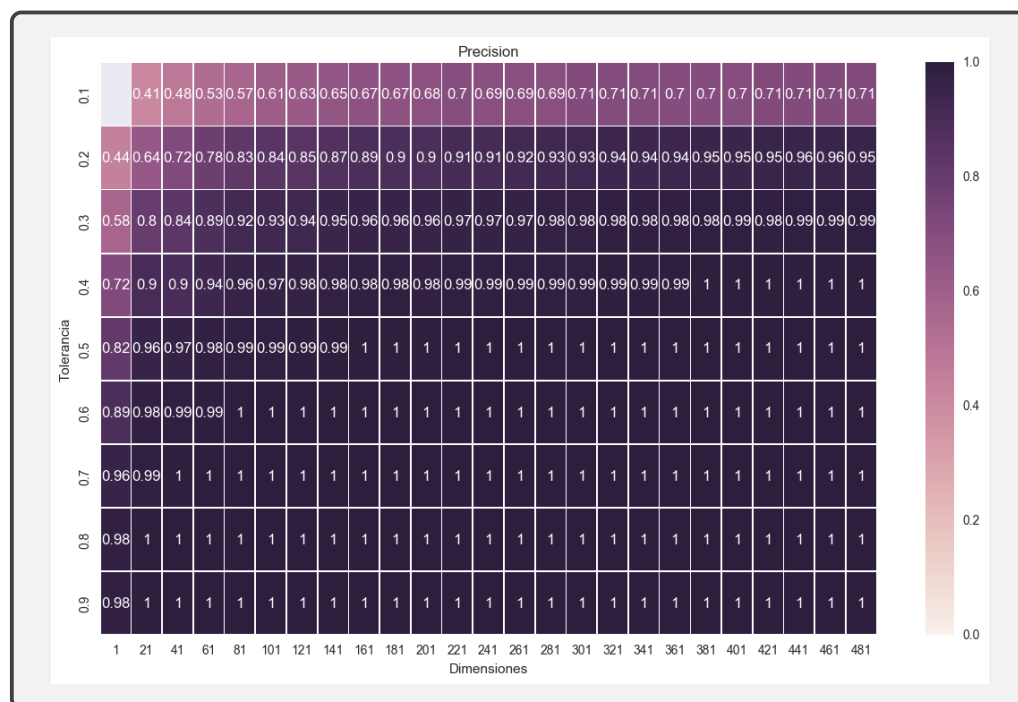
En esta sección se observarán los resultados obtenidos a partir de las distintas técnicas aplicadas para realizar un recomendador de *skills* en base a las declaraciones realizadas por parte de los usuarios.

5.3.1. NMF

La primera técnica que se utilizó fue NMF. Para el análisis de estos algoritmos se utilizaron 3 métricas: *Precision*, *Recall* y *F-Measure*. Las técnicas mencionadas anteriormente se aplicaron a 2 conjuntos de datos: entrenamiento y pruebas. A continuación se presentan las distintas métricas con los distintos conjuntos de datos y su respectivo análisis.

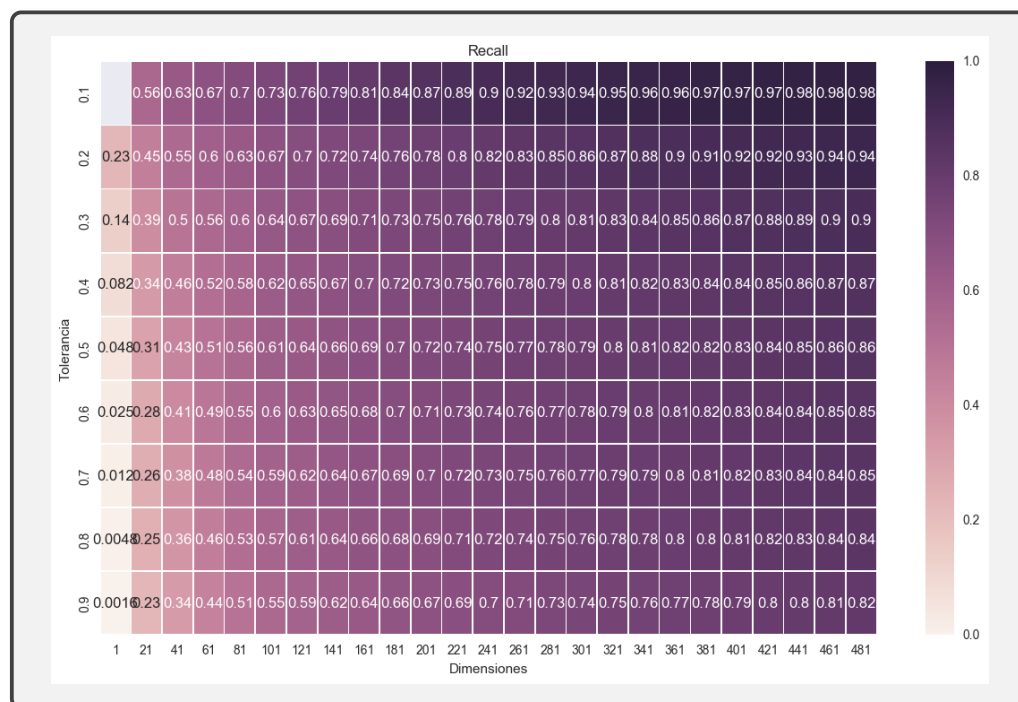


(a) NMF: *Precision* del conjunto de entrenamiento

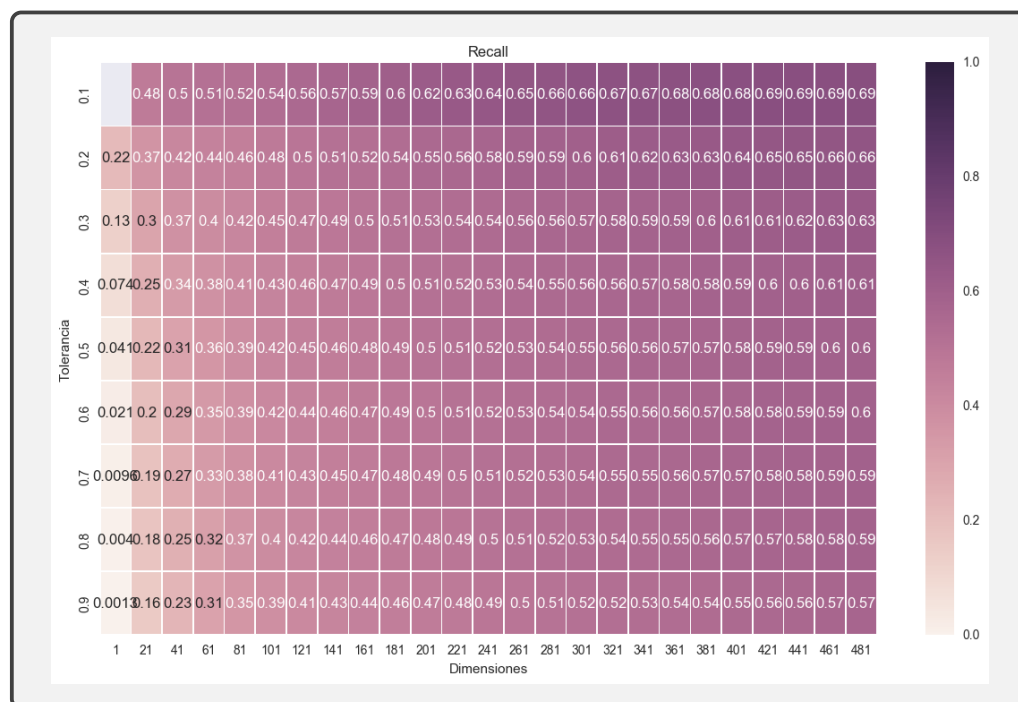


(b) NMF: *Precision* del conjunto de pruebas

Figura 5.45: NMF: *Precision* del conjunto de entrenamiento y pruebas.

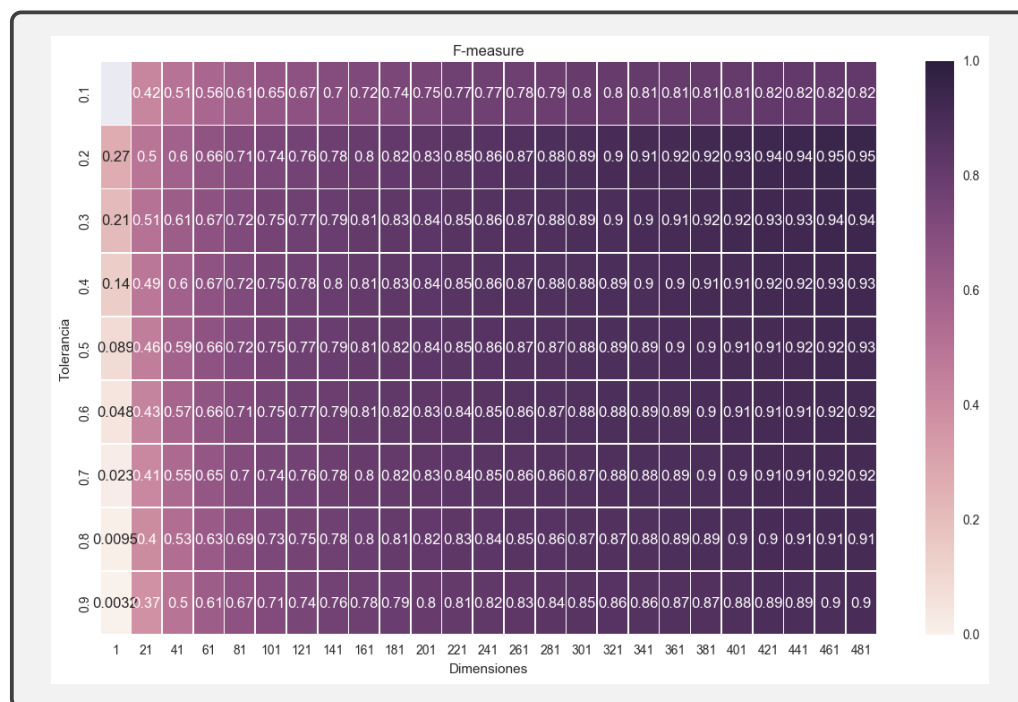


(a) NMF: *Recall* del conjunto de entrenamiento

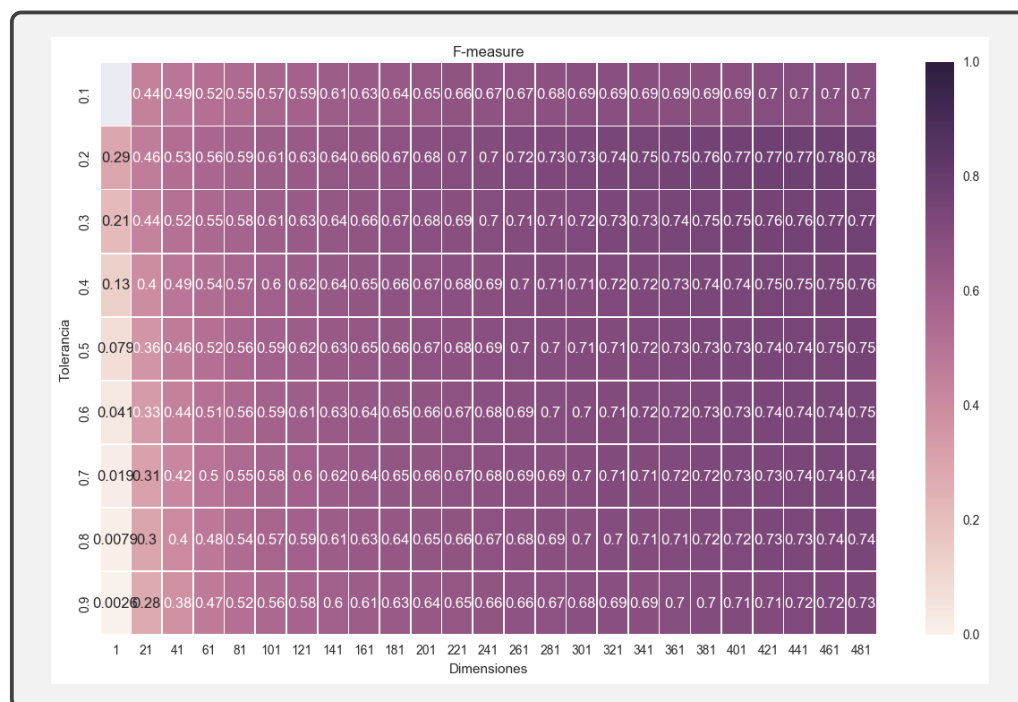


(b) NMF: *Recall* del conjunto de pruebas

Figura 5.46: NMF: *Recall* del conjunto de entrenamiento y pruebas.



(a) NMF: *F-Measure* del conjunto de entrenamiento



(b) NMF: *F-Measure* del conjunto de pruebas

Figura 5.47: NMF: *F-Measure* del conjunto de entrenamiento y pruebas.

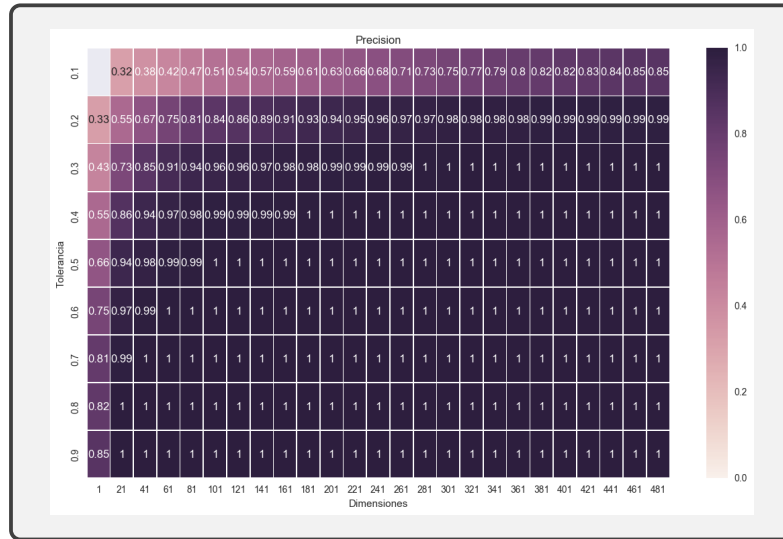
Respecto a la *Precision* observada en los gráficos 5.45 se observa que tanto el conjunto de entrenamiento como el conjunto de pruebas presenta valores muy altos para la mayoría de las configuraciones probadas. También es posible observar una relación entre las variables tolerancia y dimensiones. A medida que tolerancia toma valores más grandes es posible observar que no se necesita tantas dimensiones para obtener una buena *Precision*, esto aplica tanto para el conjunto de entrenamiento como para el conjunto de pruebas. Sin embargo, si se realiza lo contrario, es decir, se disminuye la tolerancia pero aumenta la cantidad de dimensiones, no se verá el mismo comportamiento, ya que se logran menores precisiones con tolerancia baja y dimensiones altas. Considerando valores intermedios se observa que a medida que aumenta la tolerancia hasta 0,3 el comportamiento del algoritmo mejora bastante tomando en cuenta la métrica *Precision*.

En el caso de la métrica *Recall* se observa un comportamiento distinto que en *Precision*. Si bien el conjunto de entrenamiento y pruebas poseen la misma tendencia de aumento de esta métrica, ambos difieren en el valor de ésta. Se evidencia que en el conjunto de entrenamiento posee mejores valores de *Recall*, llegando así al máximo de 0,98 con una configuración de tolerancia 0,1 y 421 dimensiones, en cambio, en el conjunto de pruebas el máximo valor es 0,69 con la misma configuración que en el conjunto de entrenamiento. Respecto al comportamiento de las variables se observa que al tener una tolerancia baja, la calidad del algoritmo aumenta considerando la métrica *Recall*, esto sucede para ambos conjuntos de datos. En el caso que se aumente la tolerancia y a su vez las dimensiones, no se tendrá tan buen resultado como con baja tolerancia.

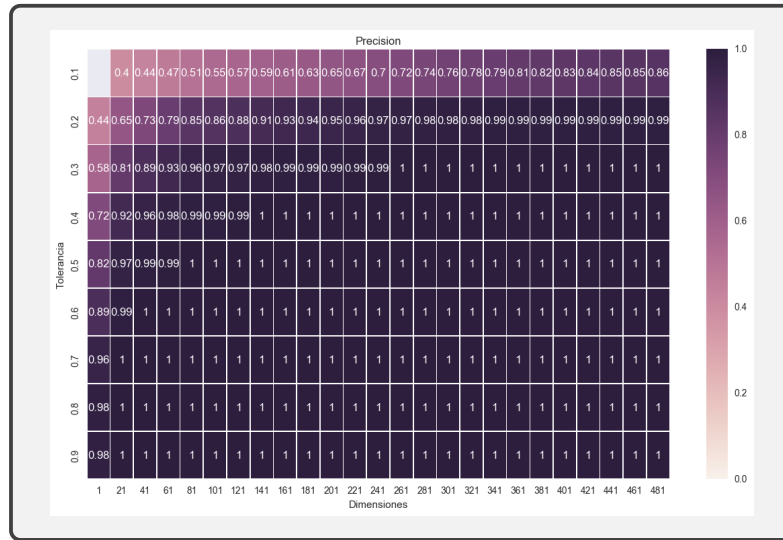
Para *F-Measure* al igual que en las 2 anteriores, se presenta el mismo comportamiento de la métrica tanto para el conjunto de entrenamiento como para el conjunto de pruebas. Esto se debe a que *F-Measure* es la media armónica entre *Precision* y *Recall* como se mencionó en la sección 2.5.3. El mejor valor para el conjunto de entrenamiento se encontró con 441 dimensiones y una tolerancia de 0,2, el valor máximo otorgado fue de 0,95. Para el conjunto de pruebas el valor máximo encontrado fue 0,78 con 461 dimensiones y una tolerancia de 0,2, este será la configuración para analizar las distintas recomendaciones realizadas por el algoritmo NMF.

5.3.2. SVD

Otra técnica que se utilizó fue SVD. Al igual que en el algoritmo NMF también se utilizarán las métricas *Precision*, *Recall* y *F-Measure*, en 2 conjuntos de datos distintos: Entrenamiento y pruebas.

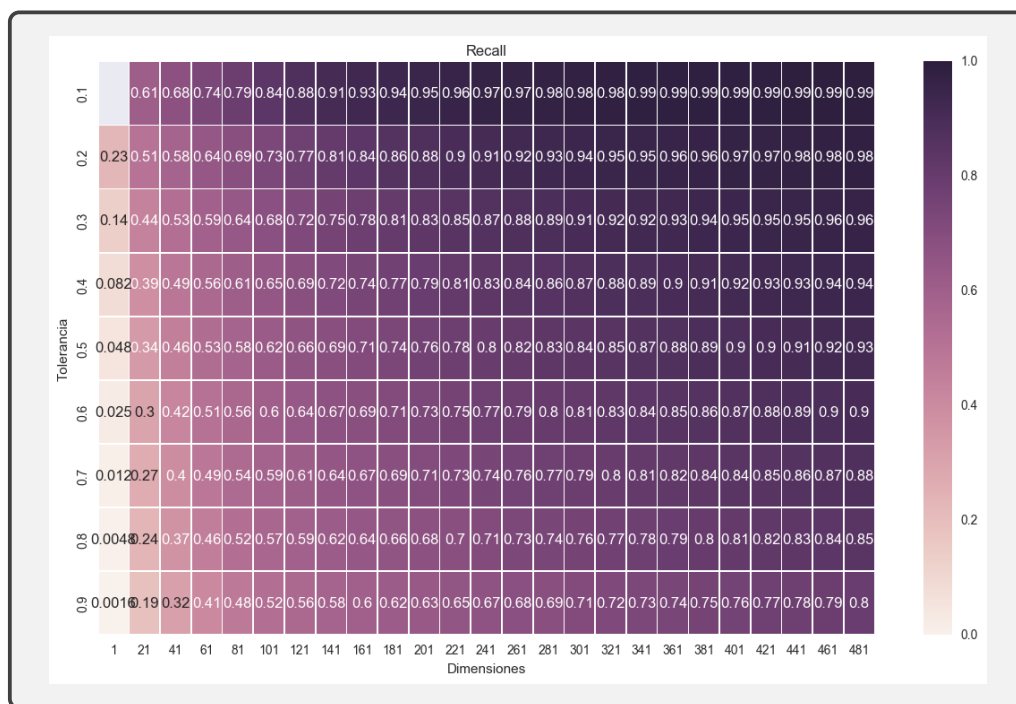


(a) SVD: *Precision* del conjunto de entrenamiento

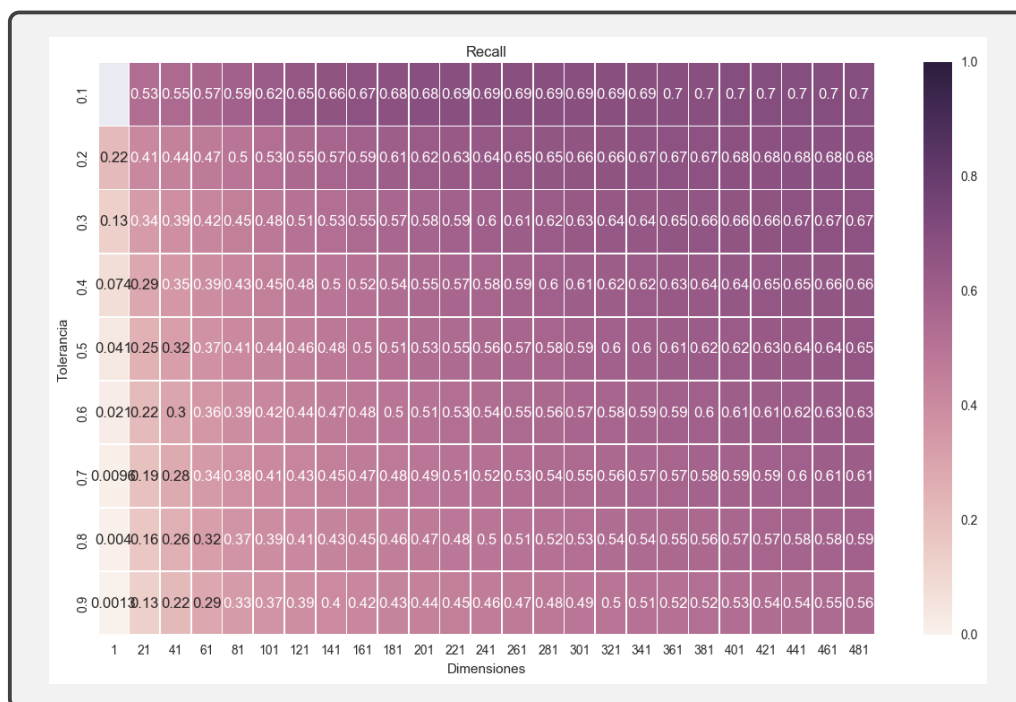


(b) SVD: *Precision* del conjunto de pruebas

Figura 5.48: SVD: *Precision* del conjunto de entrenamiento y pruebas.

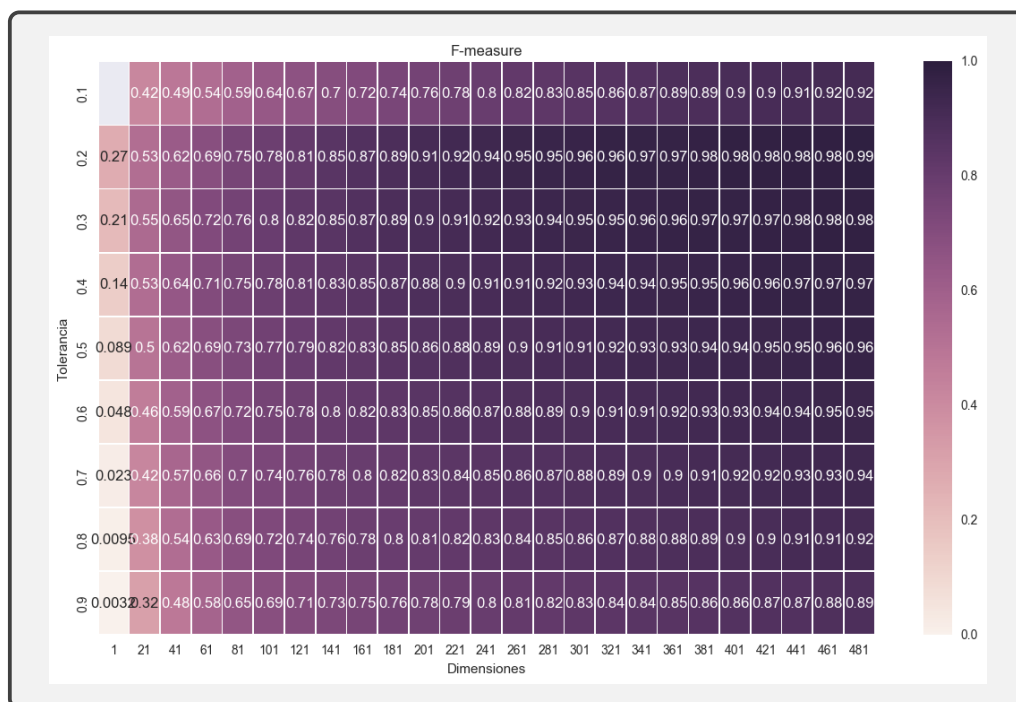


(a) SVD: *Recall* del conjunto de entrenamiento

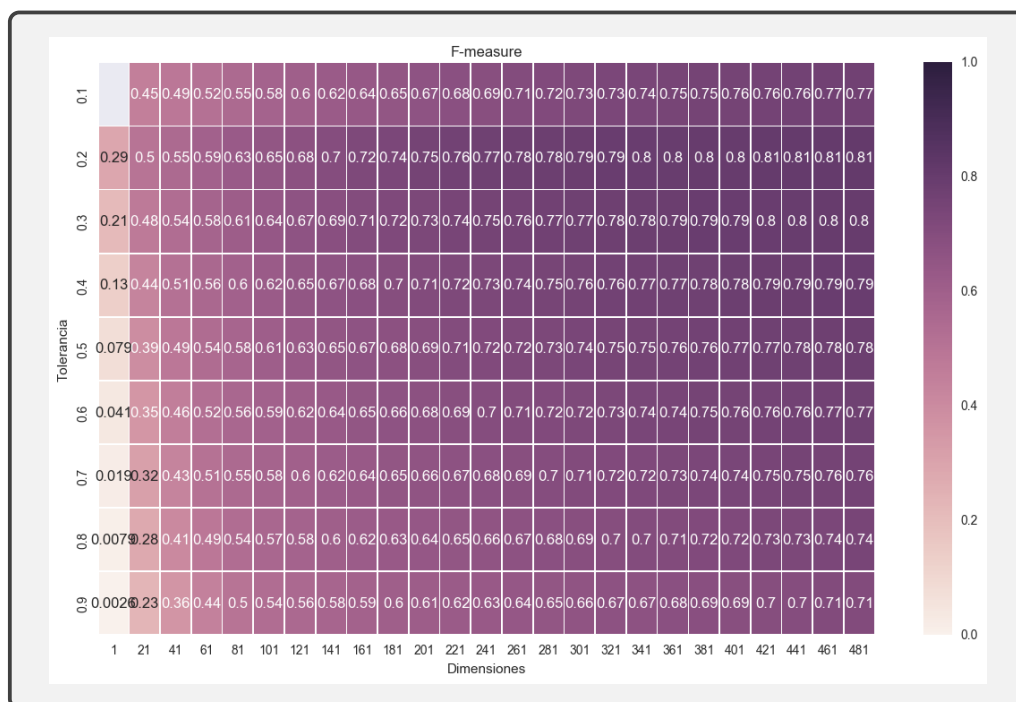


(b) SVD: *Recall* del conjunto de pruebas

Figura 5.49: SVD: *Recall* del conjunto de entrenamiento y pruebas.



(a) SVD: *F-Measure* del conjunto de entrenamiento



(b) SVD: *F-Measure* del conjunto de pruebas

Figura 5.50: SVD: *F-Measure* del conjunto de entrenamiento y *test*.

Observando los gráficos de la figura 5.48 correspondientes a la *Precision*, se evidencia que tanto el conjunto de entrenamiento como el de pruebas poseen el mismo comportamiento a medida que aumentan las variables involucradas. Se observa también que la variable que más afecta al comportamiento del algoritmo es la tolerancia, ya que al pasar el valor 0,4, los valores de la *Precision* son altas sin importar cuantas dimensiones estén seleccionadas.

Respecto a los gráficos de *Recall* 5.49, se presenta exactamente el mismo comportamiento presentado en NMF. El comportamiento de ambos conjuntos de datos son iguales con respecto al comportamiento de las variables, pero estos difieren en los valores de las métricas. Para el conjunto de entrenamiento se observa que el valor más alto de *Recall* es 0,99 y se presenta en tolerancia 0,1 y 361 dimensiones. Para el caso del conjunto de *test* el máximo valor es 0,7 y se presenta con una tolerancia de 0,7 y 361 dimensiones.

Por último, en la métrica *F-Measure* como era de esperar también se presenta el mismo comportamiento para ambos conjuntos de datos, ya que las métricas *Precision* y *Recall* también se comportaban de igual manera para estos conjuntos. El mejor valor para el conjunto de entrenamiento se encontró con 481 dimensiones y una tolerancia de 0,2. Para el caso del conjunto de pruebas se tiene que el mejor valor es 0,81 y se encuentra con una tolerancia de 0,2 y 421 dimensiones.

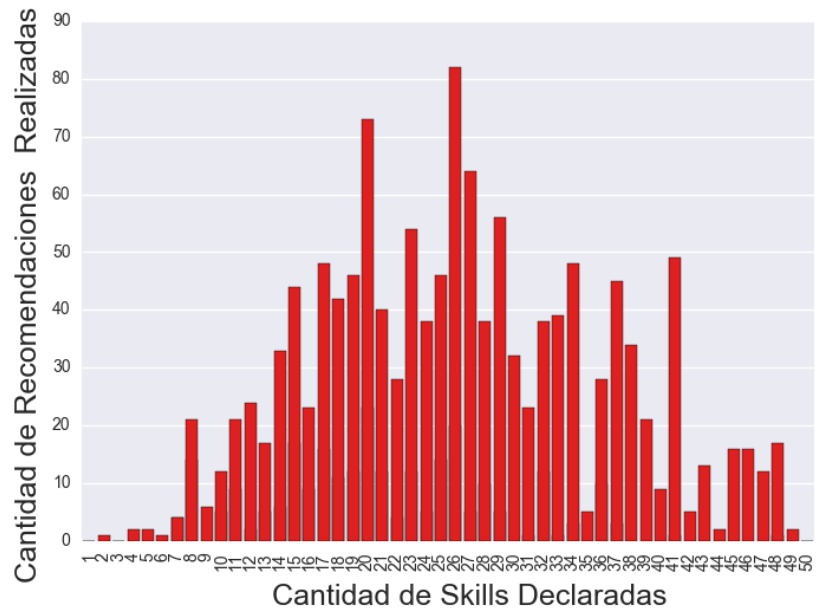
5.3.3. Recomendador de *skills*: Demostraciones

Considerando las mejores configuraciones encontradas en los algoritmos NMF y SVD, referentes a la métrica *F-Measure* en los conjuntos de pruebas se tiene lo siguiente:

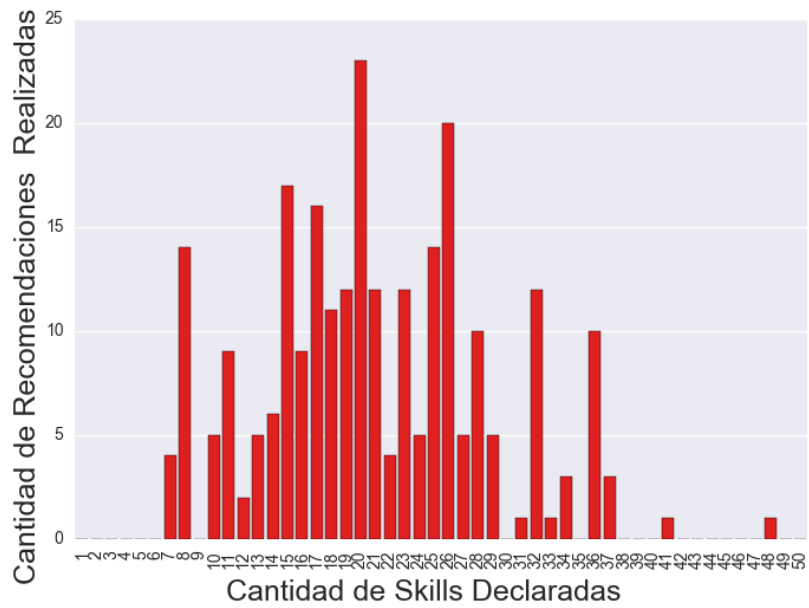
- NMF: La mejor configuración es 461 dimensiones y una tolerancia de 0,2 con un valor *F-Measure* de 0,78 para el conjunto de pruebas.
- SVD: La mejor configuración es 421 dimensiones y una tolerancia de 0,2 con un valor de *F-Measure* de 0,81 para el conjunto de pruebas.

Luego de observar las mejores configuraciones de cada algoritmo se procede a observar cuántas recomendaciones realizó el algoritmo en función de la cantidad de *skills* declaradas

por los usuarios.



(a) Distribución NMF



(b) Distribución SVD

Figura 5.51: Distribución de *skills* recomendadas en función de *skills* declaradas.

En los gráficos se observa una tendencia de los algoritmos de recomendar más *skills* a cierto

rango de usuarios que se caracterizan por declarar cierta cantidad de *skills*. En el caso de NMF, el rango de mayor recomendaciones se encuentra el rango de declaraciones 17 y 34. Para el caso de SVD este rango se encuentra entre 14 y 29. Otro punto interesante es la cantidad de declaraciones realizadas por cada algoritmo, NMF presenta un número mucho más grande de recomendaciones a comparación de SVD. Esto se puede apreciar en los rangos de máximas declaraciones, en el caso de NMF el máximo alcanza las 82 *skills* declaradas para el rango de usuarios que declaró 27 *skills*. En cambio, el máximo de SVD se encuentra en 23 para el rango de usuarios que declaró 20 *skills*.

A continuación se presentan algunos ejemplos de recomendaciones realizadas tanto por el algoritmo NMF como SVD a distintos usuarios del estudio:

Usuario 10 *Skills* declaradas:

- | | |
|--------------------------|------------------------------|
| ■ Curriculum Development | ■ Classroom |
| ■ Tecnología Educativa | ■ Classroom Management |
| ■ E-Learning | ■ Educational Technology |
| ■ Moodle | ■ Differentiated Instruction |
| ■ Lesson Planning | ■ Elementary Education |
| ■ Tutoring | |

***Skills* recomendadas con NMF:**

- Curriculum Design
- Editing
- Educational Leadership
- Instructional Design

***Skills* recomendadas con SVD:**

- Curriculum Design
- Educational Leadership

Usuario 217 *Skills* declaradas:

- | | |
|-------------------------|----------------------|
| ■ Microsoft Office | ■ Consultoría |
| ■ Gestión de proyectos | ■ Microsoft Word |
| ■ Microsoft PowerPoint | ■ Trabajo en equipo |
| ■ Microsoft Excel | ■ Spanish |
| ■ Cambio organizacional | ■ Marketing Strategy |

- Planificación estratégica
- Liderazgo de equipos
- Español
- Inteligencia empresarial
- Change Management
- SAP
- Management
- Estrategia empresarial
- Negociación
- Administración y dirección de empresas
- E-Learning
- Business Strategy
- Coaching

Skills recomendadas con NMF:

- Moodle
- Gestión del talento humano

Usuario 238 *Skills* declaradas:

- Gestión de proyectos
- Microsoft SQL Server
- Liderazgo de equipos
- Inteligencia empresarial
- Business Analysis
- Requirements Analysis
- Linux
- Software Development
- MySQL
- jQuery
- Java
- JavaScript

- Análisis de negocio
- Marketing
- Entrepreneurship
- Educación superior
- Teaching
- Recursos humanos
- Docencia
- Desarrollo organizacional
- Espíritu empresarial
- Corporate Communications
- Planning
- Public Administration
- Persona

Skills recomendadas con SVD:

- Gestión del talento humano

- Cloud Computing
- MongoDB
- Data Analysis
- NoSQL
- C (programación)
- Distributed Systems
- R
- Amazon Web Services (AWS)
- User Interface Design
- Machine Learning
- User Experience
- Hadoop

- Data Warehousing
- Data Mining
- Python
- Bases de datos
- HTML
- CSS

***Skills* recomendadas con NMF:**

- Algorithms
- Mathematical Modeling

Usuario 333 *Skills* declaradas:

- Microsoft Office
- Gestión de proyectos
- SQL
- Microsoft Excel
- Microsoft SQL Server
- Inglés
- Liderazgo de equipos
- Inteligencia empresarial
- Linux
- Oracle
- Software Development
- MySQL

***Skills* recomendadas con NMF:** No existen recomendaciones.

Usuario 1251 *Skills* declaradas:

- Tableau
- Mahout
- Spark
- Managing Database
- SpringSource
- Parallel Algorithms
- Apache Pig
- Hive
- HBase

***Skills* recomendadas con SVD:**

No existen recomendaciones.

- PHP
- Microsoft Project
- QlikView
- PL/SQL
- LUM
- Data Mining
- Pentaho
- Oracle Forms
- Extracción de datos
- Facebook API
- Oracle Discoverer

***Skills* recomendadas con SVD:**

- Information Management

- Planeamiento de proyectos
- Liderazgo de equipos
- Agile Methodologies
- Software Development

***Skills* recomendadas con NMF:**

- Outlook
- Game Design
- Game Programming

- Game Development
- Análisis de negocio
- Budgets
- Video Games

***Skills* recomendadas con SVD:**

- Game Design
- Game Programming

Usuario 1342 *Skills* declaradas:

- ERP
- SAP
- SAP R/3
- SAP ERP
- SD

***Skills* recomendadas con NMF:**

- SAP Implementation

- SAP FI
- ABAP
- SAP Netweaver
- IDOC
- BAPI

***Skills* recomendadas con SVD:**

- SAPScript
- SAP Implementation

Se puede observar en las recomendaciones anteriores que las *skills* recomendadas por los algoritmos NMF y SVD tienen bastante relación con las *skills* que ha declarado el usuario previamente. Además, se puede observar que en ocasiones, un algoritmo recomienda una *skill* que el otro algoritmo no, pero que si tiene relación con las *skills* declaradas por el usuario. Por último se observa que existen usuarios que no son recomendados por un algoritmo pero si por el otro, y las recomendaciones realizadas guardan relación con las *skills* declaradas por el usuario.

Finalmente, observando los gráficos de distribución en la figura 5.3.3 y de la métrica *F-Measure* de las figuras 5.50 5.47 se elegirá como mejor recomendador el algoritmo NMF, ya que si bien posee menor valor máximo de *F-Measure* éste difiere solamente en 0,03 pero la técnica NMF realiza muchas más recomendaciones a comparación de SVD.

Capítulo 6

Conclusiones

Por todo lo presentado en este trabajo, surgen distintas conclusiones de los resultados obtenidos en relación a los diversos objetivos presentados.

6.1. Recolección semi-automática de datos

Lo que se puede obtener de este estudio con relación a los procesos de recolección de datos, es que estos deben ser un proceso estudiado, riguroso y planificado de buena manera. Si esto no se realiza de manera metódica, el proceso recolectará datos que no guardan relación con el estudio, causando procesos que se pueden evitar tales como la limpieza de registros, eliminación de datos mal etiquetados y la reestructuración de estos datos recogidos. En esta memoria el proceso de recolección de datos fue medianamente ejecutado ya que de todas maneras hubo que realizar los procesos de limpieza descritos anteriormente, pese a esto, el objetivo de recolección de datos se cumplió de buena manera ya que a partir de éstos fue posible encontrar resultados satisfactorios considerando los demás objetivos.

6.2. Análisis Exploratorio

Si bien este no es un objetivo como tal, también fue parte importante del estudio ya que permitió observar el cómo se comportan los datos recolectados de manera más fácil e intuitiva. Como conclusión respecto a esta parte de la memoria se deduce que para este tipo de experimentos exploratorios, esta es una parte obligatoria, ya que sin un análisis previo no se sabe absolutamente nada sobre lo que se está trabajando.

6.3. Agrupación de *skills*

Este es fue uno de los objetivos más importantes de la memoria y trataba de descubrir si de alguna manera era posible encontrar una agrupación implícita ocupando distintas técnicas *clustering* descritas en la literatura. Como se puede evidenciar en la sección 5.1 efectivamente fue posible encontrar estas agrupaciones y a su vez describirlas de buena manera. Lo que se obtuvo de este objetivo es que no todas las técnicas dan los mismos resultados, ya que el funcionamiento de éstas no compatibiliza de buena manera con la naturaleza de los datos.

En este caso *K-Means* y *DBSCAN* no ofrecieron buenos resultados ya que ambas técnicas generan *clusters* cuyo funcionamiento se basa en crear agrupaciones del tipo no difusas, es decir, que los datos pertenecen solo a un *cluster* generado. En este caso, las palabras o *skills* no poseen esa naturaleza, por ejemplo, la *skill* PHP puede ser considerada dentro del grupo **Programación** como en el grupo de **Desarrollo WEB**, sin ser excluyente.

Para el caso del algoritmo *Ward*, se evidencia que este funcionó a la perfección tanto en la agrupación de *skills*, como en la creación de una Ontología de *skills*. Pese a esto, las métricas Coeficiente de *Silhouette* y el Promedio *Coherence* que se utilizaron para observar el comportamiento del algoritmo indicaban que los grupos generados no tenían coherencia alguna, por lo que se puede concluir que las métricas utilizadas para observar el comportamiento de *Ward* no era necesariamente la indicada.

Finalmente, se puede decir que el desempeño de los *Topic Models* fue impecable ya que éstos encontraron la mayoría de las agrupaciones que se presentaron en este trabajo. Cabe destacar

que estos algoritmos se basan en el hecho de que una *skill* o palabra, puede pertenecer a uno o más tópicos según un grado de importancia, por lo que al contrario de los algoritmos no difusos, estos respetan la naturaleza de las palabras.

6.4. Agrupación de usuarios

Este objetivo es muy similar al primero, tanto así que en forma de código se ocuparon cosas muy similares. En esta ocasión efectivamente fue posible encontrar agrupaciones de usuarios, pero surgieron un par de dificultades que se discutirán a continuación. Los algoritmos que se utilizaron efectivamente encontraron agrupaciones basándose en las distintas métricas propuestas, salvo el algoritmo *DBSCAN* que por el comportamiento de éste fue incapaz de encontrar grupos que tuvieran buena distribución de usuarios para ser analizados.

El problema que surgió en este objetivo fue que las *skills* declaradas por los usuarios no eran suficiente para describir todos los grupos, esto no quiere decir que las agrupaciones no tengan sentido, sino más bien que la variable *skills* declaradas no fue la adecuada para realizar este objetivo. Los grupos que no se pudieron describir de buena manera considerando las *skills* declaradas pertenecían a los encontrados por técnicas *Topic Models*.

6.5. Predictor que permita inferir *skills* no declaradas en los perfiles

El último objetivo propuesto en la memoria fue sobre la realización de un predictor que permita inferir distintas *skills* no declaradas en los perfiles que están en el estudio. Este objetivo fue cumplido a cabalidad utilizando 2 algoritmos correspondientes a filtrado colaborativo. En esta ocasión se aprecia que NMF y SVD se comportan de manera muy similar con respecto a las métricas que se utilizaron para medir los resultados, lo que complicó la elección de qué técnica definir como la mejor para realizar el predictor. Considerando lo anterior se tuvieron que realizar diversos análisis tanto de la cantidad de recomendaciones realizadas

como la calidad de éstas. Finalmente el algoritmo que resultó mejor para realizar inferencias de *skills* en los perfiles de *LinkedIn* resultó ser el algoritmo NMF considerando la cantidad de recomendaciones realizadas.

6.6. Conclusiones Generales

Respecto al comportamiento de los algoritmos se puede concluir que éstos dependen netamente de la naturaleza de los datos con los cuales se está trabajando. En este caso, uno de los algoritmos no difusos, se comportó muy distinto en 2 objetivos muy similares, esto se puede explicar debido a que la naturaleza de las *skills* o palabras, es muy distinta a la de los usuarios o documentos.

El tipo de análisis que acá se realizó fue en gran parte de forma manual al momento de hacer la descripción de cada grupo generado. Considerando esto es que existió una limitación con respecto a la cantidad de grupos que fueron analizados, por lo que puede existir el caso que algoritmos que tuvieron mal desempeño con pocos *clusters* funcionen realmente bien considerando una buena cantidad de *clusters* a generar y analizar.

Por otro lado, respecto a las métricas, estas no necesariamente definen tajantemente si un algoritmo tiene mal desempeño o no, si no más bien, significa que lo que el algoritmo considera óptimo, no necesariamente sea lo que se requiere en el estudio que se realizará. En este caso la métrica del Coeficiente *Silhouette* si bien daba buenos valores, los análisis de los resultados eran paupérrimos con respecto a las agrupaciones que se requerían.

Finalmente se debe mencionar que los trabajos de investigación o exploratorios llevan un gran trabajo por detrás, ya que al no haber un punto de comparación con respecto a lo que se está realizado, es difícil comprobar si el trabajo que se está desarrollando tendrá resultados satisfactorios. Sin embargo, como se puede apreciar a lo largo del trabajo se puede concluir tajantemente que los objetivos propuestos en esta memoria fueron realizados con éxito.

Trabajo A Futuro

Luego de realizar los distintos objetivos referentes a este estudio, se observó que queda mucho trabajo por delante con respecto al estudio de los usuarios y las *skills* que declaran éstos dentro de la plataforma *LinkedIn*.

En este caso solo se trabajó con una de las tantas variables que se pudieron rescatar dentro de los 2441 perfiles informáticos de distintas universidades. Existe la posibilidad de procesar de manera similar a como se hizo con las *skills* las distintas variables que quedaron sin utilizar en el *dataset*, como por ejemplo, el cargo y la empresa actual del usuario, el sector industrial en el cual se desarrolla, la trayectoria laboral del usuario o las distintas recomendaciones que le han realizado a éste.

Otro trabajo pendiente es proponer nuevas técnicas que sirvan para solucionar los distintos problemas de significado de *skills*, tales como que una *skill* posea 2 significados distintos, que 2 *skills* posean el mismo significado tomando siempre en cuenta la diferencia de idiomas que pueda existir entre éstas.

Respecto a los objetivos de agrupación de *skills* y usuarios, en estos se utilizaron pocas técnicas de *clustering* a comparación con las que existen en la literatura. Un trabajo pendiente que queda en esta memoria es a los mismos datos utilizados acá, aplicar técnicas de *clustering* más complejas tales como *Biclustering* o *Fuzzy Clustering*.

Siguiendo con el trabajo pendiente en los objetivos de agrupación, también se propone considerar otras métricas intrínsecas que ayuden a argumentar de mejor manera la toma de decisiones con respecto a los parámetros de los distintos algoritmos.

Para solo el caso de agrupamiento de usuarios y en conjunto al procesamiento de otras variables, existe la posibilidad de complementar este estudio utilizando otras variables para describir los grupos resultantes al aplicar las técnicas *clustering*. Recordar que en este objetivo si bien se encontraron buenas agrupaciones considerando las distintas métricas, éstas no podían ser descritas fácilmente utilizando el criterio de declaración de *skills*.

En el caso del objetivo relacionado a la creación de un recomendador, se podría elegir otras técnicas relacionadas a los sistemas de recomendación, tales como, NMF y SVD regularizado o técnicas relacionadas al agrupamiento, tales como *Biclustering*.

Finalmente como gran trabajo a futuro sería repetir este interesante estudio utilizando no solo perfiles relacionados al área de informática, sino también de otras áreas de ingeniería como eléctrica, química o construcción civil. Es más, este estudio podría generalizarse no solo a las áreas de ingeniería sino a cualquier disciplina.

Bibliografía

- [1] Mathieu Bastian, Matthew Hayes, William Vaughan, Sam Shah, Peter Skomoroch, Hyungjin Kim, Sal Uryasev, and Christopher Lloyd. LinkedIn skills: large-scale topic extraction and inference. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 1–8. ACM, 2014.
- [2] Tom Case, Adrian Gardiner, Paige Rutner, and John Dyer. A linkedin analysis of career paths of information systems alumni. *Journal of the Southern Association for Information Systems*, 1(1), 2013.
- [3] Jonathan Chang, Jordan Boyd-Graber, Sean Gerrish, Chong Wang, and David M. Blei. Reading tea leaves: How humans interpret topic models. In *Nips*, pages 1–9, 2009.
- [4] Rajnish Dashora, Harsh Bajaj, Akshat Dube, et al. Parallel algorithm for the chameleon clustering algorithm using dynamic modeling. *International Journal of Computer Applications*, 79(8):11–17, 2013.
- [5] Andrew Y. Ng, David M. Blei, and Michael I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.
- [6] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.
- [7] Julio Hurtado Gonzalez. Memoria: Geo-referenciación de sentimientos en twitter. 2015.
- [8] Micheline Kamber, Jiawei Han, and Jian Pei. *Data mining: concepts and techniques*. Elsevier, 2011.
- [9] Mingxuan Sun, Joonseok Lee, and Guy Lebanon. A comparative study of collaborative filtering algorithms. *arXiv preprint arXiv:1205.3193*, 2012.
- [10] Rahele Kafieh and Alireza Mehridehnavi. a comprehensive comparison of different clustering methods for reliability analysis of microarray data. *Journal of medical signals and sensors*, 3(1):22–30, 2013.

- [11] Ana Fernández Vilas Kais Dai, Celia González Nespereira and Rebeca P.Díaz Redondo. Scraping and clustering techniques for the characterization of linkedin profiles. *arXiv preprint arXiv:1505.00989*, 2015.
- [12] George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.
- [13] David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 262–272. Association for Computational Linguistics, 2011.
- [14] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011.
- [15] Fionn Murtagh and Pierre Legendre. Ward’s hierarchical clustering method: clustering criterion and agglomerative algorithm. *arXiv preprint arXiv:1111.6285*, pages 58–82, 2011.
- [16] Andrew Ng. The problem of overfitting. <https://es.coursera.org/learn/machine-learning/lecture/ACpTQ/the-problem-of-overfitting/>. Accessed: 2017-04-03.
- [17] Hebert Pérez-Rosés, Francesc Sebé, and Josep Maria Ribó. Endorsement deduction and ranking in social networks. *Computer Communications*, 73:200–210, 2016.
- [18] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, 2003.
- [19] F James Rohlf. Adaptive hierarchical clustering schemes. *Systematic Biology*, 19(1):58–82, 1970.
- [20] F James Rohlf and David R Fisher. Tests for hierarchical structure in random data sets. *Systematic Biology*, 17(4):407–412, 1968.
- [21] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.
- [22] Sebastian Rude. An overview of gradient descent optimization algorithms. <http://sebastianruder.com/optimizing-gradient-descent/>, 2016. Accessed: 2017-04-03.
- [23] Nicolás Torres. Memoria: Sistemas de recomendación basados en métodos de filtrado colaborativo. 2015.
- [24] Peter Wiemer-Hastings, K Wiemer-Hastings, and A Graesser. Latent semantic analysis. In *Proceedings of the 16th international joint conference on Artificial intelligence*, pages 1–14. Citeseer, 2004.

- [25] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.
- [26] Albert Au Yeung. Matrix factorization: A simple tutorial and implementation in python. <http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/>, 2010. Accessed: 2016-09-22.